An evolutionary algorithm approach to poetry generation

Hisar Maruli Manurung

Doctor of Philosophy
Institute for Communicating and Collaborative Systems
School of Informatics
University of Edinburgh

2003

Abstract

Poetry is a unique artifact of the human language faculty, with its defining feature being a strong unity between content and form. Contrary to the opinion that the automatic generation of poetry is a relatively easy task, we argue that it is in fact an extremely difficult task that requires intelligence, world and linguistic knowledge, and creativity.

We propose a model of poetry generation as a state space search problem, where a goal state is a text that satisfies the three properties of **meaningfulness**, **grammaticality**, and **poeticness**. We argue that almost all existing work on poetry generation only properly addresses a subset of these properties.

In designing a computational approach for solving this problem, we draw upon the wealth of work in natural language generation (NLG). Although the emphasis of NLG research is on the generation of informative texts, recent work has highlighted the need for more flexible models which can be cast as one end of a spectrum of search sophistication, where the opposing end is the deterministically goal-directed planning of traditional NLG. We propose satisfying the properties of poetry through the application to NLG of evolutionary algorithms (EAs), a well-studied heuristic search method.

McGonagall is our implemented instance of this approach. We use a linguistic representation based on Lexicalized Tree Adjoining Grammar (LTAG) that we argue is appropriate for EA-based NLG. Several genetic operators are implemented, ranging from baseline operators based on LTAG syntactic operations to heuristic semantic goal-directed operators. Two evaluation functions are implemented: one that measures the isomorphism between a solution's stress pattern and a target metre using the edit distance algorithm, and one that measures the isomorphism between a solution's propositional semantics and a target semantics using structural similarity metrics.

We conducted an empirical study using McGonagall to test the validity of employing EAs in solving the search problem, and to test whether our evaluation functions adequately capture the notions of semantic and metrical faithfulness. We conclude that our use of EAs offers an innovative approach to flexible NLG, as demonstrated by its successful application to the poetry generation task.

Acknowledgements

I am hugely indebted to my supervisors, Henry Thompson and Graeme Ritchie, for their invaluable guidance and comments, constant support, and for their tolerance towards my consistently creative interpretation of the word ‘deadline’.

I also owe many thanks to the various people I had fruitful discussions with concerning my work, whether remotely or in person, among others Paul Bailey, Mark Dras, Mick O’Donnell, Hasan Kamal, Nikiforos Karamanis, and Matthew Stone; and to those who provided me with a wealth of advice and support in completing my thesis, among others Ben Curry, Jacques Fleuriot, and Thomas Segler. I am especially indebted to my great friend and colleague, Ewen Maclean, who ended his thesis on the same day as me.

I am very grateful to my thesis examiners, Chris Mellish and Richard Power, whose comments and suggestions helped me to improve this thesis.

I am also very grateful to my numerous friends and colleagues here in Edinburgh, both at the South Bridge and Buccleuch Place schools, who have provided me with great friendship, a stimulating research environment, and put up with my temperament during the last stages of my thesis-writing; also to all the extremely kind folks at the Scottish Language Dictionary; and members of the Indonesian community in Edinburgh, particularly David and Yufrita Skyner.

Finally, my utmost gratitude to my family and to my beloved Anna, who showed more patience and understanding than I probably deserved.

This work was possible due to funding from the World Bank QUE Project, Faculty of Computer Science, Universitas Indonesia.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Hisar Maruli Manurung*)

To Anns.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis describes a model of computer generation of poetry as a state space search problem where a goal state is a text that satisfies the properties of **grammaticality**, **meaningfulness**, and **poeticness**. It advocates the solving of this problem using evolutionary algorithms, a well known stochastic search technique. This model was implemented in our program, MCGONAGALL, and an empirical study using this program was conducted.

In this chapter, we will first provide the context and motivation for this research, and discuss the methodological issues that underlie our work. We then present a very brief overview of the work contained in this thesis, along with the contributions that this thesis makes. We then present an outline for the remainder of this thesis.

1.1 Motivation for this research

There are two different factors that motivate the research work presented in this thesis. The first is the more philosophically-oriented aim of showing that computer programs can indeed autonomously generate texts that are considered poetic, as this can be seen as an indication of general intelligence. We shall call this the **AI motivation**. The second is the more technically-oriented aim of exploring novel architectures that provide the degree of flexibility which recent work in natural language generation calls for. We shall call this the **NLG motivation**. We will now discuss these two motivations in more detail:

1. The AI motivation

From as early as 1962, the generation of poetry by computers, whether autonomous or supervised, has been a subject of much interest (Bailey, 1974). The reason for this interest can be attributed to the fact that certain tasks are often cited as being key benchmarks of general human intelligence. Minsky (1963) claims that a suitable goal for AI research is to get a computer to do "… a task which, if done by a human, requires intelligence to perform". As remarked by Binsted (1996), these benchmarks are changeable: before the advent of systems such as Deep Blue, people considered the ability of programs to beat chess grandmasters as such a benchmark for AI. Now, more artistic-based tasks, such as the autonomous generation of music, visual art, stories, and poetry are often proposed to be the defining benchmark.

Unfortunately, given the notoriously subjective nature of appreciating and critiquing poetry, such poetry generation systems have not been subjected to serious scientific treatment until recent works such as Gervás (2002).

Although this motivation does indeed serve as the main inspiration for our work, it must be stated clearly that the passing of this benchmark is not something that we have set out to achieve in this thesis. We believe that the generation of 'genuine' poetry, i.e. texts which members of the general public can observe and with a high degree of certainty state are indeed poetry, is not the goal of this study. We believe that this is far beyond the scope of a single thesis, and that research work in this area is in far too early a stage to be achieving such ambitions. Instead, our aim is to provide a baseline model for the poetry generation task which stands up to the rigours of scientific research, and which can serve as a common framework for future work in this field.

2. The NLG motivation

   As will become evident throughout this thesis, our research work into poetry generation is heavily influenced by the discipline of natural language generation (NLG), the subfield of artificial intelligence that focuses on the production of understandable texts in a human language based on non-linguistic communicative goals (see Chapter 3). Since poetry is a specific genre of human language, our task is therefore a highly specialized instance of NLG.

   In recent years, NLG researchers have called for more flexible approaches to the generation task. In particular, the rigidly architectured conventional approach to NLG is unable to account for situations where interdependent linguistic decisions are required for the generation of fluent and natural-sounding texts, or where the communicative goal is vague (see Section 3.2). As poetry is an extreme case of natural language which exacerbates these problems, it is a useful testing domain for ideas and techniques that attempt to solve these problems, and it is our hope that the architecture and techniques introduced in this thesis may be applicable to other genres of text as well.

1.2 Methodological issues

As mentioned above, research work into the generation of poetry suffers from the inherently subjective nature of the evaluation and critiquing of art. As Binsted (1996) remarks, it is difficult, if not impossible, to disprove the claim “this is art”.

For a model to be scientific, it must be **falsifiable**. That is, it must make certain predictions about the phenomena it is intended to describe, to which one can devise a possible experiment to prove these predictions wrong. A falsifiable model of poetry makes predictions about whether a given text is a poem or not, and it should be possible to test whether this prediction is accurate through independent, objective evaluation.

Thus, one should be able to use a falsifiable model of poetry, and poetry generation, to test whether the output of a given system, which claims to generate poetry, is indeed poetry or not.

In our research, we choose to steer clear of the difficult questions of creative and artistic merit of poetry, whether generated by humans or computers, and of the abstract and subjective aspects of imagery, metaphor, and figurative language. Instead, we concentrate on the more concretely observable features of poetry, i.e. metre and propositional semantics.

We believe that our restricted definition of poetry presented in Section 2.1.4, which is used throughout our thesis, both captures the essential notions of a subclass of poetry, and crucially, can be used as a basis for falsifiable experiments.

1.3 Overview of the thesis

This thesis describes our efforts to address the issues raised in Section 1.1. Based on our restricted definition of poetry as texts that simultaneously satisfy the properties of grammaticality, meaningfulness, and poeticness (Section 2.1.4), we proposed a model of poetry generation as a state space search, which we solve using evolutionary algorithms, or EAs (Chapter 4).

Three crucial aspects of an EA system were addressed for our specific task: design of representational scheme, evaluation function, and genetic operators. We employ a flat semantic representation and a linguistic representation based on LTAG (Chapter 5). Two evaluation functions were developed: metre similarity (Section 6.3) and semantic similarity (Section 6.4), both of which employ structural alignment algorithms to yield a measure of similarity between a target structure and the appropriate features of a candidate solution. We developed an array of genetic operators which stochastically modify LTAG derivation trees in a nonmonotonic fashion while always ensuring well-formedness (Section 7.2). More semantically motivated operators, akin to tactical NLG systems, were also developed (Section 7.3). All this was implemented in our system, MCGONAGALL.

Finally, we conducted an empirical study using MCGONAGALL to justify our approach of using EAs to solve our problem of poetry generation, and to validate our metre and semantic evaluation functions as measures of conformance to metre and semantic faithfulness (Chapter 8).

1.3.1 Contributions of this thesis

1. This thesis presents a restricted definition of poetry (Section 2.1.4), and a model of poetry generation based on this definition (Section 4.1), which is computational, falsifiable, and implementable. Although it does not account for all the myriad aspects of poetry, we believe it is a useful framework for future research into this subject.

2. This thesis proposes an alternative technique for NLG based on evolutionary algorithms. Similar work has been done before, e.g. Mellish et al. (1998a) and Cheng (2002), but has typically been confined to a very specific subtask of the generation process, e.g. text structuring, aggregation.

3. This thesis provides a computational model of metre similarity (Section 6.3). Although it is based on a well-known existing method of structural alignment, i.e. the minimum edit distance, we believe we have made a significant contribution in explicitly defining a similarity metric for the metre patterns exhibited by human texts.

4. This thesis proposes an LTAG-based representational scheme (Section 5.3), and an array

of genetic operators using this representation (Chapter 7), for the generation of linguistic structures in the nonmonotonic fashion as required by evolutionary algorithms. In particular, these operators allow the stochastic, nonmonotonic incremental generation of texts which guarantee linguistic well-formedness (Section 7.2) and are semantically motivated (Section 7.3). We believe this is a contribution not only for poetry generation, but the NLG research field in general.

5. The implemented system described in this thesis, McGonagall, is a proof-of-concept system that can be employed as a testbed platform for other NLG tasks that require flexibility in achieving surface constraints, or where communicative goals are vague.

1.3.2 What this thesis is not about

As mentioned in Section 1.2, this thesis does not discuss issues of creativity theory and artistic merit. Furthermore, although these are briefly alluded to in Sections 2.1.2 and 4.4, we do not consider the more abstract features of poetry such as metaphor, imagery, and figurative language.

1.4 Chapter outline

The following chapters in this thesis describe our model of poetry generation using evolutionary algorithms, our implementation of this model, McGonagall, and the results of an empirical study carried out using McGonagall.

Chapter 2 discusses the salient features of poetry and presents a restricted definition of poetry for the purpose of this thesis. It also discusses existing poetry generation systems, and whether or not they do indeed generate poetry as previously defined.

Chapter 3 provides an overview of the natural language generation (NLG) literature. In particular, it identifies certain methods that are relevant to our task, and provides a characterisation of our poetry generation task from an NLG viewpoint.

Chapter 4 describes our model of poetry generation as a state space search, and advocates the use of evolutionary algorithms (EAs) as an appropriate mechanism to solve the search problem.

It then presents, on a theoretical level, a description of the representational scheme, evaluation functions, and genetic operators that are appropriate for poetry generation using EAs.

Chapters 5 to 7 describe the representational scheme, evaluation functions, and genetic operators of McGonagall, our implemented instance of EA-based poetry generation presented in Chapter 4.

Chapter 8 reports on an empirical study that we carried out using McGonagall to justify our approach of using EAs to solve our problem of poetry generation, and to validate our metre and semantic evaluation functions as measures of conformance to metre and semantic faithfulness.

Chapter 9 presents a summary of the conclusions that can be drawn from the work presented in this thesis, and suggests possible avenues of further work.

Chapter 2

Poetry and automatic poetry generation

In this chapter we will discuss poetry, the general target domain of our research. We will provide a general definition of poetry by first surveying the literature and describing its defining characteristics, before stating the restricted definition of what we consider to be poetry for the purpose of this thesis.

We will then discuss the process of poetry creation and give a brief survey of existing work on the automatic generation of poetry.

2.1 What is poetry?

2.1.1 First attempt at definition

Providing a concrete definition of poetry that does it justice is very difficult, as poetry varies greatly between genres, each with its own characteristics, such that for any attempted definition, one can probably find a poem that serves as a counterexample for that defintion.

Furthermore, the definition of poetry, like other forms of artistic expression, is highly subjective. The sort of recursive definitions such as *"a poem is a poem because people call it a poem"*, albeit in principle perhaps most accurate, does not provide us with much insight. However, the following definition from the Oxford English Dictionary makes for a good starting point:

> "Composition in verse or metrical language, or in some equivalent patterned arrangement of language; usually also with choice of elevated words and figurative uses, and option of a syntactical order, differing more or less from those of ordinary speech or prose writing."

Additionally, Levin (1962) states that:

> "Poetry is a literary form in which language is used in a concentrated blend of sound and imagery to create an emotional response."

These definitions point to a strong interaction between a poem's **form** and **content**. Essentially, this means that a poem's diction and grammatical construction affects the message that it conveys to the reader over and above its obvious denotative meaning. Various phonetic devices such as rhythm, rhyme, alliteration and assonance work in conjunction with the propositional meaning being conveyed to create the whole effect that a reader experiences and appreciates. Consider the difficulties of poetry translation in comparison to the translation of other documents such as newspaper articles and software user manuals. A straightforward translation of a poem may very well retain the denotational meaning but lose its emotional impact. Levin (1962) goes on to restate this claim of a strong sense of **unity** between concept and form in the following quote:

> "... in poetry the form of the discourse and its meaning are fused into a higher unity ... "form" in fact embraces and penetrates "message" in a way that constitutes a deeper and more substantial meaning than either abstract message or separable ornament."

This contrasts with the commonly-held linguistic notion that a piece of text serves purely as a medium for conveying its semantic content (see Section 3.1).

2.1.2 Characteristics and features

Here we identify various aspects that are considered to be defining features that make up the look and sound of poetry, and set it clearly apart from other types of texts, e.g. prose.

1. Rhythm and metre

   The organization of patterns in the rhythm of natural language, particularly in poetry, is known as **metre**. The rhythm in poetry is more noticeable than in prose because it is more regularly defined (Attridge, 1995, Beeferman, 1996).

Before proceeding any further, however, it is best to clarify what is meant by stress, a term which will appear extensively throughout this section. Stress is a rhythmic property of syllables, functioning as boundary markers for the division of utterances into feet, the fundamental rhythmic unit of speech. Thompson (1980) differentiates between abstract stress and manifest salience, where stress is the potential locus of salience, which is what actually occurs in an utterance. The actual acoustic phenomenon is a combination of length, pitch, and loudness, approximately in that order. For our purposes, we only consider stress as an abstract rhythmic unit, without actually considering these acoustic features. We do not use the term salience, simply because in the study of poetry and literature it has been well established to use the term stress instead.

Sansom (1994) identifies four different kinds of metre:

(a) Strong stress metre

Here the sense of rhythm is achieved through having a certain number of syllables in a line being strongly stressed, or **accented**. Between these strongly stressed syllables there may be many redundant unstressed syllables, but the lines are isochronous, meaning that despite the number of intervening unstressed syllables, the stressed syllables fall at more or less regular intervals. Examples of poetry that use this metre are the Old English poems such as *Beowulf*. Another easily recognizable example is a fragment of the popular children's rhyme, *Three Blind Mice*, in Figure 2.1. Note that the ● at the end of the last line denotes **silent stress**, which is similar to a rest in music.

They	**all** ran	**af**ter the	**far**mer's	**wife**,
She	**cut** off their	**tails** with a	**car**ving	**knife**,
Did you	**e**ver	**see** such a	**thing** in your	**life**,
as	**three**	**blind**	**mice**.	●[1]

Figure 2.1: An example of strong stress metre

This metre is similar to what is known as **sprung verse**, made popular by the poet Gerald Manley Hopkins (Hobsbaum, 1996).

(b) Syllabic metre

This metre is concerned only with the number of syllables that appear in a line.

This is used in languages that do not have a sense of stress, e.g. French poetry.

(c) Quantitative metre

This metre was supposedly used by the ancient Greek and Roman poets, where it is claimed that attention was paid to the actual time needed to pronounce syllables, i.e. in theory, *"bit"*, *"bid"*, and *"bide"* have increasingly longer pronounciation times.

(d) Syllabic stress metre

This is the most common form of metre in English poetry, and can be seen as a combination of the first two types of metre. As well as paying attention to the pattern of stressed and unstressed syllables, lines are divided up into rhythmic units known as feet, which have a pre-defined syllable count.

In syllabic stress metre poetry, there are certain types of metrical feet, defined by the syllable length and where the stressed syllable appears. These feet are based on the classical metrics of Latin and Greek. For example, some of the most common feet are:

- **Trochee**, 2 syllables, the stress falling on the first one, e.g. *incest*
- **Iamb**, 2 syllables, the stress falling on the last one, e.g. *inject*
- **Dactyl**, 3 syllables, the stress falling on the first one, e.g. *terrible*
- **Amphibrach**, 3 syllables, the stress falling on the middle one, e.g. *incumbent*
- **Anapest**, 3 syllables, the stress falling on the last one, e.g. *interrupt*

Certain sequences of feet are very well-known and often-used patterns. For example, a sequence consisting of 5 iambs, also known as iambic pentameter, is one of the most widely-used metres. It is claimed that this is probably because it most closely resembles the rhythm of naturally occuring speech. Figure 2.2 shows a famous example of this metre, the opening stanza from Thomas Grey's *Elegy Written in a Country Churchyard*. The / symbol marks a foot boundary.

Note that the main difference from the poem in Figure 2.1 is that here the number of unstressed syllables is maintained to be equal. To achieve this constrained form, several less conventional grammatical constructions have been used, for example

The	**cur**/few	**tolls** / the	**knell** / of	**par**/ting	**day**,
The	**low**/ing	**herd** / wind	**slow**/ly	**o'er** / the	**lea**,
The	**plough**/man	**home**/ward	**plods** / his	**wea**/ry	**way**,
And	**leaves** / the	**world** / to	**dark**/ness	**and** / to	**me**.

Figure 2.2: An example of syllabic stress metre

the placement of the adverb *homeward* in the third line. Moreover, boundaries of metrical feet do not necessarily coincide with word and constituent boundaries. For instance, the words *curfew* and *parting* are split up into different feet.

In our account of syllabic stress metre, we abandon the concept of feet, and instead concentrate on its underlying features, namely the number of syllables and the placement of stress. By doing this, we conveniently sidestep the huge debates that arise over the division of metre into feet, a problem which is ambiguous as shown in Figure 2.3. This account is advocated by, among others, Attridge (1995).

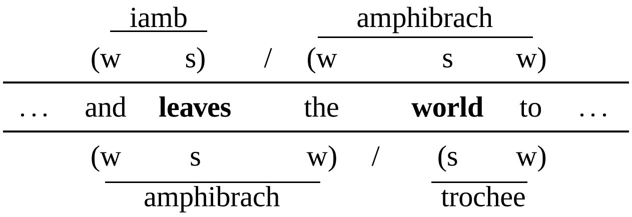


Figure 2.3: Ambiguity in partitioning metre into feet

2. Rhyme and other phonemic patterns

   Aside from rhythmic patterns, poetry also employs many phonemic patterns, both with consonants and with vowels. They commonly appear at the end of lines in a poem, but they also work with words in the middle of a line. The various phonemic patterns are (Roberts, 1986):

   (a) **Rhyme** occurs between two words when their last stressed vowel and the sounds that follow it match, e.g. *biology* and *ideology*. A distinction is often made between **masculine** rhyme, where only one syllable rhymes, e.g. *butt* and *shut*, and **feminine** rhyme, which occurs over two or more syllables, and typically only the first one is stressed, e.g. *lighting* and *fighting*.

   (b) **Alliteration** refers to the repetition of a word's opening consonant sounds, e.g. *butt* and *bend*.

   (c) **Assonance** is when only the vowel sound remains the same, e.g. *butt* and *hull*. This is sometimes referred to as a **half** or **partial** rhyme.

   (d) **Consonance** is when only the closing consonant sound remains the same, e.g. *butt* and *hate*.

3. Figurative language

   Figurative language is the means used by poets to enrich the experience of poetry interpretation through rich wordplay and imagery. Strictly speaking, figurative language is not exclusive to poetry, as it is often used in many other types of texts as well.

   Various methods known as figures of speech, or tropes, are employed. They operate at two different levels, namely the **symbolic** and **rhetorical** level (Quinn, 1982):

   (a) Symbolic level: here analogies, metaphor, and the connotative sense of carefully chosen words are used to convey meaning. One example is **synechdoche**, which is the substitution of a part for a whole, for example when Marlowe writes *"Was this the **face** that launched a thousand ships, and burnt the topless towers of Ilium?"*

   (b) Rhetorical level: here special linguistic structures are used to emphasize certain points or to create an effect unachievable by conventional linguistic means. One example is **hyperbaton**, which is the intended deviation from conventional word order of a single constituent, for example when Robert Frost writes *"I was in my life **alone**"*.

   Quinn (1982) lists a taxonomy of 60 different types of figures of speech, both at the symbolic and the rhetorical level.

2.1.3 The issue of poetic license

The last characteristic mentioned in the previous section, i.e. figurative language, poses an important philosophical issue that we believe must be addressed by research on poetry generation. As mentioned above, figurative language can be found in all genres of text, yet is most often associated with poetry, due to what is referred to as **poetic license**. According to the Oxford

English Dictionary, poetic license is the “deviation from recognized form or rule, indulged in by a writer or artist for the sake of effect”, whereas the Oxford Advanced Learner’s Dictionary states that it is the “freedom to change the normal rules of language when writing verse (e.g. by reversing word order, changing meaning etc.)”. The crucial question is: when appraising the output of a software program that is claimed to generate poetry, to what extent do we hold poetic license accountable for deviations that may be present in the text?

When assessing the output of Masterman’s “computerized haikus” (Masterman, 1971), Boden (1990) claims that:

> “Readers of [poetry] are prepared to do considerable interpretative work. … In general, the more the audience is prepared to contribute in responding to a work of art, the more chance there is that a computer’s performance (or, for that matter, a human artist’s) may be acknowledged as aesthetically valuable … Much of the beauty, might one say, is in the eye of the beholder … Hence poetry, especially poetry of a highly minimalist type (such as haiku), is more tolerant of programmed production than prose is.”

The implication here is that the automatic generation of poetry is relatively easy, due to the fact that the burden of assigning meaning to a text is carried by the reader.

We believe that those who rely on this argument to claim that a given system generates poetry are misusing poetic license, and that by taking such an argument to the extreme, one can potentially justify any randomly created text as being poetry.

In fact, we believe that the opposite of Boden’s claim should be the case: the automatic generation of poetry is harder than that of prose. The reason is that poetry is even further constrained by rules of form that prose need not adhere to. If anything, the invoking of poetic license to turn a phrase requires even greater mastery of the language than that for producing prose. Deviations from the rules and norms must be justified.

2.1.4 Our definition of poetry

For a hypothesis to be scientific it must be falsifiable, i.e. there must be a possible experiment or discovery which would prove the hypothesis false. This is the basis of Sir Karl Popper’s principle of **falsifiability** (Johnson-Laird, 1988). If we follow the line of argumentation from Boden in the previous section and allow any randomly generated output from a program to be considered poetry, then the hypothesis that that program is indeed generating poetry is not falsifiable.

Binsted (1996) points out that falsifiable models are essential for AI research work on subjectively defined artifacts, which in her case consists of jokes. We believe research into poetry generation must step up to the demands of falsifiability.

To illustrate the problem, Figures 2.4 and 2.5 are sample outputs from existing "poetry generators" that we discuss in greater detail in Section 2.3. They are known to employ pure random processes. We make no claims as to being able to determine whether these texts are indeed poems or not. More to the point, we feel that the very decision of whether these texts are poems or not is beyond the scope of this thesis.

Sparkles of whiteness fly in my eyes,
The moan of stars swang branches of trees,
The heart of time sings in the snowy night.
Seconds of Eternity fly in grass,
The Clock of rain turns,
Death of the Apples,
The Equinox penetrates the words.

Figure 2.4: Sample output from ELUAR

wheresoever amorphous – just barely the nightclub,
howsoever apostolic amidst a calamity,
a dragon will irrigate a Copernican currant – an emphysema.
His cowlick must have incinerated a housebroken revelry as per a melamine.
your inactive hydrocarbon could atone.

Figure 2.5: Sample output from ALFRED

Therefore, our account of poetry hinges primarily on its observable aspects, e.g. rhythmic patterns, rhyme, and propositional meaning. Although we clearly recognize the presence of aspects such as imagery, metaphor, and figurative language in poetry, we feel it would be far too difficult to tackle them in this thesis. Firstly, it is not fully understood what the mechanisms that lie behind these aspects are, and more crucially, it is difficult to observe and analyse these aspects.

Our research seeks to account for the unique interaction between form and content in poetry as discussed in Section 2.1.1. If we can construct a model of poetry generation which is able to capture the interaction between the dimensions of metre and propositional content, hopefully it will be extensible to the more abstract aspects in the future.

As there is no clear definition of what counts as “valid” poetry, for the purpose of this thesis we will focus our attention on texts of which the following three properties hold:

- **Meaningfulness**

  This property states that a text must intentionally convey some conceptual message which is meaningful under some interpretation, i.e. with respect to a given knowledge base.

  This is a property that can be said to be held by all types of texts, not just poetry.

  Essentially, by stipulating this property, we are more critical of what we consider *not* to be poetry, i.e. texts which cannot clearly be said to intentionally convey a conceptual message. We believe the texts in Figures 2.4 and 2.5 fall into this category.

  $\mathcal{M}$ is the set of all texts that fulfill the property of meaningfulness.

- **Grammaticality**

  A poem must obey linguistic conventions that are prescribed by a given grammar and lexicon. This is perhaps the most obvious requirement that by definition all natural language artifacts should fulfill. However, in the context of poetry, it is important to state explicitly, as there is the danger of invoking poetic license (see previous section). Despite the notion of grammaticality in poetry being perhaps less constrained than that of ordinary texts, they are nonetheless governed by rules of figurative language tropes. For example, the use of hyperbaton in Section 2.1.2 allows for the line *“I was in my life **alone**”*, but the line *“*In I life was alone my”* would be unacceptable.

  Essentially, we are ruling out simply random sequences of words.

  $\mathcal{G}$ is the set of all texts that fulfill the property of grammaticality.

- **Poeticness**

  A poem must exhibit poetic features, and we must stress again that our definition of poetry hinges on its observable aspects, and thus in the case of poeticness we refer to phonetic features such as rhythmic patterns and rhyme. We make the distinction between poetic constraints, which must be satisfied, and poetic preferences, which are desirable but not necessary.

$\mathcal{P}$ is the set of all texts that fulfill the property of poeticness.

For our purposes, then, **A poem is a natural language artifact which simultaneously fulfills the properties of meaningfulness, grammaticality and poeticness.** In other words, a text x is a poem if $x \in \mathcal{M} \cap \mathcal{G} \cap \mathcal{P}$.

Note that for this definition to be complete, these properties, particularly meaningfulness and poeticness, must be specified further. We deliberately choose not to do so here to preserve the generality of the model. In Section 5.1 we explicitly define, for the purposes of our implemented system, MCGONAGALL, our interpretations of meaningfulness as semantic similarity and poeticness as metre similarity.

This characterization, particularly poeticness, suggests a "classical" account of poetry, i.e. one where adherence to regular patterns in form is paramount. One prime candidate is the syllabic stress metre poetry (Section 1) that obeys certain rhyming patterns, commonly referred to as rhythm and rhyme poetry. This genre is often regarded as being relatively simple and traditional poetry, and can be found, among others, in popular music lyrics, children's poetry, and greeting card poetry. See Figures 2.6 to 2.8 for examples.

The Lion, the Lion, he dwells in the waste,
He has a big head and a very small waist;
But his shoulders are stark, and his jaws they are grim,
And a good little child will not play with him.

Figure 2.6: Hillaire Belloc's *"The Lion"*

There's a saying old, says that love is blind.
Still, we're often told, "Seek and Ye shall find".

Figure 2.7: George and Ira Gershwin's *"Someone To Watch Over Me"*

Far and away you may be,
But the presence of your love is here with me

Figure 2.8: Greeting card poetry

In a sense, this characterisation of what we consider to be poetry creates an artificial lower and upper bound within the realm of poetry. It is a lower bound in the sense that we require poems to adhere to classical notions of poetry, e.g. exhibiting metrical and rhythmic patterns. On the other hand, it is an upper bound in the sense that we will not be considering abstract notions of imagery and metaphor in this thesis.

2.2 The process of creating poetry

The process of writing poetry is often claimed to proceed in a much more flexible manner than other writing processes. It is perhaps almost impossible to formally define and characterise the process of writing poetry. Not only is it difficult to conduct research of an introspective nature into the thought processes of a poet, but these processes vary between poets.

Sharples (1996) models the process of writing as that of creative design, involving a cycle of analysis, known as *reflection*, and synthesis, known as *engagement*. For example, consider a journalist who is writing an article about a topical issue like an election campaign. She will have a clear mission, that is to convey to the reader the salient and newsworthy points of the event, i.e. the “5W + 1H” questions (*who*, *what*, *when*, *where*, *why*, and *how*). A rough draft of the article that conveys these points may be written. Upon reviewing this draft, she may call upon certain journalistic writing practices to make the article more appealing. Perhaps some historical background information on the candidates can be mentioned as well. Finally, the writing might have to conform to non-content specifications, such as having to fit a given amount of column space.

The process of writing poetry can be seen to follow a similar pattern, but the unity between content and form makes the coupling of the reflection and engagement processes much tighter. In particular, the role of reflection might play a bigger role in determining the eventual message of the poem, where there is often no well-defined communicative goal, save for a few vague concepts such as “wintery weather” or “a scary lion”. A poet could begin writing a poem inspired by a particular concept, or scenario, but end up writing a poem about an altogether different topic. During the reflection phase, when looking at an intermediate draft of the poem on paper, a poet may come to realize the opportunities of surface features that can be exploited, which enables further content to be explored upon subsequent engagement phases. Levy (2001) describes a similar model of poetry creation.

Another possible distinction that is suggested by both Sharples (1996) and Boden (1990) is that while a ‘conventional’ writer such as our journalist needs to accept the constraints of goals, plans, and schemas, *creative* writing requires the breaking of these constraints. Yet these constraints are still necessary, as they allow for the recognition and exploiting of opportunities.

2.3 Automatic poetry generation

Bailey (1974), van Mechelen (1992), and Gervás (2002) are the only works we are aware of that extensively document existing attempts at automatic poetry generation. It is also interesting to note that there is very little intersection between the poetry generation systems covered by these surveys, underlining the fact that the field of poetry generation, if it can be said to exist, is still in its infancy, and does not yet have a well-established community, let alone shared goals, perceptions and principles.

Bailey (1974) and van Mechelen (1992) provide information on very early systems which Gervás terms **template-based** poetry generators. A lot of these systems were written by actual poets who were keen to explore the potential of using computers in writing poetry. In fact, several of the systems reported in Bailey (1974) were not fully autonomous, but were used by these poets to assist them in their own poetry writing. This form of computer-assisted poetry is sometimes known as **aleatoric poetry**. One example is the work of the American poet Jackson MacLow.

Gervás (2002) discusses more recent, sophisticated approaches to poetry generation, and provides a taxonomy of poetry generation systems based on the approaches and techniques they use, i.e.:

- **Template-based**: ALAMO *“rimbaudelaires”* (Rubaud et al. (2000), based on reading Gervás (2002))
- **Generate and test**: WASP (Gervás, 2000), our chart generation system (Manurung, 1999)
- **Evolutionary approach**: POEVOLVE (Levy, 2001), McGonagall, i.e. the implemented system detailed in this thesis, a very early version of which was reported in Manurung et al. (2000)
- **Case-based**: ASPERA (Gervás, 2001), COLIBRI (Diaz-Agudo et al., 2002)

There are many ways of categorizing poetry generation systems. Although the technique-based taxonomy is a very useful resource, especially from an engineering viewpoint, we will attempt to categorize existing poetry generation systems based on the goals they try to achieve. In particular, we will categorize poetry generation systems in terms of how their output texts relate to the three sets that embody the properties of meaningfulness ($\mathcal{M}$), grammaticality ($\mathcal{G}$), and poeticness ($\mathcal{P}$) that we have defined in Section 2.1.4.

Under this taxonomy, existing poetry generation systems can be categorized as follows:

- Word salad
- Template and grammar-based generators ($\mathcal{G}$-text generators)
- Form-aware text generators ($\mathcal{P}$ and $\mathcal{G} \cap \mathcal{P}$-text generators)
- Poetry generation systems ($\mathcal{G} \cap \mathcal{P} \cap \mathcal{M}$-text generators)

Note that under this categorization, the natural language generation systems described in Chapter 3 can be viewed as $\mathcal{G} \cap \mathcal{M}$-text generators.

2.3.1 Word salad generators

Most early attempts at poetry generation systems were relatively simplistic programs that paid no heed whatsoever to any of the properties of meaningfulness, grammaticality, and poeticness. These systems simply concatenated random words together. Figure 2.9 shows "LYRIC 3205" by Pete Kilgannon, which van Mechelen (1992) cites as a poem that is constructed by a computer program in exactly this fashion.

judy gotta want upon someone.
wanna sadly will go about.

sammy gotta want the thief him but the
every reason. real distance carry.

Figure 2.9: Random concatenation of words

Under our restricted definition (Section 2.1.4), these systems do not produce poetry.

2.3.2 Template and grammar-based generators

A slightly more sophisticated approach is adopted by what Gervás (2002) terms **template-based** systems. Typically, generation proceeds by randomly selecting words from a lexicon and using them to fill gaps in pre-defined sentence templates. The words and gaps are categorized by part of speech, and typically only substantive words, i.e. nouns, verbs, adjectives and occasionally adverbs, can be varied. Thus, these systems can be said to satisfy, albeit in a very limited fashion, the property of grammaticality.

Certain systems use a slightly more sophisticated account of syntax by using simple phrase structure rules that allow it to generate sentences that are more varied than those generated by template-filling systems.

Figure 2.10 shows an example of some sentence templates and sample output from RETURNER, an archetypal template-filling program discussed in van Mechelen (1992), and Figure 2.11 shows the haiku template and sample output from Masterman's computerized haikus, as discussed in (Boden, 1990).

1. IN THE MORNING + noun phrase with a noun as head + WILL + APPEAR / BE / BECOME / SEEM / TURN + adjective phrase
2. Noun phrase with a noun as head + ALSO / NEVER / OFTEN / SOMETIMES + verb in the present tense + AGAIN
3. LAST NIGHT / TODAY / TOMORROW + pronoun + verb phrase (a verb in past/present/future tense) + pronoun + THROUGH THE WILLOWS

In the morning crowbars will be nearly round.
Separate blankets never step again.
Tomorrow I will ring him through the willows.

Figure 2.10: Sentence templates and sample output from RETURNER

Aside from RETURNER and Masterman's computerized haikus, van Mechelen (1992) lists several other systems of this type, e.g. APPI, BORANPO, etc. We have also found similar systems available on the World Wide Web such as ELUAR, ALFRED The Agent, The Poetry Creator, etc. Figures 2.4 (ELUAR) and 2.5 (ALFRED) in Section 2.1.4 show some output texts from these systems.

All [1] *in the* [2]
I [3][4][5] *in the* [6]
[7] *the* [8] *has* [9]

All green in the leaves
I smell dark pools in the trees
Crash the moon has fled

All white in the buds
I flash snow peaks in the spring
Bang the sun has fogged

Figure 2.11: Rules and sample output from Masterman's computerized haikus

Two notable systems are RACTER and PROSE (Hartman, 1996), which have the distinction of having their poetry being published. The 1984 poetry anthology *"The Policeman's Beard is Half Constructed"* consisted solely of poems constructed by RACTER.

These template-filling systems often employed several clever 'tricks' and heuristics on top of the randomness to give the appearance of coherence and poeticness, such as:

1. assigning ad-hoc "emotional categories", e.g. {ethereality, philosophy, nature, love, dynamism} in ELUAR,
2. choosing lexical items repetitively to give a false sense of coherence, as in RACTER, and
3. constructing highly elaborate sentence templates with only a handful of holes, often to the point that the resulting poems would have to be attributed more to the human template writer than to the program.

Once again, under our restricted definition (Section 2.1.4), these systems still do not produce poetry.

2.3.3 Form-aware text generators

These generators explicitly attempt to fulfill the properties of grammaticality and poeticness, specifically that of metre. Note that although Masterman's program produces haikus (Figure 2.11), which are a well-defined poetic form, we do not place it in this category as the fact that it satisfies poeticness is external to the system: it is imposed by the very restricted templates and careful selection of words that fill the slots.

1. Gervás' WASP system (Gervás, 2000) creates poems in the mould of classical Spanish poetry that satisfy the constraints of strophic forms such as *romances, cuartetos,* and *tercetos encadenados*. The generation makes use of *verse patterns*, which can be considered to be analogous to the templates used in template-filling systems but are more sophisticated in that they are a more abstract representation of syntactic and surface properties of the desired verses. Gervás (2000) lists information such as number of words per verse, rate of adjectives per noun, and tense, as being encoded by these verse patterns. Generation proceeds as a form of greedy search by starting from the first word in the verse pattern and incrementally selecting words that satisfy all the imposed requirements. Additional heuristics that avoid word repetition are also used. Figure 2.12 shows a sample **cuarteto** by WASP. Note the **abba** rhyme scheme.

Muérome por llamar Juanilla a Juana,
que son de tierno amor afectos vivos,
y la cruel, con ojos fugitivos,
hace papel de yegua galiciana.

Figure 2.12: Sample output from WASP (Gervás, 2000)

2. Ray Kurzweil's Cybernetic Poet, or RKCP (Kurzweil, 2001), is a commercial software product, and thus employs proprietary algorithms that we were unable to analyze in depth. However, the information specified on the product website suggests that it generates well-formed poems based on rhythmical properties of words that are compiled in a statistical language model trained on existing poems, as described in the following quote:

> "RKCP uses the following aspects of the original authors that were analyzed to create original poems: the (i) words, (ii) word structures and sequence patterns based on RKCP's language modeling techniques (while attempting not to plagiarize the original word sequences themselves), (iii) rhythm patterns, and (iv) overall poem structure. There are also algorithms to maintain thematic consistency through the poem. RKCP uses a unique recursive poetry generation algorithm to achieve the language style, rhythm patterns and poem structure of the original authors that were analyzed, without actually copying the original authors' writings."

Figure 2.13 shows a sample output from RKCP. Note that although these outputs are explicitly listed on the RKCP website as being haikus, they do not in fact satisfy the strict syllable requirement of a haiku, i.e. three lines of five, seven, and five syllables. Nevertheless, we are prepared to afford it the benefit of the doubt that this is due to the easing of constraints as discussed in the following quote, again from the product website:

> "Sometimes RKCP will discover that it is unable to write the poem or a section of a poem (i.e., a line) when it has fully and recursively exhausted all of the possibilities. It then uses an algorithm to ease the constraints inherent in the goals for particular words. It continues this process of easing constraints and recursively writing the poem (both forwards and backwards) until it can successfully write the poem or section of the poem."

Scattered sandals
a call back to myself,
so hollow I would echo.

Crazy moon child
Hide from your coffin
To spite your doom.

You broke my soul
the juice of eternity,
the spirit of my lips.

Figure 2.13: Sample output from Kurzweil's "Cybernetic Poet"

3. Another interesting example is the *Rimbaudelaires* generated by the ALAMO group (Rubaud et al. (2000), based on reading Gervás (2002)). They first create sentence templates by "cutting out" the nouns, verbs, and adjectives from a given sonnet by Rimbaud. Poems are then generated by completing these templates with words selected from the poetry of Baudelaire. Although essentially a template-filling system as the ones seen above, the words chosen to fill the gaps are claimed to follow "strong syntactic and rhythmic constraints", suggesting that conforming to form is an explicit goal of the generator.

4. Levy (2001) presents a computational model of poetry generation that is remarkably similar in several aspects to the work presented in this thesis, although completely independently arrived at.

   Like the model we introduce in Chapter 4, Levy's model is based on the theory of evolution, which he notes is the prevailing metaphor in creativity theory, after Gruber and Davis (1988). The proliferation of “genetic art” projects such as Sims (1991) bears witness to this.

Levy’s model consists of a **generator module**, which constantly creates “poetic objects” that are analysed by the **evaluator module**, which applies numerical ratings representing their interest in these objects. The generator and evaluator modules run as parallel processes. Levy also states the need for a lexical, syntactic, and conceptual knowledge base with which well-formed poetic objects can be created.

One very interesting aspect is a two-tiered approach to the evaluators, where the lower-tier performs the actual evaluation on the generated objects, and the higher-tier evaluators control the organization of the lower-tier evaluators. This serves two functions. The first is that it is intended to represent the cognitive state, or “consciousness”, of a poet that can choose to devote special attention, or focus, to promising objects. For example, it may direct the system to devote more attention, i.e. processing power, to objects that show relatively more promise and are close to being successful poems. The second function is to determine the relative importance to qualities, or combinations of qualities, that are rated by individual lower-tier evaluators.

The lower-tier evaluators contain either neural networks that have been trained on human judgments of existing texts for subjective qualities, or explicit rules for objective qualities.

An implementation of this full model has not yet been constructed, but there is a prototype, called POEVOLVE, which is hoped to show the potential of the system.

POEVOLVE creates texts that satisfy the form specifications of limericks, i.e. two lines of eight syllables with a stress pattern of *w,s,w,w,s,w,w,s* (where *‘w’* is a weakly stressed syllable, and *‘s’* is a strongly stressed one), followed by two lines of five syllables with a stress pattern of *w,s,w,w,s*, and closed with one more line similar to the first two. The first, second, and last lines must end with rhyming words, as must the third and fourth lines. No account of syntax nor semantics is attempted, for the sake of parsimony and computational feasibility.

It employs a genetic algorithm, a well known heuristic search method that we discuss in detail in Section 4.2. An initial population is created by selecting, from a pool of

1107 words that include phonetic and stress information, appropriately rhyming words that can appear at the end of the five lines, and then selecting more words to fill the rest of the line based on their stress information. Evolution was achieved by mutation and crossover operators that modified the words contained in the limericks. Evaluation was performed using a neural network that was trained on human judgments of how "creative" a selection of limericks are, with a rating from 1 to 6. 25 human-written and 25 randomly generated limericks were used. The randomly generated limericks were generated using the same system as for the initial population.

Kempe et al. (2001) report this experiment in more detail. The neural network was trained on 36 of the limericks and tested on the remaining 14. Unfortunately, no indication of the proportion of human-written and randomly-generated limericks was given. Furthermore, we were unable to obtain any examples of the randomly-generated limericks. The neural network is claimed to have captured some dimensions used by the judges to evaluate the limericks. For the human-written limericks it produced an average rating of 3.4 (compared to 4.8 by the human judges), and for the randomly generated poems it produced an average rating of 1.9, compared to 1.7 by the human judges.

When used in a GA that ran for 1000 iterations to evaluate the generated limericks, the difference in quality from the initial population, 1.2, and last 100 generations, 1.6, was claimed to be a small but significant improvement.

Once again, under our restricted definition (Section 2.1.4), these systems still do not produce poetry. Note that although Levy's theoretical model that underlies POEVOLVE describes a fully-fledged poetry generation system such as the ones described in Section 2.3.4, the implemented system itself is only a $\mathcal{P}$-text generator.

2.3.4 Poetry generation systems

This last category consists of systems that explicitly attempt to fulfill the three properties as stated in our restricted definition of poetry ($\mathcal{G} \cap \mathcal{P} \cap \mathcal{M}$).

1. ASPERA and COLIBRI

   ASPERA (Gervás, 2001) and COLIBRI (Diaz-Agudo et al., 2002) are two similar systems which use **case-based reasoning** to generate formal Spanish poetry.

Case-based reasoning is a general problem-solving technique in AI. A case-based reasoner attempts to solve a new problem by consulting an explicit database of existing problems and their solutions (Luger and Stubblefield, 1998). This process is described by Aamodt and Plaza (1994) as a cycle of four processes, namely **retrieve**, **reuse**, **revise**, and **retain**. Essentially, an existing solved problem, i.e. a **case**, that is deemed to be similar to the current problem at hand is selected ("retrieve"), and its solution is applied to the current problem ("reuse"). However, since the problems are not identical, this may involve having to alter the solution to be applicable to the current problem, or to yield an acceptable solution ("revise"). Finally, if the procedure is successful, it is stored to the database as another case ("retain").

In ASPERA, the problem to be solved is specified through user input consisting of a prose description of the intended message, and the type of poem required, i.e. length, mood, and topic. This input is first passed to a custom-built 'Poetry Expert' knowledge base which selects a particular strophic form, similar to the ones used in WASP, which is deemed to be suitable for the given input.

Cases in ASPERA are encoded as line patterns, which are similar to the templates used by the systems in Section 2.3.2. Each pattern encodes both the part of speech tags and the actual words from a line in an existing poem of the chosen strophic form. Aside from these cases, the system also loads a vocabulary that is also specified by the chosen strophic form.

Furthermore, the input message is distributed into "poem fragments", which will eventually correspond to a line in the resulting poem. For the example given in Gervás (2001), the input message `''Peter loves Mary they go together to the beach''` is split into three fragments:

```
(fragment 1 Peter loves Mary)
(fragment 2 they go together)
(fragment 3 to the beach)
```

At this point, the case-based reasoning can begin. Generation of a poetry is done on a line by line basis in the following manner:

(a) **Retrieve step**: for each poem fragment, adequate line patterns and words must be selected. This is done based on a similarity metric that prefers words appearing in the fragment, and words with the required rhyme pattern.

(b) **Reuse step**: a draft for the current line is constructed by using the part of speech information from the chosen line pattern as a template. Words from the vocabulary with the appropriate part of speech are substituted into the correct positions.

(c) **Revise step**: after a full draft has been generated, it is presented to the user, who is required to validate it. It is during this stage that modifications should be made to ensure metrical correctness. However, Gervás (2001) states that this feature is not yet implemented and therefore assumes that the user performs the necessary modifications.

(d) **Retain step**: all validated verses are stored in a personal database that can be used as an extension to the existing corpus of line patterns.

COLIBRI (Diaz-Agudo et al., 2002) performs in a vaguely similar fashion to ASPERA, but the CBR method for generation is applied to the whole verse rather than line-by-line. Additionally, the cases are stored in a very flexible representation using LOOM, a Description Logic system. This enables COLIBRI to exploit the inferences enabled by the inheritance of features in the ontology. Furthermore, COLIBRI has an implemented system of repairs and revisions that ensure rhyme and metrical properties are met following the adaptation process, that strives to achieve meaningfulness.

As an example of how these systems work, observe Figure 2.14(a)-(d). In (a) we see an example of the “input semantics”. It is not treated as an encoding of semantics as such, but as a list of keywords that will inspire the choice of words during the generation process. In (b) we see a case that is retrieved from the corpus of poems in the appropriate strophic form. The result of adaptation (i.e. reuse step) can be seen in (c): the words marked in boldface are taken from the input message, and have been chosen based on their part of speech. Finally, in (d) we see the result of the repair/revision step, where the words marked with * have been substituted in to maintain the metre and rhyme specifications.

In both these systems, we can observe that grammaticality is upheld through the use of line and verse templates, and the fact that word substitution is only done with words with the correct part of speech. It is not clear, however, whether other facets of grammaticality are considered, i.e. subcategorization, selectional restriction, agreement.

una boca ardiente pase techo y suelo

(a)

no sólo en plata o viola truncada
se vuelva mas tú y ello juntamente
en tierra en humo en polvo en sombra en nada

(b)

no sólo en **boca y** *viola* **ardiente**
se **pase** *mas tú y ello juntamente*
en tierra en **techo** *en* **suelo** *en sombra en nada*

(c)

no sólo **para* boca y** *viola* **ardiente**
se **pase** *mas tú y ello juntamente*
en **tía*** *en* **techo** *en* **suelo** *en sombra en* **serpiente***

(d)

Figure 2.14: Examples of (a)input message, (b)verse pattern, (c)output from adaptation, and (d)output from revision in COLIBRI

Meaningfulness is approximated through the rather simplified use of an "input message" that consists of a sequence of words that the system will attempt to distribute across the text. These words are intended to represent the semantic message of the resulting text.

Lastly, poeticness is achieved through the use of domain knowledge that encodes the rules of the strophic forms in formal Spanish poetry. COLIBRI possesses explicitly designed procedures for fulfilling these forms.

We note two limitations of these systems. Firstly, although we categorize ASPERA and COLIBRI as poetry generation systems ($\mathcal{G} \cap \mathcal{P} \cap \mathcal{M}$-text generators) because we see all three properties explicitly represented within these systems, their approximation of semantics using a list of keywords that are randomly distributed across an existing verse template is a very shallow account of meaning, and such a treatment cannot be said to truly satisfy the property of meaningfulness.

Secondly, we note that in both these systems meaningfulness is first handled during the retrieve and reuse steps, while poeticness is handled separately during the revise step. Regardless of the sophistication of the account of meaningfulness, modifications to the text at the latter stage, which are motivated by poeticness, may perturb whatever meaning has already been set up in the earlier stages. Thus, there is no way of ensuring that the end result satisfies meaningfulness. This is a problem reminiscent of the architectural problems in NLG (Section 3.2).

2. Our chart generation system

   Manurung (1999) reports our initial attempt at poetry generation, where we developed a system that performs an exhaustive search for a text that conveys the semantics of a given first order logic proposition (meaningfulness) and satisfies a given rhythmic stress pattern (poeticness), according to a given lexicon and grammar (grammaticality). Due to the combinatorial explosion of the problem, we employed **chart generation**, a well known technique in natural language processing which yields polynomial running times, as opposed to exponential (Kay, 1996). Chart generation is discussed in detail in Section 3.4.2.

   Figure 2.15(b) shows a sample text that exhibits the rhythmic patterns of a limerick while managing to convey the semantics specified in (a). In terms of satisfying its notions of grammaticality, meaningfulness, and poeticness, this chart generation system performs immaculately.

   Unfortunately, due to its exhaustive search, it is computationally very expensive, despite the use of the chart. Furthermore, it is limited in its flexibility, in the sense that it can **only** generate immaculate results. In cases where the given linguistic resources are unable to produce the required output, it fails completely. We believe that the robustness of a poetry generation system that is able to produce as good as possible an output, given the resources at its disposal, is a desirable property. Finally, unlike our eventually chosen model of poetry generation as stochastic search (Section 4.1), this system does not facilitate the exploiting of opportunities for generating a poem's content.

2.4 Summary

In this chapter we have examined some of the defining characteristics and features of poetry, and subsequently formulated our restricted definition of a poem as a text that fulfills the three properties of **meaningfulness, grammaticality, and poeticness**.

```
cat(c) ∧ dead(c) ∧ eat(e,c,b) ∧ past(e) ∧ bread(b) ∧ gone(b)
```

(a)

The cat is the cat which is dead.
The bread which is gone is the bread.
The cat which consumed
the bread is the cat
which gobbled the bread which is gone.

(b)

Figure 2.15: Examples of (a)input semantics and (b)output limerick from our chart generation system

We have also examined existing work on automatic poetry generation, and note that most systems only account for a subset of the three properties of our definition of poetry. Although ASPERA and COLIBRI attempt to account for meaningfulness, grammaticality, and poeticness, their account of semantics is far too shallow. Our own chart generation system is thus the only example of an existing implemented system which is a proper poetry generation system according to our restricted definition. Unfortunately, from a computational cost standpoint it is prohibitively expensive, and it lacks robustness in finding solutions.

Secondly, we noted an architectural rigidness inherent in ASPERA and COLIBRI, where meaningfulness and poeticness are each handled during separate steps, unlike our chart generation system, where every added edge is always checked against the three properties of meaningfulness, grammaticality, and poeticness.

In the next Chapter we will provide a survey of **natural language generation**, or NLG, a subfield of artificial intelligence and computational linguistics which heavily influences the design of McGonagall, our implemented poetry generation system. As poetry is an instance of natural language, we feel that our endeavour into poetry generation should be informed by the NLG literature. Furthermore, as mentioned in Section 2.3, NLG systems can be seen to satisfy grammaticality and meaningfulness, two of the three properties in our definition of poetry. Thus, studying the NLG literature is relevant to our work.

Chapter 3

Natural language generation

In this chapter we will provide an overview of research work done in the field of natural language generation, or NLG. As poetry is an instance of natural language, it seems obvious that any research endeavour into poetry generation should be well-informed of natural language generation.

This chapter is not intended to serve as a general review of the entire NLG literature. Instead, we will focus on several areas of interest that we feel address issues encountered in our task of poetry generation.

We start by describing what has come to be accepted as a fairly standard definition of what NLG is, along with the processes and tasks involved, and discuss the most prevalent paradigm, i.e. that of NLG as being **communicative goal-driven planning** (Section 3.2).

We will then discuss work that has exposed the limitations of this paradigm, along with various alternative models which attempt to overcome these limitations. In particular, we review work on

- alternative NLG architectures (Section 3.3),
- better search methods for NLG (Section 3.4),
- overgeneration and ranking methods of NLG (Section 3.5), and
- opportunistic planning in NLG (Section 3.6).

We will also describe several instances of NLG systems that implement these alternative models, especially those that are of significant relevance to our work. Finally, we will relate these problems to our task domain of poetry generation and discuss the requirements of our poetry generation system.

3.1 What is NLG?

Natural language generation, or NLG, is the sub-field of artificial intelligence and computational linguistics that focuses on the development of computer systems that can produce understandable texts in a human language, starting from some nonlinguistic representation of a **communicative goal** as input (Reiter and Dale, 2000). To achieve this, NLG systems employ knowledge resources about both the specified human language and the application domain.

Evans et al. (2002) remark that standard definitions of what an NLG system should specifically do are notoriously asymmetrical. For instance, although there seems to be a consensus as to the output of an NLG system, namely human language texts, the form of input it should receive and the different types of processes to be performed on it, vary from system to system. Indeed, comparative studies of existing NLG systems such as Paiva (1998) and Cahill and Reape (1999) reveal many differences as to what, how, and when operations are performed, if at all. However, certain theories and models have come to be widely accepted, and we will discuss them in the following sections.

3.1.1 Input specification

As we have briefly mentioned, the input to an NLG system is some nonlinguistic representation of a communicative goal coupled with various knowledge resources. Examples of such goals are informing, requesting or persuading the hearer to do something or to provide information.

As formally defined by Reiter and Dale (2000), the input to an NLG system is a four tuple $< k, c, u, d >$, where k is the knowledge source to be used, c is the communicative goal to be achieved, u is the user model of the intended audience, and d is the discourse model.

The knowledge source, k, is essentially all the background information about the task domain. Its representation varies from system to system, e.g. first order predicate logic propositions, relational database tables. For instance, ILEX (O'Donnell et al., 2001), an NLG system that generates short texts describing pieces of jewellery in a museum, uses a database of known facts and relations concerning the pieces, whereas FOG (Goldberg et al., 1994), a weather forecast report generator, receives as its input a table of sampled meteorological data.

The communicative goal, c, is the purpose of the text to be generated. This usually consists of an underspecification of some propositional information to be conveyed. Some systems further specify rhetorical relations within the propositional information that are to be conveyed by the text. Additionally, c can include non-propositional constraints that must be obeyed by the resulting text, e.g. a requirement that the resulting document fit on a certain number of pages.

The user model, u, is a characterisation of the intended audience for whom the text is generated, which can affect the generated text according to prior knowledge of that audience's expectations. For instance, a stock market reporting system could have a report tailor-made for a broker, an executive analyst, or a public newsfeed. Factors which could be affected by u include content, grammar and stylistic preferences, e.g. a broker's report could be much more in-depth and detailed in terms of content, whereas a public newsfeed could include clear explanations for technical terms.

Finally, the discourse model, d, keeps track of the history of what has been conveyed by the system so far. This is useful during the syntactic and lexical realisation stages, for instance, for referring back to material that has already been conveyed for clarity or brevity's sake, i.e. anaphora.

These last two components, u and d, are not always present in NLG systems, or sometimes they are implicitly “hardwired” into the system.

Note that linguistic resources, i.e. the grammar and lexicon of the particular human language being targeted, is not specified in Reiter and Dale (2000)'s four-tuple, implying that this resource is an internal component of the NLG system. It is plausible, however, to envision an NLG system for which these linguistic resources are viewed as a separate input, e.g. a multilingual system that can target a document to several different languages.

3.1.2 Output specification

The output from a natural language generation system is, naturally, a text, i.e. a natural language artefact that can be understood by a human. Most research in NLG is not concerned with the formatting details of the output, and will simply produce strings of words. However, recent research has looked at augmenting NLG systems with the ability to produce texts that incorporate information on visual layout (Bouayad-Agha et al., 2000) and prosody for speech synthesis (Prevost, 1996).

3.1.3 Processes and tasks

What takes us from the given input of communicative goal to the desired output of natural language text, and how is this process organized?

Reiter and Dale (2000) contend that it is unlikely for a complex and complete NLG system to be constructed as a single monolithic process. Instead, it is decomposed into distinct, well-defined and easily-integrated modules. From a practical point of view, the system becomes easier to build and debug, whereas from a theoretical point of view, the generation task can be handled at separate representational levels of language that are already widely used and studied, e.g. semantics, syntax, lexis, etc.

A common broadstroke description of the NLG process comprises two levels, the **strategic** level and the **tactical** level (Thompson, 1977, McKeown, 1985). The first level can roughly be paraphrased as being the “what-to-say” level, and the latter the “how-to-say-it” level. They have also been referred to as deep generation and surface generation (McKeown and Swartout, 1988), and the conceptualizer and formulator (De Smedt et al., 1996).

The “Reiter model” is a widely held model that refines the strategic/tactical distinction into three processes (Reiter, 1994):

1. Content determination

   The first process is to construct a selection of information from k that realizes the communicative goal in c, or in terms of propositional semantics, is subsumed by the underspecification in c. In certain cases, c may already explicitly provide the exact information to be conveyed.

   This process of selecting information can be affected by the user model u, e.g. the level of a user’s technical expertise may determine whether certain domain-specific concepts are explicitly elaborated or simply glossed over.

2. Sentence planning

   Once the propositional information has been determined, it must then be structured and ordered so as to produce a coherent and meaningful text. This includes "chunking" the information into smaller segments, determining the relationship between these segments, and constructing the global syntactic structures that will be used to realize these segments as spans of text, e.g. generating referring expressions, choosing content words, etc.

3. Surface realisation

   When the global syntactic structure has been realized, the available linguistic resources are consulted to produce actual spans of text. The tasks here include determining the complete syntactic structure of the sentences, choosing the appropriate function words to be used, and fulfilling all necessary grammatical requirements such as inflectional morphology.

Roughly speaking, content determination corresponds to the strategic level, and surface realisation corresponds to the tactical level. The process of sentence planning overlaps both the strategic and tactical levels.

Mellish and Dale (1998) refine this further by identifying six main categories of problems that are addressed in most NLG research: content determination, document structuring, lexicalization, aggregation, referring expression generation, and surface realization.

3.2 Traditional NLG system organisation

A prevalent paradigm adopted in NLG research, and one that reflects the definition in the previous section, is that of NLG as a top-down, goal-driven, planning process. The production of an utterance is a **communicative act**, an attempt to satisfy some communicative goal. As a result, the processes of content determination, text planning, and surface realisation are often implemented as a deterministic pipeline that starts from the communicative goal and gradually transforms it into a text, using all linguistic and domain knowledge sources available to it. This often involves the matching of various schema-like rules to select transformations to be applied. Figure 3.1 shows a diagram of this **pipeline** architecture.

The benefit of this is the relative simplicity and elegance of making local decisions at the various linguistic representational levels. Content determination deals with the propositional semantics of the message. Once it has selected a set of propositions, it commits to this selection and proceeds to the sentence planning phase, where syntactic structures are selected and arranged to convey the selected propositions. Finally, during the surface realisation phase, the linguistic resources are consulted and the syntactic structures are transformed into text strings. From a practical point of view, this results in a simpler, more manageable and implementable system, due to the strict decomposition of data and processes.

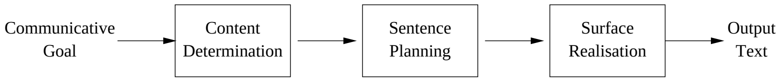


Figure 3.1: The three Reiter processes in a pipeline architecture

This paradigm is not without its limitations. A lot of decisions must be made during the generation process, particularly during text planning. For example, aggregation, pronominalization, generation of referring expressions, etc. These decisions are interdependent: making choices in one aspect can preclude the possibility of choices in other aspects. When these decisions are made by the separate modules in the pipeline architecture above, the resulting texts may be suboptimal, and in the worst case, the system may fail to generate a text at all. This problem has been identified in Meteer (1991) as the "generation gap", in Kantrowitz and Bates (1992) as "talking oneself into a corner", and also in Eddy (2002), who notes that it is not easy to determine what effect a decision taken at an early stage in the pipeline will have at the surface, and decisions taken at one stage may preclude at a later stage a choice which results in a more desirable surface form. Examples of aspects of surface form which may want to be controlled are: collocations and idiomatic constructions (Stone and Doran, 1996, Kamal, 2002), document length (Reiter and Dale, 2000), and emulating individual personality traits (Kantrowitz and Bates, 1992).

Although the drawbacks to this approach of tackling interdependent decisions one by one are known to the NLG community, nevertheless a significant proportion of NLG research and development adopts this approach. From an academic viewpoint, the individual processes and tasks themselves are often the subject of investigation, and thus it makes sense to observe them independently of other factors. From an applied viewpoint, this is due to the fact that aside from being the easiest solution to design and implement, it is often the most computationally efficient method, and it works for suitably restricted domains, which is true of most real-world NLG applications. Unfortunately, the drawback here is that often the ordering and interaction between the separate modules that handle the various tasks are so finely tuned for a specific restricted domain that they are not portable to other domains.

Nevertheless, in recent years research into alternative methods to overcome these limitations has become increasingly popular. We can roughly divide them into the following three categories, although as we shall see, there is considerable overlap in these approaches:

- **Alternative architectures**: identifying the pipeline architecture as the main culprit, works such as Nirenburg et al. (1989), Kantrowitz and Bates (1992) and Robin (1994) explore different ways of organizing the modules. This approach is discussed in Section 3.3.
- **Better search methods for NLG**: works such as Eddy (2002), Stone et al. (2001), Mellish et al. (1998a), and Kamal (2002) view NLG as a constraint satisfaction problem, and propose solving it with various search algorithms. This approach is discussed in Section 3.4.
- **Overgeneration and ranking methods**: a unique search method that places emphasis on a discriminative, as opposed to generative, model of NLG. Examples of this approach are Langkilde and Knight (1998), Oberlander and Brew (2000), and Varges (2002). This approach is discussed in Section 3.5.

3.3 Alternative architectures

Although most NLG systems seem to employ the same subtasks and modules, they differ in the way these modules are organised. This is an issue of which architecture is used for the system as a whole. Figure 3.2 gives a sketch of the different architectures that have been used in NLG systems, with the maximally-independent pipeline, or sequential, architecture at one extreme, and the integrated architecture, where knowledge at all levels act together, at the other (De Smedt et al., 1996).

We have already described the **pipeline** architecture above. Due to its computational efficiency and relative simplicity and maintainability, it is the most commonly used architecture in applied NLG systems. However, it suffers from a problem of architectural rigidness, where the possible interactions and mutual constraints between various linguistic levels might not be accounted for (see Section 3.2).

The **revision** architecture adds another phase, revision, where the resulting text originating from the pipeline is modified to repair any deficiencies that can be identified, or to greedily

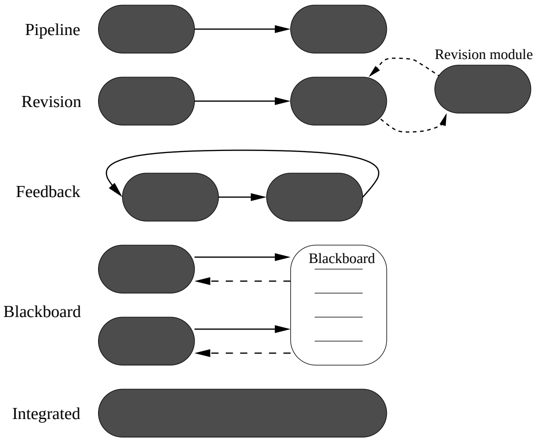


Figure 3.2: Variety of NLG system architectures

achieve remaining auxiliary goals as much as possible.

The **feedback** architecture tries to remedy this by providing some manner of feedback information flow from the latter stage, e.g. surface realiser, back to the conceptual module. Initially, content determination supplies an underspecified input. Therefore, ’gaps’ or ’holes’ will appear in the partially generated form, which is passed back to content determination as feedback to guide further provision of input. There are several ways of implementing the facilitation of feedback. In POPEL (Reithinger, 1991), a separate request handler is used to provide the necessary interface between modules, while in IGEN (Rubinoff, 1992), the surface realisation component provides feedback to the content determination component in the form of annotations that tell it how much of the content can be covered by a particular word choice (De Smedt et al., 1996). However, these approaches do not elegantly handle certain multi-level linguistic phenomena, which require a combination of many individual local effects that contribute to create a global effect.

Another form of interaction is provided by the **blackboard** architecture, which consists of several modules that operate independently and pass and receive information without knowing exactly which other modules use it. Instead, each module posts its information in a shared resource called a blackboard, and also looks in it for information posted by other modules that it needs. The advantage is a highly flexible and modular system where independent modules need not execute according to a sequential plan, but act whenever the required triggering information becomes available. Although this architecture seems capable of of handling the aforementioned multi-level linguistic phenomena, the flexible interaction requires a lot of extra control mechanisms to resolve conflicts and maintain efficiency. Examples of systems that use this architecture are DIOGENES (Nirenburg et al., 1989) and the HealthDoc sentence planner (Wanner and Hovy, 1996).

Finally, the **integrated** architecture is perhaps the most radical approach where there is no explicit modularisation at all. Instead, all decision-making processes are uniformly handled as a constraint-satisfaction problem. The particular technique used to solve this problem can vary, e.g. planning, constraint logic programming, theorem proving, or the various state space search methods. The advantage of this approach is that it enables simultaneous consideration of constraints at different representation levels, thereby overcoming the limitations described in Section 3.2. The disadvantage, however, is that finding a solution to a constraint satisfaction problem requires search, which can be computationally expensive. Moreover, the integration of different levels of linguistic constraints imposes requirements on the representation and reasoning mechanisms, and can lead to them being difficult to develop and unwieldy.

KAMP (Appelt, 1985) and GLINDA (Kantrowitz and Bates, 1992) are examples of this kind of architecture. The majority of systems described in the next section can also be seen as integrated architectures, although they are explicitly designed as tactical components of NLG systems, thus achieving integration for the sentence planning and surface realisation modules.

3.4 Better search methods for NLG

Following on from the previous section, where we introduced the integrated architecture, many NLG researchers feel that it is this approach that best overcomes the limitations of the pipeline. Reiter and Dale (2000) state that in principle, integrated architectures should allow NLG systems to generate better texts than pipeline architectures. However, they note the drawback that from a practical perspective they are very expensive to engineer. Solving multiple constraint satisfaction problems with search methods is a well-known problem in many AI domains, and to that end various techniques have been adapted for the NLG process.

We believe, as does Eddy (2002), that the distinction between pipelined and integrated architectures is not a sharp one, and can be analysed from a state space search point of view as being opposing ends on a spectrum of increasing complexity. A pipeline architecture aggressively constrains its search space when committing to decisions at earlier stages. This is both an appealing factor, in terms of computational efficiency, and a drawback, in terms of its limited expressiveness.

One feature that we have observed from the search approaches to NLG that we describe below is that they all deal with the tactical aspect of generation, i.e. how to realize a previously selected, and possibly structured, set of propositions. Thus, the integration they achieve is between the stages of sentence planning and surface realisation, which includes tasks such as aggregation, referring expression generation, ordering, embedding, and lexicalization. Indeed, it is between these two stages where the problematic interdependent linguistic constraints most commonly lie.

We will now examine several NLG systems that implement the paradigm of NLG as search.

3.4.1 Hillclimbing and greedy search

Hillclimbing search is perhaps the simplest form of heuristic search. Starting from an initial state, which is either a randomly selected state or a special state that represents some domain-dependant initial configuration, hillclimbing search considers all possible moves it can make and always selects the best one according to its heuristics. It repeatedly does this until it reaches a state where no move yields a better state. It is very efficient, but has the drawback of being liable to get trapped in a local maximum (or minimum) (Russell and Norvig, 1995). Greedy search is very closely related to hillclimbing search. Like hillclimbing, it employs a heuristic that evaluates all possible moves that can be made at any one time. Unlike hillclimbing, it uses this information as a queuing function to determine the order in which to explore other states. In other words, unless prematurely terminated by the discovery of a goal state, it will attempt to exhaustively comb the entire search space. However, as we shall see, several works seem to treat the terms hillclimbing and greedy interchangeably. Indeed, greedy search without backtracking is essentially the same as hillclimbing.

The SPUD (Sentence Planning using Descriptions) system (Stone and Doran, 1997, Stone et al., 2001) is a sentence planner that generates sentences from descriptions of syntactic, semantic and pragmatic goals. It borrows from work in referring expression generation such as Dale and Haddock (1991). The goal task of referring expression generation is to generate a noun phrase that is minimal in a sense similar to Grice’s maxim of quantity (Grice, 1975), and is able to uniquely identify the entity being referred to from a set of **distractor** entities that are in the discourse context. This is accomplished by incrementally selecting lexical items based on their property of ruling out as many distractors as possible.

SPUD adapts this approach for the generation of sentences. Given a set of semantic and pragmatic goals, it incrementally selects syntactic and lexical constructions that fulfill as many remaining goals as possible. These constructions are implemented as LTAG trees (Joshi and Schabes, 1991) annotated with semantic and pragmatic information.

The SPUD generation algorithm is an implementation of a hillclimbing search algorithm (they actually call it greedy search) which considers all syntactic operations (in the case of LTAG, substitution and adjunction) that are contextually warranted, and ranks them using an ordered list according to semantic, pragmatic and syntactic criteria. It executes the best ranked operation, and repeats this process until it can proceed no further. The list of criteria is processed in a fashion similar to optimality theory (Prince and Smolensky, 1993), and is discussed in detail in Stone et al. (2001).

Stone and Doran (1996) show how the simultaneous consideration of various constraints enables them to generate collocations and idiomatic constructions.

Nicolov et al. (1995, 1996), Nicolov (1998) describe PROTECTOR, a tactical generation system. It generates sentences that approximately convey semantics represented by an input conceptual graph (Sowa, 1984). Given a partial semantic-syntactic structure that represents the target text (in most cases an empty sentence node), it finds appropriate mapping rules that are syntactically licensed and cover a portion of the input conceptual graph. Mapping rules are semantic-syntactic structures that indicate which syntactic structures can be used to convey certain semantic specifications.

PROTECTOR is unique in the sense that it can generate sentences that more or less convey the input semantics, and keeps track of what differences there are between the input and generated semantics. One can define upper and lower bounds on the semantics that can be conveyed.

The generation process is divided into several stages: building a skeletal structure, covering the remaining semantics, and completing a derivation.

The first stage of generation attempts to select an initial mapping rule that covers as much of the input semantics as possible, using the maximal join operation of conceptual graphs, and is syntactically applicable to the partially constructed tree. A mapping rule may have internal generation goals, which are applied recursively by finding appropriate mapping rules.

The second stage involves covering as much as possible of the remaining semantics using additional mapping rules. At this stage, only mapping rules that specify the adjunction operation are considered (to be precise, sister-adjunction as defined in D-Tree grammars, the formalism they use). The choice of mapping rules is influenced by the criteria of connectivity, integration, and realisability.

Finally, the derivation is completed by adding all syntactic requirements to complete a sentence.

PROTECTOR is very similar to SPUD save for the following differences:

- it does not consider pragmatic constraints the way SPUD does,
- it formalizes the concept of approximate generation by allowing the generation of more or less than the original input semantics, and
- it does not implement the hillclimbing algorithm in the sense that SPUD does. However, the three "greedy" steps of building a skeletal structure, covering the remaining semantics, and completing a derivation, seem to encode a manner of implicit hillclimbing. Within each step, though, a systematic search is used (see Section 3.4.2).

3.4.2 Systematic, exhaustive search

The hillclimbing search algorithm has the limitation of being incomplete in that it cannot guarantee to find an optimal solution even if it exists. Although Stone et al. (2001) claim that their use of SPUD for various NLG tasks supports their decision of using this search strategy, and that its consistent search behaviour makes the system, and its specifications design, easier to understand, it is nevertheless a desirable property to have a complete search algorithm.

Exhaustive search algorithms, which systematically comb the entire space, are complete algorithms, although they come at a cost of computational inefficiency. Indeed, with the combinatorial properties of syntactic structures, they can be prohibitively expensive.

Nevertheless, certain techniques that keep the problem tractable do exist, such as:

1. **Chart generation**

   First suggested by Kay (1996), it employs the use of a data structure called the chart, which is very widely used in parsing (Jurafsky and Martin, 2000). In parsing, the pervasive nondeterminism of natural language causes inefficiency when previously parsed constituents have to be constantly reconsidered. With the chart, however, substantial savings are achieved because the chart stores all complete constituents once they are constructed. Thus regardless of the number of parses they may appear in, they will only be constructed once. This is a common technique in AI known as **memoization**. The chart also stores incomplete constituents, which are predictions of constituents yet to be found.

A chart can be viewed as a graph where nodes signify positions between words in an input string, and the edges signify analyses spanning substrings in the input string (Popowich, 1996). Edges which describe complete constituents are called inactive edges, whereas incomplete constituents are represented by active edges. Edges in the chart can interact to build up larger structures, and an algorithm that uses the chart must ensure that when adding a new edge to the chart, all its interactions with existing edges that give rise to new edges are considered as well. This guarantees completeness.

Kay (1996) views chart generation as the “parsing” of semantic representations such as first-order predicate logic, which can be treated as free word order languages.

Kay (1996), Popowich (1996), and Carroll et al. (1999) are all examples of the use of charts in generation. In a prior experiment (Manurung, 1999), we experimented with using charts to generate texts that convey a given semantics but also adhere to a given metrical pattern (see Section 2.3.4). Similarly, Gardent (2002) embeds the SPUD generation system described above in a chart generation framework that performs checking of pragmatic constraints to generate fluent and natural-sounding texts. Nicolov (1998) mentions the use of chart generation within the PROTECTOR framework described above.

2. **Truth Maintenance Systems**

In addressing the issue of lexical choice, Kamal (2002) proposes using Assumption-based Truth Maintenance Systems (de Kleer, 1986), or ATMS, as an alternative mechanism to deal with the computational costs of exhaustive search. Truth maintenance systems are used in problem solving systems as ‘book-keeping’ components in conjunction with an inference engine. The inference engine uses domain knowledge and heuristics to logically induce a solution to the problem at hand, passing on all justifications and assumptions it posits to the truth maintenance system. The truth maintenance system keeps record of possibly conflicting assumptions, and can return to the inference engine a set of beliefs and contradictions. In NLG, this system is therefore similar to chart generation in that it keeps track of all partially built structures. However, Kamal (2002) claims that an ATMS architecture is able to cache more information such as ordering information and lexical properties. It can also record reasons for contradictions, enabling it to safely prune the search space even further.

3. **Redefining search problem**

   Eddy et al. (2001) take a different approach in tackling the combinatorial explosion of exhaustive search in NLG. They present an algorithm that transforms the search space into a manageable one, by exploiting linguistic knowledge. Using the TAG formalism (Joshi and Schabes, 1997), they first define the search space as the space of all possible sequences of compositions of elementary trees. They then introduce the concept of specific lexicalisation, which allows the transformation of a TAG into a lexicalised version of itself. This enables them to identify symmetries and redundancies in the search space. The resulting grammar produces the same set of surface strings but yields a much smaller search space in terms of possible sequences of compositions.

4. **Constraint logic programming**

   This is another approach that also prunes the search space before enumerating the search space exhaustively. Power (2000) and Kibble and Power (2000) show how it can be used to address the problem of text structuring, similar to that in Mellish et al. (1998a). In their implemented system, ICONOCLAST, domain knowledge about the rhetorical structure of the target texts is encoded as hard and soft constraints. Hard constraints are encoded in the integer domain and, using constraint logic programming techniques, can be used to reduce the search space by ruling out regions that are known to violate the constraints. This reduced search space is then enumerated exhaustively and ranked by order of preference using the soft constraints.

3.4.3 Stochastic search

Stochastic search is a form of heuristic search that relies on random traversal of a search space with a bias towards good solutions. It is similar to hillclimbing search, but instead of always applying the best move possible at any given state, it allows a degree of flexibility in applying other moves. A stochastic bias is applied to favour better moves. This flexibility allows it to escape local maxima. Examples of stochastic search are, among others, simulated annealing, evolutionary programs, genetic algorithms, and ant colony optimization. They have been shown to outperform deterministic approaches in a number of domains such as planning, scheduling and constraint satisfaction (Fogel, 1995, Bäck et al., 1997).

Mellish et al. (1998a) present some initial experiments using stochastic search methods for sentence planning in ILEX, a system that generates descriptions of museum artefacts, in particular, 20th century jewellery. Their work concerns a specific aspect aspect of text planning: selecting and ordering a set of facts into a structure that yields a coherent text. They use Rhetorical Structure Theory (Mann and Thompson, 1987), where a text is analysed as a hierarchy of relations between text spans, and a relation holds between a **nucleus** text span and a **satellite** text span. An analysis of a text takes the form of an **RS tree** .

Mellish et al. (1998a) identify 3 main approaches to controlling the search for a good RS tree:

1. **Restrict what relations can appear in the nucleus and satellite**. An example of this approach is Hovy (1990) with his idea of "growth points". This approach requires explicit commitments to be made prior to the generation task as to what configurations of relations produce coherent and flowing text. These rules are a "hard" encoding of heuristics, and thus this approach resembles schema-based NLG (McKeown, 1985). It can therefore be expected to work very well in restricted domains where limited text patterns are used, but in general it detracts from the advantage that RST has over schema-based approaches, namely the capacity to generate more flexible types of text.

2. **Exploit known information about the communicative goals to limit search possibilities**. An example of this approach is Moore and Paris (1993). This approach works well if there are strong goals in the domain which can influence textual decisions. Unfortunately, this is not always the case, as a communicative goal can often be very general or vague. In the case of ILEX, a typical goal would be something like "say interesting things about item X, subject to length and coherence constraints". .

3. **Perform a state-space heuristic search through all possible trees, guided by some notion of tree "quality"**. This approach was first suggested by Marcu (1997), and forms the basis of the work in Mellish et al. (1998a), who use several stochastic search methods, including genetic algorithms.

In their experiments, legal RS trees were randomly created from an input set of facts and relations in Prolog-style notation. These trees were then modified using the genetic crossover operation of subtree swapping, as widely used in genetic programming (Koza, 1994). The trees were then evaluated by an ad-hoc scoring function which took into account factors such as topic and interestingness, substructure size, information preconditions and focus movement.

A similar approach is taken by Cheng (2002) for the task of aggregation, and Karamanis and Manurung (2002), in which a prototype of our system, MCGONAGALL, was used to address the same task in Mellish et al. (1998a), but using the principle of entity-based coherence (Karamanis, 2001, Knott et al., 2001).

3.5 Overgeneration and ranking methods

In recent years, an alternative method of NLG has become increasingly popular. It is based on the **generate and test** principle, a well known approach to problem solving in AI. The underlying idea is to shift the burden of domain knowledge from a generation component, where knowledge about the solving of complex multiple constraints is limited, to an evaluation component that can identify a possible solution as being an optimal one. An implicit assumption behind this idea is that it is easier to state domain knowledge in terms of *what* a good text should be, i.e. a discriminative model, rather than *how* a good text should be written, i.e. a generative model.

(Knight and Hatzivassiloglou, 1995, Langkilde and Knight, 1998) first introduced this novel method of generation. The motivation for building NITROGEN, their implemented system, was to increase robustness for sentence generation in the face of incomplete and possibly ill-formed input. This is a problem that they had to deal with in machine translation, their application domain. They achieve this by exploiting the power of corpus-based statistical models in place of extensive linguistic knowledge. Such models, such as *n*-grams, have enjoyed significant success in other areas of computational linguistics (Manning and Schütze, 1999).

NITROGEN consists of two main components: a symbolic generator and a statistical extractor. The symbolic generator works from underspecified input and simple lexical, morphological and grammatical knowledge bases. It is consciously designed to be simple and knowledge-poor, and hence pays no heed to linguistic decisions such as word choice, number, determinateness, agreement, tense, etc. Instead, it overgenerates many alternative utterances to convey the input, and relies heavily on the corpus knowledge of the extractor to provide the answers to these choices.

The output of the generator is a word lattice, a compact representation of multiple generation possibilities. A statistical extractor selects the most fluent path through the lattice using bigram and unigram statistics collected from two years of the Wall Street Journal.

Bangalore and Rambow (2000b,a) present a very similar approach in their system, FERGUS. The main difference is that their chosen formalism for the symbolic generation phase is TAG. This tree-based account is claimed to capture more complex linguistic features than NITROGEN's simple surface-based rules. Accordingly, they present a statistical language model based on trees for the ranking phase.

Oberlander and Brew (2000) follow Ward (1994) in arguing that NLG systems must achieve **fidelity** and **fluency** goals, where fidelity is the faithful representation of the relevant knowledge contained within the communicative goal, and fluency is the ability to do it in a natural-sounding way such that it engenders a positive evaluation of the system by the user.

It is this fluency goal that sets NLG research apart from NLU (Natural Language Understanding). An NLU system must recover meaning representations from strings of text, and whether or not a given string sounds natural, elegant, or forceful is considered far less important than identifying its propositional content.

In practice, applied NLG systems often sidestep the fluency goal by targeting a very restricted domain of output, with a limited style that may be just enough to serve the purpose of the application. Oberlander and Brew (2000) introduce two kinds of fluency goals that they argue are worth achieving from a usability engineering perspective: maximizing syntactic expectedness and maximizing user satisfaction.

To exemplify this, they stipulate two simple scenarios where NLG systems must be able to control the sentence length and the vocabulary diversity of their generated text. The difficulty is that these features are emergent, macroscopic properties of a large number of decisions, few of which are based solely on stylistic considerations. Thus it is difficult to identify and control decisions in such a way as to satisfy the requirements imposed on the text.

They propose an architecture which consists of two modules: an author and a reviewer. The author faces the task of generating a text that conveys the correct propositional content, or in short, achieving the fidelity goal. On the other hand, the reviewer must ensure that the author's output satisfies whatever macroscopic properties have been imposed on it. In other words, achieving the fluency goal. In doing so, the two modules may collaborate to produce a mutually acceptable text.

They note that Langkilde and Knight's NITROGEN system essentially embodies this architecture: the symbolic generator acts as the author, and the statistical language model is the reviewer. The difference here is that instead of collaborating, the author produces all possible 'drafts' that it considers acceptable from its point of view, and the reviewer selects the version that best meets its needs.

They show how by slightly modifying the NITROGEN model, one can implement a system that achieves the example constraints of sentence length and vocabulary diversity. The modification is to implement both the author and reviewer as different stochastic models, unlike NITROGEN where only the latter model is a stochastic model. In particular, the reviewer language model characterizes the desired distributions of sentence length and vocabulary diversity through the use of techniques such as negative binomial distributions and maximum entropy. Additionally, the word lattice from the author module is augmented with weights attached to each path, which reflects the degree of acceptability that the author assigns to the utterance represented by that path. They propose that the two stochastic models can be combined through a linear combination, where the extreme parameters would stand for systems that exclusively satisfy fidelity or fluency goals.

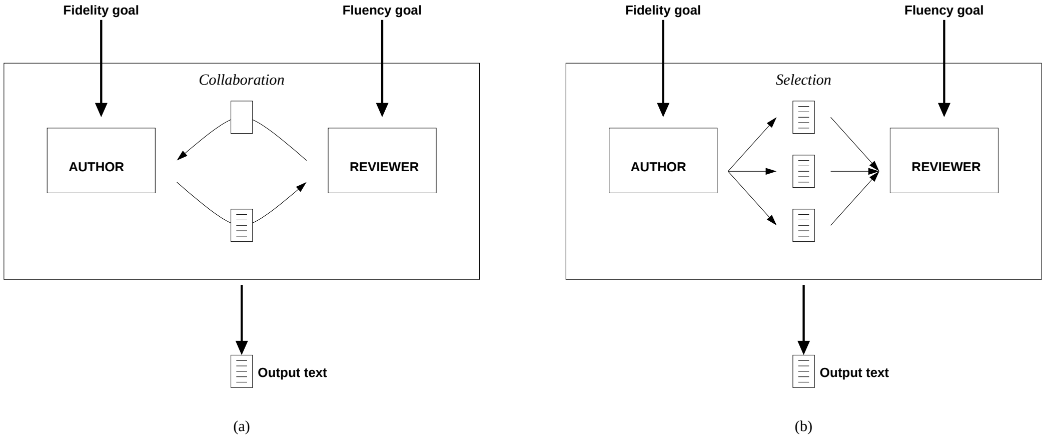


Figure 3.3: (a) The author-reviewer architecture of Oberlander and Brew (2000), and (b) as it is implemented in a NITROGEN-like system

Varges (2002) presents another overgeneration system that is inspired by NITROGEN. It is significantly different in two ways. Firstly, it adopts a "lazy learning" technique in that domain knowledge, in this case linguistic knowledge, is not compiled into a statistical model such as *n*-

grams, but rather maintained as a set of exemplar 'instances'. The candidate texts produced by the symbolic generator are evaluated by a distance metric comparing them to these instances. This technique is useful for applications where there is only a limited size corpus.

The second difference is that the phases of generation and ranking are **interleaved**. This is enabled by the use of a chart as the data structure shared between the generator and ranker, instead of a complete word lattice as in NITROGEN. As the generator incrementally adds edges to the chart, the ranker monitors the chart and, by employing the $A*$ search algorithm, an expectation-based heuristic search technique, it is able to prune the search space by removing edges that are known to be suboptimal. This can be seen as a more faithful embodiment of the author-reviewer model presented in Figure 3.3(a).

3.6 Opportunistic planning

The traditional approach to NLG as described above makes two basic assumptions: that text generation is communicative goal-driven, and that these goals dictate a top-down approach for the planning of the text's structure and decomposition of goals. However, Mellish et al. (1998b) believe that there is a class of NLG problems for which these basic assumptions do not apply.

In the case of their system, ILEX, this is due to two factors. Firstly, as ILEX is a system that produces explanation labels of museum items on display (in this case jewelry), there is often no clear plan or goal of a certain content to be conveyed. Instead, the system has a general metalevel goal, i.e. "say something interesting about this artifact, within the space available, in the context of a globally coherent discourse". Secondly, the system can not plan far in advance, as it has to generate text in real-time based on the choice of the viewer's mouse-clicks. This is a source of unexpected constraints in the selection of further items to be described.

The alternative approach they adopt is **opportunistic planning**. WordNet (Fellbaum, 1998) defines opportunity as *"a possibility due to a favorable combination of circumstances"*. Because opportunities involve combinations of circumstances, they are often unexpected, hard to predict, and too expensive, perhaps even impossible, to have complete knowledge about (Mellish et al., 1998b).

Opportunistic planning is useful when these circumstances are unknown, or are too expensive to predict, and long-term planning is impossible due to the lack of a well-defined goal, or constantly changing situations which can render such plans obsolete.

Some of the key elements of opportunistic planning are:

1. Interleaving of planning and execution: this can roughly correspond to the interleaving between the stages of content determination and surface realisation.

2. Flexible choice of tasks from an agenda.

3. Expanding "sketchy plans" as needed, taking into account the current state of the world.

4. Recognition of opportunities through detection of reference features.

In ILEX, opportunities arise when the content selection (emulating a human museum curator) realises that a particular combination of events, i.e. items of jewelry that have already been visited, allow the introduction of a new interesting fact, e.g. *"...and it was work like this which directly inspired work like the Roger Morris brooch on the stand which we looked at earlier"*.

3.7 Poetry generation from the NLG viewpoint

The main purpose of our NLG literature survey is to identify possible insights which help us in achieving our chosen task of poetry generation, which can be seen as a very specific form of NLG.

At this point, we notice that there are two distinct, though not unrelated, aspects of poetry generation that make it a unique and difficult problem within the context of NLG:

1. **Interdependent linguistic phenomena and surface constraints due to 'unity' of poetry**

   In Section 2.1.1 we introduced the concept of unity between content and form as being one of poetry's defining characteristics. This plays a crucial role in trying to analyze the task of poetry generation from an NLG viewpoint. The problem of interdependent linguistic constraints, that was identified in Section 3.2, is exacerbated by poetry's unity. For example, with regards to metre, every single linguistic decision potentially determines the success of a poem in fitting a regular defined pattern.

   This problem is made even more difficult in light of the fact that we do not possess any heuristics that can be applied universally in getting a generated text to satisfy a certain metre. In informative NLG, principles such as Centering theory (Grosz et al., 1995), Gricean maxims of quantity and quality Grice (1975), and Rhetorical Structure Theory (Mann and Thompson, 1987) have been used to guide the planning of a text's pragmatic and rhetorical structure. This provides NLG systems with a backbone with which to organize the generation process, with the possibility of allowing "repairs" to recover from problems encountered at the surface level, as in the revision/feedback architecture.

With metre, however, such principles do not apply, and it is unclear whether we can ever find similar principles, which must also be compatible with the goal of conveying a message, to motivate, for example, the inversion of syntactic structures as in Grey's *Elegy Written in a Country Churchyard* (Figure 2.2), where the adverb *'homeward'* was unusually positioned to satisfy the metre.

At this point we can already suggest that this problem indicates the need for search in poetry generation.

2. **Lack of clear, well-defined communicative goal**

Consider the task of generating Belloc's The Lion, as given in Figure 2.6. It seems unlikely that the author had, from the very outset, the communicative goal of informing the reader that the lion *"dwells in the waste"* and *"has a very small waist"*. Rather, it is the requirement of a rhyme scheme, and the fact that the words *waste* and *waist* rhyme, that led to these facts being conveyed. These could not have been deterministically arrived at from the communicative goal of, for example, *"write a cautionary poem about the ferocity of a lion, intended for children"*.

One might even argue that the goal of satisfying metre plays just as important a role in the process of generating a poem as does the goal of conveying a message.

This problem is compounded by the fact that the creation of poetry often does not even have a predefined communicative goal such as the hypothetical goal above.

Recall that in Section 2.1.4 we defined a poetry generation system as a system which generates a text which fulfills the three properties of meaningfulness, grammaticality, and poeticness. Note that this definition does not specify the realization of a given communicative goal as in the case of conventional NLG. In particular, the requirement of the generated poem being meaningful makes no assumption as to where the meaning originates from.

On one hand, it may be specified externally of the system, as in the case of requesting the system to generate a poem based on a vague concept, e.g.*“a poem about betrayal”*, or a more well-defined message, e.g.*“a descriptive poem about the railway bridge over the Tay river of Dundee”*. In the first case, this would be similar to the goal of ILEX, where the goal is essentially to “say something” about an object in the knowledge base. The latter case would be more analogous to conventional NLG.

However, we contend that poetry generation can occur without any predefined message whatsoever. In this case, it is up to the generator to construct a message based on its knowledge sources. Although it is plausible to imagine a system that employs notions of interestingness, novelty, and believability in its construction of a message to be conveyed, it is important to note that poetic constraints, such as those of metre and rhyme, play an equally crucial role in determining the content.

Thus, we can make the observation that the paradigm of NLG as communicative goal-driven process is incongruous with the poetry generation process.

Of the NLG systems we have looked at, several of them do address these two problems, although not explicitly for the purposes of poetry generation. With respect to the first problem, the alternative architectures, search methods and overgeneration approach were born out of a need to overcome similar issues. Perhaps Kamal (2002) is most similar in the sense that surface constraints are the problems being addressed. With respect to the second problem, we find the same situation arising in ILEX, which led them to employ opportunistic planning.

It is worth noting that the ILEX task resembles our task of poetry generation in that it is trying to satisfy various linguistic preferences of which there exist no formal definitions, only informal heuristic knowledge. While Mellish et al. (1998a) advocate the stochastic search approach they adopt, pointing to their results as showing promise, they strongly emphasize that the results are only as good as the evaluation function, which was constructed in an ad-hoc manner.

3.8 Summary

In this chapter we have seen the conventional paradigm of NLG as communicative goal-driven planning, which consists of a number of different, but interdependent, tasks. Due to this interdependency, the usual approach of tackling them in isolation has its limitations. To try and overcome these limitations, several different approaches have been taken, i.e. using alternative architectures, better search techniques, and overgeneration methods. Furthermore, works such as Mellish et al. (1998b) explore the use of opportunistic planning for problems where there is no clear communicative goal.

We have seen that many of the issues these different approaches try to tackle are also inherent in, and exacerbated by, poetry. In particular, the unity of poetry creates an even stronger interdependency between the various linguistic decisions to be taken, and the lack of a well-defined communicative goal in poetry generation renders it incongruous to the conventional paradigm.

Thus, with respect to our research task of poetry generation, we can conclude the following points from our survey of the NLG literature:

1. With regards to input, it is not clear whether the 4-tuple $< k, c, u, d >$ mentioned in Section 3.1.1 is required for poetry generation. In particular, d, the discourse model, is unnecessary due to poetry being a single piece of text which the reader does not engage in conversation with.

2. The strong interdependency between the various levels of linguistic representation in poetry renders conventional NLG system organisation (Section 3.2) unsuitable for our task. It is difficult to clearly decompose the system into separate modules which each only handle, for example, meaningfulness and poeticness. This is analogous to the "generation gap" problem addressed by work on alternative architectures (Section 3.3). It is also one of the limitations that we pointed out existed in ASPERA and COLIBRI (Section 2.3.4). We feel the **integrated architecture** shows promise for our task, as it embodies a problem-solving model of constraint satisfaction, which is essentially what our definition of poetry specifies (Section 2.1.4).

3. Following on from the previous point, solving multiple-constraint satisfaction problems requires us to view poetry generation as **search**. In Section 3.4 we reviewed work that applied well-known AI search techniques in NLG. Although, it is far from trivial to decide which of these search techniques is best for our poetry generation task, we feel that **stochastic search methods** are suitable for the following reasons:

    (a) They are well-known, well-studied, general purpose algorithms that can be immediately applicable to our task.

    (b) They are very suitable for problems where the domain knowledge describes *what* a solution should be rather than *how* it can be obtained (see point 4 on discriminative models below).

(c) The specific stochastic search method known as the **evolutionary algorithm** is believed to be a widely used approach in computationally “creative” systems (see Section 2.3.3).

4. We feel that the **discriminative model** of generation found in overgeneration and ranking methods (Section 3.5) is very applicable to our task. Our knowledge of poetry (Section 2.1.2), and our decision to restrict our attention to concretely observable aspects of poetry (Section 2.1.4), means that our domain knowledge is of *what* a poem should look like, and not *how* it is constructed. Stochastic search methods such as evolutionary algorithms are perfectly suited for problems with this type of domain knowledge.

5. As discussed in Section 3.7), the lack of a well-defined communicative goal in poetry generation renders it incongruous to the conventional NLG paradigm. **Opportunistic planning** has been found to be useful in such situations (Section 3.6), and thus seems suitable for our task.

In the next chapter we will introduce our model of evolutionary algorithm-based poetry generation, which is influenced by the NLG approaches and techniques we have listed above.

Chapter 4

Evolutionary algorithms for poetry generation

After defining the target domain of poetry in Chapter 2 and providing an overview of the literature in natural language generation in Chapter 3, in this chapter we will propose the central concept of our work, which is to implement our model of poetry generation as a state space search problem, and to solve it with an evolutionary algorithm (EA). We will also describe the characteristics of our EA framework for poetry generation, and NLG in general, namely the design of representational scheme, genetic operators and evaluation functions.

4.1 Poetry generation as stochastic search

In the previous chapter, we have shown that the process of poetry generation does not fit well with the prevailing paradigm of NLG as a communicative goal-driven, top-down planning process.

We will now introduce one of the central concepts in this thesis, and that is to model the poetry generation process as a state space search problem, where a state in the search space is a possible text with all its underlying representation, from semantics all the way down to phonetics. A goal state is a text that satisfies the three requirements of meaningfulness, grammaticality, and poeticness.

Figure 4.1 shows an example of an idealization of poetry generation as a state space search problem. Imagine that the goal of the task is to produce an iambic pentameter couplet that describes the act of walking and sleeping by a person named John. This goal is reflected by the “target semantics” and “target surface” that is given as input to the system. This target information could be used to create an initial, or start, state, e.g.*“John walked. John slept”*, as shown in the first state in Figure 4.1. From here, various other states can be reached by applying a wide range of operators that can occur at any level of representation. Consider the case where the step taken is one of adding detail, in this case a destination for the act of walking, e.g.*“to the store”*. Thus we reach the next state in the figure. If we follow the sequence of operations taken, eventually we might arrive at a state that represents the output text *“Into the bookshop John did slowly creep, inside he fell into a peaceful sleep.”*. This state is a goal state as it satisfies the requirements imposed by the targets.

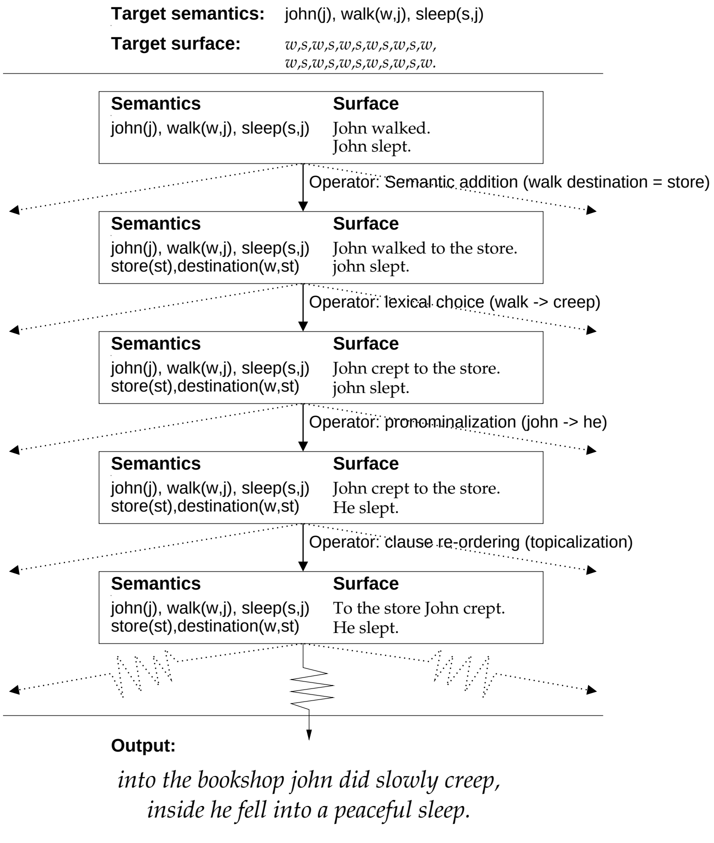

Target semantics: john(j), walk(w,j), sleep(s,j)
Target surface: w,s,w,s,w,s,w,s,w,s,w, w,s,w,s,w,s,w,s,w,s,w.
Semantics
john(j), walk(w,j), sleep(s,j)
Surface
John walked.
John slept.
Operator: Semantic addition (walk destination = store)
Semantics
john(j), walk(w,j), sleep(s,j)
store(st),destination(w,st)
Surface
John walked to the store.
john slept.
Operator: lexical choice (walk -> creep)
Semantics
john(j), walk(w,j), sleep(s,j)
store(st),destination(w,st)
Surface
John crept to the store.
john slept.
Operator: pronominalization (john -> he)
Semantics
john(j), walk(w,j), sleep(s,j)
store(st),destination(w,st)
Surface
John crept to the store.
He slept.
Operator: clause re-ordering (topicalization)
Semantics
john(j), walk(w,j), sleep(s,j)
store(st),destination(w,st)
Surface
To the store John crept.
He slept.
Output:
into the bookshop john did slowly creep,
inside he fell into a peaceful sleep.

Figure 4.1: An idealization of poetry generation as state space search

Such a search space is undoubtedly immense, and in fact when one considers the nature of recursive structures in natural language, it is infinite. Our proposed solution is to employ a form of stochastic search. Stochastic search is a form of heuristic search that strongly relies on random traversal of a search space with a bias towards more promising solutions. It has become increasingly popular for solving computationally hard combinatorial problems from various AI domains such as planning, scheduling, and constraint satisfaction, and has been shown to outperform deterministic approaches in a number of domains.

The particular search methodology that we use is the **evolutionary algorithm**, which is essentially an iteration of two phases, **evaluation** and **evolution**, applied to an ordered set (the **population**) of candidate solutions (the **individuals**). We will discuss evolutionary algorithms in more detail in Section 4.2.

This model is desirable owing to the following aspects:

1. **Solving of multiple, interdependent constraints**

   The multiple constraints of meaningfulness, grammaticality, and poeticness that are imposed on different levels of linguistic representation make it extremely difficult to organize and structure a system for producing a poem. Just as the NLG systems described in Section 3.4 employed search methods such as chart generation, truth maintenance systems, and evolutionary algorithms to overcome the “generation gap” problem, our adoption of a state space search formalism provides us with a very powerful mechanism to satisfy these constraints.

   An important point to note is that our model embodies an integrated architecture, due to the fact that a move in the search space can occur at any level of representation of the text, and in most cases will affect several levels. Thus, we abandon the conventional decomposition of the generation process into content determination, text planning, and surface realisation. Kantrowitz and Bates (1992) argue that this is necessary for handling the many interdependent linguistic phenomena, as is exhibited by poetry.

2. **Allows declarative as opposed to procedural account of poetry**

   With heuristic search methods, of which stochastic search is an instance, the encoding of domain knowledge is situated within the evaluation function, which measures the desirability of a state in the search space as a possible solution to the problem at hand.

   This enables us to concentrate on specifying **what** we seek to produce, i.e. text artifacts that are poems, as opposed to **how** to create them. In other words, we are developing a declarative account of poetry.

This is closely related to the previous point, in that there is difficulty in designing a procedural alogrithm for the satisfaction of the multiple, interdependent constraints that are inherent in poems.

Theoretically, any domain knowledge that represents the characteristics of poetry can be specified independently and encoded into the evaluation function, adding informedness to the heuristic search. However, this raises the issue of how one defines the relative importance of the contributions that each heuristic makes to the whole evaluation function. This is the problem of multi-objective optimization (see Sections 4.2.2, 4.4.3).

3. **Facilitates opportunistic planning**

   In the previous chapter we identified two major difficulties of poetry generation as viewed from an NLG perspective, the second one being the lack of a clearly defined communicative goal that can be used to drive the process of planning the resulting poem.

   In this respect, our task faces the similar problems encountered in ILEX, where they choose to adopt the paradigm of opportunistic planning (Section 3.6). Indeed, Mellish et al. (1998b) suggest that there is a class of NLG problems for which the paradigm of communicative goal-driven planning does not fit well. We believe poetry generation is one of those problems.

   In ILEX, opportunities arise due to a combination of events, primarily based on the sequence of museum objects the user has chosen to visit. This enables the system to, for example, select content that relates to previously visited objects.

   During the course of generating poetry, opportunities can arise due to a combination of partially-constructed content and form that, in an attempt to achieve some effect of a phonetic, rhetoric or semantic nature, allows the introduction of a new content or form element. For example, one can imagine something akin to the common poetry-workshop game where partially constructed poems are given, and as an exercise, writers are asked to fill in the blanks such as in the following couplet[1]:

   When you talk to a monkey he seems very wise
   He scratches his head and he . . .

[1] The original poem by Rowena Bennett (Daly, 1984) ends the line with *"blinks both his eyes"*.

The interleaving of planning and execution in opportunistic planning can be approximated through the iterative phases of evolution and evaluation in our chosen stochastic search method of evolutionary algorithms. However, an important aspect of opportunistic planning that differentiates it from an ordinary stochastic search is the fact that during the stages of execution, opportunities are both recognized and exploited. In evolutionary algorithms, this is not intentionally achieved, but merely happens as a result of bias towards more promising solutions. One can envisage, however, search operators that do exactly this. We will attempt to explore this route by experimenting with "smart" operators that implement heuristics that seek to make use of goals and opportunities (Section 4.5.2).

Another aspect that we will explore is the issue of granularity in the interleaving of stages. In ILEX, the opportunistic planning process was still structured around the two-tiered architecture of content determination and text planning. However, due to the strong unity between content and form in poetry (see Section 2.1.1), we feel that an even finer grain of interleaving is required. We seek to experiment with operators of differing levels of granularity to test this hypothesis.

It is worth noting that the interleaving of planning and execution in opportunistic planning can be seen as somewhat analogous to Sharples' model of writing as a process of creative design (Section 2.2), where the process of creating a text is an iterative, interleaved process of reflection and engagement. It also reflects the model of poetry writing suggested in Levy (2001), who suggests that evolution is the prevailing metaphor in creativity theory (Section 2.3.4).

4.2 Evolutionary algorithms

In this section we introduce **evolutionary algorithms**, our chosen method of stochastic search, along with the crucial issues of representational schemes and the implementation of constraints in an evolutionary algorithm, as this underlies several NLG-specific issues that we will encounter in later sections.

As defined in Michalewicz (1996), an evolutionary algorithm, or EA, is a multi-point stochastic search algorithm, which means that it is a form of heuristic search that simultaneously explores several points in a search space, and navigates the search space stochastically so as to prevent getting trapped in local maxima as straightforward hillclimbing algorithms do.

All EAs maintain a **population** of **individuals** over time t as follows:

$$P(t) = \{{x_i}^t, \dots, {x_n}^t\}$$

Each individual x represents a potential solution to the problem at hand, and is implemented as some possibly complex data structure.

The main algorithm of an EA is as follows:

1. **Initialization**: For $t = 0$, construct a new population $P(t) = \{{x_i}^t, \dots, {x_n}^t\}$, which represents a set of starting points to explore the search space. Ideally, the distribution of points is evenly spread out across the space.
2. **Evaluation**: Each solution ${x_i}^t$ is evaluated to give some measure of its “fitness”.
3. **Selection**: A new population $P(t+1)$ is formed by stochastically selecting individuals from $P(t)$, usually with a bias towards fitter individuals.
4. **Evolution**: Some members of the new population undergo transformation by means of “genetic” operators to form new solutions.
5. repeat steps 2 to 4 until termination. Termination is achieved either after
   (a) a given number of iterations has elapsed, or
   (b) a given fitness score has been achieved, or
   (c) the EA has converged to a near-optimal solution, i.e. it has not yielded any better solution for a given number of iterations.

Upon termination of this algorithm, the fittest individual is hoped to be an optimal, or near-optimal, solution.

We will now elaborate on each of the steps above, insofar as it is relevant to our purposes and interests.

4.2.1 Initialization

As mentioned above, during the initialization phase a population of individuals is created, where each individual represents a starting point to begin exploring the search space.

How initialization is performed varies across EA systems. Davis and Steenstrup (1987) suggest that purely random initialization is useful for research purposes, as evolving a well-adapted solution from a randomly created population is a good test of the EA. The fitness of the resulting solution will have been produced by the search, as opposed to being pre-programmed at the initialization. More practical applications, however, would benefit from a more directed approach.

If the random initialization approach is adopted, it is important to bear in mind the issues of implementing constraints (see Section 4.2.7 below), as it is possible that any ill-formed individuals that are created during initialization may propagate throughout the evolution. Hence, randomness here should be qualified as being randomness within the inviolable constraints that the problem domain specifies.

4.2.2 Evaluation

During the evaluation phase, a **fitness function** is applied to each individual, yielding a numerical score that indicates its suitability as a solution to the problem at hand.

A fitness function can actually be implemented as an aggregation of a set of separate functions, which we will call **evaluators**, each of which is designed to measure a different property, or aspect, of an individual.

This **multi-objective** nature introduces a problem of how to integrate all the scores returned by each evaluator. One can aggregate them in a linear combination, but determining the optimal weight of the contribution of each evaluator is by no means straightforward.

Other methods such as non-Pareto population based and Pareto-optimality based multi-objective optimization have been investigated (Schaffer, 1985, Fonseca and Fleming, 1995).

4.2.3 Selection

During the selection phase, a number of individuals are chosen from the current population whose offspring will be present in the next generation. This is the phase in which the driving force in Darwin's theory of evolution, natural selection, is simulated.

There are many variations of selection algorithms, such as proportionate, tournament, ranking, and Boltzmann selection (Goldberg, 1989, Bäck et al., 1997). The main property shared between all of them is the fact that selection is a function of an individual's fitness (Mitchell, 1996). Where they mainly vary is in the stochastic bias towards good individuals, where one extreme, nonregressional evolution, possibly leads to premature convergence, and the other extreme towards inefficient searching through unpromising regions of the space.

John Holland's seminal work on genetic algorithms (Holland, 1975) introduces the notion of intrinsic parallelism, which states that individuals are sampled in such a way that schemata from this space are represented in succeeding generations in proportion to their observed fitness relative to the population average. Schaffer (1987) shows that this property holds for different variations on the selection algorithm. In essence, there exists a degree of freedom of choice for the selection algorithm, as whichever is chosen, the principles of evolution will still apply.

4.2.4 Evolution

An EA has a set of "genetic operators", which are functions to modify a solution, yielding a different solution, i.e. to cause a move in the search space.

These operators are either unary functions m_i ("**mutators**"), i.e. $m_i : S \to S$ that randomly perturb the properties of a solution, or higher arity functions c_i ("**crossovers**"), i.e. $c_i : S \times \ldots \times S \to S$, that attempt to combine the properties of two or more solutions. The domain S is the set of all possible individuals, or candidate solutions.

Spears (1992) notes that historically, proponents of the Holland (1975) style of genetic algorithm tend to believe that crossover is more poweful than mutation, whereas researchers using **evolution strategies**, another type of evolutionary algorithms, view mutation as the key genetic operator. The results in Spears (1992) show that both operators have their strengths and weaknesses, that are linked to the issues of **exploitation** vs. **exploration**: mutation is best used to create random diversity in a population, while crossover serves as an accelerator that promotes emergent behaviour from partial solutions. The balance between these two processes is dependent on the specific application, and even implementation, of the EA.

4.2.5 Island model

An **island model** is an evolutionary algorithm where the population is segmented into groups called islands. Each island still behaves just like a population, and each island is isolated from each other. For example, a maximum total population of N_{total} could be spread across M islands, each island having a population size of $N_{island} = N_{total}/M$.

Such models have been reported to display better search performance than large single population models, both in terms of solution quality and reduced effort (Whitley et al., 1997).

One reason for this is that the islands develop relatively independently, allowing exploration of different regions of the search space without prematurely converging.

On the other hand, a certain degree of information sharing between islands is facilitated through **migration**, a process where islands exchange a portion of their population. This introduces the parameters $I_{migration}$ (migration interval), the number of generations between a migration, and $N_{migration}$ (migration size), the number of individuals in an island which migrate.

4.2.6 Representation

The choice of representational scheme is a key issue in all flavours of evolutionary algorithm. As in other areas of AI, there is often a trade-off between expressibility, the ability of a scheme to express each point in the search space, and tractability, the degree to which the scheme facilitates the application of operations that are inherently useful and meaningful to the problem domain.

Although there may be many ways of representing a given search space, certain schemes are more ‘natural’ than others. In particular, a good representational scheme would maintain a correspondence between the measure of proximity between two points in the search space as defined by the scheme and the proximity between the candidate solutions they encode. This is desirable as an EA is essentially a form of stochastic hillclimbing search, where a search localized around a near-optimal solution is hoped to yield similarly good results.

We will now discuss common representational schemes employed in evolutionary algorithms, but first we elaborate on an important distinction derived from neo-Darwinism, that of the genotype and the phenotype.

Genotype vs. Phenotype

In the biological sense of evolution, there is a clear distinction between an organism’s underlying genetic coding (its genotype), and the behavioural, physiological and morphological manifestation, or response, of that genetic information (its phenotype) (Fogel, 1995).

In particular, the neo-Darwinian theory of evolution suggests that natural selection occurs on the phenotypic level, in that the singular measure of evolutionary fitness is the appropriateness of a species' behaviour and physiology in terms of its ability to anticipate its environment.

On the other hand, the indirect effect of natural selection and environmental factors determines which genetic structures survive and which do not. This indirect force, coupled with genetic mutation and replicative errors, is what shapes and evolves the genotypic level over time.

This distinction is important when one considers the pleiotropic and polygenic nature of the genotype. Pleiotropy is the effect that a single gene may simultaneously affect several phenotypic traits. Polygeny is the effect that a single phenotypic characteristic may be determined by the simultaneous interaction of many genes.

Natural selection is applied to a complex integration of phenotypic characteristics, not to an individual trait. However, certain reductionist views of evolution apply fitness measures, or the process of natural selection itself, to individual genes, i.e. elements at the genotypic level. This simplification blurs the distinction between the genotype and the phenotype, and essentially ignores such complex interactions as pleiotropy and polygeny.

How does this affect the computational modelling of evolution in EAs? While one may argue that evolution as it happens in nature merely serves as an inspiration for computational models, and that there is no strict requirement for adherence to this distinction when implementing EAs, the very nature of the complex problem domains that are commonly solved with evolutionary techniques usually exhibits properties of pleiotropy and polygeny.

With respect to our specific task of poetry generation, the fitness of a solution in our model is how well it satisfies the properties of meaningfulness, grammaticality, and poeticness (Section 2.1.4). The features in our chosen representation which can be used to measure this fitness correspond to the phenotypic level. However, such features may not necessarily be the actual structures of genetic code which are mutated and evolved, i.e. the genotypic level. Indeed, the complex interactions between different linguistic representational levels (Section 3.7) suggests the existence of pleiotropy and polygeny.

Although in practice the required features for measuring fitness may be deterministically recovered from the underlying representation through some standard calculation, we believe this is still an important theoretical distinction to be made: fitness is measured based on some explicit features which are different from the actual structures being genetically mutated and reproduced. In McGonagall, the structures being manipulated are LTAG trees (Section 5.3.1), whereas the features being measured are the conveyed propositional semantics (Section 5.3.8) and lexical stress patterns (Section 6.3.2). Both these features are deterministically recoverable from the LTAG structures.

String representations

Holland (1975) introduced genetic algorithms using representations of strings of characters in a binary alphabet (bit-strings). This scheme was chosen for simplicity, tractability, and a natural resemblance to the chromosomes in living organisms. Unfortunately, a lot of subsequent work has tended to take this scheme canonically. While this scheme is appealing due to its relative simplicity and well-studied properties, a complex problem would often require a lot of ingenuity in devising such a representation.

Moreover, most work with GAs tended to ignore the genotype / phenotype distinction by calculating fitness functions directly on the string representations.

In recent years, “hybrid” solutions have experimented with different data structures such as lists, trees, and graphs. We will now look at trees, and in particular how they are used in the area of genetic programming.

Genetic programming and S-expression trees

Genetic programming is an extension of the conventional genetic algorithm in which each individual in the population is a computer program. Koza (1994) states that the aim is to enable computers to learn to solve problems without being explicitly programmed.

Since functional computer programs can be viewed as nested applications of functions (operations) to arguments (values), an individual is represented as an expression tree that represents this sequence.

The most commonly used formalism is the LISP S-expression tree. Genetic mutation operations are implemented by adding, removing, or altering nodes in the tree, and crossover is achieved through the swapping of two subtrees, each belonging to a parent individual’s expression tree.

As we will see in Section 5.3, this bears close resemblance to natural language parse trees, and would seem the obvious choice for representation when evolving natural language texts.

4.2.7 Implementation of constraints

A candidate solution for a computational problem will likely have to obey certain constraints of well-formedness, particularly if the solution involves complex data structures. As we will see in Chapter 5, natural language generation exhibits these properties. When trying to arrive at such a solution via an EA, these constraints are implemented through a combination of design issues of representation, evaluation functions and genetic operators.

Davis and Steenstrup (1987) suggest two ways that inviolable constraints can be implemented, which basically differ in the representational issue of whether invalid solutions are allowed to be present in the search space:

1. Imposing penalties on individuals that violate a constraint

   Although this has the desirable property of guiding an EA towards evolving solutions that do not violate a constraint, this is quite an ad-hoc solution, as the property of being inviolable is no longer upheld. Rather, the constraint is relaxed to become more of a violable preference.

   Davis and Steenstrup (1987) actually further divide this mechanism into imposing either "heavy" or "moderate" penalties, but we see them as essentially being the same. However, it is a useful distinction to be made, as it highlights the different problems that an EA runs into at both ends of the spectrum, and stresses the delicate nature of finding the right balance. The potential problems with either choice are:

   (a) if imposing heavy penalties:

       When an EA has a high probability of evolving solutions that violate constraints, then 1) the EA will very likely waste lots of computation wading through ill-formed solutions, and 2) once a well-behaved solution is found, it is likely that the EA would quickly converge on that solution, causing a premature convergence.

   (b) if imposing moderate penalties:

       An EA may continue to evolve solutions that violate constraints but still score better than those that do not, because the rest of the evaluation function can be satisfied better by accepting the violation penalty rather than avoiding it.

2. Ensuring that all possibly evolvable solutions never violate the constraint

   This is perhaps the most principled approach. It entails designing both the initialization phase and the genetic operators so as to always guarantee that individuals in the population will satisfy the constraint, or at least its genotype can be interpreted as a phenotype that satisfies the constraint.

   Unfortunately, it is also usually the most computationally intensive approach.

Choosing which of the above methods to use in implementing constraints is a non-trivial task, and is very task-specific. It will typically involve empirical testing to see which method yields the best results for the least computational effort.

Since each constraint can be handled differently, it is very likely that a system will end up using a combination of these methods, whichever suits a particular constraint best.

In Sections 4.3 to 4.5, we will now decide on these issues of EA design, particularly implementation of constraints, for the purposes of poetry generation. Note that this discussion will be of a broad, theoretical nature. We will revisit these issues in more detail when we describe McGonagall, our implemented system, in Chapters 5 to 7.

4.3 Linguistic representation

The choice of representational scheme for a state space search is crucial in determining the efficiency and effectiveness of the search in finding an optimal solution to the domain problem.

For the sake of being able to reason with the multiple, interdependent constraints encountered in poetry, we define the search space as the space of all possible texts, with all underlying representations being simultaneously accessible, from semantics down to phonetics.

The issue of implementing constraints is one that has an important bearing on the design of an EA's representational scheme. We have seen in Section 4.2.7 that there are essentially two ways to implement constraints in an EA:

1. Ensure candidates do not violate constraints. This is achieved through design of genetic operators.
2. Allow constraints to be violated, but penalize and/or reward each candidate solution with respect to their satisfying of the constraints. This is achieved through the design of the evaluators.

Either approach imposes requirements on the representation scheme.

Since the goal of our generation system is to create texts that satisfy the constraints of meaningfulness, grammaticality, and poeticness, we must make a decision on how they will be handled.

Choosing to ensure that all three constraints are not violated by a candidate solution is not a profitable approach. This would require the system to behave like a deterministic system that uses sophisticated procedural machinery of transforming the linguistic representation into valid solutions. It is precisely this sophisticated machinery that we lack, and forms the root of the problems mentioned in Section 3.7.

On the other hand, choosing to allow all three constraints to be violated and having a purely discriminative model that quantitatively evaluates a candidate solution for its well-formedness is inefficient.

Thus we can see a spectrum of search sophistication between these two extremes, and it is roughly along this spectrum that the distinction between pipelined and integrated architectures, first noted in Section 3.4, lies.

Upon further inspection, we can observe that our three constraints are not orthogonal. In particular, it can be argued that the satisfying of the grammaticality constraint is a prerequisite for the satisfying of meaningfulness and poeticness. The widely accepted theory of compositionality of semantics states that the meaning of an utterance is derived from the analysis of the utterance as being composed through well-formed syntactic operations. We believe that grammaticality is also an imperative precondition for poeticness to be considered. For example, we do not regard the simple concatenation of words that yielded the metrical pattern of a limerick, as output from the experimental POEVOLVE testbed implementation, as a poem.

Thus grammaticality is the prime constraint that must first and foremost be satisfied, and we decide to implement it as an inviolable constraint through the design of genetic operators.

The mechanisms by which we can ensure grammaticality are very well studied, and we can call on the immense resource of work done in the field of **computational linguistics**. It also brings our work more in line with conventional NLG, and particularly with search-based approaches such as Stone and Doran (1997) and Eddy (2002).

Finally, we note that most modern representations of syntax use tree-like structures, and the manipulation of such structures in evolutionary algorithms is also well studied in the field of **genetic programming** (Section 4.2.6).

4.3.1 Enforcing grammaticality through representation and operators

Given the complexity of modern computational linguistics theories in trying to capture the various phenomena of natural language, one can expect that an adequate representation for our purposes will have a large number of constraints on syntactic well-formedness.

Therefore, we choose to take the approach of stipulating that all possibly evolved solutions will never violate grammatical constraints, as the alternative method of imposing penalties works best in situations when the probability of evolving solutions that violate constraints is small.

We implement this approach by maintaining two properties:

1. All initial candidate solutions created during the initialization phase must be well-formed.
2. During any stage of subsequent evolution, if a candidate solution is well-formed, then all possible modifications on that candidate solution must retain its well-formedness as well.

One obvious way of doing this is to fashion the genetic operators after legal operations in a chosen grammar formalism. Grammar formalisms are formal representational schemes for natural language phenomena. Formalisms that efficiently and elegantly capture human linguistic phenomena can be very complex, due to the complex and ambiguous nature of human language usage. Essentially, a formalism defines a set of symbols that can represent linguistic artifacts, and a set of functions on those symbols that can be used to construct well-formed structures that can represent even more complex linguistic artifacts.

A **grammar**, defined under a certain grammar formalism, is a set of rules that defines, for a particular language, which operations on which symbols result in well-formed structures, i.e. grammatical utterances.

For a good introduction to this field and its related issues, we refer the reader to Jurafsky and Martin (2000).

Many modern formalisms tend to be feature-structure based and lexicalist. Examples of these are head-driven phrase structure grammar (Pollard and Sag, 1994), lexicalized tree adjoining grammar (Joshi and Schabes, 1997), and combinatorial categorial grammar (Steedman, 1996). To a certain degree, one can argue that any of these various formalisms would suit our purposes. However, instead of arbitrarily choosing any widely-used grammatical formalism as the basis of our representation, we will first consider what special requirements our chosen framework places upon our representational scheme.

We require a representational scheme that:

1. simultaneously encapsulates all levels of grammatical information, which includes semantics, rhetoric, syntax, lexis, and phonetics, and

2. elegantly allows for incremental and non-monotonic structure building.

The formalism we have chosen for our implemented system, McGonagall, is Lexicalized Tree Adjoining Grammar, or LTAG, and in Section 5.3 we discuss how LTAG fulfills these requirements. Note that although we have chosen LTAG for our implementation, we do not rule out other formalisms within the context of our theoretical model.

The choice of enforcing grammaticality through design of representation and operators promotes the syntactic structure to the status of primary data structure for the representation of a candidate solution, from which we can derive all other information, i.e. semantic, lexical, and phonetic.

4.3.2 Optimizing meaningfulness and poeticness: a trade-off

Unlike the constraint of grammaticality, the remaining two constraints of meaningfulness and poeticness will be implemented as penalties through the evaluation functions. These are the main features that will be optimized by the EA.

The constraints of meaningfulness and poeticness are roughly analogous to the concepts of fidelity and fluency discussed in Section 3.5 regarding overgeneration and ranking methods for NLG. Meaningfulness and fidelity are both concerned with how a text achieves its intended communicative goal. Regarding poeticness and fluency, although they are different in the sense that poeticness, and specifically our restricted definition of it, is concerned with how a text satisfies a rhythmic stress pattern, and fluency is how ‘natural-sounding’ a text is, they are similar in the sense that they both entail surface constraints that form a trade-off with the goal of meaningfulness/fidelity.

We have observed that the way this trade-off is dealt with in the various generation systems we have seen is markedly different between conventional NLG systems and existing poetry generation systems.

In the majority of conventional NLG systems discussed in Chapter 3, fidelity is the primary constraint, and fluency is treated more as an additional constraint that is used to prefer one text over another. This is understandable, considering that the main goal of NLG is the realization of the given communicative goal. This can be seen implicitly from the way traditionally decomposed NLG systems make semantic decisions before surface decisions, and also how in overgeneration systems fidelity is first handled during the generation phase, and fluency is achieved through the selection of the most probable candidate. In short, **fidelity/meaningfulness precedes fluency/poeticness**.

However, the reverse seems to hold in the few poetry generation systems that we discussed in Chapter 2. WASP and ASPERA, for instance, clearly place metrical well-formedness as the main constraint on their output text, as do RKCP, the POEVOLVE limerick prototype, and all other form-aware poetry generation systems. In other words, **fluency/poeticness precedes fidelity/meaningfulness**.

We believe that for poetry generation, there is in theory no justification for either of these constraints taking precedence over another. Our model therefore makes no such assumptions, and treats both meaningfulness and poeticness as two separate measures that are to be optimized in their own right. This raises the question of how these measures are combined (Section 4.4.3).

However, it must also be pointed out that handling meaningfulness and poeticness through the evaluation functions is not the whole story. There is also the possibility of encoding them in the genetic operators, over and above the enforcement of grammaticality, and we discuss these ‘smart’ operators in Section 4.5.2.

4.4 Designing Evaluators

Arguably the most crucial aspect of a stochastic search is the evaluation scheme which informs the system how desirable a solution is. We believe there are three major issues to address for designing our evaluation functions: **identifying** the features to be evaluated, **quantifying** these features as numerical scores, and **combining** these scores in a meaningful way.

4.4.1 Identification of features to be evaluated

The most elementary question to be asked in structuring our system as an optimization problem is: "*What do we optimize?*" We have already determined that our search process will be optimizing some properties that will lead to the satisfaction of meaningfulness and poeticness constraints. What exactly are these properties?

1. **Meaningfulness**

   Evaluating the quality of a generated text's semantics is a challenging task for the generation community. Mellish and Dale (1998) term this process **accuracy evaluation**, and discuss how most existing work on this is external of the NLG system and involves human judgment. It is understandable that few NLG systems incorporate the sort of self-evaluation of semantics that we require, as the conveying of semantics is the whole goal of the NLG process, and thus NLG systems are usually developed explicitly towards maximizing accuracy in the first place. Overgeneration systems, particularly Varges (2002), are perhaps the most similar systems to ours in this respect.

   The existence of a given set of semantic propositions to be conveyed, call it the **target semantics**, gives us a clear feature to measure: how well does a candidate solution convey the target semantics? In other words, how **faithful**, or similar, is the semantics of a candidate solution compared to the target semantics? Related work on this is the **preference-based approximate generation** in Nicolov (1998), which provides a good model for reasoning about how a text can convey more or less than the target semantics, and how to prefer one text over another based on how similar its semantic representation is to the target semantics. The method Nicolov defines is very specifically couched in terms of conceptual graphs (Sowa, 1984), but his intuitive notion of such a measure is essentially the same as our own model of semantic similarity (Section 6.4).

However, what if the target semantics is so vague as to be insufficient as a benchmark for comparing the candidate solution with? For example, given a communicative goal such as "write a poem about a lion", e.g. as the target semantics $lion(X)$, what measure can be used? In cases such as this, it would be better to view the target semantics not as the message to be conveyed, as is the case in conventional informative NLG, but rather as a 'pool' of ideas from which the system can draw inspiration. In cases such as this, we propose devising measures of how **internally consistent** and how **domain consistent** the semantics of a candidate solution is. In the first case, this would measure how the semantics does not contradict itself, and in the latter case, how the semantics is meaningful with respect to some underlying knowledge base. One could also speculate on evaluating some notion of how **interesting** or **novel** a candidate solution's semantics is, but this is more related to the field of creativity theory (Pease et al., 2001, Ritchie, 2001, Wiggins, 2001) and story generation (Turner, 1994, Bailey, 1999, Pérez y Pérez and Sharples, 1999), which is beyond the scope of this thesis.

2. **Poeticness**

   (a) **Phonetics**: One of the most obvious things to look for in a poem is the presence of a regular phonetic form, i.e. rhyme, metre, alliteration, etc. The information needed to detect this can be derived from a pronunciation dictionary.

   Similar to meaningfulness, one possible evaluation method is to specify a **target form** as input, i.e. the ideal phonetic form that a candidate solution should possess, and to then score a candidate solution based on how closely it matches this target form.

   For example, we could provide the system with the following target form (here `w` means a syllable with weak stress, `s` a syllable with strong stress, and `(a)` and `(b)` would determine the rhyme scheme, e.g. *aabba*), which effectively means we are requesting it to generate a limerick:

```
w,s,w,w,s,w,w,s(a)
w,s,w,w,s,w,w,s(a)
   w,s,w,w,s(b)
   w,s,w,w,s(b)
w,s,w,w,s,w,w,s(a)
```

   Alternatively, we could specify a set of these target forms, e.g. limerick, sonnet, quintain, haiku, rondeau, sestina, etc., essentially creating a knowledge base of existing poetry forms. The evaluation function would then reward candidate solutions that are found gravitating towards one of those patterns. Although a more flexible alternative, it would probably not be as informed a heuristic, as the definition of the goal becomes less focussed.

Finally, there is the possibility of detecting regularities in the rhythmic stress pattern of a candidate solution without the aid of knowledge of existing forms. The suffix tree-based methods used in computational biology and genomics (Pedersen, 1999) for detecting regular sequences in DNA are a possible technique.

(b) **Figurative language**: Aside from phonetic patterns, there are other, more subtle, features to look for in a poem, such as lexical choice, where the evaluation could reward the usage of interesting collocations and words marked as “poetic”, syntax, where reward could be given to usage of interesting syntactic constructs, e.g. inverse word and clause order, and rhetoric, where evaluation could score the usage of figurative language constructs such as metonymy.

In our implemented system, McGonagall, we choose to concentrate on the features of semantic faithfulness, for meaningfulness, and metre conformance, for poeticness (see Section 5.1).

4.4.2 Quantifying features

Having identified the range of possible features that can be evaluated, the next issue to address is how to yield a numerical score that accurately reflects the merits of a candidate solution’s fitness. By analyzing the possible features suggested above, we can categorize them into three different strategies of scoring:

1. Score relative to target

   For features which involve a direct comparison between a candidate solution and a given target, we can employ algorithms for calculating the degree of isomorphism between two structures, such as the edit distance algorithm for comparing two sequences (Sankoff and Kruskal, 1983), or the computational models of analogy (Falkenhainer et al., 1989, Love, 2000) for more complex structures. Indeed, these are the approaches that we take for our implemented evaluation functions described in Chapter 6.

2. Score relative to domain knowledge base

   In the absence of a specific target, one can derive a score for a candidate solution with respect to a given domain knowledge base. One such feature is that of domain consistency that we suggest in the case of meaningfulness. A candidate solution can be evaluated against an ontology to see how conceptually valid its propositional semantics is. If the ontology describes a lion as being a fierce desert animal, then a text that describes a lion going into a bookstore would probably not score too highly. Another feature we suggested was the evaluation of a candidate's metre against a database of existing poetry forms. This is reminiscent of the lazy learning method used in Varges (2002), where a candidate is repeatedly compared against a database of exemplars. This can be seen as an extension of the previous strategy.

3. Internal scoring

   This strategy assigns a score to a candidate solution without the reference point of a given target or knowledge base. One naive scoring method is to maintain a tally of points for every occurrence of a desired feature encountered in a text. For example, applied to the feature of alliteration, if we scored positively for each word that appeared in a line starting with the same phoneme, the final output could become ridiculously riddled with redundant repetitions of rewordings.

For now our aim is to facilitate a modular approach to the evaluation of features, so that each particular type of feature will have its own corresponding evaluation function. This will allow for more sophisticated approaches and techniques to be easily added in the future.

Apart from a modular approach, we also aim to parameterize the behaviour of these evaluation functions, e.g. allow a user to set the coefficients and weighting factors that determine the calculation of a certain score. A very interesting prospect is the interfacing of these coefficients with empirical data obtained from statistical literary analysis, or stylometry.

In our implemented system, MCGONAGALL, we choose to adopt the first strategy suggested, i.e. 'score relative to target' (see Section 5.1).

4.4.3 Combining scores

Assuming we have obtained numerical scores for each of the features we are considering, how do we combine them? This is a problem of multi-objective optimization, and there are several different methods that have been developed (see Section 4.2.2).

The simplest method is to compute a linear combination, or weighted sum, of the scores yielded by all individual evaluators. This parameterization could possibly allow a choice between a preference for a more faithful text and a preference for rigidly structured poetry.

Given a linear combination between the constraints of meaningfulness and poeticness, one extreme would be a purely semantic-driven optimization problem, more akin to the search-based NLG systems in Section 3.4, and the other extreme would be purely form-driven generation, as in the systems WASP and RKCP (Section 2.3.3).

In our empirical study using McGonagall, our implemented system, we use this simple linear combination with ad-hoc weightings (Section 8.5).

4.5 Designing operators

Having discussed how to evaluate a set of candidate solutions, we will now discuss how to create new variations of the candidates, which in effect drives the exploration of the search space. This process can be seen as applying a collection of operators on the chosen candidates.

We have already discussed the notion of a spectrum of search sophistication, where one extreme is the deterministic, rigidly architectured approach of traditional NLG typified by the pipeline (Section 3.2), and the opposing extreme is a fullblown, exhaustive search. The design of operators determines the position of a generation system on this spectrum.

The common wisdom in the EA community is to impose the burden of heuristic and domain knowledge on the evaluation function, and to keep the genetic operators deliberately simple. This is why EAs are successful in solving problems for which no algorithmic solutions are known. However, faced with a wealth of existing domain knowledge from the field of NLG, it would be a shame to ignore the potential of exploiting it in our system.

Adopting too much knowledge in our operators may lead us to the problems in Section 3.2, i.e. the generation gap problem. On the other hand, adopting too little knowledge may cause the

EA to perform more inefficiently than it optimally needs to.

Finding a suitable balance in this trade-off is an issue this thesis addresses. The general methodology that underlies our approach is to first see how much mileage one can obtain from knowledge-free operators in an EA framework for poetry generation, and NLG in general, and to gradually increase informedness through use of more sophisticated "smart" operators, or what Bäck et al. (1997) term **knowledge augmented operators**.

We will now discuss the design of our operators, starting with the baseline operators that enforce grammaticality, before proceeding to more knowledge-rich operators.

4.5.1 Baseline grammatical operators

Syntactic structure building is the staple of any NLP system, whether for generation or for parsing. In generation, the goal is to find the structure that best conveys a given message, whereas in parsing, the goal is to find the structure that best represents the surface string.

Typically, in generation this process is guided by the communicative goal in a top-down approach. The actual syntactic operations are performed based on a higher level rhetorical and semantic structure that has already been decided upon, e.g. RS trees (Mann and Thompson, 1987), schemas (McKeown, 1985), DRSs (Kamp and Reyle, 1993).

Our baseline operators follow no such guidelines. Syntactic structure building occurs randomly, but follows grammar rules that prescribe well-formedness. The hope is that with the guidance of the evaluation functions, and the principle of evolution, a desirable candidate solution will emerge from this stochastic process. Thus, this roughly takes the form of a bottom-up approach.

We introduce the following conceptual types of syntactic operations, which are neutral of any particular theory or formalism of grammar:

1. **Add**: this operation adds linguistic content to a candidate solution through the application of randomly selected grammatical rules.

   (4.1) John walked. → John walked *to the beach*.

2. **Delete**: this operation removes linguistic content from a candidate solution whilst preserving syntactic well-formedness.

(4.2) John likes Jill and Mary. → John likes Jill.

3. **Change**: this operation modifies the linguistic content of a candidate solution through the substitution of existing lexical items or syntactic structures with different, but compatible, ones.

   (4.3) John walked. → *Tom* walked.

   In theory, this operation should be achievable through a composition of the Add and Delete operations listed above. However, viewing this as an atomic operation allows us to explicitly control the behaviour, for example by imposing that the semantic interpretation of the text should remain unchanged by the operation, as in the case of paraphrasing as defined by (Dras, 1999).

   (4.4) John walked. → John *creeped*.

   (4.5) John loves Mary. → Mary *is loved by* John.

4. **Swap:** this operation modifies two existing linguistic structures by swapping two compatible sub-structures around. It is an instance of a crossover operation (Section 4.2.4).

   (4.6) John walked to the store. → *Mary* walked to the store.
   Mary laughed. *John* laughed.

One issue to be addressed in applying grammatical rules within an EA is the level of granularity of the operators. As EAs are an interleaved process of evolution and evaluation, partially constructed solutions are usually called upon to be meaningfully evaluated. This may present a problem for certain grammatical formalisms. We differentiate between the following two approaches to the stochastic building of syntactic structures:

1. **Incomplete derivation building**

   With this approach, operators may introduce partial linguistic structures, i.e. incomplete derivations. The potential benefit of this is that, provided we have a good evaluation metric, this finer granularity may provide more opportunity for the search to be guided. However, the drawback is that we have to deal with the evaluation of partially constructed derivations. In this respect, the notion of grammaticality of candidate solutions is by no means trivial. Furthermore, our characterisation of the search space as the space of all possible texts (Sections 4.1 and 4.3) must incorporate an extended notion of grammaticality where such incomplete derivations should be viewed as **underspecifications** of syntactic structures, and they will not violate any syntactic rules once they are fully completed.

2. **Complete derivation building**

   With this approach, we limit the search space to that of all possible complete derivations allowed by the grammar. This is achieved by designing the operators so that they always produce candidate solutions that represent complete derivations. One example of this is the explicit handling of the 'closing off' of a derivation as in PROTECTOR (Section 3.4.1). The benefit of this is that the notion of grammaticality of the candidate solutions is well-defined. However, this coarser-grained operator may cause the EA to 'miss out' on potential solutions.

In our implemented system, McGONAGALL, we implement (Sections 6.3.7 and 7.4.1) and test (Sections 8.3.2 and 8.3.3) both these approaches.

4.5.2 Knowledge augmented smart operators

Recall from Section 4.3 that we choose to enforce grammaticality through our design of representation and operators, and to optimize for meaningfulness and poeticness through search. Thus, first of all any smart operators must be built "on top" of the baseline grammatical operators described in the previous section, i.e. to ensure grammaticality.

Smart operators seek to exploit goal-specific knowledge and heuristics by explicitly trying to satisfy given targets, i.e. target semantics, target metre, etc. In the case of meaningfulness, we can adapt the hillclimbing algorithm of SPUD and PROTECTOR (Section 3.4.1). In the case of poeticness, we can employ the strategies used by WASP and RKCP (Section 2.3.3). However, this would likely impose the requirement of a strict left-to-right derivation on the syntactic operators used.

In Section 7.3 we present our design for semantically smart operators, and empirical tests using them are discussed in Section 8.4.2.

4.6 Summary

We have argued for a model of poetry generation as a state space search problem, where a state is a possible text with all its underlying representation, and a goal state is a text that satisfies the three requirements of meaningfulness, grammaticality, and poeticness. Furthermore, we have argued to solve this search using evolutionary algorithms, which we introduced in detail in Section 4.2. We choose to implement our constraints in an EA by enforcing grammaticality through design of representation and operators, and optimizing for meaningfulness and poeticness. We discussed issues of representation, evaluators and operators for the task of poetry generation, and natural language generation in general. Note that the discussion in this chapter was of a broad, theoretical nature, and that in the next three chapters we will discuss these issues as we have specifically implemented them in McGonagall. Chapters 5 to 7 will elaborate on the various representational schemes, evaluation functions, and genetic operators respectively.

Chapter 5

Representations for McGONAGALL, an Instance of EA-based NLG

In Chapter 4, we presented a rationale and a theoretical model of implementing NLG, specifically for poetry, using evolutionary algorithms. In this chapter we will describe McGONAGALL, an instance of such an EA-based NLG system, which we have implemented.

5.1 McGONAGALL in a nutshell

Based on our restricted definition of poetry as a text that simultaneously satisfies the properties of grammaticality, meaningfulness, and poeticness (Section 2.1.4), in Section 4.1 we defined poetry generation as a state space search problem, and advocated the use of evolutionary algorithms (Section 4.2) to solve this problem. In Sections 4.3 to 4.5 we discussed the issues of representational scheme, design of evaluation functions, and design of genetic operators, for performing EA-based poetry generation. However, this discussion was of a broad, theoretical level. In the following three chapters we will discuss these aspects as we have implemented them in McGONAGALL, an instance of our poetry generation model:

- Chapter 5: representation of various levels of linguistic knowledge, i.e. semantic, syntactic, and lexical information.
- Chapter 6: evaluation functions for semantic and metrical constraints.

- Chapter 7: genetic operators for exploring the search space.

As will become evident in these chapters, for the purposes of McGonagall, we have further specified our definition of meaningfulness as propositional semantics similarity, and poeticness as metre pattern similarity, using the ‘score relative to target’ strategy proposed in Section 4.4.2.

5.2 Semantic representation

Input for NLG varies greatly between systems. For our purposes, we adopt a simple ‘flat’ semantic representation that is quite widely used in tactical NLG components (Nicolov et al., 1995, Stone and Doran, 1997, Koller and Striegnitz, 2002) and machine translation systems (Whitelock, 1992, Brew, 1992). We will now define what we mean by such a representation, and motivate its choice for McGonagall.

5.2.1 Definition

Essentially, a flat semantic representation is one without embedded structures, or hierarchy, of any kind. An encoding in such a representation typically consists of a set of first order logic literals, where a literal is a predicate followed by a parenthesized list of its arguments. The set is logically interpreted as a conjunction of all its members. We will call such a set a **semantic expression**. The arguments of these literals represent concepts in the domain such as objects and events, while the predicates state properties of these concepts, in the case of unary predicates, or relations between concepts, in the case of n-ary predicates, where $n > 1$.

In our simple implementation, we do not handle negative literals. Instead, lexical items that indicate negation, e.g.*“not”*, simply convey the literal $not(X)$, where X is some argument that refers to an event or object that has a predication which is negated, e.g. see 5.4 below, where l is the event of John loving Mary.

For example, the representation of the semantics of the sentence *“John loves Mary”* is given in Example 5.1, where l is the event of j, who has the property of ‘being John’, loving m, who has the property of ‘being Mary’.

(5.1) $\{john(j), mary(m), love(l, j, m)\}$
“John loves Mary”

(5.2) $\{john(j), mary(m), love(l, j, m), believe(j, l)\}$
"John believes that he loves Mary"

(5.3) $\{john(j), mary(m), love(l, j, m), past(l)\}$
"John loved Mary"

(5.4) $\{john(j), mary(m), love(l, j, m), not(l)\}$
"John does not love Mary"

Within this regime, events and situations are often represented as entities in their own right (Hobbs, 1985). This allows one to simulate certain higher order constructions, such as modality (Example 5.2), or linguistic phenomena such as tense (Example 5.3). Note that the logical form in (5.1) can be realized by both the utterances in (5.1) and (5.3) if we allow for underspecification of tense (but not vice versa, i.e. the logical form in (5.3) being represented by the utterance in (5.1), unless we allow approximate generation, which we will discuss later).

Examples of flat semantic representations are Hobbs' ontologically promiscuous semantics (Hobbs, 1985), the 'bag of concepts' in shake and bake machine translation (Brew, 1992, Whitelock, 1992), and minimal recursion semantics, or MRS (Copestake et al., 1995). MRS is a further development that handles representation of quantification and scope.

5.2.2 Benefits

There are various desirable properties of a flat semantic representation detailed in the works cited above. Of particular interest to us is that it provides a simple syntax-semantics interface. Since semantic expressions are flat sets, compositional rules usually just involve unions of sets with variable binding.

Moreover, such flat representations allow for underspecification of semantics, which is useful for interleaved incremental generation of both semantic and syntactic structures.

Furthermore, as stated in Copestake et al. (1995), a flat representation conveniently sidesteps some aspects of the logical equivalence problem, i.e. where the semantic representation of paraphrases are slightly different due to the compositional semantics rules implicitly encoding syntactic information. For example (taken from Copestake et al. (1995)), the logical form of the English phrase *fierce black cat* as given by a simplistic grammar may be the one given in Example (5.5), while the logical form of its Spanish translation, *gato negro y feroz*, might be the one in Example (5.6):

(5.5) $\lambda x[\text{fierce}(x) \wedge (\text{black}(x) \wedge \text{cat}(x))]$

(5.6) $\lambda x[\text{cat}(x) \wedge (\text{black}(x) \wedge \text{fierce}(x))]$

The difference arises due to the binary nature of ∧ giving rise to spurious bracketing ambiguity. Although logically equivalent, some logical processing must first be executed to equate the two. A flat semantic representation, with a canonical ordering, say alphabetically, would yield the same formula for both utterances, avoiding any computational expense of deciding their equivalence:

$$\{black(x), cat(x), fierce(x)\}$$

However, for our purposes, the main advantage of such a representation is increased paraphrasing power. If we view this logical equivalence problem from a generation viewpoint, we can see that an encoding of semantics in a flat representation can be realized by many different utterances. Contrast this with hierarchical representations which often already encode high level syntactic decisions, e.g. which concept is to be realized as a verb, as its complements, how adjuncts are ordered, etc. This issue is raised by Nicolov et al. (1995, 1996) with respect to conceptual graphs, but applies also to our representation.

5.2.3 Drawbacks

The main drawback of generating from flat representations is its computational expense. Both Brew (1992) and Koller and Striegnitz (2002) show that this is an NP-complete problem. However, since we are employing evolutionary algorithms for our generation process, for reasons discussed in Chapter 4, we are already equipped with a powerful search mechanism for solving problems of this nature. Furthermore, computational efficiency is not the focus of our research.

5.2.4 Examples

An important application of this semantic representation is as a language for defining the input, or target, semantics. For example, one possible encoding of the first two lines in Belloc's *The Lion*,

The lion, the lion, he dwells in the waste,
He has a big head and a very small waist.

could be

$$\{lion(\_,l),\ dwell(d,l),\ inside(\_,d,was),\ waste(\_,was),\ own(\_,l,h),\ head(\_,h),\ big(\_,h),\ own(\_,l,wai),\ small(s,wai),\ waist(\_,wai),\ very(\_,s)\}$$

In our encoding, we assign for all predicates a first argument that corresponds to the event entity of this predication holding. For example, in $\{lion(x,l)\}$, the constant x represents the *event* of l being a lion, or the *state* of l's 'lion-ness'. This is one of the main points of Hobbs (1985)'s ontological promiscuity: virtually any predication can be referred to as some event or state. Hobbs (1985) introduces a notation where for every n-ary predicate p, there is an $n+1$-ary predicate p' whose first argument is the condition that holds when p is true of its arguments. For example, if $big(X)$ means that X is big, $big'(E,X)$ means that E is the condition of X being big. However, in our encoding above, we do not use this notation, choosing instead to explicitly assign every predicate a corresponding event argument. We use a Prolog-style notation of '_' to indicate when we do not care about an argument's value.

Thus, in our encoding of *The Lion*, we are only interested in the dwelling event entity d, because we need to specify its location *("dwells in the waste")*, and the condition of being small s, because we emphasize its nature *("very small")*.

There are a vast number of paraphrases that could be generated to realize this semantic encoding. For example:

(5.7) *The big headed lion, who dwells inside the waste, has an extremely tiny waist.*

(5.8) *A lion lives on the waste. Its head is big, and its waist is very small.*

(5.9) *The dwelling of the lion, with its very small waist and big head, is in the waste.*

none of which possess the same metre pattern as the original text.

We will see how the semantics are represented in the lexicon, and how they are composed within the grammar formalism, in Section 5.3.8.

5.3 Linguistic representation

The grammar formalism we use for McGonagall is Lexicalized Tree Adjoining Grammar (Schabes, 1990), or LTAG. We also employ unification-based feature structures (Vijay-Shanker and Joshi, 1988) for the handling of linguistic phenomena. In this section, we will first describe the aspects of this formalism pertaining to our work before discussing the advantages it provides for computational linguistics, specifically generation purposes. Lastly, we will discuss implications of using it within a non-monotonic, incremental generation process as performed by our EA framework.

5.3.1 Lexicalized Tree Adjoining Grammars

LTAG is based on Tree Adjoining Grammar (Joshi and Schabes, 1997), or TAG, a grammar formalism that is widely used due to its elegant account of certain linguistic phenomena.

TAG is based on the composition of **elementary trees**, the primitive elements of TAG. There are two types of elementary trees, **initial trees** and **auxiliary trees**.

A tree is an initial tree if:

1. all its internal nodes are labelled by non-terminals, and
2. all its leaf nodes are labelled by terminals, or by non-terminal nodes marked for substitution.

Initial trees represent minimal linguistic structures that contain no recursion. They are minimal in the sense that they account for *all* and *only* the arguments of the head of the syntactic constituent they represent, e.g. a sentential structure would contain a verb and all its complements.

Initial trees are often notated with the symbol α. An initial tree is said to be of type X if its root is labelled with type X. Nodes marked for substitution, or **substitution nodes**, are notated with a $\downarrow$ following the node's label.

A tree is an auxiliary tree if:

1. all its internal nodes are labelled by non-terminals,
2. all its leaf nodes are labelled by terminals, or by non-terminal nodes marked for substitution, except for exactly one non-terminal node, called the foot node, and

3. its foot node has the same label as its root node.

Like initial trees, auxiliary trees represent minimal linguistic structures. However, auxiliary trees represent recursive structures. Linguistically, they account for constituents that are adjuncts to basic structures, e.g. adjectives, adverbials, etc..

Auxiliary trees are usually notated with the symbol β. Foot nodes of auxiliary trees are notated with a $*$ after the node's label.

Furthermore, nodes are often referred to using the Gorn tree addressing scheme. A Gorn address is a list of integers that indicates the position of a node within a tree. The Gorn address of the *i*-th child of node *n* is the Gorn address of *n* with *i* appended to the end of the list. The Gorn address of a root node is ε, or sometimes 0 for convenience.

Figure 5.1 shows an example of an initial and an auxiliary tree, both as a schematic figure that shows their characteristics, and as actual phrase structure trees. The nodes are also labelled with their Gorn addresses.

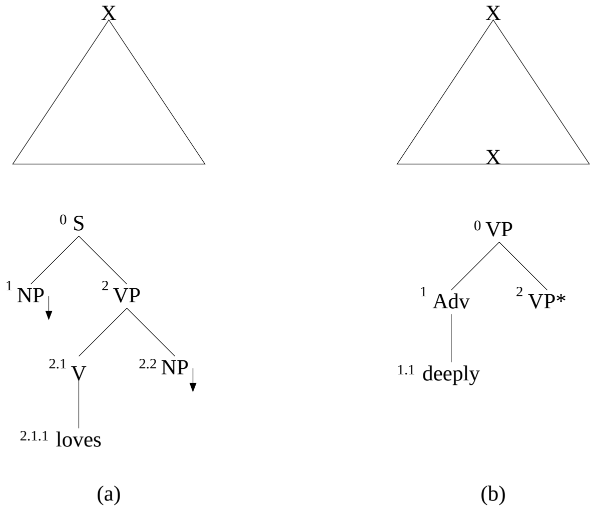


Figure 5.1: Elementary trees in TAG consist of (a) Initial trees and (b) Auxiliary trees

In TAG, there are two types of operations that can be used to compose elementary trees, **substitution** and **adjunction**. In substitution, the root node of an initial tree is merged with a non-terminal leaf node that is marked for substitution in another tree, called the substitution node, producing a new tree. The root node and the substitution node must share the same label for the operation to be warranted. Figure 5.2 shows an example of substitution taking place.

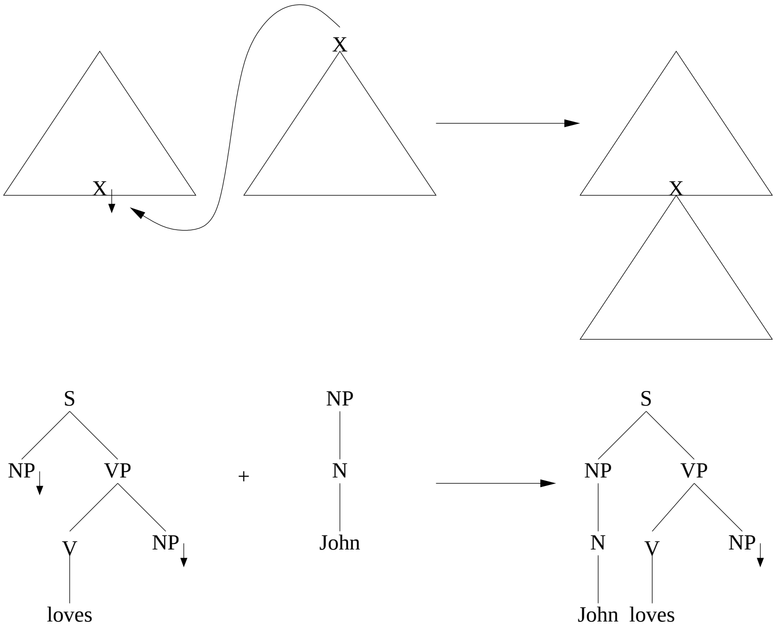


Figure 5.2: The substitution operation in TAG

In adjunction, an auxiliary tree is inserted into another tree at a non-terminal node, called the adjunction node, as follows: the subtree dominated by the adjunction node is “excised” and attached to the auxiliary tree’s foot node, and the root node of the auxiliary tree is then merged with the adjunction node. Figure 5.3 shows an example of adjunction.

A grammar *G* consists of a set of initial trees, *I*, and auxiliary trees, *A*.

Finally, a **lexicalized** grammar is one in which each structure is related to a lexical item. The example trees in Figures 5.1, 5.2, and 5.3 are actually specific examples of lexicalized elementary trees. The lexical item associated with each elementary tree is called the **anchor**.

5.3.2 Deep lexicon and the sharing of syntactic information

With lexicalized grammars it is often the case that most of the linguistic knowledge is embedded within the lexicon (hence a ‘deep’ lexicon). These linguistic structures are then combined with a limited set of syntactic rules. The space of possible combinations of structures in LTAG is governed by the operations of substitution and adjunction.

Because lexical items share a lot of syntactic structure in common with each other, this is

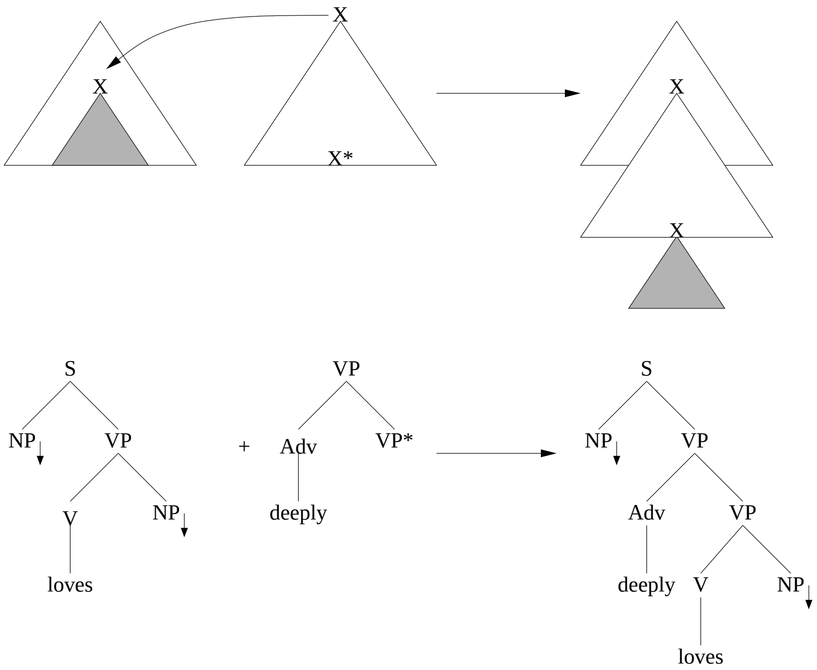


Figure 5.3: The adjunction operation in TAG

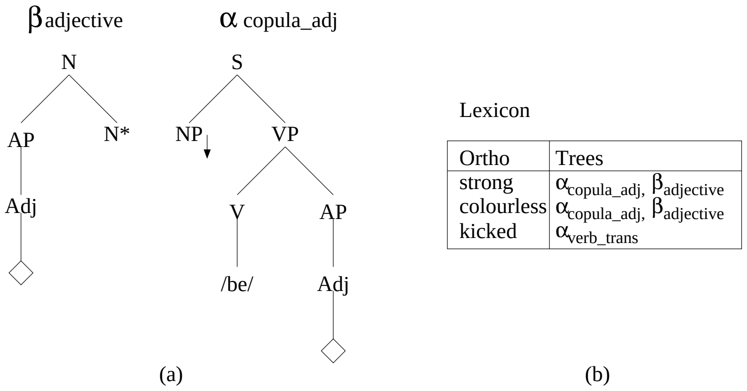
efficiently encoded within a set of elementary trees. Lexical items need only specify a list of names of elementary trees which they can potentially anchor.

Figure 5.4 is an example of a snippet of a grammar and lexicon in LTAG, showing how lexical items, in this case adjectives, can be associated with different elementary trees. The trees in this case are $\beta_{adjective}$, an auxiliary tree for adjoining in adjectives, and α_{copula_adj}, a copula sentence construction headed by an adjective (e.g.*"John is strong"*). Note that as these elementary trees do not have a specific lexical item anchoring them yet, their anchor node is notated with a ⋄.

Lexical entries can contain arbitrarily complex knowledge, and during the process of anchoring trees with lexical items, one can perform further syntactic checks through the use of feature structures (Section 5.3.4).

5.3.3 Derivation tree as primary data structure

We have seen that in TAG, complex phrase structure trees are constructed through the composition of elementary trees using the substitution and adjunction operators. The **derivation tree** is a record of this process. It uniquely specifies its **derived tree**, the phrase structure tree



Lexicon

Ortho	Trees
strong	$\alpha_{\text{copula\_adj}}$, $\beta_{\text{adjective}}$
colourless	$\alpha_{\text{copula\_adj}}$, $\beta_{\text{adjective}}$
kicked	$\alpha_{\text{verb\_trans}}$

Figure 5.4: Specifying linguistic knowledge in the grammar and lexicon

obtained by the recorded operations.

Figure 5.5 gives an example of this. The derived tree in (c) is obtained by a series of operations on the elementary trees in (a). These operations are recorded by the derivation tree in (b). Each node in the derivation tree specifies an elementary tree. Furthermore, each node aside from the root node also specifies the Gorn address of the node in its parent's elementary tree where the operation occurs. We call this the **operation address**, and it is indicated by the label of the edge connecting a node to its parent. For example, α_{book} subtitutes into α_{read} at address 2.2, the object NP node. The process of composing the elementary trees is illustrated in (d).

Although a derivation tree is simply a historical record, or a roadmap, of how a derived tree is constructed, it is of significant importance for generation, and in particular our stochastic, incremental approach to generation.

Firstly, Joshi (1987) points out that a derivation tree implicitly characterizes the surface tree (i.e. derived tree) as well as serving as the basis for compositional semantic interpretation. Note how the derivation tree in 5.5(b) exhibits the predicate argument structure of the underlying semantics $\{read(r, j, b), john(\_, j), book(\_, b), yesterday(r)\}$. Candito and Kahane (1998) state that a derivation tree can be viewed as a semantic dependency graph in Meaning-Text Theory. From a generation viewpoint, the derivation tree can be seen as the basic formal object that is constructed during the course of sentence generation from a semantic representation (Joshi, 1987). This can be seen in generation systems such as SPUD and PROTECTOR (Section 3.4.1), and in our implementation of semantically-motivated operators (Section 7.3).

Secondly, as elaborated in the next few sections, it is the appropriate data structure on which to

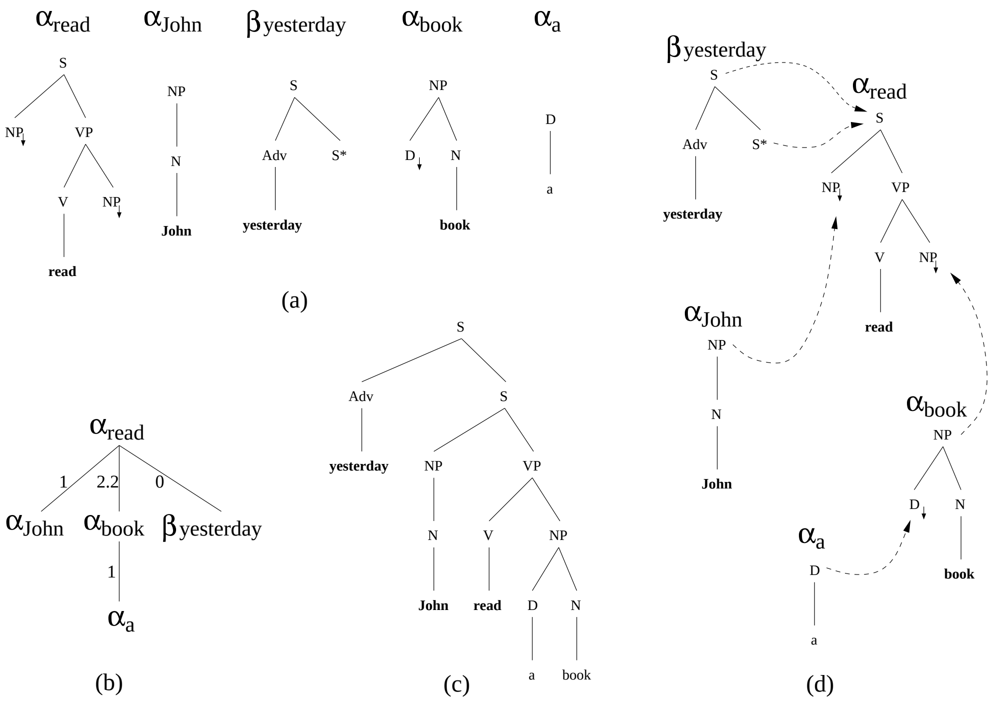


Figure 5.5: A derivation tree captures the process of composition of elementary trees

perform nonmonotonic operations in our stochastic and incremental generation framework.

Thus, the TAG derivation tree is the primary data structure that an individual in our evolutionary algorithm is composed of.

5.3.4 Feature-structures for unification

The unification of feature structures in unification-based grammars is a mechanism for capturing various linguistic phenomena in a compact manner (Gazdar and Mellish, 1989). Extending TAG with a unification framework is potentially problematic due to the adjunction operator. This is because adjunction ‘splits’ a node in two, whereas unification is essentially a destructive operation. We recommend Vijay-Shanker and Joshi (1988) and Kilger (1992) for a discussion of this problem.

Following Vijay-Shanker and Joshi (1988), our implemented extension to TAG for unification is as follows: each node in an elementary tree is associated with a feature structure which consists of a **top** part and a **bottom** part. The top part contains information relating to the tree that dominates it, and the bottom part contains information relating to the tree it dominates. Substitution nodes only have the top part features, as the tree substituting in carries the bottom features.

When performing substitution, the top feature structure of the new resulting node is the union of the top features of the original nodes. The bottom feature structure of the new resulting node is simply the bottom feature of the root node of the substituting tree, as the substitution node has no bottom features. For adjunction, the node being adjoined into is 'split' into two, and its top feature is unified with the top feature of the adjoining tree's root node, whereas its bottom feature is unified with the bottom feature of the adjoining tree's foot node. Figures 5.6 and 5.7 show these operations in a graphical manner.

For a TAG derivation to be valid, the top and bottom features structures at every node must be unified, or "collapsed". Since adjunction requires both the top and bottom feature structures, nodes may not be collapsed until it is known that adjunction will not occur at that node. In practice, the collapsing of nodes is carried out at the end of the derivation.

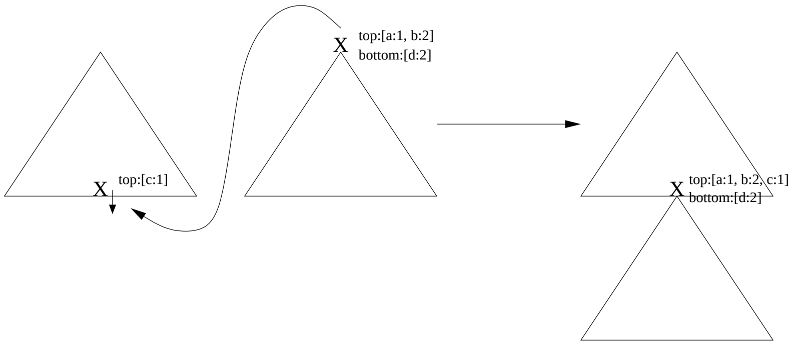


Figure 5.6: Unification of features in substitution

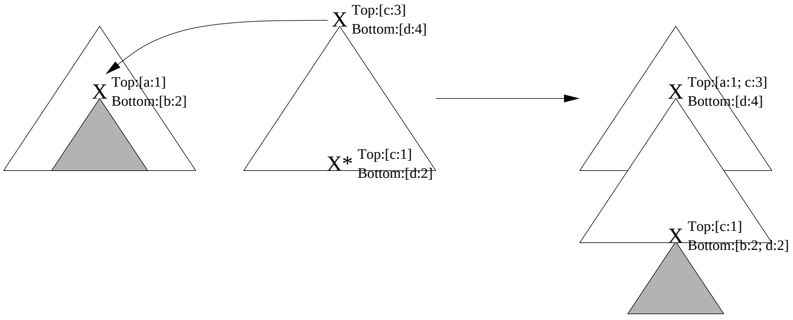


Figure 5.7: Unification of features in adjunction

Within this unification-based framework, one can dynamically specify local constraints that would otherwise require a proliferation of elementary trees.

Figure 5.8 shows a simplified account of how one might use feature structures to ensure agreement between a verb and its subject. Verbs and nouns in the lexicon are marked with agreement features. For example, the intransitive verb *'reads'* has [AGR [3SG : +]] to indicate that it requires a subject noun phrase that is third person singular. The noun *'John'* has [AGR [3SG : +]] and *'dogs'* has [AGR [3SG : -]].

When a lexical item anchors an elementary tree, the feature structure of the lexical item is copied to the bottom feature of the elementary tree's preterminal node, thus requiring it to unify with the preterminal's top feature once the node is collapsed. This can be seen in the four anchored trees of Figure 5.8.

These features are in turn propagated further by coindexed features that nodes in the grammar are marked with. For example, the indices $\boxed{4}$ and $\boxed{5}$ ensure that the noun phrases that are projected by *'John'* and *'dogs'* will inherit the features from the lexical anchor. When substituted into the subject position of α_{reads}, these features will have to unify with $\boxed{1}$, which itself inherits features from its anchor, the verb *'reads'*. This ensures that α_{dogs} would fail to appear as the subject of α_{reads}, but only after the top and bottom features at each node have been unified.

Within α_{reads}, one could possibly co-index the features between the NP and V node directly. In this example, however, we allow the possibility of a tree adjoining in at VP that can alter the relationship of agreement between subject and verb, hence the use of $\boxed{1}$ and $\boxed{2}$. If adjunction does not occur at that node, they will simply unify. In the case of adverbs such as $\beta_{quietly}$, agreement is simply passed along between the root and foot nodes.

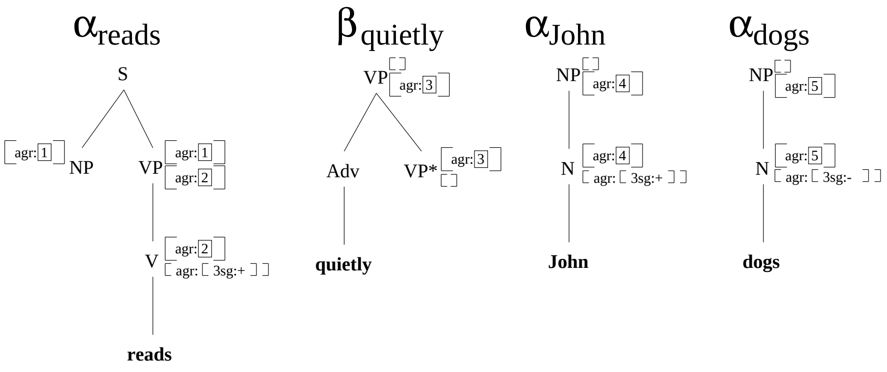


Figure 5.8: Dynamically specifying local constraints using feature structures

5.3.5 Unification and nonmonotonicity

The use of unification of feature structures to dynamically specify linguistic constraints within our stochastic and incremental generation framework requires special attention. This is because unification can mainly be seen as a monotonic operation: it always builds larger structures. However, our system involves two sources of nonmonotonic operations: the adjunction operation, and the overtly nonmonotonic operators of subtree deleting and swapping (see Sections 7.2.2 and 7.2.3).

Recall that Vijay-Shanker's approach to unification in TAG requires the specification of top and bottom features for each node. This is because the adjunction operator can be seen as splitting a node into two, corresponding to the root and foot nodes of the auxiliary tree adjoining in. Recall also that the top and bottom features of every node must eventually be unified for a derivation to be complete. This presents a problem within our incremental and interleaved generation framework: when should these features be unified?

The common approach is to defer unification of top and bottom features until the end of the derivation process. However, in an evolutionary algorithm, operators and evaluators are interleaved, thus evaluators are required to evaluate candidate solutions even though they may not be complete derivations yet. For this purpose, the feature structures of a candidate solution must be unified at every iteration so that grammaticality can be verified.

If we do collapse top and bottom features at every iteration, the destructive nature of unification prevents us from recovering the original features, which are needed for adjunction to occur at a node at a later iteration. Figure 5.9(a) shows an example of the sentence *"John reads"*, where the top and bottom features have been unified to ensure agreement between subject and verb. However, we can no longer adjoin at the VP node because the original features are no longer available.

On the other hand, deferring unification defeats the purpose of using feature structures to specify grammatical constraints.

Kilger (1992) explores this problem and proposes two possible solutions. The first solution is to always store a copy of the original local feature structures of the elementary trees, and to unify top and bottom features during the derivation process. When adjunction requires access to feature structures before they were unified, the currently built global feature structure is discarded and rebuilt from the original feature structures. The second solution is to perform unification in a non-destructive manner. Instead of structure-sharing when two features are unified, pointers between the corresponding original features are created. Adjunction can then be handled by redirecting the pointers to new feature structures as necessary. Kilger (1992) finds that both approaches have a balance of advantages and disadvantages. For the first approach, adjunction is a very costly operation as the global feature structure must be discarded and rebuilt. However, for the second approach, features cannot be read off directly, but must be recovered through a traversal of the pointers. In the worst case, this could be a traversal of the entire global feature structure.

We have chosen to use the first solution, mainly due to the relative simplicity of its implementation within MCGONAGALL. Since our primary data structure is the derivation tree, we are implicitly storing all necessary local feature structures. When required, all we need to do is rebuild the derived tree without unified top and bottom feature structures. This would produce a structure as in 5.9(b), where we can adjoin in $\beta_{quietly}$ at the VP node to obtain the structure in 5.9(c). Once the top and bottom features of this tree are unified, we obtain the final derived tree as in 5.9(d).

We can minimize redundant computation by reusing the previously built derived tree if we know it has not been modified. Thus every derivation tree will be associated with a **cached** derived tree, and a flag to indicate whether the cached copy is still up to date.

This approach also allows us to handle the second instance of nonmonotonic operations, i.e. the overtly nonmonotonic operations of subtree deleting and swapping as detailed in Chapter 7.

5.3.6 Obligatory adjunction and incomplete syntactic structures

Even if we do account for the destructive nature of unification, as in the previous section, the notion of grammaticality of a partially derived tree is not trivial.

A particularly difficult case is that of **obligatory adjunction**, where the top and bottom features of a node cannot unify until an auxiliary tree is adjoined in. Figure 5.10 shows an example of this. In (a) we see the elementary trees for a transitive and auxiliary verb, $\alpha_{transitive}$ and $\beta_{auxiliary}$, along with a lexicon. For the sake of clarity, only the relevant features are shown. The top feature structure for the *V* node in $\alpha_{transitive}$ indicates that the form of its verb must be finite. Notice that it is placed in the top section, as this constraint can be seen to be coming from ‘above’, i.e. what a sentence requires its verb must be in order to be grammatical.

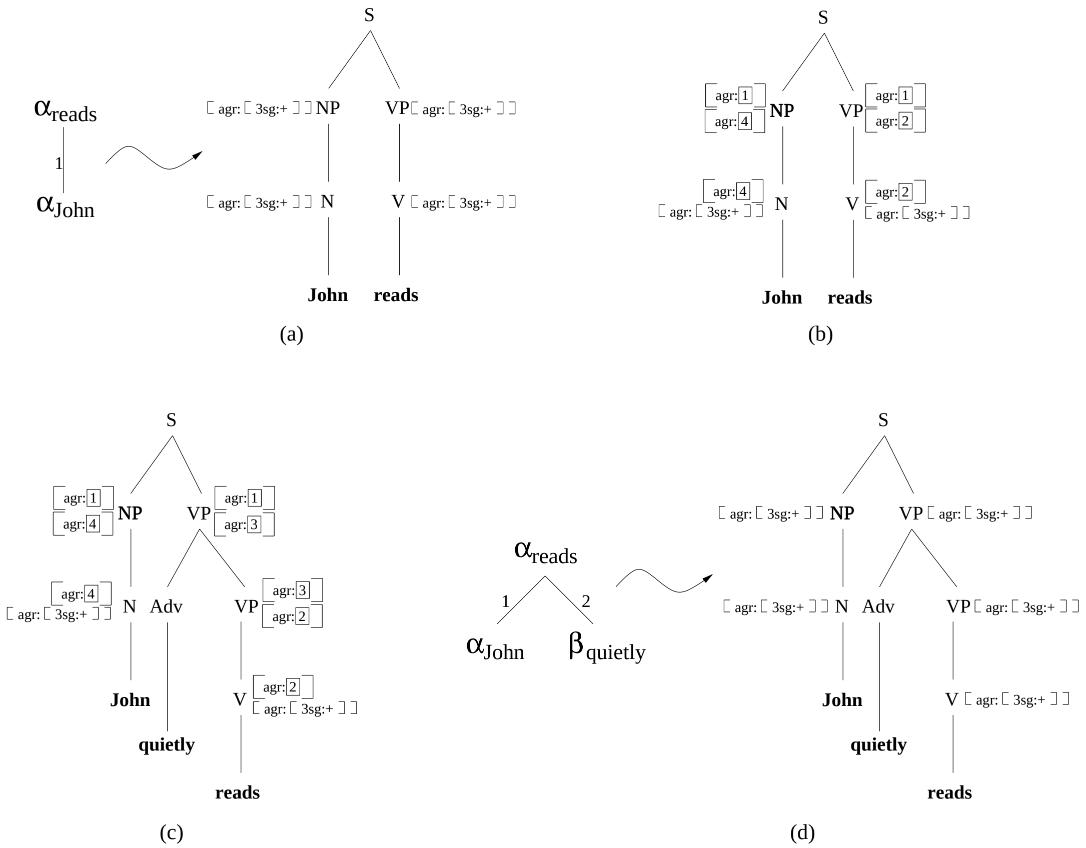
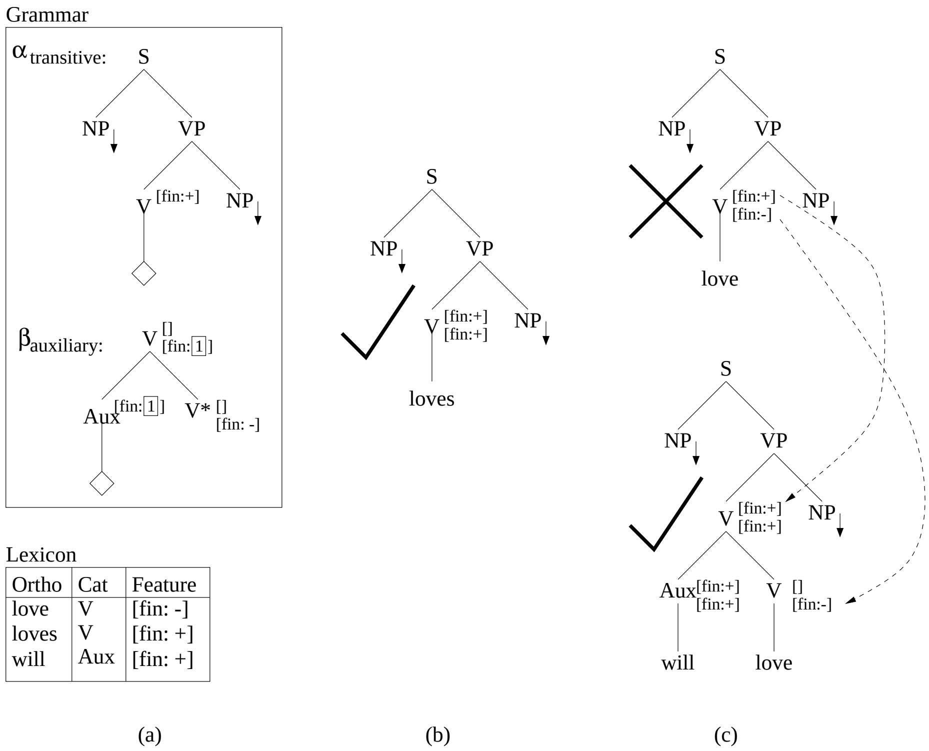


Figure 5.9: The derivation tree facilitates access to original local feature structures during adjunction

Verbs of finite form, such as *‘loves’*, can happily anchor $\alpha_{transitive}$, as in (b). However, the infinitive form, *‘love’*, cannot, as its feature does not unify with the preterminal’s top feature (remember that upon anchoring, the lexical item’s feature structure is copied to the preterminal’s bottom feature, and that the top and bottom features must unify). This is not to say that it cannot anchor $\alpha_{transitive}$ at all. This problem can be remedied by adjoining in $\beta_{auxiliary}$ at the V node. As we see in (c), V is split into two, the top feature unifies with $\beta_{auxiliary}$’s root, and the bottom feature unifies with its foot. In $\beta_{auxiliary}$, we co-index the [FIN] at the root with that of the preterminal. Thus, only auxiliary verbs with [FIN : +] will unify. Furthermore, we specify [FIN : -] at the root $V*$ node, to prevent the auxiliary verb from adjoining onto elementary trees anchored by verbs of finite form.

In our incremental framework, a partial derivation may have its top and bottom features not unify, even though it may well be on its way to becoming a syntactic construction. For parsing



Ortho	Cat	Feature
love	V	[fin: -]
loves	V	[fin: +]
will	Aux	[fin: +]

Figure 5.10: Obligatory adjunction

this is not a problem, as the unifying of top and bottom features can be deferred until the very end of the construction.

We believe there are three ways that this can be handled:

1. Design the grammar so as not to employ obligatory adjunction. This is the simplest solution, and is what we have chosen to do with our small test grammar that is used for most of our experiments.

2. Design the operators such that obligatory adjunctions are always immediately handled. This can be accomplished by a cascading of operations, i.e. if an operation introduces a tree that contains a node with obligatory adjunction, then a follow-up adjunction at that node must be performed and seen as an "atomic" operator by the EA. In MCGONAGALL, this could be implemented using compound operators (Section 7.4).

3. Allow unification errors, and penalize them as a component of the evaluation function. This would be similar to the approach used in SPUD (Stone et al., 2001). However, SPUD uses this as a criteria for ranking candidates in an exhaustive greedy search, which

means that ungrammatical constructions will virtually never be considered. We feel that implementing this approach in a stochastic heuristic search such as an EA would be excessively inefficient.

5.3.7 TAG and Generation

Elementary trees are minimal linguistic structures that are enough to capture dependencies locally. Put another way, an elementary tree accounts for all and only the arguments of its head. This allows the local definability of all dependencies, the locality of feature checking, and the locality of argument structure (Joshi, 1987). These trees are said to have an **extended domain of locality** compared to, for example, the rules in CFG-based formalisms. Furthermore, the adjunction operation factors recursion in such a way that argument structure is preserved over arbitrarily long distances. Finally, within LTAG, each elementary tree has a lexical anchor which is the semantic head of the constituent represented by the tree.

These features make elementary trees the appropriate syntactic building blocks to work with while incrementally consuming semantic input to be generated. Recent tactical generators such as SPUD and PROTECTOR employ TAG for a flexible approach to generation (Section 3.4.1). They first construct the skeletal frame of a sentence, i.e. the verb and its complements, via substitution. Later on, they add modifiers and adjuncts via the adjoining in of adjectives, adverbials, etc.

5.3.8 Semantic-syntactic interface

Following Stone and Doran (1997), we have designed and implemented a method of handling compositional semantics based on the notion of what we call semantic **signatures**. It roughly serves the same purpose as lambda calculus expressions in conventional Montague-like compositional semantics (see Dowty et al. (1981) for an introduction to this).

Each word in the lexicon is labelled with a list of variables called its signature. This list corresponds to the argument variables of the word's semantic expression. Nodes in an elementary tree may also be labelled with signatures. The signature of an anchor node corresponds to the signature of the tree's lexical anchor. When a word anchors an elementary tree, its signature is unified with the anchor node's signature, and the semantics of the elementary tree is simply the semantics of the word.

Figure 5.11 shows an example of an elementary tree being anchored by a lexical item, and the resulting semantics. When the verb *'loves'* anchors the transitive verb tree $\alpha_{transitive}$, its signature, $[Loving, Lover, Lovee]$, is unified with that of the $\alpha_{transitive}$'s anchor, $[X,Y,Z]$. The semantics of α_{loves}, $S_{\alpha_{loves}}$, thus becomes $\{loves(X,Y,Z)\}$.

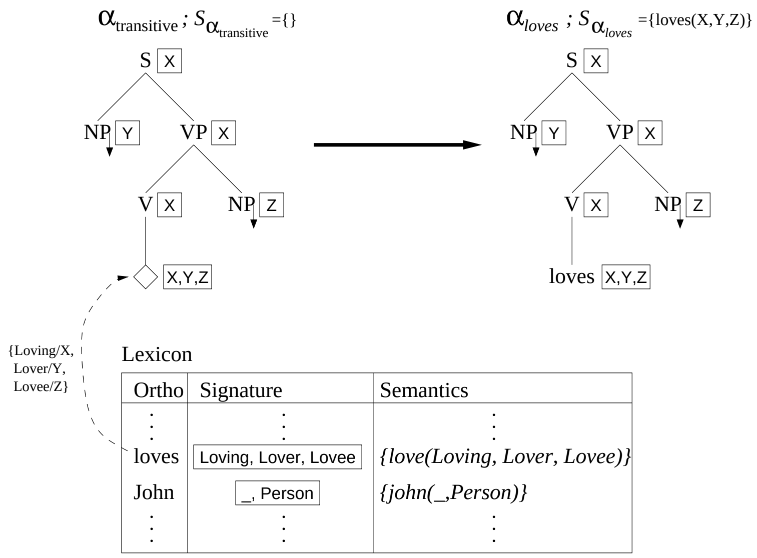


Figure 5.11: The semantics of an anchored elementary tree

The signatures of all other nodes, i.e. internal nodes, are used to encode the correct predicate-argument structure when composing elementary trees. Roughly speaking, they correspond to the objects and events the syntactic constituents they represent are 'about'.

When substitution occurs, the signatures of the substitution node and the root of the substituted initial tree are unified. Likewise, when adjunction occurs, the signatures of the adjunction site node and the root of the adjoined auxiliary tree are unified. We can imagine the auxiliary tree's foot node is also labelled with a signature that is identical to that of its root, but since in TAG adjunction does not occur at a foot node it is unnecessary to specify this. All this is in addition to the previously mentioned unification of the signatures of a lexical item and the anchor node of the elementary tree it anchors.

These unifications serve to maintain the predicate argument structure of the derivation, and the compositional semantics of a derivation is simply the union of semantic expressions introduced by words anchored at the elementary trees.

Figure 5.12 shows an example of the substituting in of two noun phrases as the subject and object of a transitive tree. During the substitution of α_{John} at the subject NP position, the signatures $[Y]$ and $[A]$ are unified. Likewise, for the substitution of α_{Mary} at the object NP position, the signatures $[Z]$ and $[B]$ are unified. From this, we can recover the semantics of the derivation $S = \{love(X,Y,Z), john(\_,Y), mary(\_,Z)\}$.

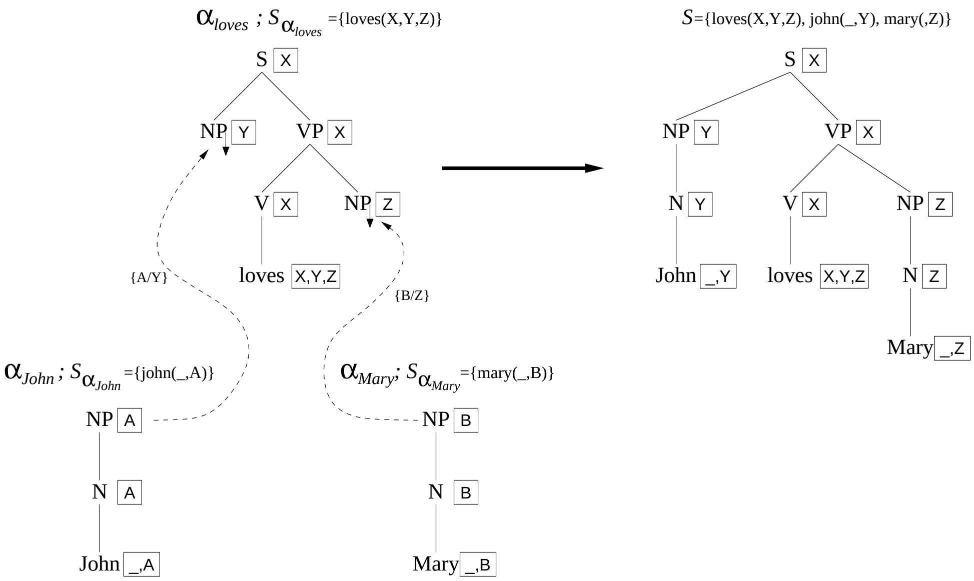

α_{loves} ; $S_{\alpha_{loves}}$ ={loves(X,Y,Z)}
S X
NP Y
VP X
V X
NP Z
loves X,Y,Z
{A/Y}
{B/Z}
α_{John} ; $S_{\alpha_{John}}$ ={john(_,A)}
NP A
N A
John _,A
α_{Mary}; $S_{\alpha_{Mary}}$ ={mary(_,B)}
NP B
N B
Mary _,B
S={loves(X,Y,Z), john(_,Y), mary(,Z)}
S X
NP Y
VP X
N Y
V X
NP Z
John _,Y
loves X,Y,Z
N Z
Mary _,Z

Figure 5.12: Compositional semantics through unification of signatures

We have subsequently discovered a very similar approach presented in Kallmeyer (2002), which also uses a flat semantic representation, and unifies argument variables upon substitution or adjunction. Similar to our signatures, semantic arguments are linked with nodes in elementary tree trees through the specification of their Gorn address. For example, a transitive verb would have its subject linked with node 1 and its object with node 2.2 of $\alpha_{transitive}$ in Figure 5.11.

Kallmeyer notes that the proper way to define compositional semantics for LTAG is with respect to derivation trees, as they express predicate argument dependencies.

Pronoun resolution

In McGONAGALL, we need a method of **pronoun resolution**, i.e. the resolving of the antecedents of words that exhibit anaphoric reference. Although this alone is a subject of intensive research, we will choose to adopt a simple method, as this is not a central aspect of our thesis. We have chosen to implement a purely syntactic pronoun resolution algorithm that is based on Hobbs (1978).

Note that pronoun resolution is commonly found in natural language understanding applications, and not in NLG. This is because in NLG the semantics is already well-defined and thus all anaphoric reference can be determined unambiguously. However, McGONAGALL does not proceed in the communicative goal-driven process of informative NLG, where anaphoric reference is intentionally used to convey the semantics. McGONAGALL proceeds in a stochastic manner where only syntactic well-formedness is ensured, and thus it has no specific intention of anaphoric reference when adding linguistic content in the form of pronouns. In a purely generate-and-test setting such as this, we have to recover the meaning that is conveyed by the text in order to measure its fitness.

There are two categories of words in our lexicon that exhibit anaphoric reference: pronouns, e.g.*'she', 'they', 'it'*, and genitive determiners, e.g.*'his', 'their', 'its'*. In our lexicon, these words' semantic expressions are empty sets. They merely serve as syntactic pointers to another word in the derivation, i.e. their antecedent, that conveys the appropriate semantic expression.

This pointer is implemented by unifying the signature variables of these pronouns (or genitive determiners) with that of their respective antecedents. In particular, we unify the variable that represents the object entity. This unification must be applied to a derived tree after all other required unifications of signatures have been performed (Section 5.3.4).

We first build a list of relevant anchor nodes by performing a preorder traversal of the derived tree and appending anchor nodes that are either nouns or genitive determiners to the list. For every pronoun or genitive determiner in the list, we then unify its signature with that of its antecedent. Its antecedent is the first **compatible** non-pronoun noun that precedes it in the list, where by compatible we mean that their feature structures, representing agreement and selectional restrictions, unify.

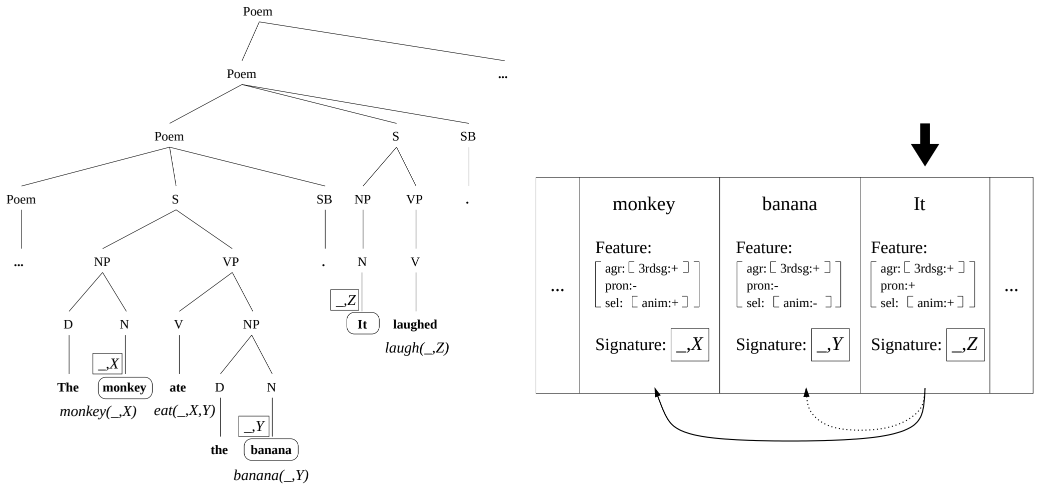
An example of this is given in Figure 5.13. Part (a) shows the derived tree of a text containing the sentences *"The monkey ate the banana. It laughed"*, with the relevant anchor nodes, i.e. nouns, highlighted. These nodes are compiled into a list by performing a traversal of the tree, which is shown in (b). The pronoun we are trying to resolve is *'It'*, as indicated by the large arrow. To find its antecedent, we move to the left of the pronoun, and try to find a noun that is not a pronoun, i.e. has the feature [PRON : -], and has AGR and SEL features that unify with the pronoun's features. Note that the pronoun *'It'* has [SEL [ANIM : +]]. This is a selectional restriction imposed by the verb *'laughed'*, i.e. only animate objects can laugh. This selectional restriction rules out the first noun encountered, *'banana'*, and eventually it finds the antecedent is *'monkey'*.

Having found the antecedent, the corresponding signature variables, X and Z, are unified. Note that before pronoun resolution, the conveyed semantics was

$$\{eat(\_,X,Y), monkey(\_,X), banana(\_,Y), laugh(\_,Z)\},$$

but after resolution, this becomes

$$\{eat(\_,X,Y), monkey(\_,X), banana(\_,Y), laugh(\_,X)\}.$$



(a) (b)

Figure 5.13: (a) Anchor nodes of nouns and genitives (b) Computing antecedents

Our simple algorithm has the following limitations:

1. Pronouns cannot refer to events, e.g.*"John ran. It was very tiring."*.

2. No use of domain knowledge. For example, if the text in the example above read *"The monkey ate the spider. It laughed."*, it is reasonable to assume that animate objects, once eaten, cannot laugh.

3. No handling of reflexive pronouns. Given the sentence *"John hit him."*, the pronoun *'him'* is considered to refer to *'John'*.

5.3.9 Lexical representation

In previous sections, we have mentioned that our lexicon stores syntactic (Section 5.3.2 and 5.3.4) as well as semantic information (Section 5.3.8). We will now describe this in more detail, as well as introduce another aspect that is carried by the lexicon: phonetic information. This is needed for the handling of phonetic features such as metre.

For syntactic information, we follow the approach used in XTAG (XTAG Research Group, 2001) and SPUD (Stone and Doran, 1997), where each word specifies all the different elementary trees it can potentially anchor. Thus, each word is associated with a list of elementary tree names. Furthermore, each word is also associated with a feature structure that specifies all its syntactic properties such as its category, agreement, selectional restrictions, etc. In theory, this feature structure alone is sufficient to determine which elementary trees it can anchor. However, the list of tree names reduces the need to search through the entire grammar every time we need to find an elementary tree a given word can anchor.

For semantic information, we have already defined that a lexical entry contains a semantic expression and a corresponding signature.

For phonetic information, each word is associated with its **phonetic spelling**, which we take from the Carnegie Mellon University pronouncing dictionary (Weide, 1996). This dictionary lists the pronunciation of over 125.000 words using a transcription based on the phoneme set shown in Table 5.1. Additionally, the vowels are marked with lexical stress, where `0` indicates no stress, `1` for primary stress, and `2` for secondary stress. For example, the phonetic spelling of *‘dictionary’* is `[D,IH1,K,SH,AH0,N,EH2,R,IY0]`.

Certain entries in this dictionary have more than one phonetic spelling, which normally indicates region and dialect-specific variations. In these cases we have simply taken the first variation available. Moreover, many monosyllabic words appear with both stressed and unstressed forms. We assign closed class monosyllabic words no stress, and open class monosyllabic words primary stress.

We also define a special ‘phoneme’ for punctuation marks that indicate the potential site of a linebreak, e.g.‘.’ (fullstop), ‘,’ (comma), and ‘;’ (semicolon). This is interpreted as the special linebreaking syllable for metre evaluation purposes (see Section 6.3.2 for discussion of this).

The complete listing of information associated with a lexical entry in as follows:

1. Unique key for hashing and reference purposes. This is required as several words with the same orthographic spelling may have different syntactic and even semantic properties.
2. Orthographic spelling.
3. Phonetic spelling. As defined above, this is taken from the CMU pronouncing dictionary.
4. Semantic expression. This is a conjunctively interpreted set of literals as defined in Section 5.2.1.
5. Semantic signature. This is a list of argument variables appearing in the semantic expression.
6. List of names of elementary trees it can anchor.
7. Feature structure detailing the word's syntactic properties.

Table 5.2 shows an example entry for the noun `lion_n`. Its orthographic spelling is *'lion'*. It can anchor the following trees:

1. `I_N_N`: noun phrase tree, e.g.: *"The lion dwells in the waste."*
2. `I_C_NP` copula construction sentence, e.g.: *"Leo is a lion"*.
3. `A_R_C_NP` relative clause variant of the copula construction, e.g.: *"Leo, who is a lion, dwells in the waste"*.

Its semantic expression is $lion(X,Y)$, with its signature being $\boxed{X,Y}$. Note that Y is the variable representing the object entity of the lion. X represents the event entity of Y's 'lion-ness', and is not used in our grammar. The phonetic spelling `[L,AY1,AH0,N]` indicates that it consists of two syllables, the first one receiving primary stress and the second one no stress. Lastly, the feature structure indicates that this word is a third singular noun that is neither a pronoun nor a proper noun. Its selectional restriction information indicates that it is an animate object.

5.4 Individuals

Essentially, a candidate solution, or individual, in our evolutionary algorithm is an LTAG derivation tree that completely describes a syntactic derivation. Its semantics can be derived through a compositional process based on unification of signatures (Section 5.3.8), and its phonetics can be derived from the phonetic spelling of lexical items at the anchor nodes.

When an individual is created at the beginning of an evolutionary run, or at any other point during the run, we execute the **initialization** of an individual. An initialized individual starts off with a derivation tree with one node representing the initial tree shown in Figure 5.14. This is the special 'kickoff' tree, called `I_T_KO` [1], that every individual is initialized with. It represents a derivation rooted by the distinguished symbol, *Poem*, and consists of a single sentence substitution node followed by a sentence boundary lexical anchor (i.e. fullstop). This structure represents an empty text that also specifies the syntactic nature of the required solution, i.e. a poem, which at the very least must consist of a sentence.

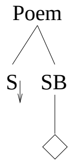


Figure 5.14: `I_T_KO`: "Kickoff tree" for a newly initialized individual

Before being added to the population, this initialized individual is first randomly manipulated by applying to it a sequence of randomly chosen genetic operators from those presented in Chapter 7. Note that no evaluation is applied during this process. This introduces random linguistic content to the individual, and is an implementation of 'randomness within the inviolable constraints that the problem domain specifies' as mentioned in Section 4.2.1. When an entire population is filled with such randomly initialized individuals, it creates genetic diversity, and is expected to cover different portions of the search space.

5.5 Summary

In this chapter we have presented our chosen semantic (Section 5.2), syntactic (Section 5.3.1), and lexical (Section 5.3.9) representational schemes as well as the motivations that underly these choices. We also discussed candidate solutions, or individuals, in our evolutionary algorithm (Section 5.4), which are essentially LTAG derivation trees that completely describe a

[1] We refer the reader to Appendix C.2 regarding the specific grammar constructed for McGONAGALL along with our tree-naming scheme.

syntactic derivation. An individual's semantics can be derived through a compositional process based on unification of signatures (Section 5.3.8), and its phonetics can be derived from the phonetic spelling of lexical items at the anchor nodes. In Chapter 6 we will present methods for evaluating the quality of these features, i.e. semantics and phonetics, and in Chapter 7 we will present functions that manipulate individuals.

Phoneme	Example word	Phonetic spelling
AA	*odd*	AA D
AE	*at*	AE T
AH	*hut*	HH AH T
AO	*ought*	AO T
AW	*cow*	K AW
AY	*hide*	HH AY D
B	*be*	B IY
CH	*cheese*	CH IY Z
D	*dee*	D IY
DH	*thee*	DH IY
EH	*Ed*	EH D
ER	*hurt*	HH ER T
EY	*ate*	EY T
F	*fee*	F IY
G	*green*	G R IY N
HH	*he*	HH IY
IH	*it*	IH T
IY	*eat*	IY T
JH	*gee*	JH IY
K	*key*	K IY
L	*lee*	L IY
M	*me*	M IY
N	*knee*	N IY
NG	*ping*	P IH NG
OW	*oat*	OW T
OY	*toy*	T OY
P	*pee*	P IY
R	*read*	R IY D
S	*sea*	S IY
SH	*she*	SH IY
T	*tea*	T IY
TH	*theta*	TH EY T AH
UH	*hood*	HH UH D
UW	*two*	T UW
V	*vee*	V IY
W	*we*	W IY
Y	*yield*	Y IY L D
Z	*zee*	Z IY
ZH	*seizure*	S IY ZH ER

Table 5.1: Carnegie Mellon University pronouncing dictionary phoneme set

Field	**Value**
Key	`lion_n`
Orthographic spelling	*lion*
Phonetic spelling	`[L,AY1,AH0,N]`
Semantic expression	lion(X,Y)
Semantic signature	$\boxed{X,Y}$
Anchored trees	`I_N_N, I_C_NP, A_R_C_NP`
Feature structure	$\begin{bmatrix} \textsc{cat} & n \\ \textsc{agr} & \begin{bmatrix} \textsc{num} & sg \\ \textsc{pers} & 3 \\ \textsc{3rdsg} & + \end{bmatrix} \\ \textsc{pron} & - \\ \textsc{prop} & - \\ \textsc{sub} & \begin{bmatrix} \textsc{anim} & + \end{bmatrix} \end{bmatrix}$

Table 5.2: Lexical entry for *‘lion’*

Chapter 6

Evaluation functions for poetry generation

In this chapter we describe the two main evaluation functions that we have designed and implemented in MCGONAGALL: metre similarity and semantic similarity, both of which yield a normalized score between 0 and 1 that indicates the degree of isomorphism between a given target structure (i.e. stress list, semantic expression) and the appropriate features of a candidate solution. We validate both evaluators by testing them on sample structures.

6.1 Design considerations

Fitness evaluation is perhaps the most crucial aspect of an evolutionary algorithm, as it is where the bulk of domain knowledge and heuristics is encoded. The suitability of an evaluation function to a given problem has a huge bearing on the ability of an EA to find an optimal (enough) solution, and to find it efficiently.

Recall from Section 4.3 that from our three properties of poetry, we choose to enforce grammaticality through design of representation and genetic operators, and optimize for meaningfulness and poeticness using the EA. In Sections 4.4.1 and 4.4.2 we suggested various features and methods of evaluating the fitness of a text with respect to these properties.

For our implemented system, MCGONAGALL, we choose to evaluate the features of **stress patterns** and **propositional semantics**, and choose to measure them using the ‘score to target’ method, i.e. by measuring the degree of isomorphism between a given target structure, such as stress list or semantic expression, and the appropriate features of a candidate solution.

We make these choices based on the following considerations:

1. These features enable a concrete and objective evaluation, both for the purposes of fitness evaluation within the EA and for our assessment of the performance of the system itself, thereby reducing the need for experimentation involving subjective human evaluation.

2. There already exist well studied algorithms for the computation of structural isomorphism such as the minimum edit distance between two sequences. In this chapter we will describe certain adaptations that we made for our specific purposes, e.g. context sensitivity, asymmetry, etc.

3. The computation of such measures do not require potentially expensive resources such as domain knowledge bases of world knowledge, poetic metre, etc.

At this point, we make no assumptions as to how the individuals to be evaluated are generated, save for the fact that they are guaranteed to be grammatical with respect to the given linguistic resources. However, as we noted in Section 4.5.1, within the interleaved process of an EA evaluation functions may be called upon to provide judgment on partially constructed solutions. Thus, our evaluation functions must be able to handle these grammatically underspecified constructions. Moreover, they must be able to provide a score that reflects the **potential** of a partial solution.

6.2 Decoding and scaling functions

Bäck et al. (1997) state that for an optimization problem $f : M \rightarrow \Re$, where M is the search space of an objective function f and $\Re$ is the set of reals, a fitness evaluation function $\mathcal{F}$ in an EA is defined as follows:

$$\mathcal{F} : R \xrightarrow{d} M \xrightarrow{f} \Re \xrightarrow{s} \Re_+$$
$$\mathcal{F} = s \circ f \circ d$$

where R is the space of the chromosome representation, d is a decoding function, and s is a scaling function to guarantee positive fitness values and fitness maximization.

In Section 5.3.3 we defined our primary data structure as an LTAG derivation tree. Thus, R is the space of all valid derivation trees in our grammar, and d_{metre} and $d_{semantics}$ are decoding functions that derive the lexical stress pattern and conveyed semantics from a derivation tree. d_{metre} returns the concatenation of lexical stress information carried by words at the anchor nodes (Section 5.3.9), and we discuss this in more detail in Section 6.3.2. Likewise, $d_{semantics}$ returns the union of semantic expressions carried by words at the anchor nodes after the necessary signature unification as described in Section 5.3.8.

Finally, for the sake of consistency and clarity in the presentation of our results, we design our evaluation functions to yield normalized scores between 0 and 1. Thus, the scaling function s for both our metre and semantic evaluators has domain and range as follows:

$$s : \mathfrak{R} \to [0,1]$$

Details on the specific scaling functions for metre and semantic evaluators are given in the following sections.

6.3 Metre evaluation

Our metre evaluation function must yield a score that indicates the similarity between a given stress pattern, called the **target form**, and the exhibited stress pattern of a candidate solution, called the **candidate form**.

To illustrate the sort of computation that is required, consider the five lines shown in Figure 6.1. Line (1) shows the first line of Grey's *Elegy Written in a Country Churchyard* (Figure 2.2), which is a perfect example of iambic pentameter. Consider this line to exemplify the target form, and lines (2) to (5) represent candidate forms.

(1)	The	**cur**few	**tolls** the	**knell** of	**par**ting	**day**
(2)	The	**day**	**tolls** the	**cur**few of	**par**ting	**knell**
(3)	The	**cur**few	**tolls** the	**knell**		
(4)	There	**once** was a	**man** from Bra-	**zil**		
(5)	A	**me**tre	**patt**ern	**holds** a	**bunch** of	**beats**

Figure 6.1: An example of comparing stress patterns

Ignoring for the moment all other aspects but metre, which line from among (2) to (5) can we say is the most similar to line (1)? Line (2) is simply a rearrangement of the words in line (1), and thus has exactly the same number of syllables and beats. However, the pattern exhibited is different, as there are no weak syllables preceding the second beat, whereas there are two weak syllables preceding the fourth beat. Line (3) consists of the first six syllables of line (1), and clearly falls short of an iambic pentameter. Line (4) could be a starting line for a limerick, and although it has more syllables than line (3), it still only has three beats. Line (5) exhibits the same stress pattern, and thus is clearly the most similar to (1).

We can justify this choice by keeping a count of ‘mistakes’ exhibited by each line. Line (2) has two mistakes: the lack of a weak syllable before the second beat and an extraneous weak syllale before the fourth beat. Line (3) has four mistakes, i.e. it is missing the last four syllables. Line (4) has six mistakes: the same four mistakes of line (3), but additionally, extraneous weak syllables before the second and third beat. Line (5) has no mistakes. Note that one can change the definition of a mistake to arrive at different scores for these lines.

This type of calculation is the basis of the **minimum edit distance**, the measure that we adopt for our evaluation function, which we will describe in the next section.

6.3.1 Minimum edit distance

Our measure is based on the minimum edit distance (Jurafsky and Martin, 2000, Ristad and Yianilos, 1998), a well known model of string similarity that is defined in terms of the cost of operations that are required to transform one string into another. The valid operations are symbol insertion, deletion, and substitution, each of which is assigned a cost. The minimum edit distance between two strings is the minimal sum of costs of operations that transform one string into another.

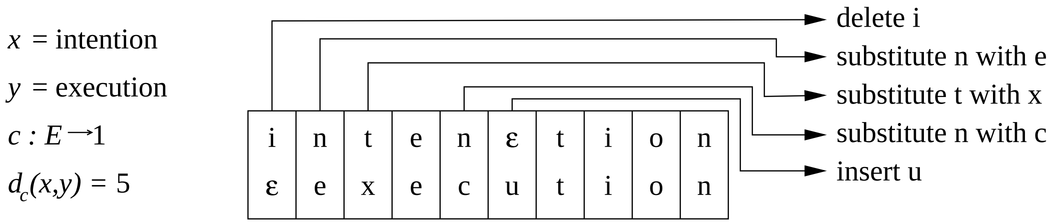
Formally, the edit distance is characterized by a triple $< A, B, c >$, where A and B are finite alphabets of distinct symbols, and $c : E \rightarrow \Re_+$ is the *cost function*. $\Re_+$ is the set of nonnegative reals, and $E = E_s \cup E_d \cup E_i$ is the set of all possible edit operations, $E_s = A \times B$ is the set of substitutions, $E_d = A \times \varepsilon$ is the set of deletions, and $E_i = \varepsilon \times B$ is the set of insertions. Given such a triple, the distance function $d_c : A^* \times B^* \rightarrow \Re_+$ maps a pair of strings to a nonnegative value.

Let $x \in A$ be an arbitrary string over the alphabet A, x_i be the i-th symbol in x, and x^t be the prefix string of length t of x. The distance $d_c(x,y)$ between two strings $x \in A$, of length m, and $y \in B$, of length n, can then be defined recursively as

$$d_c(x^m,y^n) = \begin{cases} c(x_m,y_n) + d_c(x^{m-1},y^{n-1}) \\ c(x_m,\varepsilon) + d_c(x^{m-1},y^n) \\ c(\varepsilon,y_n) + d_c(x^m,y^{n-1}) \end{cases}$$

where $d_c(\varepsilon,\varepsilon) = 0$.

The minimum edit distance between two strings can be represented as a pairwise alignment of the symbols in the strings (plus ε), which indicates the choice of substitution, insertion, and deletion operations that yield the minimum cost. Figure 6.2(a) shows the alignment for the minimum edit distance between the strings *"intention"* and *"execution"*, assuming all operations cost the same. The alignment can also be represented as a list of operations, as in Figure 6.2(b).



x = intention

y = execution

$c : E \rightarrow 1$

$d_c(x,y) = 5$

i	n	t	e	n	ε	t	i	o	n
ε	e	x	e	c	u	t	i	o	n

delete i

substitute n with e

substitute t with x

substitute n with c

insert u

(a) (b)

Figure 6.2: (a) the alignment between *"intention"* and *"execution"* that yields the minimum edit distance, (b) represented as a list of operations (example from Jurafsky and Martin (2000))

The minimum edit distance can be computed in $O(m.n)$ using dynamic programming, a method of efficiently computing values from subcomputations of smaller problems. The algorithm is given in Algorithm 6.1. The EDITDISTANCE function takes two strings, x and y, and returns the minimum edit distance $d_c(x,y)$. The LENGTH function returns the length of a string, and MIN returns the minimum value of its arguments.

By augmenting this algorithm to not only store the d_c value at each cell but also a pointer to the cell which yields the minimum value among $D[i-1,j] + c(x_i,\varepsilon)$, $D[i-1,j-1] + c(x_i,y_j)$, and $D[i,j-1] + c(\varepsilon,y_j)$, we can recover the alignment of the two strings by tracing the pointer at $D[m,n]$ backwards until we reach $D[0,0]$.

Algorithm 6.1 function EDITDISTANCE(x, y) returns $d_c(x,y)$

```
$m \leftarrow$ LENGTH($x$)
$n \leftarrow$ LENGTH($y$)
create matrix $D[m+1,n+1]$
$D[0,0] \leftarrow 0$
for $i = 0$ to $m$ do
    for $j = 0$ to $n$ do
        $D[i,j] =$ MIN($D[i-1,j] + c(x_i,\varepsilon)$,
                       $D[i-1,j-1] + c(x_i,y_j)$,
                       $D[i,j-1] + c(\varepsilon,y_j)$)
    end for
end for
return $D[m,n]$
```

6.3.2 Target and candidate form

The target form, F_{target} is a specification of the metrical constraints that we want our optimal solution to satisfy. It is encoded as a list of **target syllables**, the alphabet of which is listed in Table 6.1.

w	Unstressed, or weak, syllable
s	Stressed, or strong, syllable
x	Any syllable
b	Line break

Table 6.1: Alphabet of target syllables

Note that line breaks are encoded as the special line break ‘syllable’, b. Strictly speaking, b is not a syllable in the same sense that w, s, and x are. These three target syllables specify actual lexical content, whereas b is an indicator for formatting purposes.

Figure 6.3 shows example target forms for a limerick, a haiku, and Belloc’s *“The Lion”* poem (Figure 2.6). Note that they are simply flat lists, and we have visually formatted them for readability purposes.

The candidate form, $F_{candidate}$, is a representation of the metre exhibited by an individual’s text. To obtain this, first of all the stress pattern of a word is derived from its phonetic spelling.

```
[w,s,w,w,s,w,w,s,b,
w,s,w,w,s,w,w,s,b,
w,s,w,w,s,b,
w,s,w,w,s,b,
w,s,w,w,s,w,w,s,b]
```

(a)

```
[x,x,x,x,x,b,
x,x,x,x,x,x,x,b,
x,x,x,x,x,b]
```

(b)

```
[w,s,w,w,s,w,w,s,w,w,s,b,
w,s,w,w,s,w,w,s,w,w,s,b,
w,s,w,w,s,w,w,s,w,w,s,b,
w,s,w,w,s,w,w,s,w,w,s,b]
```

(c)

Figure 6.3: Target forms for (a) a limerick, (b) a haiku, and (c) Belloc's *"The Lion"*

Recall from Section 5.3.9 that in the CMU pronouncing dictionary, vowels are marked with lexical stress, i.e.`0` for no stress, `1` for primary stress and `2` for secondary stress. By ignoring consonants and stripping away phonetic information, we are left with basic lexical stress. We then indicate whether the word is monosyllabic or polysyllabic. This is encoded as a list of **candidate syllables**, the alphabet of which is listed in Table 6.2.

Note the special candidate syllable *b* which represents the potential site of a line break. This is carried by certain punctuation marks such as fullstop and comma. As is the case with the target syllable `b`, the candidate syllable *b* is not an actual syllable in the sense that $0_1, 1_1, 0_n, 1_n$, and 2 are, as they represent actual lexical content. Instead, *b* represents a location which provides a possibly natural linebreak, as fullstops and commas usually serve as sentence or clause boundaries, and linebreaks usually occur at such points.

This allows for a very flexible handling of linebreaking anywhere in the text, in contrast to an explicitly defined hierarchical model of metrical forms consisting of lines of syllables, e.g. Gervás (2000, 2001), Diaz-Agudo et al. (2002). For instance, we can vary between the strict adherence to linebreaks coinciding with sentence boundaries and more flexible enjambment by adjusting the costs of substitution, insertion and deletion involving linebreak syllables. This is discussed in Section 6.3.3.

For example, the stress pattern for *'the'* (`[DH,AH0]`), is $[0_1]$, and for *'dictionary'* (`[D,IH1,K,SH,AH0,N,EH2,R,`
it is $[1_n, 0_n, 2_n, 0_n]$.

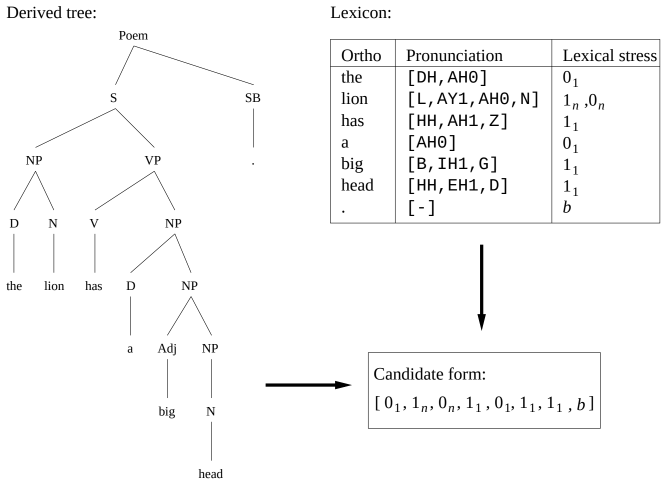
Having derived stress patterns for words, the stress pattern of a candidate solution is simply the concatenation of stress patterns of all the words at the leaf nodes of its derived tree. This process can be seen as the embodiment of the decoding function d_{metre} (Section 6.2).

Figure 6.4 shows the derived tree of an individual representing the text *"The lion has a big*

0_1	Unstressed monosyllabic word
1_1	Stressed monosyllabic word
0_n	Unstressed syllable in polysyllabic word
1_n	Primary stressed syllable in polysyllabic word
2_n	Secondary stressed syllable in polysyllabic word
b	Line break

Table 6.2: Alphabet of candidate syllables

head", and how its candidate form is obtained.



Ortho	Pronunciation	Lexical stress
the	`[DH,AH0]`	0_1
lion	`[L,AY1,AH0,N]`	1_n ,0_n
has	`[HH,AH1,Z]`	1_1
a	`[AH0]`	0_1
big	`[B,IH1,G]`	1_1
head	`[HH,EH1,D]`	1_1
.	`[-]`	b

Figure 6.4: The candidate form is a concatenation of lexical stress

Note that this candidate form only indicates the lexical stress in **citation form** of the words in the derivation. That is to say, the stress of these words as if they are pronounced in isolation of anything else. Lexical stress, however, is amenable to change within certain contexts. A discussion of this along with a description of how we account for it is given in Section 6.3.4.

6.3.3 Operation costs

We use the edit distance algorithm presented in Algorithm 6.1 to compute an alignment from the candidate form to the target form that returns the minimum cost. What is needed now is a definition of the cost function, d_c, that specifies the cost of operations, i.e. insertions of target syllables, deletions of candidate syllables, and substitutions of candidate syllables to target syllables.

The main consideration that should inform the assigning of operation costs is the relative desirability of performing one operation with respect to another in computing the optimal alignment. The most obvious decision is to assign a cost of 0 to substitutions that should naturally occur, namely:

- substituting a 0_1 or 0_n candidate syllable with a w target syllable
- substituting a 1_1 or 1_n candidate syllable with an s target syllable
- substituting any candidate syllable except *b* with an x target syllable
- substituting a *b* candidate syllable with a b target syllable

The next obvious cost is that of "substituting" a $0_1, 0_n, 1_1, 1_n$, or 2 candidate syllable with a b target syllable, or substituting a *b* candidate syllable with either a w, s, or x target syllable. These are illegal operations and must incur a maximum penalty (i.e. ∞) so as never to be chosen. They are illegal because the target b and the candidate *b* are not actual syllables that represent lexical content. Instead, they are merely placemarkers for a formatting aspect of the text. Thus, it does not make sense, for example, to substitute the word *"the"* for a linebreak, the same way that it does not make sense to substitute a comma for an s target syllable. There are only three methods for handling linebreaks:

- Insertion of a target b. This occurs when a linebreak is forced in the text because it is demanded by the target form. This means that the linebreak does not coincide with any clause or sentence boundary, resulting in what is known as **enjambment**, i.e. the running on of a sentence from one line of a poem to the next.
- Deletion of a candidate *b*. This occurs when a punctuation mark representing a sentence boundary, e.g. a comma or a fullstop, is not used as a location for a linebreak. This is quite a normal phenomenon, as lines in poetry are often made up of more than one clause.
- Substituting a candidate *b* with a target *b*. This is when a linebreak is placed at the point of a clause or sentence boundary, resulting in a natural linebreak. As mentioned above, this substitution incurs a cost of 0.

The remaining operations should incur some form of penalty, as they result in suboptimal alignments. This specification of costs is by no means trivial, and one possible solution would be to learn values from examples as in Ristad and Yianilos (1998). However, we have implemented a baseline cost function using our intuitions which is detailed in Table 6.3.

Substitution

Cost	w	s	x	b
0_1	0	2	0	∞
0_n	0	2	0	∞
1_1	3	0	0	∞
1_n	3	0	0	∞
2_n	1	1	0	∞
b	∞	∞	∞	0

Insertion

	Cost
w	1
s	3
x	1
b	10

Deletion

	Cost
0_1	1
0_n	1
1_1	3
1_n	3
2_n	2
b	0

Table 6.3: Operation costs for substitution, insertion, and deletion

Specifically regarding linebreaks, we have imposed a penalty of 10 for inserting a line break. This is to enforce the breaking of lines to coincide with sentence boundaries. On the other hand, there is a penalty of 0 for deleting a *b* candidate syllable. This is because *b* is introduced by punctuation marks, and we consider it acceptable, for instance, for sentences to end and begin within a line. In our empirical study (Chapter 8), we examine the effects of varying these costs on the performance of the EA and the quality of the output.

We use the edit distance algorithm primarily for its alignment purposes, i.e. how to best align the syllables in the the two metre patterns. As we will see in the next few sections, there are other factors which affect the metre evaluation function, such as context-sensitive penalties and rewards, score normalization, and accounting for empty substitution nodes, but they rely on the edit distance algorithm's alignment.

6.3.4 Context-sensitive compensation scoring

The candidate forms described in Section 6.3.2 only indicate lexical stress in citation form. However, once words are put into the context of utterances and sentences, and taking into consideration semantic issues such as contrasting and highlighting, stress placement becomes more complex (Thompson, 1980, Hayes, 1995). One example is the phenomenon known as the **rhythm rule**, which is the avoiding of clashes between stressed syllables appearing consecutively. For example, the word *'thirteen'* in citation form is mainly stressed on its second syllable, but in the compound noun phrase *"**thir**teen **men**"*, it is stressed on its first syllable.

This shift of stress prevents the clash of two consecutive stressed syllables. Thompson (1980) describes this phenomena as a pressure towards alternating stress.

Most of the existing research in prosody, however, has examined stress placement in conventional texts and in human dialogue. In poetry, however, stress placement is much more flexible and fluid, and often it is the dominant metre that takes precedence over the kind of aspects that have been looked at by prosody researchers. Roberts (1986) suggests the existence of a rhythmic expectation in the minds of poetry readers, allowing them to correctly place stress to fit the metre. For example, if the line *"Gone is our nightingale"* appeared in the context of a predominantly dactylic stanza, we would have no difficulty in recognizing its stress as in (6.1), where the last syllable of *'nightingale'* receives no stress. However, if the same line appeared in the context of a predominantly iambic stanza, its stress would be as in (6.2), where the last syllable of *'nightingale'* now receives stress. Note that its pronunciation in the CMU pronouncing dictionary is `[N,AY1,T,IH0,NG,G,EY0,L]`.

(6.1) ***Gone*** *is our* ***night****ingale*

(6.2) *Our* ***night****in****gale*** *is* ***gone***

These factors are context-sensitive in the sense that the alternation of stress received by a syllable is governed by the stress patterns of its surrounding syllables. Unfortunately, the edit distance algorithm is context-free: operation costs are specified for individual syllables, regardless of where they appear in the string. Therefore, it cannot account for these interactions of metre in sequences of syllables.

One possible way of accounting for context-sensitivity is by using probabilistic finite state automata models and algorithms such as the Viterbi algorithm (Jurafsky and Martin, 2000), which can be seen as generalizations of the edit distance algorithm. However, we have chosen to implement a simpler solution that is less powerful and computationally less expensive than the Viterbi algorithm, but offers a good enough account of context-sensitive features in metre.

We iterate through an alignment as produced by the edit distance algorithm, and search for occurrences of pre-defined patterns. Figure 6.5 shows the pattern of the destressing of a stressed syllable due to it appearing just after an already stressed syllable. This is present in the last line of Belloc's *"The Lion"*, where the first syllable of 'little' is destressed as the preceding word, 'good', is already stressed. As $c(1_n, \mathtt{w}) = 3$, a compensation score of -1 is added to d_c. A list of all compensation patterns and scores is given in Appendix B.

Note that in this approach, we account for the stress received by the last syllable of *'nightingale'* in (6.2) mainly as a result of it being flanked by unstressed syllables, and not as a result of an iambic rhythmic expectation.

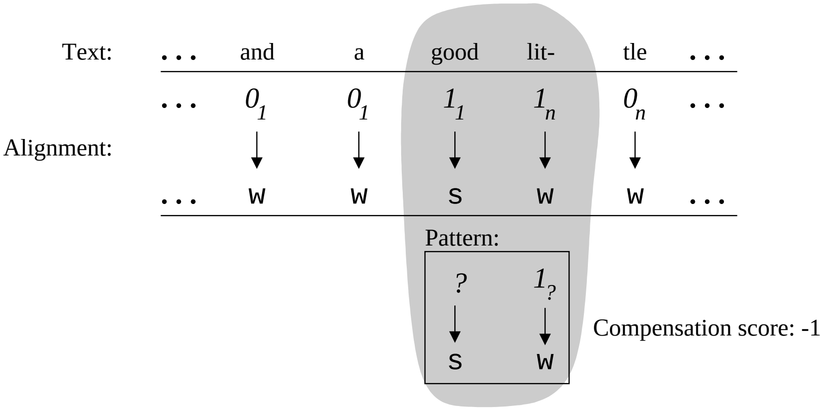


Figure 6.5: An alignment is pattern-matched against pre-defined patterns with associated compensation scores

6.3.5 Normalization

As stated in Section 6.2, we require our evaluators to yield scores of reals between 0 and 1. Hence, we must apply normalization to our objective function, in this case the edit distance $d_c(F_{target}, F_{candidate})$. Marzal and Vidal (1993) introduce the normalized edit distance algorithm, which measures the distance between two strings with respect to the sizes of the strings being compared. For instance, two operations in a comparison between strings of length 3 are more important than three errors in a comparison of strings of length 9. They also show that this cannot be computed by first computing the standard edit distance, and by later on applying a normalizing factor, e.g. the length of the strings. However, if we make the fair assumption that F_{target} remains constant throughout an EA run, and hence at least one string in the comparison is known to be of a given length, then the standard edit distance algorithm we have already presented earlier should suffice. This is because what we are interested in is only the relative merit of individuals in a population in satisfying F_{target}.

Thus, to obtain a normalized score in the interval [0,1], we use the scaling function

$$s_{edit}(x) = \frac{\lambda_1 |F_{target}|}{\lambda_1 |F_{target}| + x}$$

where $|F_{target}|$ is the number of syllables in F_{target}, x is the value of our objective function, and $\lambda_1 > 0$ is a parameter for controlling how strict conformance to the target metre should be.

For example, given the target form of Belloc's *"The Lion"* as in Figure 6.3(c), which has a length of 48 syllables, Figure 6.6 shows the effect varying λ_1 has on s_{edit}. It plots the objective function against the normalized function s_{edit} for λ_1 values of 0.5, 1.0 and 2.0. As we can see, increasing λ_1 makes the function more 'forgiving' towards edit operations.

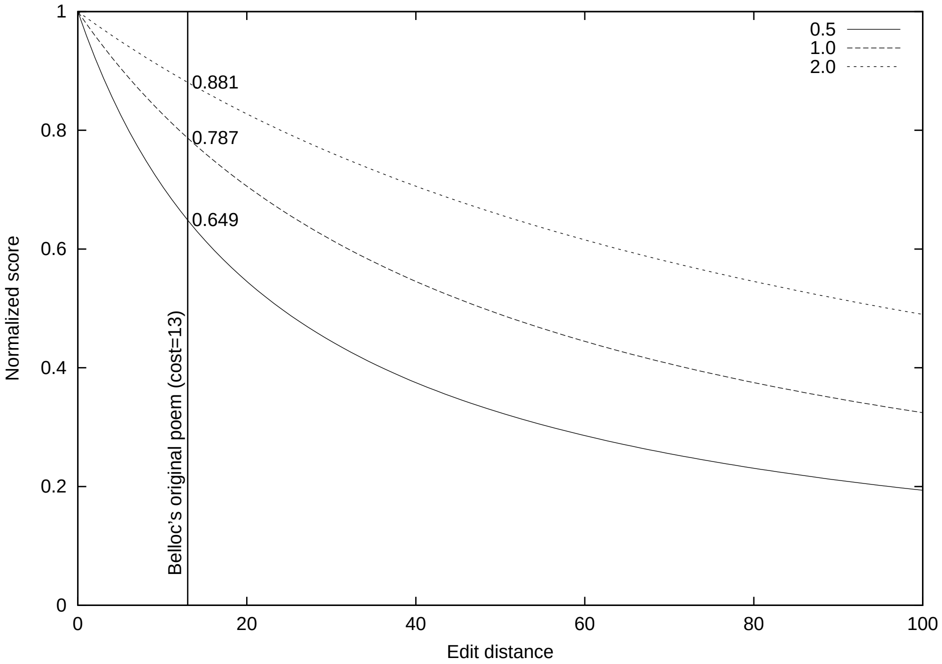


Figure 6.6: The effects of varying λ_1 on s_{edit} for *"The Lion"*

Note that Belloc's original poem itself does not perfectly satisfy the target form as specified in Figure 6.3(c), thus incurring a penalty cost. For instance, several syllables which are lexically stressed are destressed in the context of the poem, e.g.*'big'* and *'small'* in the second line, and the first syllable of *'little'* and *'play'* in the last line. Furthermore, there are extraneous upbeat syllables at the beginning of the last two lines. Given the operation costs defined in Table 6.3, the compensation scores in Appendix B, and the lexical stress as given by the CMU pronouncing dictionary, Belloc's original poem incurs a cost of 13 when compared with the target form in Figure 6.3(c). This is marked in Figure 6.6 as the vertical bar at edit cost = 13. We can see that for $\lambda_1 = 0.5$ it yields a score of 0.649, whereas for $\lambda_1 = 2.0$ it yields a score of 0.881. These scores can be used as a rough yardstick for determining the normalization, i.e. what score does Belloc's poem 'deserve' from the point of view of metre?

Finally, given the characterisation of an evaluation function given in Section 6.2, we define our edit distance-based metre similarity evaluation function, $\mathcal{F}_{edit}$, as follows:

$$\mathcal{F}_{edit}(F_{target}, F_{candidate}) = \frac{\lambda_1 |F_{target}|}{\lambda_1 |F_{target}| + d_c(F_{target}, F_{candidate}) + comp(F_{target}, F_{candidate})}$$

where $comp$ represents the context-sensitive compensation scores we described in Section 6.3.4.

6.3.6 Validation of evaluation function

We will now validate the edit distance-based evaluation function we have described above by applying it towards candidate forms, and their corresponding targets, for which we possess a priori judgments as to their fitness. This is similar to the approach used in Cheng (2002) for the validation of an evaluation metric used in a genetic algorithm-based text planner. We use two sets of candidates that aim to satisfy, respectively, the target forms of iambic pentameter and Belloc's *"The Lion"*.

For the first set, we take the example lines in Figure 6.1 and encode them as the forms `iambic1` to `iambic4` shown in Table 6.4. We have ordered them according to our subjective judgment in terms of how well they achieve the intended target form, starting from `iambic1`, a perfect match, down to `iambic4`. Note that the relative ordering between `iambic3` and `iambic4`, in particular, is contentious: as we stated in Section 6.3, we feel that `iambic4` is inferior due to `iambic3` due to it having "the same four mistakes of [`iambic3`], but additionally, extraneous weak syllables before the second and third beat."

For the second set, we have constructed four candidate forms, `lion1` to `lion4`, also subjectively ordered in terms of how well they achieve the target form (Figure 6.3(c)). `lion1` is Belloc's original poem, `lion2` is an instance of a limerick, taken from Lear (1947), `lion3` is simply the first sentence of the abstract of this thesis, and `lion4` is a trivially inferior form representing the simple sentence *"John loves Mary"*.

Table 6.6 is a summary of the fitness scores obtained by applying the evaluation function to the comparison of these two sets of candidates to their corresponding target forms. We tested our evaluation function both with and without the context-sensitive compensation scoring (Section 6.3.4), and with normalizing factor λ_1 of 0.5, 1.0, and 2.0.

Name	Text	$F_{candidate}$
`iambic1`	*A metre pattern holds a bunch of beats*	$[0_1, 1_n, 0_n, 1_n, 0_n, 1_1, 0_1, 1_1, 0_1, 1_1]$
`iambic2`	*The day tolls the curfew of parting knell*	$[0_1, 1_1, 1_1, 0_1, 1_n, 0_n, 0_1, 1_n, 0_n, 1_1]$
`iambic3`	*The curfew tolls the knell*	$[0_1, 1_n, 0_n, 1_1, 0_1, 1_1]$
`iambic4`	*There once was a man from Brazil*	$[0_1, 1_1, 0_1, 0_1, 1_1, 0_1, 0_n, 1_n]$

Table 6.4: Candidate forms for iambic pentameter

Name	Text	$F_{candidate}$
`lion1`	*The Lion, the Lion, he dwells in the waste. He has a big head and a very small waist. But his shoulders are stark, and his jaws they are grim, and a good little child will not play with him.*	$[0_1, 1_n, 0_n, b, 0_1, 1_n, 0_n, b, 0_1, 1_1, 0_1, 0_1, 1_1, b, 0_1, 1_1, 0_1, 1_1, 1_1, 0_1, 0_1, 1_n, 0_n, 1_1, 1_1, b, 0_1, 0_1, 1_n, 0_n, 0_1, 1_1, b, 0_1, 0_1, 1_1, 0_1, 0_1, 1_1, b, 0_1, 0_1, 1_1, 1_n, 0_n, 1_1, 0_1, 0_1, 1_1, 0_1, 0_1, b]$
`lion2`	*There was an old man with a beard, who said, "it is just as i feared! two owls and a hen, four larks and a wren, have all built their nests in my beard!"*	$[0_1, 0_1, 0_1, 1_1, 1_1, 0_1, 0_1, 1_1, b, 0_1, 1_1, b, 0_1, 0_1, 1_1, 0_1, 0_1, 1_1, b, 1_1, 1_1, 0_1, 0_1, 1_1, b, 1_1, 1_1, 0_1, 0_1, 1_1, b, 1_1, 1_1, 1_1, 0_1, 1_1, 0_1, 0_1, 1_1, b]$
`lion3`	*Poetry is a unique artifact of the human language faculty, with its defining feature being a strong unity between content and form.*	$[1_n, 0_n, 0_n, 0_1, 0_1, 0_n, 1_n, 1_n, 0_n, 2_n, 0_1, 0_1, 1_n, 0_n, 1_n, 0_n, 1_n, 0_n, 0_n, b, 0_1, 0_1, 0_n, 1_n, 0_n, 1_n, 0_n, 1_n, 0_n, 0_1, 1_1, 1_n, 0_n, 0_n, 0_n, 1_n, 1_n, 0_n, 0_1, 1_1, b]$
`lion4`	*John loves Mary.*	$[1_1, 1_1, 1_n, 0_n, b]$

Table 6.5: Candidate forms for Figure 6.3(c)

For the `iambic` candidates, we can see first of all that `iambic1`, as expected, yields a perfect score of 1.0 for all configurations. This is because it incurs a cost of 0. `iambic2` yields the second highest score for all configurations. Note that there is no difference between the scores for `iambic2` whether the compensation scoring is used or not. This is because the operations taken are the insertion of a `w` syllable before the second beat and the deletion of the 0_1 syllable before the fourth beat, i.e. the one introduced by *'of'*. For these operations, no compensation patterns apply. For `iambic3` and `iambic4`, however, the scores are different from our initial subjective judgments. In all configurations, `iambic4` scores better than `iambic3`. Upon closer inspection of the alignments, the operations applied to `iambic4` are the insertions of `s` syllables between the two subsequences of consecutive unstressed syllables. This is exemplified by the

Text	No compensation			With compensation		
	$\lambda_1 = 0.50$	1.00	2.00	0.50	1.00	2.00
`iambic1`	1.000	1.000	1.000	1.000	1.000	1.000
`iambic2`	0.714	0.833	0.909	0.714	0.833	0.909
`iambic3`	0.384	0.555	0.714	0.312	0.476	0.645
`iambic4`	0.454	0.625	0.769	0.714	0.833	0.909
`lion1`	0.649	0.787	0.881	0.649	0.787	0.881
`lion2`	0.511	0.676	0.807	0.522	0.686	0.813
`lion3`	0.387	0.558	0.716	0.369	0.539	0.701
`lion4`	0.200	0.333	0.500	0.152	0.264	0.417

Table 6.6: Fitness scores using default operation costs

line *"There once was* **born** *a man from* **south** *Brazil"*. These operations are different from our characterization of operations above, and in retrospect clearly reveals `iambic4` to be a better candidate than `iambic3`. Note that when context-sensitive compensation scoring is used the gap between `iambic3` and `iambic4` widens, as the consecutive insertions needed for `iambic3` incur further penalties.

Finally, we can see that λ_1 has an effect on the scores, although crucially, the relationship of candidate scores with respect to their rank in the ordering remains constant throughout. In other words, if what we are interested in is simply a relative comparison whether a given candidate is better or worse than another candidate, λ_1 is irrelevant. The fitness assigned to `iambic3` can be used as a rough benchmark for setting the appropriate value of λ_1. Since `iambic3` represents 60% of the optimal target form, it is reasonable to expect that its fitness score should be the same proportion towards an optimal solution. The values for $\lambda_1 = 1.0$ without compensation (0.625) and $\lambda_1 = 2.0$ with compensation (0.55) seem the most appropriate.

For the `lion` candidates, all of the fitness scores confirm our subjective ordering in the sense that each candidate scores higher than the ones ranked below it, with Belloc's original poem, as encoded in `lion1`, scoring the highest. However, it is interesting to see that the context-sensitive compensation scores do not change the scores for `lion1` at all. This is unexpected, as several of the compensation patterns used are designed to account for phenomena that occur in Belloc's poem, such as the destressing of a primary stressed syllable that appears next to a strong syllable, e.g. the first syllable of *'little'* in the last line.

By analyzing the alignment yielded by the edit distance algorithm, we discovered the cause for this issue. As in the case of iambic4 above, the edit distance algorithm yields an alignment that scores better but seems counterintuitive. (6.3) below shows an alignment of the last line as our intuitions expect it to be, and (6.4) shows the alignment as calculated by the edit distance algorithm. Syllables in normal type are either 0_1 or 0_n candidate syllables that are correctly substituted to w target syllables, whereas syllables in bold type are either 1_1 or 1_n candidate syllables that are correctly substituted to s target syllables. Additionally, syllables enclosed in parentheses indicate 0_1 or 0_n candidate syllables that are deleted, and syllables that are underlined indicate 1_1 or 1_n candidate syllables that are substituted to w target syllables, i.e. are destressed. Lastly, an asterisk marks the insertion of a w target syllable.

(6.3) (and) a **good** little **child** * will **not** play with **him**

(6.4) (and) a **good** * * **lit**tle * **child** will not **play** (with) (him)

As can be seen, instead of shifting the stress received by the syllables, it inserts two w syllables between the first and second beats, and then deletes the last two 0_1 syllables, *'with'* and *'him'*. This results in an extremely awkward way of reading the line.

Regarding the compensation scores, the alignment we intuitively expect should yield an initial cost of 19, but a compensation score of -7 results in an edit distance of 12. The actual alignment calculated by the edit distance algorithm, on the other hand, yields an initial cost of 13 with a compensation score of 0. This is because the compensation reward given for substitutions in the first few lines is offset by the penalty incurred for the consecutive deletions and consecutive insertions in the last line. This highlights the drawback of our approach to context-sensitive scoring: as it is handled after the edit distance algorithm, the domain knowledge encapsulated by the compensation patterns are ignored by the greedy nature of the algorithm. The resulting alignment also suggests that the cost of arbitrarily inserting target syllables and deleting candidate syllables should be higher than that specified in Table 6.3.

As a result, we have experimented with a modified cost function, shown in Table 6.7. The costs for substitution are unchanged, but the costs for insertion and deletion are now increased. A summary of the results using this cost function is shown in Table 6.8. Although it does not change the overall behaviour of the evaluation function, i.e. it still ranks the candidate solutions in the same manner as before, we believe that the fitness scores assigned to them are a fairer reflection of their metrical merit. In particular, the fitness score of lion1 is considerably higher than that of `lion2`, whereas in Table 6.6 the difference is not as marked.

Insertion

	Cost
w	3
s	5
x	3
b	10

Deletion

	Cost
0_1	3
0_n	3
1_1	5
1_n	5
2_n	4
b	0

Table 6.7: Modified insertion and deletion costs

Text	No compensation			With compensation		
	$\lambda_1 = 0.50$	1.00	2.00	0.50	1.00	2.00
iambic1	1.000	1.000	1.000	1.000	1.000	1.000
iambic2	0.454	0.625	0.769	0.454	0.625	0.769
iambic3	0.238	0.384	0.555	0.208	0.344	0.512
iambic4	0.333	0.500	0.666	0.454	0.625	0.769
lion1	0.545	0.706	0.827	0.600	0.750	0.857
lion2	0.358	0.527	0.691	0.364	0.533	0.696
lion3	0.276	0.432	0.604	0.329	0.495	0.662
lion4	0.120	0.214	0.353	0.101	0.183	0.310

Table 6.8: Fitness scores using modified operation costs

6.3.7 Scoring potential of incomplete derivations

Up to this point, we already have an evaluation function that adequately measures the similarity between F_{target} and $F_{candidate}$. However, $F_{candidate}$ is simply the concatenation of the lexical stress patterns belonging to the words at the leaf nodes of an individual. This overlooks a crucial aspect, namely substitution nodes that may appear as leaf nodes of incomplete derivations.

As these substitution nodes represent syntactic structures that will eventually affect the metre, this must be reflected in the evaluation function. Consider the following two examples, where $\boxed{X}$ represents a substitution node of category X:

(6.5) $\boxed{NP}$ *resides in the waste.* ($F_{candidate} = [0_n, 1_n, 0_1, 0_1, 1_1, b]$)

(6.6) *The lion is mean.* ($F_{candidate} = [0_1, 1_n, 0_n, 0_1, 1_1, b]$)

If we ignore the distinction of monosyllabic and polysyllabic words, both sentences have an identical $F_{candidate}$. Consider the case where F_{target} = [w,s,w,w,s,b]. Although the edit distance algorithm will find a transformation of cost 0 from $F_{candidate}$ to F_{target} for both sentences, in reality Example (6.6) is better than (6.5), which still needs to realize its subject noun phrase (NP) in order to be grammatical. This substitution will obviously increase its distance to F_{target}.

On the other hand, consider the case where F_{target} = [w,s,w,w,s,w,w,s,b]. Although both examples again yield the same edit distance, in reality Example (6.5) is better than (6.6), as the presence of an NP substitution node shows it has better *potential* to realize the target metre, for instance by substituting the noun phrase *"The lion"*.

What these examples show is that substitution nodes affect, both positively and detrimentally, a text's ability to satisfy F_{target} in a manner which our edit distance-based evaluation function does not cover.

One simple solution is to penalize the presence of substitution nodes. This would work for the first case above, where F_{target} = [w,s,w,w,s,b], but not for the second case, where F_{target} = [w,s,w,w,s,w,w,s,b]. Moreover, as there is pressure to eliminate substitution nodes, there would be a preferential bias to substitute these nodes with structures that did not introduce even more substitution nodes. This could result in a premature 'stunting' of syntactic trees.

We believe that an ideal account of substitution nodes requires a **prediction** of the lexical stress patterns of the syntactic constituents eventually being substituted at these substitution nodes. However, predicting this is extremely difficult. Fortunately, we can approximate this with the slightly easier prediction of the length, or syllable count, of these constituents instead. The idea is that there must be a balance between the syllable count of F_{target} and the sum of the syllable count of $F_{candidate}$ and the total predicted syllable count of these constituents. If l_{target} and $l_{candidate}$ are the lengths, i.e. syllable count, of the target and candidate forms, and $l_{estimate}$ is the total predicted syllable count of constituents substituting at the substitution nodes, then ideally we want to minimize $|l_{target} - (l_{candidate} + l_{estimate})|$, where $|X|$ is the absolute value of X. If this equals zero then we have roughly the 'right amount' of syntactic structure to generate a text that satisfies our target metre.

Thus we define the 'syntax-surface balance' evaluation function, using a similar normalization approach used for $\mathcal{F}_{edit}$ in Section 6.3.5:

$$\mathcal{F}_{balance} = \frac{\lambda_2.l_{target}}{\lambda_2.l_{target} + |l_{target} - (l_{candidate} + l_{estimate})|}$$

where $\lambda_2 > 0$ is a parameter for controlling how strict conformance to the target metre syllable count should be.

This raises the question of how to obtain $l_{estimate}$. We suggest two different approaches that can be adopted: a target-driven and a resource-driven approach. A target-driven approach would estimate the syllable count for each category of substitution node from a corpus of example texts that satisfy the target metre. Thus, for example, if we were to satisfy the target form in Figure 6.3(c), we could estimate these values from Belloc's original poem. Table 6.9 gives one such estimation. It lists the different categories that may appear as substitution nodes in an incomplete derivation along with the estimated syllable count, and some example phrases taken from *"The Lion"*. A resource-driven approach would estimate these values from the linguistic resources used, i.e. grammar and lexicon. Note that some assumptions must be made regarding the recursive structures, e.g. noun phrases. For example, we could place an upper bound on the depth of adjunction. We have not attempted this approach.

Category	$l_{estimate}$	Example
$\boxed{S}$	7	*his shoulders are stark, he has a big head and a very small waist*
$\boxed{NP}$	3	*the lion, he, a good little child*
$\boxed{P}$, $\boxed{Aux}$, $\boxed{D}$, $\boxed{CV}$, $\boxed{Comp}$	1	*in, with, will, the, a, is, are, that*
$\boxed{Punc}$	0	*comma, fullstop*

Table 6.9: Sample $l_{estimate}$ values

Table 6.10 shows an example of applying $\mathcal{F}_{balance}$ on various candidate forms, given $F_{target} = [\mathtt{w},\mathtt{s},\mathtt{w},\mathtt{w},\mathtt{s},\mathtt{w},\mathtt{w},\mathtt{s}]$ ($l_{target} = 8$). We set $\lambda_2 = 1$ and use the $l_{estimate}$ values in Table 6.9. The table lists the values for $l_{candidate}$, $l_{estimate}$, and $\mathcal{F}_{balance}$. For comparison, it also lists the scores given by the edit distance-based evaluation function, $\mathcal{F}_{edit}$.

Note that the first two texts satisfy F_{target} perfectly, as shown by the 1.0 score for $\mathcal{F}_{edit}$. However, the second text is an incomplete derivation and has a lot of substitution nodes. Thus, $\mathcal{F}_{balance}$ correctly scores it considerably lower.

The last two texts are also incomplete derivations, but their substitution nodes indicate potential

Text	$l_{candidate}$	$l_{estimate}$	$\mathcal{F}_{balance}$	$\mathcal{F}_{edit}$	$\mathcal{F}_{metre}$
There once was a man from Brazil.	8	0	1.0	1.0	1.0
$\boxed{NP}$ *is* $\boxed{NP}$. $\boxed{NP}$ *read it* $\boxed{P}$ *the child.* $\boxed{NP}$ *will be grim.*	8	13	0.615	1.0	0.884
There once was a $\boxed{NP}$.	4	3	0.889	0.333	0.499
$\boxed{S}$.	0	7	0.889	0.21	0.413

Table 6.10: Example of applying $F_{balance}$

for further realization. Their values for $\mathcal{F}_{balance}$ represent this potential, whereas their values for $\mathcal{F}_{edit}$ represent their currently poor ability to approximate the target form.

Note that $\mathcal{F}_{balance}$ and $\mathcal{F}_{edit}$ are in fact two separate evaluation functions. We have briefly discussed the issue of multi-objective evaluation in Sections 4.2.2 and 4.4.3. In our empirical studies (Chapter 8) we adopt the simple mechanism of linearly combining different evaluation scores. We show here how this approach would work for these two evaluators by the following linear combination:

$$\mathcal{F}_{metre} = \lambda_3 \mathcal{F}_{edit} + (1 - \lambda_3)\mathcal{F}_{balance}$$

where $0 \leq \lambda_3 \leq 1$ is a parameter for controlling the relative weight of the edit distance measure with respect to the syntax-surface balance score. As we consider the edit distance-based evaluation function to be the primary measure, we feel λ_3 should be suitably high. The last column of Table 6.10 shows values for $\mathcal{F}_{metre}$ given $\lambda_3 = 0.7$.

Unfortunately $\mathcal{F}_{edit}$ and $\mathcal{F}_{balance}$ are not orthogonal, as in the case when there are no substitution nodes remaining, a candidate's failure to satisfy the target metre due to it simply being too short will be penalized in both values. In the case of using genetic operators that guarantee complete derivations (Section 7.4.1), λ_3 should be set to 1.0.

6.4 Semantic Similarity

In Section 4.4 we suggested several possible methods for measuring fidelity of a generated text, i.e. similarity, meaningfulness, consistency, etc. In this section, we concentrate on the only method that we have designed and implemented: measuring the similarity between the semantics of a generated text and that of a given target semantics. This is similar to the evaluation function we use for measuring metre similarity. Both functions are a measure of isomorphism between an individual's features and a defined target structure. In the case of metre, we measure the similarity between the metre pattern exhibited by an individual and the given target metre, using the edit distance algorithm. The semantic similarity function yields a score measuring the similarity between the semantics conveyed by an individual, which we call the **candidate semantics**, and the given **target semantics**.

Both the candidate and target semantics use the flat semantic representation defined in Section 5.2.1. The candidate semantics are obtained from an individual using the compositional rules described in Section 5.3.8, whereas the target semantics are provided as input to the system.

In the following sections, we will first introduce a model of semantic similarity as a process of structural alignment, before detailing our implemented greedy algorithm that computes a structurally consistent mapping in polynomial time, and a function that computes the similarity score from the resulting mapping.

6.4.1 A model of semantic similarity

As an example of what we are trying to achieve, consider the following expressions:

(6.7) $love(l,j,m) \wedge john(\_,j) \wedge mary(\_,m)$ ("*John loves Mary.*")

(6.8) $adore(a,x,y) \wedge mary(\_,x) \wedge applepie(\_,y)$ ("*Mary adores apple pie.*")

(6.9) $run(r,s) \wedge sprinter(\_,s)$ ("*The sprinter runs.*")

Which pairs of expressions are considered more similar than the others? What issues motivate this choice? Are (6.7) and (6.8) considered most similar due to the synonymity between *love* and *adore*, or perhaps because they both convey information about *mary* (that we can assume to be the same object)? Does the fact that they both state a binary predication over two objects, each of which is further elaborated by a unary predicate, play a factor?

Love (2000) proposes a computational theory of similarity that is psychologically grounded. It states that the similarity between two objects is a function not only of their common and distinctive features, as is established in earlier accounts of similarity such as Tversky (1977), but also of the higher-order compatibility relations among these features. For example, Falkenhainer et al. (1989) claim that people judge the solar system and an atom as being similar not because the elements of the two systems are inherently similar (the sun and an atom's nucleus differ in size and composition), but because our representation of these two systems share a number of higher order relational matches, such as the revolving of electrons around a larger nucleus corresponding with the revolving of planets around a larger sun.

Following Love (2000), we propose two factors that our evaluation function must take into consideration: **structural similarity** and **conceptual similarity**.

1. Structural similarity is a measure of isomorphism between the structure of two semantic expressions. Do their literals convey the same predicate-argument structure? Do they attribute conceptual predications to the same object variables?

   An algorithm for comparing two semantic expressions is necessarily more complex than that of the string edit distance algorithm for metre similarity, as there is higher-order structure inherent within the propositions. One approach that can be used is the graph distance metric based on the computation of the **maximal common subgraph** (Bunke and Shearer, 1998). The graph isomorphism problem is known to be NP-hard. Another approach that can be used is Gentner's structure mapping theory (Gentner, 1983), which is employed in the Structure Mapping Engine, or SME(Falkenhainer et al., 1989).

2. Conceptual similarity is a measure of relatedness between two concepts, in this case literal functors. The simplest instance of conceptual similarity is identity: 2 literals are the same if they share the same functor, e.g.*mary* in (6.7) and (6.8).

   A more sophisticated approach than this binary distinction can be achieved through the comparison of concepts with respect to some underlying ontology. A general purpose ontology such as WordNet (Fellbaum, 1998) can be employed, as can existing concept-relatedness metrics that have been developed for WordNet (Foo et al., 1992, Beeferman, 1998), which measure path distance in the hypo/hyper-nymy tree. For instance, the functors *love* and *adore* would be more closely related than *mary* and *applepie*.

As mentioned in Section 4.4.1, Nicolov (1998) presents a generation system that can convey more or less than the target semantics, and thus requires a way of preferring one text over another based on how similar its semantic representation is to the target semantics. Nicolov constructs a relation $<_G$ which holds between two semantic representations g_1 and g_2 ($g_1 <_G g_2$) if and only if g_2 is a better match for the input semantics G than g_1 is. This is slightly different from our approach, where rather than devising a metric that calculates an absolute distance between two semantic representations, Nicolov defines a relation between two semantic representations in relation to a third, i.e. the input semantics. Nevertheless, the intuitive notion of semantic similarity is essentially the same as our own model. The definition of $<_G$ takes into account both structural similarity, through the use of the maximal projection operator applicable to conceptual graphs, and conceptual similarity, through the calculation of distance between two corresponding concepts in the maximal projection.

Love (2000) notes that although most current models of comparison and analogy account for these aspects of similarity, it is not clear how they are weighted and manifested. Providing a clear model of the relationship between structural and conceptual similarity is one of the key contributions of Love (2000). It specifies a computational level theory of similarity that takes the form of a linear combination of four terms, which we discuss in Section 6.4.3.

This similarity equation requires the establishment of correspondences, or mappings, between the features and components of two objects, and these correspondences are chosen so as to maximize the similarity equation. Thus, quoting Love (2000): “Similarity both drives the comparison process and is an outcome of it”. What this means is that there are two distinct processes: the establishment of mappings and the evaluation of its ‘quality’. Love’s model is primarily concerned with the latter, and makes no assumptions regarding the former. Gentner’s structure mapping theory (Gentner, 1983), which SME is based on, is one method of computing such mappings.

In Section 6.4.2, we describe our implemented algorithm of computing an optimal mapping between two semantic expressions for the purpose of evaluating their similarity. It is based on the algorithm of SME. In Section 6.4.3, we describe our implemented semantic similarity evaluation function which assigns a score to such a mapping. It is based on Love’s model.

6.4.2 Mapping two semantic expressions

Before describing our algorithm, we will first define several important concepts. Firstly, recall from Section 5.2.1 that a semantic expression, S, is an unordered set of literals, l.

- **Definition 1.** For any two semantic expressions S_1 and S_2, $M \subseteq S_1 \times S_2$ is a **mapping** between them.

  From this point onwards, we also assume that we are constructing a mapping between S_{target}, the target semantics, and $S_{candidate}$, the candidate semantics.

- **Definition 2.** In more detail, M can be seen as a set of **local matches**, where a local match is a triple, $m \in M$, of either of the following type:

  - **Proper match**: $m = (l_t, l_c, b)$, where $l_t \in S_{target}$, $l_c \in S_{candidate}$, and corresponding arguments of l_t and l_c that appear in the same position are bound to each other in b, a **dictionary**. A dictionary is an abstract data type that stores items, or keys, associated with values. In AI research, they are also commonly referred to as association lists. We use the notation $b(x) = y$ to indicate that the value y is stored for key x in dictionary b. In our case, both keys and values are argument variables. Thus, for our argument binding purposes, if $arg_n(l)$ is the n-th argument of l, then for all n, $b(arg_n(l_c)) = arg_n(l_t)$ and $b(arg_n(l_t)) = arg_n(l_c)$. Because of this "mirroring" effect, where $b(x) = y$ iff $b(y) = x$, we can use the more compact notation $b = \{x_1 \Leftrightarrow y_1, \ldots, x_n \Leftrightarrow y_n\}$, which states that dictionary b stores values $y_1, \ldots, y_n$ for keys $x_1, \ldots, x_n$ respectively, and values $x_1, \ldots, x_n$ for keys $y_1, \ldots, y_n$ respectively.

    For example, if $l_t = love(l, j, m)$ and $l_c = adore(a, x, y)$, then for proper match $m = (l_t, l_c, b)$, dictionary $b = \{l \Leftrightarrow a, j \Leftrightarrow x, m \Leftrightarrow y\}$.

  - **Dangling match**: $m_i = (l_{t_i}, \varepsilon, \varepsilon)$ or $m_i = (\varepsilon, l_{c_i, \varepsilon})$. The former is a **target literal dangling match**, and the latter a **candidate literal dangling match**. These simply represent literals that are not part of any proper matches. They exist because we require a mapping to possess a local match for every literal in the semantic expressions being mapped.

  Finally, for all $m_i, m_j \in M$, $l_{t_i} \neq l_{t_j}$ and $l_{c_i} \neq l_{c_j}$. That is to say, a literal can only be matched within a mapping once.

- **Definition 3a.** A mapping M is said to be **structurally consistent** if for every proper match $m_i, m_j \in M$ and for all argument variables x, $b_i(x) = b_j(x)$, unless either $b_i(x) = \varepsilon$ or $b_j(x) = \varepsilon$ (i.e. x is not bound by m_i or m_j). In other words, there is no argument variable within M that is bound to two different variables.

  For example, given the following two mappings between the semantic expressions of (6.7) and (6.8) above:

$$\begin{aligned} M_1 &= \{(love(l,j,m), adore(a,x,y), \{l \Leftrightarrow a, j \Leftrightarrow x, m \Leftrightarrow y\}), \\ &\quad (mary(\_,m), applepie(\_,y), \{m \Leftrightarrow y\})\} \\ M_2 &= \{(love(l,j,m), adore(a,x,y), \{l \Leftrightarrow a, j \Leftrightarrow x, m \Leftrightarrow y\}), \\ &\quad (mary(\_,m), mary(\_,x), \{m \Leftrightarrow x\})\} \end{aligned}$$

we can see that M_1 is structurally consistent whereas M_2 is not. In M_2, the first proper match binds $j \Leftrightarrow x$ and $m \Leftrightarrow y$, whereas the second proper match binds $m \Leftrightarrow x$. Note that in these mappings we are only showing the proper matches for better readability.

- **Definition 3b.** A proper match m is said to be structurally consistent with respect to a mapping M if M', the resultant mapping of adding m to M, is structurally consistent.

  For example, the proper match $m = (john(\_,j), mary(\_,x), \{j \Leftrightarrow x\})$ is structurally consistent with respect to M_1 above.

- **Definition 4.** If $|M|$ is the number of proper matches in M, a structurally consistent mapping M_i is a **maximal structurally consistent mapping** if there exists no structurally consistent mapping M_j such that $|M_j| > |M_i|$. Note that $|M| \leq min(|S_{target}|, |S_{candidate}|)$, because a literal can only be matched within a mapping once, thus there can only be as many proper matches as there are literals to be matched. Note also that for any S_{target} and $S_{candidate}$ there may be more than one maximal structurally consistent mapping.

The process of establishing correspondencies between two semantic expressions is one of structural alignment. Two examples of this are the maximal common subgraph algorithm and the structure mapping algorithm of SME mentioned above.

However, the maximal common subgraph algorithm is only concerned with finding a maximal structurally consistent mapping. Strictly speaking, this is not the measure we are seeking: it corresponds to the maximization of structural similarity, but remember that our similarity equation also takes into account conceptual similarity.

SME accounts for this by first constructing a set of **local match hypotheses**, which are individual correspondences between features of the two objects being compared that satisfy given constraints. These constraints are defined by rule sets which govern what is allowed to match. In Falkenhainer et al. (1989), three types of rule sets are used: literal similarity, analogy, and mere appearance. From these local matches, SME constructs global matches, which are essentially all maximal structurally consistent mappings that can be built from the restricted space of local match hypotheses. Thus, both conceptual and structural similarity are accounted for efficiently. Our greedy algorithm adopts a similar approach.

Consider the following mappings between the semantic expressions of (6.7) and (6.8) above:

(6.10) $$M_1 = \{(love(l,j,m), adore(a,x,y), \{l \Leftrightarrow a, j \Leftrightarrow x, m \Leftrightarrow y\})\}$$

(6.11) $$\begin{aligned} M_2 = \{&(love(l,j,m), adore(a,x,y), \{l \Leftrightarrow a, j \Leftrightarrow x, m \Leftrightarrow y\}),\\ &(mary(\_,m), applepie(\_,y), \{m \Leftrightarrow y\}),\\ &(john(\_,j), mary(\_,x), \{j \Leftrightarrow x\})\}\end{aligned}$$

(6.12) $$M_3 = \{(mary(\_,m), mary(\_,x), \{m \Leftrightarrow x\})\}$$

(6.13) $$\begin{aligned} M_4 = \{&(mary(\_,m), mary(\_,x), \{m \Leftrightarrow x\}),\\ &(love(l,j,m), adore(a,x,y), \{m \Leftrightarrow x, l \Leftrightarrow a, j \Leftrightarrow x, m \Leftrightarrow y\})\}\end{aligned}$$

M_1, M_2, and M_3 are structurally consistent mappings, whereas M_4 is not, because the variable m is mapped to both x and y, and likewise, x is mapped to both m and j.

Our initial attempts at the mapping process included an exhaustive search through all possible mappings, and a greedy algorithm that sought to maximize conceptual similarity. The first attempt proved unfeasible due to the combinatorial explosion of the number of possible mappings. The second attempt worked by repeatedly choosing the local match that yields the highest conceptual similarity score between the mapped literals. The evaluation function would then score the quality of the whole mapping, taking into account both conceptual and structural similarity. Unfortunately this proved to be a faulty heuristic, as exemplified by the mappings shown above. As the heuristic initially chooses to map the two *mary* literals due to their conceptual similarity (resulting in the M_3 mapping), there is then no way of incorporating *love* and *adore* in a structurally consistent manner, i.e. M_4 is structurally inconsistent. Note that the maximal structurally consistent mapping is M_2, and that it does not map the two *mary* literals together.

To summarize, a mapping algorithm that simply maximizes the conceptual similarity of local matches can yield suboptimal mappings from a structural viewpoint. On the other hand, M_2 does not seem to intuitively convey the similarity in meaning between the candidate and target semantics. It is difficult to imagine a situation in which matching *mary* and *applepie* is warranted. In SME, this would be prevented by the application of the rule sets in constructing local match hypotheses. They define what local matches are conceptually acceptable. Similarly, we define a function $c(m)$ that returns the conceptual similarity between the functors of the literals mapped in proper match m, and determine c_{min} as the minimum threshold value of conceptual similarity for matches to be considered acceptable. Our currently implemented function is simply that of identity, where $c_{min} = 1$, and given $m = (l_t, l_c)$,

$$c(m) = \begin{cases} 1 & \text{if } functor(l_t) = functor(l_c), \text{ where } functor(X) \text{ is the functor of literal } X, \\ 0 & \text{otherwise.} \end{cases}$$

However, there is still a problem in using such a hillclimbing procedure to map literals together. Consider the following target and candidate semantics, which both represent the meaning conveyed by the utterance *"The lion has a head and a waist"*:

- $S_{target} = \{lion(\_, l), own(\_, l, h), own(\_, l, w), head(\_, h), waist(\_, w)\}$
- $S_{candidate} = \{lion(\_, a), own(\_, a, c), own(\_, a, b), head(\_, b), waist(\_, c)\}$

Clearly the candidate semantics conveys the target semantics perfectly well, as they are in fact isomorphic. However, as there are two literals with the functor *own* in both S_{target} and $S_{candidate}$, there is an ambiguity in how they should be mapped. If the wrong mapping is made, i.e. $(own(\_, l, h), own(\_, a, c))$, then only $(head(\_, h), waist(\_, c))$ can be considered as a structurally consistent proper match, when clearly $(head(\_, h), head(\_, b))$ is the ideal match.

We can account for this problem by deferring the matching of ambiguous literals until more information is known. Also, the chosen mapping must be sensitive to the structural consistency of all bound variables known at that point.

We can now describe our algorithm as such:

Starting with an empty mapping M, we firstly build a pool of all **acceptable** proper matches, i.e. proper matches that

- are structurally consistent with respect to M, and
- yield a conceptual similarity score higher than a given threshold, i.e. $c(m) \geq c_{min}$.

Next, we select the best proper match from this selected pool by using a prioritized list of criteria, shown in Table 6.11. Each subsequent criteria is only considered if the previous criteria cannot uniquely determine a single proper match. This approach is similar to the ranking of constraints in optimality theory (Prince and Smolensky, 1993), and is also adopted by the greedy algorithm of SPUD (Stone et al., 2001). If upon consideration of the last criteria there is still more than one proper match, randomly select either.

- **Criteria 1 - Unambiguity**: prefer proper matches that map literals that also appear in the fewest other proper matches in the pool.
- **Criteria 2 - Connectedness**: prefer proper matches that map the most unbound argument variables.
- **Criteria 3 - Conceptual bias**: prefer proper matches that yield the highest conceptual similarity score.

Table 6.11: Prioritized criteria for selecting best proper match

The selected proper match is added to M, the mapped literals are removed from S_{target} and $S_{candidate}$ respectively, and the process is repeated until no more proper matches can be found, at which point all remaining literals are added as dangling matches.

This mapping algorithm is specified in Algorithms 6.2 and 6.3.

Algorithm 6.2 `Mapping greedyMap(Semantics` T`, Semantics` C`, Mapping` M`)`

```
$m = (l_t, l_c) \leftarrow$ bestMatch($T$,$C$,$M$)
if $m \neq$ null then
  $M = M + m$
  $T = T - l_t$
  $C = C - l_c$
  return greedyMap($T$,$C$,$M$)
else
  for $i \in T$ do
    $M = M + (i, \varepsilon)$
  end for
  for $i \in C$ do
    $M = M + (\varepsilon, i)$
  end for
  return $M$
end if
```

We will now summarise the similarities and differences between our mapping algorithm that we have just described and SME, which we base our algorithm on. Essentially, both follow

Algorithm 6.3 `Match bestMatch(Semantics` T`, Semantics` C`, Mapping` M`)`

```
$Pool = \{\}$
for $i \in T$ do
  for $j \in C$ do
    $m \leftarrow (i, j)$
    if $m$ is consistent w.r.t. $M$ and $c(m) > c_{min}$ then
      $Pool = Pool + m$
    end if
  end for
end for
select $m \in Pool$ according to prioritized criteria in Table 6.11
return $m$
```

the same approach of constructing a set of possible local matches (i.e. match hypothesis in SME, acceptable proper matches in our algorithm) and then collecting these local matches into a maximal structurally consistent global match, or *gmap*s in SME.

However, there is a difference in the construction of global matches. SME employs a more powerful algorithm that constructs all valid structurally consistent *gmap*s, whereas our hill-climbing algorithm which uses the prioritized criteria in Table 6.11 is deterministic. In terms of complexity, the worst case complexity of global match construction in SME is $O(N!)$, whereas in our algorithm it is $O(N^3)$. This can be observed from Algorithms 6.2 and 6.3. The `greedyMap` function simply takes the best local match suggested by `bestMatch`, recursing until the semantics in T and C have been consumed, thus resulting in a worst case of $O(N)$. `bestMatch` iterates over T and C in constructing acceptable local matches and has a worst case complexity of $O(N^2)$. As a result, the worst case complexity for our global match construction is $O(N^3)$.

There is also a difference in what happens after the global match construction procedure. Whereas we proceed to score the mapping using our evaluation function based on Love's model (Section 6.4.3), SME performs a calculation of candidate inferences, i.e. a set of inferred expressions that are suggested to hold by the comparison represented by a *gmap*.

Finally, SME has a flexible method of computing the initial set of local matches by using a set of match rule constructors, i.e. rules that specify which local matches are plausible. By using different sets of rules, they are able to use SME to test different theories of analogy, i.e.

analogy, literal similarity, and mere appearance. This is indeed one of the aims of SME, i.e. as a tool to test Gentner's structure mapping theory. In contrast, we use the simple $c(m)$ function which is currently implemented as simple identity.

6.4.3 An evaluation function for semantic similarity

Once we have found a mapping between literals in the target semantics and literals in the candidate semantics, we need an evaluation function that takes the set of mappings and assigns a fitness score to it. Our evaluation function is based on Love's computational model of similarity, which defines similarity as the following linear combination (Love, 2000):

$$F(S_{target}, S_{candidate}, M) = \alpha_1 \Theta(S_{target}, S_{candidate}) + \alpha_2 \Upsilon(M) + \alpha_3 \Omega(M) + \alpha_4 \Phi(M)$$

where $\alpha_1, \alpha_2, \alpha_3, \alpha_4 \geq 0$

Note that although we use the same four terms as Love's similarity equation, our formulation for each term is different due to our domain-specific knowledge of the structures being compared, i.e. semantic expressions.

The first term, Θ, compares 'raw' conceptual similarity, and corresponds to the contrast model of Tversky (1977). It does not score any aspect of the mapping M, but rather examines the occurrence of similar elements present in both the semantic expressions S_{target} and $S_{candidate}$. For example, if the two semantic expressions being compared are (6.7) and (6.8), this term captures the occurrence of *mary* in both expressions, regardless of how they are mapped. The equation for Θ is as follows:

$$\Theta(S_{target}, S_{candidate}) =$$
$$\varphi_1 f(S_{target} \cap S_{candidate}) - (1 - \varphi_1)(\delta_1 f(S_{target} - S_{candidate}) + (1 - \delta_1) f(S_{candidate} - S_{target}))$$

where $1 \geq \varphi_1 \geq 0$, and $1 \geq \delta_1 \geq 0$.

φ_1 determines the relative importance of commonalities and differences, and δ_1 allows control over how asymmetric the similarity judgment between the two expressions is, and f is a function over sets that is related to the importance of the elements in a set. We follow Love in making the simplifying assumption that all elements are equally salient, and thus $f(X) = |X|$.

Essentially, to compute Θ we must partition the literals into 3 sets: $S_{target} \cap S_{candidate}$, $S_{target} - S_{candidate}$, and $S_{candidate} - S_{target}$. We use our conceptual threshold value, c_{min}, to decide membership among these sets.

The second term, Υ, measures the similarity arising from correspondences. Unlike Θ, Υ only captures the similarity between literals that are mapped under M. Love cites psychological experiments that show that commonalities and differences arising from correspondences are processed differently from those that are not in correspondence. The equation for Υ is as follows:

$$\Upsilon(M) = \sum_{i=1}^{n} c(m_i)$$

where n is the number of proper matches in M, m_i is the i-th proper match in M.

The third term, Ω, captures higher-order matches, as in the solar system and atom example cited in Falkenhainer et al. (1989). Structural similarity is accounted for by this term. In the case of semantic expressions, a higher order match occurs when when two literals are mapped under a proper match and their arguments contain variables that are similarly mapped by another proper match. The equation for Ω is as follows:

$$\Omega(M) = \sum_{i=1}^{n} \sum_{j=1}^{n} c(m_i)c(m_j)share(m_i, m_j)$$

where and $share(m_i, m_j) = 1$ if $i \neq j$ and l_{t_i} and l_{t_j}, or l_{c_i} and l_{c_j}, share a common argument variable, 0 otherwise.

The fourth term in Love's similarity equation, Φ, is also a structural measure, which calculates the degree to which the mapping is a bijection, or one-to-one mapping. Note that since our greedy algorithm enforces a one-to-one mapping, we do not require the full complexity of Love's equation for Φ. However, we still require a term that measures the degree of completeness of the mapping, i.e. how many dangling matches are in M. Thus, the equation for Φ is as follows:

$$\Phi(M) = \delta_1 \frac{1}{1+dangle_{target}(M)} + (1-\delta_1)\frac{1}{1+dangle_{candidate}(M)}$$

where $dangle_{target}(X)$ and $dangle_{candidate}(X)$ are the number of target literal dangling matches and candidate literal dangling matches in X respectively.

In our experiments, we use a default "neutral" setting of parameters, where we weight all four terms equally, i.e. $\alpha 1 = \alpha 2 = \alpha 3 = \alpha 4 = 1$. and treat the comparison as symmetrical, i.e. $\varphi_1 = \delta_1 = 0.5$. These values are suggested in Love (2000).

Finally, in our implementation, each of these terms is normalized by a corresponding optimal score which is obtained by mapping a target semantics with itself.

6.4.4 Validation of evaluation function

We will now validate both the greedy mapping algorithm and the evaluation function we have described in the last few sections by applying it towards candidate semantic expressions and their corresponding targets. This is similar to the approach we used in Section 6.3.6 for our metre evaluator.

We will test our evaluator on two sets of candidate semantic expressions. The first set will be compared against the target semantics that represents the utterance *"John loves Mary"*, i.e.:

$$S_{target} = \{john(\_0,J), mary(\_1,M), love(\_2,J,M)\}$$

where $\_0, \_1$, and $\_2$ are variables generated for arguments that we are not interested in, i.e. are not bound to anything else. The candidate semantics are shown from (6.14) to (6.20), where we list their semantic expressions, $S_{candidate}$, along with the mappings M obtained by comparing them against S_{target} above using the greedy algorithm presented in Section 6.4.2. Note that for readability purposes, in these mappings we only show the literals being mapped, and not the argument bindings, b. Also, for each candidate we provide an utterance that $S_{candidate}$ can be seen to represent.

Candidate (6.14) is merely a copy of the target, to verify that the mapping algorithm correctly maps the corresponding literals, which it indeed succeeds at doing. Candidates (6.15) and (6.16) are subsets of S_{target}, and the mapping algorithm correctly maps the literals present in $S_{candidate}$ and adds the remaining literals in S_{target} as dangling matches. Candidate (6.17) introduces a different literal in the place of *mary*, and we see that in the resulting mapping M, both the target literal *mary* and the candidate literal *pizza* are dangling matches, as our conceptual similarity function c is simply one of identity. In candidates (6.18) to (6.20) we see the existence of $john$ appearing in the wrong argument position of $love$. As a result, the two $john$ literals in S_{target} and $S_{candidate}$ are not mapped in a proper match. This is because the two $love$ literals are mapped first, according to the list of prioritized criteria in Table 6.11, and thus mapping the two $john$ literals would result in a structurally inconsistent mapping. This is also true of the *mary* literals in (6.19).

(6.14) *“John loves Mary”*

$$\begin{aligned} S_{candidate} &= \{john(\_3,X), mary(\_4,Y), love(\_5,X,Y)\} \\ M &= \{(love(\_2,J,M), love(\_5,X,Y)), \\ &\phantom{=\ \{}(john(\_0,J), john(\_3,X)), \\ &\phantom{=\ \{}(mary(\_1,M), mary(\_4,Y))\} \end{aligned}$$

(6.15) *“John loves . . . ”*

$$\begin{aligned} S_{candidate} &= \{john(\_6,X), love(\_7,X,\_8)\} \\ M &= \{(love(\_2,J,M), love(\_7,X,\_8)), \\ &\phantom{=\ \{}(john(\_0,J), john(\_6,X)), \\ &\phantom{=\ \{}(mary(\_1,M), \varepsilon)\} \end{aligned}$$

(6.16) *“. . . loves . . . ”*

$$\begin{aligned} S_{candidate} &= \{love(\_9,\_10,\_11)\} \\ M &= \{(love(\_2,J,M), love(\_9,\_10,\_11)), \\ &\phantom{=\ \{}(john(\_0,J), \varepsilon), \\ &\phantom{=\ \{}(mary(\_1,M), \varepsilon)\} \end{aligned}$$

(6.17) *“John loves pizza”*

$$\begin{aligned} S_{candidate} &= \{john(\_12,X), love(\_13,X,Y), pizza(\_14,Y)\} \\ M &= \{(love(\_2,J,M), love(\_13,X,Y)), \\ &\phantom{=\ \{}(john(\_0,J), john(\_12,X)), \\ &\phantom{=\ \{}(mary(\_1,M), \varepsilon), \\ &\phantom{=\ \{}(\varepsilon, pizza(\_14,Y)\} \end{aligned}$$

(6.18) *“. . . loves John”*

$$\begin{aligned} S_{candidate} &= \{john(\_15,X), love(\_16,\_17,X)\} \\ M &= \{(love(\_2,J,M), love(\_16,\_17,X)), \\ &\phantom{=\ \{}(john(\_0,J), \varepsilon), \\ &\phantom{=\ \{}(mary(\_1,M), \varepsilon), \\ &\phantom{=\ \{}(\varepsilon, john(\_15,X))\} \end{aligned}$$

(6.19) *“Mary loves John”*

$$\begin{aligned} S_{candidate} &= \{john(\_18,X), love(\_19,Y,X), mary(\_20,Y)\} \\ M &= \{(love(\_2,J,M), love(\_19,Y,X)), \\ &\phantom{=\ \{}(john(\_0,J), \varepsilon), \\ &\phantom{=\ \{}(mary(\_1,M), \varepsilon), \\ &\phantom{=\ \{}(\varepsilon, john(\_18,X)), \\ &\phantom{=\ \{}(\varepsilon, mary(\_20,Y))\} \end{aligned}$$

(6.20) *"Pizzas love John"*

$$\begin{aligned} S_{candidate} &= \{john(\_21,X), love(\_22,Y,X), pizza(\_23,Y)\} \\ M &= \{(love(\_2,J,M), love(\_22,Y,X)), \\ & \quad (john(\_0,J),\varepsilon), \\ & \quad (mary(\_1,M),\varepsilon), \\ & \quad (\varepsilon, john(\_21,X)), \\ & \quad (\varepsilon, pizza(\_23,Y))\} \end{aligned}$$

Having obtained mappings for this set of candidate semantic expressions, we assign a fitness score to them using the similarity equation in Section 6.4.3. As mentioned above, we weight all four terms equally, i.e. $\alpha 1 = \alpha 2 = \alpha 3 = \alpha 4 = 1$. Also, we treat the comparison as symmetrical, i.e. $\varphi_1 = \delta_1 = 0.5$. The fitness scores can be seen in Table 6.12. We present the values for the individual terms Θ, Υ, Ω, and Φ, as well as the total fitness score, F.

$S_{candidate}$	Θ	Υ	Ω	Φ	F
(6.14)	1.0	1.0	1.0	1.0	1.0
(6.15)	0.667	0.667	0.5	0.75	0.646
(6.16)	0.334	0.334	0.0	0.667	0.334
(6.17)	0.556	0.667	0.5	0.5	0.556
(6.18)	0.667	0.334	0.0	0.417	0.354
(6.19)	1.0	0.334	0.0	0.334	0.417
(6.20)	0.556	0.334	0.0	0.334	0.306

Table 6.12: Semantic similarity fitness scores for candidates (6.14) to (6.20)

We will now discuss these fitness scores for each term. Recall from Section 6.4.3 that Θ captures raw conceptual similarity. We can see that candidates (6.14) and (6.19) yield a maximum score of 1.0, as they both contain all the literals in S_{target}. The other candidates are correctly scored lower as they are missing one or more literals. Note that candidates (6.17) and (6.20) score lower than candidates (6.15) and (6.18) due to the penalty incurred by the presence of a different concept, *pizza*.

The second term, Υ, captures conceptual similarity of mapped literals. Again (6.14) yields a maximum score as all literals are correctly mapped. Note that (6.19) only yields a score of 0.334, as despite having all three literals, only $love$ can be mapped in a structurally consistent manner.

The third term, Ω, captures the structural similarity within the predicate-argument structures. Again, (6.14) yields a maximum score. Only (6.15) and (6.17) yield a non-zero score as they correctly convey the argument structure of *john* being the subject of *love*.

Finally, the fourth term, Φ, is another structural term that penalizes dangling matches. We can see that the more dangling matches there are, the lower the score for Φ.

For these candidates, we feel that their overall fitness score, F, is a fair and appropriate measure of how similar they are to S_{target}. Of notable interest is that candidates (6.17) and (6.20) score lower than (6.15) and (6.18), respectively, due to the presence of the unmapped concept *pizza*.

For the second set of candidate semantic expressions, we use a more complex target semantic expression that represents the second line of Belloc's *"The Lion"*, i.e.*"[The lion] has a big head and a very small waist"*. The semantic expression is as follows:

$$\begin{aligned} S_{target} &= \{lion(\_0,L), own(\_1,L,H), head(\_2,H), big(\_3,H), \\ &\quad own(\_4,L,W), waist(\_5,W), small(S,W), very(\_6,S)\} \end{aligned}$$

This is a subset of the encoding of the first two lines of Belloc's *"The Lion"* which we presented in Section 5.2.4.

The second set of candidate semantic expressions is shown from (6.21) to (6.26). As with the first set of candidate semantics, (6.21) is simply an exact copy of the target semantics. (6.22) and (6.23) are different subsets of S_{target}. Candidate (6.24) is the semantic encoding of the first line of *"The Lion"*. Candidate (6.25) is a completely unrelated semantic expression. Finally, note that candidate (6.26) is subsumed by (6.22). By examining the mappings obtained for all these candidates, we can see that they are as we expect them to be, i.e. they are the maximal structurally consistent mappings that can be built given the proper matches that are deemed acceptable by our conceptual similarity function $c(m)$, with all unmapped literals being added as dangling matches.

(6.21) *"The lion has a big head and a very small waist"*

$$\begin{aligned} S_{candidate} &= \{lion(\_0,L), own(\_1,L,H), head(\_2,H), big(\_3,H), \\ &\quad own(\_4,L,W), small(S,W), waist(\_5,W), very(\_6,S)\} \\ M &= \{(lion(\_0,L), lion(\_0,L)), (head(\_2,H), head(\_2,H)), \\ &\quad (own(\_1,L,H), own(\_1,L,H)), (own(\_4,L,W), own(\_4,L,W)), \\ &\quad (big(\_3,H), big(\_3,H)), (small(S,W), small(S,W)), \\ &\quad (waist(\_5,W), waist(\_5,W)), (very(\_6,S), very(\_6,S))\} \end{aligned}$$

(6.22) *"The lion has a big head"*

$$\begin{aligned}
S_{candidate} &= \{lion(\_7,L), own(\_8,L,H), head(\_9,H), big(\_10,H)\}\\
M &= \{(lion(\_0,L), lion(\_7,L)), (head(\_2,H), head(\_9,H)),\\
&\quad (own(\_1,L,H), own(\_8,L,H)), (big(\_3,H), big(\_10,H)),\\
&\quad (own(\_4,L,W), \varepsilon), (small(S,W), \varepsilon),\\
&\quad (waist(\_5,W), \varepsilon), (very(\_6,S), \varepsilon)\}
\end{aligned}$$

(6.23) *"The lion has a head and a waist"*

$$\begin{aligned}
S_{candidate} &= \{lion(\_11,L), own(\_12,L,H), head(\_13,H),\\
&\quad own(\_14,L,W), waist(\_15,W)\}\\
M &= \{(lion(\_0,L), lion(\_11,L)), (head(\_2,H), head(\_13,H)),\\
&\quad (own(\_1,L,H), own(\_12,L,H)), (own(\_4,L,W), own(\_14,L,W)),\\
&\quad (waist(\_5,W), waist(\_15,W)), (big(\_3,H), \varepsilon),\\
&\quad (small(S,W), \varepsilon), (very(\_6,S), \varepsilon)\}
\end{aligned}$$

(6.24) *"The lion dwells in the waste"*

$$\begin{aligned}
S_{candidate} &= \{lion(\_16,L), dwell(d,L), inside(\_17,d,W), waste(\_18,W)\}\\
M &= \{(lion(\_0,L), lion(\_16,L)), (own(\_1,L,H), \varepsilon),\\
&\quad (head(\_2,H), \varepsilon), (big(\_3,H), \varepsilon), (own(\_4,L,W), \varepsilon),\\
&\quad (small(S,W), \varepsilon), (waist(\_5,W), \varepsilon), (very(\_6,S), \varepsilon),\\
&\quad (\varepsilon, dwell(d,L)), (\varepsilon, inside(\_17,d,W)), (\varepsilon, waste(\_18,W))\}
\end{aligned}$$

(6.25) *"John loves Mary"*

$$\begin{aligned}
S_{candidate} &= \{john(\_19,j), love(\_20,j,m), mary(\_21,m)\}\\
M &= \{(lion(\_0,L), \varepsilon), (own(\_1,L,H), \varepsilon), (head(\_2,H), \varepsilon),\\
&\quad (big(\_3,H), \varepsilon), (own(\_4,L,W), \varepsilon), (small(S,W), \varepsilon),\\
&\quad (waist(\_5,W), \varepsilon), (very(\_6,S), \varepsilon), (\varepsilon, john(\_19,j)),\\
&\quad (\varepsilon, love(\_20,j,m)), (\varepsilon, mary(\_21,m))\}
\end{aligned}$$

(6.26) *"John and Mary love the lion's big head"*

$$\begin{aligned}
S_{candidate} &= \{john(\_22,j), love(\_23,j,H), mary(\_24,m), love(\_25,m,H),\\
&\quad lion(\_26,L), own(\_27,L,H), head(\_28,H), big(\_29,H)\}\\
M &= \{(lion(\_0,L), lion(\_26,L)), (head(\_2,H), head(\_28,H)),\\
&\quad (own(\_1,L,H), own(\_27,L,H)), (big(\_3,H), big(\_29,H)),\\
&\quad (own(\_4,L,W), \varepsilon), (small(S,W), \varepsilon), (waist(\_5,W), \varepsilon),\\
&\quad (very(\_6,S), \varepsilon), (\varepsilon, john(\_22,j)), (\varepsilon, love(\_23,j,H)),\\
&\quad (\varepsilon, mary(\_24,m)), (\varepsilon, love(\_25,m,H))\}
\end{aligned}$$

The fitness scores assigned to these candidate semantic expressions are shown in Table 6.13. We use the same parameters as before, i.e. $\alpha 1 = \alpha 2 = \alpha 3 = \alpha 4 = 1$ and $\varphi_1 = \delta_1 = 0.5$.

$S_{candidate}$	Θ	Υ	Ω	Φ	F
(6.21)	1.0	1.0	1.0	1.0	1.0
(6.22)	0.571	0.5	0.428	0.6	0.525
(6.23)	0.571	0.625	0.571	0.625	0.598
(6.24)	0.0	0.125	0.0	0.187	0.078
(6.25)	0.0	0.0	0.0	0.181	0.0451
(6.26)	0.428	0.5	0.428	0.2	0.389

Table 6.13: Semantic similarity fitness scores for candidates (6.21) to (6.26)

The first term, Θ, captures raw conceptual similarity. As expected, candidate (6.21) yields a perfect score, and candidates (6.22) and (6.23), being subsets of S_{target} are scored proportionately. Note that (6.24) scores 0.00, despite containing the literal $lion$. This is due to the penalty incurred by the differences, which is parameterized by φ_1. This penalty is also apparent in the value for Θ that is assigned to (6.26). Note that although it contains the same literals as (6.22), it scores lower due to the penalty incurred by the concepts $john$, $love$, and $mary$.

The second term, Υ, captures conceptual similarity of mapped literals, and the values assigned to these candidates is as expected. Note that the value for candidate (6.24) is 0.125, in recognition of the fact that it contains the mapped literal $lion$. Unlike Θ, it does not incur any penalty, as it is only concerned with mapped literals. Similarly, note also that candidate (6.26) scores the same as (6.22), as they have the exact same proper matches.

The third term, Ω, captures the structural similarity within the predicate-argument structures, and we feel the scores assigned to these candidates accurately reflect this. The highest scoring candidate apart from (6.21) is (6.23), where each literal has at least one argument that is similarly bound by another proper match in M.

Lastly, the fourth term, Φ, penalizes dangling matches. Note that the value for (6.25) is the only non-zero term assigned to this candidate. This is because Φ is an asymptotic function that approaches zero for infinitely large semantic expressions with unmapped literals. Note that even an empty candidate semantic expression, $\{\}$, would score higher than candidate (6.25). This intuitively makes sense, as from a semantic point of view it is 'closer' in distance to S_{target}[1].

[1] The idiom "silence is golden" comes to mind!

6.5 Summary

In this chapter we have defined evaluation functions for metre similarity and semantic similarity, both of which utilize a common strategy, i.e. measuring the degree of isomorphism between a given target structure and the appropriate features of a candidate solution. For metre evaluation, the well-known edit distance algorithm is used, and the resulting distance is used as the basis of an objective function, which we augment with a simple account of context-sensitive scoring. A separate function that measures the potential score of substitution nodes in incomplete derivations is also presented. For semantics, we present a greedy mapping algorithm that is reminiscent of the algorithm used in SME (Falkenhainer et al., 1989), and measure the quality of the resulting mapping using a similarity equation proposed by (Love, 2000). All these evaluation functions have been validated by testing them on candidate solutions for which we have a priori judgments as to their quality, albeit ones based on intuition.

Chapter 7

Genetic operators for linguistic structure building

In this chapter we describe the genetic operators we have designed and implemented in McGONAGALL. We will start by listing the design considerations for our operators. We then describe the baseline ‘blind’ operators, which only adhere to syntactic well-formedness, followed by a description of ‘smart’ operators which borrow techniques from NLG tactical systems seen in Section 3.4.1 to explicitly achieve the goal of meaningfulness. Finally we describe compound operators, particularly ones that ensure complete derivations. For each of the operators described we present an example of its application towards some simple structures. We also discuss the issue of determining the probability that a particular operator is applied to an individual.

7.1 Design considerations

As described in Chapter 4, an evolutionary algorithm is essentially a form of heuristic search. It stochastically explores the search space and considers which points to explore further based on their fitness scores. A move in the search space is achieved by applying a genetic operator, henceforth simply operator, to an individual. An operation that takes one individual and yields a different one is called mutation, whereas one that takes two or more individuals is called crossover or recombination.

The following is a list of properties that inform our design of operators:

1. Both mutation and crossover operations in evolutionary algorithms are **stochastic** in nature, i.e. they modify individuals in a random manner.

2. Recall from Section 4.3.1 that we choose to enforce **grammaticality** of our individuals through design of representation and operators. Thus, when designing an operator, we can be assured that the individual it is attempting to modify is grammatical, but conversely, we must assure that the resulting individual must be grammatical as well.

3. We do not expect individuals to be complete solutions from the offset, i.e. from the point of their initialization. Rather, solutions are constructed **incrementally**, benefitting from the guiding hand of evolution. With respect to our individuals, this incrementality can be realized in two different fashions: operators that may yield incomplete derivations and operators that ensure complete derivations (Section 7.4.1).

4. As operators are stochastic, it is highly improbable that optimal solutions can be constructed using monotonic structure-building operators alone, for example by only ever randomly substituting and adjoining LTAG elementary trees together. More likely, optimal solutions are gradually built through the 'trial and error' design mechanism that evolution affords. Therefore, we need one or more **nonmonotonic** operators to facilitate this, such as deletion (Section 7.2.2) and swapping (Section 7.2.3).

Michalewicz (1996) states that the representation of an individual in the population and the set of operators used to alter its genetic code are crucial in determining a system's success or failure. Moreover, these two aspects are strongly related: the design of representational scheme must consider the operators that will eventually manipulate them, and the design of operators must exploit characteristics of the representation.

Since our individuals are represented using the LTAG formalism (Section 5.3.1), it would be natural to try to base our genetic operators on the syntactic operations within LTAG theory, i.e. substitution and adjunction. However, for our purpose of stochastic, nonmonotonic structure building, we also look towards the genetic operators used in genetic programming, as they manipulate similar structures (Section 4.2.6). Our design of baseline operators, which we describe in the next section, is informed by such operators.

7.2 Baseline operators: ensuring grammaticality

Angeline (1996) defines four search operators on S-expression parse trees, which are the structures manipulated in genetic programming: **grow**, **shrink**, **switch**, and **cycle**. The grow operator randomly selects a leaf from a tree and replaces it with a randomly generated new subtree. The shrink operator does the opposite: it randomly takes an internal node and replaces the subtree it dominates with a randomly generated new leaf node. The switch operator randomly selects two nodes and swaps their position, and finally the cycle operator randomly chooses a node and replaces it with a randomly generated node that is similar. Figure 7.1 shows an example of these operations.

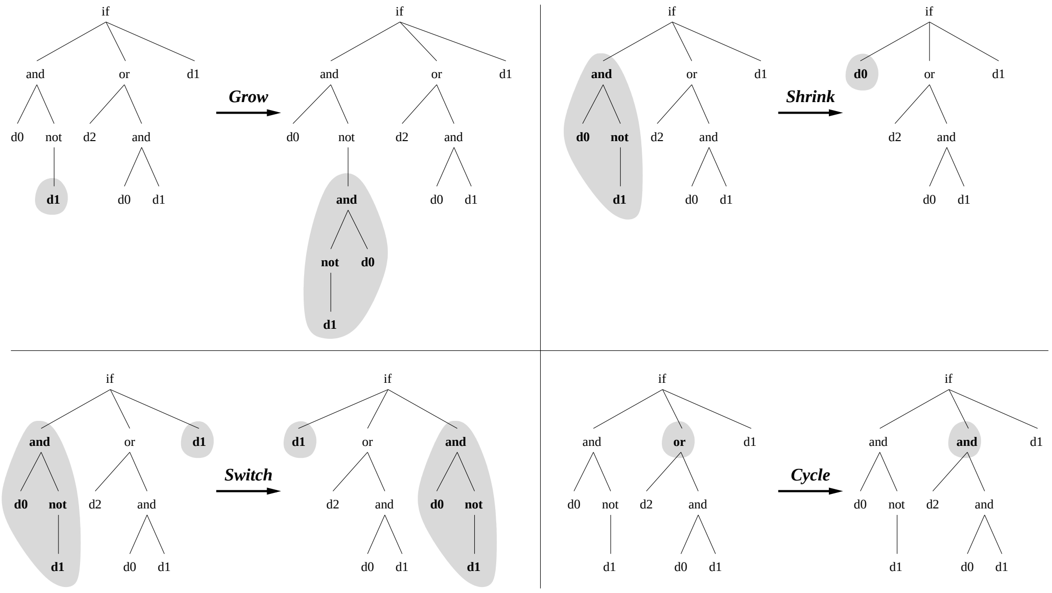


Figure 7.1: Examples of search operators on S-expression parse trees, taken from Angeline (1996)

Observing these operations, it is tempting to suggest that these operators can be applied to LTAG phrase structure trees, i.e. derived trees. However, LTAG trees possess unique characteristics not found in the parse trees manipulated in Angeline (1996). For instance, leaf nodes in LTAG are either lexical anchor nodes, foot nodes, or substitution nodes. Of these three, the grow operator is only meaningful on a substitution node, i.e. by applying the substitution operation. Moreover, the grow operator as defined does not account for the adjunction operation.

Randomly applying the shrink and switch operators on nodes in a derived tree could easily result in syntactically ill-formed constructions that are impossible to derive from the grammar. The notion of grammaticality, which in LTAG is defined in terms of substitution and adjunction of elementary trees, would be disrupted. Moreover, the derivation tree becomes obsolete.

We believe that the appropriate structure to apply such operators on is not the derived tree, but rather the derivation tree. The equivalent of the grow operator would be to simply add a node in the derivation tree. Furthermore, this can account for both substitution and adjunction. This is the BLINDADD operator that we discuss in Section 7.2.1. The equivalent of the shrink operator would be to remove a node in the derivation tree, along with the subtree that it dominates. This is the BLINDDELETE operator that we discuss in Section 7.2.2. And finally, the switch operator can be implemented on the derivation tree, i.e. by swapping the positions of two subtrees, either belonging to the same derivation (mutation) tree, or to another derivation tree (crossover). This is the BLINDSWAP operator that we discuss in Section 7.2.3.

Note that although these operations are defined on the derivation tree, they clearly result in manipulations on the resulting derived tree as well, and these manipulations are all defined in terms of substitution and adjunction operations. Hence, these operators are still compatible with LTAG theory, and the process of checking for grammaticality of the resulting individuals is well defined, e.g. the feature structure unification discussed in Section 5.3.4.

Recall from Section 5.4 that the individuals manipulated by these operators are initialized with the I_T_KO tree that represents an empty sentence substitution node.

In the next three sections, we will discuss these baseline operators in more detail.

7.2.1 BLINDADD operator

As defined in Section 4.3, our search space comprises all possible syntactic constructions. In our chosen formalism, LTAG, this equates to all valid compositions of elementary trees in our grammar, anchored by all possible valid lexical items in our lexicon. Thus the simplest possible operator would be to randomly add an elementary tree onto an individual's derived tree through either substitution or adjunction. Since an individual's primary data structure is the derivation tree, which keeps record of all elementary tree compositions, this operation amounts to randomly choosing a node in the derivation tree and adding a child to it that represents a legal operation, i.e. substitution or adjunction.

We call such an operator the BLINDADD operator, i.e. one that does not explicitly attempt to

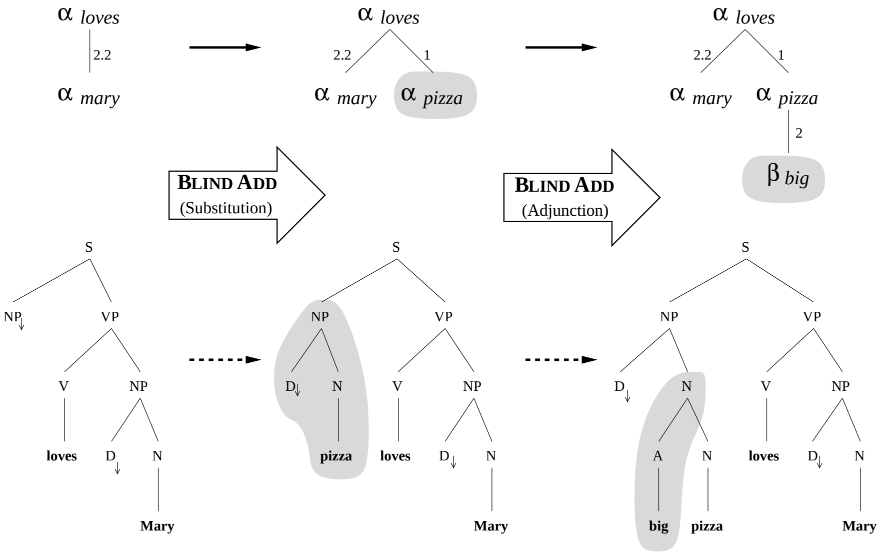


Figure 7.2: Example of the BLINDADD operator and its effect on both derivation and derived tree

fulfill any generation goals, but instead simply adds to the linguistic content of an individual while maintaining its syntactic well-formedness.

Figure 7.2 shows an example of two instances of the BLINDADD operator, one involving substitution and the other involving adjunction. The top row shows the derivation trees that represent individuals, and the bottom row shows the resulting derived trees. The first BLINDADD operation substitutes α_{pizza}, a noun phrase tree anchored by the noun *'pizza'*, at the subject position of the transitive verb tree α_{loves}, which is still unsubstituted. Note the similarity with the grow operator in Figure 7.1. The second BLINDADD operation takes the resulting individual from the previous one and adjoins β_{big}, an adjective tree anchored by the adjective *'big'*, at the noun preterminal node of α_{pizza}. Note that the operation address is defined relative to the elementary tree α_{pizza}. We can see that adjunction facilitates a more flexible structure-building approach than the grow operator.

More formally, given an individual's derivation tree T, a grammar G, and a lexicon L, BLINDADD stochastically yields a 4-tuple $< d, n, t_g, l >$ where:

d is a node $\in T$,
n is a node $\in t_d$, the elementary tree represented by d,
t_g is an elementary tree $\in G$, and
l is a lexical item $\in L$.

Furthermore, for the operation to be syntactically valid, all of the following must apply:

1. n must be a node which licenses a syntactic operation. For substitution this means a substitution node which has not been substituted yet (in the context of T), and for adjunction, an internal node which has not been the site for an adjunction yet (also in the context of T).

2. All feature structures of t_g must license the operation at n in the context of T (to be more precise, the derived tree obtained from T).

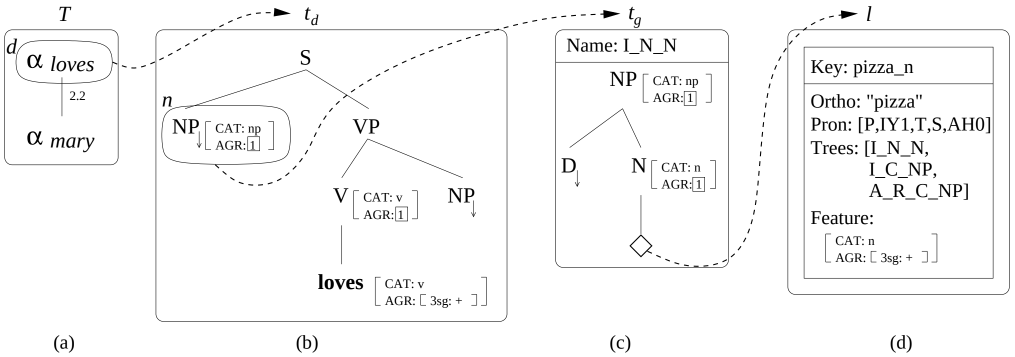
3. All feature structures of l must license its anchoring of t_g in the context of the previous point, i.e. where t_g is added as a child of n in T.

T' is the resulting derivation tree of substituting/adjoining t_g, anchored by l, at n.

Figure 7.3 shows such a configuration for the first operation shown in Figure 7.2, i.e. the substitution of α_{pizza}. The derivation tree T in (a) represents a partially constructed derivation of the transitive verb *'loves'* with the object *'mary'*. This can be seen from the operation address (2.2) which indicates the object NP position. Choosing the root node as d, in (b) we see the elementary tree t_d at that node, and the choice of the empty NP substitution node at Gorn address 1 as n. Note that the NP node at 2.2 is not a valid choice as we can see from T that it has already been substituted. Going through our grammar G, we choose the tree I_N_N, the basic noun phrase tree, as t_g, which is shown in (c). Finally, (d) shows the chosen l, the lexical item pizza_n. We have also shown the relevant feature structures, i.e. category and agreement, that license the operation.

The BLINDADD operator does not attempt to satisfy any communicative goal, hence the strange choice of l (we assume that the lexicon in this example does not have selectional restrictions that would require, for example, that the subject of *'loves'* is an animate entity). The only constraint on this operation is grammaticality, which is why it selected the I_N_N tree and a 3rd person singular noun to anchor it.

Furthermore, there is no restriction that the application of BLINDADD must obey any linear



(a) (b) (c) (d)

Figure 7.3: choosing (a) node d in T, (b) node n in t_d, (c) tree t_g, and (d) word l that results in a valid operation

order. Although α_{mary} precedes α_{pizza} in the derivation tree, in the obtained derived tree they are correctly placed in their proper positions as indicated by their operation addresses.

Lastly, derivation trees that are built by BLINDADD are not guaranteed to be complete derivations, i.e. in the resulting derived tree there may be substitution nodes which are not yet substituted, or obligatory adjunction (OA) nodes that are not yet adjoined (Section 5.3.6). Hence, our notion of grammaticality must be extended to include these underspecified constructions: they are as yet ungrammatical, but repeated application of BLINDADD will eventually yield complete and syntactically well-formed texts. Note that in the grammar we have developed for our empirical testing, we do not use obligatory adjunction.

Since we are looking for any random 4-tuple that satisfies the conditions stated above, we perform a stochastic depth-first search through the four levels of choices of derivation tree node, elementary tree node, elementary tree, and lexical item. At each level of representation we construct the set of all possible choices, taking into account all constraints at that point, randomly choose one, and continue to the next level of representation. This continues until the 4-tuple is completed or until there are no more possible choices, at which point we backtrack to the previous level and randomly choose another element not yet tried. If, after exhausting all possible choices, no valid 4-tuple can be found, the operator indicates to the EA that it has failed to do so, i.e. by returning a null value.

We will now present our algorithm for this search, but we first define the datatypes used:

`DTree`:	a derivation tree
`Tree`:	an elementary tree
`DNode`:	a node in a `DTree`
`Node`:	a node in a `Tree`
`Word`:	a lexical item

In addition, we define four functions, `DNodeFilter`, `NodeFilter`, `TreeFilter`, and `WordFilter`, which will aid the search in constructing the set of valid choices given a particular context. A `DNodeFilter` is a function which, given a derivation tree T, returns a set of potential nodes to be explored. In general, we would want to consider all nodes in the derivation tree, but there are instances when we can restrict our search space to specific nodes. A `NodeFilter` is a function which, given a derivation tree T and a node $d \in T$, returns a set of potential nodes from t_d, the elementary tree at d. For the substitution operation, for example, it will only consider substitution nodes that have not been substituted yet, whereas for adjunction it will only consider internal nodes that have not been adjoined yet. A `TreeFilter` is a function which, given a grammar G, a derivation tree T, a node $d \in T$, and a node $n \in t_d$, returns a set of trees from G which license operation at n. A `WordFilter` is a function which, given a lexicon L, a derivation tree T, a node $d \in T$, a node $n \in t_d$, and a tree t_g, returns a set of lexical items from L which license anchoring of t_g in the context of it being operated at n.

For the BlindAdd operator, the only constraint being checked by these filters is that of syntactic well-formedness, i.e. feature structure unification.

Algorithm 7.1 is the main function of the depth first search, `getAddCandidate(DTree)`, as it handles the first level of representation. Given a `DTree`, it selects a `DNode` d and calls `getAddCandidate(DTree, DNode)`, shown in Algorithm 7.2. Given a `DTree` and a `DNode`, this function selects a `Node` n and calls `getAddCandidate(DTree, DNode, Node)`, shown in Algorithm 7.3. Given a `DTree`, a `DNode`, and a `Node`, this function selects a `Tree` t_g and calls `getAddCandidate(DTree, DNode, Node, Tree)`, shown in Algorithm 7.4. Finally, given a `DTree`, a `DNode`, a `Node`, and a `Tree`, this function selects a `Word` l. All these functions return a *candidate*, which is a 4-tuple $< d, n, t_g, l >$ that specifies a valid LTAG operation.

Figure 7.4 shows a sketch of this algorithm being applied for the second operation in Figure 7.2, i.e. the adjunction of β_{big}. The four representational levels of choice are shown in order from top to bottom. At the top level, the set of choices returned by `DNodeFilter` is simply that of all nodes in the derivation tree, T. Here we choose the α_{pizza} node as d. At the second

Algorithm 7.1 getAddCandidate(DTree T)

```
$dnodes \leftarrow$ DNodeFilter($T$)
while $dnodes$ is not empty do
    $d \leftarrow$ randomly removed DNode from $dnodes$
    $candidate \leftarrow$ getAddCandidate$(T,d)$
    if $candidate \neq$ null then
        return $candidate$
    end if
end while
return null
```

Algorithm 7.2 getAddCandidate(DTree T,DNode d)

```
$nodes \leftarrow$ NodeFilter($T$,$d$)
while $nodes$ is not empty do
    $n \leftarrow$ randomly removed Node from $nodes$
    $candidate \leftarrow$ getAddCandidate$(T,d,n)$
    if $candidate \neq$ null then
        return $candidate$
    end if
end while
return null
```

Algorithm 7.3 getAddCandidate(DTree T, DNode d, Node n)

```
$trees \leftarrow$ TreeFilter$(G,T,d,n)$
while $trees$ is not empty do
    $t_g \leftarrow$ randomly removed Tree from $trees$
    $candidate \leftarrow$ getAddCandidate$(T,d,n,t_g)$
    if $candidate \neq$ null then
        return $candidate$
    end if
end while
return null
```

Algorithm 7.4 `getAddCandidate(DTree T, DNode d, Node n, Tree t_g)`

```
$words \leftarrow$ WordFilter$(L, T, d, n, t_g)$
if $words$ is not empty then
    $l \leftarrow$ randomly removed Word from $words$
    return $< d, n, t_g, l >$
else
    return null
end if
```

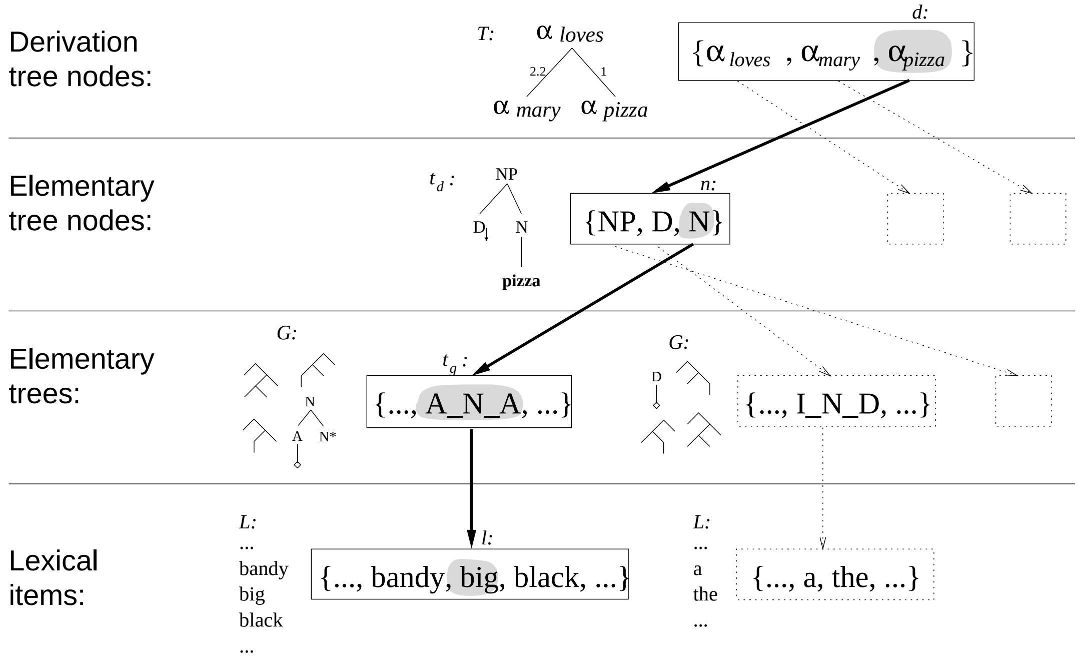


Figure 7.4: The stochastic depth-first search explicitly traverses four levels of representation

level, the set of choices returned by `NodeFilter` is that of nodes in t_d that license syntactic operation, i.e. the substitution node D and the internal nodes NP and N. Here we choose the noun preterminal node, N, as *n*. At the third level, `TreeFilter` constructs a set of choices by going through the grammar, *G*, and selecting valid trees. In our implementation, we perform a quick first pass that selects trees that are rooted by nodes that share the same label as *n*. At this point we also already know whether to look for initial or auxiliary trees based on the choice of *n*. A more thorough second pass goes through this initial selection and tests that the appropriate feature structures unify (Section 5.3.4). In this example we choose `A_N_A`, the adjective auxiliary tree, as t_g. Finally, at the last level of representation, `WordFilter` constructs a set of choices by going through the lexicon, *L*, and selecting valid words. Again this can be implemented through a quick first pass which selects words that list t_g as one of the elementary trees they can anchor, followed by a more thorough second pass that tests feature structure unification.

The arrows and boxes in dashed lines in Figure 7.4 indicate other states that can be stochastically explored with backtracking should the algorithm reach a dead-end, i.e. exhaust all choices at a given level. In particular, we show an alternative path that would select an operation that substitutes a determiner for the *'pizza'* noun phrase.

Note that the two quick first pass operations mentioned above can be efficiently implemented by appropriate indexing of the grammar and lexicon.

In MCGONAGALL, we have implemented two different operators, BLINDADDSUBSTITUTE and BLINDADDADJOIN, which deal with substitution and adjunction operations separately. Although it is possible to just have one BLINDADD operator that handles both, we will see the benefits of making this distinction when testing the operators (Section 7.2.4) and also when designing compound operators (Section 7.4.1).

7.2.2 BLINDDELETE operator

The BLINDADD operator described above already suffices to cover the search space of all syntactically well-formed constructions. However, due to the stochastic nature of our search, we add two nonmonotonic operators that allow more flexible movement within the search space: the BLINDDELETE and BLINDSWAP operators.

Like BLINDADD, BLINDDELETE operates on the derivation tree, thus guaranteeing that manipulations of the phrase structure tree, i.e. derived tree, are still carried out in terms of substitution and adjunction of elementary trees. The operator simply removes the subtree that is dominated by a randomly selected node in the derivation tree. This operator effectively 'undoes' the composition of elementary trees as performed by BLINDADD.

Figure 7.5 shows an example of the BLINDDELETE operation, and the effect it has on a derivation tree, shown in the top row, and the obtained derived tree, shown in the bottom row. Here, the subtree rooted at the β_{deeply} node is removed. As shown in the derived tree, it corresponds to the adverbial phrase *"very deeply"*. Note that the removal of this phrase from the resulting derived tree is not something that can be achieved by Angeline's shrink operator as described in Section 7.2.

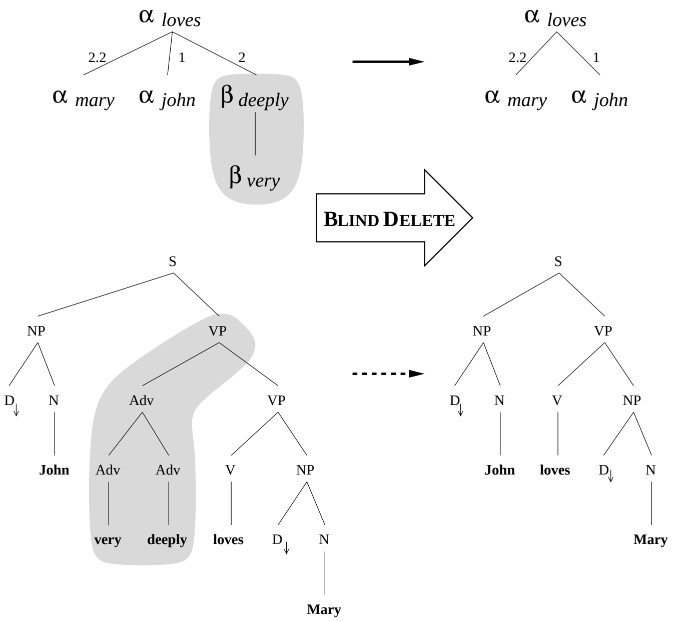


Figure 7.5: Example of the BLINDDELETE operator and its effect on both derivation and derived tree

More formally, the BLINDDELETE operator takes an individual's derivation tree, T, and stochastically yields a node $d \in T$, where the subtree T_d that is rooted at d is deleted. The resulting derivation tree, where T_d has been removed, is called T'.

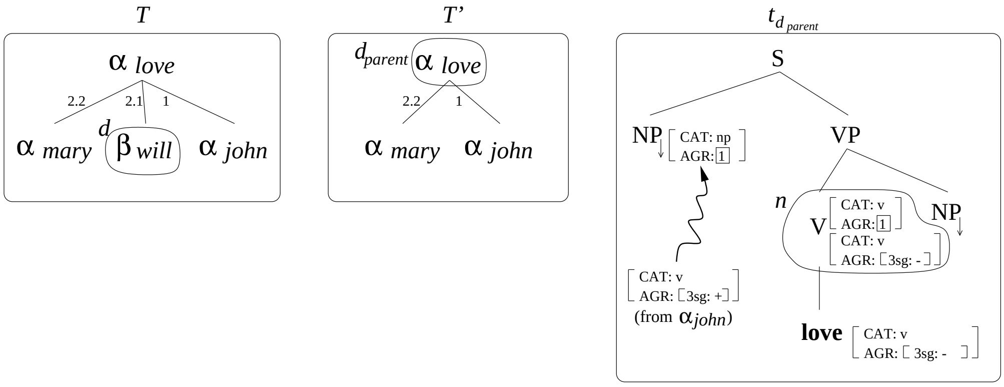
Note that if we choose d to be the root node of T, we would in fact delete the entire derivation tree. This would present a problem for some of our operators, e.g. the BLINDADD operator requires at least a single node to exist (observe Algorithm 7.1). Thus we restrict the choice of $d \in T$ to exclude the root node of T. Fortunately, this is not a problem in McGonagall, because the root node of an individual's derivation tree always represents the special 'kickoff-tree', `I_T_KO`, which represents an empty sentence substitution node (see Section 5.4).

At first glance, it seems logical to assume that, provided the derived tree obtained from T is syntactically well-formed, the derived tree obtained from T' is also syntactically well-formed, albeit possibly underspecified. However, this is only true if the elementary tree t_d represented by d is an initial tree, i.e. is the subject of a substitution operation. If it is an auxiliary tree, we must first check that the resulting phrase structure is well-formed. This is because of the nonmonotonic nature of the adjunction operation itself with respect to the feature structures (see Section 5.3.5).

To formally specify this check, let $address_d$ be the operation address of the deleted node d, $d_{parent} \in T'$ be the former parent of the deleted node d, and $t_{d_{parent}}$ be the elementary tree represented by d_{parent}. For the operation to be syntactically valid, the feature structures at node $n \in t_{d_{parent}}$ in the context of the derived tree obtained from T' must unify, where the address of n in $t_{d_{parent}}$ is specified by $address_d$.

Figure 7.6 shows a configuration where this check reveals that the feature structures at n fail to unify, resulting in an invalid T'. In (a) we see the derivation tree T that represents the text *"John will love Mary"*. The choice of d is β_{will}, an auxiliary tree that adjoins the auxiliary verb *'will'* onto the bare stem verb *'love'*. The resulting derivation tree of this deletion, T', is shown in (b), which also indicates d_{parent} is the root node (α_{love}). Finally, (c) shows $t_{d_{parent}}$, the elementary tree at d_{parent}, and the node n whose feature structures must be checked (its Gorn address is 2.1, the operation address of d in T). Note that the bottom agreement feature of n is [AGR [3SG : -]], which comes from its anchor, the bare stem *'love'*, whereas the top agreement feature is coindexed with that of the subject NP ([AGR: $\boxed{1}$]). However, within the context of T', we can see that the feature structure from the substitution of α_{john} at the subject NP position contains [AGR [3SG : +]]. Thus, the features will fail to unify. This failure was prevented in the derived tree obtained from T because the auxiliary verb tree, β_{will}, 'splits' the V node in two.

The BlindDelete operator indiscriminately removes both 'good' and 'bad' subtrees. Since the only constraint being checked is that of syntactic well-formedness, i.e. feature structure unification, it may well delete constituents that are contributing positively towards the individual's semantics or metre similarity fitness score.



(a) (b) (c)

Figure 7.6: Example of BLINDDELETE where choice of d results in an invalid T'

Unlike the four-level deep search of BLINDADD, BLINDDELETE simply needs to search the first level of representation, i.e. the nodes in derivation tree T. Algorithm 7.5 is the main function of this search. The boolean function `verifyDeletion` performs the check described above, and returns true if the feature structures at n are unifiable, and false otherwise. The datatypes used are the same as those in Section 7.2.1.

Algorithm 7.5 `getDeleteCandidate(DTree` T`)`

```
$dnodes$ ← DNodeFilter($T$)
while $dnodes$ is not empty do
  $d$ ← randomly removed DNode from $dnodes$
  if verifyDeletion($T$, $d$) then
    return $d$
  end if
end while
return null
```

7.2.3 BLINDSWAP operator

The last of our baseline operators is the BLINDSWAP operator, which takes two compatible derivation tree subtrees and swaps their position. By compatible we mean that the resulting derived tree, or trees, are syntactically well-formed, i.e. all feature structures unify. Note that if the two subtrees are taken from the same derivation tree, it is an instance of mutation, but if they are taken from two different derivation trees, i.e. belonging to two different individuals, it is an instance of crossover. In our implementation we use BLINDSWAP as a crossover operation.

Operations are recorded in the derivation tree relative to their parents, i.e. if T_n is a subtree of derivation tree T that is rooted at node n, then the derived tree D_n obtained from T_n is a corresponding subtree of the derived tree D obtained from T that is rooted at the node specified by the operation address of n. This creates an elegant correspondence in that the swapping of subtrees between derivation trees corresponds with the swapping of subtrees in the derived trees obtained from the resulting derivation trees.

Figure 7.7 shows an example of this swapping operation. The object noun phrase in the first derivation tree, that represents the sentence *"John loves Mary"*, is swapped with the object noun phrase in the second derivation tree, that represents the sentence *"The quick sprinter runs"*. As a result, the two resulting derivation trees represent the sentences *"John loves the quick sprinter"* and *"Mary runs"*. Note the correspondences in the subtrees being swapped at the level of derivation and derived tree. Note also that for all nodes in the derivation trees except the two nodes that are the roots of the swapped subtrees, the operation addresses remain the same, as they are defined relative to their parent's elementary trees.

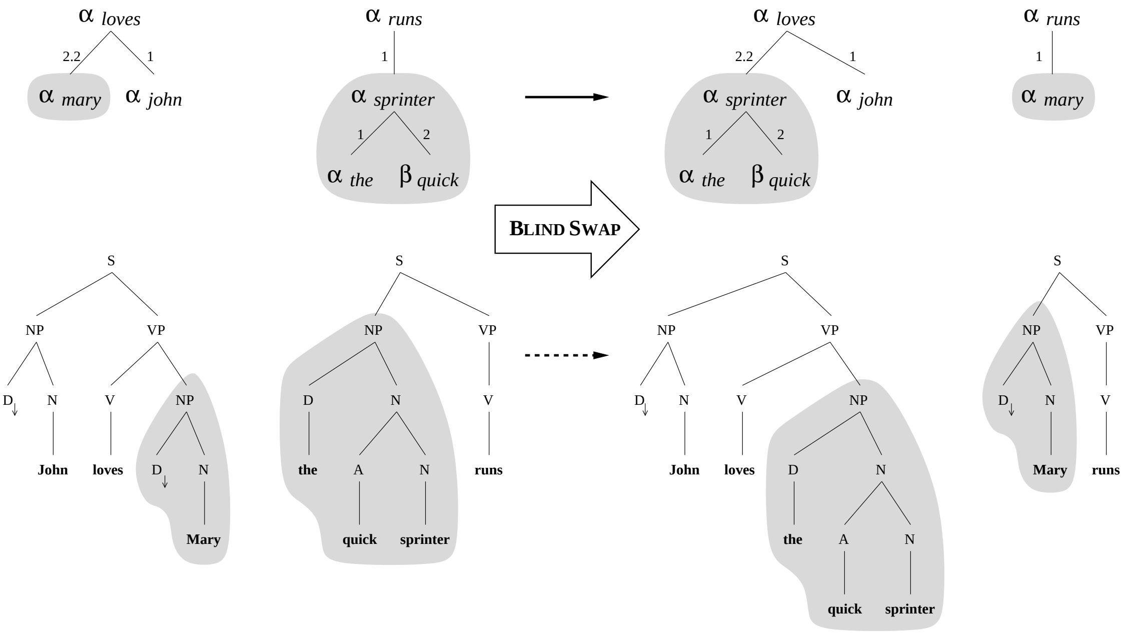


Figure 7.7: Example of the BLINDSWAP operator and its effect on both derivation and derived tree

More formally, given two derivation trees T_1 and T_2, the BLINDSWAP operator yields a pair $< d_1, d_2 >$, where:

d_1 is a node $\in T_1$, and
d_2 is a node $\in T_2$

If T_{d_1} is the subtree of T_1 rooted at d_1 and T_{d_2} is the subtree of T_2 rooted at d_2, then T_1' and T_2' are the resulting derivation trees after T_{d_1} and T_{d_2} have been swapped around.

Furthermore, for the operation to be syntactically valid, the following must apply:

1. The elementary trees t_{d_1} and t_{d_2}, which are represented by the nodes d_1 and d_2, must be the same type, i.e. they are either both initial trees or both auxiliary trees.

2. The top and bottom feature structures of the two nodes that are the swapping points in the derived trees obtained from T_1' and T_2' must be unifiable. These two nodes are $n_{swap_{d_1}}$, the node in the elementary tree of d_1's parent that has the Gorn address indicated by d_1's operation address, and $n_{swap_{d_2}}$, the node in the elementary tree of d_2's parent that has the Gorn address indicated by d_2's operation address.

Like the BLINDDELETE operator, BLINDSWAP only needs to search the first level of representation, i.e. derivation tree nodes. But since there are two derivation trees, T_1 and T_2, it performs a two-level deep stochastic depth first search. Algorithm 7.6 is the main function that searches the nodes of T_1, and Algorithm 7.7 is an auxiliary function that searches the nodes of T_2. The boolean function `verifySwap` performs the check described above, and returns true if the top and bottom feature structures at $n_{swap_{d_1}}$ and $n_{swap_{d_2}}$ are unifiable, and false otherwise.

Algorithm 7.6 `getSwapCandidate(DTree` T_1`, DTree` T_2`)`

```
$dnodes \leftarrow$ DNodeFilter($T_1$)
while $dnodes$ is not empty do
  $d \leftarrow$ randomly removed DNode from $dnodes$
  $candidate \leftarrow$ getSwapCandidate($T_1, T_2, d$)
  if $candidate \neq$ null then
    return $candidate$
  end if
end while
return null
```

Algorithm 7.7 `getSwapCandidate(DTree` T_1`, DTree` T_2`, DNode` d_1`)`

```
$dnodes \leftarrow$ DNodeFilter($T_2$)
while $dnodes$ is not empty do
  $d_2 \leftarrow$ randomly removed DNode from $dnodes$
  if verifySwap($T_1$, $T_2$, $d_1$, $d_2$) then
    return $< d_1, d_2 >$
  end if
end while
return null
```

7.2.4 Testing the baseline operators

We will now present the results of some simple testing performed with our implementation of these baseline operators in McGonagall. Table 7.1 shows a roundup of all the baseline operators presented so far.

Name	Description
BlindAddSubstitute	Takes an individual and randomly attempts to add an initial tree to the derivation tree, whilst maintaining syntactic well-formedness (See Figure 7.2).
BlindAddAdjoin	Takes an individual and randomly attempts to add an auxiliary tree to the derivation tree, whilst maintaining syntactic well-formedness (See Figure 7.2).
BlindDelete	Takes an individual and randomly attempts to delete a subtree from the derivation tree, whilst maintaining syntactic well-formedness (See Figure 7.5).
BlindSwap	Takes two individuals and randomly attempts to swap two subtrees between the respective derivation trees, whilst maintaining syntactic well-formedness (See Figure 7.7).

Table 7.1: Roundup of baseline operators

Firstly, Table 7.2 shows the behaviour of BlindAddSubstitute. Recall from Section 5.4 that we start off with a derivation tree with one node representing the initial tree `I_T_KO`, shown in Figure 7.8. This is the special tree that every individual is initialized with. Although in a normal evolutionary run these initialized individuals would then be randomly manipulated by a combination of genetic operators before being added to the population, in this testing we will start from the initialized individual with just the root node to clearly show the effect of the specific operators being tested.

Table 7.2 shows three instances of the incremental building of a derivation by applying only BLINDADDSUBSTITUTE five times on a single individual. Note that in the first instance, the resulting derivation is still incomplete.

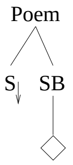


Figure 7.8: I_T_K0: "Kickoff tree" for a newly initialized individual

$\boxed{S}$.
$\boxed{NP}$ $\boxed{CV}$ *in* $\boxed{NP}$.
$\boxed{D}$ *whale* $\boxed{CV}$ *in* $\boxed{NP}$.
$\boxed{D}$ *whale* $\boxed{CV}$ *in* $\boxed{D}$ *dish* .
$\boxed{D}$ *whale* $\boxed{CV}$ *in a dish* .
$\boxed{D}$ *whale* $\boxed{Aux}$ *be in a dish* .
$\boxed{S}$.
$\boxed{NP}$ *play* .
$\boxed{D}$ *they play* .
they play .
they play .
they play .
$\boxed{S}$.
$\boxed{NP}$ *boil* $\boxed{NP}$.
$\boxed{NP}$ *boil* $\boxed{D}$ *whiskers* .
$\boxed{NP}$ *boil his whiskers* .
$\boxed{D}$ *they boil his whiskers* .
they boil his whiskers .

Table 7.2: Testing of BLINDADDSUBSTITUTE

Table 7.3 shows the behaviour of BLINDADDADJOIN when it is applied five times to an individual initialized with I_T_K0. It shows that only new sentence 'frames' can be adjoined, as it is the only valid auxiliary tree found in the grammar.

We therefore test BLINDADDADJOIN on an individual that already has existing structure. We

$\boxed{S}$.
$\boxed{S}$. $\boxed{S}$.
$\boxed{S}$. $\boxed{S}$. $\boxed{S}$.
$\boxed{S}$. $\boxed{S}$. $\boxed{S}$. $\boxed{S}$.
$\boxed{S}$. $\boxed{S}$. $\boxed{S}$. $\boxed{S}$. $\boxed{S}$.
$\boxed{S}$. $\boxed{S}$. $\boxed{S}$. $\boxed{S}$. $\boxed{S}$. $\boxed{S}$.

Table 7.3: Testing of BLINDADDADJOIN

first apply BLINDADDSUBSTITUTE five times to an initialized individual, as in Table 7.2, before applying BLINDADDADJOIN five times. The results of three instances are shown in Table 7.4. Unlike the test in Table 7.3, BLINDADDADJOIN now finds many possibilities of adjoining in auxiliary trees. In these instances we can see examples of adjectives, prepositional phrases and sentence frames being adjoined. However, in the last instance we can see that it is struggling to add more content to the sentence *"it is extinct."*, opting to adjoin new sentence frames instead. At this point, it is clear that BLINDADDSUBSTITUTE should be applied to fill these frames with linguistic structure.

mothers play . *tender mothers play .* *tender mothers play in* $\boxed{NP}$. *tender mothers play in* $\boxed{NP}$ *with* $\boxed{NP}$. *tender mothers with* $\boxed{NP}$ *play in* $\boxed{NP}$ *with* $\boxed{NP}$. *tender mothers with* $\boxed{NP}$ *play in* $\boxed{NP}$ *with* $\boxed{NP}$ *in* $\boxed{NP}$.
the table is his trouble . *the bandy table is his trouble .* *in* $\boxed{NP}$ $\boxed{Punc}$ *the bandy table is his trouble .* *in* $\boxed{NP}$ $\boxed{Punc}$ *the bandy table is his platinum trouble .* $\boxed{S}$. *in* $\boxed{NP}$ $\boxed{Punc}$ *the bandy table is his platinum trouble .* $\boxed{S}$. *in* $\boxed{NP}$ $\boxed{Punc}$ *the bandy table is his very platinum trouble .*
it is extinct . $\boxed{S}$. *it is extinct .* $\boxed{S}$. *it is extinct .* $\boxed{S}$. $\boxed{S}$. *it is extinct .* $\boxed{S}$. $\boxed{S}$. $\boxed{S}$. *it is extinct .* $\boxed{S}$. $\boxed{S}$. $\boxed{S}$. $\boxed{S}$. *it is very extinct .* $\boxed{S}$. $\boxed{S}$. $\boxed{S}$.

Table 7.4: Testing of BLINDADDADJOIN on existing derivation

In Table 7.5, we show the results of stochastically applying both BLINDADDSUBSTITUTE and BLINDADDADJOIN. Given an initialized individual, we apply either BLINDADDSUBSTITUTE or BLINDADDADJOIN ten times. Each time, the operator is probabilistically decided. In (a) the probability that the operator applied is BlindAddSubstitute is 0.75, and for BlindAddAdjoin it is 0.25. In (b) it is 0.5 and 0.5 respectively, and in (c) it is 0.25 and 0.75. Unlike Tables 7.2 to 7.4, here we only show the final derivation after ten iterations. For each probability weighting we show five different results. As can be expected, the more BlindAddSubstitute is applied, the less incomplete the derivations are, in the sense that there are fewer remaining substitution nodes. Another effect is that the resulting texts are shorter, e.g. in number of sentences. By handling these two LTAG syntactic operations in separate operators, we can control the intended behaviour by altering the probability weighting of their application.

in $\boxed{NP}$ *, the baboon ,* $\boxed{Comp}$ *possesses* $\boxed{NP}$ $\boxed{Punc}$ *is in* $\boxed{D}$ *stark child .* $\boxed{D}$ *extremely sensitive waist* $\boxed{CV}$ *with* $\boxed{D}$ *it in* $\boxed{NP}$ *. its mothers play .* *in the head , his men play .* $\boxed{S}$ *.* $\boxed{D}$ *they play . the tiger* $\boxed{Aux}$ *be extremely large .* $\boxed{S}$ *.* *in* $\boxed{NP}$ *,* $\boxed{NP}$ *dwells with men in the blubber in* $\boxed{NP}$ *.* (a)
$\boxed{S}$ *. his mothers* $\boxed{Punc}$ *who with* $\boxed{D}$ *families* $\boxed{Punc}$ *play , will play .* $\boxed{D}$ *it is in* $\boxed{D}$ *facts* $\boxed{Punc}$ *that with* $\boxed{NP}$ $\boxed{Punc}$ $\boxed{CV}$ *his knees* $\boxed{Punc}$ *.* $\boxed{NP}$ $\boxed{CV}$ $\boxed{D}$ *town .* *men boil a extinct wilderness* $\boxed{Punc}$ $\boxed{Comp}$ $\boxed{CV}$ *in* $\boxed{NP}$ $\boxed{Punc}$ *.* $\boxed{NP}$ *dwells* $\boxed{P}$ $\boxed{NP}$ *.* $\boxed{S}$ *.* $\boxed{NP}$ *boil* $\boxed{D}$ *soil in* $\boxed{NP}$ *with* $\boxed{NP}$ *.* $\boxed{NP}$ *dwells* $\boxed{P}$ $\boxed{NP}$ *.* $\boxed{S}$ *.* $\boxed{S}$ *.* $\boxed{NP}$ $\boxed{CV}$ $\boxed{D}$ *dish .* *with its child* $\boxed{Punc}$ $\boxed{Comp}$ $\boxed{NP}$ *has* $\boxed{Punc}$ $\boxed{Punc}$ $\boxed{NP}$ $\boxed{CV}$ *the boy .* $\boxed{NP}$ *is with* $\boxed{NP}$ *.* $\boxed{S}$ *.* (b)
$\boxed{NP}$ $\boxed{CV}$ *little . in* $\boxed{NP}$ $\boxed{Punc}$ $\boxed{NP}$ *dwells* $\boxed{P}$ $\boxed{NP}$ *with* $\boxed{NP}$ *with* $\boxed{NP}$ *.* $\boxed{S}$ *.* $\boxed{S}$ *.* $\boxed{NP}$ *possesses* $\boxed{NP}$ *.* $\boxed{S}$ *.* $\boxed{NP}$ *possesses* $\boxed{NP}$ *. with* $\boxed{NP}$ *,* $\boxed{NP}$ $\boxed{Aux}$ *be extremely round with* $\boxed{NP}$ *with* $\boxed{NP}$ *.* $\boxed{NP}$ *dwells* $\boxed{P}$ $\boxed{NP}$ *.* $\boxed{S}$ *.* $\boxed{S}$ *.* $\boxed{NP}$ *play in* $\boxed{NP}$ *in* $\boxed{NP}$ *.* $\boxed{S}$ *.* $\boxed{S}$ *.* $\boxed{S}$ *.* $\boxed{S}$ *. in* $\boxed{NP}$ $\boxed{Punc}$ *with* $\boxed{NP}$ $\boxed{Punc}$ *with* $\boxed{NP}$ $\boxed{Punc}$ $\boxed{NP}$ *resides* $\boxed{P}$ $\boxed{D}$ *grin in* $\boxed{NP}$ *with* $\boxed{NP}$ *.* $\boxed{S}$ *.* $\boxed{S}$ *.* $\boxed{S}$ *.* $\boxed{S}$ *. in* $\boxed{NP}$ $\boxed{Punc}$ *in* $\boxed{NP}$ $\boxed{Punc}$ *his toad dwells in* $\boxed{NP}$ *with* $\boxed{NP}$ *with* $\boxed{NP}$ *.* (c)

Table 7.5: Testing of BlindAddSubstitute and BlindAddAdjoin with probabilities of (a) 0.75 and 0.25, (b) 0.5 and 0.5, and (c) 0.25 and 0.75

In Table 7.6 we show three instances of applying the BlindDelete operator. We first create an individual with sufficient linguistic information using the BlindAdd operators, and show the individual before and after applying BlindDelete. In the first instance, the entire last sentence is deleted. In the second instance, only the determiner *'the'* is deleted. In the last instance, the adverb *'extremely'* is deleted. In each instance we can see that the derivations after the deletion remain grammatical.

Before: *they play with his species , that possesses her , with its trouble . its sense* $\boxed{CV}$ *his tail .* **After:** *they play with his species , that possesses her , with its trouble .* $\boxed{S}$ *.*
Before: *mothers play . the enormous mothers play . a man possesses her .* **After:** *mothers play .* $\boxed{D}$ *enormous mothers play . a man possesses her .*
Before: *its boy is extremely african with the tail with them . the men play .* **After:** *its boy is african with the tail with them . the men play .*

Table 7.6: Testing of BLINDDELETE

In Table 7.7 we show three instances of applying the BLINDSWAP operator. We first create two different individuals with sufficient linguistic information by stochastically applying the BLINDADD operators to two initialized individuals. For each instance, we show the two individuals before and after applying BLINDSWAP. In the first instance, two whole sentences are swapped. In the second instance, two noun phrases are swapped, i.e.*"its trunk, that is him"* and *"its head, that is grim"*. Note that in the third instance, although two sentences are swapped, their positions relative to the derivation are different. This is because in our grammar we have two 'sentence frame' auxiliary trees: one that adjoins to the left of the derivation, and one that adjoins to the right. In the first text, the swapped sentence is *"the product is his whale."*, and we can see that it appears at the end of the second text after the swap. Conversely, in the second text, the swapped sentence is *"its jaws will be platinum."*, and we can see that it appears at the beginning of the first text after the swap. For all these instances, the derivations after applying BLINDSWAP remain grammatical.

Before: *the skin will be with a fish . they play .* *the frog dwells in a african town .* **After:** *the skin will be with a fish . the frog dwells in a african town .* *they play .*
Before: *his enormous waistline , it will not be with its trunk , that is him , .* *its head , that is grim , will be with its town .* $\boxed{NP}$ *will be with* $\boxed{NP}$ *. the tiger* $\boxed{CV}$ $\boxed{D}$ *hand .* **After:** *his enormous waistline , it will not be with its head , that is grim , .* *its trunk , that is him , will be with its town .* $\boxed{NP}$ *will be with* $\boxed{NP}$ *. the tiger* $\boxed{CV}$ $\boxed{D}$ *hand .*
Before: *the boy dwells in a boy . the product is his whale .* *its jaws will be platinum . with its product , that possesses* $\boxed{D}$ *table , , a hippopotamus possesses his product .* **After:** *its jaws will be platinum . the boy dwells in a boy .* *with its product , that possesses* $\boxed{D}$ *table , , a hippopotamus possesses his product . the product is his whale .*

Table 7.7: Testing of BlindSwap

7.3 Smart operators: achieving meaningfulness

Assuming that our EA employs the evaluation functions presented in Chapter 6, the three baseline operators presented in the previous section, BlindAdd, BlindDelete, and BlindSwap, should suffice as the set of genetic operators to be applied to individuals in the population. These operators ensure that the individuals satisfy the property of grammaticality, while the metre similarity (Section 6.3) and semantic similarity (Section 6.4) evaluation functions guide the EA search towards optimization of the poeticness and meaningfulness properties.

Although one of the central principles of the theory of evolution is that populations are changed by purposeless forces, i.e. random genetic variation and natural selection (Dawkins, 1991), if there exists domain knowledge that can be exploited, we need not limit our artificial evolutionary system by not using it simply for the sake of remaining faithful to evolution as it occurs in nature.

In Section 4.5, particularly Section 4.5.2, we mentioned the possibility of using knowledge augmented operators, i.e. operators that are not just simple stochastic manipulators of an individual's genetic code, but ones that explicitly and purposefully exploit domain knowledge to solve the problem at hand. One of the most common approaches to the design of knowledge augmented operators is the combination of EAs with **local search** (Goldberg, 1989, Bäck et al., 1997, Michalewicz, 1996). In this approach, given an individual representing the candidate solution x, genetic operations implement a search algorithm over the restricted space $N(x)$, called the **neighbourhood** of x. Formally, $N(x)$ can be an arbitrary subset of the entire search space, but typically it is defined as a set of solutions obtained from x by manipulating it in a specified manner. A candidate solution x is **locally optimal** if there is no $y \in N(x)$ such that $f(y) > f(x)$, where f is the objective function being optimized.

This local search in the space of $N(x)$ itself can be either stochastic or systematic (i.e. exhaustive). Additionally, variations of local search can either take the first improved solution $y \in N(x)$ where $f(y) > f(x)$, or the best one.

In the last few sections we have in fact already presented genetic operators that employ local search. Recall that we have chosen to enforce grammaticality because we believe that an operator that allows any sequence of lexical items to appear as a candidate solution would cause the EA search to be very inefficient, and that furthermore, we do not wish to invoke poetic license as a justification for ungrammatical texts. The baseline operators BLINDADD, BLINDDELETE, and BLINDSWAP all perform some degree of local search to adhere to syntactic well-formedness as legislated by the linguistic resources. The local search is implicit in the stochastic depth-first searching for legal syntactic elements, e.g. derivation tree nodes, elementary trees, words. There is no objective function $f(x)$ being optimized, as we treat the property of grammaticality as discrete: a text is either grammatical or ungrammatical.

In the next few sections we will describe local search operators inspired by works mentioned in Section 3.4.1 such as Nicolov et al. (1996), Nicolov (1998), Stone and Doran (1997), Stone et al. (2001) that explicitly and purposefully satisfy meaningfulness. We will also compare our operators with these systems in Section 7.3.7.

7.3.1 The principle of semantic consumption

The BLINDADD operator exploits linguistic knowledge to ensure that only syntactically licensed substitutions and adjunctions are considered. However, given a target semantics of $\{love(l, j, m), john(\_, j), mary(\_, m)\}$, there seems little point in substituting the noun phrase *“big pizza”* as seen in Figure 7.2. Therefore, our smart operator should only consider substituting or adjoining elementary trees which are anchored by words that bring the semantics of a candidate solution closer to that of the target semantics.

In other words, we can informally describe our semantically smart operator, call it SMARTADD, as one that explicitly tries to realize the target semantics, specifically that which has not yet been realized, while simultaneously maintaining syntactic well-formedness. Nicolov calls this gradual process the **consumption** of semantics.

Note that the baseline operator BLINDADD in Section 7.2.1 serves as a blueprint for SMARTADD, as it already guarantees grammaticality. All we need to describe here is how to further constrain the search to only consider LTAG operations that improve a candidate solution's semantics. In particular, given the modular approach with which we have designed and implemented BLINDADD, essentially we only need to modify the `WordFilter` function.

Over and above the enforcement of grammaticality implemented by BLINDADD, SMARTADD must also

1. compute the unrealized semantics, i.e. the portion of the target semantics, S_{target}, that is not yet already conveyed by the candidate solution's semantics, $S_{candidate}$,
2. identify lexical items that convey a subset of the unrealized semantics, and
3. ensure that operations not only select the appropriate lexical items, but also place them in the right context in the text represented by the candidate solution, i.e. one where it satisfies the predicate-argument structure of the target semantics.

We will now discuss these three processes in detail in Sections 7.3.2 to 7.3.4.

7.3.2 Computing the unrealized semantics

If SMARTADD is to take an individual and modify it in such a way that the semantics it conveys, $S_{candidate}$, better realizes the target, S_{target}, it should know what portion of S_{target} has not yet been realized by the current $S_{candidate}$. This portion is called the unrealized semantics, $S_{unrealized}$. At the beginning of our generation process, $S_{candidate}$ is empty, and $S_{unrealized}$ is equal to S_{target}. For example, given

$$S_{target} = \{love(l,j,m), john(\_,j), mary(\_,m)\}$$

and an individual representing the incomplete derivation “$\boxed{NP}$ *loves Mary*”, e.g. the first derivation tree in Figure 7.2, where

$$S_{candidate} = \{love(\_0,\_1,\_2), mary(\_3,\_2)\},$$

then clearly $S_{unrealized} = \{john(\_, j)\}$, i.e. the actor of the $love$ action.

As semantic expressions are sets of literals, the simplest way of calculating $S_{unrealized}$ is to use set subtraction, specifically as used in the computation of the first term, Θ, of our similarity equation (Section 6.4.3), where we decide membership based solely on functor equality:

$$S_{unrealized} = S_{target} - S_{candidate}$$

However, there are several cases where this results in an incorrect $S_{unrealized}$. In Figure 7.9 we present two individuals which represent the incomplete texts
(a) “$\boxed{NP}$ *loves* $\boxed{NP}$. $\boxed{D}$ *Mary runs.*” and
(b) “*Mary loves* $\boxed{NP}$.”.

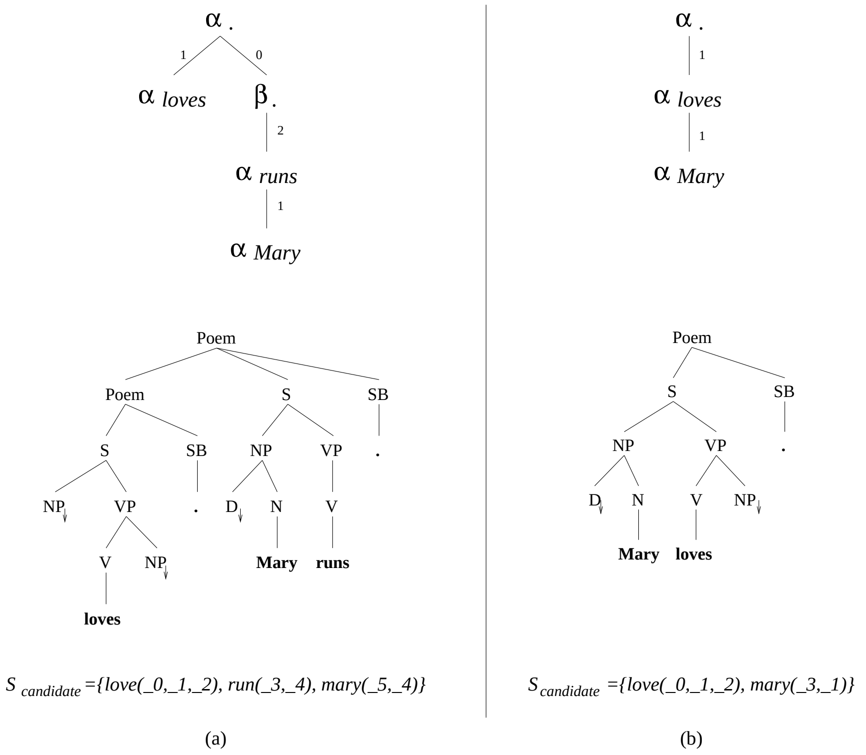


Figure 7.9: Individuals where literals in $S_{candidate}$ do not properly convey S_{target}

Note that $\alpha_{.}$ and $\beta_{.}$ are instances of **sentence frame** elementary trees in our grammar, which are rooted by the distinguished category, *Poem*, and introduce a sentential substitution node, $S_{\downarrow}$. They are anchored by sentence boundary punctuation marks, i.e. fullstops.

Given the target semantics $S_{target} = \{love(l, j, m), john(\_, j), mary(\_, m)\}$, set substraction would yield the unrealized semantics $S_{unrealized} = \{john(\_, j)\}$ for both these individuals, as their semantics contain literals with the functors $love$ and $mary$.

However, in (a) we can see that *'Mary'* is the subject of the second sentence, i.e.*"Mary runs"*, and therefore is unrelated to the *'loves'* verb. This is reflected in the $S_{candidate}$, where the object entity that represents $mary$ is $\_4$ and the object of $love$ is $\_2$. Similarly, in (b) *'Mary'* is the subject of *'love'*, and not the object, as is required. In the $S_{candidate}$, the object entity of $mary$ is $\_1$, and it occupies the second argument position of $love$ (compare this with S_{target}, where m is the third argument). Thus in both cases, $mary(\_, m)$ should be part of $S_{unrealized}$, as the substitution of *'Mary'* at the object NP position of the transitive verb *'loves'* would improve $S_{candidate}$.

To achieve this, one alternative method for computing $S_{unrealized}$ is to use the semantic mapping algorithm described in Section 6.4.2. By mapping S_{target} and $S_{candidate}$, $S_{unrealized}$ is calculated as a byproduct, and can be found in the set of all target literal dangling matches. In both cases shown in Figure 7.9, this method yields the correct $S_{unrealized} = \{john(\_, j), mary(\_, m)\}$. Note that the argument binding carried out by this semantic mapping algorithm is not unification, but strict symbol equality, hence it cannot include $mary(\_5, \_4)$ in a structurally consistent mapping once $love(\_0, \_1, \_2)$ has been mapped. This method is the method we have adopted in our implementation of MCGONAGALL.

7.3.3 Semantically-aware lexical choice

Given $S_{unrealized}$, we can now augment BLINDADD to only consider words whose lexical semantics, $S_{lexical}$, convey a subset of this set. If we observe the stochastic depth-first search as shown in Figure 7.4, we can see that the only stage that needs to be augmented is the last stage, the selection of l, the word that will anchor t_g. More specifically, we must augment the functionality of `WordFilter`, which in BLINDADD only checks for unification of features structures concerning agreement, subcategorization, selectional restrictions, etc.

Once again, the simplest method would be to use set operations where we decide membership based solely on functor equality (Section 6.4.3), and in this instance we can check that if

$$S_{lexical} \cap S_{unrealized} \neq \emptyset,$$

then a word can potentially improve the semantics of a candidate solution. For instance, given $S_{unrealized} = \{john(\_, j), mary(\_, m)\}$, then the noun *'John'*, whose $S_{lexical} = \{john(X,Y)\}$, is an appropriate choice, whereas *'pizza'* ($S_{lexical} = \{pizza(X,Y)\}$ is not.

7.3.4 Smart signatures: grounding $S_{lexical}$ in terms of S_{target}

The semantically-motivated lexical choice presented in the previous section does not yet guarantee that the resulting operation is a definite realization of the target semantics. One outstanding issue to be resolved is that of predicate-argument structure. There are two principles that can be applied here:

1. SMARTADD must ensure that the semantics of the chosen word, l, within the context of the resulting $S_{candidate}$ does not **violate** the predicate-argument structure as found in S_{target}. For example, by substituting *'Mary'* as the subject of the transitive verb *'loves'* when it should be the object.
2. Whenever possible, SMARTADD should **match** the predicate-argument structure in S_{target}. An instance of this would be the correct substitution of *'Mary'* as the object of *'loves'*, instead of at an unrelated site in the derivation.

These two principles are similar to what Nicolov calls the criteria of **integration** and **connectivity** (Nicolov et al., 1996).

Figure 7.10(a) shows an example of these principles. The derived tree represents the incomplete text "$\boxed{NP}$ *loves* $\boxed{NP}$. $\boxed{NP}$ *runs*.". Having calculated $S_{unrealized}$ and determined that the noun *'Mary'* conveys a subset of it, the operator is now trying to substitute α_{Mary} into the derived tree. As we can see, there are three nodes where α_{Mary} can be substituted, i.e. the subject and object positions of *'loves'* and the subject of *'runs'*. We have included the semantic signatures at these positions (see Section 5.3.8). Substituting α_{Mary} at the subject position of *'loves'* violates the predicate-argument structure in S_{target}. Although substituting it at the subject position of *'runs'* does not violate the predicate-argument structure, it also does not match it. Only the substitution of α_{Mary} at the object position of *'loves'* will result in a realization of S_{target}.

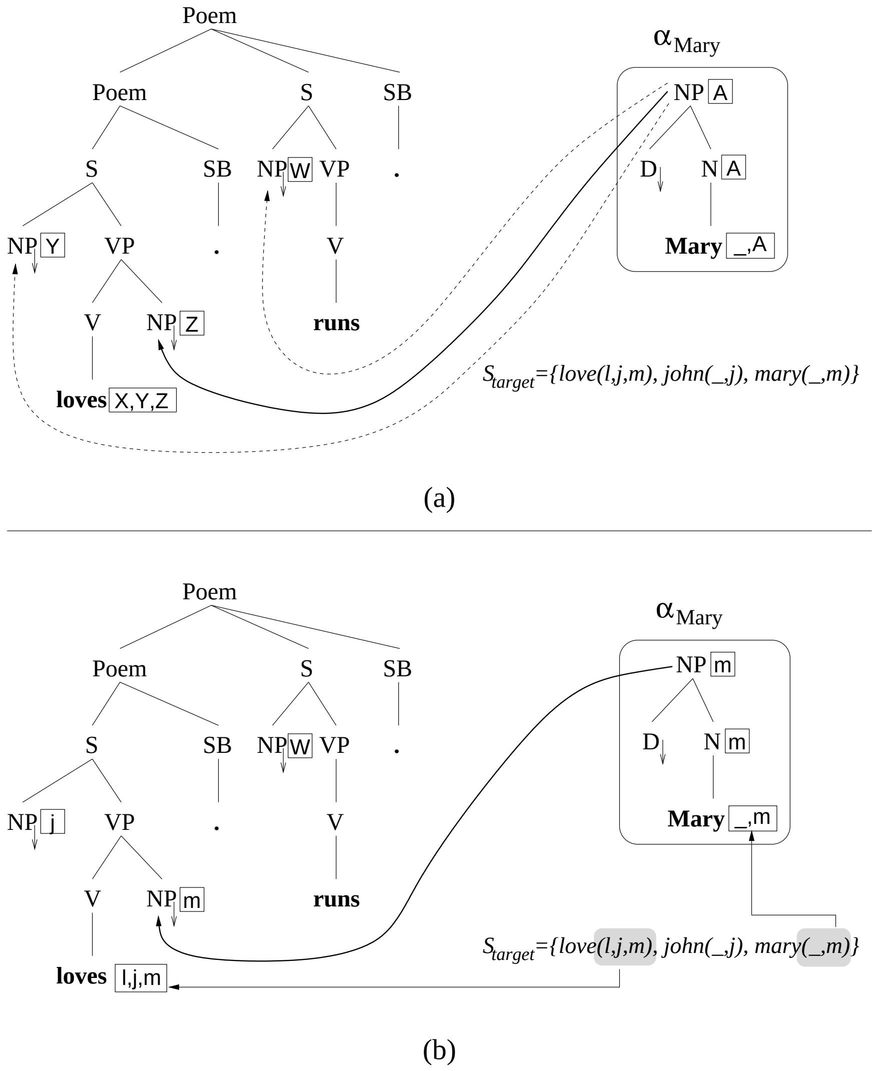


Figure 7.10: (a) SMARTADD should not violate the predicate-argument structure of S_{target}, (b) Smart signatures derived from S_{target} facilitate this

To implement these principles, the lexical items in the derivation must be associated with information denoting which literals in S_{target} they are conveying. More precisely, for SMARTADD to be able to check for violation and matching of predicate-argument structure, the signature variables must be bound to the arguments of the literals they are conveying in S_{target}. We shall call these **smart signatures**, and an example of this is given in Figure 7.10(b). Crucially, the arguments of the literals in S_{target} are defined as constants, and appear in lower case. In this example, we can see that the signature of *'loves'* in the derived tree has been unified with the arguments (l, j, m), and as a consequence the signatures at the subject and object positions now indicate the object entity they are expecting to be substituted for. Similarly, the signature of *'Mary'* in α_{Mary} has been unified with the arguments $(\_, m)$. At this point, α_{Mary} can no longer appear at the subject position of *'loves'*, because the constants j and m do not unify. Thus the first principle above, of non-violation, is achieved. It can, however, still be substituted at the subject position of *'runs'*. We can impose a further constraint that not only must signatures unify, they must also share a constant. This achieves the second principle above, of matching predicate-argument structure. Obviously, this can not always be satisfied, for example at the beginning of the generation process when nothing has yet been realized. Therefore, we create two versions of this smart operator, SMARTADD, which merely checks that signatures unify, and SMARTADDMATCH, which requires that they must also share a constant.

How are these smart signatures obtained? As in the case of calculating $S_{unrealized}$ in Section 7.3.2, we can achieve this by moving from simple set operations to the semantic mapping algorithm in Section 6.4.2. By mapping $S_{lexical}$ with $S_{unrealized}$ to check whether a given word conveys a subset of the unrealized semantics, we will also obtain the appropriate argument bindings as a byproduct.

Note that both these smart signatures and the 'regular' signatures must be kept, as it is the regular signatures that are used for deriving $S_{candidate}$ for evaluation purposes. This is because the smart signatures can falsely establish predicate-argument structure that does not arise from the unifications described in Section 5.3.8. For instance, consider the case if SMARTADD substituted α_{Mary} as the subject of *'runs'* in Figure 7.10(b). If we only kept the smart signatures, the $S_{candidate}$ obtained from this derived tree would be $\{love(l, j, m), mary(\_, m), runs(\_, m)\}$, when clearly the text does not state that *'Mary'* is the object of *'loves'*.

7.3.5 Semantically-motivated nonmonotonic operations

By observing the the principle of semantic consumption (Section 7.3.1), one can imagine a complementary principle of "semantic elimination". This can be used to create a semantically-motivated SMARTDELETE operator, i.e. one that removes subtrees of a derivation tree that do not contribute to the conveying of S_{target}.

As before, we can either use simple set substraction to obtain the subset of $S_{candidate}$ that is extraneous to the task, i.e.

$$S_{extraneous} = S_{candidate} - S_{target},$$

or we can use the semantic mapping algorithm in Section 6.4.2. By mapping S_{target} and $S_{candidate}$, $S_{extraneous}$ is calculated as a byproduct, and can be found in the set of all candidate literal dangling matches.

In the same way that SMARTADD is similar to BLINDADD with the exception that its `WordFilter` explicitly attempts to convey $S_{unrealized}$, SMARTDELETE is thus the same as BLINDDELETE with the exception that its `DNodeFilter` only considers nodes whose elementary trees are anchored by words with $S_{lexical} \in S_{extraneous}$.

Similarly, we can create a SMARTSWAP operator that is similar to BLINDSWAP with the exception that its `verifySwap` also checks that the proposed swap does not violate unification of the relevant smart signatures.

7.3.6 Testing the smart operators

We will now present the results of some simple testing performed with all the smart operators implemented in MCGONAGALL. Table 7.8 shows a roundup of these operators.

In all these tests, we set S_{target} as the encoding of the first two lines of Belloc's *"The Lion"*, but with a slight alteration where we have replaced the original opening noun phrase, *"the lion, the lion"*, with *"the african lion"*. This is actually one of the target semantics used in our empirical study (Chapter 8), and is shown in Appendix A as the `lionhalf` target. We repeat it here as Figure 7.11 for convenience.

Firstly, Table 7.9 shows three instances of applying SMARTADDSUBSTITUTE five times on a newly initialized individual. The individual is shown after every application. Note that all

Name	Description
SMARTADDSUBSTITUTE	The same as BLINDADDSUBSTITUTE, except the `WordFilter` now chooses words that explicitly convey $S_{unrealized}$ according to the principle of not violating predicate-argument structure (Section 7.3.4).
SMARTADDADJOIN	The same as BLINDADDADJOIN, except the `WordFilter` now chooses words that explicitly convey $S_{unrealized}$ according to the principle of not violating predicate-argument structure (Section 7.3.4).
SMARTADDMATCHSUBSTITUTE	The same as BLINDADDSUBSTITUTE, except the `WordFilter` now chooses words that explicitly convey $S_{unrealized}$ according to the principle of matching predicate-argument structure (Section 7.3.4).
SMARTADDMATCHADJOIN	The same as BLINDADDADJOIN, except the `WordFilter` now chooses words that explicitly convey $S_{unrealized}$ according to the principle of matching predicate-argument structure (Section 7.3.4).
SMARTDELETE	The same as BLINDDELETE, except the `DNodeFilter` only considers nodes whose elementary trees are anchored by words that convey $S_{extraneous}$ (Section 7.3.5).
SMARTSWAP	The same as BLINDSWAP, except the `verifySwap` function also checks that the proposed swap does not violate unification of smart signatures (Section 7.3.5).

Table 7.8: Roundup of smart operators

the resulting derivations are still incomplete, unlike in Table 7.2 where only the first one was incomplete. Moreover, the fact that in each instance the last few derivations remain unchanged indicates that SMARTADDSUBSTITUTE is in fact unable to find any more valid operations. This is understandable when we consider that it requires a lexical item that conveys a subset of $S_{unrealized}$. In each of these instances, the final derivation does not provide opportunity for this.

In the last instance we see an example where the principle of not violating the predicate-argument structure is met, but not that of matching it (Section 7.3.4). This can be seen in the substituting of *'lion'* at the prepositional phrase following *'dwells'*. Although this operation does not violate the predicate-argument strucure from the perspective of the local search carried out, it is clear that it will prevent S_{target} from being conveyed, where the prepositional phrase should be *"in the waste"*, and *'lion'* should appear as the subject of *'dwells'*. This occurs because in our representation, the predicate-argument strucure between $dwells(d,l)$ and

"The african lion, he dwells in the waste,
He has a big head and a very small waist."

$$\begin{array}{lll} S_{target} & = & \{african(\_,l), lion(\_,l), dwell(d,l), inside(\_,d,was), waste(\_,was), \\ & & own(\_,l,h), head(\_,h), big(\_,h), \\ & & own(\_,l,wai), small(s,wai), waist(\_,wai), very(\_,s)\} \end{array}$$

Figure 7.11: The target semantics used for testing in this section

$\boxed{S}$.
$\boxed{NP}$ *dwells* $\boxed{P}$ $\boxed{NP}$.
$\boxed{D}$ *lion dwells* $\boxed{P}$ $\boxed{NP}$.
$\boxed{D}$ *lion dwells in* $\boxed{NP}$.
$\boxed{D}$ *lion dwells in* $\boxed{D}$ *waste* .
$\boxed{D}$ *lion dwells in* $\boxed{D}$ *waste* .
$\boxed{S}$.
$\boxed{NP}$ $\boxed{CV}$ $\boxed{D}$ *waste* .
$\boxed{NP}$ $\boxed{CV}$ $\boxed{D}$ *waste* .
$\boxed{NP}$ $\boxed{CV}$ $\boxed{D}$ *waste* .
$\boxed{NP}$ $\boxed{CV}$ $\boxed{D}$ *waste* .
$\boxed{NP}$ $\boxed{CV}$ $\boxed{D}$ *waste* .
$\boxed{S}$.
$\boxed{NP}$ *dwells* $\boxed{P}$ $\boxed{NP}$.
$\boxed{NP}$ *dwells* $\boxed{P}$ $\boxed{D}$ *lion* .
$\boxed{NP}$ *dwells* $\boxed{P}$ $\boxed{D}$ *lion* .
$\boxed{NP}$ *dwells* $\boxed{P}$ $\boxed{D}$ *lion* .
$\boxed{NP}$ *dwells* $\boxed{P}$ $\boxed{D}$ *lion* .

Table 7.9: Testing of SMARTADDSUBSTITUTE

$waste(\_,was)$ is indirectly represented by $inside(\_,d,was)$. Had the preposition *'in'* been substituted at $\boxed{P}$ first, *'lion'* would never be substituted as its object, as it would then result in a violation. This issue is the difference between SMARTADDSUBSTITUTE and SMARTADDMATCHSUBSTITUTE, which enforces both the principles mentioned in Section 7.3.4.

When testing SMARTADDADJOIN, SMARTADDMATCHSUBSTITUTE, and SMARTADDMATCHADJOIN in a similar fashion, i.e. applying them five times on an initialized individual, we obtain the behaviour exhibited shown in Table 7.10. Recall from Table 7.3 that BLINDADDADJOIN could only adjoin in sentence frames, and since they do not convey any semantics, we can see why the adjunction operators here fail to introduce anything new. In the case of SMARTADDMATCHSUBSTITUTE, recall that it is required to satisfy the principle of matching the

$\boxed{S}$.
$\boxed{S}$.
$\boxed{S}$.
$\boxed{S}$.
$\boxed{S}$.
$\boxed{S}$.

Table 7.10: The “dead-end” encountered by SmartAddAdjoin, SmartAddMatchSubstitute, and SmartAddMatchAdjoin

predicate-argument structure. As there is nothing yet to match with in the derivation, it fails to introduce anything new.

Similar to what we did in Table 7.4 for BlindAddAdjoin, in Table 7.11 we show the results of applying SmartAddAdjoin to individuals that already have existing structure. We first apply SmartAddSubstitute five times to an initialized individual, as in Table 7.9, before applying SmartAddAdjoin five times. These instances show that SmartAddAdjoin now finds several possibilities of adjoining in auxiliary trees that convey a subset of the computed $S_{unrealized}$, such as relative clauses (e.g. “$\boxed{Punc}$ $\boxed{Comp}$ $\boxed{CV}$ *african* $\boxed{Punc}$”, which, once fully realized, would be something like *“, that is african,”*), and adjectives (e.g.*‘big’*, *‘african’*). Note that the adjunction of *‘very’* in all three instances does not violate the predicate-argument structure, but only in the last instance does it match, e.g.*“very small waist”*.

In Table 7.12 we show the results of applying SmartAddMatchSubstitute and SmartAddMatchAdjoin on an initialized individual that has been altered by SmartAddSubstitute. The transitive verb *‘has’* provides these operators with something to match with, unlike in Table 7.10. We show the state of the individual after applying SmartAddMatchSubstitute five times, followed by applying SmartAddMatchAdjoin five times. We can see that SmartAddMatchSubstitute fills in the required complements *‘lion’* and *‘head’* and nothing else. Subsequently, SmartAddMatchAdjoin adjoins *‘african’* as an adjective and *‘big’* and *‘has’* as relative clauses. This second *‘has’* conveys the $own(\_, l, wai)$ literal in S_{target}. If we were to complete all the purely syntactic substitution nodes present in the last derivation, such as determiner ($\boxed{D}$), punctuation ($\boxed{Punc}$), complementizer ($\boxed{Comp}$), and copula ($\boxed{CV}$), we would obtain the following sentence:

“The african lion, that has $\boxed{NP}$*, has a head, which is big,.”*

where *“a very small waist”* can potentially be substituted at the $\boxed{\text{NP}}$ substitution node. Note

$\boxed{D}$ *lion dwells in* $\boxed{D}$ *waste .*
$\boxed{D}$ *lion* $\boxed{Punc}$ $\boxed{Comp}$ $\boxed{CV}$ *african* $\boxed{Punc}$ *dwells in* $\boxed{D}$ *waste .*
$\boxed{D}$ *lion* $\boxed{Punc}$ $\boxed{Comp}$ $\boxed{CV}$ *very african* $\boxed{Punc}$ *dwells in* $\boxed{D}$ *waste .*
$\boxed{D}$ *lion* $\boxed{Punc}$ $\boxed{Comp}$ $\boxed{CV}$ *very african* $\boxed{Punc}$ *dwells in* $\boxed{D}$ *waste .*
$\boxed{D}$ *lion* $\boxed{Punc}$ $\boxed{Comp}$ $\boxed{CV}$ *very african* $\boxed{Punc}$ *dwells in* $\boxed{D}$ *waste .*
$\boxed{D}$ *lion* $\boxed{Punc}$ $\boxed{Comp}$ $\boxed{CV}$ *very african* $\boxed{Punc}$ *dwells in* $\boxed{D}$ *waste*
$\boxed{D}$ *lion has* $\boxed{D}$ *head .*
in $\boxed{NP}$ $\boxed{Punc}$ $\boxed{D}$ *lion has* $\boxed{D}$ *head .*
in $\boxed{NP}$ $\boxed{Punc}$ $\boxed{D}$ *lion has* $\boxed{D}$ *big head .*
in $\boxed{NP}$ $\boxed{Punc}$ $\boxed{D}$ *lion has* $\boxed{D}$ *very big head .*
in $\boxed{NP}$ $\boxed{Punc}$ $\boxed{D}$ *lion* $\boxed{Punc}$ $\boxed{Comp}$ *dwells* $\boxed{P}$ $\boxed{NP}$ $\boxed{Punc}$ *has* $\boxed{D}$ *very big head .*
in $\boxed{NP}$ $\boxed{Punc}$ $\boxed{D}$ *african lion* $\boxed{Punc}$ $\boxed{Comp}$ *dwells* $\boxed{P}$ $\boxed{NP}$ $\boxed{Punc}$ *has* $\boxed{D}$ *very big head .*
his waist $\boxed{CV}$ *its waist .*
his waist $\boxed{CV}$ *its small waist .*
his waist $\boxed{CV}$ *its small waist in* $\boxed{NP}$ *.*
his waist $\boxed{CV}$ *its small waist* $\boxed{Punc}$ $\boxed{Comp}$ $\boxed{CV}$ *small* $\boxed{Punc}$ *in* $\boxed{NP}$ *.*
his small waist $\boxed{CV}$ *its small waist* $\boxed{Punc}$ $\boxed{Comp}$ $\boxed{CV}$ *small* $\boxed{Punc}$ *in* $\boxed{NP}$ *.*
his very small waist $\boxed{CV}$ *its small waist* $\boxed{Punc}$ $\boxed{Comp}$ $\boxed{CV}$ *small* $\boxed{Punc}$ *in* $\boxed{NP}$ *.*

Table 7.11: Testing of SMARTADDADJOIN on existing derivation

that with these SMARTADDMATCH operators, there are no operations at unrelated positions in the derivation that can hinder realization of S_{target}, such as the substitution of *'lion'* in the third instance of Table 7.9.

In Table 7.13, we show the results of stochastically applying both SMARTADDSUBSTITUTE and SMARTADDADJOIN. Given an initialized individual, we apply an operator twenty times. Each time, the operator is probabilistically decided. In (a) the probability that the operator applied is SMARTADDSUBSTITUTE is 0.75, and for SMARTADDADJOIN it is 0.25. In (b) it is 0.5 and 0.5 respectively, and in (c) it is 0.25 and 0.75. For each probability weighting we show five different results, and we only show the final derivation after the twenty iterations.

Unlike the stochastic application of baseline operators in Table 7.5, the difference between weightings is not as marked. This is because the smart operators are limited by the opportunities that arise, whereas BLINDADDSUBSTITUTE can usually find a random substitution node to substitute, and BLINDADDADJOIN can always find a random internal node to adjoin to, hence the distinct difference in terms of completeness of derivations.

It is clear that these smart operators should be used together with other operators, such as the baseline operators in Section 7.2, in order to yield complete texts. We discuss this issue in Section 7.4.

NP has NP .

NP has D head .

D lion has D head .

D lion has D head .

D lion has D head .

D lion has D head .

D lion has D head Punc Comp CV big Punc .

D african lion has D head Punc Comp CV big Punc .

D african lion Punc Comp has NP Punc has D head Punc Comp CV big Punc .

D african lion Punc Comp has NP Punc has D head Punc Comp CV big Punc .

D african lion Punc Comp has NP Punc has D head Punc Comp CV big Punc .

Table 7.12: Testing SMARTADDMATCHSUBSTITUTE and SMARTADDMATCHADJOIN on existing derivation

D very african lion has D big head in D waste .

D lion Punc Comp CV african Punc has D head Punc Comp CV big Punc in D waste .

NP CV its very big head in D waste .

in NP Punc NP CV D very african lion Punc Comp dwells P D waste Punc .

NP CV in D waste .

(a)

D very african lion Punc Comp has D big head Punc dwells in D waste .

NP CV in D waste .

D lion Punc Comp dwells in D waste Punc CV very african .

its very small waist Punc Comp in D waste Punc CV his small waist Punc CV its small waist Punc Comp NP has Punc .

NP CV in D waste .

(b)

D lion Punc Comp has NP Punc dwells in D waste .

NP CV D waste in NP .

its small waist Punc Comp NP has Punc CV his very small waist Punc Comp D african lion has in D waste Punc .

D head Punc Comp D very african lion has Punc CV big in D waste .

D very african lion Punc Comp has NP Punc dwells P its small waist Punc Comp CV small Punc in D waste .

(c)

Table 7.13: Testing of SMARTADDSUBSTITUTE and SMARTADDADJOIN with probabilities of (a) 0.75 and 0.25, (b) 0.5 and 0.5, and (c) 0.25 and 0.75

Before: *his pole is his very big head in his waste .* **After:** $\boxed{\textit{NP}}$ *is his very big head in his waste .*
Before: *in its waste , his very african lion , who has his small waist , dwells with his waist .* **After:** *in its waste , his very african lion , who has his small waist , dwells* $\boxed{\textit{P}}$ *his waist .*

Table 7.14: Testing SmartDelete on an individual constructed using a combination of baseline and smart operators

Tables 7.14 and 7.15 show the effect of applying SmartDelete and SmartSwap. Recall from Section 7.3.5 that these nonmonotonic operators behave very similarly to their baseline counterparts, with the exception that they are sensitive towards the smart signatures at each node in the derivation tree.

If we test SmartDelete on an individual that is entirely constructed by smart operators, it will fail to find any candidates for removal. Thus, we construct an individual using a combination of both baseline and smart operators. In Table 7.14 we show two instances of individuals that are created by first stochastically applying SmartAddSubstitute and SmartAddAdjoin, and then applying BlindAddSubstitute to ‘close’ the derivation. In the first instance, SmartDelete removes the noun phrase *“his pole”*, which is not semantically motivated.

For SmartSwap, we want to test that it can find candidates for subtree swapping that maintain the unification of smart signatures. In Table 7.15 we show two instances of swapping between two individuals that are constructed stochastically by applying SmartAddSubstitute and SmartAddAdjoin. In the first instance, the relative clause “$\boxed{\textit{Punc}}$ $\boxed{\textit{Comp}}$ *has* $\boxed{\textit{D}}$ *big head* $\boxed{\textit{Punc}}$” (e.g.*“, that has a big head,”*) is swapped with the relative clause “$\boxed{\textit{Punc}}$ $\boxed{\textit{Comp}}$ $\boxed{\textit{CV}}$ *african* $\boxed{\textit{Punc}}$” (e.g.*“, that is african,”*). Note that both these relative clauses relate to *l*, the lion, thus the swap is semantically licensed. In the second instance, the noun phrase *“his head”* is swapped with “$\boxed{\textit{D}}$ *big head”*, which is also a semantically licensed operation.

Before: $\boxed{D}$ *african lion* $\boxed{Punc}$ $\boxed{Comp}$ *has* $\boxed{D}$ *big head* $\boxed{Punc}$ *dwells in* $\boxed{D}$ *waste .* $\boxed{D}$ *lion* $\boxed{Punc}$ $\boxed{Comp}$ $\boxed{CV}$ *african* $\boxed{Punc}$ *has* $\boxed{D}$ *head* $\boxed{Punc}$ $\boxed{Comp}$ $\boxed{CV}$ *big* $\boxed{Punc}$ *.* **After:** $\boxed{D}$ *african lion* $\boxed{Punc}$ $\boxed{Comp}$ $\boxed{CV}$ *african* $\boxed{Punc}$ *dwells in* $\boxed{D}$ *waste .* $\boxed{D}$ *lion* $\boxed{Punc}$ $\boxed{Comp}$ *has* $\boxed{D}$ *big head* $\boxed{Punc}$ *has* $\boxed{D}$ *head* $\boxed{Punc}$ $\boxed{Comp}$ $\boxed{CV}$ *big* $\boxed{Punc}$ *.*
Before: *his head* $\boxed{CV}$ *big .* *in* $\boxed{D}$ *waste* $\boxed{Punc}$ $\boxed{D}$ *african lion* $\boxed{Punc}$ $\boxed{Comp}$ *has* $\boxed{D}$ *big head* $\boxed{Punc}$ *dwells* $\boxed{P}$ $\boxed{NP}$ *.* **After:** $\boxed{D}$ *big head* $\boxed{CV}$ *big .* *in* $\boxed{D}$ *waste* $\boxed{Punc}$ $\boxed{D}$ *african lion* $\boxed{Punc}$ $\boxed{Comp}$ *has his head* $\boxed{Punc}$ *dwells* $\boxed{P}$ $\boxed{NP}$ *.*

Table 7.15: Testing SMARTSWAP on an individual constructed using smart operators

7.3.7 Comparison with PROTECTOR and SPUD

As outlined in Section 3.4.1, PROTECTOR is a tactical NLG system that generates sentences which approximately convey a set of semantics. The generation process is divided into three stages, i.e. building a skeletal structure, covering the remaining semantics, and completing a derivation. All three stages involve similar computation, i.e. given an input semantics, $InitialGraph$, and a partially built semantic-syntactic structure, $Partial$, the greedy algorithm in PROTECTOR tries to find a **mapping rule** which

1. covers a portion of $InitialGraph$, and
2. is syntactically and semantically applicable to $Partial$.

These mapping rules are added to $Partial$, and at any stage of processing it represents the current semantic-syntactic structure already built.

At this point we can already see the similarities with how our SMARTADD works. Within the context of our operators, $InitialGraph$ corresponds to $S_{unrealized}$, and $Partial$ corresponds to the derived tree obtained from an individual's derivation tree, T. Furthermore, the selection of mapping rules corresponds to the selection of a valid 4-tuple $< d, n, t_g, l >$. Although we do not use explicit mapping rules like PROTECTOR, they are implicitly represented through the combination of feature structures, semantics, and signatures of lexical items and elementary trees in the grammar and lexicon.

The analogue to the requirement in PROTECTOR that a mapping rule must cover a portion of *InitialGraph* is the fact that SMARTADD will only consider lexical items whose semantics have a non-empty mapping to the target semantics (Section 7.3.3).

The analogue to the second requirement, i.e. that the chosen mapping rule must be syntactically and semantically applicable to *Partial*, is the validation of unification of feature structures (Section 7.2.1) and semantic signatures (Section 7.3.4) checked by the various `Filter` functions.

The main difference is that PROTECTOR's mapping rules can represent more complex applicability semantics. For example, these semantics can be linked to internal generation rules, which trigger recursive generation, e.g. for the complements of a verb. This way, one could theoretically devise arbitrarily complex mapping rules that represent large spans of texts (although PROTECTOR seems to stay within the boundaries of a sentence). In contrast, the scope of knowledge specified for our SMARTADD is limited to a single elementary tree and the lexical item that anchors it. In a sense, our "window" on the input semantics is much narrower than PROTECTOR's. However, given that the extended domain of locality of LTAG elementary trees allows the specification of all complements, this is still a sufficiently large window.

In this respect, the SMARTADD operator is more similar to SPUD, which also incrementally constructs a derivation by adding an elementary tree at each step. The SPUD generation system as described in Section 3.4.1 is also very similar to PROTECTOR and our smart operator. The algorithm used in SPUD, taken from Stone and Doran (1997) is shown as the pseudocode in Algorithm 7.8.

Algorithm 7.8 Pseudocode for greedy algorithm of SPUD (Stone and Doran, 1997)

```
while there are still unsatisfied goals do
  determine which uninflected forms apply
  determine which associated trees apply
  evaluate progress towards goals
  incorporate most specific, best <form, tree>:
    perform adjunction or substitution
    conjoin new semantics
    add any additional goals
end while
```

Note that Algorithm 7.8 implements a search algorithm that is very similar to our stochastic depth-first search shown in Algorithm 7.1 to 7.4. The main difference is that where ours is a stochastic search that selects the first valid solution found (i.e. a valid 4-tuple $< d, n, t_g, l >$), SPUD performs a systematic and complete search of the neighbourhood and selects the best solution. Furthermore, SPUD dynamically updates its goals, as it is designed to handle pragmatic goals which change throughout the generation process.

Note that SMARTADD is only a **local** search, and is used as a component of the EA search that (hopefully) achieves the global optimum of realizing the target semantics (hence the need for the nonmonotonic SMARTDELETE and SMARTSWAP operators), whereas PROTECTOR and SPUD are globally greedy/hillclimbing search algorithms. In PROTECTOR, this is implemented through the three different processing stages, i.e. it first builds a skeletal structure using substitution operations before covering the remaining semantics with adjunction. Finally, it syntactically completes a derivation.

One idea of approximating this greediness is by forcing the operators to address these goals in a "cascading" fashion. We will now examine this approach in Section 7.4.

7.4 Compound operators

By testing our baseline and smart operators in Sections 7.2.4 and 7.3.6, we have seen that each operator behaves in a slightly different fashion, and is particularly useful for different conditions. For instance, the SMARTADDMATCH operators are the most accurate operators for conveying S_{target}, but without the opportunities presented by existing linguistic structure, they are unable to contribute anything, i.e. they end up in a 'dead-end' as in Table 7.10. Furthermore, all the smart operators are unable to yield complete derivations, as their requirement of having to anchor trees with words that convey a subset of $S_{unrealized}$ rules out non-semantic operations such as the substitution of determiners and complementizers, and the adjunction of sentence frames.

In striving to show the behaviour of these operators, we ended up using them cooperatively, for example by first using SMARTADDSUBSTITUTE to build an initial structure before 'fleshing out' the derivation with SMARTADDMATCH operators (Table 7.12), or by 'closing' a derivation built by smart operators by repeatedly applying BLINDADDSUBSTITUTE (Table 7.14).

This is reminiscent of the globally greedy search found in the three stages of PROTECTOR. Recall from Section 7.3.7 that the difference between our smart operators and PROTECTOR (and SPUD) is that SMARTADD is a local search operator within an EA, and PROTECTOR and SPUD are globally greedy/hillclimbing search algorithms. The main issue here is that our operators are limited to the addition of one elementary tree to an individual's derivation tree, thus limiting the linguistic and semantic scope within which they can exploit domain knowledge. In this section we will consider moving beyond this limitation through a composition of operators, i.e. by building **compound operators** that use the operators we have described in Section 7.2 and 7.3 as building blocks.

We will use this approach in three different ways. Firstly, to merely ensure complete derivations (Section 7.4.1); secondly, to ensure complete derivations while greedily attempting to consume the target semantics (Section 7.4.2); and lastly, to simulate the PROTECTOR and SPUD systems (Section 7.4.3).

7.4.1 Ensuring complete derivations

In Section 4.5.1 we proposed two approaches to our incremental structure-building operations: incomplete and complete derivation building, each with their relative merits and drawbacks. The operators we have presented thus far in Sections 7.2 and 7.3 can yield incomplete derivations. We can use these operators in cohort to ensure that only complete derivations are considered as candidate solutions. Recall from Section 7.2.1 that the search performed by BLINDADD for a valid 4-tuple $< d, n, t_g, l >$, although stochastic, is complete, i.e. it will eventually find a valid 4-tuple if one exists. Recall also that in our grammar we do not use obligatory adjunction, i.e. substitution nodes are the only features of an incomplete derivation. Thus, by repeatedly performing BLINDADDSUBSTITUTE on an individual until no more valid 4-tuples are found, we can guarantee a derivation will be complete. This is similar to the last stage of PROTECTOR, the completing of a derivation.

We treat all our baseline and smart operators as boolean functions that, whilst achieving the side-effect of implementing an operation, return `true` if they succeed in finding and implementing a valid operation, and `false` otherwise.

Thus we can define an auxiliary operator DERIVATIONCLOSER, that takes an individual and repeatedly applies BLINDADDSUBSTITUTE to that individual until no longer possible. At this point, its derivation is guaranteed to be complete. The algorithm for this operator is given in Algorithm 7.9.

Algorithm 7.9 DERIVATIONCLOSER(`Individual` *I*)

```
incomplete ← true
while incomplete do
  incomplete ← BLINDADDSUBSTITUTE(I)
end while
```

Using this auxiliary operator, we define three compound operators that ensure complete derivations, BLINDCREATECOMPLETE, BLINDADJOINCOMPLETE and BLINDDELETECOMPLETE. The algorithms for these operators are shown in Algorithms 7.10 to 7.12.

Algorithm 7.10 BLINDCREATECOMPLETE(`Individual` *I*)

```
INITIALIZE(I)
DERIVATIONCLOSER(I)
return true
```

Algorithm 7.11 BLINDADJOINCOMPLETE(`Individual` *I*)

```
if BLINDADDADJOIN(I) = false then
  return false
end if
DERIVATIONCLOSER(I)
return true
```

None of these operators are semantically motivated, as they are based on the baseline operators defined in Section 7.2. They all work in a similar fashion, i.e. by first applying one primary operator, and then applying DERIVATIONCLOSER. Note that within this regime, there is no longer a need for a compound operator that handles substitution, as all individuals are guaranteed to be complete. Thus, BLINDCREATECOMPLETE actually creates a completely new individual by re-initializing an individual (the function INITIALIZE resets a derivation tree to a single node representing the ‘kickoff’ tree in Figure 7.8).

Note that there is no need for a dedicated compound operator that ensures subtree swapping of complete derivations: since all individuals are already complete derivations, the result of BLINDSWAP is guaranteed to be complete as well.

We now present some results of testing these compound operators. Table 7.16 shows five

Algorithm 7.12 BlindDeleteComplete(Individual I)

```
if BlindDelete(I) = false then
    return false
end if
DerivationCloser(I)
return true
```

he dwells with his names .
his blubber has the epithets .
it dwells in its expense .
a whale dwells with the whale .
its mothers will be with him .

Table 7.16: Testing of BlindCreateComplete

instances of individuals that have been created by BlindCreateComplete. They are all complete single sentences, which is to be expected given the kickoff-tree.

Table 7.17 shows three instances of applying BlindAdjoinComplete five times to an individual yielded by BlindCreateComplete. Again we can see that at the beginning they are all complete single sentences. The adjunctions that are performed vary from simple adjectives (e.g.*'grim'*), relative clauses (e.g.*", that will be slimy,"*), prepositional phrases (e.g.*"in his tiger,"*), to complete sentences (e.g.*"his mothers play."*).

Finally, in Table 7.18 we show three instances of applying BlindDeleteComplete to a complete individual that has been created using both BlindCreateComplete and BlindAdjoinComplete. Note that when the subtree deleted is rooted by an adjunction operation, DerivationCloser does not have any effect. This can be seen in the first instance, when the adjective phrase *"very shocking"* is deleted. In the second and third instances, DerivationCloser ensures that all substitution nodes created as a result of the deletion are filled.

7.4.2 Ensuring complete derivations with greedy semantic consumption

The compound operators defined in the previous section succeed in creating individuals that are guaranteed to be complete derivations. However, they are not semantically motivated. In this section we present compound operators that make use of the smart operators defined in Section 7.3 to yield complete derivations that are semantically motivated.

its shoulders boil its fish . *its shoulders boil its grim fish .* *its shoulders boil its grim fish with it .* *its shoulders boil its grim fish with it in families .* *in his tiger , its shoulders boil its grim fish with it in families .* *in his tiger , its shoulders will boil its grim fish with it in families .*
a soil is him . *a soil , that will be slimy , is him .* *a soil , that will not be slimy , is him .* *a soil , that will not not be slimy , is him .* *a soil , that will not not be very slimy , is him .* *a soil , that with its trouble , will not not be very slimy , is him .*
its epithets are tender . *its epithets are tender . his mothers play .* *its epithets are tender . with it , his mothers play .* *in his whiskers , its epithets are tender . with it , his mothers play .* *in his whiskers , its epithets are tender . in the men , with it , his mothers play .* *with it , in his whiskers , its epithets are tender . in the men , with it , his mothers play .*

Table 7.17: Testing of BlindAdjoinComplete

Before *with his names , a very shocking treatment is with its elephant .* **After** *with his names , a treatment is with its elephant .*
Before *a animal is large . his men are very big . the man will be african .* **After** *a animal is large . his men are very big . the skin will be african .*
Before *his expense , that is ugly with the gap , is a shocking boy .* **After** *its whale is a shocking boy .*

Table 7.18: Testing of BlindDeleteComplete

As revealed by our testing in Section 7.3.6, these smart operators are by definition incapable of yielding complete derivations, even when applied indefinitely. Thus our smart compound operators will have to rely on baseline operators to ensure that a derivation is completed. They implement a greedy mechanism similar to SPUD and PROTECTOR in that at any point they will always try to apply the most semantically principled operator. The order of precedence is as follows:

$$\text{SmartAddMatch} > \text{SmartAdd} > \text{BlindAdd}$$
$$\text{SmartDelete} > \text{BlindDelete}$$

The DerivationCompleter operator is replaced by the GreedySmartMatch, GreedySmart, and GreedyBlind operators, shown in Algorithms 7.13 to 7.15, that implement increasingly semantically unprincipled operators. Note that GreedySmart calls GreedySmartMatch, and similarly, GreedyBlind calls GreedySmartMatch and GreedySmart. This is to ensure that the operator with the highest order of semantic precedence is always applied. In most cases, and certainly this is true of our simple grammar, these external function calls are unnecessary. However, we include them here for completeness sake.

Algorithm 7.13 GreedySmartMatch(Individual I)

```
possibleSmartMatch ← true
while possibleSmartMatch do
    possibleSmartMatch ← SmartAddMatchSubstitute(I)
end while
```

Algorithm 7.14 GreedySmart(Individual I)

```
possibleSmart ← true
while possibleSmart do
    possibleSmart ← SmartAddSubstitute(I)
    GreedySmartMatch(I)
end while
```

Using these auxiliary operators, we define our three smart compound operators, SmartCreateComplete, SmartAdjoinComplete, and SmartDeleteComplete, which are shown in Algorithms 7.16 to 7.18. They behave very similarly to their ‘blind’ counterparts in the previous section, i.e. they first apply one primary operator before attempting to complete the derivation using GreedySmartMatch, GreedySmart, and GreedyBlind. The primary

Algorithm 7.15 GREEDYBLIND(`Individual` I)

```
possibleBlind ← true
while possibleBlind do
    possibleBlind ← BLINDADDSUBSTITUTE)(I)
    GREEDYSMARTMATCH(I)
    GREEDYSMART(I)
end while
```

operator being applied also follows the order of semantic precedence defined above. Note that in SMARTCREATECOMPLETE we must first apply either SMARTADDSUBSTITUTE or BLINDADDSUBSTITUTE once so that GREEDYSMARTMATCH will have some existing linguistic content to match with.

Algorithm 7.16 SMARTCREATECOMPLETE(`Individual` I)

```
INITIALIZE(I)
if SMARTADDSUBSTITUTE(I) = false then
    if BLINDADDSUBSTITUTE(I) = false then
        return false
    end if
end if
GREEDYSMARTMATCH(I)
GREEDYSMART(I)
GREEDYBLIND(I)
return true
```

We now present some results of testing these semantically motivated compound operators. As in Section 7.3.6, we use the first two lines of *The Lion* as our target semantics (Figure 7.11). Table 7.19 shows five instances of individuals that have been modified by SMARTCREATECOMPLETE. They all manage to convey a subset of S_{target}. Note that genitive determiners, e.g.*'his'*, *'its'*, convey the lexical semantics $own(Owning, Owner, Ownee)$, and thus their appearance in these texts may be as a result of smart operators that are explicitly conveying S_{target}, e.g.*"his waist"*, or as a result of purely syntactic baseline operators, e.g.*"its waste"*.

In Table 7.20 we show three instances of applying SMARTADJOINCOMPLETE five times to an individual yielded by SMARTCREATECOMPLETE. What these results show is that the initial linguistic content to which adjunction is gradually applied to is crucial in determining the

Algorithm 7.17 SmartAdjoinComplete(`Individual` I)

```
if SmartAddMatchAdjoin(I) = false then
  if SmartAddAdjoin(I) = false then
    if BlindAddAdjoin(I) = false then
      return false
    end if
  end if
end if
GreedySmartMatch(I)
GreedySmart(I)
GreedyBlind(I)
return true
```

Algorithm 7.18 SmartDeleteComplete(`Individual` I)

```
if SmartDelete(I) = false then
  if BlindDelete(I) = false then
    return false
  end if
end if
GreedySmartMatch(I)
GreedySmart(I)
GreedyBlind(I)
return true
```

his waist will be his waist .
the lion has the head .
a lion dwells in its waste .
his waist will be small .
the lion has the head .

Table 7.19: Testing of SmartCreateComplete

his waist is small . *his waist , that will be his waist , is small .* *his waist , that will be his waist , is very small .* *his waist , that will be his small waist , is very small .* *his small waist , that will be his small waist , is very small .* *his small waist , that will be his small waist in a waste , is very small .*
the lion dwells in the waste . *the lion , who is african , dwells in the waste .* *the lion , who is very african , dwells in the waste .* *the bandy lion , who is very african , dwells in the waste .* *the bandy lion , who in his waist , is very african , dwells in the waste .* *the bandy lion , who in his waist , that will be small , , is very african , dwells in the waste .*
a lion has its head . *a lion has its big head .* *a lion , who is african , has its big head .* *in the waste , a lion , who is african , has its big head .* *in the waste , a lion , who is very african , has its big head .* *in the waste , a lion , who is very african , has its very big head .*

Table 7.20: Testing of SmartAdjoinComplete

quality of the eventual text. In the first instance, the sentence *"his waist is small."* does not give SmartAdjoinComplete much opportunity to realize the semantics. Contrast this with the other two instances where *'lion'* is included at the beginning. These instances can be seen to support the reasoning behind the search for a mapping rule during PROTECTOR's first stage, where it tries to find an applicable mapping rule that covers as much as possible of the *InitialGraph*.

Finally, in Table 7.21 we show two instances of SmartDeleteComplete, where it chooses to delete semantically insignificant phrases, e.g.*'is'*, *'the'*.

In these test results, there are relatively few instances of the baseline operators being applied. One such example is the adjunction of the adjective *'bandy'* in the second instance in Table 7.20. This is most likely due to the input semantics being fairly rich, thus there is generally enough opportunity for the smarter operators to find something semantically motivated to add, especially given that we have limited this test to only five applications of the respective operators.

Before *its african head is its african lion , who is african , .* **After** *its african head is its african lion , who will be african , .*
Before *in the waste , the grin is its very big head .* **After** *in a waste , the grin is its very big head .*

Table 7.21: Testing of SmartDeleteComplete

7.4.3 Simulating PROTECTOR and SPUD

In this section we present our attempts at simulating the PROTECTOR and SPUD systems in McGonagall as compound operators that use the baseline, smart, and compound operators described thus far. The goal is to create operators that almost entirely deterministically convey the target semantics, S_{target}. Although this exercise is not central to the goal of this thesis, we feel it is of significant value as it shows the potential and flexibility of this system.

In Section 7.4.2 we presented compound operators that ensure complete derivations while greedily consuming the target semantics. The design of these operators was necessarily a trade-off between attempting to strictly convey semantics and closing off a derivation. Furthermore, in designing these operators, we limited ourselves by only using the baseline and smart operators described in Section 7.2 and 7.3. In this section we explicitly try to simulate the PROTECTOR and SPUD systems, and this includes developing extra baseline operators, namely SyntaxAddSubstitute and SyntaxAddAdjoin.

PROTECTORLike is a compound operator that embodies the entire generation process in PROTECTOR that consists of three stages: building a skeletal structure, covering the remaining semantics, and completing the derivation. The resulting individual is hoped to be an optimal solution. The algorithm of this operator is shown in Algorithm 7.19.

Assuming the individual I is newly initialized, it first applies SmartAddSubstitute to create a minimal linguistic structure which SmartAddMatchSubstitute, through GreedySmartMatch, expands upon. At the end of this stage the individual should consist of a complete sentence with all its complements realized. It then repeatedly calls SmartAddMatchAdjoin to convey as much additional semantics as possible. Finally, the derivation is closed off using the SYNTAXADDSUBSTITUTE operator. Essentially it is the complement of BLINDADDSUBSTITUTE and SMARTADDSUBSTITUTE: it only considers lexical items whose $S_{lexical} = \emptyset$, e.g. determiners, punctuation marks, complementizers, and copula verbs.

However, there is a fundamental difference between our PROTECTORLIKE operator and Nicolov's actual PROTECTOR system: although PROTECTORLIKE attempts to emulate the three-stage process of PROTECTOR, i.e. building a skeletal structure, covering the remaining semantics, and completing the derivation, each individual stage is implemented differently. In particular, PROTECTOR systematically searches for the optimal local decision to be made during each stage, whereas our operator stochastically hillclimbs.

Algorithm 7.19 PROTECTORLIKE(`Individual` I)

```
SMARTADDSUBSTITUTE($I$) {Stage 1: Building a skeletal structure}
GREEDYSMARTMATCH($I$)
$possible \leftarrow$ true
while $possible$ do
  $possible \leftarrow$ SMARTADDMATCHADJOIN($I$) {Stage 2: Cover remaining semantics}
  GREEDYSMARTMATCH($I$)
end while
$possible \leftarrow$ true
while $possible$ do
  $possible \leftarrow$ SYNTAXADDSUBSTITUTE($I$) {Stage 3: Completing the derivation}
end while
```

SPUDLIKE is a compound operator that emulates the greedy algorithm of SPUD which always implements the **best** operation possible, as opposed to the **first** valid operation found, as is implemented by the stochastic depth-search of our operators. The precedence of operators is as follows:

$$\text{SMARTADDMATCHSUBSTITUTE} > \text{SMARTADDMATCHADJOIN} >$$
$$\text{SMARTADDSUBSTITUTE} > \text{SYNTAXADDSUBSTITUTE}$$

It is different from PROTECTORLIKE in that it does not embody the entire generation process in a single operator. Rather, it only performs one LTAG operation, and it relies on the iterative nature of the EA to execute the greedy search. As long as it finds and implements a valid operation, it will return `true`, and `false` otherwise. An EA with a population size of one individual and SPUDLIKE as the only available genetic operator behaves remarkably similar to SPUD. The algorithm for this operator is shown in Algorithm 7.20.

Algorithm 7.20 SPUDLIKE(Individual I)

```
if SMARTADDMATCHSUBSTITUTE($I$) = true then
    return true
else if SMARTADDMATCHADJOIN($I$) = true then
    return true
else if SMARTADDSUBSTITUTE($I$) = true then
    return true
else if SYNTAXADDSUBSTITUTE($I$) = true then
    return true
else
    return false
end if
```

Given an individual initialized with the kickoff-tree in Figure 7.8, both PROTECTORLIKE and SPUDLIKE are limited to the generation of one sentence, as the only adjunction operator used is SMARTADDMATCHADJOIN. Thus, we also experiment with a slight modification of SPUDLIKE which, failing all other operations, will adjoin a new sentence frame. This is achieved through the SYNTAXADDADJOIN operator, which only considers lexical items whose $S_{lexical} = \emptyset$. In terms of anchors for auxiliary trees, the only candidates are sentence boundary punctuation marks, e.g. fullstops. We call this modified operator SPUDLIKEDISCOURSE, and the algorithm for this operator is shown in Algorithm 7.21.

We now present some results of testing these operators on newly initialized individuals. For these results, shown in Table 7.22 to 7.24, we set S_{target} as the first two lines of *The Lion* (Figure 7.11).

Table 7.22 shows five instances of individuals yielded by PROTECTORLIKE. They all accurately convey a subset of the target semantics. However, in the last instance, the pronoun *'they'* is an unfortunate choice as it is of different agreement to *'lion'*. Note that auxiliary verbs ($\boxed{Aux}$) are not selected as they carry lexical semantics, thus they are not considered by SYNTAXADDADJOIN. The reason these $\boxed{Aux}$ nodes appear in the first place is due to a substitution of a copula verb node with the I_C_AuxCV tree (see Appendix C.2.3). Thus, we can see that in all these results, the $\boxed{Aux}$ nodes always precede the copula *'be'*.

Algorithm 7.21 SPUDLikeDiscourse(Individual I)

```
if SmartAddMatchSubstitute($I$) = true then
    return true
else if SmartAddMatchAdjoin($I$) = true then
    return true
else if SmartAddSubstitute($I$) = true then
    return true
else if SyntaxAddSubstitute($I$) = true then
    return true
else if SyntaxAddAdjoin($I$) = true then
    return true
else
    return false
end if
```

the african lion , who dwells in a waste , has the head , that $\boxed{\textit{Aux}}$ *be big , .*

it is in a waste .

a african lion , who has the big head , dwells in the waste .

a lion , who dwells in a waste , $\boxed{\textit{Aux}}$ *be african .*

they $\boxed{\textit{Aux}}$ *be in the waste .*

Table 7.22: Testing of ProtectorLike

Table 7.23 shows four instances where SPUDLIKE is repeatedly applied to an initialized individual until no longer possible, i.e. it returns `false`. The quality of the final derivations are similar to those in Table 7.22. By observing the incremental generation in this table, we can clearly see the phenomenon that we first mentioned when discussing Table 7.20, i.e. that the selection of initial linguistic content is crucial in determining the quality of the eventual text. In the first and last instance, not much opportunity is presented for the operator to expand upon. Another way of viewing this is that it is simply a natural drawback of greedy/hillclimbing search, i.e. it is liable to get trapped in local maxima.

Finally, in Table 7.24 we show one instance of repeatedly applying SPUDLIKEDISCOURSE to an initialized individual. Unlike SPUDLIKE, which is limited to generating one sentence, SPUDLIKEDISCOURSE has the ability to adjoin in sentence frames, thus opening up new possibilities for subsequent smart operations. As result it manages to convey the entire S_{target}. In fact, it already does so at the point which we have highlighted, i.e. where its text is:

“*a african lion, who has a very small waist,*
has the head, that is big,. he dwells in a waste.”

From that point on, it can only perform non-semantic operations, i.e. substituting pronouns (‘*they*’, ‘*them*’), and adding new sentence frames. Unfortunately, this is the drawback of SPUDLIKEDISCOURSE: given the opportuntiy, it will never terminate. However, since it would be used as a genetic operator in an EA, typically there already exists an external termination method, such as once an optimal solution is found. Nevertheless, one can artifically limit its operation by imposing a maximum number of sentence frames that can be adjoined.

7.4.4 Ensemble of operators

In the last few sections (Section 7.4.1 to 7.4.3) we have introduced an array of compound operators which are undoubtedly more knowledge augmented than the single-operation baseline and smart operators. Table 7.25 shows a roundup of these compound operators.

It is important to note, however, that the more knowledge-rich the genetic operators become, the more constrained the EA search, and while this possibly improves the conditions for converging to a satisfying solution, we must be careful of the potential trade-off in that it may render certain solutions in the space unreachable. We have already seen in Sections 7.4.2 and 7.4.3 that the

$\boxed{S}$.
$\boxed{NP}$ $\boxed{CV}$ in $\boxed{NP}$.
$\boxed{NP}$ $\boxed{CV}$ in $\boxed{D}$ waste .
$\boxed{NP}$ is in $\boxed{D}$ waste .
$\boxed{NP}$ is in a waste .
$\boxed{D}$ it is in a waste .
it is in a waste .
$\boxed{S}$.
$\boxed{NP}$ dwells $\boxed{P}$ $\boxed{NP}$.
$\boxed{D}$ lion dwells $\boxed{P}$ $\boxed{NP}$.
$\boxed{D}$ lion dwells in $\boxed{NP}$.
$\boxed{D}$ lion dwells in $\boxed{D}$ waste .
$\boxed{D}$ lion $\boxed{Punc}$ $\boxed{Comp}$ has $\boxed{NP}$ $\boxed{Punc}$ dwells in $\boxed{D}$ waste .
$\boxed{D}$ lion $\boxed{Punc}$ $\boxed{Comp}$ has $\boxed{D}$ head $\boxed{Punc}$ dwells in $\boxed{D}$ waste .
$\boxed{D}$ african lion $\boxed{Punc}$ $\boxed{Comp}$ has $\boxed{D}$ head $\boxed{Punc}$ dwells in $\boxed{D}$ waste .
$\boxed{D}$ african lion $\boxed{Punc}$ $\boxed{Comp}$ has $\boxed{D}$ big head $\boxed{Punc}$ dwells in $\boxed{D}$ waste .
$\boxed{D}$ african lion $\boxed{Punc}$ $\boxed{Comp}$ has a big head $\boxed{Punc}$ dwells in $\boxed{D}$ waste .
a african lion $\boxed{Punc}$ $\boxed{Comp}$ has a big head $\boxed{Punc}$ dwells in $\boxed{D}$ waste .
a african lion $\boxed{Punc}$ $\boxed{Comp}$ has a big head $\boxed{Punc}$ dwells in the waste .
a african lion $\boxed{Punc}$ $\boxed{Comp}$ has a big head , dwells in the waste .
a african lion $\boxed{Punc}$ who has a big head , dwells in the waste .
a african lion , who has a big head , dwells in the waste .
$\boxed{S}$.
$\boxed{NP}$ has $\boxed{NP}$.
$\boxed{NP}$ has $\boxed{D}$ head .
$\boxed{D}$ lion has $\boxed{D}$ head .
$\boxed{D}$ lion has $\boxed{D}$ big head .
$\boxed{D}$ lion $\boxed{Punc}$ $\boxed{Comp}$ $\boxed{CV}$ african $\boxed{Punc}$ has $\boxed{D}$ big head .
a lion $\boxed{Punc}$ $\boxed{Comp}$ $\boxed{CV}$ african $\boxed{Punc}$ has $\boxed{D}$ big head .
a lion $\boxed{Punc}$ $\boxed{Comp}$ $\boxed{CV}$ african $\boxed{Punc}$ has the big head .
a lion $\boxed{Punc}$ who $\boxed{CV}$ african $\boxed{Punc}$ has the big head .
a lion , who $\boxed{CV}$ african $\boxed{Punc}$ has the big head .
a lion , who $\boxed{CV}$ african , has the big head .
a lion , who $\boxed{Aux}$ be african , has the big head .
$\boxed{S}$.
$\boxed{NP}$ $\boxed{CV}$ big .
$\boxed{D}$ head $\boxed{CV}$ big .
his head $\boxed{CV}$ big .
his head $\boxed{Aux}$ be big .

Table 7.23: Testing of SPUDLIKE

$\boxed{\textit{S}}$.
$\boxed{\textit{NP}}$ *has* $\boxed{\textit{NP}}$.
$\boxed{\textit{D}}$ *lion has* $\boxed{\textit{NP}}$.
$\boxed{\textit{D}}$ *lion has* $\boxed{\textit{D}}$ *head* .
$\boxed{\textit{D}}$ *african lion has* $\boxed{\textit{D}}$ *head* .
$\boxed{\textit{D}}$ *african lion has* $\boxed{\textit{D}}$ *head* $\boxed{\textit{Punc}}$ $\boxed{\textit{Comp}}$ $\boxed{\textit{CV}}$ *big* $\boxed{\textit{Punc}}$.
$\boxed{\textit{D}}$ *african lion* $\boxed{\textit{Punc}}$ $\boxed{\textit{Comp}}$ *has* $\boxed{\textit{NP}}$ $\boxed{\textit{Punc}}$ *has* $\boxed{\textit{D}}$ *head* $\boxed{\textit{Punc}}$ $\boxed{\textit{Comp}}$ $\boxed{\textit{CV}}$ *big* $\boxed{\textit{Punc}}$.
$\boxed{\textit{D}}$ *african lion* $\boxed{\textit{Punc}}$ $\boxed{\textit{Comp}}$ *has* $\boxed{\textit{D}}$ *waist* $\boxed{\textit{Punc}}$ *has* $\boxed{\textit{D}}$ *head* $\boxed{\textit{Punc}}$ $\boxed{\textit{Comp}}$ $\boxed{\textit{CV}}$ *big* $\boxed{\textit{Punc}}$.
$\boxed{\textit{D}}$ *african lion* $\boxed{\textit{Punc}}$ $\boxed{\textit{Comp}}$ *has* $\boxed{\textit{D}}$ *small waist* $\boxed{\textit{Punc}}$ *has* $\boxed{\textit{D}}$ *head* $\boxed{\textit{Punc}}$ $\boxed{\textit{Comp}}$ $\boxed{\textit{CV}}$ *big* $\boxed{\textit{Punc}}$.
$\boxed{\textit{D}}$ *african lion* $\boxed{\textit{Punc}}$ $\boxed{\textit{Comp}}$ *has* $\boxed{\textit{D}}$ *very small waist* $\boxed{\textit{Punc}}$ *has* $\boxed{\textit{D}}$ *head* $\boxed{\textit{Punc}}$ $\boxed{\textit{Comp}}$ $\boxed{\textit{CV}}$ *big* $\boxed{\textit{Punc}}$.
$\boxed{\textit{D}}$ *african lion* $\boxed{\textit{Punc}}$ $\boxed{\textit{Comp}}$ *has* $\boxed{\textit{D}}$ *very small waist , has* $\boxed{\textit{D}}$ *head* $\boxed{\textit{Punc}}$ $\boxed{\textit{Comp}}$ $\boxed{\textit{CV}}$ *big* $\boxed{\textit{Punc}}$.
a african lion $\boxed{\textit{Punc}}$ $\boxed{\textit{Comp}}$ *has* $\boxed{\textit{D}}$ *very small waist , has* $\boxed{\textit{D}}$ *head* $\boxed{\textit{Punc}}$ $\boxed{\textit{Comp}}$ $\boxed{\textit{CV}}$ *big* $\boxed{\textit{Punc}}$.
a african lion $\boxed{\textit{Punc}}$ $\boxed{\textit{Comp}}$ *has a very small waist , has* $\boxed{\textit{D}}$ *head* $\boxed{\textit{Punc}}$ $\boxed{\textit{Comp}}$ $\boxed{\textit{CV}}$ *big* $\boxed{\textit{Punc}}$.
a african lion , $\boxed{\textit{Comp}}$ *has a very small waist , has* $\boxed{\textit{D}}$ *head* $\boxed{\textit{Punc}}$ $\boxed{\textit{Comp}}$ $\boxed{\textit{CV}}$ *big* $\boxed{\textit{Punc}}$.
a african lion , who has a very small waist , has $\boxed{\textit{D}}$ *head* $\boxed{\textit{Punc}}$ $\boxed{\textit{Comp}}$ $\boxed{\textit{CV}}$ *big* $\boxed{\textit{Punc}}$.
a african lion , who has a very small waist , has $\boxed{\textit{D}}$ *head* $\boxed{\textit{Punc}}$ $\boxed{\textit{Comp}}$ *is big* $\boxed{\textit{Punc}}$.
a african lion , who has a very small waist , has $\boxed{\textit{D}}$ *head* $\boxed{\textit{Punc}}$ *that is big* $\boxed{\textit{Punc}}$.
a african lion , who has a very small waist , has the head $\boxed{\textit{Punc}}$ *that is big* $\boxed{\textit{Punc}}$.
a african lion , who has a very small waist , has the head $\boxed{\textit{Punc}}$ *that is big , .*
a african lion , who has a very small waist , has the head , that is big , .
a african lion , who has a very small waist , has the head , that is big , . $\boxed{\textit{S}}$.
a african lion , who has a very small waist , has the head , that is big , . $\boxed{\textit{NP}}$ *dwells* $\boxed{\textit{P}}$ $\boxed{\textit{NP}}$.
a african lion , who has a very small waist , has the head , that is big , . $\boxed{\textit{NP}}$ *dwells in* $\boxed{\textit{NP}}$.
a african lion , who has a very small waist , has the head , that is big , . $\boxed{\textit{NP}}$ *dwells in* $\boxed{\textit{D}}$ *waste .*
a african lion , who has a very small waist , has the head , that is big , . $\boxed{\textit{NP}}$ *dwells in a waste .*
a african lion , who has a very small waist , has the head , that is big , . $\boxed{\textit{D}}$ *it dwells in a waste .*
$\boxed{\textit{a african lion , who has a very small waist , has the head , that is big , . it dwells in a waste .}}$
a african lion , who has a very small waist , has the head , that is big , . it dwells in a waste . $\boxed{\textit{S}}$.
a african lion , who has a very small waist , has the head , that is big , . it dwells in a waste . $\boxed{\textit{NP}}$ $\boxed{\textit{CV}}$ $\boxed{\textit{D}}$ *them .*
a african lion , who has a very small waist , has the head , that is big , . it dwells in a waste . $\boxed{\textit{D}}$ *they* $\boxed{\textit{CV}}$ $\boxed{\textit{D}}$ *them .*
a african lion , who has a very small waist , has the head , that is big , . it dwells in a waste . they $\boxed{\textit{CV}}$ $\boxed{\textit{D}}$ *them .*
a african lion , who has a very small waist , has the head , that is big , . it dwells in a waste . they $\boxed{\textit{Aux}}$ *be* $\boxed{\textit{D}}$ *them .*
a african lion , who has a very small waist , has the head , that is big , . it dwells in a waste . they $\boxed{\textit{Aux}}$ *be them .*
$\boxed{\textit{S}}$ *. a african lion , who has a very small waist , has the head , that is big , . it dwells in a waste . they* $\boxed{\textit{Aux}}$ *be them .*

Table 7.24: Testing of SPUDLIKEDISCOURSE

highly elaborate semantically greedy compound operators are particularly sensitive to the initial selection of linguistic content.

One simple way to overcome this is to use these knowledge augmented operators in conjunction with the simple baseline operators, and rely on the stochastic process of the EA to find a solution.

In a multi-objective optimization problem such as poetry generation, one can develop separate operators that each embody domain knowledge for optimizing separate objectives. For example, operators that uses domain knowledge to satisfy target semantics (as defined in Section 7.3), one for satisfying the target metre, one for the rhyme, and so forth. By using these operators in conjunction in an EA, we conveniently sidestep the difficult issue of how best to order control over these operations in a deterministic algorithm (Section 3.2).

7.5 Summary

In this chapter we first presented two distinct flavours of genetic operators: baseline operators that ensure grammaticality and smart operators that explicitly attempt to achieve meaningfulness. Each operator behaves in a slightly different fashion, and is particularly useful for different conditions. By using them in cooperation through our compound operators, we can extend the linguistic and semantic scope within which they can exploit domain knowledge. This enables us to create operators that, for instance, ensure complete derivations, and almost deterministically convey the target semantics.

Note that the compound operators, especially the semantically-motivated ones, resemble the more conventional NLG methods seen in Chapter 3. However, we believe that our characterisation of NLG as a search problem (Section 4.1) still holds. The wide array of operators we have introduced, starting from the baseline operators (Section 7.2), the smart operators (Section 7.3), the compound operators that ensure complete derivations (Section 7.4.1), and finally the PROTECTOR-like and SPUD-like operators (Section 7.4.3), signify different points along a spectrum of search sophistication between strictly goal-directed planning (Section 3.2) and our view of the entire NLG process as explicit state space search (Section 4.1). However, all these operators still fit within the view of NLG as state space search, as even the most sophisticated operators are still taking a state, i.e. individual, and randomly jumping to a different state, i.e. through mutation or crossover.

We also propose using these various operators as an ensemble (Section 7.4.4), where we can develop separate operators that each embody domain knowledge for optimizing separate objectives. By using these operators in conjunction in an EA, we also conveniently sidestep the issue of how to control these operations in a deterministic algorithm (Section 3.2).

At this point we have now described all the main components of our EA: representations, evaluation functions, and genetic operators. In the next chapter we will present the results of the empirical studies we conducted using McGonagall.

Name	Description
Ensuring complete derivations (Section 7.4.1)	
DerivationCloser	Repeatedly applies BlindAddSubstitute until a derivation is complete.
BlindCreateComplete	Re-initializes an individual and then applies DerivationCloser.
BlindAdjoinComplete	Applies BlindAddAdjoin if possible and then applies DerivationCloser.
BlindDeleteComplete	Applies BlindDelete if possible and then applies DerivationCloser.
Semantically-motivated complete derivations (Section 7.4.2)	
GreedySmartMatch	Repeatedly applies SmartAddMatchSubstitute.
GreedySmart	Repeatedly applies SmartAddSubstitute and GreedySmartMatch.
GreedyBlind	Repeatedly applies BlindAddSubstitute, GreedySmartMatch, and GreedySmart.
SmartDerivationCloser	Applies GreedySmartMatch, GreedySmart, and GreedyBlind, ensuring a derivation is complete.
SmartCreateComplete	Re-initializes an individual and then applies SmartDerivationCloser.
SmartAdjoinComplete	Applies smartest adjoin operator, i.e.SmartAddMatchAdjoin, SmartAddAdjoin, or BlindAddAdjoin if possible and then applies SmartDerivationCloser.
SmartDeleteComplete	Applies smartest deletion operator, i.e.SmartDelete or BlindDelete if possible and then applies SmartDerivationCloser.
Simulating PROTECTOR and SPUD (Section 7.4.3)	
PROTECTORLike	Emulates the three stages of PROTECTOR: building a skeletal structure, covering remaining semantics, and completing the derivation.
SPUDLike	Emulates the greedy approach of SPUD by always prioritizing the best operator, i.e. the most semantically-motivated one.

Table 7.25: Roundup of compound operators

Chapter 8

Empirical Study and Discussion

In this chapter we report on an empirical study that we conducted using McGonagall, our implemented system that embodies the EA-based approach we have presented in previous chapters. We first present the objective of the study, and based on this, the methodology and design used for the study. We then present the results of the three stages of the study, namely testing McGonagall with form constraints only, semantic constraints only, and both form and semantic constraints simultaneously. For each stage we provide a summary of the observations and discussions for the tests carried out.

8.1 Objective of the study

There are two distinct objectives that motivate our empirical study. The first is to test whether our approach of employing an EA search method to solve the constraint satisfaction problem of poetry generation is valid. This relies wholly on the self-evaluation properties that an EA affords, as finding a very high scoring solution is a clear indication of the EA's success. Under this objective, we will be exploring the effect that the various possibilities of configurations have on the performance of the EA.

However, the EA's success in finding an optimal solution given a certain evaluation function does not necessarily reflect the actual quality of that solution. As Bäck et al. (1997) warns, "The objective function must reflect the relevant measures to be optimized. Evolutionary algorithms are notoriously opportunistic, and there are several known instances of an algorithm optimizing the stated objective function, only to have the user realize that the objective function did not actually represent the intended measure.”. Thus, this is the second objective of the study: to test whether the evaluation functions presented in Chapter 6 accurately capture the notions of semantic faithfulness and conformance to metre. Although the evaluators have been validated to a degree in Sections 6.3.6 (for metre) and 6.4.4 (for semantics), their actual usage in this empirical study is hoped to be a more rigorous examination of their behaviour.

In works such as Binsted (1996) and Cheng (2002), evaluation of this kind was achieved through experiments using subjective human evaluation. This was due to the fact that the features being analyzed were highly subjective, such as humour and textual coherence. We believe that our chosen aspects of semantic faithfulness and conformance to metre are relatively objective and thus concretely observable features, diminishing the need for such experimentation.

Note that the production of ‘genuine’ poetry, i.e. texts which members of the general public can observe and with a high degree of certainty state are indeed poetry, is not the goal of this study. We feel that the necessary assumptions and simplifications made within the framework of our thesis limit the possibility of this, and that research work in this area is in far too early a stage to be achieving such ambitions.

8.2 Methodology and design of the study

Owing to the nature of both our domain task and the method of problem solving through EAs, there are many unknown factors that we face from the outset. Firstly, our task domain of generating poetry has not received much serious treatment, with the exception of the work of Gervás and Levy which we discussed in Section 2.3.4. Furthermore, we have found no reports of explicit evaluation performed on these systems. Thus, it is not immediately clear whether these works can inform the design of our study. Secondly, solving problems with EAs is a highly domain-specific task which typically requires extensive empirical testing and fine-tuning to achieve success.

We divide the study into three stages. In the first stage, we test the capacity of MCGONAGALL to perform as a form-aware text generation system, i.e. where only the metre evaluation function is used. In effect, the output texts are only expected to satisfy the properties of grammaticality and poeticness ($\mathcal{G} \cap \mathcal{P}$). This stage is reported in Section 8.3. In the second stage, we test the capacity of McGonagall to perform as a tactical NLG component, i.e. where only the semantic evaluation function is used. Thus, the output texts can only be expected to satisfy the properties of grammaticality and meaningfulness ($\mathcal{G} \cap \mathcal{M}$). This stage is reported in Section 8.4. Finally, in the last stage we test the capacity of McGonagall to perform as a poetry generation system by using both the metre and semantic evaluation functions. The output texts are expected to satisfy the properties of grammaticality, meaningfulness, and poeticness ($\mathcal{G} \cap \mathcal{M} \cap \mathcal{P}$). This stage is reported in Section 8.5. We refer the reader back to our definition of poetry in Section 2.1.4 for details on these three properties.

Apart from making the task more manageable, testing these optimization tasks in isolation should provide us with a clearer picture of the behaviour of the various components, and we hope that insight revealed during testing at each stage can inform the specific design of tests for subsequent stages.

In designing the actual tests that we carry out, we adopt a principle of attempting to achieve our goals with the least bias required. More specifically, we are interested in observing the mileage that can be obtained by using the simplest components, i.e. genetic operators, EA selection algorithms and parameters, that introduce the least stochastic bias. For each of the three stages, we start out with an initial test that defines a baseline for all aspects (EA configuration, evaluation functions, genetic operators, etc.), perhaps even to the extent of naivety given what we have already discussed in previous chapters regarding these aspects. Only in the event of failing to achieve the task do we replace these aspects with more sophisticated approaches, and these replacements form the basis for subsequent tests. By adopting this principle, we will be able to see the relative contribution of each particular component.

8.3 McGonagall as form-aware generator

8.3.1 Initial form-aware test

For our initial test, we deliberately choose a simplistic configuration. The main purpose is to observe the behaviour of the EA in manipulating the available linguistic resources to optimize the metre evaluation function. To that end, we adopt baseline methods and parameters for almost all aspects. Some of these decisions can be considered naive, given what we already know about EAs and the domain task, for example the decision to ignore substitution holes and linebreaking. Nevertheless, it is useful to examine baseline performance.

The task set in this test is to generate texts that satisfy a given target form. We use three different forms: `haiku`, `limerick` and `mignonne`, as shown in Appendix A. The three forms are intended to represent the various possible form constraints that can be handled. In particular, `haiku` is only concerned with syllable count, `limerick` requires fairly complex sequences of metre patterns, and `mignonne` can be seen as emphasizing the placement of linebreaks.

However, this initial test is simplified by ignoring the issues of where linebreaking should occur and how ‘holes’ left by substitution nodes should be accounted for. The main objective is simply to see whether the EA can find a text that exhibits the desired metre pattern, or in the case of `haiku`, syllable count. In this respect, `limerick` and `mignonne` become very similar goals to achieve.

EA setup

- **Selection algorithm:** for this test, we employ one of the simplest selection algorithms, **proportionate** selection, which assigns a distribution that accords parents a probability to reproduce that is proportional to its fitness (Bäck et al., 1997). It is a well-studied algorithm that is intuitive and simple to understand and implement, which is the chief motivation for its use in this test. Individuals are sampled from this distribution using **stochastic universal sampling**, which minimises chance fluctuations in sampling (Baker, 1987).

  There are two potential problems that are known for proportionate selection algorithms: premature convergence due to a **supersolution** and stagnation due to similar fitness of individuals in the population (Bäck et al., 1997). However, one simple approach to overcome these problems is through the use of an **elitist strategy**, which we test here. An elitist strategy ensures that a portion of the fittest individuals in the population survive in subsequent iterations. We experiment with elitist populations of 20% and 40% of the entire population, and also non-elitist selection (i.e. 0%).

- **Population size:** We choose a population size of 40. The choice of this parameter is inspired by Cheng (2002), who in turn points to empirical studies by Goldberg (1989). Although the tasks are markedly different, and it is known that the optimal setting of these parameters is problem-dependent, nevertheless it is a useful starting point.

Substitution

Cost	w	s	x	b
0_1	0	2	0	∞
0_n	0	2	0	∞
1_1	3	0	0	∞
1_n	3	0	0	∞
2_n	1	1	0	∞
b	∞	∞	∞	0

Insertion

	Cost
w	1
s	3
x	1
b	0

Deletion

	Cost
0_1	1
0_n	1
1_1	3
1_n	3
2_n	2
b	0

Table 8.1: Edit distance costs for initial form-aware test

Evaluators

For this test we only use the edit distance function, $\mathcal{F}_{edit}$, defined in Section 6.3.5, that has been normalized with $\lambda_1 = 1.0$. 'Holes' left by substitution nodes are ignored, even though the function $\mathcal{F}_{balance}$ (Section 6.3.7) is designed to account for this. The cost function c is shown in Table 8.1. In particular, note that the cost incurred for inserting linebreaks is 0. The context-sensitive compensation scores and patterns in Appendix B are also used.

Operators

The baseline blind operators presented in Section 7.2 are used. The probabilities of being applied when mutation is called for are as follows:

- BlindAddSubstitute = 0.5
- BlindAddAdjoin = 0.3
- BlindDelete = 0.2

The choice of these probabilities is intended to account for the fact that the substitution operation typically introduces the essential linguistic structures, and that adjunction is for auxiliary constituents. Note that these operators can be applied on a text in any order, which is different from the conventional TAG approach where all substitutions are carried out first before adjunction is attempted, for example during the three stages of PROTECTOR.

In this test, we do not use the crossover operation, i.e. BlindSwap, as we believe that the mutation operators are the fundamental actions that create genetic content, and we are interested in examining how they perform on their own.

Note that these operators do not guarantee resulting individuals to be complete derivations, as substitution nodes are possibly not yet substituted.

Resources

The handcrafted grammar and lexicon given in Appendix C are used. It consists of 33 syntactic structures (i.e. elementary trees), and 116 lexical items, 18 of which are closed class words. The content words are compiled from a selection of Hilaire Belloc's children's poems about animals, *"The Bad Child's Book of Beasts"* (Belloc, 1991).

Expectations

- Satisfying the `haiku` target form should be easier to accomplish than `limerick` and `mignonne`, as it only involves getting the correct syllable count. In fact, given the simplifications of ignoring linebreaking penalties, any text with 17 syllables worth of lexical items will be an optimal solution.
- The selection pressure applied by the elitist strategy should prevent stagnation of the population and drive the EA to an optimal solution. We hope that for each target form there should be at least one run where the EA finds a candidate solution with a fitness score of 1.0.
- However, we expect the actual output to be of limited poetic quality due to the oversimplification of the fitness function being used.

Results

For this test, we run the EA with all the parameters as described above. The two factors being varied are the target form and the elitist ratio. Hence we run this test nine times: `haiku` with elitist ratios of 0%, 20%, and 40%, `limerick` with elitist ratios of 0%, 20%, and 40%, and `mignonne` with elitist ratios of 0%, 20%, and 40%. Each individual test is run ten times, and each run lasts for 500 iterations.

Summary statistics and graphs for this test are presented in Section D.1.1 of Appendix D.

Tables 8.2, 8.3, and 8.4 show the individuals that yield the best fitness scores from the last populations of each test. They show the best individuals for the `haiku`, `limerick`, and `mignonne` target forms respectively. For each target form, we present the best individual for each elitist ratio used, showing its fitness score, raw surface text, and formatted surface text.

The following is a guide on how to interpret the formatted surface form:

1. Syllables in normal type are substituted to `w` target syllables, i.e. receive no stress. If they are underlined, they indicate 1_1 or 1_n candidate syllables that are destressed. If they have a horizontal line directly above them, they indicate 2_n candidate syllables. Otherwise, they indicate 0_1 or 0_n candidate syllables.

2. Syllables in bold type are substituted to `s` target syllables, i.e. receive stress. If they are underlined, they indicate 0_1 or 0_n candidates. If they have a horizontal line directly above them, they indicate 2_n candidate syllables. Otherwise, they indicate 1_1 or 1_n candidate syllables.

3. Syllables enclosed in parentheses indicate candidate syllables that are deleted.

4. An asterisk (*) marks the insertion of a `w` target syllable.

5. An exclamation mark (!) marks the insertion of a `s` target syllable.

6. Substitution nodes are ignored/skipped.

7. Linebreaks are displayed according to the computed alignment, i.e. where a `b` target syllable is either inserted or substituted for a *b* candidate syllable.

8. Some further ‘prettifying’ is performed, namely using the appropriate form of the determiner ‘*a*’/‘*an*’, and changing the first letter of the first word of a sentence to uppercase.

Note that as these formatted surface texts skip substitution nodes in the raw surface, they may be ungrammatical.

Observations

The EA has no problem at all in optimizing the target (see values for the `haiku` target form in Table D.1). Although it is a stochastic process, it is virtually inevitable that the given operators will eventually yield a candidate text that contains 17 syllables worth of lexical items. Bear

	elitist 0%	elitist 20%	elitist 40%
Score	1.00	1.00	1.00
Surface	$\boxed{\text{S}}$. $\boxed{\text{NP}}$ play . in $\boxed{\text{NP}}$ $\boxed{\text{Punc}}$ it has $\boxed{\text{NP}}$ with $\boxed{\text{NP}}$. a man in $\boxed{\text{NP}}$ $\boxed{\text{CV}}$ in $\boxed{\text{NP}}$ in $\boxed{\text{NP}}$. with $\boxed{\text{NP}}$ $\boxed{\text{Punc}}$ the treatment with $\boxed{\text{NP}}$ will play .	it will be shocking in $\boxed{\text{NP}}$. a treatment is the hippopotamus . $\boxed{\text{NP}}$ $\boxed{\text{CV}}$ $\boxed{\text{D}}$ thing .	big good jaws $\boxed{\text{CV}}$ with his kind mind with her with $\boxed{\text{D}}$ hippopotamus with $\boxed{\text{NP}}$. $\boxed{\text{NP}}$ dwells $\boxed{\text{P}}$ $\boxed{\text{NP}}$. $\boxed{\text{S}}$. $\boxed{\text{S}}$. $\boxed{\text{S}}$.
Formatted	. play . In it has with . A man in in in . With the treatment with will play .	It will be shocking in . A treatment is the hippopotamus . thing .	Big good jaws with his kind mind with her with hippopotamus with . dwells . . . .

Table 8.2: Best individual for $F_{target} = \texttt{haiku}$, initial form-aware test

	elitist 0%	elitist 20%	elitist 40%
Score	0.82	0.95	0.95
Surface	men will play . with his very tender frog with a thing , the facts will be very little . she is $\boxed{\text{D}}$ soil . the mothers , they boil the baboon .	$\boxed{\text{NP}}$ $\boxed{\text{CV}}$ her . $\boxed{\text{NP}}$ dwells $\boxed{\text{P}}$ $\boxed{\text{NP}}$. in $\boxed{\text{NP}}$, the treatment $\boxed{\text{Aux}}$ be very slimy . a bill $\boxed{\text{Aux}}$ be his hippopotamus . lonely families boil his species with $\boxed{\text{D}}$ frog in a frog . $\boxed{\text{S}}$.	$\boxed{\text{NP}}$ $\boxed{\text{CV}}$ in $\boxed{\text{NP}}$. $\boxed{\text{NP}}$ has $\boxed{\text{NP}}$. $\boxed{\text{S}}$. $\boxed{\text{S}}$. in $\boxed{\text{NP}}$ $\boxed{\text{Punc}}$ with $\boxed{\text{NP}}$ $\boxed{\text{Punc}}$ $\boxed{\text{D}}$ sensitive trunk with african african platinum whiskers with $\boxed{\text{D}}$ little trouble $\boxed{\text{Aux}}$ be very sensitive . $\boxed{\text{NP}}$ $\boxed{\text{CV}}$ $\boxed{\text{D}}$ child with a gap . $\boxed{\text{S}}$.
Formatted	* **men** * will **play** . With his **ve**- ry **ten** * der **frog** with a **thing** , the **facts** will be **ve**- ry **lit**(tle) . She is **soil** . the **mo**thers , they **boil** the ba**boon** .	her . **dwells** . in , the **treat**ment be **ve**- ry **sli**my . A **bill** be his **hip**- po**po**tamus . **Lone**- ly **fa**milies **boil** his **spe**cies with **frog** in a **frog** . .	in . **has** . . . In with **sen**sitive **trunk** with **af**rican **af**rican **pla**- (ti)num **whis**kers with **lit**- tle **trou**ble be **ve**- ry **sen**sitive . **child** with a **gap** . .

Table 8.3: Best individual for $F_{target} = \texttt{limerick}$, initial form-aware test

	elitist 0%	elitist 20%	elitist 40%
Score	0.82	0.90	0.94
Surface	$\boxed{\text{NP}}$, with a bandy very large waste with the platinum lion , the mind is his waste with the product . in a boy , with a african pole in his bill with his whiskers , his platinum toad in his bill in her dwells in his bean . his hippopotamus will be the frog in a african child in a soil with the fish with the tiger with the grin in his bean .	$\boxed{\text{S}}$. $\boxed{\text{S}}$. $\boxed{\text{S}}$. $\boxed{\text{S}}$. with $\boxed{\text{D}}$ baboon in $\boxed{\text{NP}}$, $\boxed{\text{D}}$ baboon , it is platinum . in $\boxed{\text{D}}$ facts , in the platinum boy , in the sensitive animal in $\boxed{\text{D}}$ lion , a sensitive platinum elephant dwells with a elephant in $\boxed{\text{D}}$ platinum tail with the facts in a sensitive elephant with $\boxed{\text{D}}$ large hippopotamus in $\boxed{\text{D}}$ african head with a head with $\boxed{\text{D}}$ expense with $\boxed{\text{NP}}$ in $\boxed{\text{NP}}$. $\boxed{\text{NP}}$ dwells with his sensitive jaws . $\boxed{\text{S}}$. $\boxed{\text{S}}$.	$\boxed{\text{S}}$. in his african blubber , a elephant dwells with his sensitive dish with his sensitive sensitive dish with the african tiger in $\boxed{\text{D}}$ animal in $\boxed{\text{D}}$ frog in a sense in a treatment . in facts , with him , african mothers , they are with the sensitive species in $\boxed{\text{D}}$ platinum dish with a sense in a blubber . his animal dwells with a child .
Formatted	, with a **ban-** * dy **ve-** ry $\underline{\text{large}}$ **waste** with the **pla-** tinum **li-** on , the **mind** is his **waste** with the **pro**(duct) . In a **boy** , with an **af-** rican **pole** in his **bill** with his **whis-** kers , his **pla-** tinum **toad** in his **bill** in her **dwells** in his **bean** . (His) $\overline{\text{hippo}}$**pot-** amus $\underline{\textbf{will}}$ be the **frog** in an **af-** rican **child** in a **soil** with the **fish** with the **ti-** (ger) with the **grin** in his **bean**.	. . . . with ba**boon** in , ba**boon** , it is **pla-** (ti)num . In **facts** , in the **pla-** tinum **boy** , in the **sen-** sitive **a-** (ni)mal in **li-** on , a **sen-** sitive **pla-** tinum **e-** lephant **dwells** with an **e-** (le)phant in **pla-** tinum **tail** with the **facts** in a **sen-** sitive **e-** $\underline{\text{(le)}}$phant with **large** $\overline{\text{hippo}}$**pot-** (a)mus in **af-** rican **head** with a **head** with ex**pense** with in . **dwells** with his **sen-** sitive **jaws** . . .	. In his **af-** rican **blub-** ber , an **e-** lephant **dwells** with his **sen-** sitive **dish** with his **sen-** sitive **sen-** sitive **dish** with the **af-** rican **ti-** ger in **a-** (ni)mal in **frog** in a **sense** in a **treat-** ment . In **facts** , with him , **af-** rican **mo-** thers , they $\underline{\textbf{are}}$ with the **sen-** sitive **spe-** cies in **pla-** tinum **dish** with a **sense** in a **blub-** ber . His **a-** nimal **dwells** with a **child** .

Table 8.4: Best individual for $F_{target} = \texttt{mignonne}$, initial form-aware test

in mind that this is the only requirement for an optimal solution, given the simplifications of ignoring linebreaks and substitution holes.

The EA solves this problem easily. After roughly 10 iterations, an optimal score of 1.00 is always achieved (Figure D.1(a)). This is true of all elitist ratios used.

The three sample outputs in Table 8.2, however, are distinctly unimpressive as haikus. Since every run manages to produce an optimal solution, we have simply chosen the first run for each of the elitist ratios.

The main problem that we can observe from these solutions is that they are incomplete derivations, due to the presence of substitution nodes that are not yet substituted. This is particularly evident in the solutions for elitist ratios 0% and 40%. Although these candidate solutions can be viewed as underspecifications of grammatical constructions, the metre edit distance function used does not take this fact into consideration. It simply compares the sequence of existing syllables with the target metre, ignoring substitution nodes, and produces an optimal alignment.

The formatted surface texts are intended to show this alignment. However, with the removal of the substitution nodes, the formatted surface forms are clearly ungrammatical. Nevertheless, it can be observed that these solutions are indeed satisfying the target metre insofar as their lexical items comprise a sequence of seventeen syllables.

Another problem that can be observed is the awkward linebreaking, for example at the end of the second line of the haiku for elitist ratio 20% (*'hip-popotamus'*).

Finding an optimal solution for the `limerick` target form is more of a challenge than finding one for the `haiku` target form. For the non elitist-strategy EA, one run yielded a best candidate solution that has a fitness of only 0.68 (see Table D.1).

The best fitness score obtained with a non-elitist EA is 0.82, and the individual with this score is shown in Table 8.3. It requires several deviations from the target form, namely the insertion of `w` syllables (two in the first line, one in the second line), and the deletion of an unstressed syllable (the second syllable of *'little'* in the fourth line). Even after ignoring the fact that it is ungrammatical, this solution barely reads like a limerick.

For elitist ratio 0%, the EA occasionally finds solutions with high fitness (indicated by the 'spikes' in Figure D.1(b)), but due to its non-elitist nature, it is not preserved, resulting in a missed opportunity.

Using an elitist ratio of 20%, however, enforces a monotonic increase of the maximum of the best scores. As can be seen in Table 8.3, it produces a text that is almost metrically perfect with a score of 0.95. The only edit required is the stressing of the secondary stressed syllable in *'hippopotamus'* at the end of the second line, which incurs a cost of 1 (see Table 8.1). Strictly speaking, the stressing of a secondary stressed syllable is not a grave metrical problem. However, this particular instance is made worse by the fact that it is followed by a very awkward linebreak. In fact, this particular limerick has awkward enjambment for the first three lines (i.e.*'ve-ry', 'hip-popotamus',* and *'lone-ly'*).

The behaviour of the test runs with elitist ratio 40% is almost identical to that of elitist ratio 20%. They both achieve a maximum best score of 0.95 and a minimum best score of 0.79. It is interesting to note that the mean of the best scores for elitist ratio 40% is lower than for 20%. Moreover, for elitist ratio 20%, 3 out of 10 test runs scored 0.95, whilst only 1 out of the 10 test runs for elitist ratio 40% managed this score. This suggests that at this rate of elitist population the selection pressure may be too high, leading to premature convergence.

As with the `haiku` texts, the `limerick` texts also exhibit the serious problem of being ungrammatical due to their incompleteness.

As for the `mignonne` target form, the behaviour of the test runs are similar to those for `limerick`. Utilizing an elitist strategy clearly improves the performance of the EA for this task. Although it may seem that an elitist ratio of 40% performs better than 20% due to the maximum best fitness score of 0.94 as opposed to 0.90, the average of the best fitness scores is quite similar, and in fact the minimum best fitness score of 0.75 is worse than that of elitist ratio 20% (see Table D.1).

For a non-elitist EA the average of the best fitness scores eventually stagnates around 0.7 (see Figure D.1(c)). We also see for the non-elitist EA instances of high scoring individuals appearing throughout the evolutionary process but being lost in subsequent populations. Unlike for `limerick`, however, these scores never approach the best scores reached by the elitist EAs (>0.90).

As for the sample output as shown in Table 8.4, we can see from the formatted surface forms that they are rife with awkward enjambments, e.g.*'e-lephant', 've-ry', 'pla-tinum'*, etc. This shows the potential of the `mignonne` target of being a good test for the ability of an EA to satisfy linebreaking in a target form adequately. Since we ignore linebreaking in this test, we end up with these solutions. What is optimized, however, is the metre, and by and large the elitist strategy EAs achieve a reasonable solution. The solution found by elitist ratio 40% only has two metrical problems: the deletion of the unstressed syllable *'ni'* in *'animal'* on line 13, and the stressing of the unstressed syllable *'are'* on line 19.

Discussion

The results of the initial test roughly correspond to our expectations.

Firstly, the `haiku` target form is an almost-trivial optimisation task, given the simplifications adopted during this initial test.

The initial hillclimbing for `haiku` is extremely rapid and almost instantaneous (see Figure D.1). For the more difficult targets of `limerick` and `mignonne` the ascent becomes gradually smoother.

Secondly, from the `limerick` and `mignonne` test runs, it is evident that the elitist strategy implemented by using an elitist population of 20% and 40% shows a clear improvement in the performance of the EA, and this is reflected both by the fitness scores obtained and by the quality of the output. For example, compare the formatted surface of the best individual for the `limerick` target form with elitist ratio 0% and elitist ratios 20% and 40%. There are many other methods for applying selection pressure in a proportionate selection EA, such as sigma scaling or altering the normalization of $\mathcal{F}_{edit}$ through the parameter λ_1. However, the simple elitist strategy used here seems sufficient for this task.

Unfortunately, contrary to our expectation, the EA failed to find an optimal score of 1.0 for all three target forms. For `limerick`, the highest score achieved was 0.95, and for `mignonne`, it was 0.94. Finding the reason for this failure is non-trivial: it could be any combination of factors ranging from the specific EA parameters used, the domain knowledge available, or the genetic operators. However, it is a feature of EAs that it provides such "satisfycing" solutions, and given the metrical qualities of these candidate solutions, they seem of sufficient similarity to the targets to be similar to human-written texts.

Finally, as expected, the output text of the candidate solutions severely suffer from the following two problems:

- **Syntactic holes:** the candidate solutions being produced by the baseline operators are not guaranteed to be complete derivations, and the evaluation function used ignores this aspect completely. Thus, the formatted surface texts shown in Tables 8.2 to 8.4, which simply skip over substitution nodes, are clearly ungrammatical.

- **Linebreaking:** awkward enjambment, particularly for the `mignonne` target form which consists of 28 lines.

Note that these problems are not failings of the EA. We deliberately chose to ignore these aspects for this initial test. In the next two sections we will address the issue of substitution nodes, and in Section 8.3.4 we will address the issue of linebreaking.

8.3.2 Plugging the holes: syntax-surface balance test

This is the first of two tests in which we try to repeat the initial test but with slight alterations in an attempt to overcome the problem of substitution nodes that result in candidate solutions with ungrammatical texts.

In this test we aim to solve the problem by using $\mathcal{F}_{metre}$, defined in Section 6.3.7, as the evaluation function. It incorporates the syntax-surface balance function $\mathcal{F}_{balance}$, which is intended to account for substitution nodes. We set $\lambda_3 = 0.7$. For the $l_{estimate}$ values, we use the very ad-hoc estimates in Table 6.9. The parameters for $\mathcal{F}_{edit}$ are as they were during the initial test.

We also use the same three forms used in the initial test: `haiku`, `limerick` and `mignonne`.

The EA setup is similar to that used in the initial test. However, in this test we only use an elitist strategy EA with an elitist ratio of 20% of the population size, as the initial test indicated this to be an appropriate choice for the task.

We also use the same genetic operators and linguistic resources used during the initial test.

Expectations

The best individuals found by the EA should have far fewer holes than the ones found during the initial test. Note that the fitness scores in this test are not comparable to those in the initial test, as the evaluation function used is different. Thus, we can only rely on observations from the actual output texts. Fortunately, it should be easy to accomplish this objectively.

Score	1.00
Surface	his families are in a baboon . his epithets are platinum .
Formatted	His families are in a baboon. His epi- thets are platinum.

Table 8.5: Best individual for F_{target} = `haiku`, syntax-surface balance test

Score	0.83
Surface	a lion has a baboon . in $\boxed{\text{D}}$ mothers , his mothers are his bullets , that boil its treatment , with $\boxed{\text{D}}$ bullets . $\boxed{\text{D}}$ mothers boil his treatment .
Formatted	A **li**on * **has** a ba **boon**. In **mo**thers, his **mo**thers * **are** his **bul**lets, that **boil** its **treat**ment, with **bul**- lets. **mo**thers * **boil** * his **treat**(ment) .

Table 8.6: Best individual for F_{target} = `limerick`, syntax-surface balance test

Results

We ran this test three times, once for each target form. Each test was run five times. Summary statistics and graphs for this test are presented in Section D.1.2 of Appendix D.

Tables 8.5, 8.6, and 8.7 show the best individuals that yield the best fitness scores from the last populations of each test. They show the best individuals for the `haiku`, `limerick`, and `mignonne` target forms respectively. For each individual we show its fitness score, raw surface text, and formatted surface text. See the guide given in Section 8.3.1 for an explanation of these formatted surface texts and how to interpret them.

Observations

As in the initial test, the EA has no problems in achieving the `haiku` target form (see Table D.2). Figure D.2(a) is virtually identical to Figure D.1(a).

Looking at the raw surface text of the best individual in Table 8.5, we see that there is an improvement over the individual in Table 8.2, i.e. there are no more spurious substitution nodes, as they have all been substituted, thus leading to a grammatical text in the formatted surface form shown.

Score	0.84
Surface	with a animal , $\boxed{\text{D}}$ very sensitive epithets , $\boxed{\text{Comp}}$ are his very sensitive shoulders , are very mild with his very big knees , $\boxed{\text{Comp}}$ are the very lonely whiskers , with $\boxed{\text{D}}$ very extinct hippopotamus , $\boxed{\text{Comp}}$ $\boxed{\text{CV}}$ african , with $\boxed{\text{D}}$ very mild hippopotamus . $\boxed{\text{NP0}}$ dwells $\boxed{\text{P}}$ a baboon . a lion has a expense .
Formatted	With an **a-** nimal, **ve-** ry * **sen-** sitive **e-** * pi**thets** , are his **ve-** ry * **sen-** sitive **shoul-** ders , are **ve-** ry * **mild** with his **ve-** ry * **big** * * **knees** , are the **ve-** ry * **lone-** ly * **whis-** kers , with **ve-** ry ex**tinct** h*ip*po**pot-** amus , **af-** (ri)can , with **ve-** * ry **mild** h*ip*po**pot-** amus . **dwells** a ba**boon** . * A **li-** * on **has** an ex**pense** .

Table 8.7: Best individual for F_{target} = `mignonne`, syntax-surface balance test

The scores for the `limerick` and `mignonne` target forms in Table D.2 show that the EA is failing to find an optimal solution for these tasks. Intuitively, this can be explained by the fact that the task is now more constrained, i.e. it must achieve syntax-surface balance as well. The EA very rapidly converges to its best fitness scores and stagnates at that level (see Figure D.2(b) and (c)).

The tradeoff between $\mathcal{F}_{edit}$ and $\mathcal{F}_{balance}$ can be seen in the best individuals for `limerick` (Table 8.6) and `mignonne` (Table 8.7). Compared to their counterparts in the initial test (Table 8.3 and 8.4, elitist ratio 20%), they have far fewer substitution nodes, and of those still remaining, they are for categories with $l_{estimate} = 1$, e.g. $\boxed{D}$, $\boxed{Comp}$, $\boxed{CV}$, and $\boxed{P}$, except for the single $\boxed{NP}$ node in the `mignonne` candidate.

As a result, the formatted surface texts, where substitution nodes are skipped over, are more grammatical. However, from a metre evaluation viewpoint, they are inferior to the individuals obtained during the initial test, i.e. there are many more edit operations required.

Again, awkward linebreaking and enjambment is still very evident for these best individuals.

Discussion

The results obtained for the three target forms show an improvement in the output quality achieved by accounting for the scoring potential of substitution nodes, where there are no longer any remaining substitution nodes, as in Table 8.5, and for those few remaining in Table 8.6 and 8.7, they are almost all categories with $l_{estimate} = 1$.

Unfortunately, there is a tradeoff between $\mathcal{F}_{balance}$ and $\mathcal{F}_{edit}$, and the metre similarity of the best individuals for `limerick` and `mignonne` are inferior to their counterparts obtained during the initial test.

The populations stagnate with individuals of similar scores (see Figure D.2(b) and (c)), suggesting that this multiple objective task is diminishing the informedness of heuristics provided by the metre evaluation function, $\mathcal{F}_{edit}$.

As expected, the awkward linebreaking and enjambment problem is still evident.

8.3.3 Plugging the holes: complete derivation operators test

This is the second test where we attempt to overcome the problem of substitution nodes. In this test we aim to solve the problem by using the compound operators that ensure complete derivations, as defined in Section 7.4.1.

Again, we use the `haiku`, `limerick` and `mignonne` target forms used during the initial test (Section 8.3.1). We also use the same evaluation function, i.e. $\mathcal{F}_{edit}$, and linguistic resources that were used during the initial test.

The EA setup is similar to that used in Section 8.3.2, where we only use an elitist strategy EA with an elitist ratio of 20% of the population size.

The only aspect that we vary in this test is the use of the compound operators that 'close off' derivations presented in Section 7.4.1. We use the three operators that correspond to the baseline blind operators used in the initial test, and their probabilities being applied are also the same as in the initial test:

- BLINDCREATECOMPLETE = 0.5
- BLINDADJOINCOMPLETE = 0.3
- BLINDDELETECOMPLETE = 0.2

Again, no crossover operation is used.

Expectations

The compound operators used here will close off the derivation of any candidate solution, thus all solutions will be grammatical. However, this also means a coarser level of granularity: the operators are blindly performing more operations to an individual before subjecting them to evaluation and selection. Whether or not this coarser granularity will be detrimental to the performance of the EA remains to be seen, but we predict the overall performance of the EA to be, at best, equal to that obtained during the initial test, and most probably slightly worse. Note that unlike in Section 8.3.2, the fitness scores are directly comparable to those obtained from the initial test due to the same evaluation function being used.

Score	1.00
Surface	facts , they are round . african facts , they are in a child . a bill is rare .
Formatted	Facts, they are round. Af- rican facts, they are in a child. A bill is rare.

Table 8.8: Best individual for F_{target} =`haiku`, complete derivation operators test

Score	1.00
Surface	in facts , with a bill with a shocking town in a tail in his fish , his blubber will boil his jaws in a bean in mothers . his boy is a mind .
Formatted	In **facts**, with a **bill** with a **shock**- ing **town** in a **tail** in his **fish**, his **blub**ber will **boil** his **jaws** in a **bean** in **mo**thers. His **boy** is a **mind**.

Table 8.9: Best individual for F_{target} = `limerick`, complete derivation operators test

Results

We ran this test three times, once for each target form. Each test was run five times. Summary statistics and graphs for this test are presented in Section D.1.3 of Appendix D.

Tables 8.8, 8.9, and 8.10 show the individual that yields the best fitness scores from the last populations of each test. They show the best individual for the `haiku`, `limerick`, and `mignonne` target forms respectively, and for each individual, we show its fitness score, raw surface form, and formatted surface text. See the guide given in Section 8.3.1 for an explanation of these formatted surface texts and how to interpret them.

Observations

The `haiku` target form is consistently met with an optimal solution with fitness of 1.0. This is achieved even faster than during the initial test (see Figure D.3(a)). This is to be expected, as the compound operators will likely produce 17 syllables worth of text much faster. The sample haiku in Table 8.8 is a complete and grammatical text that has the correct syllable count, albeit rather awkward linebreaking (*'af-rican'*).

The summary scores for the `limerick` target form in Table D.3 show that the EA is doing even

Score	0.96
Surface	in a product , a animal dwells in his blubber . his names with a elephant boil his mothers . his sensitive mothers , they boil the expense . with a bean with the boy , in a bean in a blubber , a fish with the jaws in it has them . the waste is his blubber . his names will be names with a tiger . a dish is a lion . his men , they will play .
Formatted	In a **pro-** duct, an **a-** nimal **dwells** in his **blub-** ber. His **names** with an **e-** lephant **boil** * his **mo-** thers. His **sen-** sitive **mo-** thers, they **boil** the ex**pense**. With a **bean** with the **boy** , in a **bean** in a **blub-** ber, a **fish** with the **jaws** in it **has** them. The **waste** is his **blub-** ber. His **names** will be **names** with a **ti-** ger. A **dish** is a **li-** on. His **men**, they will **play**.

Table 8.10: Best individual for $F_{target} = \texttt{mignonne}$, complete derivation operators test

better than during the initial test (Table D.1). It also manages to find an optimal solution with a fitness of 1.0, which is shown in Table 8.9. This is somewhat surprising, given our expectation that it would do slightly worse than the initial test.

The fitness scores obtained for the `mignonne` target form in Table D.3 are also consistenly better than during the initial test (Table D.1). The highest scoring solution does not reach an optimal 1.0, but only 0.96. It requires one edit of an insertion of a weak syllable, which we feel is acceptable given the size of the target form. Unfortunately, the awkward linebreaking prevents it from reading like a natural-sounding text that satisfies the `mignonne` form.

Discussion

The results from this test were surprisingly better than expected. That the output texts are grammatical due to them being complete derivations was to be expected, as this is something that the compound operators used will guarantee. However, the scores for metre similarity are better than during the initial test, which, given the coarser level of granularity, is rather counter-intuitive. One possible explanation is that the BLINDCREATECOMPLETE operator used introduces a lot of genetic diversity that is beneficial to the EA. This is because unlike BLINDADDSUBSTITUTE, it does not perform substitution incrementally to an existing individual, as all individuals no longer have remaining substitution nodes. Instead, it introduces a completely new individual, i.e. starting from the distinguished symbol. In effect, it is a far less local move in the search space than BLINDADDSUBSTITUTE achieves. Throughout the evolutionary process, this possibly results in faster coverage of the search space.

These results suggest that there is little justification for using the baseline operators on their own. Not only do these compound operators solve the problem of substitution nodes, the performance of the EA is better than those observed in the previous two tests.

8.3.4 Enjambment and linebreaking test

In this test we attempt to overcome the problem of awkward linebreaking. All the previous tests ignored this aspect, resulting in texts which, although possibly impeccable in terms of metre similarity, suffered from awkward linebreaking and enjambment. We aim to solve the problem by imposing a heavy penalty for inserting linebreaks.

Substitution

Cost	w	s	x	b
0_1	0	2	0	∞
0_n	0	2	0	∞
1_1	3	0	0	∞
1_n	3	0	0	∞
2_n	1	1	0	∞
b	∞	∞	∞	0

Insertion

	Cost
w	1
s	3
x	1
b	10

Deletion

	Cost
0_1	1
0_n	1
1_1	3
1_n	3
2_n	2
b	0

Table 8.11: Edit distance costs for enjambment and linebreaking test

We use the same target forms, linguistic resources, EA parameters and genetic operators used in the previous test (Section 8.3.3).

As for the evaluation function, we use the metre edit distance function, $\mathcal{F}_{edit}$, used in the previous test, but with a different operation cost table, shown in Table 8.11. The only difference is that inserting a linebreak target syllable, b, incurs a relatively expensive cost of 10.

Expectations

Given the cost of inserting linebreaks, the task of satisfying the target forms is now more difficult. This is particularly true for mignonne, which has 28 lines. Thus, we expect the scores for mignonne to be lower than those obtained in the previous test (Table D.3). However, given the high penalty incurred for arbitrarily inserting linebreaks, we expect lines to end with punctuation marks which cost nothing to substitute with a b target syllable.

Results

We ran this test three times, once for each target form. Each test was run five times. Summary statistics and graphs for this test are presented in Section D.1.4 of Appendix D.

Tables 8.12, 8.13, and 8.14 show the individual that yields the best fitness scores from the last populations of each test. They show the best individual for the haiku, limerick, and mignonne target forms respectively, and for each individual, we show its fitness score, raw surface form, and formatted surface text. See the guide given in Section 8.3.1 for an explanation

Score	1.00
Surface	men , they play . in her , a boy dwells in his treatment . his species is rare .
Formatted	Men, they play. In her, a boy dwells in his treatment. His species is rare.

Table 8.12: Best individual for $F_{target} = \texttt{haiku}$, enjambment and linebreaking test

Score	1.00
Surface	they play . a expense is a waist . a lion , he dwells in a dish . he dwells in a skin . a sensitive child , he dwells in a child with a fish .
Formatted	They **play**. An ex**pense** is a **waist**. A **lion**, he **dwells** in a **dish**. He **dwells** in a **skin**. A **sens**itive **child**, he **dwells** in a **child** with a **fish**.

Table 8.13: Best individual for $F_{target} = \texttt{limerick}$, enjambment and linebreaking test

of these formatted surface texts and how to interpret them.

Observations

Despite the more difficult task, the haiku target form is still easily achieved by the EA (see Figure D.4(a)). The main difference, however, is that the output in Table 8.12 is a grammatical, well-formed haiku that has linebreaks coinciding with punctuation marks. Note that in the first line the linebreak occurs mid-sentence *("In her, a boy dwells in his treatment").*

As for the limerick target form, Table D.4 the EA is essentially doing as well as during the previous test (Table 8.9), where the only difference was the lack of a cost for inserting a linebreak. The rate of growth in fitness is also similar (compare Figure D.4(b) with Figure D.3(b)). The output in Table 8.13, however, shows a clear improvement in terms of linebreaking. Every line consistently ends with either a sentence or phrase boundary, whilst retaining the same quality of metre similarity as before.

The mignonne target form is a good form to use in testing the ability of the EA in achieving natural linebreaking, due to it being 28 lines long. As expected, it turns out to be the most difficult test yet for the EA. The scores for mignonne in Table D.4 are considerably lower than during the previous test (Table D.3), and it fails to find an optimal solution. Moreover, the rate

Score	0.82
Surface	with a bean , in a toad , in a soil , his bullets , they with a sense , they are sensitive . in a frog , in a mind , with his trunk , with his fish , with the toad , in a mind , the mothers , they with facts , they are men . in a man , in the bean , in a hand , with the hand , in a bill , in a boy , his men , they , they play . with his fish , in his town , in his knees , in the mothers , she is his blubber .
Formatted	With a **bean**, in a **toad**, in a **soil**, * his **bul**(lets), (they) with a **sense**, they are **sen**(si)(tive). In a **frog**, in a **mind**, with his **trunk**, with his **fish**, with the **toad**, in a **mind**, * the **mo**(thers), they with **facts**, they are **men**. In a **man**, in the **bean**, in a **hand**, with the **hand**, in a **bill**, in a **boy**, * his **men**, they, they **play**. With his **fish**, in his **town**, in his **knees**, in the **mo**(thers), (she) is his **blub**(ber).

Table 8.14: Best individual for $F_{target} = \texttt{mignonne}$, enjambment and linebreaking test

of growth for the maximum and average of the best fitness scores is much slower than in the previous test (compare Figure D.4(c) with Figure D.3(c).

The sample output in Table 8.14 shows that the linebreaks are indeed occurring at sentence and phrase boundaries, but at the expense of metre similarity: compared to the output in Table 8.10, it requires far more insertions and deletions of syllables. One interesting point is that the linebreaking is achieved through extensive use of the prepositional phrase auxiliary tree that adjoins at the beginning of a sentence.

Discussion

The heavy penalty incurred for arbitrarily inserting a linebreak results in solutions that have naturally occurring linebreaks, i.e. ones that coincide with sentence and phrase boundaries. For the `haiku` and `limerick` target forms, this means that the EA has managed to produce a syntactically well-formed and metrically perfect text.

Unfortunately, the 28 lines of `mignonne` make it a very difficult target form to satisfy, and the results in this test show that it is unable to find an optimal solution. The slow growth rate of the scores (see Figure D.4(c)) suggests that running the EA for a longer time may yield better scores for this particular test. Furthermore, it will be interesting to see how the EA will perform without the availability of the prepositional phrase auxiliary tree, which it uses extensively in this test to achieve natural linebreaking for `mignonne`. These two aspects are the focus of the next two tests.

8.3.5 Effect of scale test

In this test we essentially repeat the previous test, i.e. the enjambment and linebreaking test (Section 8.3.4), specifically for the `mignonne` target form, and extend the EA run to 2000 iterations. Every other aspect of the test is the same. We perform this test due to our observation of the slow growth rate in Figure D.4(c), which suggests that running the EA for longer may improve the final solution.

Expectations

We expect the output of the best individual obtained to be a marked improvement over the one obtained in the previous test (Table 8.14).

Results

Due to the heavy computational cost of this task, we ran this test three times as opposed to five. Summary statistics and graphs for this test are presented in Section D.1.5 of Appendix D.

Table 8.15 presents the individual that yields the best fitness score obtained from these runs, showing its fitness score, raw surface text, and formatted surface text. See the guide given in Section 8.3.1 for an explanation of these formatted surface texts and how to interpret them.

Observations and discussion

Comparing the statistics in Table D.5 with those for the `mignonne` target form in Table D.4, we see that increasing the duration of the EA search does not markedly improve the best fitness score achievable.

There is a slight increase, e.g. from 0.83 to 0.85, as reflected by the fewer edit operations required in the best individual in Table 8.15 (6 deletions and 3 insertions as opposed to 8 deletions and 3 insertions for the individual in Table 8.14), but it is perhaps not what one would hope for given the fourfold increase in computational effort. However, the minimum and mean values for the best fitness scores does show that the extended search duration is yielding consistently high scores, and that the 0.82 fitness achieved by the run in the previous test was a ‘lucky’ one.

The best fitness is obtained after roughly 1100 iterations (see Figure D.5).

We can also observe from the best individual that it is still relying heavily on the prepositional phrase to achieve natural linebreaks.

Score	0.85
Surface	they are whiskers . he will play . in a boy , it is a bean . in a bean , in a bean , in a grin , with a soil , with a soil , with a grin , he dwells in her . it will play . in a bean , with a trunk , a gap , it is with mothers . they will play . in a frog , in a dish , in a bill , in a bean , in a child , with a grin , they are knees . in a tail , with a grin , his men , they will play .
Formatted	They are **whis**(kers). He will **play**. In a **boy**, (it) is a **bean**. In a **bean**, in a **bean**, in a **grin**, with a **soil**, with a **soil**, with a **grin**, * he **dwells** (in) (her). It will **play**. In a **bean**, with a **trunk**, * a **gap**, (it) is with **mo**(thers). They will **play**. In a **frog**, in a **dish**, in a **bill**, in a **bean**, in a **child**, with a **grin**, They are **knees**. In a **tail**, with a **grin**, * his **men**, they will **play**.

Table 8.15: Best individual for effect of scale test

8.3.6 Reduced grammar test

In this test we essentially repeat the enjambment and linebreaking test (Section 8.3.4) for the `mignonne` target form, but with a reduced set of linguistic resources, in order to observe the impact the linguistic resources can have on the performance of the EA. More specifically, we do not use the family of prepositional phrases (Appendix C.2.6) and relative clauses (Appendix C.2.7), two groups of auxiliary trees that we believe can be used as in Table 8.14 and 8.15 to achieve the required lines of three syllables.

Apart from this reduced grammar, all other aspects of this test are the same as for the enjambment and linebreaking test in Section 8.3.4.

Expectations

We expect the performance of the EA to be worse than that achieved in the previous two tests, where the prepositional phrase was very extensively used to satisfy the `mignonne` target form.

Results

We ran this test five times. Summary statistics and graphs for this test are presented in Section D.1.6 of Appendix D.

Table 8.16 presents the individual that yields the best fitness score obtained from these runs, showing its fitness score, raw surface text, and formatted surface text. See the guide given in Section 8.3.1 for an explanation of these formatted surface texts and how to interpret them.

Observations

With this reduced grammar, the EA clearly struggles to assemble a text that satisfies the `mignonne` target form. The drastic reduction in quality, both from the scores in Table D.6 and Figure D.6, and the surface form in Table 8.16, indicates that the limited linguistic resources do not provide ample opportunities to satisfy the target form. In place of the prepositional phrases that were used extensively in Table 8.14 and 8.15, it relies heavily on the left dislocated noun phrase (e.g.*"His men, they will play"*), but this requires a lot of syllable insertions and deletions to satisfy the target form.

Score	0.62
Surface	he will play . his knees , they are in her . his knees , they are mild . they are mild . his men , they will play . a table , it is with them . his men , they are with jaws . a hippopotamus , he is good . a child , he dwells in her . a treatment , it has them . a boy , he is in her . a sense , it is platinum . a toad , it is with knees . they will play . a dish , it is lonely . it will play .
Formatted	He will **play**. * His **knees**, (they) are in **her**. * His **knees**, they are **mild**. They are **mild**. * His **men**, they will **play**. * A **ta**(ble), (it) is with **them**. * His **men**, (they) are with **jaws**. (A) hippo**pot**(a)(mus), he is **good**. * A **child**, * he **dwells** (in) (her). * A **treat**(ment), * it **has** (them). * A **boy**, (he) is in **her**. * A **sense**, it is **pla**(ti)(num). * A **toad**, (it) is with **knees**. They will **play**. * A **dish**, it is **lone**(ly). It will **play**.

Table 8.16: Best individual for reduced grammar test

Discussion

This test shows the effect the linguistic resources can have on the quality of the output. Although we have demonstrated an extreme case by targeting a specific family of elementary trees that was observed to be extensively used, it suggests that in general, the more linguistic resources available, the more opportunities it will provide the EA to satisfy the target metre with.

8.3.7 Crossover operator test

Up until this point, the most constrained task that we have tested is the satisfying of the `haiku`, `limerick`, and `mignonne` target forms with the edit distance costs used in the enjambment and linebreaking test (Table 8.11), using the compound operators that ensure complete derivations. These represent ‘perfect’ texts in that they are definitely grammatical (due to the use of compound operators), should satisfy the target metre, and naturally align linebreaks with punctuation marks that mark phrase boundaries, e.g. commas and fullstops. In Section 8.3.4 we obtained optimal solutions to this task for the `haiku` and `limerick` forms, shown in Tables 8.12 and 8.13. However, thus far we have been unable to obtain an optimal solution for the `mignonne` target form, even after letting the EA run for 2000 iterations (Section 8.3.5).

There are various aspects that we can alter in an attempt to solve this problem, such as choice of selection algorithm, population size, and other EA parameters, but in this test we will observe the impact of using the crossover operator, BLINDSWAP.

Recall from Section 4.2.4 that mutation is often viewed to be best at creating genetic diversity within a population, and that crossover promotes emergent behaviour from partial solutions. Given the nature of the `mignonne` target form, we believe that the task of satisfying it is amenable to the crossover operator.

In previous tests, the genetic operators that have been used when constructing a population have been limited to mutation. In other words, the probability of applying mutation to a selected parent, $p_{mutation}$, is 1, and of crossover, $p_{crossover}$, is 0. In this test we will examine the effect of varying these probabilities.

In this test we will only use the `mignonne` target form. All remaining aspects of the test will be the same as for the enjambment and linebreaking test, i.e. linguistic resources, evaluation function, and mutation operators.

Expectations

We hope the crossover operator used, BLINDSWAP, will be able to combine partial solutions to create an individual that is an optimal solution, i.e. has a fitness score of 1.0.

Results

We ran this test three times, assigning $p_{mutation}$ the values of 0.25, 0.5, and 0.75 (and thus assigning $p_{crossover}$ 0.75, 0.5, and 0.25). Each test was run five times. Summary statistics and graphs for this test are presented in Section D.1.7 of Appendix D.

Table 8.17 presents the individuals that yield the best fitness score obtained from these tests, showing its fitness score, raw surface text, and formatted surface text. See the guide given in Section 8.3.1 for an explanation of these formatted surface texts and how to interpret them.

Observations and discussion

Comparing the results in Table D.7 to the results for `mignonne` in Table D.4, we see that the runs where $p_{crossover} = 0.25$ and 0.50 yield better solutions, i.e. with fitness scores of 0.9 and 0.96 respectively. Unfortunately these are still not optimal solutions.

For $p_{crossover} = 0.75$, however, the best individual obtained only has a fitness of 0.75, worse than that obtained during the enjambment and linebreaking test, and one might interpret this as being caused by a lack of genetic diversity which BLINDSWAP can manipulate.

Observing the individuals themselves in Table 8.17, and comparing them to the individual in Table 8.14, we can clearly see the role that the crossover operator is playing in the manipulation of the texts to satisfy the metre. For all three individuals shown, there is an extensive repetition of phrases, both in terms of single lines and larger groups of lines.

In the first individual, i.e. for $p_{crossover} = 0.25$, the sentence *"In a bean, in a bean, with a bean, his knees, they will play."* is essentially repeated six times to make up the entire text, with a slight variation in the third instance, where two of the three prepositional phrases preceding the main sentence are deleted. Similar repetitions can be seen for the other two individuals.

	$p_{mutation} = 0.75$ $p_{crossover} = 0.25$	$p_{mutation} = 0.50$ $p_{crossover} = 0.50$	$p_{mutation} = 0.25$ $p_{crossover} = 0.75$
Score	0.90	0.96	0.75
Surface	in a bean , in a bean , with a bean , his knees , they will play . in a bean , in a bean , with a bean , his knees , they will play . in a bean , his knees , they will play . in a bean , in a bean , with a bean , his knees , they will play . in a bean , in a bean , with a bean , his knees , they will play . in a bean , in a bean , with a bean , his knees , they will play .	in a child , in his facts , that they boil , , in a child , in his facts , that they boil , , his facts , that they boil , are with facts , that they boil , in his facts , that they boil , with his facts , that they boil , . in a child , in his facts , that they boil , , in a child , in his facts , that they boil , , his facts , that they boil , are with facts , that they boil , in his facts , that they boil , with his facts , that they boil , .	his mothers , who will play , are with mothers , who will play , in his mothers , who will play , . his mothers , who will play , are with mothers , who will play , in his mothers , who will play , . his mothers , who will play , are with him . his mothers , who will play , are with mothers , who will play , in a man . they are kind . his mothers , who will play , are with mothers , who will play , in his mothers , who will play , . they are kind .
Formatted	In a **bean**, in a **bean**, with a **bean**, * his **knees**, they will **play**. In a **bean**, in a **bean**, with a **bean**, * his **knees**, they will **play**. In a **bean**, * his **knees**, they will **play**. In a **bean**, in a **bean**, with a **bean**, * his **knees**, they will **play**. In a **bean**, in a **bean**, with a **bean**, * his **knees**, they will **play**. In a **bean**, in a **bean**, with a **bean**, * his **knees**, they will **play**.	In a **child**, in his **facts**, that they **boil**,, in a **child**, in his **facts**, that they **boil**,, * his **facts**, that they **boil**, are with **facts**, that they **boil**, in his **facts**, that they **boil**, with his **facts**, that they **boil**,. In a **child**, in his **facts**, that they **boil**,, in a **child**, in his **facts**, that they **boil**,, * his **facts**, that they **boil**, are with **facts**, that they **boil**, in his **facts**, that they **boil**, with his **facts**, that they **boil**,.	* His **mo**(thers), who will **play**, are with **mo**(thers), who will **play**, in his **mo**(thers), who will **play**,. * His **mo**(thers), who will **play**, are with **mo**(thers), who will **play**, in his **mo**(thers), who will **play**,. * His **mo**(thers), who will **play**, are with **him**. * His **mo**(thers), who will **play**, are with **mo**(thers), who will **play**, in a **man**. They are **kind**. * His **mo**(thers), who will **play**, are with **mo**(thers), who will **play**, in his **mo**(thers), who will **play**,. They are **kind**.

Table 8.17: Best individuals for crossover operator test

An **a**nimal **dwells** in a **bill**.
An **a**nimal **dwells** in a **bill**.
He **has** * a **boy**,
(who) in **facts**, will be **large**.
An **a**nimal **dwells** in a **bill**.
(Score: 0.90)

Figure 8.1: Sample `limerick` output using BlindSwap crossover operator

It is interesting to note that despite McGonagall's lack of any hierarchical concept of poetic structure, such as that found in ASPERA and COLIBRI (Section 2.3.4), these individuals do show what can be seen as an inherent pattern of lines and repeating structures. This is, of course, dictated by pressure to align the linebreak target syllables with punctuation marks.

We experimented with satisfying the `haiku` and `limerick` forms using the crossover operator as well, and although we do not present the full results for these tests, a sample output is shown in Figure 8.1. We can see a similar repetition of phrases that conveniently coincide with lines in the given target forms.

8.3.8 Summary of discussion

The tests conducted during this stage reveal that McGonagall is able to find optimal solutions when faced with the task of only having to satisfy form constraints, and that these solutions do indeed exhibit metre patterns that are similar to their specified targets.

We believe that the compound operators that ensure complete derivations are the most appropriate genetic operators to use. Not only do they solve the awkward problem of substitution nodes, but the higher fitness scores obtained in Section 8.3.2 suggest that the genetic diversity introduced by the BlindCreateComplete operator helps the EA in rapidly covering the search space.

By imposing a relatively expensive penalty for arbitrarily inserting linebreaks in the text, we are able to guide the EA in producing texts which exhibit naturally occurring linebreaks, albeit at the possible expense of metrical similarity, particularly for the `mignonne` form.

Finally, the crossover operator, BlindSwap, is useful in combining partial solutions to build optimal solutions. It is only with this operator that a "satisfycing" near-optimal solution for the `mignonne` target form was obtained (Table 8.17).

8.4 McGonagall as tactical NLG component

Having experimented with McGonagall as a text generator that seeks only to satisfy form constraints in the first stage, in the second stage we will experiment with McGonagall as a more traditional NLG system, i.e. one that only strives to convey a given input semantics, and has no consideration for metre.

The tests are conducted in a similar fashion to those in the first stage: an initial test (Section 8.4.1) is conducted that defines the baseline configuration of operators, evaluators and EA parameters. Based on this, further tests are conducted in an attempt to overcome the encountered limitations. In particular, we examine the use of our semantically smart operators in Section 8.4.2, and the use of the even more explicitly goal-directed PROTECTORLike and SPUDLike operators in Section 8.4.3.

8.4.1 Initial tactical NLG test

For the initial test of McGonagall as tactical NLG component, we will define our baselines with regard to the insight gleaned from the previous stage. In particular, we will adopt the following three aspects:

1. We will use an elitist strategy EA.
2. We will use the compound operators that ensure complete derivations.
3. We will use crossover as a possible genetic operator.

Although it is arguable that our principle of least bias (Section 8.2) should dictate a more simplistic baseline configuration similar to that used in the initial form-aware test (Section 8.3.1), the results obtained in the first stage suggests that these are fair assumptions to make. These assumptions allow us to concentrate on presenting the primary issue under investigation, i.e. the use of our semantic similarity evaluation function in an EA framework to generate texts that show faithfulness towards a given target semantics.

“The Lion”: The african lion, he dwells in the waste, he has a big head and a very small waist; but his shoulders are stark, and his jaws they are grim, and a good little child will not play with him.
`lionhalf`: *lion(_,l), african(_,l), dwell(d,l), inside(_,d,was), waste(_,was), own(_,l,h), head(_,h), big(_,h), own(_,l,wai), small(s,wai), waist(_,wai), very(_,s)*
`lion`: *lion(_,l), african(_,l), dwell(d,l), inside(_,d,was), waste(_,was), own(_,l,h), head(_,h), big(_,h), own(_,l,wai), small(s,wai), waist(_,wai), very(_,s), own(_,l,sho), shoulders(_,sho), stark(_,sho), own(_,l,ja), jaws(_,ja), grim(_,ja), boy(_,c), good(_,c), little(_,c), play(pl,c), with(_,pl,l), will(wpl,pl), not(_,wpl)*

Table 8.18: `lionhalf` and `lion` encode the first 2 and 4 lines of “The Lion”

Target

Two sets of semantic propositions are used as S_{target}: `lionhalf` and `lion`, as shown in Table 8.18 (see also Appendix A.1). They are our representations of, respectively, the first two and four lines of Belloc’s poem *“The Lion”*, with a slight alteration where we have replaced the original opening noun phrase *“the lion, the lion”* with *“the african lion”*.

EA setup

- **Selection algorithm:** this test uses the proportionate selection algorithm with stochastic universal sampling, as used in the tests of the first stage. We employ an elitist strategy EA with elitist populations of 20% and 40% of the entire population.
- **Population size:** as in the first stage, we choose a population size of 40.

Evaluators

We use the semantic evaluation function defined in Section 6.4 to determine the semantic similarity between the semantics of the individuals generated during the EA and the target seman-

	α_1	α_2	α_3	α_4
Weighting scheme 1	1.0	1.0	1.0	1.0
Weighting scheme 2	4.0	2.0	1.0	1.0
Weighting scheme 3	1.0	2.0	4.0	1.0

Table 8.19: Three weighting schemes for the terms in the similarity equation

tics. We test three different weighting schemes for the terms in our similarity equation (see Section 6.4.3), and they are shown in Table 8.19.

Weighting scheme 1 is a baseline scheme where all four terms of our similarity equation are given equal weight. This is the weighting scheme used in Section 6.4.4. Scheme 2 is designed to emphasize the raw conceptual similarity between the target and candidate semantics, whereas scheme 3 is designed to emphasize the higher-order structural similarity as computed by our mapping algorithm (Section 6.4.2).

For all our tests we assume a symmetry between the target and candidate semantics, i.e. we choose $\varphi_1 = 0.5$ and $\delta_1 = 0.5$. In addition, as mentioned in Section 6.4.2, we use a simple binary-valued function $c(m)$ for conceptual similarity: two literals are considered conceptually similar if they share the same functor.

Operators

In this test we use the compound operators that ensure complete derivations, presented in Section 7.4.1 and first used in Section 8.3.3. The probabilities of them being applied when mutation is called for are also the same as during the first stage:

- BLINDCREATECOMPLETE = 0.5
- BLINDADJOINCOMPLETE = 0.3
- BLINDDELETECOMPLETE = 0.2

Additionally, when crossover is called for, the subtree swapping operator BLINDSWAP is used. As the three mutation operators above guarantee complete derivations, BLINDSWAP always guarantees complete derivations as well.

We assign the probabilities of applying genetic operators as $p_{mutation} = 0.6$ and $p_{crossover} = 0.4$.

Resources

We use the same linguistic resources used in the initial form-aware test, i.e. the handcrafted grammar and lexicon shown in Appendix C.

Expectations

- We hope that for each S_{target} there is at least one obtained individual where each literal in S_{target} is conveyed. This will be evident in the mapping shown for each obtained individual, where there should not be any dangling (i.e. unmapped) target literals.
- We expect that the task of conveying `lionhalf` should be easier than that of conveying `lion` given its reduced complexity.

Results

As there are three factors being varied in this test, i.e. target semantics, weighting scheme, and elitist ratio, we conduct twelve separate tests: $S_{target} =$ `lionhalf` with weighting schemes 1, 2, and 3, and $S_{target} =$ `lion` with weighting schemes 1, 2, and 3, all of which are run with elitist ratios of 20% and 40%. Each individual test is run ten times, and each run lasts for 500 iterations. Summary statistics and graphs for this test are presented in Section D.2.1 of Appendix D.

Tables 8.20 to 8.22 and Tables 8.23 to 8.25 show the individuals that yield the best fitness scores for `lionhalf` and `lion` respectively. For each target semantics, we present the best individual for each weighting scheme used, showing its fitness score, raw surface text, and the mapping of S_{target} to $S_{candidate}$ as computed by our mapping algorithm in Section 6.4.2. Just as the formatted surface texts in the first stage show the alignment obtained from the edit distance algorithm, which forms the basis of our metre evaluation function, this mapping shows the alignment between the target and candidate semantics, which forms the basis of our semantic similarity equation.

For each individual, we show the set of proper matches, the set of dangling (i.e. unmapped) target literals, and the set of dangling (i.e. unmapped) candidate literals. Recall from Section 6.4.2 that a proper match is a pair (x,y) where x is a target literal and y is a candidate literal.

	Elitist = 20%	Elitist = 40%
Score	0.81	0.99
Surface	*a african lion , it dwells in her in a waste . its big head , it has its very small waist .*	*a african lion , who has a big head , , it dwells in a waste , that is it , . he is its very small waist .*
Proper match	$\{(lion(\_0,l),lion(\_17,\_11)),$ $(african(\_1,l),african(\_18,\_11)),$ $(dwell(d,l),dwell(\_10,\_11)),$ $(waste(\_3,was),waste(\_14,\_13)),$ $(inside(\_2,d,was),inside(\_12,\_10,\_13)),$ $(head(\_5,h),head(\_26,\_20)),$ $(own(\_4,l,h),own(\_28,\_11,\_20)),$ $(own(\_7,l,wai),own(\_23,\_11,\_21)),$ $(big(\_6,h),big(\_27,\_20)),$ $(small(s,wai),small(\_24,\_21)),$ $(waist(\_8,wai),waist(\_22,\_21)),$ $(very(\_9,s),very(\_25,\_24))\}$	$\{(inside(\_2,d,was),inside(\_37,\_29,\_38)),$ $(lion(\_0,l),lion(\_31,\_30)),$ $(african(\_1,l),african(\_32,\_30)),$ $(dwell(d,l),dwell(\_29,\_30)),$ $(waste(\_3,was),waste(\_39,\_38)),$ $(head(\_5,h),head(\_35,\_34)),$ $(own(\_4,l,h),own(\_33,\_30,\_34)),$ $(own(\_7,l,wai),own(\_42,\_30,\_41)),$ $(big(\_6,h),big(\_36,\_34)),$ $(small(s,wai),small(\_43,\_41)),$ $(waist(\_8,wai),waist(\_40,\_41)),$ $(very(\_9,s),very(\_44,\_43))\}$
Target dangling	$\emptyset$	$\emptyset$
Candidate dangling	$\{inside(\_15,\_10,\_16),own(\_19,\_20,\_21)\}$	$\emptyset$

Table 8.20: Best individual for weighting 1, S_{target} =`lionhalf`, initial tactical NLG test

Note that given the different emphasis of terms in the similarity equation, two fitness scores obtained under different weighting schemes are not comparable. However, by comparing the obtained mappings, we can observe the relative semantic qualities of these individuals across different weightings.

Observations and discussion

The conveying of `lionhalf` is a much easier task than the conveying of `lion` (see the values in Tables D.8 and D.9). For all the weighting schemes used, the EA achieves near optimal solutions for `lionhalf`, with the test for weighting scheme 1 and an elitist ratio of 40% yielding the highest score. However, we will have to scrutinize the individuals themselves to find out whether our semantic similarity equation indeed reflects the faithfulness of the texts. The maximum values for `lion`, however, are considerably lower, in particular weighting scheme 1. Again, we must examine the individuals that yield these scores.

The first observation one can make by briefly observing the individuals in Tables 8.20 to 8.25 are that that are all roughly ‘about’ the concepts within the target semantics `lionhalf` and

	Elitist = 20%	Elitist = 40%
Score	0.93	0.93
Surface	*a african lion , it dwells in a waste . its big head , that is him , , it has its very small waist .*	*a african lion , who has a very small waist , , it dwells in a waste in its big head , that is her , .*
Proper match	$\{(inside(\_2,d,was),inside(\_47,\_45,\_48)),$ $(lion(\_0,l),lion(\_50,\_46)),$ $(african(\_1,l),african(\_51,\_46)),$ $(dwell(d,l),dwell(\_45,\_46)),$ $(waste(\_3,was),waste(\_49,\_48)),$ $(head(\_5,h),head(\_59,\_53)),$ $(own(\_4,l,h),own(\_61,\_46,\_53)),$ $(own(\_7,l,wai),own(\_56,\_46,\_54)),$ $(big(\_6,h),big(\_60,\_53)),$ $(small(s,wai),small(\_57,\_54)),$ $(waist(\_8,wai),waist(\_55,\_54)),$ $(very(\_9,s),very(\_58,\_57))\}$	$\{(lion(\_0,l),lion(\_64,\_63)),$ $(african(\_1,l),african(\_65,\_63)),$ $(dwell(d,l),dwell(\_62,\_63)),$ $(waste(\_3,was),waste(\_71,\_72)),$ $(inside(\_2,d,was),inside(\_73,\_62,\_72)),$ $(head(\_5,h),head(\_76,\_75)),$ $(own(\_4,l,h),own(\_78,\_63,\_75)),$ $(own(\_7,l,wai),own(\_66,\_63,\_67)),$ $(big(\_6,h),big(\_77,\_75)),$ $(small(s,wai),small(\_69,\_67)),$ $(waist(\_8,wai),waist(\_68,\_67)),$ $(very(\_9,s),very(\_70,\_69))\}$
Target dangling	$\emptyset$	$\emptyset$
Candidate dangling	$\{own(\_52,\_53,\_54)\}$	$\{inside(\_74,\_62,\_75)\}$

Table 8.21: Best individual for weighting 2, $S_{target} =$`lionhalf`, initial tactical NLG test

	Elitist = 20%	Elitist = 40%
Score	0.90	0.93
Surface	*the african lion , it dwells in a waste . a big head , it is its expense in a very small man , who is its waist , .*	*a african lion , it dwells in a waste . he is its big head with its very small waist .*
Proper match	$\{(lion(\_0,l),lion(\_96,\_86)),$ $(african(\_1,l),african(\_97,\_86)),$ $(dwell(d,l),dwell(\_92,\_86)),$ $(inside(\_2,d,was),inside(\_95,\_92,\_94)),$ $(waste(\_3,was),waste(\_93,\_94)),$ $(head(\_5,h),head(\_90,\_80)),$ $(own(\_4,l,h),own(\_89,\_86,\_80)),$ $(own(\_7,l,wai),own(\_85,\_86,\_82)),$ $(big(\_6,h),big(\_91,\_80)),$ $(small(s,wai),small(\_87,\_82)),$ $(waist(\_8,wai),waist(\_84,\_82)),$ $(very(\_9,s),very(\_88,\_87))\}$	$\{(inside(\_2,d,was),inside(\_112,\_109,\_111)),$ $(lion(\_0,l),lion(\_113,\_101)),$ $(african(\_1,l),african(\_114,\_101)),$ $(dwell(d,l),dwell(\_109,\_101)),$ $(waste(\_3,was),waste(\_110,\_111)),$ $(head(\_5,h),head(\_98,\_99)),$ $(own(\_4,l,h),own(\_100,\_101,\_99)),$ $(own(\_7,l,wai),own(\_106,\_101,\_104)),$ $(big(\_6,h),big(\_102,\_99)),$ $(small(s,wai),small(\_107,\_104)),$ $(waist(\_8,wai),waist(\_105,\_104)),$ $(very(\_9,s),very(\_108,\_107))\}$
Target dangling	$\emptyset$	$\emptyset$
Candidate dangling	$\{expense(\_79,\_80),inside(\_81,\_79,\_82),$ $man(\_83,\_82)\}$	$\{with(\_103,\_98,\_104)\}$

Table 8.22: Best individual for weighting 3, $S_{target} =$`lionhalf`, initial tactical NLG test

	Elitist = 20%	Elitist = 40%
Score	0.55	0.60
Surface	*he will not play . a african lion , who has stark shoulders , , it dwells in a waste .*	*with a african lion , who dwells in her , , a boy will not play . its stark shoulders , they are them .*
Proper match	$\{(inside(\_2,d,was),inside(\_35,\_25,\_28)),$ $(lion(\_0,l),lion(\_29,\_26)),$ $(african(\_1,l),african(\_34,\_26)),$ $(dwell(d,l),dwell(\_25,\_26)),$ $(waste(\_3,was),waste(\_27,\_28)),$ $(shoulders(\_11,sho),shoulders(\_32,\_31)),$ $(own(\_10,l,sho),own(\_30,\_26,\_31)),$ $(stark(\_12,sho),stark(\_33,\_31)),$ $(play(pl,c),play(\_21,\_22)),$ $(will(wpl,pl),will(\_23,\_21)),$ $(not(\_20,wpl),not(\_24,\_23)),$	$\{(inside(\_2,d,was),inside(\_45,\_44,\_46)),$ $(with(\_19,pl,l),with(\_41,\_36,\_42)),$ $(lion(\_0,l),lion(\_43,\_42)),$ $(african(\_1,l),african(\_47,\_42)),$ $(dwell(d,l),dwell(\_44,\_42)),$ $(shoulders(\_11,sho),shoulders(\_48,\_49)),$ $(own(\_10,l,sho),own(\_51,\_42,\_49)),$ $(stark(\_12,sho),stark(\_50,\_49)),$ $(boy(\_16,c),boy(\_38,\_37)),$ $(play(pl,c),play(\_36,\_37)),$ $(will(wpl,pl),will(\_39,\_36))\}$ $(not(\_20,wpl),not(\_40,\_39))\}$
Target dangling	$\{own(\_4,l,h),head(\_5,h),$ $big(\_6,h),own(\_7,l,wai),$ $small(s,wai),waist(\_8,wai),$ $very(\_9,s),own(\_13,l,ja),$ $jaws(\_14,ja),grim(\_15,ja),$ $boy(\_16,c),good(\_17,c),$ $little(\_18,c),with(\_19,pl,l)\}$	$\{waste(\_3,was),own(\_4,l,h),$ $head(\_5,h),big(\_6,h),$ $own(\_7,l,wai),small(s,wai),$ $waist(\_8,wai),very(\_9,s),$ $own(\_13,l,ja),jaws(\_14,ja),$ $grim(\_15,ja),good(\_17,c),$ $little(\_18,c)\}$
Candidate dangling	$\emptyset$	$\emptyset$

Table 8.23: Best individual for weighting 1, S_{target} =`lion`, initial tactical NLG test

	Elitist = 20%	Elitist = 40%
Score	0.82	0.83
Surface	*they play . a african lion dwells in a waste , that a little boy has , . its stark shoulders , they will not be pretty . a good boy dwells with it in its grim jaws , that it has , . a big waste , that it has , has its very small waist , that has its head , .*	*they play . a african lion dwells in a waste , that is with its big head , in its very small waist . its shoulders are stark . a waste , that is with a little waist , will not be good . with its grim jaws , a boy dwells in a waist .*
Proper match	$\{(with(\_19,pl,l),with(\_75,\_73,\_53)),$ $(inside(\_2,d,was),inside(\_62,\_52,\_57)),$ $(lion(\_0,l),lion(\_54,\_53)),$ $(african(\_1,l),african(\_55,\_53)),$ $(dwell(d,l),dwell(\_52,\_53)),$ $(waste(\_3,was),waste(\_56,\_57)),$ $(head(\_5,h),head(\_93,\_92)),$ $(own(\_4,l,h),own(\_94,\_53,\_92)),$ $(small(s,wai),small(\_89,\_86)),$ $(own(\_7,l,wai),own(\_88,\_53,\_86)),$ $(waist(\_8,wai),waist(\_87,\_86)),$ $(very(\_9,s),very(\_90,\_89)),$ $(shoulders(\_11,sho),shoulders(\_70,\_66)),$ $(own(\_10,l,sho),own(\_72,\_53,\_66)),$ $(stark(\_12,sho),stark(\_71,\_66)),$ $(jaws(\_14,ja),jaws(\_80,\_79)),$ $(grim(\_15,ja),grim(\_82,\_79)),$ $(good(\_17,c),good(\_77,\_74)),$ $(boy(\_16,c),boy(\_76,\_74)),$ $(not(\_20,wpl),not(\_69,\_67)),$ $(own(\_13,l,ja),own(\_81,\_53,\_79))\}$	$\{(lion(\_0,l),lion(\_128,\_118)),$ $(african(\_1,l),african(\_129,\_118)),$ $(dwell(d,l),dwell(\_120,\_118)),$ $(head(\_5,h),head(\_125,\_124)),$ $(own(\_4,l,h),own(\_126,\_118,\_124)),$ $(big(\_6,h),big(\_127,\_124)),$ $(small(s,wai),small(\_134,\_131)),$ $(inside(\_2,d,was),inside(\_136,\_120,\_122)),$ $(own(\_7,l,wai),own(\_133,\_118,\_131)),$ $(waste(\_3,was),waste(\_121,\_122)),$ $(waist(\_8,wai),waist(\_132,\_131)),$ $(very(\_9,s),very(\_135,\_134)),$ $(shoulders(\_11,sho),shoulders(\_141,\_140)),$ $(own(\_10,l,sho),own(\_142,\_118,\_140)),$ $(own(\_13,l,ja),own(\_117,\_118,\_114)),$ $(stark(\_12,sho),stark(\_139,\_140)),$ $(jaws(\_14,ja),jaws(\_115,\_114)),$ $(grim(\_15,ja),grim(\_116,\_114)),$ $(boy(\_16,c),boy(\_119,\_109)),$ $(will(wpl,pl),will(\_105,\_106)),$ $(not(\_20,wpl),not(\_107,\_105))\}$
Target dangling	$\{big(\_6,h),little(\_18,c),$ $play(pl,c),will(wpl,pl)\}$	$\{good(\_17,c),little(\_18,c),$ $play(pl,c),with(\_19,pl,l)\}$
Candidate dangling	$\{own(\_58,\_59,\_57),boy(\_60,\_59),$ $little(\_61,\_59),play(\_63,\_64),$ $pretty(\_65,\_66),will(\_67,\_68),$ $dwell(\_73,\_74),inside(\_78,\_73,\_79),$ $own(\_83,\_53,\_79),own(\_84,\_85,\_86),$ $own(\_91,\_86,\_92),waste(\_95,\_85),$ $big(\_96,\_85),own(\_97,\_53,\_85)\}$	$\{good(\_98,\_99),waste(\_100,\_99),$ $with(\_101,\_99,\_102),waist(\_103,\_102),$ $little(\_104,\_102),dwell(\_108,\_109),$ $waist(\_110,\_111),inside(\_112,\_108,\_111),$ $with(\_113,\_108,\_114),with(\_123,\_122,\_124),$ $inside(\_130,\_120,\_131),play(\_137,\_138)\}$

Table 8.24: Best individual for weighting 2, $S_{target} = \texttt{lion}$, initial tactical NLG test

Score	0.73	0.73
Surface	*his men play . a african frog , who his grim jaws in his stark shoulders , that are with them , will not boil , dwells in a grim waste . his very small waist , it has his big head , that has a very stark bean in a good tiger , who is a little boy , , .*	*the jaws are the jaws . with a good boy , a african child dwells in his big head in a waste , that is it , . with his very small waist , the big shoulders , they will not boil his stark shoulders with him with his grim shoulders , that are little , . the men play .*
Proper match	$\{(african(\_1,l),african(\_170,\_160)),$ $(dwell(d,l),dwell(\_165,\_160)),$ $(inside(\_2,d,was),inside(\_184,\_165,\_167)),$ $(waste(\_3,was),waste(\_166,\_167)),$ $(head(\_5,h),head(\_146,\_145)),$ $(own(\_4,l,h),own(\_159,\_160,\_145)),$ $(big(\_6,h),big(\_147,\_145)),$ $(small(s,wai),small(\_163,\_144)),$ $(own(\_7,l,wai),own(\_162,\_160,\_144)),$ $(waist(\_8,wai),waist(\_161,\_144)),$ $(very(\_9,s),very(\_164,\_163)),$ $(shoulders(\_11,sho),shoulders(\_178,\_177)),$ $(own(\_10,l,sho),own(\_179,\_160,\_177)),$ $(own(\_13,l,ja),own(\_174,\_160,\_172)),$ $(stark(\_12,sho),stark(\_181,\_177)),$ $(jaws(\_14,ja),jaws(\_173,\_172)),$ $(grim(\_15,ja),grim(\_175,\_172)),$ $(boy(\_16,c),boy(\_156,\_154)),$ $(good(\_17,c),good(\_158,\_154)),$ $(little(\_18,c),little(\_157,\_154)),$ $(will(wpl,pl),will(\_182,\_171)),$ $(not(\_20,wpl),not(\_183,\_182))\}$	$\{(african(\_1,l),african(\_202,\_197)),$ $(with(\_19,pl,l),with(\_226,\_212,\_197)),$ $(dwell(d,l),dwell(\_196,\_197)),$ $(waste(\_3,was),waste(\_200,\_199)),$ $(inside(\_2,d,was),inside(\_198,\_196,\_199)),$ $(head(\_5,h),head(\_207,\_208)),$ $(own(\_4,l,h),own(\_210,\_197,\_208)),$ $(big(\_6,h),big(\_209,\_208)),$ $(small(s,wai),small(\_224,\_221)),$ $(own(\_7,l,wai),own(\_223,\_197,\_221)),$ $(waist(\_8,wai),waist(\_222,\_221)),$ $(very(\_9,s),very(\_225,\_224)),$ $(stark(\_12,sho),stark(\_218,\_214)),$ $(own(\_10,l,sho),own(\_219,\_197,\_214)),$ $(own(\_13,l,ja),own(\_232,\_197,\_228)),$ $(shoulders(\_11,sho),shoulders(\_217,\_214)),$ $(grim(\_15,ja),grim(\_230,\_228)),$ $(boy(\_16,c),boy(\_205,\_204)),$ $(good(\_17,c),good(\_206,\_204)),$ $(will(wpl,pl),will(\_233,\_212)),$ $(not(\_20,wpl),not(\_234,\_233))\}$
Target dangling	$\{lion(\_0,l),play(pl,c),$ $with(\_19,pl,l)\}$	$\{lion(\_0,l),jaws(\_14,ja),$ $little(\_18,c),play(pl,c)\}$
Candidate dangling	$\{own(\_143,\_144,\_145),own(\_148,\_145,\_149),$ $bean(\_150,\_149),stark(\_151,\_149),$ $very(\_152,\_151),inside(\_153,\_149,\_154),$ $tiger(\_155,\_154),grim(\_168,\_167),$ $frog(\_169,\_160),boil(\_171,\_172,\_160),$ $inside(\_176,\_172,\_177),with(\_180,\_177,\_177),$ $play(\_185,\_186),men(\_187,\_186),$ $own(\_188,\_189,\_186)\}$	$\{play(\_190,\_191),men(\_192,\_191),$ $jaws(\_193,\_194),jaws(\_195,\_194),$ $child(\_201,\_197),with(\_203,\_196,\_204),$ $inside(\_211,\_196,\_208),boil(\_212,\_213,\_214),$ $shoulders(\_215,\_213),big(\_216,\_213),$ $with(\_220,\_212,\_221),with(\_227,\_212,\_228),$ $shoulders(\_229,\_228),little(\_231,\_228)\}$

Table 8.25: Best individual for weighting 3, $S_{target} = \texttt{lion}$, initial tactical NLG test

lion. Compare these texts with those obtained from the tests during the first stage, in particular Section 8.3.3, where the same baseline operators were used. However, this in itself is not an impressive feat, as it could easily be achieved by only selecting appropriate lexical items prior to the search process (a method used in Eddy et al. (2001)). Nevertheless, it shows the definite effect on the behaviour of the EA by changing the evaluation function from metre similarity to semantic similarity.

Observing the individual in Table 8.20 for an elitist ratio of 20%, we see that there are no target dangling literals, as the mapping algorithm computes a structurally consistent mapping that maps the entire set of literals in lionhalf. In a sense, this indicates success in conveying S_{target}. However, if we observe the text itself, there are two peculiarities. Firstly, the verb *'dwells'* has two prepositional phrases attached to it: *"in her"* and *"in a waste"*. Secondly, the last sentence is in fact conveying that the *"big head"* has the *"very small waist"*, which is certainly not a fact conveyed by lionhalf. However, the genitive determiners *"its"* for both the subject and object of this sentence are resolved to *"lion"*, and thus they introduce the two *own* literals that are correctly mapped to the two *own* literals in S_{target}. We can see that there are two dangling candidate literals, and they correspond to these two peculiarities: $inside(\_15, \_10, \_16)$ is introduced by the spurious prepositional phrase *"in her"*, and $own(\_19, \_20, \_21)$ is introduced by the transitive verb *'has'* in the last sentence. Note that the variables $\_20$ and $\_21$ are bound to *h* and *wai*, which represent the head and the waist. Thus we can see that the mapping is justified: the candidate semantics obtained from the composition of lexical semantics does indeed convey lionhalf, and it correctly identifies the extraneous semantics as dangling candidate literals, hence the non-optimal score of 0.81.

The other individual in Table 8.20, i.e. the one obtained for an elitist ratio of 40%, yields a fitness score of 0.99, and for all intents and purposes is an optimal solution, i.e. it fails to be 1.0 due to a rounding error. Both the target and candidate semantics are completely mapped in a structurally consistent manner, and there are no dangling literals. However, if we observe the text itself, there are still some slight peculiarities. Firstly, the relative clause attached to *"a waste"*, i.e.*"that is it"*, and secondly, the second sentence that is a copula verb construction, *"He is its very small waist"*. Once again, the genitive *'its'* correctly resolves to *'lion'*, and thus the candidate semantics establish that the very small waist belongs to the lion. Note that these peculiarities, i.e. the relative clause *"that is it"* and the copula verb construction *"He is"*, do not introduce any semantics, and thus have no bearing on the semantic evaluation function.

Both the individuals in Table 8.21 behave similarly: some sentences convey slightly different meanings, which are recorded as dangling candidate literals, but the genitive *"its"* establishes the correct meaning. This is reflected in the fitness scores.

The individual in Table 8.22 for elitist ratio 40% also behaves similarly. However, the individual for elitist ratio 20% exhibits a more severe case of the peculiarities we have observed. Although the mapping algorithm correctly finds a structurally consistent mapping for all of the literals in S_{target}, the candidate semantics derived from the composition of lexical semantics, particularly for the last sentence, is slightly awkward. Note that the copula verb *'is'* simply unifies the entity variables of its subject and object. Thus, the phrases *"a big head, it is its expense"* and *"a very small man, who is its waist"*, equate the entities of *head* and *expense*, and of *man* and *waist*. Furthermore, the genitives in these phrases resolve ownership to *'lion'*, and thus introduce *own* literals that are used in the mapping. Finally, since *man* and *waist* are deemed to be the same, the mapping to the target semantics representing the noun phrase *"very small waist"* is structurally consistent. Thus, while we can see that the candidate semantics are indeed correct, the texts that convey them are awkward. This awkwardness does not originate from the semantic evaluation function, but rather arises from a rather unfortunate exploitation of our handcrafted grammar. Unfortunately, this awkwardness affects subsequent tests as well, as we have chosen to use the same grammar throughout our empirical study for consistency of results. While it does not affect the correct scoring of texts that indeed explicitly convey the target semantics, it skews the fitness scores favorably for individuals where the meaning is rather unintentionally conveyed. Thus, we urge the reader to bear this in mind when observing the individuals.

As for the individuals obtained for $S_{target} = \texttt{lion}$ (Tables 8.23 to 8.25), we can see a clear difference in the behaviour of the EA under different weighting schemes, particularly between weighting scheme 1 and the other two schemes. The individuals in Table 8.23 yield relatively low fitness scores of 0.55 and 0.60, and we can see this is due to the fact that the candidate semantics are only conveying a subset of S_{target}: there are 14 dangling target literals for the individual obtained with elitist ratio 20%, and 13 for elitist ratio 40%. However, there are no dangling candidate literals for either individual. Contrast this with the individuals obtained under weighting schemes 2 and 3, in Tables 8.24 and 8.25. For these individuals, there are only four dangling target literals. In a sense, they can be considered to be better than those obtained under weighting scheme 1, as they convey more of the target semantics. However, there is a tradeoff in that there are many dangling candidate literals, which represent extraneous semantics conveyed by the texts. Coupled with the problem caused by the grammar described above, they cause the target semantics to be obscured in the surface text. Note that although weighting schemes 2 and 3 exhibit similar results, they are obtained for different reasons. Weighting scheme 2 emphasizes raw conceptual similarity, whereas scheme 3 emphasizes structural similarity. This is reflected by the dangling candidate literals. In Table 8.24, they are mainly literals that are members of S_{target}, whereas in Table 8.24 they are mainly different literals altogether: $bean, tiger, frog, boil,$ etc.

Unfortunately, in seeking to emphasize structural similarity, we only increased α_2 and α_3 for weighting scheme 3, when it is evident from these results that α_4, which penalizes dangling literals, should be increased as well.

8.4.2 Smart operators test

In this test, we will examine the effect of using the semantically motivated operators presented in Section 7.3. We will employ the compound operators that simultaneously ensure complete derivations while greedily attempting to consume semantics (Section 7.4.2). Thus, the mutation operators used in this test, along with their probabilities of being applied, are as follows:

- SMARTCREATECOMPLETE = 0.5
- SMARTADJOINCOMPLETE = 0.3
- SMARTDELETECOMPLETE = 0.2

Additionally, when crossover is called for, the subtree swapping operator SMARTSWAP is used. As the three mutation operators above guarantee complete derivations, SMARTSWAP guarantees complete derivations as well.

All other aspects of the test, i.e. target semantics, EA parameters, linguistic resources, evaluation function and weighting schemes, are the same as during the initial test in Section 8.4.1.

Expectations

We hope that the semantically smart operators used in this test will enable the EA to obtain individuals that successfully convey the target semantics, and at the very least show a marked improvement over the results obtained in the previous test. In the mappings shown for the

	Elitist = 20%	Elitist = 40%
Score	0.93	1.00
	a african lion , who has its very small waist , dwells in a waste . its head is big .	*a african lion , who has a very small waist , dwells in a waste . its head , it is big .*
Proper match	$\{(inside(\_2,d,was),inside(\_12,\_10,\_13)),$ $(lion(\_0,l),lion(\_15,\_11)),$ $(african(\_1,l),african(\_16,\_11)),$ $(dwell(d,l),dwell(\_10,\_11)),$ $(waste(\_3,was),waste(\_14,\_13)),$ $(head(\_5,h),head(\_25,\_24)),$ $(own(\_4,l,h),own(\_26,\_11,\_24)),$ $(big(\_6,h),big(\_23,\_24)),$ $(small(s,wai),small(\_21,\_18)),$ $(waist(\_8,wai),waist(\_19,\_18)),$ $(very(\_9,s),very(\_22,\_21)),$ $(own(\_7,l,wai),own(\_17,\_11,\_18))\}$	$\{(inside(\_2,d,was),inside(\_33,\_32,\_34)),$ $(lion(\_0,l),lion(\_36,\_31)),$ $(african(\_1,l),african(\_42,\_31)),$ $(dwell(d,l),dwell(\_32,\_31)),$ $(waste(\_3,was),waste(\_35,\_34)),$ $(head(\_5,h),head(\_29,\_28)),$ $(own(\_4,l,h),own(\_30,\_31,\_28)),$ $(own(\_7,l,wai),own(\_37,\_31,\_38)),$ $(big(\_6,h),big(\_27,\_28)),$ $(small(s,wai),small(\_40,\_38)),$ $(waist(\_8,wai),waist(\_39,\_38)),$ $(very(\_9,s),very(\_41,\_40))\}$
Target dangling	$\emptyset$	$\emptyset$
Candidate dangling	$\{own(\_20,\_11,\_18)\}$	$\emptyset$

Table 8.26: Best individual for weighting 1, S_{target} = `lionhalf`, smart operators test

obtained individuals, there should not be any dangling target literals. Moreover, an optimal solution should also not have any dangling candidate literals. Again, we expect that the task of conveying `lionhalf` should be easier than that of conveying `lion` given its reduced complexity.

Results

As in the previous section, we conduct twelve separate tests: S_{target} = `lionhalf` with weighting schemes 1, 2, and 3, and S_{target} = `lion` with weighting schemes 1, 2, and 3, all of which are run with elitist ratios of 20% and 40%. Summary statistics and graphs for this test are presented in Section D.2.2 of Appendix D.

Tables 8.26 to 8.28 and Tables 8.29 to 8.31 show the individuals that yield the best fitness scores for `lionhalf` and `lion` respectively. For each target semantics, we present the best individual for each weighting scheme used, showing its fitness score, raw surface text, and the mapping of S_{target} to $S_{candidate}$. See the initial test (Section 8.4.1) for a description of this mapping.

	Elitist = 20%	Elitist = 40%
Score	0.99	0.99
Surface	*a african lion , who has a very small waist , , it dwells in a waste . its head , it is big .*	*a african lion , who has a big head , , it dwells in a waste . its waist , it is very small .*
Proper match	$\{(inside(\_2,d,was),inside(\_45,\_43,\_46)),$ $(lion(\_0,l),lion(\_48,\_44)),$ $(african(\_1,l),african(\_49,\_44)),$ $(dwell(d,l),dwell(\_43,\_44)),$ $(waste(\_3,was),waste(\_47,\_46)),$ $(head(\_5,h),head(\_57,\_56)),$ $(own(\_4,l,h),own(\_58,\_44,\_56)),$ $(own(\_7,l,wai),own(\_50,\_44,\_51)),$ $(big(\_6,h),big(\_55,\_56)),$ $(small(s,wai),small(\_53,\_51)),$ $(waist(\_8,wai),waist(\_52,\_51)),$ $(very(\_9,s),very(\_54,\_53))\}$	$\{(inside(\_2,d,was),inside(\_61,\_59,\_62)),$ $(lion(\_0,l),lion(\_64,\_60)),$ $(african(\_1,l),african(\_69,\_60)),$ $(dwell(d,l),dwell(\_59,\_60)),$ $(waste(\_3,was),waste(\_63,\_62)),$ $(head(\_5,h),head(\_67,\_66)),$ $(own(\_4,l,h),own(\_65,\_60,\_66)),$ $(own(\_7,l,wai),own(\_74,\_60,\_71)),$ $(big(\_6,h),big(\_68,\_66)),$ $(small(s,wai),small(\_70,\_71)),$ $(waist(\_8,wai),waist(\_73,\_71)),$ $(very(\_9,s),very(\_72,\_70))\}$
Target dangling	$\emptyset$	$\emptyset$
Candidate dangling	$\emptyset$	$\emptyset$

Table 8.27: Best individual for weighting 2, S_{target} = `lionhalf`, smart operators test

	Elitist = 20%	Elitist = 40%
Score	1.00	1.00
Surface	*a african lion , who has a very small waist , , it dwells in a waste . its head , it is big .*	*a african lion , who has a very small waist , , it dwells in a waste , that is him , . its head , it is big .*
Proper match	$\{(inside(\_2,d,was),inside(\_90,\_80,\_89)),$ $(lion(\_0,l),lion(\_81,\_79)),$ $(african(\_1,l),african(\_87,\_79)),$ $(dwell(d,l),dwell(\_80,\_79)),$ $(waste(\_3,was),waste(\_88,\_89)),$ $(head(\_5,h),head(\_77,\_76)),$ $(own(\_4,l,h),own(\_78,\_79,\_76)),$ $(own(\_7,l,wai),own(\_82,\_79,\_83)),$ $(big(\_6,h),big(\_75,\_76)),$ $(small(s,wai),small(\_85,\_83)),$ $(waist(\_8,wai),waist(\_84,\_83)),$ $(very(\_9,s),very(\_86,\_85))\}$	$\{(inside(\_2,d,was),inside(\_97,\_96,\_98)),$ $(lion(\_0,l),lion(\_99,\_95)),$ $(african(\_1,l),african(\_105,\_95)),$ $(dwell(d,l),dwell(\_96,\_95)),$ $(waste(\_3,was),waste(\_106,\_98)),$ $(head(\_5,h),head(\_93,\_92)),$ $(own(\_4,l,h),own(\_94,\_95,\_92)),$ $(own(\_7,l,wai),own(\_100,\_95,\_101)),$ $(big(\_6,h),big(\_91,\_92)),$ $(small(s,wai),small(\_103,\_101)),$ $(waist(\_8,wai),waist(\_102,\_101)),$ $(very(\_9,s),very(\_104,\_103))\}$
Target dangling	$\emptyset$	$\emptyset$
Candidate dangling	$\emptyset$	$\emptyset$

Table 8.28: Best individual for weighting 3, S_{target} = `lionhalf`, smart operators test

	Elitist = 20%	Elitist = 40%
Score	0.54	0.54
Surface	*a boy , who is little , will not be good . a african lion , who has a very small waist , dwells in a waste .*	*a african lion , who has a very small waist , dwells in a waste . its shoulders will not be stark .*
Proper match	$\{(inside(\_2,d,was),inside(\_23,\_21,\_24)),$ $(lion(\_0,l),lion(\_26,\_22)),$ $(african(\_1,l),african(\_27,\_22)),$ $(dwell(d,l),dwell(\_21,\_22)),$ $(waste(\_3,was),waste(\_25,\_24)),$ $(small(s,wai),small(\_31,\_29)),$ $(own(\_7,l,wai),own(\_28,\_22,\_29)),$ $(waist(\_8,wai),waist(\_30,\_29)),$ $(very(\_9,s),very(\_32,\_31)),$ $(boy(\_16,c),boy(\_35,\_34)),$ $(good(\_17,c),good(\_33,\_34)),$ $(little(\_18,c),little(\_36,\_34)),$ $(will(wpl,pl),will(\_37,\_38)),$ $(not(\_20,wpl),not(\_39,\_37))\}$	$\{(inside(\_2,d,was),inside(\_56,\_48,\_57)),$ $(lion(\_0,l),lion(\_49,\_44)),$ $(african(\_1,l),african(\_50,\_44)),$ $(dwell(d,l),dwell(\_48,\_44)),$ $(waste(\_3,was),waste(\_58,\_57)),$ $(small(s,wai),small(\_54,\_52)),$ $(own(\_7,l,wai),own(\_51,\_44,\_52)),$ $(waist(\_8,wai),waist(\_53,\_52)),$ $(very(\_9,s),very(\_55,\_54)),$ $(shoulders(\_11,sho),shoulders(\_42,\_41)),$ $(own(\_10,l,sho),own(\_43,\_44,\_41)),$ $(stark(\_12,sho),stark(\_40,\_41)),$ $(will(wpl,pl),will(\_45,\_46)),$ $(not(\_20,wpl),not(\_47,\_45))\}$
Target dangling	$\{own(\_4,l,h),head(\_5,h),$ $big(\_6,h),own(\_10,l,sho),$ $shoulders(\_11,sho),stark(\_12,sho),$ $own(\_13,l,ja),jaws(\_14,ja),$ $grim(\_15,ja),play(pl,c),$ $with(\_19,pl,l)\}$	$\{own(\_4,l,h),head(\_5,h),$ $big(\_6,h),own(\_13,l,ja),$ $jaws(\_14,ja),grim(\_15,ja),$ $boy(\_16,c),good(\_17,c),$ $little(\_18,c),play(pl,c),$ $with(\_19,pl,l)\}$
Candidate dangling	$\emptyset$	$\emptyset$

Table 8.29: Best individual for weighting 1, S_{target} = `lion`, smart operators test

	Elitist = 20%	Elitist = 40%
Score	0.88	0.61
Surface	*they play . in a good boy , who is little , , a african lion , who with jaws , that are in its stark shoulders , , has a very small waist , dwells in a waste . with a boy , who has its grim jaws , , its head will not be big .*	*a african lion , who has a very small waist , dwells in a waste . its head will not be big .*
Proper match	$\{(lion(\_0,l),lion(\_71,\_62)),$ $(african(\_1,l),african(\_85,\_62)),$ $(dwell(d,l),dwell(\_61,\_62)),$ $(waste(\_3,was),waste(\_63,\_64)),$ $(inside(\_2,d,was),inside(\_65,\_61,\_64)),$ $(head(\_5,h),head(\_91,\_87)),$ $(own(\_4,l,h),own(\_92,\_62,\_87)),$ $(big(\_6,h),big(\_86,\_87)),$ $(small(s,wai),small(\_83,\_73)),$ $(own(\_7,l,wai),own(\_72,\_62,\_73)),$ $(waist(\_8,wai),waist(\_82,\_73)),$ $(very(\_9,s),very(\_84,\_83)),$ $(shoulders(\_11,sho),shoulders(\_79,\_78)),$ $(own(\_10,l,sho),own(\_80,\_62,\_78)),$ $(own(\_13,l,ja),own(\_99,\_62,\_97)),$ $(stark(\_12,sho),stark(\_81,\_78)),$ $(jaws(\_14,ja),jaws(\_98,\_97)),$ $(grim(\_15,ja),grim(\_100,\_97)),$ $(good(\_17,c),good(\_70,\_67)),$ $(boy(\_16,c),boy(\_68,\_67)),$ $(little(\_18,c),little(\_69,\_67)),$ $(will(wpl,pl),will(\_88,\_89)),$ $(not(\_20,wpl),not(\_90,\_88))\}$	$\{(inside(\_2,d,was),inside(\_110,\_109,\_111)),$ $(lion(\_0,l),lion(\_112,\_108)),$ $(african(\_1,l),african(\_113,\_108)),$ $(dwell(d,l),dwell(\_109,\_108)),$ $(waste(\_3,was),waste(\_119,\_111)),$ $(head(\_5,h),head(\_106,\_102)),$ $(own(\_4,l,h),own(\_107,\_108,\_102)),$ $(big(\_6,h),big(\_101,\_102)),$ $(small(s,wai),small(\_117,\_115)),$ $(own(\_7,l,wai),own(\_114,\_108,\_115)),$ $(waist(\_8,wai),waist(\_116,\_115)),$ $(very(\_9,s),very(\_118,\_117)),$ $(will(wpl,pl),will(\_103,\_104)),$ $(not(\_20,wpl),not(\_105,\_103))\}$
Target dangling	$\{play(pl,c),with(\_19,pl,l)\}$	$\{own(\_10,l,sho),shoulders(\_11,sho),$ $stark(\_12,sho),own(\_13,l,ja),$ $jaws(\_14,ja),grim(\_15,ja),$ $boy(\_16,c),good(\_17,c),$ $little(\_18,c),play(pl,c),$ $with(\_19,pl,l)\}$
Candidate dangling	$\{play(\_59,\_60),inside(\_66,\_61,\_67),$ $with(\_74,\_72,\_75),jaws(\_76,\_75),$ $inside(\_77,\_75,\_78),with(\_93,\_86,\_94),$ $boy(\_95,\_94),own(\_96,\_94,\_97)\}$	$\emptyset$

Table 8.30: Best individual for weighting 2, $S_{target} =$`lion`, smart operators test

	Elitist = 20%	Elitist = 40%
Score	0.79	0.66
Surface	*in a good boy , who is little , , a african lion , who has a waist , , it dwells in a waste . with a skin , that has its stark shoulders , , with her , a very small waist , it is its waist , that has its grim jaws , . its big child , she will not be a head . they play .*	*a very african lion , who with its small waist , has its stark shoulders with a tail , that has its grim jaws , , dwells in a waste . his head , that is its head , will not be big .*
Proper match	$\{(lion(\_0,l),lion(\_122,\_121)), (african(\_1,l),african(\_126,\_121)), (dwell(d,l),dwell(\_120,\_121)), (waste(\_3,was),waste(\_129,\_128)), (inside(\_2,d,was),inside(\_127,\_120,\_128)), (head(\_5,h),head(\_156,\_157)), (own(\_4,l,h),own(\_162,\_121,\_157)), (big(\_6,h),big(\_163,\_157)), (small(s,wai),small(\_139,\_136)), (own(\_7,l,wai),own(\_137,\_121,\_136)), (very(\_9,s),very(\_140,\_139)), (shoulders(\_11,sho),shoulders(\_148,\_147)), (own(\_10,l,sho),own(\_150,\_121,\_147)), (stark(\_12,sho),stark(\_149,\_147)), (jaws(\_14,ja),jaws(\_153,\_152)), (own(\_13,l,ja),own(\_154,\_121,\_152)), (grim(\_15,ja),grim(\_155,\_152)), (boy(\_16,c),boy(\_132,\_131)), (good(\_17,c),good(\_133,\_131)), (little(\_18,c),little(\_134,\_131)), (will(wpl,pl),will(\_158,\_159)), (not(\_20,wpl),not(\_160,\_158)), (waist(\_8,wai),waist(\_135,\_136))\}$	$\{(inside(\_2,d,was),inside(\_177,\_176,\_178)), (lion(\_0,l),lion(\_179,\_172)), (african(\_1,l),african(\_180,\_172)), (dwell(d,l),dwell(\_176,\_172)), (waste(\_3,was),waste(\_200,\_178)), (big(\_6,h),big(\_165,\_166)), (own(\_4,l,h),own(\_171,\_172,\_166)), (small(s,wai),small(\_199,\_196)), (own(\_7,l,wai),own(\_198,\_172,\_196)), (waist(\_8,wai),waist(\_197,\_196)), (shoulders(\_11,sho),shoulders(\_184,\_183)), (own(\_13,l,ja),own(\_194,\_172,\_191)), (stark(\_12,sho),stark(\_186,\_183)), (jaws(\_14,ja),jaws(\_192,\_191)), (grim(\_15,ja),grim(\_193,\_191)), (will(wpl,pl),will(\_173,\_174)), (not(\_20,wpl),not(\_175,\_173)), (own(\_10,l,sho),own(\_182,\_172,\_183)), (head(\_5,h),head(\_167,\_166))\}$
Target dangling	$\{play(pl,c),with(\_19,pl,l)\}$	$\{very(\_9,s),boy(\_16,c), good(\_17,c),little(\_18,c), play(pl,c),with(\_19,pl,l)\}$
Candidate dangling	$\{own(\_123,\_121,\_124),waist(\_125,\_124), inside(\_130,\_120,\_131),waist(\_138,\_136), with(\_141,\_135,\_142),with(\_143,\_135,\_144), skin(\_145,\_144),own(\_146,\_144,\_147), own(\_151,\_136,\_152),child(\_161,\_157), play(\_164,\_152)\}$	$\{own(\_168,\_169,\_166),head(\_170,\_166), very(\_181,\_180),own(\_185,\_172,\_183), with(\_187,\_182,\_188),tail(\_189,\_188), own(\_190,\_188,\_191),with(\_195,\_182,\_196)\}$

Table 8.31: Best individual for weighting 3, $S_{target} =$`lion`, smart operators test

Observations and discussion

The results obtained for the `lionhalf` target semantics show that the EA is succeeding in finding optimal solutions, i.e. texts that convey the semantics without dangling target or literal candidates. The only exception is the individual obtained under weighting scheme 1 for an elitist ratio of 20%, shown in Table 8.26, where there is one dangling candidate literal. Observing the surface text for this individual, we see that the dangling literal $own(\_20, \_11, \_18)$ is introduced due to the redundancy in the relative clause *"who has its very small waist"*, where both the verb *'has'* and the genitive *'its'* indicate ownership of the waist by the lion. Note that the candidate literal in the proper match that conveys this fact is $own(\_17, \_11, \_18)$, where the second and third arguments, i.e. the owner and ownee entity variables, are the same as in the dangling literal. They convey exactly the same fact, but unfortunately only one literal can be mapped by our mapping algorithm, and thus the other is forced to be a dangling literal. All the other individuals for this target semantics are optimal solutions. Note that one optimal solution for `lionhalf` was obtained in the initial test, i.e. in Table 8.20, and that the other individuals were also very close to being optimal solutions. However, by comparing Figure D.10 to Figure D.8 we can see that the semantically smart operators are helping the EA find these optimal solutions much more rapidly and consistently.

The results obtained for the `lion` target semantics, unfortunately, are not as we had hoped for. The individuals obtained by the EA under weighting scheme 1 (Table 8.29) show similar properties to those in the previous test (Table 8.23). Under this weighting scheme, the EA seems to behave very conservatively, in the sense that it considerably penalizes individuals with dangling candidate literals. Although this ensures that the candidate semantics of the solutions are subsets of S_{target}, which can be beneficial in instances where such faithfulness is required, we believe that it is also limiting the opportunity for individuals to expand their linguistic structure, which can ultimately convey more of the target semantics. Thus, it is something of a "needle in a haystack" evaluation function (Bäck et al., 1997).

The individual obtained under weighting scheme 2 for an elitist ratio of 20% (Table 8.30) shows the best solution obtained for the `lion` target form using the smart operators. Compared to the individual obtained with similar parameters in the previous test (Table 8.24), we can see that there are slightly fewer dangling target literals, but more importantly, considerably fewer dangling candidate literals.

Note, however, the relatively low fitness of the best individual obtained for $S_{target} =$ `lion` under weighting scheme 2 (Table 8.30) for an elitist ratio of 40%. This suggests that when using the semantically smart operators, the selection pressure when using an elitist ratio of 40% is too high, causing the EA to become trapped in a local maximum. In Table 8.31, i.e. under weighting scheme 3, we can see that an elitist ratio of 40% also performs poorly. Observing Figure D.11 we can see a significant difference between the two elitist ratios used, which is in contrast to Figure D.9, where the results for the two ratios are very similar.

8.4.3 PROTECTOR and SPUD-like operator test

Up to this point, McGonagall has managed to produce optimal solutions for the task of satisfying the `lionhalf` target semantics, but not `lion`. The best solution obtained is shown in Table 8.30, and it still contains several dangling target and candidate literals. In this test, we attempt to satisfy this target semantics by using the most explicitly goal-directed operators we have designed, i.e. those that are designed to simulate the mechanisms of the PROTECTOR and SPUD NLG systems (Section 7.4.3).

We run two different tests, one for each type of operator. For the PROTECTOR-like test, the mutation operators used, along with their probabilities of being applied, are as follows:

- PROTECTORLike = 0.6
- BlindDeleteComplete = 0.2
- SyntaxAddAdjoin = 0.2

We use the SyntaxAddAdjoin operator because otherwise, the EA will be unable to generate texts of more than one sentence.

For the SPUD-like test, the mutation operators used, along with their probabilities of being applied, are as follows:

- SPUDLikeDiscourse = 0.8
- BlindDelete = 0.2

We use the baseline blind delete operators because the structure-building operations in these compound operators are primarily semantically motivated, and thus the SmartDelete operator would find little opportunity to be applied.

Additionally, in both these tests, when crossover is called for, the subtree swapping operator SMARTSWAP is used.

Both these tests will use the `lion` target semantics, and we decide to only use weighting scheme 2 for the semantic evaluation function. The EA uses an elitist ratio of 20%. All other aspects are the same as during the previous test.

Expectations

We hope that these explicitly goal-directed operators will be able to produce optimal solutions for the task of conveying `lion`, i.e. in the mappings for the obtained individuals, there should not be any dangling target or candidate literals.

Results

The two tests for the PROTECTOR-like and SPUD-like operators were run ten times. Summary statistics and graphs for this test are presented in Section D.2.3 of Appendix D.

Tables 8.32 and 8.33 show the individuals that yield the best fitness scores for the tests using the PROTECTOR-like and SPUD-like operators respectively. For each individual we show its fitness score, raw surface text, and the mapping of S_{target} to $S_{candidate}$. See the initial test (Section 8.4.1) for a description of this mapping.

Observations and discussion

The results achieved by the EA using PROTECTORLIKE show that it too is unable to satisfy the `lion` target semantics. Furthermore, we can see the presence of an unrelated word in the text of the best individual obtained (Table 8.32), i.e.*'knees'*, despite the explicitly greedy semantic consumption of PROTECTORLIKE. This word is introduced by the SMARTDELETECOMPLETE operator, where due to the pressure of having to complete a derivation, it can fall back to the BLINDADDSUBSTITUTE operator if the more semantically motivated substitution operators fail to find a better alternative.

Recall from Section 7.4.3 that PROTECTORLIKE works by trying to cram in as much semantic content as possible using the SMARTADDMATCH operators, starting from one initial

Score	0.85
Surface	*he dwells* $\boxed{P}$ *it . a waste* $\boxed{Aux}$ *be her . a very small waist , that* $\boxed{Aux}$ *be small ,* $\boxed{Aux}$ *be her . knees are in it . a good boy , who* $\boxed{Aux}$ *be little , will not play with a african lion , who has a big head , . stark shoulders , that he has , are its stark shoulders , that he has , . it has grim jaws , that are grim , .*
Proper match	({inside(_2,d,was),inside(_57,_58,_59)),(with(_19,pl,l),with(_23,_21,_24)), (lion(_0,l),lion(_25,_24)),(african(_1,l),african(_30,_24)), (head(_5,h),head(_28,_27)),(own(_4,l,h),own(_26,_24,_27)), (big(_6,h),big(_29,_27)),(waist(_8,wai),waist(_53,_52)), (very(_9,s),very(_55,_54)),(small(s,wai),small(_54,_52)), (jaws(_14,ja),jaws(_46,_45)),(own(_10,l,sho),own(_38,_24,_37)), (own(_13,l,ja),own(_44,_24,_45)),(boy(_16,c),boy(_33,_22)), (good(_17,c),good(_34,_22)),(little(_18,c),little(_35,_22)), (play(pl,c),play(_21,_22)),(will(wpl,pl),will(_31,_21)), (not(_20,wpl),not(_32,_31)),(shoulders(_11,sho),shoulders(_36,_37)), (stark(_12,sho),stark(_42,_37)),(grim(_15,ja),grim(_47,_45))}
Target dangling	{dwell(d,l),waste(_3,was),own(_7,l,wai)}
Candidate dangling	{own(_39,_22,_37),shoulders(_40,_37),own(_41,_22,_37),stark(_43,_37), grim(_48,_45),dwell(_49,_50),waste(_51,_52),small(_56,_52), knees(_60,_58)}

Table 8.32: Best individual for PROTECTOR-like operator test

Score	0.93
Surface	$\boxed{D}$ *good boy ,* $\boxed{Comp}$ $\boxed{CV}$ *little , will not play with* $\boxed{D}$ *african lion ,* $\boxed{Comp}$ *dwells in* $\boxed{D}$ *waste , . its waist* $\boxed{Aux}$ *be very small .* $\boxed{D}$ *head* $\boxed{CV}$ *big . its jaws* $\boxed{CV}$ *grim .* $\boxed{NP}$ $\boxed{CV}$ *its shoulders* $\boxed{Punc}$ $\boxed{Comp}$ $\boxed{CV}$ *stark* $\boxed{Punc}$ *.*
Proper match	$(inside(\_2,d,was),inside(\_72,\_71,\_73)),(with(\_19,pl,l),with(\_68,\_61,\_69)),$ $(lion(\_0,l),lion(\_70,\_69)),(african(\_1,l),african(\_75,\_69)),$ $(dwell(d,l),dwell(\_71,\_69)),(waste(\_3,was),waste(\_74,\_73)),$ $(head(\_5,h),head(\_83,\_82)),(big(\_6,h),big(\_81,\_82)),$ $(small(s,wai),small(\_76,\_77)),(own(\_7,l,wai),own(\_79,\_69,\_77)),$ $(waist(\_8,wai),waist(\_78,\_77)),(very(\_9,s),very(\_80,\_76)),$ $(shoulders(\_11,sho),shoulders(\_88,\_89)),(own(\_10,l,sho),own(\_91,\_69,\_89)),$ $(own(\_13,l,ja),own(\_87,\_69,\_85)),(stark(\_12,sho),stark(\_90,\_89)),$ $(jaws(\_14,ja),jaws(\_86,\_85)),(grim(\_15,ja),grim(\_84,\_85)),$ $(boy(\_16,c),boy(\_65,\_62)),(good(\_17,c),good(\_66,\_62)),$ $(little(\_18,c),little(\_67,\_62)),(play(pl,c),play(\_61,\_62)),$ $(will(wpl,pl),will(\_63,\_61)),(not(\_20,wpl),not(\_64,\_63)),$
Target dangling	$own(\_4,l,h)]$
Candidate dangling	$\emptyset$

Table 8.33: Best individual for SPUD-like operator test

structure, which in our grammar is a sentence frame. Thus, the quality of the resulting text crucially hinges on the choice of the initial structure. In Nicolov's PROTECTOR, the initial skeletal frame is carefully chosen by selecting the mapping rule that covers as much as possible of the input conceptual graph. Our operator has no such heuristic, and thus it is left to random chance. In our initial testing of PROTECTORLIKE, the results obtained were highly variable depending on the initial structure chosen (Table 7.22).

Furthermore, tactical NLG components such as PROTECTOR are typically designed to operate in tandem with text planning modules which 'chunk' the input semantics into more manageable segments. All our semantically smart operators, save for SPUDLIKEDISCOURSE, have no account whatsoever for discourse. Considering that it is virtually impossible to convey the entire semantics of `lion` within a single sentence, we can see why it poses such a difficult challenge. Indeed, the success of MCGONAGALL in satisfying the `lionhalf` target semantics, even using blind operators (Section 8.4.1), suggests that it is quite capable of faithfully conveying target semantics of more modest size, e.g. that which can be realized within the scope of a sentence.

Another related point is that within our system, pronouns are added completely opportunistically, and pronoun reference is only computed during evaluation. A fairly sophisticated treatment of referring expressions is crucial for the generation of multi-sentential texts, and is an important aspect of most modern NLG systems.

These factors might also explain the awkwardness of the texts in the individuals obtained for the `lion` target semantics in Sections 8.4.1 and 8.4.2.

The best individual achieved by using the SPUDLikeDiscourse operator (Table 8.33) is by far the best result obtained by McGonagall in this second stage. It manages to convey the entire set of literals in `lion` save for $own(\_4,l,h)$, the owning of the head by the lion. Moreover, it manages to achieve this without any dangling candidate literals. This suggests that the incremental, per-elementary-tree approach it takes is more amenable to be used within the stochastic framework of an EA than that of PROTECTORLike.

The growth rate for the EA using PROTECTORLike starts from a much higher fitness score, but fairly rapidly converges to the local maximum (0.85), whereas SPUDLikeDiscourse increases more gradually (see Figure D.12(a) and (b)).

Furthermore, unlike PROTECTORLike, SPUDLikeDiscourse can be thought of possessing a small measure of discourse heuristics. Crucially, its algorithm (Algorithm 7.21) will perform the adjoining of a new sentence frame, i.e.SyntaxAddAdjoin, only after it has exhausted all other possibilities of operators.

Another aspect that possibly contributes to its success is that it is under no pressure to immediately create a complete derivation. The drawback, of course, is that there are many substitution nodes, and as a result, the individual is not yet a fully grammatical text . Fortunately, aside from the single $\boxed{NP}$ subject node in the last sentence, they are all closed class categories: determiners, complementizers, copula and auxiliary verbs, and punctuation marks. If we were to fill in these substitution nodes with obvious syntactic choices, we would obtain the following text:

> "*A good boy, who is little, will not play with the african lion, who dwells in the waste. Its waist will be very small. Its head is big. Its jaws are grim.* $\boxed{NP}$ *are its shoulders, which are stark.*"

From this text, we can see why the EA does not substitute anything at the $\boxed{NP}$ node. The most plausible noun phrases to substitute it would either be the pronoun *'they'*, which would refer to *'jaws'*, or the noun *'shoulders'*, both of which would decrease fitness.

8.4.4 Summary of discussion

The results obtained in the tests of this second stage show that McGonagall succeeded in finding optimal solutions for the `lionhalf` target semantics, even with the blind operators (Table 8.20). Unfortunately, the task of conveying the `lion` target semantics was met with less success, and it is only with the explicitly goal-directed operator SPUDLikeDiscourse that a "satisfycing" near-optimal solution is obtained. One possible explanation for this is that aside from this operator, our genetic operators have no account for discourse whatsoever. Considering that it is virtually impossible to convey the entire semantics of `lion` within a single sentence, we can see why it poses such a difficult challenge. Ideally, some sort of sentence planning, or 'chunking', should be performed for a target semantics of this size. Unfortunately, due to to time constraints, this is not an issue we were able to explore further in this thesis.

Unlike during the first stage, where it was fairly clear to see that the resulting texts did indeed exhibit the correct metre patterns, in this stage it is slightly more difficult to observe whether the resulting texts convey the intended semantics. This is particularly true of the longer texts that are intended to convey the `lion` target semantics. This was due to peculiarities in the texts which arose from a combination of our very simple treatment of pronoun resolution, and the exploitation of certain rules in the grammar, such as the nominal copula constructions, e.g.*"a big head, it is its expense"*.

Although we found weighting scheme 1 to yield the most accurate realizations of the target semantics, in the sense that the candidate semantics of the solutions are proper subsets of S_{target}, we found that it caused the EA to behave too conservatively in this respect, and did not provide enough opportunity to expand linguistic content. In terms of producing texts that conveyed most of the given S_{target}, weighting scheme 2 was found most useful, although this comes at a cost of more dangling candidate literals, which could obscure the meaning of S_{target} in the candidate solution's text.

8.5 McGonagall as poetry generation system

Throughout the first and second stages of our empirical study we have experimented using McGonagall as a form-aware text generator (Section 8.3) and as a tactical NLG component (Section 8.4). We have seen that for these tasks, McGonagall is able to achieve optimal solutions, and that these solutions possess the features that they are required to exhibit. The only exception to this is the conveying of the `lion` target semantics, for reasons discussed above in Section 8.4.4.

Although these tests have revealed several interesting points (see the discussions in Sections 8.3.8 and 8.4.4 for a summary), we have yet to test McGONAGALL, and thus the EA-based NLG theory that underlies it, at the task which it has specifically been designed for, namely the generation of texts that simultaneously satisfy grammaticality, meaningfulness, and poeticness, i.e. our definition of poetry.

In this section we report on our attempts at this multi-objective optimization task. As in the first two stages, we first perform an initial test (Section 8.5.1) which defines the baseline for comparison. The subsequent tests examine the use of our semantically smart operators (Section 8.5.2), the behaviour of the EA when we attempt to distribute the heuristics for optimizing meaningfulness and poeticness between the genetic operators and evaluation function used (Section 8.5.3), and the generation of texts on a line-by-line basis (Section 8.5.4).

8.5.1 Initial poetry generation test

In this test we simultaneously use the metre similarity and semantic similarity evaluation functions, and examine their effect on the behaviour and performance of the EA and the best individual obtained. In a sense, we are combining the efforts of the enjambment and linebreaking test (Section 8.3.4) and the initial tactical NLG test (Section 8.4.1). We base the configuration of this test on these two previous tests.

Target

Following the tests we conducted in the first stage of our study (Section 8.3), we test three different target forms: `haiku`, `limerick`, and `mignonne`.

For target semantics, we choose `lionhalf`. As discovered during the second stage of our study (Section 8.4), McGONAGALL was able to find optimal solutions for this target semantics, and it will be interesting to see how it will fare given the additional constraint of metre satisfaction.

Evaluators

For the metre evaluation function, we use the edit distance function, $\mathcal{F}_{edit}$, using the operation costs defined in Table 8.11, where the insertion of a linebreak target syllable, b, incurs a relatively expensive cost of 10. This was found in Section 8.3.4 to guide the EA towards solutions that have naturally occurring linebreaks.

For the semantic evaluation function, we use the semantic similarity equation used throughout the second stage of our study (Section 8.4), and we choose weighting scheme 2, which we found to be most useful in finding texts that maximize the mapping to S_{target} (although at a cost of increased dangling candidate literals).

In combining the two evaluation functions, we simply use a linear combination where their scores are averaged:

$$\mathcal{F}_{poetry} = \frac{\mathcal{F}_{edit} + \mathcal{F}_{sem}}{2}$$

EA Setup

In this test, we use the same EA parameters used in Section 8.4.1, except we only use an elitist ratio of 20%. More specifically:

- **Selection algorithm:** this test uses the proportionate selection algorithm with stochastic universal sampling, as used in the tests of the first stage. We employ an elitist strategy EA with elitist populations of 20% of the entire population.
- **Population size:** as in the previous stages, we choose a population size of 40.

Operators

We also use the same genetic operators used in Section 8.4.1, i.e. the compound operators that ensure complete derivations, presented in Section 7.4.1 and first used in Section 8.3.3. The probabilities of them being applied when mutation is called for are also the same as during the first stage:

- BLINDCREATECOMPLETE = 0.5
- BLINDADJOINCOMPLETE = 0.3

- BlindDeleteComplete = 0.2

Additionally, when crossover is called for, the subtree swapping operator BlindSwap is used. As the three mutation operators above guarantee complete derivations, BlindSwap always guarantees complete derivations as well.

We assign the probabilities of applying genetic operators as $p_{mutation} = 0.6$ and $p_{crossover} = 0.4$.

Resources

We use the same linguistic resources used throughout the first two stages, i.e. the handcrafted grammar and lexicon shown in Appendix C.

Expectations

Our hope is that the multi-objective evaluation function, $\mathcal{F}_{poetry}$, will guide the EA towards an optimal solution which satisfies both the given F_{target} and S_{target}.

Note that there is an interesting relationship between the target form and target semantics. There should be a balance between them, in the sense that the target form should be large enough to provide space for the target semantics to be conveyed, but not excessively large. Conversely, the target semantics should provide ample semantic content to be realized over the target form, but not too much. Such notions of ‘enough space’ and ‘ample semantic content’ are very vague. However, given that the first two lines of Belloc’s poem, i.e. a text which we already know accurately conveys `lionhalf`, consists of 22 syllables, we can provide a rough prediction of how `lionhalf` is conveyed as a `haiku` (17 syllables), `limerick` (34 syllables), and `mignonne` (84 syllables). We would expect `haiku` to be the most appropriate form given the syllable count, followed by `limerick` and `mignonne`.

Results

We ran this test three times, once for each target form. Each test was run five times. Summary statistics and graphs for this test are presented in Section D.3.1 of Appendix D.

Tables 8.34, 8.35, and 8.36 show the individual that yields the best fitness scores from the last populations of each test. They show the best individual for the `haiku`, `limerick`, and

Score	0.77
Surface	he dwells in a waist . his very big head has her . he dwells in his waist .
Formatted	He dwells in a waist. His very big head has her. He dwells in his waist.
Proper match	$\{(head(\_5,h),head(\_13,\_11)),(own(\_4,l,h),own(\_16,\_17,\_11)),$ $(own(\_7,l,wai),own(\_26,\_17,\_24)),(inside(\_2,d,was),inside(\_21,\_18,\_20)),$ $(dwell(d,l),dwell(\_18,\_17)),(big(\_6,h),big(\_14,\_11)),$ $(waist(\_8,wai),waist(\_25,\_24))\}$
Target dangling	$\{lion(\_0,l),african(\_1,l),waste(\_3,was),small(s,wai),$ $very(\_9,s)\}$
Candidate dangling	$\{own(\_10,\_11,\_12),very(\_15,\_14),waist(\_19,\_20),dwell(\_22,\_17),$ $inside(\_23,\_22,\_24)\}$

Table 8.34: Best individual for F_{target} =`haiku`, initial poetry generation test

`mignonne` target forms respectively, and for each individual, we show its fitness score, raw surface form, formatted surface text, and mapping between S_{target} and $S_{candidate}$. See Sections 8.3.1 and 8.4.1 for an explanation of the formatted surface texts and semantic mappings.

Observations and discussion

Comparing the scores obtained in Table D.13 to those obtained when satisfying F_{target} and S_{target} independently, i.e. Table D.4 and D.8, we can see that the multi-objective optimization task in this test is a much more difficult one.

Examining the best individuals obtained for the `haiku` and `limerick` form, we can see that the formatted surface texts are both metrically perfect. However, in terms of conveying `lionhalf`, they both perform suboptimally.

In our expectations for this test, we predicted that the `haiku` form would be the most appropriate, due to its syllable count. However, the first two lines of the slightly altered version of Belloc's *"The Lion"* shown in Table 8.18 is already a very dense text that conveys `lionhalf`. Thus, it is not a trivial task to write an even shorter paraphrase of it[1]. Additionally, it uses conjunction, e.g.*"a big head and a very small waist"*, a space-saving turn of phrase which our handcrafted grammar does not provide.

[1] Indeed, our own attempts at a terse paraphrase has 18 syllables: "the big headed, very small waisted, african lion dwells in the waste", one more than is provided by `haiku`!

Score	0.81
Surface	a lion , it dwells in a waste . a lion , it dwells in a waste . a waste will be rare . its head will be rare . its waist , that is small , will be rare .
Formatted	A **li**on, it **dwells** in a **waste**. A **li**on, it **dwells** in a **waste**. A **waste** will be **rare**. Its **head** will be **rare**. Its **waist**, that is **small**, will be **rare**.
Proper match	$\{(head(\_5,h),head(\_53,\_52)),(own(\_4,l,h),own(\_54,\_28,\_52)),$ $(own(\_7,l,wai),own(\_48,\_28,\_45)),(lion(\_0,l),lion(\_29,\_28)),$ $(dwell(d,l),dwell(\_27,\_28)),(inside(\_2,d,was),inside(\_30,\_27,\_31)),$ $(waste(\_3,was),waste(\_32,\_31)),(small(s,wai),small(\_47,\_45)),$ $(waist(\_8,wai),waist(\_46,\_45))\}$
Target dangling	$\{african(\_1,l),big(\_6,h),very(\_9,s)\}$
Candidate dangling	$\{rare(\_33,\_34),will(\_35,\_36),waste(\_37,\_34),dwell(\_38,\_39),$ $lion(\_40,\_39),inside(\_41,\_38,\_42),waste(\_43,\_42),rare(\_44,\_45),$ $will(\_49,\_50),rare(\_51,\_52),will(\_55,\_56)\}$

Table 8.35: Best individual for F_{target} = `limerick`, initial poetry generation test

However, syllable count is not the only aspect. Note that since our edit cost function is designed to heavily penalize arbitrary insertion of linebreaks, there is pressure to have linebreaks coincide with phrase boundaries. Given that the lines of `haiku` are only either 5 or 7 syllables, it can be difficult to convey certain aspects of `lionhalf`.

The individual obtained for the `limerick` target form contains instances of the word *'rare'*, which is unrelated to the semantics of `lionhalf`. We can see that while the phrase it appears in, *"will be rare"*, contributes positively to achieving the target metre, they end up as dangling candidate literals. Weighting scheme 2, however, is 'forgiving' enough of these dangling literals, and in fact we can see from the mapping that this text is actually conveying a large subset of `lionhalf`.

In the case of the individual for the `mignonne` target form (Table 8.36), we can see from the formatted surface text that the metre is not perfect. However, it is interesting to note that the five lines which introduce edit operations, i.e. deletion of candidate syllables, are precisely the lines which introduce the necessary semantics to convey `lionhalf`. The other lines are metrically accurate, but semantically they are extraneous, as can be seen by the abundance of dangling candidate literals. Due to the imbalance of the size of the target semantics and target form that we discussed in our expectations, McGONAGALL was forced to "pad" the remaining

Score	0.78
Surface	with a pole , a african lion , it dwells in a head , that is small , in its big head , that is small , in a head , that is small , in a head , that is very small , in a waist , that is a waste , in a head , that is small , in a head , that is small , in a head , that is small , in a head , that is small , in a head , that is small , in a head , that is small , in a head , that is small , with its waist , that is small , .
Formatted	With a **pole**, (an) (af)rican **li**(on), (it) (dwells) in a **head**, that is **small**, in its (big) **head**, that is **small**, in a **head**, that is **small**, in a **head**, (that) is (ve)ry **small**, in a **waist**, (that) is a **waste**, in a **head**, that is **small**, in a **head**, that is **small**, in a **head**, that is **small**, in a **head**, that is **small**, in a **head**, that is **small**, in a **head**, that is **small**, in a **head**, that is **small**, with its **waist**, that is **small**.
Proper match	$\{(lion(\_0,l),lion(\_111,\_58)),(african(\_1,l),african(\_112,\_58)),$ $(dwell(d,l),dwell(\_57,\_58)),(waste(\_3,was),waste(\_105,\_70)),$ $(inside(\_2,d,was),inside(\_69,\_57,\_70)),(big(\_6,h),big(\_62,\_60)),$ $(own(\_4,l,h),own(\_64,\_58,\_60)),(own(\_7,l,wai),own(\_91,\_58,\_88)),$ $(head(\_5,h),head(\_61,\_60)),(small(s,wai),small(\_90,\_88)),$ $(waist(\_8,wai),waist(\_89,\_88))\}$
Target dangling	$\{very(\_9,s)\}$
Candidate dangling	$\{inside(\_59,\_57,\_60),small(\_63,\_60),inside(\_65,\_57,\_66),inside(\_67,\_57,\_68),$ $inside(\_71,\_57,\_72),inside(\_73,\_57,\_74),inside(\_75,\_57,\_76),inside(\_77,\_57,\_78),$ $inside(\_79,\_57,\_80),inside(\_81,\_57,\_82),inside(\_83,\_57,\_84),head(\_85,\_84),$ $small(\_86,\_84),with(\_87,\_57,\_88),head(\_92,\_82),small(\_93,\_82),$ $head(\_94,\_80),small(\_95,\_80),head(\_96,\_78),small(\_97,\_78),$ $head(\_98,\_76),small(\_99,\_76),head(\_100,\_74),small(\_101,\_74),$ $head(\_102,\_72),small(\_103,\_72),waist(\_104,\_70),head(\_106,\_68),$ $small(\_107,\_68),very(\_108,\_107),head(\_109,\_66),small(\_110,\_66),$ $inside(\_113,\_57,\_114),head(\_115,\_114),small(\_116,\_114),with(\_117,\_57,\_118),$ $pole(\_119,\_118)\}$

Table 8.36: Best individual for F_{target} =`mignonne`, initial poetry generation test

syllables.

Note that for all three individuals shown in Tables 8.34 to 8.36, they exhibit the repetition of lines to satisfy the target form. This is because the genetic operators in this test include the crossover operator BLINDSWAP, and as we first saw in Section 8.3.7, this repetition is an effective method of satisfying the target form.

8.5.2 Smart operators poetry generation tests

In this test we will examine the effect of using the semantically smart operators. The configuration of genetic operators is the same as in Section 8.4.2, i.e. we use the compound operators that ensure complete derivations and greedily consume S_{target}.

All other aspects of the test, i.e. target form and target semantics, EA parameters, evaluation functions, and linguistic resources, are the same as during the previous test (Section 8.5.1).

Expectations

We hope that the use of semantically smart operators will improve the semantic similarity of the individuals obtained. However, this may be achieved at the cost of metre similarity, where the semantically goal-directed operators may miss certain opportunities for satisfying the metre, as in the limerick in Table 8.35, where the last three lines ends with the metrically perfect but semantically extraneous *"will be rare"*.

Results

We ran this test three times, once for each target form. Each test was run five times. Summary statistics and graphs for this test are presented in Section D.3.2 of Appendix D.

Tables 8.37, 8.38, and 8.39 show the individual that yields the best fitness scores from the last populations of each test. They show the best individual for the `haiku`, `limerick`, and `mignonne` target forms respectively, and for each individual, we show its fitness score, raw surface form, formatted surface text, and mapping between S_{target} and $S_{candidate}$. See Sections 8.3.1 and 8.4.1 for an explanation of the formatted surface texts and semantic mappings.

Score	0.86
Surface	in a waste , a lion , who has a very small waist , dwells in a big head .
Formatted	(In) a waste, a lion, who has a very small waist, dwells in a big head.
Proper match	$\{(lion(\_0,l), lion(\_124,\_121)), (dwell(d,l), dwell(\_120,\_121)),$ $(waste(\_3,was), waste(\_134,\_133)), (inside(\_2,d,was), inside(\_132,\_120,\_133)),$ $(head(\_5,h), head(\_130,\_123)), (own(\_7,l,wai), own(\_125,\_121,\_126)),$ $(big(\_6,h), big(\_131,\_123)), (small(s,wai), small(\_128,\_126)),$ $(waist(\_8,wai), waist(\_127,\_126)), (very(\_9,s), very(\_129,\_128))\}$
Target dangling	$\{african(\_1,l), own(\_4,l,h)\}$
Candidate dangling	$\{inside(\_122,\_120,\_123)\}$

Table 8.37: Best individual for $F_{target} = \texttt{haiku}$, smart operators poetry generation test

Score	0.83
Surface	a very african lion , who is african , dwells in a waste . its head , that is big , is very big . a waist , that is its waist , , it is small .
Formatted	A **ve**ry * **af**rican **li**(on), (who) is **af**rican, **dwells** in a **waste**. Its **head**, that is **big**, is **ve**ry * **big**. A **waist**, (that) is its **waist**, it is **small**.
Proper match	$\{(inside(\_2,d,was), inside(\_150,\_147,\_149)), (lion(\_0,l), lion(\_151,\_140)),$ $(dwell(d,l), dwell(\_147,\_140)), (waste(\_3,was), waste(\_148,\_149)),$ $(head(\_5,h), head(\_138,\_136)), (own(\_4,l,h), own(\_139,\_140,\_136)),$ $(own(\_7,l,wai), own(\_146,\_140,\_143)), (small(s,wai), small(\_142,\_143)),$ $(african(\_1,l), african(\_152,\_140)), (big(\_6,h), big(\_135,\_136)),$ $(waist(\_8,wai), waist(\_144,\_143))\}$
Target dangling	$\{very(\_9,s)\}$
Candidate dangling	$\{very(\_137,\_135), big(\_141,\_136), waist(\_145,\_143), very(\_153,\_152),$ $african(\_154,\_140)\}$

Table 8.38: Best individual for $F_{target} = \texttt{limerick}$, smart operators poetry generation test

Score	0.80
Surface	a african lion , who dwells in a waste , has a waist , that in a waist , that is small , , is very small , . in a waste , in a waist , that is small , , in a waist , that is small , , in a waist , that is small , , its head , that is big , is a head , that in a waist , that in a waste , in a waist , that is small , , in a waist , that is small , , in a waist , that is small , , in a waist , that is small , , will be small , , will be big , .
Formatted	(An) (af)rican **li**(on), (who) (dwells) in a **waste**, has a **waist**, (that) in a **waist**, that is **small**, is (ve)ry **small**. In a **waste**, in a **waist**, that is **small**, in a **waist**, that is **small**, in a **waist**, that is **small**, * its **head**, that is **big**, is a **head**, (that) in a **waist**, (that) in a **waste**, in a **waist**, that is **small**, in a **waist**, that is **small**, in a **waist**, that is **small**, in a **waist**, that is **small**, will be **small**, will be **big**.
Proper match	$\{(lion(\_0,l),lion(\_158,\_156)),(african(\_1,l),african(\_159,\_156)),$ $(dwell(d,l),dwell(\_160,\_156)),(inside(\_2,d,was),inside(\_161,\_160,\_162)),$ $(waste(\_3,was),waste(\_163,\_162)),(very(\_9,s),very(\_170,\_165)),$ $(small(s,wai),small(\_165,\_157)),(own(\_4,l,h),own(\_217,\_156,\_172)),$ $(own(\_7,l,wai),own(\_155,\_156,\_157)),(waist(\_8,wai),waist(\_164,\_157)),$ $(head(\_5,h),head(\_171,\_172)),(big(\_6,h),big(\_173,\_172))\}$
Target candidate	$\emptyset$
Candidate dangling	$\{inside(\_166,\_165,\_167),waist(\_168,\_167),small(\_169,\_167),will(\_174,\_175),$ $inside(\_176,\_173,\_177),waist(\_178,\_177),small(\_179,\_177),inside(\_180,\_179,\_181),$ $inside(\_182,\_179,\_183),waist(\_184,\_183),small(\_185,\_183),inside(\_186,\_179,\_187),$ $inside(\_188,\_179,\_189),inside(\_190,\_179,\_191),waste(\_192,\_191),waist(\_193,\_189),$ $small(\_194,\_189),waist(\_195,\_187),small(\_196,\_187),waist(\_197,\_181),$ $small(\_198,\_181),will(\_199,\_200),inside(\_201,\_171,\_202),waist(\_203,\_202),$ $small(\_204,\_202),inside(\_205,\_171,\_206),inside(\_207,\_171,\_208),inside(\_209,\_171,\_210),$ $waste(\_211,\_210),waist(\_212,\_208),small(\_213,\_208),waist(\_214,\_206),$ $small(\_215,\_206),head(\_216,\_172),big(\_218,\_172)\}$

Table 8.39: Best individual for F_{target} =`mignonne`, smart operators poetry generation test

Observations and discussion

Comparing the scores obtained in this test (Table D.14) with those in the previous test (Table D.13) where blind operators were used, we can see that the maximum fitness scores achieved are slightly better, but we cannot drawn conclusions from these numbers alone. Given the multi-objective nature of the evaluation function used, we must scrutinize the individuals themselves.

Indeed, the individuals for the `haiku`, `limerick` and `mignonne` target forms all show a similar behaviour: in terms of metre similarity, they are inferior to the solutions obtained in Section 8.5.1, but they are semantically better, in the sense that there are fewer dangling target and candidate literals.

In particular, we can see from Table 8.37 that, for the first time in the entire study, the EA failed to find a perfect haiku. This is indicated by the deletion of the very first candidate syllable, *'in'*. The text in Table 8.38, unlike the metrically perfect limerick in Table 8.35, also requires several edit operations (2 insertions and 2 deletions). However, it conveys `lionhalf` much better. The only aspect that it fails to convey is the fact that the small waist is *very* small. Note that the dangling candidate literals are mainly semantically accurate duplicates of the literals mapped in the proper matches, and not extraneous semantics. This can be seen by their arguments, i.e. in the computed mapping, _136 represents the head, _143 is the waist, and _140 is the lion.

Unfortunately the best individual obtained for the `mignonne` target form (Table 8.39) is very similar to the one obtained in the previous test (Table 8.36), where it is forced to pad out the text with repeated prepositional phrases and relative clauses which serve only to increase the amount of dangling candidate literals.

We believe that the results in this test, especially for the `haiku` and `limerick` forms, are comparable to the results obtained from our chart generation system, where we adopted an exhaustive search approach to poetry generation (see Section 2.3.4).

8.5.3 Distribution of heuristics test

In this final test, we experiment with the interesting notion of distributing the heuristics used for achieving the multi-objective optimization of meaningfulness and poeticness between the genetic operators and evaluation function. More specifically, we will test running the EA using

Score	0.90
Surface	his waist is his waist . a lion , who dwells in a waste , will be african .
Formatted	His waist is his waist. (A) lion, who dwells in a waste, will be african.
Proper match	$\{(inside(\_2,d,was),inside(\_230,\_229,\_231)),(lion(\_0,l),lion(\_228,\_227)),$ $(african(\_1,l),african(\_226,\_227)),(dwell(d,l),dwell(\_229,\_227)),$ $(waste(\_3,was),waste(\_232,\_231)),(waist(\_8,wai),waist(\_219,\_220))\}$
Target dangling	$\{own(\_4,l,h),head(\_5,h),big(\_6,h),own(\_7,l,wai),$ $small(s,wai),very(\_9,s)\}$
Candidate dangling	$\{waist(\_221,\_220),own(\_222,\_223,\_220),own(\_224,\_225,\_220),will(\_233,\_234)\}$

Table 8.40: Best individual for F_{target} =`haiku`, distribution of heuristics test

the semantically smart operators, but only using the metre evaluation function.

For this test, we use the exact same setup as in the previous test (Section 8.5.2), except that we only use the metre edit distance function, F_{edit}, as our evaluation function.

Expectations

We expect the results to be at best equal to those obtained in the previous test, but expect it to be slightly inferior due to the reduced informedness provided by the evaluation function.

Results

We ran this test three times, once for each target form. Each test was run five times. Summary statistics and graphs for this test are presented in Section D.3.3 of Appendix D.

Tables 8.40, 8.41, and 8.42 show the individual that yields the best fitness scores from the last populations of each test. They show the best individual for the `haiku`, `limerick`, and `mignonne` target forms respectively, and for each individual, we show its fitness score, raw surface form, formatted surface text, and mapping between S_{target} and $S_{candidate}$. See Sections 8.3.1 and 8.4.1 for an explanation of the formatted surface texts and semantic mappings.

Score	0.82
Surface	its lion will not be its head , that is very big , . a species , that will be a waste , will not be a waste , that will be a waste in a waste , .
Formatted	Its **li**on will **not** be its **head**, (that) is **ve**ry * **big** *. A **spe**(cies), that **will** be a **waste**, will **not** be a **waste**, that **will** be a **waste** in a **waste**.
Proper match	$\{(inside(\_2,d,was),inside(\_258,\_255,\_259)),(lion(\_0,l),lion(\_243,\_236)),$ $(waste(\_3,was),waste(\_260,\_259)),(very(\_9,s),very(\_242,\_241))\}$
Target dangling	$\{african(\_1,l),dwell(d,l),own(\_4,l,h),head(\_5,h),$ $big(\_6,h),own(\_7,l,wai),small(s,wai),waist(\_8,wai)\}$
Candidate dangling	$\{head(\_235,\_236),own(\_237,\_236,\_236),will(\_238,\_239),not(\_240,\_238),$ $big(\_241,\_236),own(\_244,\_245,\_236),waste(\_246,\_247),will(\_248,\_249),$ $not(\_250,\_248),species(\_251,\_247),waste(\_252,\_247),will(\_253,\_254),$ $waste(\_255,\_247),will(\_256,\_257)\}$

Table 8.41: Best individual for F_{target} =`limerick`, distribution of heuristics test

Observations and discussion

In a sense, this test can be see as an approximation of the author-reviewer model of generation in Oberlander and Brew (2000) (see Section 3.5), where the genetic operators correspond to the author module, which seeks to achieve **fidelity**, and the evaluation function corresponds to the reviewer module, which maximizes **fluency**.

From a semantic similarity point of view, these individuals are not as good as the ones obtained in Sections 8.5.1 and 8.5.2. This is to be expected, however, as the EA is exerting no pressure whatsoever towards the conveying of S_{target}. From a metre similarity point of view, these individuals are not as good as the ones obtained in Section 8.5.1, which had the advantage of exploiting non-semantically motivated operations (recall that it used the blind baseline operators). They are roughly equivalent in terms of metrical quality to those in Section 8.5.2, which were equally constrained by having to use the semantically smart operators.

It would have been interesting to test the explicitly semantically-motivated operators, PROTECTORLIKE and SPUDLIKE, in this test, but unfortunately we were unable to do so due to lack of time. Nevertheless, the results obtained in this simple experiment show the potential provided by the EA framework.

Score	0.81
Surface	a head , that in a head , that in a head , that in a waist , that is small , , is a waste , , is a waste , , is a waste , is a waste , that in a head , that in a head , that in a waist , that is small , , is a waste , , is a waste , , is a waste , . a head , that in a head , that in a head , that in a head , that in a waist , that is small , , is a waste , , is a waste , , is a waste , , is a waste , is a waste , that is a waste , .
	* A **head**, (that) in a **head**, (that) in a **head**, (that) in a **waist**, that is **small**, is a **waste**, is a **waste**, is a **waste**, is a **waste**, (that) in a **head**, (that) in a **head**, (that) in a **waist**, that is **small**, is a **waste**, is a **waste**, is a **waste**. * A **head**, (that) in a **head**, (that) in a **head**, (that) in a **head**, (that) in a **waist**, that is **small**, is a **waste**, is a **waste**, is a **waste**, is a **waste**, is a **waste**, (that) is a **waste**.
Proper match	$\{(small(s,wai),small(\_280,\_278)),(waist(\_8,wai),waist(\_279,\_278)),$ $(inside(\_2,d,was),inside(\_265,\_264,\_266)),(waste(\_3,was),waste(\_268,\_266)),$ $(head(\_5,h),head(\_263,\_262))\}$
Target dangling	$\{lion(\_0,l),african(\_1,l),dwell(d,l),own(\_4,l,h),$ $big(\_6,h),own(\_7,l,wai),very(\_9,s)\}$
Candidate dangling	$\{waste(\_261,\_262),waste(\_264,\_262),head(\_267,\_266),inside(\_269,\_268,\_270),$ $head(\_271,\_270),waste(\_272,\_270),inside(\_273,\_272,\_274),head(\_275,\_274),$ $waste(\_276,\_274),inside(\_277,\_276,\_278),waste(\_281,\_262),waste(\_282,\_283),$ $head(\_284,\_283),waste(\_285,\_283),inside(\_286,\_285,\_287),head(\_288,\_287),$ $waste(\_289,\_287),inside(\_290,\_289,\_291),head(\_292,\_291),waste(\_293,\_291),$ $inside(\_294,\_293,\_295),waist(\_296,\_295),small(\_297,\_295),waste(\_298,\_283),$ $inside(\_299,\_298,\_300),head(\_301,\_300),waste(\_302,\_300),inside(\_303,\_302,\_304),$ $head(\_305,\_304),waste(\_306,\_304),inside(\_307,\_306,\_308),waist(\_309,\_308),$ $small(\_310,\_308)\}$

Table 8.42: Best individual for F_{target} =`mignonne`, distribution of heuristics test

8.5.4 Line-by-line generation

In this test, we attempt to generate a limerick as before, but this time we also break the task down and run McGonagall on each line of the limerick individually, i.e. the targets are the meaning and metre of the relevant line. The purpose of this test is to see whether McGonagall can generate optimal solutions for both semantics and metre when given these smaller targets.

The parameters used for this test in terms of EA setup, evaluators and genetic operators are the same as those found in Section 8.5.2, i.e. we use proportionate selection with stochastic universal sampling and an elitist ratio of 20%, a population size of 40, the $\mathcal{F}_{poetry}$ evaluation function which simply averages the scores of metre and semantic evaluation, and the semantically-motivated compound operators that ensure complete derivations.

Only the semantic targets, metre targets, and linguistic resources are changed, which we now discuss in more detail.

Target

We have based this test on an entirely different limerick in order to show the portability and flexibility of McGonagall, i.e. by encoding the appropriate targets and augmenting the linguistic resources accordingly, we should be able to generate different texts. In choosing a limerick for this test, we were constrained by the fact that each individual line should be able to stand on its own as a syntactically well-formed sentence. We selected the well known ‘relativity’ limerick shown in Table 8.43, along with our encoding of the `relativity` semantic target. The Oxford Dictionary of Quotations (Knowles, 1999) attributes authorship of this limerick to Arthur H.R. Buller, who first published it in Punch magazine.

Note that the original limerick consists of two sentences: the first sentence spans the first two lines and the second sentence spans the last three lines. Since we will be trying to generate this limerick on a line by line basis, we have modified the text slightly so that it consists of four complete sentences which McGonagall can generate individually. These four sentences are shown in Table 8.44, along with the respective semantic targets, `relativity1` to `relativity4`.

This modified limerick preserves the metre and syllable count of the original. However, the

There was a young lady called Bright *who could travel much faster than light.* *She set out one day* *in a relative way* *and returned on the previous night.*
`relativity`: $\{lady(\_,l), young(\_,l), name(\_,l,b), bright(\_,b),$ $travel(t,l), faster(f,t,li), light(\_,li), much(\_,f), can(\_,t),$ $leave(le,l), relative(\_,le), oneday(\_,le),$ $return(r,l), on(\_,r,n), night(\_,n), previous(\_,n)\}$

Table 8.43: Arthur H.R. Buller's 'relativity' limerick with `relativity` semantic target

Line 1	*There was a young lady called Bright.* `relativity1`: $\{lady(\_,l), young(\_,l), name(\_,l,b), bright(\_,b)\}$
Line 2	*She could travel much faster than light.* `relativity2`: $\{travel(t,l), faster(f,t,li), light(\_,li), much(\_,f), can(\_,t)\}$
Line 3	*She set out one day in a relative way.* `relativity3`: $\{leave(le,l), relative(\_,le), oneday(\_,le)\}$
Line 4	*She returned on the previous night.* `relativity4`: $\{return(r,l), on(\_,r,n), night(\_,n), previous(\_,n)\}$

Table 8.44: Modified 'relativity' limerick consisting of four complete sentences

third and fourth lines from the original limerick have now been merged into one line. The form targets for lines 1, 2, and 4 of the modified limerick are the same, as follows:

`limerickline1`: `[w,s,w,w,s,w,w,s,b]`

The form target for line 3 is as follows:

`limerickline2`: `[w,s,w,w,s,w,s,w,w,s,b]`

Note the lack of a linebreak syllable in the middle of `limerickline2`. This is because line 3 is one sentence, and it would thus be 'unfair' on McGONAGALL if we penalized it for not having a proper linebreak at this point.

To summarize, we will be using S_{target} =`relativity` along with F_{target} =`limerick` to generate the entire modified limerick, S_{target} =`relativity1`, `relativity2`, and `relativity4` along with F_{target} =`limerickline1` to generate lines 1, 2, and 4, and finally S_{target} =`relativity3` and F_{target}=`limerickline2` to generate line 3.

Note that although the four lines of our modified limerick perfectly satisfy their respective semantic targets, in terms of metre they are actually suboptimal. We will discuss this issue in more depth along with the discussion of our test results.

Resources

For McGonagall to be able to generate the ‘relativity’ limerick and its individual lines, we must augment the available linguistic resources, i.e. grammar and lexicon, accordingly. Furthermore, we ensured that there are several ways of realizing the targets, i.e. different paraphrases, so as to provide ample opportunity for McGonagall to convey the semantics.

In order to generate the ‘relativity’ limerick, we had to account for several syntactic structures that we had not previously come across. For example, the existential copula sentence form very widely used in limerick openings, i.e. “*There was a* $\boxed{NP}$”. Other examples of syntactic structures we had to account for are auxiliary NP postmodifiers (e.g. “*The lady called Bright*”), comparative VP postmodifiers (e.g. “*travels faster than light*”), and adverbial VP postmodifiers (e.g. “*travels relatively*”).

To provide McGonagall with ample opportunities in realizing the targets, we ensured that there were several different ways of conveying the same semantics. For example:

- Alternative paraphrasing: “*travelled in a relative way*” vs. “*travelled relatively*”.
- Pre and postmodification: “*yesterday left*” vs. “*left yesterday*”.
- Synonyms: ‘*set out*’ vs. ‘*left*’, ‘*faster*’ vs. ‘*speedier*’, ‘*previous*’ vs. ‘*preceding*’.

Note that these paraphrases are considered semantically equivalent, but have different metre patterns. In particular, we chose as synonyms lexical entries with either different syllable counts (‘*faster*’ vs. ‘*speedier*’) or different stress patterns (‘*PREvious*’ vs. ‘*preCEding*’).

The additional lexical and syntactic resources are fully listed in Appendix C.1.5 and C.2.9. In this test, we used these additional resources together with all the previously defined grammar and lexicon used in preceding tests.

Before conducting the test proper, we informally verified that the resources could indeed be used to generate texts that convey the given target semantics. This was done by running McGONAGALL with only semantic evaluation. For example, for S_{target} =`relativity3`, we obtained the following optimal solutions:

- she relatively left one day .
- she left relatively one day .
- she set out relatively one day .
- she one day relatively set out .
- she set out in a relative way, it one day .

For S_{target} =`relativity4`, we obtained, among others, the following optimal solutions:

- on a preceding night , he returned .
- on a preceding evening , she returned . she returned .
- she returned on a night . it is previous .
- it is it . he returned on a previous night .

In fact, we even achieved several optimal solutions in terms of both semantics and metre, even though this was purely coincidental, given that only semantic evaluation was used to guide the generation. For example, for S_{target} =`relativity2`, we obtained the optimal solution:

- she could travel much faster than light .

Finally, for S_{target} =`relativity`, i.e. the entire limerick's semantics, we obtained the following best solution:

- a young lady , who set out one day relatively , , she called them could travel much speedier than light . she returned . a night , it is preceding .

This solution had a score of 0.80, and only failed to convey the semantics that the lady is called BRIGHT ($bright(\_,b)$) and that she returned ON the preceding night ($on(\_,r,n)$).

Score	0.69
Surface	a lady could be on a evening , that could be preceding , one day . a young lady called bright , who set out one day , travelled much faster than light .
Formatted	A **la**dy could **be** on an **eve**(ning), that **could** be pre**ce**ding, one **day**. A (young) **la**dy called **bright**, who **set** out one **day**, * **tra**velled much **fas**ter than **light**.
Proper match	$\{(\mathit{faster}(f,t,li),\mathit{faster}(\_167,\_178,\_169)),(\mathit{name}(\_2,l,b),\mathit{name}(\_184,\_166,\_186)),$ $(\mathit{on}(\_9,r,n),\mathit{on}(\_210,\_197,\_198)),(\mathit{travel}(t,l),\mathit{travel}(\_178,\_166)),$ $(\mathit{much}(\_5,f),\mathit{much}(\_170,\_167)),(\mathit{light}(\_4,li),\mathit{light}(\_172,\_169)),$ $(\mathit{lady}(\_0,l),\mathit{lady}(\_177,\_166)),(\mathit{young}(\_1,l),\mathit{young}(\_179,\_166)),$ $(\mathit{bright}(\_3,b),\mathit{bright}(\_187,\_186)),(\mathit{leave}(le,l),\mathit{leave}(\_192,\_166)),$ $(\mathit{oneday}(\_8,le),\mathit{oneday}(\_194,\_192)),(\mathit{night}(\_10,n),\mathit{night}(\_208,\_198)),$ $(\mathit{previous}(\_11,n),\mathit{previous}(\_210,\_198))\}$
Target dangling	$\{\mathit{can}(\_6,t),\mathit{relative}(\_7,le),\mathit{return}(r,l)\}$
Candidate dangling	$\{\mathit{oneday}(\_199,\_210),\mathit{lady}(\_201,\_197),\mathit{can}(\_221,\_207),\mathit{can}(\_228,\_213)\}$

Table 8.45: Best individual for entire relativity limerick

Expectations

We hope that given the appropriate resources, and the smaller targets, McGonagall should be able to obtain optimal solutions for the individual lines. For the generation of the entire limerick, we hope it performs well as it did with the `lionhalf` semantics in Section 8.5.2.

Results

We ran this test five times, once for the entire limerick and once for each individual line. Each test was run ten times. Summary statistics for this test are presented in Section D.3.4 of Appendix D.

Tables 8.45 to 8.49 show the individual that yields the best fitness scores from the last populations of each test. They show the best individual for the `relativity`, `relativity1`, `relativity2`, `relativity3`, and `relativity4` target semantics respectively. For each individual, we show its fitness score, raw surface form, formatted surface text, and mapping between S_{target} and $S_{candidate}$. See Sections 8.3.1 and 8.4.1 for an explanation of the formatted surface texts and semantic mappings.

Score	0.82
Surface	a lady called bright could be young .
Formatted	A **la**dy called **bright** could be **young**.
Proper match	$\{(name(\_2,l,b),name(\_72,\_66,\_74)),(young(\_1,l),young(\_41,\_66)),$ $(lady(\_0,l),lady(\_67,\_66)),(bright(\_3,b),bright(\_75,\_74))\}$
Target dangling	$\emptyset$
Candidate dangling	$\{can(\_57,\_81)\}$

Table 8.46: Best individual for first line, relativity limerick

Score	0.66
Surface	she travelled . the light could be light .
Formatted	she **tra**velled. The **light** could be **light**.
Proper match	$\{(light(\_0,li),light(\_604,\_524)),(can(\_2,t),can(\_607,\_526)),$ $()\}$
Target dangling	$\{travel(t,l),faster(f,t,li),much(\_1,f)\}$
Candidate dangling	$\{light(\_527,\_524),travel(\_603,\_536)\}$

Table 8.47: Best individual for second line, relativity limerick

Score	0.78
Surface	she set out one day . she set out one day .
Formatted	She **set** out one **day**. She **set** out one **day**.
Proper match	$\{(leave(le,l),leave(\_360,\_323)),(oneday(\_1,le),oneday(\_324,\_360))\}$
Target dangling	$\{relative(\_0,le)\}$
Candidate dangling	$\{leave(\_359,\_323),oneday(\_333,\_359)\}$

Table 8.48: Best individual for third and fourth line, relativity limerick

Score	0.86
Surface	she is on a previous night .
Formatted	She **is** on a **pre**vious **night**.
Proper match	$\{(on(\_0,r,n),on(\_43,\_229,\_230)),(night(\_1,n),night(\_236,\_230)),$ $(previous(\_2,n),previous(\_241,\_230))\}$
Target dangling	$\{return(r,l)\}$
Candidate dangling	$\emptyset$

Table 8.49: Best individual for last line, relativity limerick

Observations and discussion

In general, these results show that McGonagall is still not managing to find optimal solutions despite the reduced complexity of the task when generating individual lines.

Regarding the generation of the entire limerick, the individual in Table 8.45 does indeed show a quality reminiscent of the limerick for `lionhalf` in Table 8.38, where there are imperfections in both metre and semantics. Note that `relativity` is a harder semantic target to satisfy as it is slightly larger. Nevertheless, we can see that a large portion of the semantics is properly conveyed, i.e. matched, particularly in the last sentence (*"A young lady called Bright, who set out one day, travelled much faster than light."*).

Of the four individually generated lines, only for the first line (Table 8.46) does McGonagall manage to convey the entire S_{target}, which in this case is `relativity1`. For the second line (Table 8.47), it fails to convey the literals $faster(f,t,li)$ and $much(\_,f)$. Note that the dangling target and candidate *travel* literals are because our mapping algorithm chose to match the *can* literals, and thus cannot incorporate *travel* in a structurally consistent fashion. However, the computed mapping is still a maximal structurally consistent mapping. For the third line (Table 8.48), it fails to convey the literal $relative(\_,le)$, whereas for the last line (Table 8.49), it fails to convey the literal $return(r,l)$.

We know for certain that this is not due to McGonagall's inability to simply satisfy the semantics, as during our informal test to validate the linguistic resources described above, it indeed managed to find optimal solutions for each individual line. Thus it is clear that the additional constraint imposed by the target metre is keeping McGonagall trapped in local maxima as far as meaningfulness is concerned.

However, we can clearly observe that for the individually generated lines, MCGONAGALL is performing much better with respect to the target metre. The individuals for the second and third lines are perfect, whereas the individuals for the first and last lines only have one imperfection each: the destressing of '*called*' and the stressing of '*is*'. In each case, the lines generated by MCGONAGALL are in fact superior to the 'optimal' solutions in Table 8.44! Although we assume the lines in Table 8.44 to be optimal solutions, when scored by the evaluation function used in this test, they actually yield scores of 0.68, 0.91, 0.87, and 0.91 respectively. This is due to the fact that they either have too many syllables, as is the case with lines 2, 3, and 4, or the stressing/destressing of syllables, as is the case with line 1, i.e. the stressing of '*was*' and the destressing of '*young*' and '*called*'. Thus, as far as our evaluation function is concerned, the individual in Table 8.46 is actually **better** than the first line in Table 8.44.

This suggests that our simplistic multi-objective optimization method of simply averaging the scores, i.e.

$$\mathcal{F}_{poetry} = \frac{\mathcal{F}_{edit} + \mathcal{F}_{sem}}{2}$$

is unbalanced in the sense that it is not guiding the EA to optimize semantics as well as it is for metre. Another fact that suggests this is the following individual which was generated during the test runs for `relativity4`:

- a night is preceding . she returned on it .

This individual has a fitness score of 0.76, which is lower than that of the individual in Table 8.49, despite the fact that it perfectly satisfies the target semantics. Unfortunately, it has far too many syllables (11), which is what reduces its fitness.

One simple test which we conducted was to modify the linear combination of $\mathcal{F}_{edit}$ and $\mathcal{F}_{sem}$. We tried generating the individual lines using the following evaluation function, which effectively doubles the weight of semantic evaluation:

$$\mathcal{F}_{poetry} = \frac{\mathcal{F}_{edit} + 2\mathcal{F}_{sem}}{3}$$

Note that using this evaluator, the four lines in Table 8.44 now yield scores of 0.78, 0.94, 0.91, and 0.94 respectively.

Using this modified evaluator, we repeated the test run. Tables 8.50 to 8.53 show the individ-

Score	0.78
Surface	there is a young lady called bright .
Formatted	There **is** a young **la**dy called **bright**.
Proper match	$\{(name(\_2,l,b),name(\_421,\_415,\_423)),(lady(\_0,l),lady(\_445,\_415)),$ $(young(\_1,l),young(\_416,\_415)),(bright(\_3,b),bright(\_424,\_423))\}$
Target dangling	$\emptyset$
Candidate dangling	$\emptyset$

Table 8.50: Best individual for first line, relativity limerick, double semantic fitness

Score	0.79
Surface	she will travel much faster than light .
Formatted	She (will) **tra**vel much **fas**ter than **light**.
Proper match	$\{(faster(f,t,li),faster(\_399,\_415,\_401)),(travel(t,l),travel(\_415,\_393)),$ $(light(\_0,li),light(\_402,\_401)),(much(\_1,f),much(\_407,\_399))\}$
Target dangling	$\{can(\_2,t)\}$
Candidate dangling	$\{will(\_409,\_415)\}$

Table 8.51: Best individual for second line, relativity limerick, double semantic fitness

ual that yields the best fitness scores from the last populations of each test. They show the best individual for the `relativity1`, `relativity2`, `relativity3`, and `relativity4` target semantics respectively.

In general, these individuals show an improvement in meaningfulness (semantic faithfulness) with a tradeoff in poeticness (metre satisfaction), which is to be expected given the shift in weight towards $\mathcal{F}_{sem}$.

The individual for the first line (Table 8.50) is now exactly the same as the ‘optimal’ first line in Table 8.44 in terms of metre and semantics. Note that the score is 0.78 due to the various stressing and destressing to satisfy the target metre. The `relativity1` target semantics is perfectly conveyed.

The individual for the second line (Table 8.51) is also better than the one in Table 8.47. The only difference from the second line in Table 8.44 is the use of ‘*will*’ instead of ‘*could*’. Unfortunately, this reduces the semantic fitness.

Score	0.95
Surface	she set out one day relatively .
Formatted	She **set** out one **day** * **re**lative**ly**.
Proper match	$\{(leave(le,l), leave(\_253,\_280)), (oneday(\_1,le), oneday(\_281,\_253)),$ $(relative(\_0,le), relative(\_283,\_253))\}$
Target dangling	$\emptyset$
Candidate dangling	$\emptyset$

Table 8.52: Best individual for third and fourth line, relativity limerick, double semantic fitness

Score	0.76
Surface	she is on a preceding night .
Formatted	She **is** on (a) pre**ce**ding * **night**.
Proper match	$\{(on(\_0,r,n), on(\_355,\_333,\_334)), (night(\_1,n), night(\_335,\_334)),$ $(previous(\_2,n), previous(\_337,\_334))\}$
Target dangling	$\{return(r,l)\}$
Candidate dangling	$\emptyset$

Table 8.53: Best individual for last line, relativity limerick, double semantic fitness

The individual for the third line (Table 8.52) perfectly conveys the `relativity3` target semantics. Moreover, the metre fitness is deemed better than that of the third line in Table 8.44. Thus, McGonagall finds this solution to be better than the original line.

Unfortunately, the individual for the fourth line (Table 8.53) does not improve on the one in Table 8.49. Semantically, it is still missing the $return(r,l)$ literal, and metre-wise, it is even worse. Note that the fitness score is lower not just because the metre is worse, but the emphasis on semantics makes the lack of $return(r,l)$ all the more significant.

Finally, if we take these individually generated lines and join them together as a limerick, we obtain the following text:

There **is** a young **la**dy called **bright**.
She (will) **tra**vel much **fas**ter than **light**.
She **set** out one **day** * **re**lative**ly**.
She **is** on (a) pre**ce**ding * **night**.

This text very closely resembles our limerick in Table 8.44, and compared to the quality of the limerick that was generated in whole (Table 8.45), it is certainly an improvement. This suggests that the decomposition of the task into several smaller sized tasks improves the performance of McGonagall, and is similar to the 'chunking' function played by sentence planning in traditional NLG.

We have also seen that the very simple modification of increasing the weight of semantic evaluation in our linear combination results in a marked improvement in the quality of the generated texts. Although we do not claim that this in itself proves McGonagall's success, nevertheless it shows the potential benefit that may be gained from more advanced multi-objective optimization methods such as Pareto-optimality based and population-based approaches. This is certainly an area for future research.

8.5.5 Summary of discussion

The results from these tests, particularly the first two tests (Sections 8.5.1 and 8.5.2), show that McGonagall is encountering difficulties in achieving optimal solutions for both semantics and metre. We believe that our simple approach to multi-objective optimization used in these tests, i.e. the averaging of scores obtained from metre and semantic evaluation functions, is the limiting factor. The fact that the results obtained for optimizing metre similarity and semantic similarity simultaneously are not as good as when optimizing them independently is to be expected, due to the tradeoffs involved. In Section 8.5.4, we observed that MCGONAGALL was in fact performing better at satisfying metre than it was at satisfying semantics. This led us to experiment with the weighting of the linear combination between evaluation functions. Indeed, the simple approach of doubling the weight of semantic evaluation resulted in a marked improvement in the quality of the output. This suggests a potential benefit that may be gained from various methods of increasing sophistication, such as Pareto-optimality based and population-based approaches.

Nevertheless, we feel that the metrical and semantic quality of the best outputs obtained in Sections 8.5.2 and 8.5.4 are comparable to that of the sample output from our initial attempt at poetry generation based on chart generation (Figure 2.15). Note that chart generation is an exhaustive search method.

We believe that our EA framework provides great flexibility in the implementation of the various heuristics used to optimize the constraints of meaningfulness and poeticness, which roughly correspond to the concepts of fidelity and fluency in Oberlander and Brew (2000).

8.6 Summary

Our empirical study was divided into three stages, and the tests carried out were conducted on a principle of least bias, where for each test we started with an initial test that defined a baseline, and we gradually introduced components that were more heuristically-motivated.

In general, our study showed that MCGONAGALL is perfectly capable of generating solutions that satisfy various metre constraints. With respect to semantic constraints, it encountered difficulties when the size of the semantic target grew too large. We believe this is due to the fact that MCGONAGALL has no real account of discourse. Finally, when trying to satisfy both semantic and metre targets, i.e. in the case of poetry generation, there was clearly a tradeoff between the two dimensions. By modifying the weighting of the linear combination between evalution functions, we were able to improve performance. Although we do not claim that this in itself proves MCGONAGALL's success, nevertheless it shows the potential benefit that may be gained from more advanced multi-objective optimization.

A more detailed summary of the observations and discussions for each of the three stages of our empirical study is provided in Sections 8.3.8, 8.4.4, and 8.5.5.

Chapter 9

Conclusions and future work

In this chapter we conclude our exposition by summarizing the main contributions of this thesis in light of the context that originally motivated our research. We then propose several avenues of research that would be of interest for future work.

9.1 Conclusions

As stated in Section 1.1, there are two different aspects that motivate our work: the AI motivation and the NLG motivation. We now present our conclusions based on the motivational aspect that they most pertinently address.

1. AI motivation

   A fundamental decision that we made early on was to concentrate our efforts on a subclass of poetry which we could define in a falsifiable model (Section 2.1.4). Although our restricted definition of poetry as a text that satisfies the properties of grammaticality, meaningfulness, and poeticness, is deliberately general, we further clarified in Section 5.1 that this thesis further defines meaningfulness as propositional semantics similarity to a given target semantics, and poeticness as metre similarity to a given target form. These are the features that our implementation, MCGONAGALL, seeks to optimize. Samples of existing poetry which we consider to fall under this specific definition are presented in Figures 2.6 to 2.8.

This model allowed us to characterise existing poetry generation systems in terms of what properties their output was satisfying (Section 2.3). We found that the majority of these systems only address a subset of these properties.

We feel that the success of future poetry generation systems will hinge on their specific accounts of these three properties, particularly meaningfulness and poeticness. In general, the more sophisticated the account, the greater the potential for better quality output.

Our model of metre similarity is an important step in providing a computational model for the myriad aspects of poeticness.

Finally, our results show that our chosen mechanism of solving the state space search problem of poetry generation, evolutionary algorithms, succeeds in generating texts that achieve, either optimally or "satisfycingly", these properties. When only optimizing poeticness (Section 8.3), MCGONAGALL found solutions that managed to exhibit the metre patterns prescribed by our three target forms, `haiku`, `limerick`, and `mignonne`. When only optimizing meaningfulness (Section 8.4), MCGONAGALL found solutions that managed to convey the semantics in the `lionhalf` and `lion` targets (although for `lion`, this was only achieved with the extensively knowledge-rich SPUDLIKEDISCOURSE operator). Unfortunately, for the simultaneous optimization of meaningfulness and poeticness (Section 8.5), MCGONAGALL did not perform as well as when optimizing them in isolation. This is to be expected, given the tradeoffs involved. However, despite their suboptimality, we believe the best output obtained, exemplified by the texts found in Tables 8.37 and 8.38, are comparable to the output of our previous chart-generation based system (Figure 2.15), which employed an exhaustive search. Furthermore, by modifying the weighting of the linear combination between evalution functions, we were able to improve performance.

2. NLG motivation

In attempting to solve our poetry generation search problem using evolutionary algorithms, we have formulated an alternative technique for doing NLG. Although NLG research employing similar techniques has been done before, e.g. Mellish et al. (1998a) and Cheng (2002), it has typically been confined to a very specific subtask of the generation process, e.g. text structuring, aggregation.

The implemented system described in this thesis, MCGONAGALL, is a proof-of-concept system that can be employed as a testbed platform for other NLG tasks that require flexibility in achieving surface constraints, or where communicative goals are vague. In this thesis, we have attempted to show how it can simulate the workings of various existing NLG systems and techniques, i.e. Nicolov's PROTECTOR and Stone's SPUD (Sections 7.4.3 and 8.4.3), and Oberlander and Brew's author-reviewer model (Section 8.5.3). Note that this is not intended to be a formal comparison between MCGONAGALL and these other systems. Such an exercise is beyond the scope of this thesis, and is an interesting area of future research. Our attempts at simulating these systems in this thesis are merely intended to show the flexibility of MCGONAGALL.

We observed, however, that due to the lack of account for discourse in our genetic operators, generating multi-sentential texts, such as those required to convey our `lion` target semantics, was an extremely challenging task.

Finally, on a more technical note, this thesis proposes an LTAG-based representational scheme (Section 5.3), and an array of genetic operators using this representation (Chapter 7), for the generation of linguistic structures in the nonmonotonic fashion as required by evolutionary algorithms. In particular, these operators allow the stochastic, nonmonotonic incremental generation of texts which guarantee linguistic well-formedness (Section 7.2) and are semantically motivated (Section 7.3). We believe this is a contribution not only for poetry generation, but the NLG research field in general.

9.2 Future Work

1. **More extensive testing of parameters**

   As noted in Section 8.2, due to the nature of both our domain task and the method chosen (EAs), there were simply too many factors to test individually within the time allotted for the completion of this thesis. For our research purposes, we adopted a principle of least bias, as we are most interested in observing the mileage that can be obtained from baseline components. EA parameters and techniques such as alternative selection algorithms, population size, and niching methods (Section 4.2.5) may improve the results.

   Furthermore, running experiments with larger scale linguistic resources, e.g. the XTAG grammar (XTAG Research Group, 2001), would be a very interesting endeavour. Since good human poets command a superior mastery of their language, it is reasonable to expect that a poetry generation system with extensive linguistic resources has more opportunities to create high quality texts.

2. **Accounting for more poetic aspects:**

   In this thesis, we limited our account of poeticness to metre similarity. Phonetic aspects such as rhyme and alliteration are obvious choices of the next aspects to be accounted for. We also chose to ignore figurative language, but we believe that computational models for the comprehension of analogy (Gentner, 1983) and metonymy (Markert and Hahn, 2002) may be adapted for generation. Quinn (1982), a 'recipe book' of over 60 ways to turn a phrase using figures of speech, is also an exciting prospect of a resource that can be used.

3. **Interfacing with stylometry**

   Stylometry is a research sub-field of humanities studies which performs statistical analysis of existing literary works (Holmes, 1998). One primary example application of this field is the attributing of authorship to disputed documents. One can imagine exploiting the wealth of information obtained from research in this field as heuristic bias in our EA. Indeed, this approach is adopted, albeit rather superficially, by Kurzweil's "Cybernetic Poet" (Section 2.3.3). Techniques such as probabilistic grammars (Bod and Scha, 1997) are possibly applicable.

4. **Opportunistic content determination:**

   One of the powerful features provided by our framework is the ability to generate texts with little or no communicative goal whatsoever. Although we chose to limit the work in this thesis to that of semantic similarity to explicitly defined targets, we believe that MCGONAGALL has the potential to incorporate aspects of content determination. As with the opportunistic planning of ILEX (Section 3.6), we envisage a complete NLG system where the notion of meaningfulness is defined with respect to an underlying knowledge base. This would require considerable inference and reasoning, and metrics of coherence and possibly interestingness. Further down the line, notions of narrative (i.e. story generation, see Turner (1994)) and even humour (Binsted, 1996) are not beyond the realms of imagination.

5. **Testing of "proper" NLG applications:**

   As mentioned above, our implemented system, MCGONAGALL, can be used as a testbed platform for NLG tasks that require flexibility in achieving surface constraints, or where communicative goals are vague. Already, an early version of MCGONAGALL was used for the task of text structuring in Karamanis and Manurung (2002). Other possible research areas which are of particular interest are readability (Eddy, 2002) and personalization (Oberlander and Brew, 2000).

6. **Aspects of creativity and artistic theory, cognitive science:**

   Although this thesis has next to nothing to say about the subjects of creativity and artistic theory and cognitive science, we believe that our computational model of poetry generation may provide insight into these areas, such as an emphasis on **intention** in theories of creativity as stochastic design process, and a possibly psycholinguistically plausible model of poetry writing.

Appendix A

Target semantics and metre

In this appendix we present the target structures used in our empirical study, both for semantic similarity (Section A.1) and metre similarity (Section A.2). Although some of them have been presented in the main body of the thesis, we have compiled them here for convenience.

A.1 Semantic targets

For the majority of our tests, we used two different semantic expressions for our empirical study, `lionhalf` and `lion`, where the prior is a proper subset of the latter. They are both encodings of Hilaire Belloc's poem, *"The Lion"*, taken from his collection *"The Bad Child's Book of Beasts"* (Belloc, 1991), with a slight alteration where we have replaced the original opening noun phrase *"the lion, the lion"* with *"the african lion"* (see Table A.1).

The `lionhalf` semantic expression is our encoding of the first two lines of this poem, shown in Table A.2. The `lion` semantic expression is our encoding of the entire poem, shown in Table A.3. See Section 5.2 for a discussion of our semantic representation scheme.

The african lion, he dwells in the waste,
he has a big head and a very small waist;
but his shoulders are stark, and his jaws they are grim,
and a good little child will not play with him.

Table A.1: Our slightly altered version of Belloc's *"The Lion"*

{lion(_,l), african(_,l), dwell(d,l), inside(_,d,was), waste(_,was), own(_,l,h), head(_,h), big(_,h), own(_,l,wai), small(s,wai), waist(_,wai), very(_,s)}

Table A.2: `lionhalf` encodes the first two lines of the poem in Table A.1

{lion(_,l), african(_,l), dwell(d,l), inside(_,d,was), waste(_,was), own(_,l,h), head(_,h), big(_,h), own(_,l,wai), small(s,wai), waist(_,wai), very(_,s), own(_,l,sho), shoulders(_,sho), stark(_,sho), own(_,l,ja), jaws(_,ja), grim(_,ja), boy(_,c), good(_,c), little(_,c), play(pl,c), with(_,pl,l), will(wpl,pl), not(_,wpl)}

Table A.3: `lion` encodes all four lines of the poem in Table A.1

Additionally, in Section 8.5.4 we used a different semantic target based on Arthur H.R. Buller's "relativity" limerick, shown in Table A.4.

The semantics of this limerick are encoded in the semantic expressions shown in Table A.5. `relativity1`, `relativity2`, and `relativity4` encodes the semantics of the first, second, and last lines respectively, whereas `relativity3` encodes the semantics of the third and fourth lines. Additionally, the semantic expression `relativity` is simply the union of `relativity1`, `relativity2`, `relativity3`, and `relativity4`.

A.2 Metre targets

We used three different target forms during our empirical study, `haiku`, `limerick`, and `mignonne`, which we present in the following three sections. See Section 6.3.2 for a discussion of our rep-

There was a young lady called Bright
who could travel much faster than light.
She set out one day
in a relative way
and returned on the previous night.

Table A.4: Arthur H.R. Buller's 'relativity' limerick

`relativity1`	$\{lady(\_,l), young(\_,l), name(\_,l,b), bright(\_,b)\}$
`relativity2`	$\{travel(t,l), faster(f,t,li), light(\_,li), much(\_,f), can(\_,t)\}$
`relativity3`	$\{leave(le,l), relative(\_,le), oneday(\_,le)\}$
`relativity4`	$\{return(r,l), on(\_,r,n), night(\_,n), previous(\_,n)\}$

Table A.5: `relativity1` to `relativity4` semantic targets

Form	Sample
`[x,x,x,x,x,b`	*To convey one's mood*
`x,x,x,x,x,x,x,b`	*in seventeen syllables*
`x,x,x,x,x,b]`	*is very diffic*

Table A.6: `haiku` target form and sample haiku by John Cooper Clarke

resentation of target forms.

A.2.1 `haiku`

The haiku is a very traditional form of Japanese poetry. Traditionally, there are several properties which must be met of a haiku, for example, it must contain a *kigo*, i.e. a season word, which indicates in which season the haiku is set. For example, cherry blossoms indicate spring, snow indicates winter, and so forth. However, for our purposes, we concentrate purely on its metrical form, which is a 17-syllable verse form consisting of three lines of five, seven, and five syllables.

We chose this form as a baseline target to examine McGONAGALL's performance with the relatively simple task of obtaining a specific syllable count.

Table A.6 shows our representation of the `haiku` form along with a well-known sample haiku by John Cooper Clarke[1].

A.2.2 `limerick`

The limerick is a very well-known verse form in the English language. Variants of this form date as far back as the fourteenth century, and have typically been used in folk poetry, e.g.

[1] This is an oft-cited poem, but is unpublished. See the author's homepage at `http://www.cyberspike.com/clarke/poemlist.html`

Form	Sample
`[w,s,w,w,s,w,w,s,b,`	*There **was** an old **man** with a **beard**,*
`w,s,w,w,s,w,w,s,b,`	*who **said**, "It is **just** as i **feared**!*
`w,s,w,w,s,b,`	*Two **owls** and a **hen**,*
`w,s,w,w,s,b,`	*four **larks** and a **wren**,*
`w,s,w,w,s,w,w,s,b]`	*have **all** built their **nests** in my **beard**!"*

Table A.7: `limerick` target form and sample limerick from Lear (1947)

nursery rhymes, drinking songs, and so forth Legman (1974). The subject matter of limericks is very often bawdy. In comparatively recent times, limericks were popularized by the works of Edward Lear, such as his several volumes of *"Book of Nonsense"*, compiled in (Lear, 1947).

A limerick consists of five anapestic lines, with a rhyme scheme of *aabba*. Recall from Section 2.1.2 that an anapest consists of three syllables, where the stress falls on the last one. The first, second, and fifth lines are trimeter (i.e. three anapests), while the third and fourth are dimeter (i.e. three anapests). The first anapest on each line almost always has a missing upbeat, i.e. is only two syllables long. Note that for our purposes, we ignore the rhyme scheme.

We chose this form as it has a very well defined and easily recognizable metrical pattern.

Table A.7 shows our representation of the `limerick` form along with a sample limerick taken from Lear (1947).

Additionally, for the test in Section 8.5.4, we created two additional metre targets, `limerickline1` and `limerickline2`. These were used to enable McGONAGALL to generate a limerick on a line-by-line basis.

A.2.3 `mignonne`

This target form is based on a poem written by Clement Marot, a 16th century poet, entitled *"A une Damoyselle malade"*. It is the subject of a fascinating book on poetry translation by Douglas Hofstadter, *"Le Ton Beau De Marot"* (Hofstadter, 1997).

The poem has a very distinctive form, consisting of 28 lines, where each line consists of three syllables, with the stress falling on the last line (i.e. an anapest). In Marot's original poem, there is a rich rhyme scheme where every pair of lines ends in a rhyme. In this thesis, we ignore this rhyme scheme.

We chose this form as it is a relatively challenging task in terms of linebreaking and enjambment.

Table A.8 shows our representation of the `mignonne` form along with the original poem by Marot (taken from Hofstadter (1997).

Form	Sample
[w,w,s,b,	*Ma mi**gnonne,***
w,w,s,b,	*Je vous **donne***
w,w,s,b,	*Le bon **jour;***
w,w,s,b,	*Le sé**jour***
w,w,s,b,	*C’est pri**son.***
w,w,s,b,	*Guéri**son***
w,w,s,b,	*Recouv**rez,***
w,w,s,b,	*Puis ouv**rez***
w,w,s,b,	*Votre **porte***
w,w,s,b,	*Et qu’on **sorte***
w,w,s,b,	*Vite**ment,***
w,w,s,b,	*Car Clé**ment***
w,w,s,b,	*Le vous **mande.***
w,w,s,b,	*Va, fri**ande***
w,w,s,b,	*De ta **bouche,***
w,w,s,b,	*Qui se **couche***
w,w,s,b,	*En dan**ger***
w,w,s,b,	*Pour man**ger***
w,w,s,b,	*Confi**tures;***
w,w,s,b,	*Si tu **dures***
w,w,s,b,	*Trop ma**lade,***
w,w,s,b,	*Couleur **fade***
w,w,s,b,	*Tu pren**dras,b,***
w,w,s,b,	*Et per**dras***
w,w,s,b,	*L’embon**point.***
w,w,s,b,	*Dieu te **doint***
w,w,s,b,	*Santé **bonne,***
w,w,s,b]	*Ma mi**gnonne***

Table A.8: `mignonne` target form and original poem by Clément Marot

Appendix B

Metre compensation patterns

In this appendix we present the various metre compensation patterns and associated costs that we have devised. See Section 6.3.4 for a discussion of our context-sensitive compensation scoring mechanism.

Each pattern shown here shows two consecutive syllable alignments from a candidate syllable to a target syllable. These alignments are produced by the minimum edit distance algorithm (Section 6.3.1). Where the candidate syllable is ε, it is an insertion operation, and where the target syllable is ε, it is a deletion operation. Otherwise, it is a substitution operation. Patterns are shown using regular expression notation, where $\{\ldots|\ldots\}$ is a disjunction, and ? is a 'wildcard'.

Note that as an ideal alignment has an edit distance cost of 0, positive compensation scores are penalties and negative scores are rewards.

- Consecutive deletion

  This pattern detects the consecutive deletion of two candidate syllables, excluding the special linebreaking candidate syllable, *b*. It incurs a penalty cost of 1.

  The pattern can be seen in Figure B.1.

- Consecutive insertion

  This pattern detects the consecutive insertion of two target syllables, excluding the special linebreaking target syllable, `b`. It incurs a penalty cost of 1.

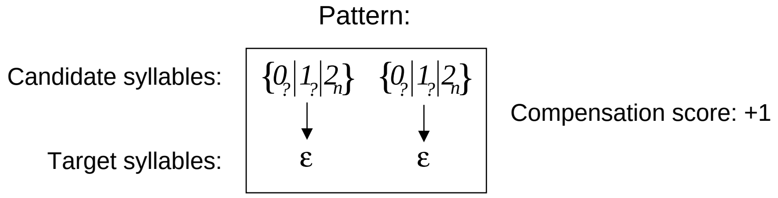


Figure B.1: Pattern and cost for consecutive deletion of candidate syllables

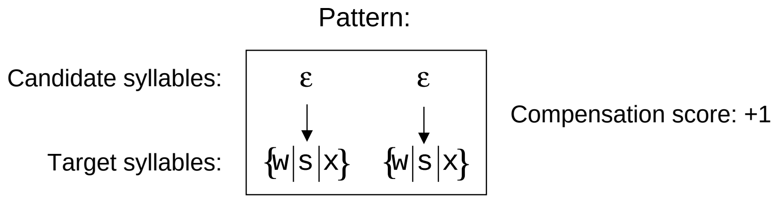


Figure B.2: Pattern and cost for consecutive insertion of target syllables

The pattern can be seen in Figure B.2.

- Natural destressing

  This pattern detects the destressing (i.e. substitution to w) of a candidate syllable that has primary stress in the CMU pronouncing dictionary (i.e. either 1_1 or 1_n) when it appears next to a strong (s) target syllable. There are two different patterns shown in Figure B.3, which account for when the destressing occurs to the left or right of the s target syllable. Each pattern earns a reward cost of 1.

- Natural stressing

  This pattern detects the stressing (i.e. substitution to s) of a candidate syllable that has no stress in the CMU pronouncing dictionary (i.e. either 0_1 or 0_n) when it appears next to a weak (w) target syllable. There are two different patterns shown in Figure B.4, which account for when the stressing occurs to the left or right of the w target syllable. Each pattern earns a reward cost of 1.

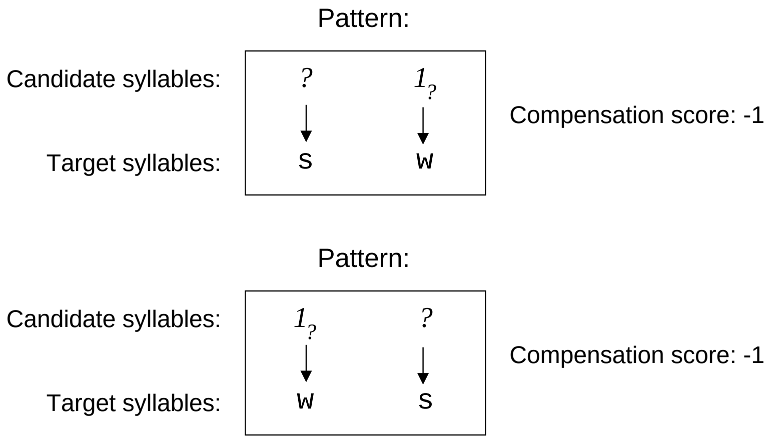


Figure B.3: Patterns and cost for natural destressing of syllables

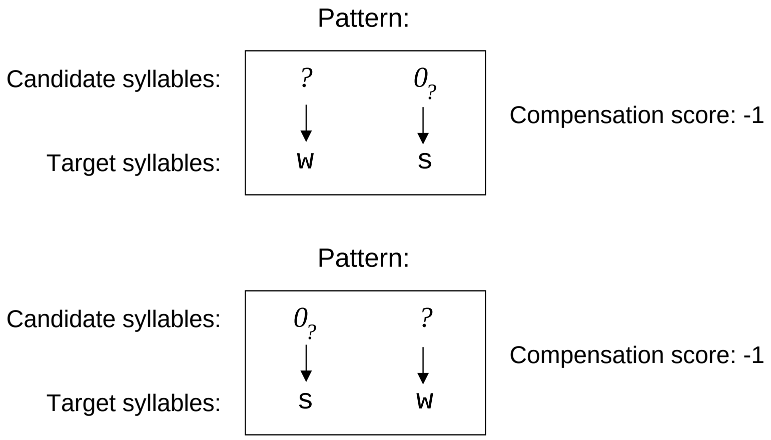


Figure B.4: Patterns and cost for natural stressing of syllables

Appendix C

Linguistic resources

In this appendix we present our handcrafted linguistic resources used during our empirical study in Chapter 8. Section C.1 presents the lexicon and Section C.2 presents the grammar.

C.1 Lexicon

In this section we present our handcrafted lexicon used during our empirical study in Chapter 8. We divide the lexicon into five sections. The first four sections contain lexical entries used for the tests in Section 8.3.1 to 8.5.3. Each section groups entries of a certain syntactic type, i.e. closed class words (Section C.1.1), nouns (Section C.1.2), verbs (Section C.1.3), and adjectives (Section C.1.4). These words are compiled from a selection of Hilaire Belloc's poems about animals, *"The Bad Child's Book of Beasts"* (Belloc, 1991).

The fifth section contains additional lexical entries used for the test in Section 8.5.4, where we attempt to generate Arthur H.R. Buller's 'relativity' limerick (see Table 8.43).

For each word we show its unique key, orthographic spelling, names of elementary trees it can anchor, lexical semantics, semantic signature, phonetic spelling, and feature structure. See Section 5.3.9 for more detail on our lexical representation.

C.1.1 Closed class words

Key	Ortho	Trees	Semantics	Signature	Phonetics	Feature
in-prep	*in*	I-C-PP, I-P-P, A-P-NP, A-P-VP, A-P-S, A-R-C-PP	inside(X,Y,Z)	$\boxed{X,Y,Z}$	*ih0,n*	$\begin{bmatrix} \text{CAT} & p \end{bmatrix}$
with-prep	*with*	I-C-PP, I-P-P, A-P-NP, A-P-VP, A-P-S, A-R-C-PP	with(X,Y,Z)	$\boxed{X,Y,Z}$	*w,ih0,dh*	$\begin{bmatrix} \text{CAT} & p \end{bmatrix}$
that-comp	*that*	I-R-Comp		□	*dh,ah0,t*	$\begin{bmatrix} \text{CAT} & comp \\ \text{SUB} & \begin{bmatrix} \text{ANIM} & - \end{bmatrix} \end{bmatrix}$
who-comp	*who*	I-R-Comp		□	*hh,uw0*	$\begin{bmatrix} \text{CAT} & comp \\ \text{SUB} & \begin{bmatrix} \text{ANIM} & + \end{bmatrix} \end{bmatrix}$
empty-det-pron		I-N-D		$\boxed{X,Y,Z}$		$\begin{bmatrix} \text{CAT} & d \\ \text{PRON} & + \\ \text{GEN} & - \end{bmatrix}$
empty-det-prop		I-N-D		$\boxed{X,Y,Z}$		$\begin{bmatrix} \text{CAT} & d \\ \text{PROP} & + \\ \text{GEN} & - \end{bmatrix}$
empty-det-plural		I-N-D		$\boxed{X,Y,Z}$		$\begin{bmatrix} \text{CAT} & d \\ \text{GEN} & - \\ \text{AGR} & \begin{bmatrix} \text{NUM} & pl \end{bmatrix} \end{bmatrix}$
empty-det-mass		I-N-D		$\boxed{X,Y,Z}$		$\begin{bmatrix} \text{CAT} & d \\ \text{GEN} & - \\ \text{AGR} & \begin{bmatrix} \text{NUM} & ms \end{bmatrix} \end{bmatrix}$
a-det	*a*	I-N-D		$\boxed{X,Y,Z}$	*ah0*	$\begin{bmatrix} \text{CAT} & d \\ \text{GEN} & - \\ \text{PRON} & - \\ \text{PROP} & - \\ \text{AGR} & \begin{bmatrix} \text{NUM} & sg \end{bmatrix} \end{bmatrix}$
the-det	*the*	I-N-D		$\boxed{X,Y,Z}$	*dh,ah0*	$\begin{bmatrix} \text{CAT} & d \\ \text{GEN} & - \\ \text{PRON} & - \\ \text{PROP} & - \end{bmatrix}$
his-det	*his*	I-N-D	own(X,$Owner$,Y)	$\boxed{X,Owner,Y}$	*hh,ih0,z*	$\begin{bmatrix} \text{PRON} & - \\ \text{PROP} & - \\ \text{GEN} & + \\ \text{CAT} & d \\ \text{AGRGEN} & \begin{bmatrix} \text{NUM} & sg \\ \text{PERS} & 3 \\ \text{GNDR} & masc \\ \text{3RDSG} & + \end{bmatrix} \\ \text{SUBGEN} & \begin{bmatrix} \text{ANIM} & + \end{bmatrix} \end{bmatrix}$

continued on next page

continued from previous page

Key	Ortho	Trees	Semantics	Signature	Phonetics	Feature
its-det	*its*	I-N-D	own(*X*,*Owner*,*Y*)	$\boxed{X,Owner,Y}$	*ih0,t,s*	$\begin{bmatrix}\textsc{pron} & - \\ \textsc{prop} & - \\ \textsc{gen} & + \\ \textsc{cat} & d \\ \textsc{agrgen} & \begin{bmatrix}\textsc{num} & sg \\ \textsc{pers} & 3 \\ \textsc{gndr} & neut \\ 3\textsc{rdsg} & +\end{bmatrix} \\ \textsc{subgen} & \begin{bmatrix}\textsc{anim} & +\end{bmatrix}\end{bmatrix}$
he-pron	*he*	I-N-N, I-C-NP, A-R-C-NP, A-N-LDS		$\boxed{X,Y}$	*hh,iy0*	$\begin{bmatrix}\textsc{cat} & n \\ \textsc{agr} & \begin{bmatrix}\textsc{num} & sg \\ \textsc{pers} & 3 \\ \textsc{gndr} & masc \\ 3\textsc{rdsg} & +\end{bmatrix} \\ \textsc{pron} & + \\ \textsc{prop} & - \\ \textsc{case} & nom \\ \textsc{sub} & \begin{bmatrix}\textsc{anim} & +\end{bmatrix}\end{bmatrix}$
him-pron	*him*	I-N-N, I-C-NP, A-R-C-NP		$\boxed{X,Y}$	*hh,ih0,m*	$\begin{bmatrix}\textsc{cat} & n \\ \textsc{agr} & \begin{bmatrix}\textsc{num} & sg \\ \textsc{pers} & 3 \\ \textsc{gndr} & masc \\ 3\textsc{rdsg} & +\end{bmatrix} \\ \textsc{pron} & + \\ \textsc{prop} & - \\ \textsc{case} & acc \\ \textsc{sub} & \begin{bmatrix}\textsc{anim} & +\end{bmatrix}\end{bmatrix}$
she-pron	*she*	I-N-N, I-C-NP, A-R-C-NP, A-N-LDS		$\boxed{X,Y}$	*sh,iy0*	$\begin{bmatrix}\textsc{cat} & n \\ \textsc{agr} & \begin{bmatrix}\textsc{num} & sg \\ \textsc{pers} & 3 \\ \textsc{gndr} & fem \\ 3\textsc{rdsg} & +\end{bmatrix} \\ \textsc{pron} & + \\ \textsc{prop} & - \\ \textsc{case} & nom \\ \textsc{sub} & \begin{bmatrix}\textsc{anim} & +\end{bmatrix}\end{bmatrix}$
her-pron	*her*	I-N-N, I-C-NP, A-R-C-NP		$\boxed{X,Y}$	*hh,er0*	$\begin{bmatrix}\textsc{cat} & n \\ \textsc{agr} & \begin{bmatrix}\textsc{num} & sg \\ \textsc{pers} & 3 \\ \textsc{gndr} & fem \\ 3\textsc{rdsg} & +\end{bmatrix} \\ \textsc{pron} & + \\ \textsc{prop} & - \\ \textsc{case} & acc \\ \textsc{sub} & \begin{bmatrix}\textsc{anim} & +\end{bmatrix}\end{bmatrix}$

continued on next page

continued from previous page

Key	Ortho	Trees	Semantics	Signature	Phonetics	Feature
it-anim-neut-pron	*it*	I-N-N, I-C-NP, A-R-C-NP, A-N-LDS		$\boxed{X,Y}$	*ih0*	$\begin{bmatrix} \text{CAT} & n \\ \text{AGR} & \begin{bmatrix} \text{NUM} & sg \\ \text{PERS} & 3 \\ \text{GNDR} & neut \\ \text{3RDSG} & + \end{bmatrix} \\ \text{PRON} & + \\ \text{PROP} & - \\ \text{SUB} & \begin{bmatrix} \text{ANIM} & + \end{bmatrix} \end{bmatrix}$
it-inanim-pron	*it*	I-N-N, I-C-NP, A-R-C-NP, A-N-LDS		$\boxed{X,Y}$	*ih0*	$\begin{bmatrix} \text{CAT} & n \\ \text{AGR} & \begin{bmatrix} \text{NUM} & sg \\ \text{PERS} & 3 \\ \text{3RDSG} & + \end{bmatrix} \\ \text{PRON} & + \\ \text{PROP} & - \\ \text{SUB} & \begin{bmatrix} \text{ANIM} & - \end{bmatrix} \end{bmatrix}$
they-pron	*they*	I-N-N, I-C-NP, A-R-C-NP, A-N-LDS		$\boxed{X,Y}$	*dh,ey0*	$\begin{bmatrix} \text{CAT} & n \\ \text{AGR} & \begin{bmatrix} \text{NUM} & pl \\ \text{PERS} & 3 \\ \text{3RDSG} & - \end{bmatrix} \\ \text{PRON} & + \\ \text{PROP} & - \\ \text{CASE} & nom \end{bmatrix}$
them-pron	*them*	I-N-N, I-C-NP, A-R-C-NP		$\boxed{X,Y}$	*dh,eh0,m*	$\begin{bmatrix} \text{CAT} & n \\ \text{AGR} & \begin{bmatrix} \text{NUM} & pl \\ \text{PERS} & 3 \\ \text{3RDSG} & - \end{bmatrix} \\ \text{PRON} & + \\ \text{PROP} & - \\ \text{CASE} & acc \end{bmatrix}$
is-cv	*is*	I-C-CV		□	*ih0,z*	$\begin{bmatrix} \text{CAT} & cv \\ \text{AGR} & \begin{bmatrix} \text{NUM} & sg \end{bmatrix} \end{bmatrix}$
are-cv	*are*	I-C-CV		□	*aa0,r*	$\begin{bmatrix} \text{CAT} & cv \\ \text{AGR} & \begin{bmatrix} \text{NUM} & pl \end{bmatrix} \end{bmatrix}$
be-cv	*be*	I-C-AuxCV		□	*b,iy0*	$\begin{bmatrix} \text{CAT} & cv \end{bmatrix}$
will-aux	*will*	A-V-Aux, I-V-Aux	will(X,Y)	$\boxed{X,Y}$	*w,ih0,l*	$\begin{bmatrix} \text{CAT} & aux \end{bmatrix}$
very-adv	*very*	A-A-Adj	very(X,Y)	$\boxed{X,Y}$	*v,eh1,r,iy0*	$\begin{bmatrix} \text{CAT} & adv \end{bmatrix}$
,-punct	,	I-M-Punct		□	—	$\begin{bmatrix} \text{CAT} & punct \end{bmatrix}$
.-sb	.	I-T-KO, A-T-SR, A-T-SL		□	—	$\begin{bmatrix} \text{CAT} & sb \end{bmatrix}$

continued on next page

continued from previous page

Key	Ortho	Trees	Semantics	Signature	Phonetics	Feature
not-neg	*not*	A-M-Neg	not(X,Y)	$\boxed{X,Y}$	*n,aa1,t*	$\begin{bmatrix}\textsc{cat} & neg\end{bmatrix}$

Table C.1: List of closed class words

C.1.2 Nouns

Key	Ortho	Trees	Semantics	Signature	Phonetics	Feature
bullets-np	*bullets*	I-N-N, A-R-C-NP, I-C-NP	bullets(X,Y)	$\boxed{X,Y}$	*b,uh1,l,ah0,t,s*	$\begin{bmatrix}\textsc{cat} & n \\ \textsc{prop} & - \\ \textsc{sub} & \begin{bmatrix}\textsc{anim} & -\end{bmatrix} \\ \textsc{agr} & \begin{bmatrix}\textsc{num} & pl \\ 3\textsc{rdsg} & - \\ \textsc{pers} & 3\end{bmatrix} \\ \textsc{pron} & -\end{bmatrix}$
epithets-np	*epithets*	I-N-N, A-R-C-NP, I-C-NP	epithets(X,Y)	$\boxed{X,Y}$	*eh1,p,ah0,th,eh2,t,s*	$\begin{bmatrix}\textsc{cat} & n \\ \textsc{prop} & - \\ \textsc{sub} & \begin{bmatrix}\textsc{anim} & -\end{bmatrix} \\ \textsc{agr} & \begin{bmatrix}\textsc{num} & pl \\ 3\textsc{rdsg} & - \\ \textsc{pers} & 3\end{bmatrix} \\ \textsc{pron} & -\end{bmatrix}$
facts-np	*facts*	I-N-N, A-R-C-NP, I-C-NP	facts(X,Y)	$\boxed{X,Y}$	*f,ae1,k,t,s*	$\begin{bmatrix}\textsc{cat} & n \\ \textsc{prop} & - \\ \textsc{sub} & \begin{bmatrix}\textsc{anim} & -\end{bmatrix} \\ \textsc{agr} & \begin{bmatrix}\textsc{num} & pl \\ 3\textsc{rdsg} & - \\ \textsc{pers} & 3\end{bmatrix} \\ \textsc{pron} & -\end{bmatrix}$
families-np	*families*	I-N-N, A-R-C-NP, I-C-NP	families(X,Y)	$\boxed{X,Y}$	*f,ae1,m,ah0,l,iy0,z*	$\begin{bmatrix}\textsc{cat} & n \\ \textsc{prop} & - \\ \textsc{sub} & \begin{bmatrix}\textsc{anim} & -\end{bmatrix} \\ \textsc{agr} & \begin{bmatrix}\textsc{num} & pl \\ 3\textsc{rdsg} & - \\ \textsc{pers} & 3\end{bmatrix} \\ \textsc{pron} & -\end{bmatrix}$
jaws-np	*jaws*	A-R-C-NP, I-N-N, I-C-NP	jaws(X,Y)	$\boxed{X,Y}$	*jh,ao1,z*	$\begin{bmatrix}\textsc{cat} & n \\ \textsc{agr} & \begin{bmatrix}\textsc{num} & pl \\ \textsc{pers} & 3 \\ 3\textsc{rdsg} & -\end{bmatrix} \\ \textsc{pron} & - \\ \textsc{prop} & - \\ \textsc{sub} & \begin{bmatrix}\textsc{anim} & -\end{bmatrix}\end{bmatrix}$

continued on next page

continued from previous page

Key	Ortho	Trees	Semantics	Signature	Phonetics	Feature
knees-np	*knees*	I-N-N, A-R-C-NP, I-C-NP	knees(*X*,*Y*)	$\boxed{X,Y}$	*n,iy1,z*	$\begin{bmatrix} \text{CAT} & n \\ \text{PROP} & - \\ \text{SUB} & \begin{bmatrix}\text{ANIM} & -\end{bmatrix} \\ \text{AGR} & \begin{bmatrix}\text{NUM} & pl \\ \text{3RDSG} & - \\ \text{PERS} & 3\end{bmatrix} \\ \text{PRON} & - \end{bmatrix}$
men-np	*men*	I-N-N, A-R-C-NP, I-C-NP	men(*X*,*Y*)	$\boxed{X,Y}$	*m,eh1,n*	$\begin{bmatrix} \text{CAT} & n \\ \text{PROP} & - \\ \text{SUB} & \begin{bmatrix}\text{ANIM} & +\end{bmatrix} \\ \text{AGR} & \begin{bmatrix}\text{NUM} & pl \\ \text{3RDSG} & - \\ \text{PERS} & 3\end{bmatrix} \\ \text{PRON} & - \end{bmatrix}$
mothers-np	*mothers*	I-N-N, A-R-C-NP, I-C-NP	mothers(*X*,*Y*)	$\boxed{X,Y}$	*m,ah1,dh,er0,z*	$\begin{bmatrix} \text{CAT} & n \\ \text{PROP} & - \\ \text{SUB} & \begin{bmatrix}\text{ANIM} & +\end{bmatrix} \\ \text{AGR} & \begin{bmatrix}\text{NUM} & pl \\ \text{3RDSG} & - \\ \text{PERS} & 3\end{bmatrix} \\ \text{PRON} & - \end{bmatrix}$
names-np	*names*	I-N-N, A-R-C-NP, I-C-NP	names(*X*,*Y*)	$\boxed{X,Y}$	*n,ey1,m,z*	$\begin{bmatrix} \text{CAT} & n \\ \text{PROP} & - \\ \text{SUB} & \begin{bmatrix}\text{ANIM} & -\end{bmatrix} \\ \text{AGR} & \begin{bmatrix}\text{NUM} & pl \\ \text{3RDSG} & - \\ \text{PERS} & 3\end{bmatrix} \\ \text{PRON} & - \end{bmatrix}$
shoulders-np	*shoulders*	A-R-C-NP, I-N-N, I-C-NP	shoulders(*X*,*Y*)	$\boxed{X,Y}$	*sh,ow1,l,d,er0,z*	$\begin{bmatrix} \text{CAT} & n \\ \text{AGR} & \begin{bmatrix}\text{NUM} & pl \\ \text{PERS} & 3 \\ \text{3RDSG} & -\end{bmatrix} \\ \text{PRON} & - \\ \text{PROP} & - \\ \text{SUB} & \begin{bmatrix}\text{ANIM} & -\end{bmatrix} \end{bmatrix}$
whiskers-np	*whiskers*	I-N-N, A-R-C-NP, I-C-NP	whiskers(*X*,*Y*)	$\boxed{X,Y}$	*w,ih1,s,k,er0,z*	$\begin{bmatrix} \text{CAT} & n \\ \text{PROP} & - \\ \text{SUB} & \begin{bmatrix}\text{ANIM} & -\end{bmatrix} \\ \text{AGR} & \begin{bmatrix}\text{NUM} & pl \\ \text{3RDSG} & - \\ \text{PERS} & 3\end{bmatrix} \\ \text{PRON} & - \end{bmatrix}$

continued on next page

continued from previous page

Key	Ortho	Trees	Semantics	Signature	Phonetics	Feature
animal-ns	*animal*	I-N-N, A-R-C-NP, I-C-NP	animal(*X*,*Y*)	$\boxed{X,Y}$	*ae1,n,ah0,m,ah0,l*	$\begin{bmatrix} \text{CAT} & n \\ \text{PROP} & - \\ \text{SUB} & \begin{bmatrix} \text{ANIM} & + \end{bmatrix} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & sg \\ \text{3RDSG} & + \\ \text{PERS} & 3 \end{bmatrix} \\ \text{PRON} & - \end{bmatrix}$
baboon-ns	*baboon*	I-N-N, A-R-C-NP, I-C-NP	baboon(*X*,*Y*)	$\boxed{X,Y}$	*b,ah0,b,uw1,n*	$\begin{bmatrix} \text{CAT} & n \\ \text{PROP} & - \\ \text{SUB} & \begin{bmatrix} \text{ANIM} & + \end{bmatrix} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & sg \\ \text{3RDSG} & + \\ \text{PERS} & 3 \end{bmatrix} \\ \text{PRON} & - \end{bmatrix}$
bean-ns	*bean*	I-N-N, A-R-C-NP, I-C-NP	bean(*X*,*Y*)	$\boxed{X,Y}$	*b,iy1,n*	$\begin{bmatrix} \text{CAT} & n \\ \text{PROP} & - \\ \text{SUB} & \begin{bmatrix} \text{ANIM} & - \end{bmatrix} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & sg \\ \text{3RDSG} & + \\ \text{PERS} & 3 \end{bmatrix} \\ \text{PRON} & - \end{bmatrix}$
bill-ns	*bill*	I-N-N, A-R-C-NP, I-C-NP	bill(*X*,*Y*)	$\boxed{X,Y}$	*b,ih1,l*	$\begin{bmatrix} \text{CAT} & n \\ \text{PROP} & - \\ \text{SUB} & \begin{bmatrix} \text{ANIM} & - \end{bmatrix} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & sg \\ \text{3RDSG} & + \\ \text{PERS} & 3 \end{bmatrix} \\ \text{PRON} & - \end{bmatrix}$
blubber-ns	*blubber*	I-N-N, A-R-C-NP, I-C-NP	blubber(*X*,*Y*)	$\boxed{X,Y}$	*b,l,ah1,b,er0*	$\begin{bmatrix} \text{CAT} & n \\ \text{PROP} & - \\ \text{SUB} & \begin{bmatrix} \text{ANIM} & - \end{bmatrix} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & sg \\ \text{3RDSG} & + \\ \text{PERS} & 3 \end{bmatrix} \\ \text{PRON} & - \end{bmatrix}$
boy-ns	*boy*	I-N-N, A-R-C-NP, I-C-NP	boy(*X*,*Y*)	$\boxed{X,Y}$	*b,oy1*	$\begin{bmatrix} \text{CAT} & n \\ \text{PROP} & - \\ \text{SUB} & \begin{bmatrix} \text{ANIM} & + \end{bmatrix} \\ \text{AGR} & \begin{bmatrix} \text{GNDR} & masc \\ \text{NUM} & sg \\ \text{3RDSG} & + \\ \text{PERS} & 3 \end{bmatrix} \\ \text{PRON} & - \end{bmatrix}$

continued on next page

continued from previous page

Key	Ortho	Trees	Semantics	Signature	Phonetics	Feature
child-ns	*child*	A-R-C-NP, I-N-N, I-C-NP	child(*X*,*Y*)	$\boxed{X,Y}$	*ch,ay1,l,d*	$\begin{bmatrix}\text{CAT} & n \\ \text{AGR} & \begin{bmatrix}\text{NUM} & sg \\ \text{PERS} & 3 \\ \text{3RDSG} & + \end{bmatrix} \\ \text{PRON} & - \\ \text{PROP} & - \\ \text{SUB} & \begin{bmatrix}\text{ANIM} & + \end{bmatrix}\end{bmatrix}$
dish-ns	*dish*	I-N-N, A-R-C-NP, I-C-NP	dish(*X*,*Y*)	$\boxed{X,Y}$	*d,ih1,sh*	$\begin{bmatrix}\text{CAT} & n \\ \text{PROP} & - \\ \text{SUB} & \begin{bmatrix}\text{ANIM} & - \end{bmatrix} \\ \text{AGR} & \begin{bmatrix}\text{NUM} & sg \\ \text{3RDSG} & + \\ \text{PERS} & 3 \end{bmatrix} \\ \text{PRON} & - \end{bmatrix}$
elephant-ns	*elephant*	I-N-N, A-R-C-NP, I-C-NP	elephant(*X*,*Y*)	$\boxed{X,Y}$	*eh1,l,ah0,f,ah0,n,t*	$\begin{bmatrix}\text{CAT} & n \\ \text{PROP} & - \\ \text{SUB} & \begin{bmatrix}\text{ANIM} & + \end{bmatrix} \\ \text{AGR} & \begin{bmatrix}\text{NUM} & sg \\ \text{3RDSG} & + \\ \text{PERS} & 3 \end{bmatrix} \\ \text{PRON} & - \end{bmatrix}$
expense-ns	*expense*	I-N-N, A-R-C-NP, I-C-NP	expense(*X*,*Y*)	$\boxed{X,Y}$	*ih0,k,s,p,eh1,n,s*	$\begin{bmatrix}\text{CAT} & n \\ \text{PROP} & - \\ \text{SUB} & \begin{bmatrix}\text{ANIM} & - \end{bmatrix} \\ \text{AGR} & \begin{bmatrix}\text{NUM} & sg \\ \text{3RDSG} & + \\ \text{PERS} & 3 \end{bmatrix} \\ \text{PRON} & - \end{bmatrix}$
fish-ns	*fish*	I-N-N, A-R-C-NP, I-C-NP	fish(*X*,*Y*)	$\boxed{X,Y}$	*f,ih1,sh*	$\begin{bmatrix}\text{CAT} & n \\ \text{PROP} & - \\ \text{SUB} & \begin{bmatrix}\text{ANIM} & + \end{bmatrix} \\ \text{AGR} & \begin{bmatrix}\text{NUM} & sg \\ \text{3RDSG} & + \\ \text{PERS} & 3 \end{bmatrix} \\ \text{PRON} & - \end{bmatrix}$
frog-ns	*frog*	I-N-N, A-R-C-NP, I-C-NP	frog(*X*,*Y*)	$\boxed{X,Y}$	*f,r,aa1,g*	$\begin{bmatrix}\text{CAT} & n \\ \text{PROP} & - \\ \text{SUB} & \begin{bmatrix}\text{ANIM} & + \end{bmatrix} \\ \text{AGR} & \begin{bmatrix}\text{NUM} & sg \\ \text{3RDSG} & + \\ \text{PERS} & 3 \end{bmatrix} \\ \text{PRON} & - \end{bmatrix}$

continued on next page

continued from previous page

Key	Ortho	Trees	Semantics	Signature	Phonetics	Feature
gap-ns	*gap*	I-N-N, A-R-C-NP, I-C-NP	gap(X,Y)	$\boxed{X,Y}$	*g,ae1,p*	$\left[\begin{array}{ll} \text{CAT} & n \\ \text{PROP} & - \\ \text{SUB} & \left[\begin{array}{ll} \text{ANIM} & - \end{array}\right] \\ \text{AGR} & \left[\begin{array}{ll} \text{NUM} & sg \\ 3\text{RDSG} & + \\ \text{PERS} & 3 \end{array}\right] \\ \text{PRON} & - \end{array}\right]$
grin-ns	*grin*	I-N-N, A-R-C-NP, I-C-NP	grin(X,Y)	$\boxed{X,Y}$	*g,r,ih1,n*	$\left[\begin{array}{ll} \text{CAT} & n \\ \text{PROP} & - \\ \text{SUB} & \left[\begin{array}{ll} \text{ANIM} & - \end{array}\right] \\ \text{AGR} & \left[\begin{array}{ll} \text{NUM} & sg \\ 3\text{RDSG} & + \\ \text{PERS} & 3 \end{array}\right] \\ \text{PRON} & - \end{array}\right]$
hand-ns	*hand*	I-N-N, A-R-C-NP, I-C-NP	hand(X,Y)	$\boxed{X,Y}$	*hh,ae1,n,d*	$\left[\begin{array}{ll} \text{CAT} & n \\ \text{PROP} & - \\ \text{SUB} & \left[\begin{array}{ll} \text{ANIM} & - \end{array}\right] \\ \text{AGR} & \left[\begin{array}{ll} \text{NUM} & sg \\ 3\text{RDSG} & + \\ \text{PERS} & 3 \end{array}\right] \\ \text{PRON} & - \end{array}\right]$
head-ns	*head*	A-R-C-NP, I-N-N, I-C-NP	head(X,Y)	$\boxed{X,Y}$	*hh,eh1,d*	$\left[\begin{array}{ll} \text{CAT} & n \\ \text{AGR} & \left[\begin{array}{ll} \text{NUM} & sg \\ \text{PERS} & 3 \\ 3\text{RDSG} & + \end{array}\right] \\ \text{PRON} & - \\ \text{PROP} & - \\ \text{SUB} & \left[\begin{array}{ll} \text{ANIM} & - \end{array}\right] \end{array}\right]$
hippopotamus-ns	*hippopotamus*	I-N-N, A-R-C-NP, I-C-NP	hippopotamus(X,Y)	$\boxed{X,Y}$	*hh,ih2,p,ah0,p,aa1,t,ah0,m,ah0,s*	$\left[\begin{array}{ll} \text{CAT} & n \\ \text{PROP} & - \\ \text{SUB} & \left[\begin{array}{ll} \text{ANIM} & + \end{array}\right] \\ \text{AGR} & \left[\begin{array}{ll} \text{NUM} & sg \\ 3\text{RDSG} & + \\ \text{PERS} & 3 \end{array}\right] \\ \text{PRON} & - \end{array}\right]$
lion-ns	*lion*	A-R-C-NP, I-N-N, I-C-NP	lion(X,Y)	$\boxed{X,Y}$	*l,ay1,ah0,n*	$\left[\begin{array}{ll} \text{CAT} & n \\ \text{AGR} & \left[\begin{array}{ll} \text{NUM} & sg \\ \text{PERS} & 3 \\ \text{GNDR} & neut \\ 3\text{RDSG} & + \end{array}\right] \\ \text{PRON} & - \\ \text{PROP} & - \\ \text{SUB} & \left[\begin{array}{ll} \text{ANIM} & + \end{array}\right] \end{array}\right]$

continued on next page

continued from previous page

Key	Ortho	Trees	Semantics	Signature	Phonetics	Feature
man-ns	*man*	I-N-N, A-R-C-NP, I-C-NP	man(X,Y)	$\boxed{X,Y}$	*m,ae1,n*	$\begin{bmatrix}\textsc{cat} & n\\ \textsc{prop} & -\\ \textsc{sub} & \begin{bmatrix}\textsc{anim} & +\end{bmatrix}\\ \textsc{agr} & \begin{bmatrix}\textsc{gndr} & masc\\ \textsc{num} & sg\\ 3\textsc{rdsg} & +\\ \textsc{pers} & 3\end{bmatrix}\\ \textsc{pron} & -\end{bmatrix}$
mind-ns	*mind*	I-N-N, A-R-C-NP, I-C-NP	mind(X,Y)	$\boxed{X,Y}$	*m,ay1,n,d*	$\begin{bmatrix}\textsc{cat} & n\\ \textsc{prop} & -\\ \textsc{sub} & \begin{bmatrix}\textsc{anim} & -\end{bmatrix}\\ \textsc{agr} & \begin{bmatrix}\textsc{num} & sg\\ 3\textsc{rdsg} & +\\ \textsc{pers} & 3\end{bmatrix}\\ \textsc{pron} & -\end{bmatrix}$
pole-ns	*pole*	I-N-N, A-R-C-NP, I-C-NP	pole(X,Y)	$\boxed{X,Y}$	*p,ow1,l*	$\begin{bmatrix}\textsc{cat} & n\\ \textsc{prop} & -\\ \textsc{sub} & \begin{bmatrix}\textsc{anim} & -\end{bmatrix}\\ \textsc{agr} & \begin{bmatrix}\textsc{num} & sg\\ 3\textsc{rdsg} & +\\ \textsc{pers} & 3\end{bmatrix}\\ \textsc{pron} & -\end{bmatrix}$
product-ns	*product*	I-N-N, A-R-C-NP, I-C-NP	product(X,Y)	$\boxed{X,Y}$	*p,r,aa1,d,ah0,k,t*	$\begin{bmatrix}\textsc{cat} & n\\ \textsc{prop} & -\\ \textsc{sub} & \begin{bmatrix}\textsc{anim} & -\end{bmatrix}\\ \textsc{agr} & \begin{bmatrix}\textsc{num} & sg\\ 3\textsc{rdsg} & +\\ \textsc{pers} & 3\end{bmatrix}\\ \textsc{pron} & -\end{bmatrix}$
sense-ns	*sense*	I-N-N, A-R-C-NP, I-C-NP	sense(X,Y)	$\boxed{X,Y}$	*s,eh1,n,s*	$\begin{bmatrix}\textsc{cat} & n\\ \textsc{prop} & -\\ \textsc{sub} & \begin{bmatrix}\textsc{anim} & -\end{bmatrix}\\ \textsc{agr} & \begin{bmatrix}\textsc{num} & sg\\ 3\textsc{rdsg} & +\\ \textsc{pers} & 3\end{bmatrix}\\ \textsc{pron} & -\end{bmatrix}$
skin-ns	*skin*	I-N-N, A-R-C-NP, I-C-NP	skin(X,Y)	$\boxed{X,Y}$	*s,k,ih1,n*	$\begin{bmatrix}\textsc{cat} & n\\ \textsc{prop} & -\\ \textsc{sub} & \begin{bmatrix}\textsc{anim} & -\end{bmatrix}\\ \textsc{agr} & \begin{bmatrix}\textsc{num} & sg\\ 3\textsc{rdsg} & +\\ \textsc{pers} & 3\end{bmatrix}\\ \textsc{pron} & -\end{bmatrix}$

continued on next page

continued from previous page

Key	Ortho	Trees	Semantics	Signature	Phonetics	Feature
soil-ns	*soil*	I-N-N, A-R-C-NP, I-C-NP	soil(*X*,*Y*)	$\boxed{X,Y}$	*s,oy1,l*	$\begin{bmatrix} \text{CAT} & n \\ \text{PROP} & - \\ \text{SUB} & \begin{bmatrix} \text{ANIM} & - \end{bmatrix} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & sg \\ \text{3RDSG} & + \\ \text{PERS} & 3 \end{bmatrix} \\ \text{PRON} & - \end{bmatrix}$
species-ns	*species*	I-N-N, A-R-C-NP, I-C-NP	species(*X*,*Y*)	$\boxed{X,Y}$	*s,p,iy1,sh,iy0,z*	$\begin{bmatrix} \text{CAT} & n \\ \text{PROP} & - \\ \text{SUB} & \begin{bmatrix} \text{ANIM} & - \end{bmatrix} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & sg \\ \text{3RDSG} & + \\ \text{PERS} & 3 \end{bmatrix} \\ \text{PRON} & - \end{bmatrix}$
table-ns	*table*	I-N-N, A-R-C-NP, I-C-NP	table(*X*,*Y*)	$\boxed{X,Y}$	*t,ey1,b,ah0,l*	$\begin{bmatrix} \text{CAT} & n \\ \text{PROP} & - \\ \text{SUB} & \begin{bmatrix} \text{ANIM} & - \end{bmatrix} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & sg \\ \text{3RDSG} & + \\ \text{PERS} & 3 \end{bmatrix} \\ \text{PRON} & - \end{bmatrix}$
tail-ns	*tail*	I-N-N, A-R-C-NP, I-C-NP	tail(*X*,*Y*)	$\boxed{X,Y}$	*t,ey1,l*	$\begin{bmatrix} \text{CAT} & n \\ \text{PROP} & - \\ \text{SUB} & \begin{bmatrix} \text{ANIM} & - \end{bmatrix} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & sg \\ \text{3RDSG} & + \\ \text{PERS} & 3 \end{bmatrix} \\ \text{PRON} & - \end{bmatrix}$
thing-ns	*thing*	I-N-N, A-R-C-NP, I-C-NP	thing(*X*,*Y*)	$\boxed{X,Y}$	*th,ih1,ng*	$\begin{bmatrix} \text{CAT} & n \\ \text{PROP} & - \\ \text{SUB} & \begin{bmatrix} \text{ANIM} & - \end{bmatrix} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & sg \\ \text{3RDSG} & + \\ \text{PERS} & 3 \end{bmatrix} \\ \text{PRON} & - \end{bmatrix}$
tiger-ns	*tiger*	I-N-N, A-R-C-NP, I-C-NP	tiger(*X*,*Y*)	$\boxed{X,Y}$	*t,ay1,g,er0*	$\begin{bmatrix} \text{CAT} & n \\ \text{PROP} & - \\ \text{SUB} & \begin{bmatrix} \text{ANIM} & + \end{bmatrix} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & sg \\ \text{3RDSG} & + \\ \text{PERS} & 3 \end{bmatrix} \\ \text{PRON} & - \end{bmatrix}$

continued on next page

continued from previous page

Key	Ortho	Trees	Semantics	Signature	Phonetics	Feature
toad-ns	*toad*	I-N-N, A-R-C-NP, I-C-NP	toad(*X*,*Y*)	$\boxed{X,Y}$	*t,ow1,d*	$\begin{bmatrix}\textsc{cat} & n\\ \textsc{prop} & -\\ \textsc{sub} & \begin{bmatrix}\textsc{anim} & +\end{bmatrix}\\ \textsc{agr} & \begin{bmatrix}\textsc{num} & sg\\ 3\textsc{rdsg} & +\\ \textsc{pers} & 3\end{bmatrix}\\ \textsc{pron} & -\end{bmatrix}$
town-ns	*town*	I-N-N, A-R-C-NP, I-C-NP	town(*X*,*Y*)	$\boxed{X,Y}$	*t,aw1,n*	$\begin{bmatrix}\textsc{cat} & n\\ \textsc{prop} & -\\ \textsc{sub} & \begin{bmatrix}\textsc{anim} & -\end{bmatrix}\\ \textsc{agr} & \begin{bmatrix}\textsc{num} & sg\\ 3\textsc{rdsg} & +\\ \textsc{pers} & 3\end{bmatrix}\\ \textsc{pron} & -\end{bmatrix}$
treatment-ns	*treatment*	I-N-N, A-R-C-NP, I-C-NP	treatment(*X*,*Y*)	$\boxed{X,Y}$	*t,r,iy1,t,m,ah0,n,t*	$\begin{bmatrix}\textsc{cat} & n\\ \textsc{prop} & -\\ \textsc{sub} & \begin{bmatrix}\textsc{anim} & -\end{bmatrix}\\ \textsc{agr} & \begin{bmatrix}\textsc{num} & sg\\ 3\textsc{rdsg} & +\\ \textsc{pers} & 3\end{bmatrix}\\ \textsc{pron} & -\end{bmatrix}$
trouble-ns	*trouble*	I-N-N, A-R-C-NP, I-C-NP	trouble(*X*,*Y*)	$\boxed{X,Y}$	*t,r,ah1,b,ah0,l*	$\begin{bmatrix}\textsc{cat} & n\\ \textsc{prop} & -\\ \textsc{sub} & \begin{bmatrix}\textsc{anim} & -\end{bmatrix}\\ \textsc{agr} & \begin{bmatrix}\textsc{num} & sg\\ 3\textsc{rdsg} & +\\ \textsc{pers} & 3\end{bmatrix}\\ \textsc{pron} & -\end{bmatrix}$
trunk-ns	*trunk*	I-N-N, A-R-C-NP, I-C-NP	trunk(*X*,*Y*)	$\boxed{X,Y}$	*t,r,ah1,ng,k*	$\begin{bmatrix}\textsc{cat} & n\\ \textsc{prop} & -\\ \textsc{sub} & \begin{bmatrix}\textsc{anim} & -\end{bmatrix}\\ \textsc{agr} & \begin{bmatrix}\textsc{num} & sg\\ 3\textsc{rdsg} & +\\ \textsc{pers} & 3\end{bmatrix}\\ \textsc{pron} & -\end{bmatrix}$
waist-ns	*waist*	A-R-C-NP, I-N-N, I-C-NP	waist(*X*,*Y*)	$\boxed{X,Y}$	*w,ey1,s,t*	$\begin{bmatrix}\textsc{cat} & n\\ \textsc{agr} & \begin{bmatrix}\textsc{num} & sg\\ \textsc{pers} & 3\\ 3\textsc{rdsg} & +\end{bmatrix}\\ \textsc{pron} & -\\ \textsc{prop} & -\\ \textsc{sub} & \begin{bmatrix}\textsc{anim} & -\end{bmatrix}\end{bmatrix}$

continued on next page

continued from previous page

Key	Ortho	Trees	Semantics	Signature	Phonetics	Feature
waste-ns	*waste*	A-R-C-NP, I-N-N, I-C-NP	waste(*X*,*Y*)	$\boxed{X,Y}$	*w,ey1,s,t*	$\begin{bmatrix}\text{CAT} & n\\ \text{AGR} & \begin{bmatrix}\text{NUM} & sg\\ \text{PERS} & 3\\ \text{3RDSG} & +\end{bmatrix}\\ \text{PRON} & -\\ \text{PROP} & -\\ \text{SUB} & \begin{bmatrix}\text{ANIM} & -\end{bmatrix}\end{bmatrix}$
whale-ns	*whale*	I-N-N, A-R-C-NP, I-C-NP	whale(*X*,*Y*)	$\boxed{X,Y}$	*w,ey1,l*	$\begin{bmatrix}\text{CAT} & n\\ \text{PROP} & -\\ \text{SUB} & \begin{bmatrix}\text{ANIM} & +\end{bmatrix}\\ \text{AGR} & \begin{bmatrix}\text{NUM} & sg\\ \text{3RDSG} & +\\ \text{PERS} & 3\end{bmatrix}\\ \text{PRON} & -\end{bmatrix}$

Table C.2: List of nouns

C.1.3 Verbs

Key	Ortho	Trees	Semantics	Signature	Phonetics	Feature
boil-vt	*boil*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	boil(*X*,*Y*,*Z*)	$\boxed{X,Y,Z}$	*b,oy1,l*	$\begin{bmatrix}\text{CAT} & v\\ \text{AGR} & \begin{bmatrix}\text{3RDSG} & -\end{bmatrix}\end{bmatrix}$
boiled-vt	*boiled*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	boil(*X*,*Y*,*Z*)	$\boxed{X,Y,Z}$	*b,oy1,l,d*	$\begin{bmatrix}\text{CAT} & v\\ \text{AGR} & \end{bmatrix}$
boils-vt	*boils*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	boil(*X*,*Y*,*Z*)	$\boxed{X,Y,Z}$	*b,oy1,l,z*	$\begin{bmatrix}\text{CAT} & v\\ \text{AGR} & \begin{bmatrix}\text{3RDSG} & +\end{bmatrix}\end{bmatrix}$
call-vd	*call*	I-V-D	call(*W*,*X*,*Y*,*Z*)	$\boxed{W,X,Y,Z}$	*k,ao1,l*	$\begin{bmatrix}\text{CAT} & v\\ \text{AGR} & \begin{bmatrix}\text{3RDSG} & -\end{bmatrix}\end{bmatrix}$
called-vd	*called*	I-V-D	call(*W*,*X*,*Y*,*Z*)	$\boxed{W,X,Y,Z}$	*k,ao1,l,d*	$\begin{bmatrix}\text{CAT} & v\\ \text{AGR} & \end{bmatrix}$
calls-vd	*calls*	I-V-D	call(*W*,*X*,*Y*,*Z*)	$\boxed{W,X,Y,Z}$	*k,ao1,l,z*	$\begin{bmatrix}\text{CAT} & v\\ \text{AGR} & \begin{bmatrix}\text{3RDSG} & +\end{bmatrix}\end{bmatrix}$
cut-vt	*cut*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	cut(*X*,*Y*,*Z*)	$\boxed{X,Y,Z}$	*k,ah1,t*	$\begin{bmatrix}\text{CAT} & v\\ \text{AGR} & \begin{bmatrix}\text{3RDSG} & -\end{bmatrix}\end{bmatrix}$

continued on next page

continued from previous page

Key	Ortho	Trees	Semantics	Signature	Phonetics	Feature
cuts-vt	*cuts*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	cut(*X*,*Y*,*Z*)	$\boxed{X,Y,Z}$	*k,ah1,t,s*	$\begin{bmatrix} \text{CAT} & v \\ \text{AGR} & \begin{bmatrix} \text{3RDSG} & + \end{bmatrix} \end{bmatrix}$
dwells-vipp	*dwells*	A-R-V-IPP, I-V-IPP	dwell(*X*,*Y*)	$\boxed{X,Y}$	*d,w,eh1,l,z*	$\begin{bmatrix} \text{CAT} & v \\ \text{AGR} & \begin{bmatrix} \text{3RDSG} & + \end{bmatrix} \\ \text{SUBSUB} & \begin{bmatrix} \text{ANIM} & + \end{bmatrix} \end{bmatrix}$
find-vt	*find*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	find(*X*,*Y*,*Z*)	$\boxed{X,Y,Z}$	*f,ay1,n,d*	$\begin{bmatrix} \text{CAT} & v \\ \text{AGR} & \begin{bmatrix} \text{3RDSG} & - \end{bmatrix} \end{bmatrix}$
found-vt	*found*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	find(*X*,*Y*,*Z*)	$\boxed{X,Y,Z}$	*f,aw1,n,d*	$\begin{bmatrix} \text{CAT} & v \\ \text{AGR} & \end{bmatrix}$
finds-vt	*finds*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	find(*X*,*Y*,*Z*)	$\boxed{X,Y,Z}$	*f,ay1,n,d,z*	$\begin{bmatrix} \text{CAT} & v \\ \text{AGR} & \begin{bmatrix} \text{3RDSG} & + \end{bmatrix} \end{bmatrix}$
flatten-vt	*flatten*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	flatten(*X*,*Y*,*Z*)	$\boxed{X,Y,Z}$	*f,l,ae1,t,ah0,n*	$\begin{bmatrix} \text{CAT} & v \\ \text{AGR} & \begin{bmatrix} \text{3RDSG} & - \end{bmatrix} \end{bmatrix}$
flattened-vt	*flattened*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	flatten(*X*,*Y*,*Z*)	$\boxed{X,Y,Z}$	*f,l,ae1,t,ah0,n,d*	$\begin{bmatrix} \text{CAT} & v \\ \text{AGR} & \end{bmatrix}$
goes-vipp	*goes*	I-V-IPP, A-R-V-IPP	go(*X*,*Y*)	$\boxed{X,Y}$	*g,ow1,z*	$\begin{bmatrix} \text{CAT} & v \\ \text{AGR} & \begin{bmatrix} \text{3RDSG} & + \end{bmatrix} \end{bmatrix}$
grew-vt	*grew*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	grow(*X*,*Y*,*Z*)	$\boxed{X,Y,Z}$	*g,r,uw1*	$\begin{bmatrix} \text{CAT} & v \\ \text{AGR} & \end{bmatrix}$
grow-vt	*grow*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	grow(*X*,*Y*,*Z*)	$\boxed{X,Y,Z}$	*g,r,ow1*	$\begin{bmatrix} \text{CAT} & v \\ \text{AGR} & \begin{bmatrix} \text{3RDSG} & - \end{bmatrix} \end{bmatrix}$
grows-vt	*grows*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	grow(*X*,*Y*,*Z*)	$\boxed{X,Y,Z}$	*g,r,ow1,z*	$\begin{bmatrix} \text{CAT} & v \\ \text{AGR} & \begin{bmatrix} \text{3RDSG} & + \end{bmatrix} \end{bmatrix}$

continued on next page

continued from previous page

Key	Ortho	Trees	Semantics	Signature	Phonetics	Feature
has-vt	*has*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	own(*X,Y,Z*)	$\boxed{X,Y,Z}$	*hh,ae1,z*	$\begin{bmatrix}\text{CAT} & v \\ \text{AGR} & \begin{bmatrix}\text{3RDSG} & +\end{bmatrix}\end{bmatrix}$
keep-vt	*keep*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	keep(*X,Y,Z*)	$\boxed{X,Y,Z}$	*k,iy1,p*	$\begin{bmatrix}\text{CAT} & v \\ \text{AGR} & \begin{bmatrix}\text{3RDSG} & -\end{bmatrix}\end{bmatrix}$
keeps-vt	*keeps*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	keep(*X,Y,Z*)	$\boxed{X,Y,Z}$	*k,iy1,p,s*	$\begin{bmatrix}\text{CAT} & v \\ \text{AGR} & \begin{bmatrix}\text{3RDSG} & +\end{bmatrix}\end{bmatrix}$
kept-vt	*kept*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	keep(*X,Y,Z*)	$\boxed{X,Y,Z}$	*k,eh1,p,t*	$\begin{bmatrix}\text{CAT} & v \\ \text{AGR} & \end{bmatrix}$
melt-vt	*melt*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	melt(*X,Y,Z*)	$\boxed{X,Y,Z}$	*m,eh1,l,t*	$\begin{bmatrix}\text{CAT} & v \\ \text{AGR} & \begin{bmatrix}\text{3RDSG} & -\end{bmatrix}\end{bmatrix}$
melted-vt	*melted*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	melt(*X,Y,Z*)	$\boxed{X,Y,Z}$	*m,eh1,l,t,ah0,d*	$\begin{bmatrix}\text{CAT} & v \\ \text{AGR} & \end{bmatrix}$
melts-vt	*melts*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	melt(*X,Y,Z*)	$\boxed{X,Y,Z}$	*m,eh1,l,t,s*	$\begin{bmatrix}\text{CAT} & v \\ \text{AGR} & \begin{bmatrix}\text{3RDSG} & +\end{bmatrix}\end{bmatrix}$
play-vi	*play*	A-R-V-I, I-V-I	play(*X,Y*)	$\boxed{X,Y}$	*p,l,ey1*	$\begin{bmatrix}\text{CAT} & v \\ \text{AGR} & \begin{bmatrix}\text{3RDSG} & -\end{bmatrix} \\ \text{SUBSUB} & \begin{bmatrix}\text{ANIM} & +\end{bmatrix}\end{bmatrix}$
ruminated-vi	*ruminated*	I-V-I, A-R-V-I	ruminate(*X,Y*)	$\boxed{X,Y}$	*r,uw1,m,ah0,n,ey2,t,ah0,d*	$\begin{bmatrix}\text{CAT} & v \\ \text{AGR} & \end{bmatrix}$
serve-vt	*serve*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	serve(*X,Y,Z*)	$\boxed{X,Y,Z}$	*s,er1,v*	$\begin{bmatrix}\text{CAT} & v \\ \text{AGR} & \begin{bmatrix}\text{3RDSG} & -\end{bmatrix}\end{bmatrix}$
served-vt	*served*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	serve(*X,Y,Z*)	$\boxed{X,Y,Z}$	*s,er1,v,d*	$\begin{bmatrix}\text{CAT} & v \\ \text{AGR} & \end{bmatrix}$

continued on next page

continued from previous page

Key	Ortho	Trees	Semantics	Signature	Phonetics	Feature
serves-vt	*serves*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	serve(*X*,*Y*,*Z*)	$\boxed{X,Y,Z}$	*s,er1,v,z*	$\begin{bmatrix}\text{CAT} & v\\ \text{AGR} & \begin{bmatrix}\text{3RDSG} & +\end{bmatrix}\end{bmatrix}$
shoot-vt	*shoot*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	shoot(*X*,*Y*,*Z*)	$\boxed{X,Y,Z}$	*sh,uw1,t*	$\begin{bmatrix}\text{CAT} & v\\ \text{AGR} & \begin{bmatrix}\text{3RDSG} & -\end{bmatrix}\end{bmatrix}$
shoots-vt	*shoots*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	shoot(*X*,*Y*,*Z*)	$\boxed{X,Y,Z}$	*sh,uw1,t,s*	$\begin{bmatrix}\text{CAT} & v\\ \text{AGR} & \begin{bmatrix}\text{3RDSG} & +\end{bmatrix}\end{bmatrix}$
shot-vt	*shot*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	shoot(*X*,*Y*,*Z*)	$\boxed{X,Y,Z}$	*sh,aa1,t*	$\begin{bmatrix}\text{CAT} & v\\ \text{AGR} & \end{bmatrix}$
survive-vi	*survive*	I-V-I, A-R-V-I	survive(*X*,*Y*)	$\boxed{X,Y}$	*s,er0,v,ay1,v*	$\begin{bmatrix}\text{CAT} & v\\ \text{AGR} & \begin{bmatrix}\text{3RDSG} & -\end{bmatrix}\end{bmatrix}$
survived-vi	*survived*	I-V-I, A-R-V-I	survive(*X*,*Y*)	$\boxed{X,Y}$	*s,er0,v,ay1,v,d*	$\begin{bmatrix}\text{CAT} & v\\ \text{AGR} & \end{bmatrix}$
survives-vi	*survives*	I-V-I, A-R-V-I	survive(*X*,*Y*)	$\boxed{X,Y}$	*s,er0,v,ay1,v,z*	$\begin{bmatrix}\text{CAT} & v\\ \text{AGR} & \begin{bmatrix}\text{3RDSG} & +\end{bmatrix}\end{bmatrix}$
use-vt	*use*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	use(*X*,*Y*,*Z*)	$\boxed{X,Y,Z}$	*y,uw1,s*	$\begin{bmatrix}\text{CAT} & v\\ \text{AGR} & \begin{bmatrix}\text{3RDSG} & -\end{bmatrix}\end{bmatrix}$
used-vt	*used*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	use(*X*,*Y*,*Z*)	$\boxed{X,Y,Z}$	*y,uw1,z,d*	$\begin{bmatrix}\text{CAT} & v\\ \text{AGR} & \end{bmatrix}$
uses-vt	*uses*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	use(*X*,*Y*,*Z*)	$\boxed{X,Y,Z}$	*y,uw1,s,ah0,z*	$\begin{bmatrix}\text{CAT} & v\\ \text{AGR} & \begin{bmatrix}\text{3RDSG} & +\end{bmatrix}\end{bmatrix}$
wander-vipp	*wander*	I-V-IPP, A-R-V-IPP	wander(*X*,*Y*)	$\boxed{X,Y}$	*w,aa1,n,d,er0*	$\begin{bmatrix}\text{CAT} & v\\ \text{AGR} & \begin{bmatrix}\text{3RDSG} & -\end{bmatrix}\end{bmatrix}$
wandered-vipp	*wandered*	I-V-IPP, A-R-V-IPP	wander(*X*,*Y*)	$\boxed{X,Y}$	*w,aa1,n,d,er0,d*	$\begin{bmatrix}\text{CAT} & v\\ \text{AGR} & \end{bmatrix}$
wanders-vipp	*wanders*	I-V-IPP, A-R-V-IPP	wander(*X*,*Y*)	$\boxed{X,Y}$	*w,aa1,n,d,er0,z*	$\begin{bmatrix}\text{CAT} & v\\ \text{AGR} & \begin{bmatrix}\text{3RDSG} & +\end{bmatrix}\end{bmatrix}$
want-vt	*want*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	want(*X*,*Y*,*Z*)	$\boxed{X,Y,Z}$	*w,aa1,n,t*	$\begin{bmatrix}\text{CAT} & v\\ \text{AGR} & \begin{bmatrix}\text{3RDSG} & -\end{bmatrix}\end{bmatrix}$

continued on next page

continued from previous page

Key	Ortho	Trees	Semantics	Signature	Phonetics	Feature
wanted-vt	*wanted*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	want(X,Y,Z)	$\boxed{X,Y,Z}$	*w,ao1,n,t,ah0,d*	$\begin{bmatrix}\textsc{cat} & v \\ \textsc{agr} & \end{bmatrix}$
wants-vt	*wants*	I-V-T, A-R-V-T-Sub, A-R-V-T-Obj	want(X,Y,Z)	$\boxed{X,Y,Z}$	*w,aa1,n,t,s*	$\begin{bmatrix}\textsc{cat} & v \\ \textsc{agr} & \begin{bmatrix}\text{3}\textsc{rdsg} & + \end{bmatrix}\end{bmatrix}$

Table C.3: List of verbs

C.1.4 Adjectives

Key	Ortho	Trees	Semantics	Signature	Phonetics	Feature
african-a	*african*	A-R-C-A, I-C-A, A-N-A	african(X,Y)	$\boxed{X,Y}$	*ae1,f,r,ih0,k,ah0,n*	$\begin{bmatrix}\textsc{cat} & a\end{bmatrix}$
bandy-a	*bandy*	A-N-A, A-R-C-A, I-C-A	bandy(X,Y)	$\boxed{X,Y}$	*b,ae1,n,d,iy0*	$\begin{bmatrix}\textsc{cat} & a\end{bmatrix}$
big-a	*big*	A-R-C-A, I-C-A, A-N-A	big(X,Y)	$\boxed{X,Y}$	*b,ih1,g*	$\begin{bmatrix}\textsc{cat} & a\end{bmatrix}$
extinct-a	*extinct*	A-N-A, A-R-C-A, I-C-A	extinct(X,Y)	$\boxed{X,Y}$	*ih0,k,s,t,ih1,ng,k,t*	$\begin{bmatrix}\textsc{cat} & a\end{bmatrix}$
good-a	*good*	A-R-C-A, I-C-A, A-N-A	good(X,Y)	$\boxed{X,Y}$	*g,uh1,d*	$\begin{bmatrix}\textsc{cat} & a\end{bmatrix}$
grim-a	*grim*	A-R-C-A, I-C-A, A-N-A	grim(X,Y)	$\boxed{X,Y}$	*g,r,ih1,m*	$\begin{bmatrix}\textsc{cat} & a\end{bmatrix}$
kind-a	*kind*	A-N-A, A-R-C-A, I-C-A	kind(X,Y)	$\boxed{X,Y}$	*k,ay1,n,d*	$\begin{bmatrix}\textsc{cat} & a\end{bmatrix}$
large-a	*large*	A-N-A, A-R-C-A, I-C-A	large(X,Y)	$\boxed{X,Y}$	*l,aa1,r,jh*	$\begin{bmatrix}\textsc{cat} & a\end{bmatrix}$
little-a	*little*	A-R-C-A, I-C-A, A-N-A	little(X,Y)	$\boxed{X,Y}$	*l,ih1,t,ah0,l*	$\begin{bmatrix}\textsc{cat} & a\end{bmatrix}$
lonely-a	*lonely*	A-N-A, A-R-C-A, I-C-A	lonely(X,Y)	$\boxed{X,Y}$	*l,ow1,n,l,iy0*	$\begin{bmatrix}\textsc{cat} & a\end{bmatrix}$

continued on next page

continued from previous page

Key	Ortho	Trees	Semantics	Signature	Phonetics	Feature
mild-a	*mild*	A-N-A, A-R-C-A, I-C-A	mild(X,Y)	$\boxed{X,Y}$	*m,ay1,l,d*	$\begin{bmatrix}\text{CAT} & a\end{bmatrix}$
platinum-a	*platinum*	A-N-A, A-R-C-A, I-C-A	platinum(X,Y)	$\boxed{X,Y}$	*p,l,ae1,t,ah0,n,ah0,m*	$\begin{bmatrix}\text{CAT} & a\end{bmatrix}$
pretty-a	*pretty*	A-N-A, A-R-C-A, I-C-A	pretty(X,Y)	$\boxed{X,Y}$	*p,r,ih1,t,iy0*	$\begin{bmatrix}\text{CAT} & a\end{bmatrix}$
small-a	*small*	A-R-C-A, I-C-A, A-N-A	small(X,Y)	$\boxed{X,Y}$	*s,m,ao1,l*	$\begin{bmatrix}\text{CAT} & a\end{bmatrix}$
stark-a	*stark*	A-R-C-A, I-C-A, A-N-A	stark(X,Y)	$\boxed{X,Y}$	*s,t,aa1,r,k*	$\begin{bmatrix}\text{CAT} & a\end{bmatrix}$
rare-a	*rare*	A-N-A, A-R-C-A, I-C-A	rare(X,Y)	$\boxed{X,Y}$	*r,eh1,r*	$\begin{bmatrix}\text{CAT} & a\end{bmatrix}$
round-a	*round*	A-N-A, A-R-C-A, I-C-A	round(X,Y)	$\boxed{X,Y}$	*r,aw1,n,d*	$\begin{bmatrix}\text{CAT} & a\end{bmatrix}$
sensitive-a	*sensitive*	A-N-A, A-R-C-A, I-C-A	sensitive(X,Y)	$\boxed{X,Y}$	*s,eh1,n,s,ah0,t,ih0,v*	$\begin{bmatrix}\text{CAT} & a\end{bmatrix}$
shocking-a	*shocking*	A-N-A, A-R-C-A, I-C-A	shocking(X,Y)	$\boxed{X,Y}$	*sh,aa1,k,ih0,ng*	$\begin{bmatrix}\text{CAT} & a\end{bmatrix}$
slimy-a	*slimy*	A-N-A, A-R-C-A, I-C-A	slimy(X,Y)	$\boxed{X,Y}$	*s,l,ay1,m,iy0*	$\begin{bmatrix}\text{CAT} & a\end{bmatrix}$
tender-a	*tender*	A-N-A, A-R-C-A, I-C-A	tender(X,Y)	$\boxed{X,Y}$	*t,eh1,n,d,er0*	$\begin{bmatrix}\text{CAT} & a\end{bmatrix}$
ugly-a	*ugly*	A-N-A, A-R-C-A, I-C-A	ugly(X,Y)	$\boxed{X,Y}$	*ah1,g,l,iy0*	$\begin{bmatrix}\text{CAT} & a\end{bmatrix}$

Table C.4: List of adjectives

C.1.5 Additional lexicon for “Relativity” limerick

Key	Ortho	Trees	Semantics	Signature	Phonetics	Feature
there-dnp	*there*	I-E-DNP		□	*dh,eh0,r*	$\begin{bmatrix}\textsc{cat} & dnp\end{bmatrix}$
was-cv	*was*	I-C-CV		□	*w,ah0,z*	$\begin{bmatrix}\textsc{cat} & cv\\ \textsc{agr} & \begin{bmatrix}\textsc{num} & sg\end{bmatrix}\end{bmatrix}$
young-a	*young*	A-R-C-A, I-C-A, A-N-A	young(*X*,*Y*)	$\boxed{X,Y}$	*y,ah1,ng*	$\begin{bmatrix}\textsc{cat} & a\end{bmatrix}$
lady-ns	*lady*	A-R-C-NP, I-N-N, I-C-NP, I-E-E	lady(*X*,*Y*)	$\boxed{X,Y}$	*l,ey1,d,iy0*	$\begin{bmatrix}\textsc{cat} & n\\ \textsc{agr} & \begin{bmatrix}\textsc{num} & sg\\ \textsc{pers} & 3\\ 3\textsc{rdsg} & +\\ \textsc{gndr} & fem\end{bmatrix}\\ \textsc{pron} & -\\ \textsc{prop} & -\\ \textsc{sub} & \begin{bmatrix}\textsc{anim} & +\end{bmatrix}\end{bmatrix}$
called-vm	*called*	A-E-NPPostMod	name(*Event*,*Actor*,*Object*)	$\boxed{Event,Actor,Object}$	*k,ao1,l,d*	$\begin{bmatrix}\textsc{cat} & v\\ \textsc{agr} & \begin{bmatrix}3\textsc{rdsg} & +\end{bmatrix}\end{bmatrix}$
bright-npr	*bright*	A-R-C-NP, I-N-N, I-C-NP, I-E-E	bright(*X*,*Y*)	$\boxed{X,Y}$	*b,r,ay1,t*	$\begin{bmatrix}\textsc{cat} & n\\ \textsc{agr} & \begin{bmatrix}\textsc{num} & sg\\ \textsc{pers} & 3\\ 3\textsc{rdsg} & +\end{bmatrix}\\ \textsc{pron} & -\\ \textsc{prop} & +\\ \textsc{sub} & \begin{bmatrix}\textsc{name} & +\end{bmatrix}\end{bmatrix}$
could-aux	*could*	A-V-Aux, I-V-Aux	can(*X*,*Y*)	$\boxed{X,Y}$	*k,uh0,d*	$\begin{bmatrix}\textsc{cat} & aux\end{bmatrix}$
travel-vi	*travel*	A-R-V-I, I-V-I	travel(*Event*,*Actor*)	$\boxed{Event,Actor}$	*t,r,ae1,v,ah0,l*	$\begin{bmatrix}\textsc{cat} & v\\ \textsc{agr} & \begin{bmatrix}3\textsc{rdsg} & -\end{bmatrix}\\ \textsc{subsub} & \begin{bmatrix}\textsc{anim} & +\end{bmatrix}\end{bmatrix}$
travelled-vi	*travelled*	A-R-V-I, I-V-I	travel(*Event*,*Actor*)	$\boxed{Event,Actor}$	*t,r,ae1,v,ah0,l,d*	$\begin{bmatrix}\textsc{cat} & v\\ \textsc{agr} & \begin{bmatrix}3\textsc{rdsg} & +\end{bmatrix}\\ \textsc{subsub} & \begin{bmatrix}\textsc{anim} & +\end{bmatrix}\end{bmatrix}$
much-adv	*much*	A-E-CompAdv	much(*X*,*Y*)	$\boxed{X,Y}$	*m,ah0,ch*	$\begin{bmatrix}\textsc{cat} & adv\end{bmatrix}$
faster-acomp	*faster*	A-R-C-A, I-C-A, A-N-A, A-E-Comparative	faster(*X*,*Y*,*Z*)	$\boxed{X,Y,Z}$	*f,ae1,s,t,er0*	$\begin{bmatrix}\textsc{cat} & acomp\end{bmatrix}$

continued on next page

continued from previous page

Key	Ortho	Trees	Semantics	Signature	Phonetics	Feature
speedier-acomp	*speedier*	A-R-C-A, I-C-A, A-N-A, A-E-Comparative	faster(*X,Y,Z*)	$\boxed{X,Y,Z}$	*s,p,iy1,d,iy0,er0*	$\begin{bmatrix}\textsc{cat} & acomp\end{bmatrix}$
than-compcomp	*than*	I-E-Comp		□	*dh,ah0,n*	$\begin{bmatrix}\textsc{cat} & compcomp\end{bmatrix}$
light-nms	*light*	A-R-C-NP, I-N-N, I-C-NP	light(*X,Y*)	$\boxed{X,Y}$	*l,ay1,t*	$\begin{bmatrix}\textsc{cat} & n \\ \textsc{agr} & \begin{bmatrix}\textsc{num} & ms \\ \textsc{pers} & 3 \\ \textsc{3rdsg} & + \end{bmatrix} \\ \textsc{pron} & - \\ \textsc{prop} & - \\ \textsc{sub} & \begin{bmatrix}\textsc{anim} & -\end{bmatrix}\end{bmatrix}$
setout-vi	*set out*	A-R-V-I, I-V-I	leave(*Event,Actor*)	$\boxed{Event,Actor}$	*s,eh1,t,aw0,t*	$\begin{bmatrix}\textsc{cat} & v \\ \textsc{subsub} & \begin{bmatrix}\textsc{anim} & +\end{bmatrix}\end{bmatrix}$
left-vi	*left*	A-R-V-I, I-V-I	leave(*Event,Actor*)	$\boxed{Event,Actor}$	*l,eh1,f,t*	$\begin{bmatrix}\textsc{cat} & v \\ \textsc{subsub} & \begin{bmatrix}\textsc{anim} & +\end{bmatrix}\end{bmatrix}$
oneday-timenp	*one day*	A-E-PostVP-TimeNP, A-E-PreVP-TimeNP	oneday(*Event,Actor*)	$\boxed{Event,Actor}$	*w,ah0,n,d,ey1*	$\begin{bmatrix}\textsc{cat} & timenp\end{bmatrix}$
piaw	*in*	I-E-Piaw		□	*ih0,n*	$\begin{bmatrix}\textsc{cat} & piaw\end{bmatrix}$
diaw	*a*	I-E-Diaw		□	*ah0*	$\begin{bmatrix}\textsc{cat} & diaw\end{bmatrix}$
niaw	*way*	I-E-Niaw		□	*w,ey0*	$\begin{bmatrix}\textsc{cat} & niaw\end{bmatrix}$
relatively-adv	*relatively*	A-E-PreVP-Adv, A-E-PostVP-Adv	relative(*X,Y*)	$\boxed{X,Y}$	*r,eh1,l,ah0,t,ih0,v,l,iy0*	$\begin{bmatrix}\textsc{cat} & adv\end{bmatrix}$
relative-a	*relative*	A-R-C-A, I-C-A, A-N-A, A-E-IAW	relative(*X,Y*)	$\boxed{X,Y}$	*r,eh1,l,ah0,t,ih0,v*	$\begin{bmatrix}\textsc{cat} & a\end{bmatrix}$
on-prep	*on*	I-C-PP, I-P-P, A-P-NP, A-P-VP, A-P-S, A-R-C-PP	on(*X,Y,Z*)	$\boxed{X,Y,Z}$	*ao0,n*	$\begin{bmatrix}\textsc{cat} & p\end{bmatrix}$

continued on next page

continued from previous page

Key	Ortho	Trees	Semantics	Signature	Phonetics	Feature
returned-vi	*returned*	A-R-V-I, I-V-I	return(*Event*,*Actor*)	$\boxed{Event,Actor}$	*r,ih0,t,er1,n,d*	$\begin{bmatrix}\text{CAT} & v \\ \text{SUBSUB} & \begin{bmatrix}\text{ANIM} & +\end{bmatrix}\end{bmatrix}$
night-ns	*night*	A-R-C-NP, I-N-N, I-C-NP, I-E-E	night(*X*,*Y*)	$\boxed{X,Y}$	*n,ay1,t*	$\begin{bmatrix}\text{CAT} & n \\ \text{AGR} & \begin{bmatrix}\text{NUM} & sg \\ \text{PERS} & 3 \\ \text{3RDSG} & +\end{bmatrix} \\ \text{PRON} & - \\ \text{PROP} & - \\ \text{SUB} & \begin{bmatrix}\text{ANIM} & -\end{bmatrix}\end{bmatrix}$
evening-ns	*evening*	A-R-C-NP, I-N-N, I-C-NP, I-E-E	night(*X*,*Y*)	$\boxed{X,Y}$	*iy1,v,n,ih0,ng*	$\begin{bmatrix}\text{CAT} & n \\ \text{AGR} & \begin{bmatrix}\text{NUM} & sg \\ \text{PERS} & 3 \\ \text{3RDSG} & +\end{bmatrix} \\ \text{PRON} & - \\ \text{PROP} & - \\ \text{SUB} & \begin{bmatrix}\text{ANIM} & -\end{bmatrix}\end{bmatrix}$
previous-a	*previous*	A-R-C-A, I-C-A, A-N-A	previous(*X*,*Y*)	$\boxed{X,Y}$	*p,r,iy1,v,iy0,ah0,s*	$\begin{bmatrix}\text{CAT} & a\end{bmatrix}$
preceding-a	*preceding*	A-R-C-A, I-C-A, A-N-A	previous(*X*,*Y*)	$\boxed{X,Y}$	*p,r,iy0,s,iy1,d,ih0,ng*	$\begin{bmatrix}\text{CAT} & a\end{bmatrix}$

Table C.5: List of additional words for ‘relativity’ limerick (Section 8.5.4)

C.2 Grammar

In this section we present our handcrafted grammar used during our empirical study in Chapter 8. The grammar is loosely based on the XTAG grammar (XTAG Research Group, 2001). We divide the grammar into nine groups of trees, shown from Section C.2.1 to C.2.9.

Each tree has a unique name that is referred to by the words in Section C.1. The naming convention for a tree is as follows: the first letter is either `A` if it is an auxiliary tree or `I` if it is an initial tree. The second letter indicates the group of trees it belongs to, i.e.:

- `T`: Top-level, i.e. sentence frame trees (Section C.2.1)
- `V`: Verb phrases and related trees (Section C.2.2)
- `C`: Copula constructions and related trees (Section C.2.3)
- `N`: Noun phrases and related trees (Section C.2.4)

- A: Adverbial phrases and related trees (Section C.2.5)
- P: Prepositional phrases and related trees (Section C.2.6)
- R: Relative clauses and related trees (Section C.2.7)
- M: Miscellanous trees (Section C.2.8)
- E: Extra trees developed specifically for the generation of the ‘relativity’ limerick (see Section 8.5.4).

This is followed by a short descriptive code of the tree. Thus, for example, A_A_VP_Post is an auxiliary tree belonging to the adverbial phrase group, and represents an adverbial postmodifier to a verb phrase.

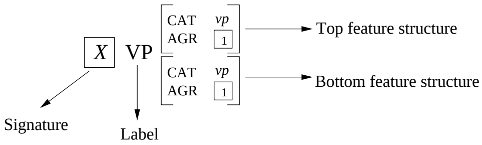


Figure C.1: Example of a node in the grammar

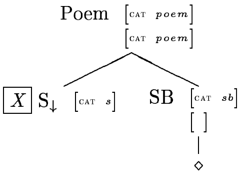
Each node shows the semantic signature, syntactic label, and top and bottom feature structure. An example is given in Figure C.1. If a label is followed by $\downarrow$, it is a substitution node, and if it is followed by *, it is a foot node. Anchor nodes are labelled by $\diamond$.

As it is a lexicalized grammar, each tree has exactly one anchor node.

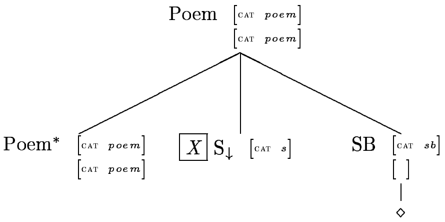
Where appropriate, we provide an example phrase for a tree, highlighting the lexical anchor in boldface type.

C.2.1 Sentence frame trees

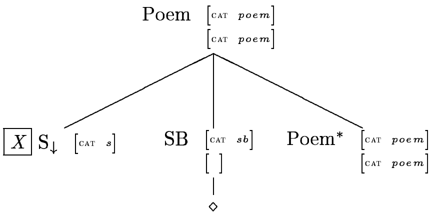
- I_T_KO: “Kick-off” tree that is used to initialize a new individual. It represents a single sentence frame.



- A_T_SR: Adds a new sentence frame to the right of the derivation.

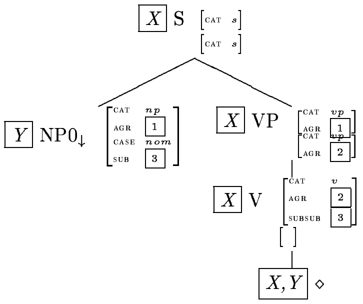


- A_T_SL Adds a new sentence frame to the left of the derivation.

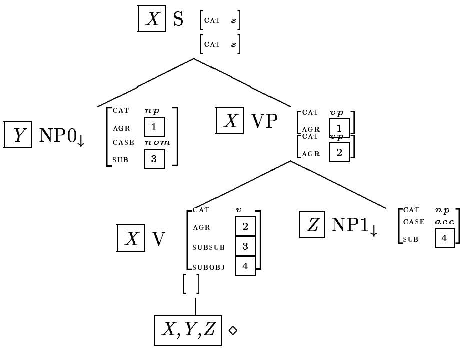


C.2.2 Verb phrases and related trees

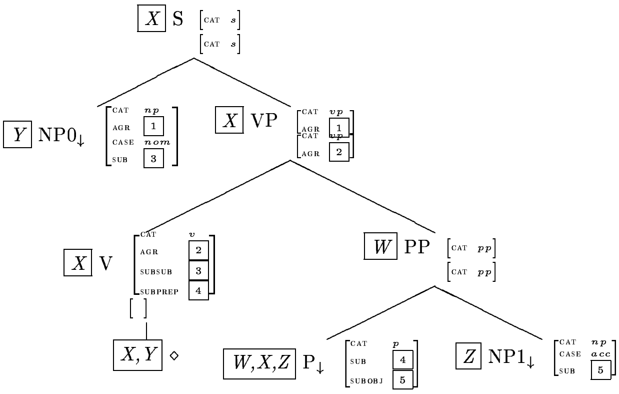
- I_V_I: Intrinsitive verb, e.g.*"The lion **survived**."*


X S [CAT s] [CAT s]
Y NP0↓ [CAT np, AGR 1, CASE nom, SUB 3]
X VP [CAT vp, AGR 1] [CAT vp, AGR 2]
X V [CAT v, AGR 2, SUBSUB 3] [ ]
X,Y ◇

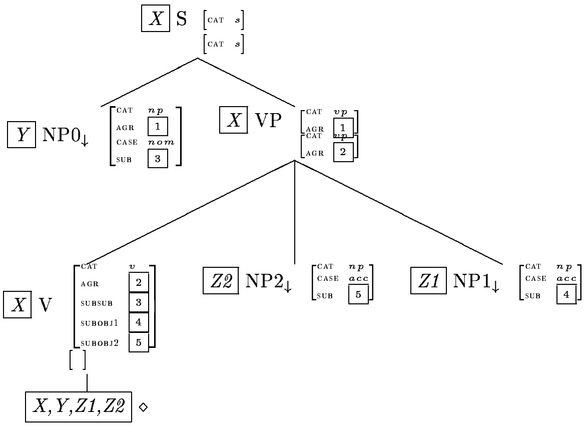
- I_V_T: Transitive verb, e.g.*"The lion **has** a big head."*


X S [CAT s] [CAT s]
Y NP0↓ [CAT np, AGR 1, CASE nom, SUB 3]
X VP [CAT vp, AGR 1] [CAT vp, AGR 2]
X V [CAT v, AGR 2, SUBSUB 3, SUBOBJ 4] [ ]
X,Y,Z ◇
Z NP1↓ [CAT np, CASE acc, SUB 4]

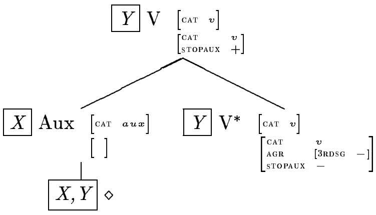
- I_V_IPP: Intransitive verb with prepositional phrase complement, e.g.*“The lion **dwells** in the waste.”*


X S [CAT s] [CAT s]
Y NP0↓ [CAT np, AGR 1, CASE nom, SUB 3]
X VP [CAT vp, AGR 1] [CAT vp, AGR 2]
X V [CAT v, AGR 2, SUBSUB 3, SUBPREP 4] [ ]
W PP [CAT pp] [CAT pp]
X,Y ◇
W,X,Z P↓ [CAT p, SUB 4, SUBOBJ 5]
Z NP1↓ [CAT np, CASE acc, SUB 5]

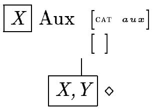
- I_V_D: Ditransitive verb, e.g.*“The boy **called** the lion a tiger.”*


X S [CAT s] [CAT s]
Y NP0↓ [CAT np, AGR 1, CASE nom, SUB 3]
X VP [CAT vp, AGR 1] [CAT vp, AGR 2]
X V [CAT v, AGR 2, SUBSUB 3, SUBOBJ1 4, SUBOBJ2 5] [ ]
Z2 NP2↓ [CAT np, CASE acc, SUB 5]
Z1 NP1↓ [CAT np, CASE acc, SUB 4]
X,Y,Z1,Z2 ◇

- A_V_Aux: Auxiliary verb (auxiliary tree), e.g.*“**will**”*.

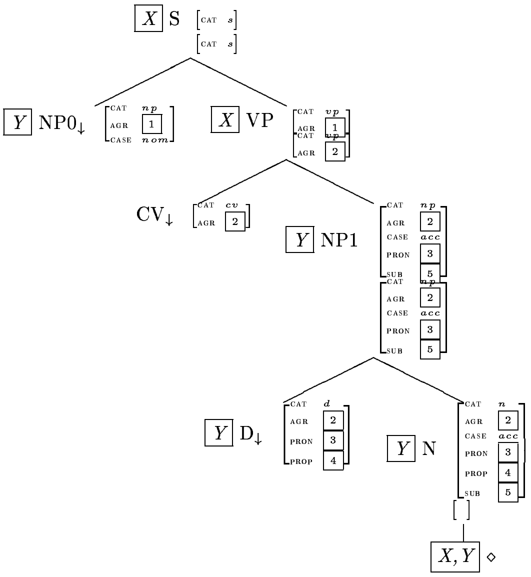

Y V [CAT v] [CAT v, STOPAUX +]
X Aux [CAT aux] [ ]
Y V* [CAT v] [CAT v, AGR [3RDSG −], STOPAUX −]
X,Y ◇

- I_V_Aux: Auxiliary verb (initial tree), e.g.*“**will**”*.

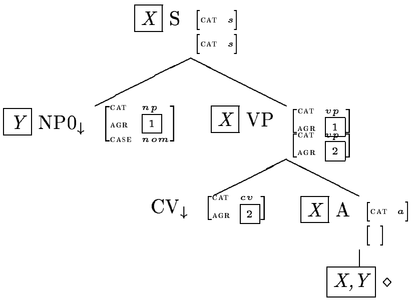


C.2.3 Copula constructions and related trees

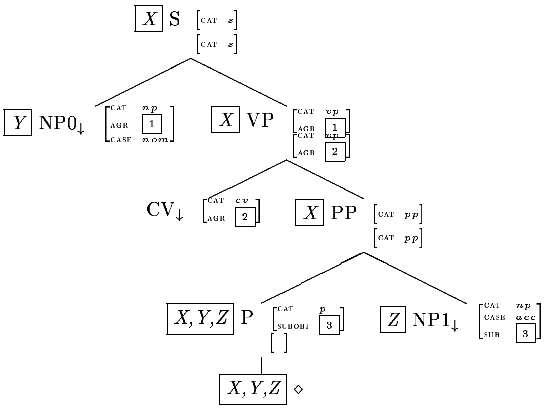
- `I_C_NP`: Nominal copula construction, e.g.*“The lion is* ***an animal****.”*



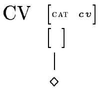
- `I_C_A`: Adjectival copula construction, e.g.*“Its jaws are* ***grim****.”*



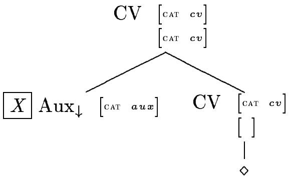
- `I_C_PP`: Prepositional copula construction, e.g.*“The fish is* ***in*** *the whale.”*



- I_C_CV: Copula verb preterminal, e.g.*“**is**”*

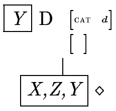


- I_C_AuxCV: Copula verb preterminal (infinitive), e.g.*“will **be**”*

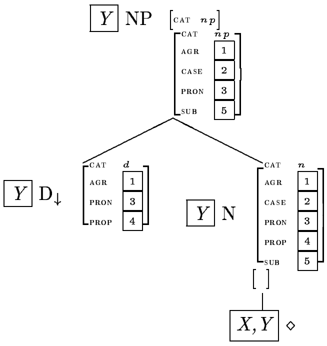


C.2.4 Noun phrases and related trees

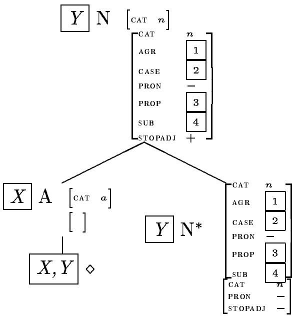
- I_N_D: Determiner preterminal, e.g.*“**the**”*



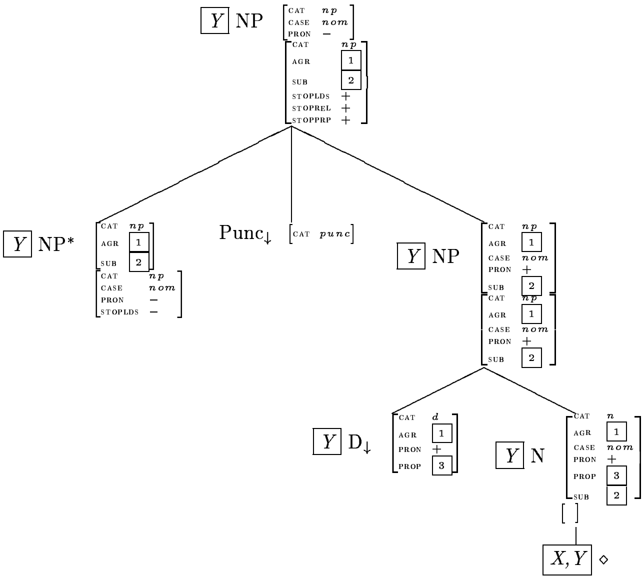
- I_N_N: Noun phrase, e.g.*“The **lion**”*



- A_N_A: Adjective, e.g. “***grim***”

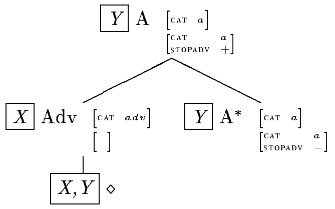


- A_N_LDS: Left dislocated noun phrase, e.g.“*The african lion, **it***” (Note: only applicable to nouns of nominative case)



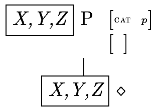
C.2.5 Adverbial phrases and related trees

- A_A_Adj: Adverbial modifier for adjective, e.g.*“**very** small”*

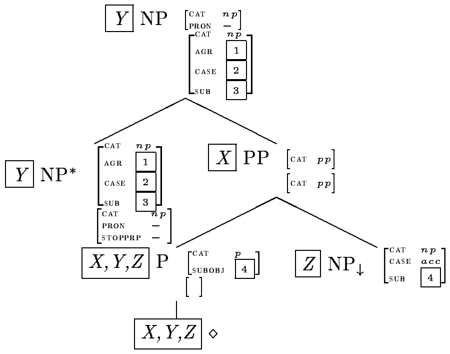


C.2.6 Prepositional phrases and related trees

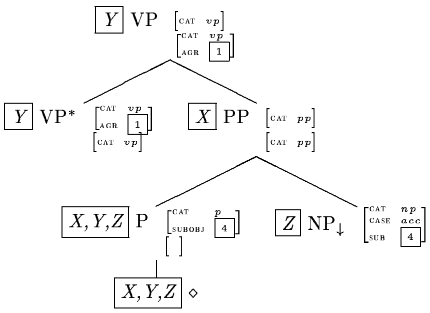
- I_P_P: Preposition preterminal, e.g.*“**with**”*



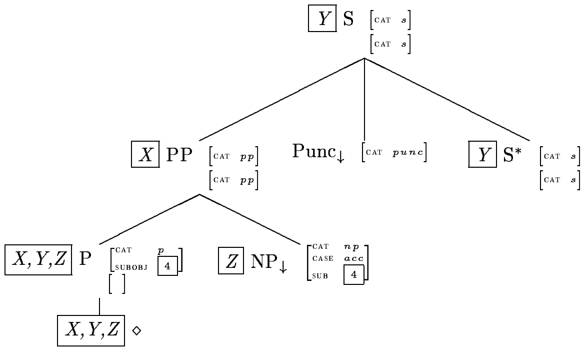
- A_P_NP: Noun phrase postmodifier prepositional phrase, e.g.*“The lion **in** the waste.”*



- A_P_VP: Verb phrase postmodifier prepositional phrase, e.g.*“boiled the frog **with** the product.”*

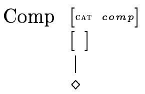


- A_P_S: Sentential premodifier prepositional phrase, e.g.*“**With** the child, the baboon flattened the elephant.”*

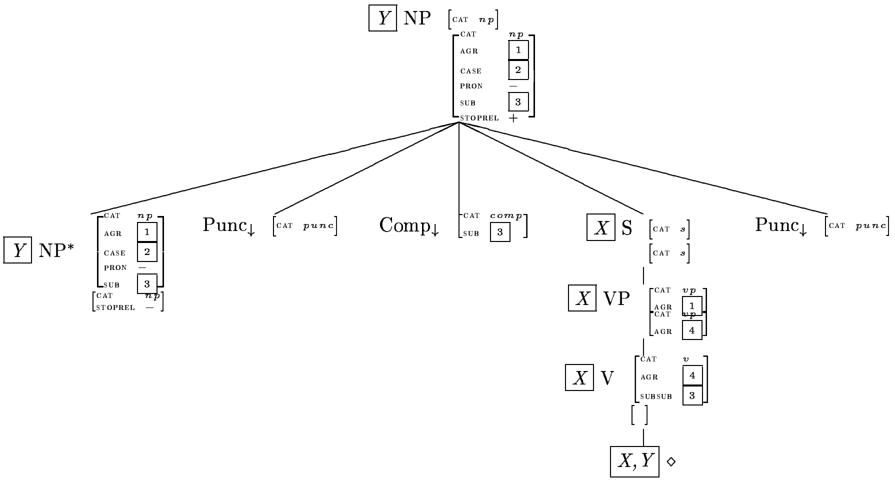


C.2.7 Relative clauses and related trees

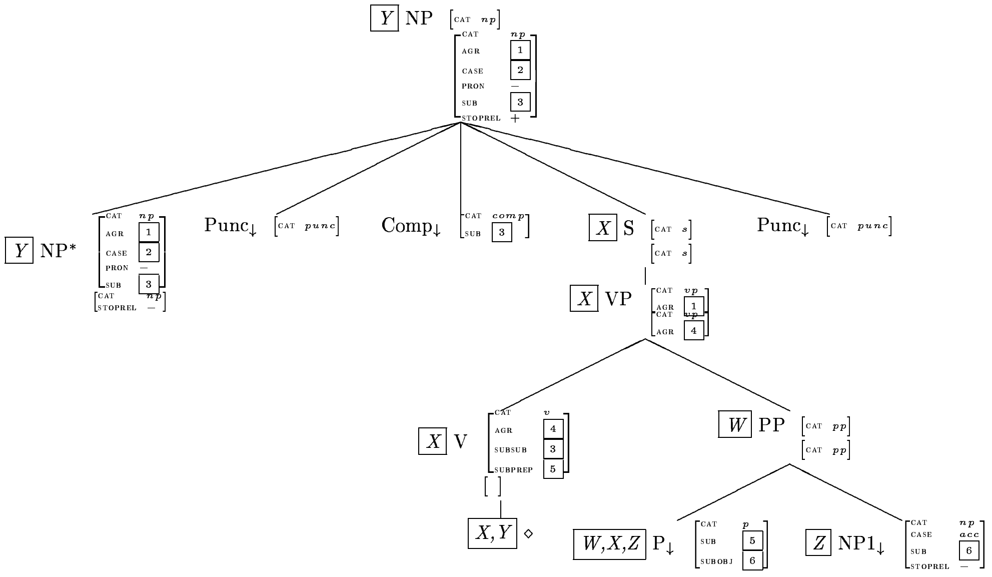
- I_R_Comp: Complementizer preterminal, e.g.*“**that**”*



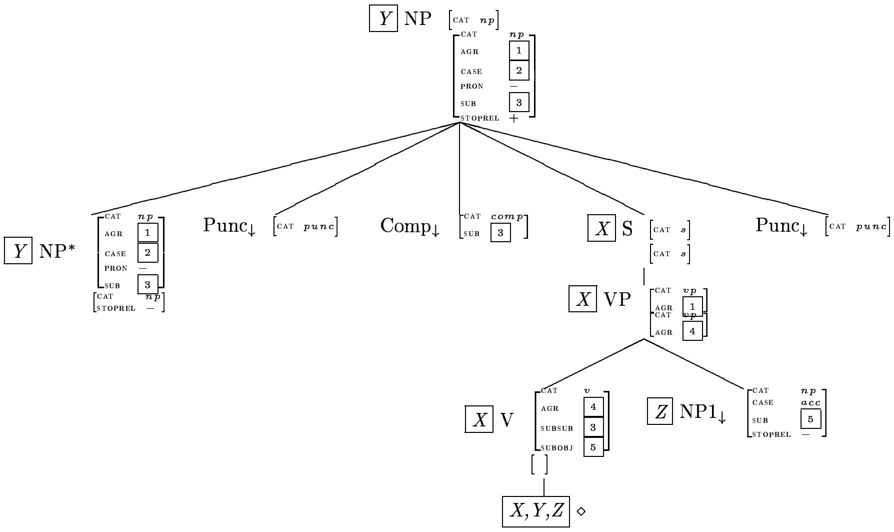
- A_R_V_I: Intransitive verb relative clause, e.g.*“The lion, who **survived**,”*


Y NP [CAT np] [CAT np, AGR 1, CASE 2, PRON −, SUB 3, STOPREL +]
Y NP* [CAT np, AGR 1, CASE 2, PRON −, SUB 3] [CAT np, STOPREL −]
Punc↓ [CAT punc]
Comp↓ [CAT comp, SUB 3]
X S [CAT s] [CAT s]
Punc↓ [CAT punc]
X VP [CAT vp, AGR 1] [CAT vp, AGR 4]
X V [CAT v, AGR 4, SUBSUB 3] [ ]
X,Y ◇

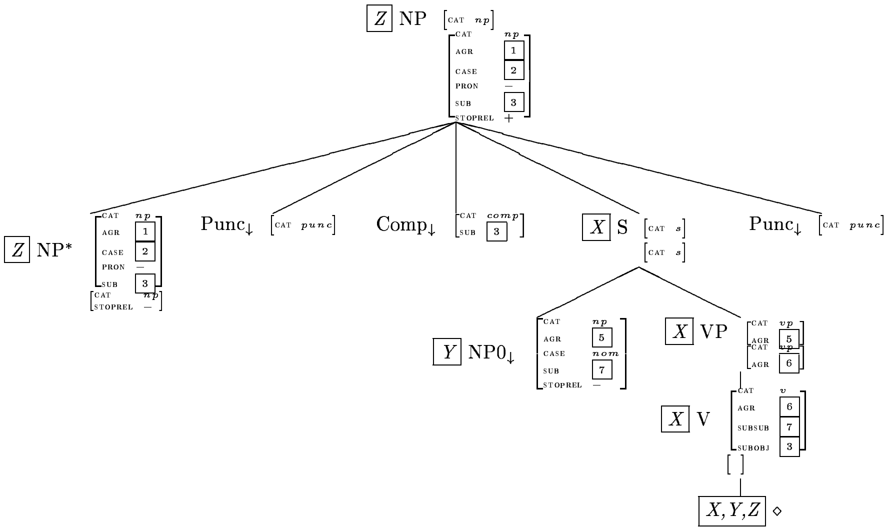
- A_R_V_IPP: Intransitive verb with prepositional phrase complement relative clause, e.g.*“The lion, who **dwells** in the waste,”*



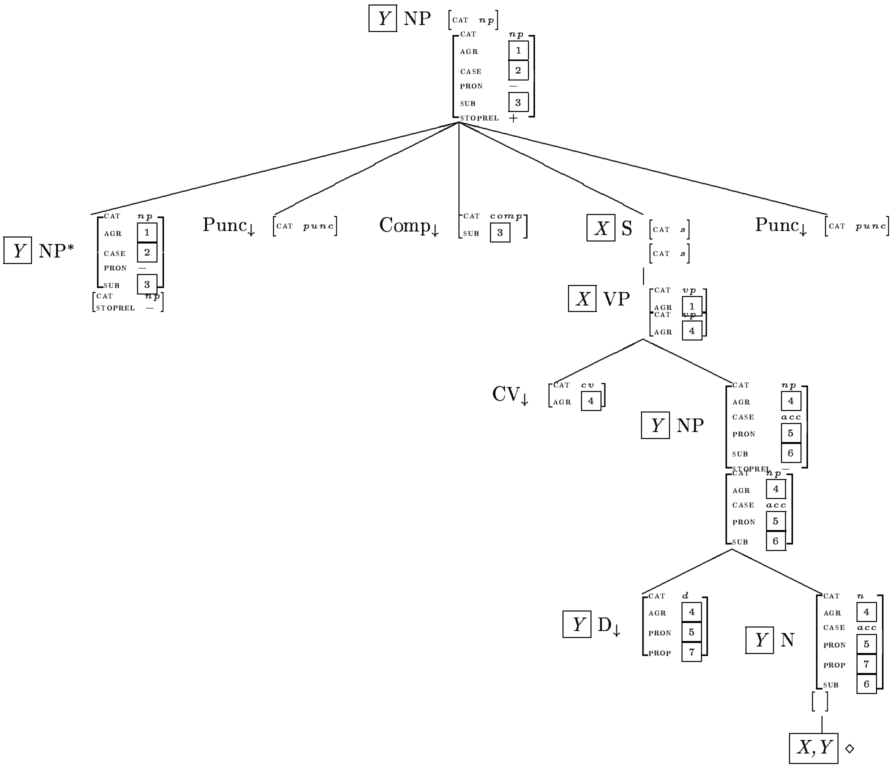
- A_R_V_T_Sub: Transitive verb relative clause (subject), e.g.*"The boy, who* ***served*** *the dish,"*



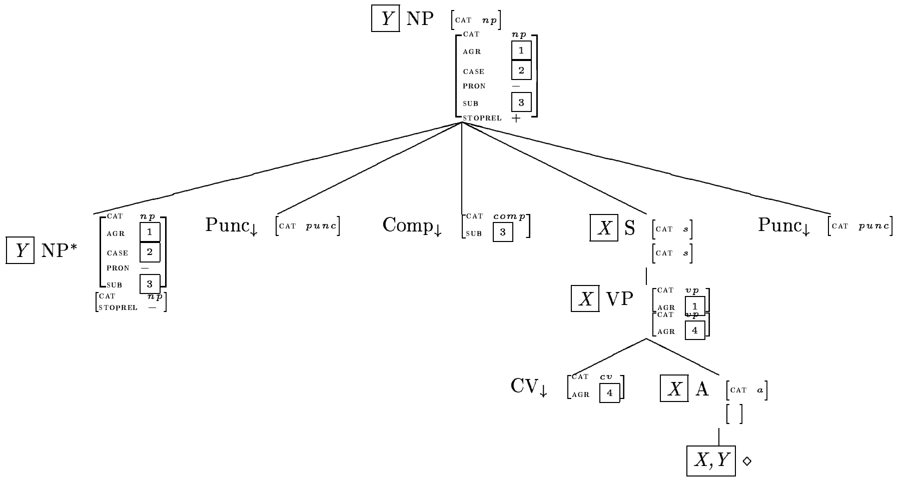
- A_R_V_T_Obj: Transitive verb relative clause (object), e.g.*"The dish, that the boy* ***served****,"*



- A_R_C_NP: Nominal copula construction relative clause, e.g.*“The boy, who is a* ***child****,”*

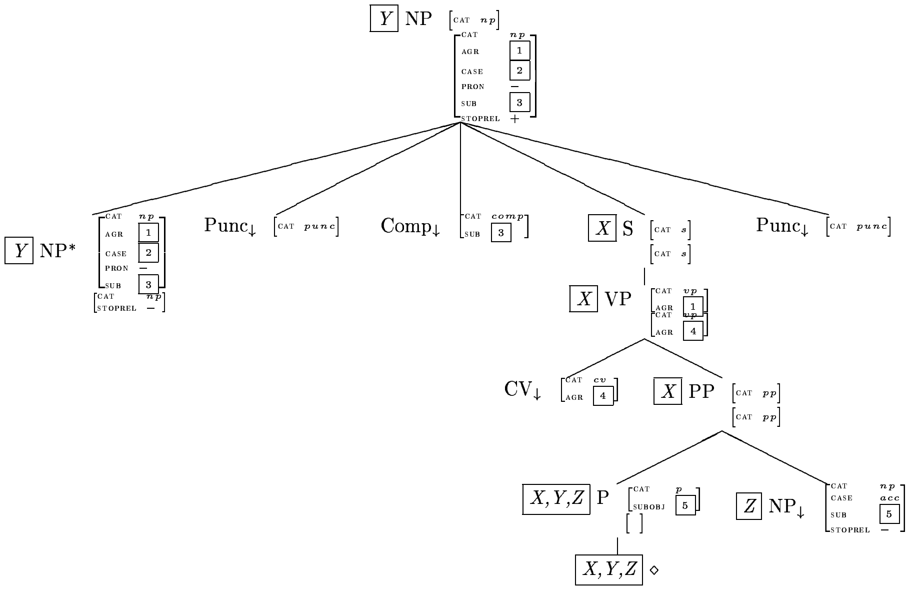


- A_R_C_A: Adjectival copula construction relative clause, e.g.*“The jaws, that are* ***grim****,”*



Y NP [cat np]
cat np, agr 1, case 2, pron −, sub 3, stoprel +
Y NP* [cat np, agr 1, case 2, pron −, sub 3] [cat np, stoprel −]
Punc↓ [cat punc]
Comp↓ [cat comp, sub 3]
X S [cat s] [cat s]
X VP [cat vp, agr 1] [cat vp, agr 4]
CV↓ [cat cv, agr 4]
X A [cat a] [ ]
X,Y ◇
Punc↓ [cat punc]

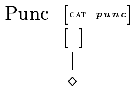
- A_R_C_PP: Prepositional copula construction relative clause, e.g.*“The fish, that is **in** the whale,”*



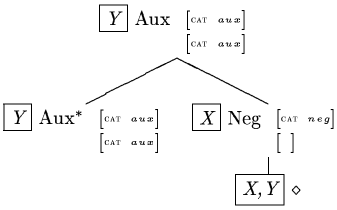
Y NP [cat np]
cat np, agr 1, case 2, pron −, sub 3, stoprel +
Y NP* [cat np, agr 1, case 2, pron −, sub 3] [cat np, stoprel −]
Punc↓ [cat punc]
Comp↓ [cat comp, sub 3]
X S [cat s] [cat s]
X VP [cat vp, agr 1] [cat vp, agr 4]
CV↓ [cat cv, agr 4]
X PP [cat pp] [cat pp]
X,Y,Z P [cat p, subobj 5] [ ]
X,Y,Z ◇
Z NP↓ [cat np, case acc, sub 5, stoprel −]
Punc↓ [cat punc]

C.2.8 Miscellanous trees

- I_M_Punct: Punctuation mark preterminal, e.g.*“**,** ”*

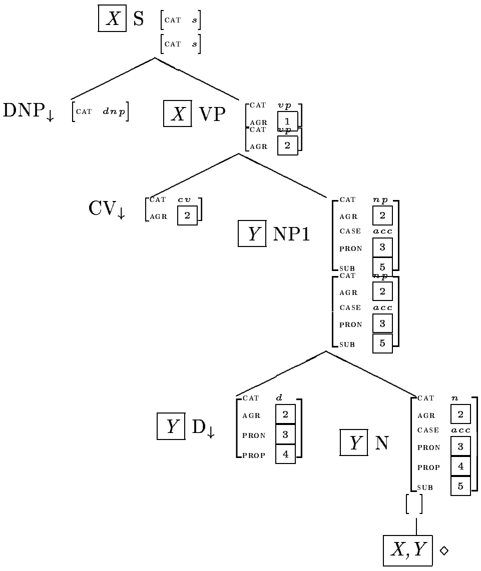


- A_M_Neg: Negation for auxiliary verb, e.g.*“will **not**”*

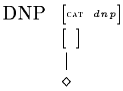


C.2.9 Additional syntax for “Relativity” limerick

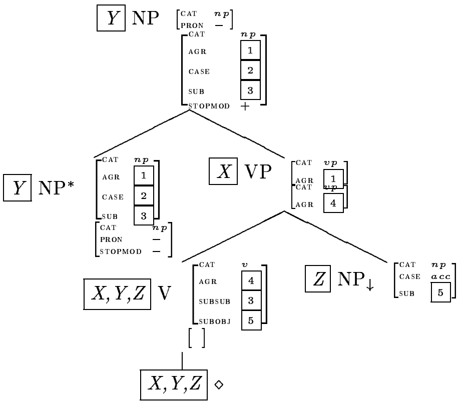
- `I_E_E`: “Existential” copula construction sentence with NP predication, e.g.*“There is a boy.”*



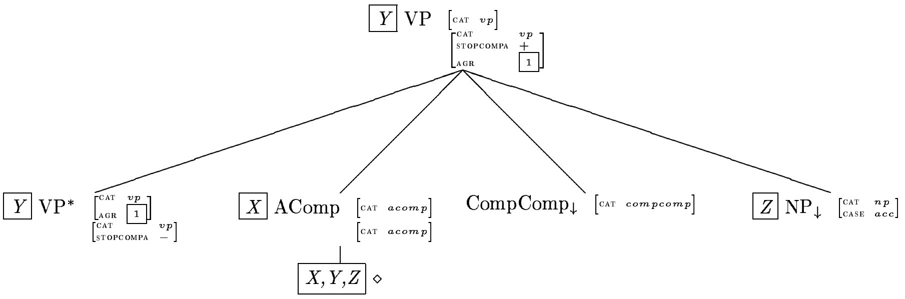
- `I_E_DNP`: The demonstrative noun phrase, e.g.*‘there’*, preterminal tree. Used specifically by `I_E_E`.



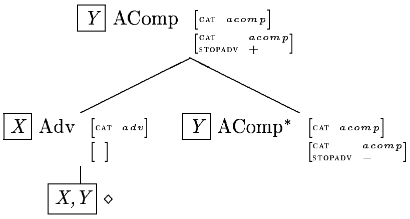
- `A_E_NPPostMod`: Auxiliary noun phrase postmodifier, e.g.*“The lady **called Bright**.”*



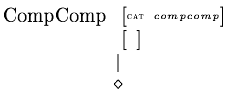
- A_E_Comparative: Verb phrase comparative postmodifier e.g.*“travels **faster than light**.”*



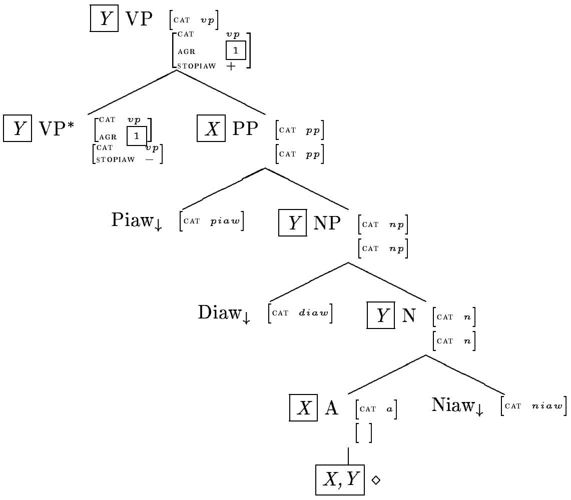
- A_E_CompAdv: Comparative adjective adverbial modifier tree, e.g.*“**much** faster”*



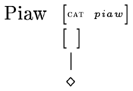
- I_E_Comp: Preterminal tree for comparative complementizers, e.g.*‘than’*



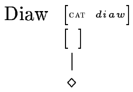
- A_E_IAW: Auxiliary verb phrase ”in a ... way” adjectival modifier e.g.: *“travelled **in a relative way**”*



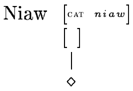
- `I_E_Piaw`: Special ”in a ... way” phrase preposition, e.g.*‘in’*, preterminal tree



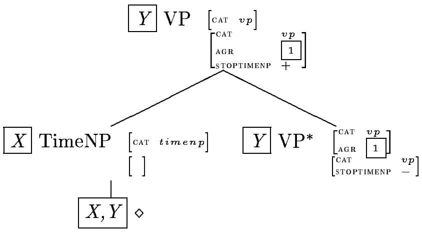
- `I_E_Diaw`: Special ”in a ... way” phrase determinar, e.g.*‘a’*, preterminal tree



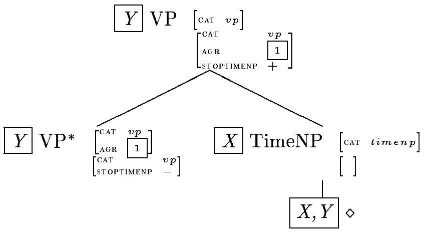
- `I_E_Niaw`: Special ”in a ... way” phrase noun, e.g.*‘way’*, preterminal tree



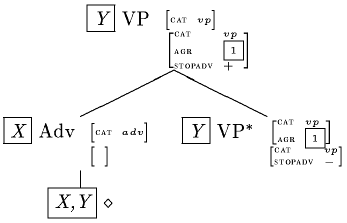
- `A_E_PreVPTimeNP`: Time noun phrase verb phrase premodifier, e.g.*“**yesterday** travelled”*



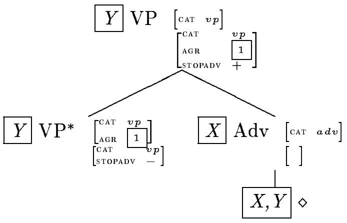
- `A_E_PostVPTimeNP`: Time noun phrase verb phrase postmodifier, e.g.*“travelled **yesterday**”*



- A_E_PreVPAdv: Adverbial verb phrase premodifier, e.g.*“**relatively** travelled”*



- A_E_PostVPAdv: Adverbial verb phrase postmodifier, e.g.*“travelled **relatively**”*



Appendix D

Empirical study statistics

In this appendix we present statistical summaries and graphs plotting the maximum and average of the best fitness scores throughout the search process obtained during the runs conducted for our empirical study.

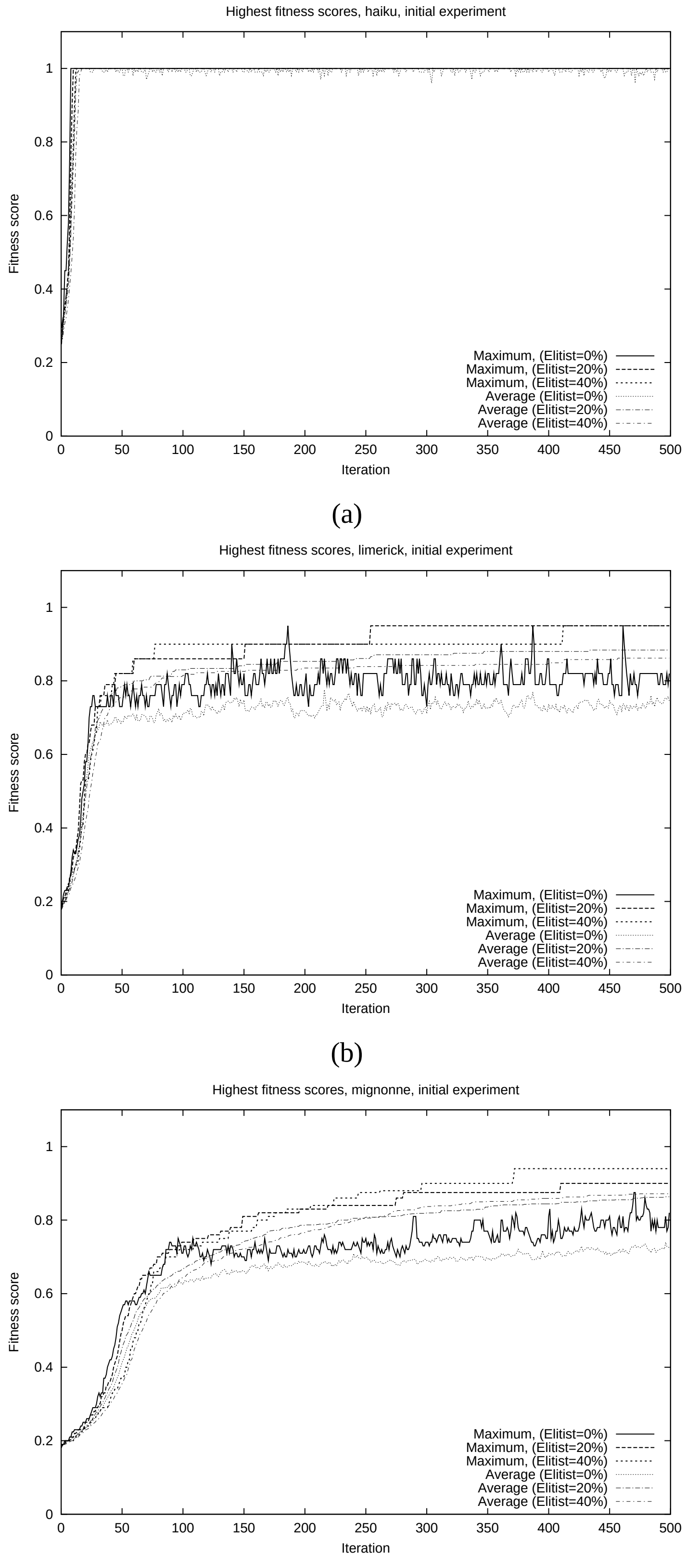
D.1 McGonagall as form-aware generator

D.1.1 Initial form-aware test

Details for this test are given in Section 8.3.1. The two factors being varied are the target form and the elitist ratio. Hence we run this test nine times: `haiku` with elitist ratios of 0%, 20%, and 40%, `limerick` with elitist ratios of 0%, 20%, and 40%, and `mignonne` with elitist ratios of 0%, 20%, and 40%. Each individual test is run ten times, and each run lasts for 500 iterations.

Table D.1 shows a summary of the results obtained from these runs. For each test, we take the best fitness scores from the last populations (i.e. the population at the end of the last iteration) of the ten test runs. We show the minimum, maximum, mean and standard deviation values for these best scores. Note that the range of values for all our evaluators is between 0.0, being the lowest score, and 1.0, that of an optimal solution.

Figure D.1 plots the maximum and average of the best fitness scores of the 10 runs for each test throughout the search process. Figure D.1(a), (b), and (c) show these scores for the `haiku`, `limerick`, and `mignonne` target forms respectively.



Highest fitness scores, haiku, initial experiment

(a)

Highest fitness scores, limerick, initial experiment

(b)

Highest fitness scores, mignonne, initial experiment

(c)

Figure D.1: Maximum and average of best fitness scores for F_{target} = (a)`haiku`, (b)`limerick`, and (c)`mignonne`, initial form-aware test

	Min	Max	Mean	Std.Dev
F_{target} =`haiku`				
Elitist 0%	1.00	1.00	1.00	0.00
Elitist 20%	1.00	1.00	1.00	0.00
Elitist 40%	1.00	1.00	1.00	0.00
F_{target} =`limerick`				
Elitist 0%	0.68	0.82	0.75	0.04
Elitist 20%	0.79	0.95	0.88	0.06
Elitist 40%	0.79	0.95	0.86	0.05
F_{target} =`mignonne`				
Elitist 0%	0.66	0.82	0.73	0.04
Elitist 20%	0.80	0.90	0.86	0.03
Elitist 40%	0.75	0.94	0.87	0.05

Table D.1: Summary statistics of best fitness scores from final populations for initial form-aware test

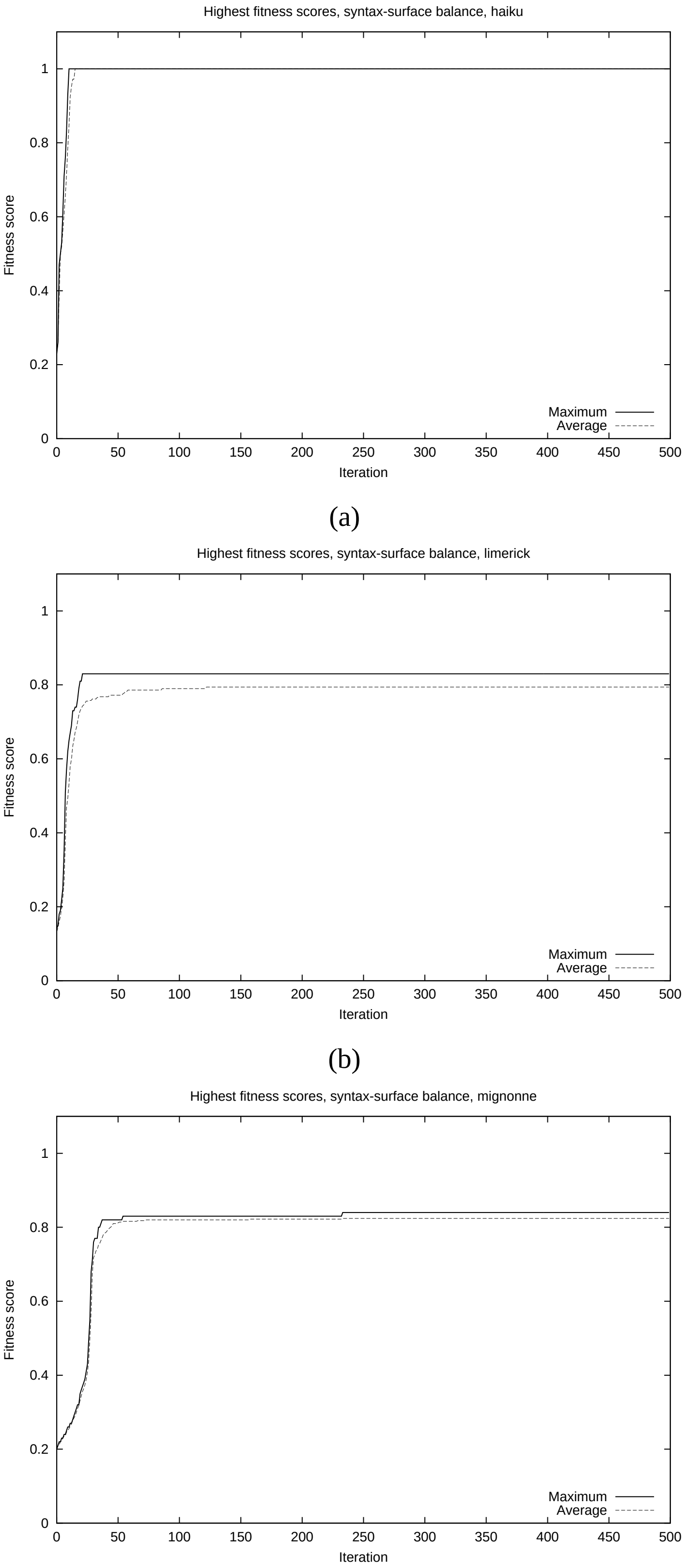
Test	Min	Max	Mean	Std.Dev
`haiku`	1.00	1.00	1.00	0.00
`limerick`	0.76	0.83	0.79	0.03
`mignonne`	0.80	0.84	0.82	0.01

Table D.2: Summary statistics of best fitness scores from final populations for syntax-surface balance test

D.1.2 Plugging the holes: syntax-surface balance test

Details for this test are given in Section 8.3.2. We ran this test three times, once for each target form. Each test was run five times. Table D.2 shows a summary of the results obtained from this test. For each test, we take the best fitness scores from the last populations (i.e. the population at the end of the last iteration) of the five test runs. We show the minimum, maximum, mean and standard deviation values for these best scores.

Figure D.2 plots the maximum and average of the best fitness scores of the five runs for each test throughout the search process. Figure D.2(a), (b), and (c) shows these scores for the `haiku`, `limerick`, and `mignonne` target forms respectively.



(a)

(b)

(c)

Figure D.2: Maximum and average of best fitness scores for $F_{target} =$ (a)`haiku`, (b)`limerick`, and (c)`mignonne`, syntax-surface balance test

Test	Min	Max	Mean	Std.Dev
haiku	1.00	1.00	1.00	0.00
limerick	0.86	1.00	0.92	0.05
mignonne	0.88	0.96	0.92	0.03

Table D.3: Summary statistics of best fitness scores from final populations for complete derivation operators test

Test	Min	Max	Mean	Std.Dev
haiku	1.00	1.00	1.00	0.00
limerick	0.86	1.00	0.93	0.05
mignonne	0.65	0.82	0.76	0.07

Table D.4: Summary statistics of best fitness scores from final populations for enjambment and linebreaking test

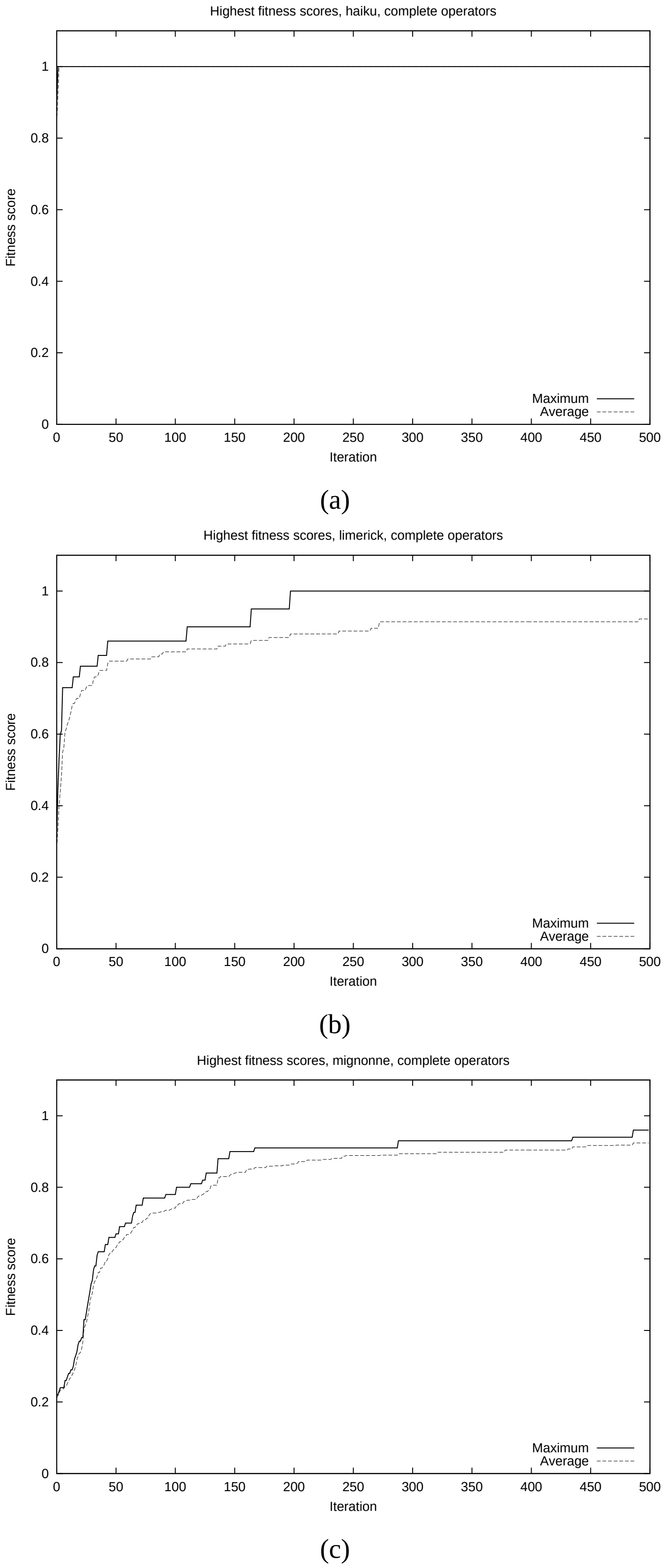
D.1.3 Plugging the holes: complete derivation operators test

Details for this test are given in Section 8.3.3. We ran this test three times, once for each target form. Each test was run five times. Table D.3 shows a summary of the results obtained from these tests. For each test, we take the best fitness scores from the last populations (i.e. the population at the end of the last iteration) of the five runs. We show the minimum, maximum, mean and standard deviation values for these best scores.

Figure D.3 plots the maximum and average of the best fitness scores of the five runs for each test throughout the search process. Figure D.3(a), (b), and (c) show these scores for the haiku, limerick, and mignonne target forms respectively.

D.1.4 Enjambment and linebreaking test

Details for this test are given in Section 8.3.4. We ran this test three times, once for each target form. Each test was run five times. Table D.4 shows a summary of the results obtained from this test. For each test, we take the best fitness scores from the last populations (i.e. the population at the end of the last iteration) of the five test runs. We show the minimum, maximum, mean and standard deviation values for these best scores.



(a)

(b)

(c)

Figure D.3: Maximum and average of best fitness scores for $F_{target} =$ (a)`haiku`, (b)`limerick`, and (c)`mignonne`, complete derivation operators test

Test	Min	Max	Mean	Std.Dev
`mignonne`	0.81	0.85	0.83	0.02

Table D.5: Summary statistics of best fitness scores from final populations for effect of scale test

Test	Min	Max	Mean	Std.Dev
`mignonne`	0.55	0.62	0.59	0.02

Table D.6: Summary statistics of best fitness scores from final populations for reduced grammar test

Figure D.4 plots the maximum and average of the best fitness scores of the five runs for each test throughout the search process. Figure D.4(a), (b), and (c) show these scores for the `haiku`, `limerick`, and `mignonne` target forms respectively.

D.1.5 Effect of scale test

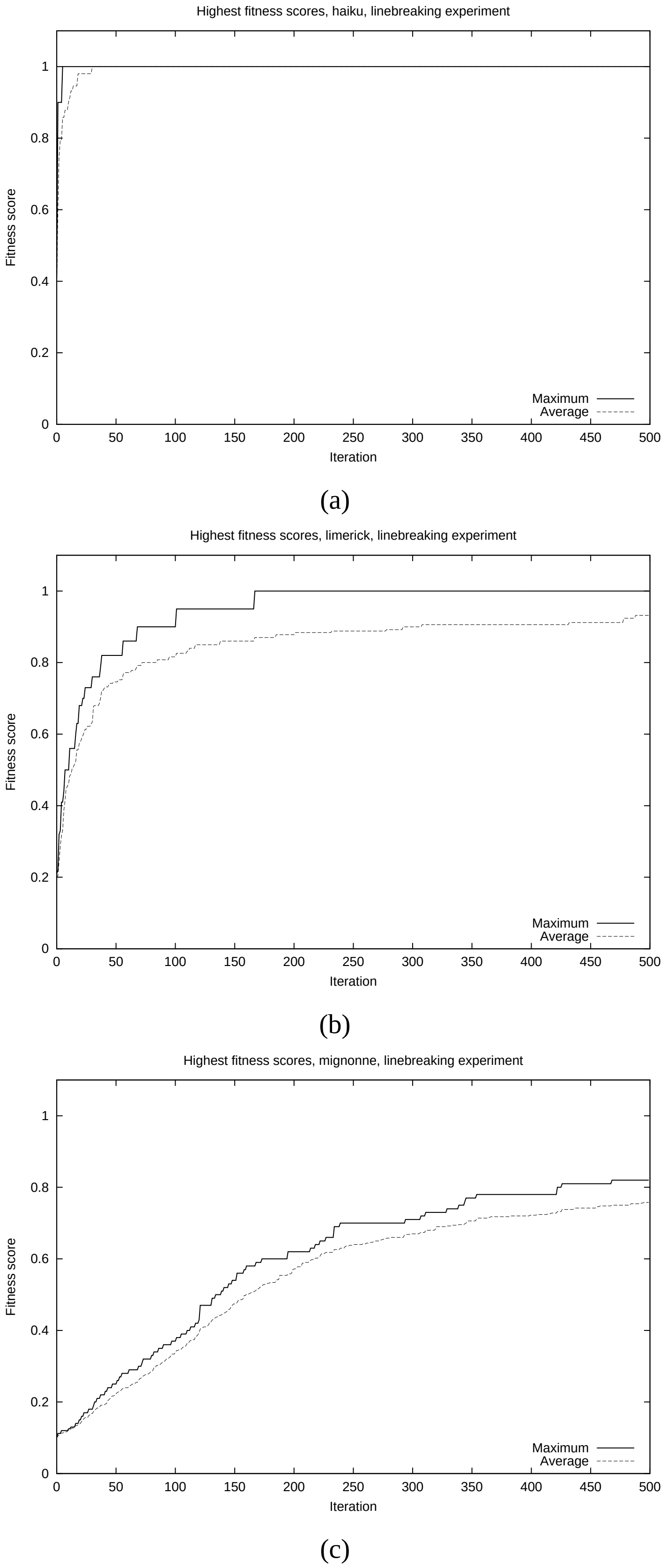
Details for this test are given in Section 8.3.5. Due to the heavy computational cost of this task, we ran this test three times as opposed to five. Table D.5 shows a summary of the results obtained from these runs. We take the best fitness scores from the last populations (i.e. the population at the end of the last iteration) of the three test runs, and show the minimum, maximum, mean and standard deviation values for these scores.

Figure D.5 plots the maximum and average of the best fitness scores of the three runs throughout the search process.

D.1.6 Reduced grammar test

Details for this test are given in Section 8.3.6. We ran this test five times. Table D.6 shows a summary of the results obtained from these runs. We take the best fitness scores from the last populations (i.e. the population at the end of the last iteration) of the five runs, and show the minimum, maximum, mean and standard deviation values for these scores.

Figure D.6 plots the maximum and average of the best fitness scores of the five runs throughout the search process.



(a)

(b)

(c)

Figure D.4: Maximum and average of best fitness scores for $F_{target} =$ (a)`haiku`, (b)`limerick`, and (c)`mignonne`, enjambment and linebreaking test

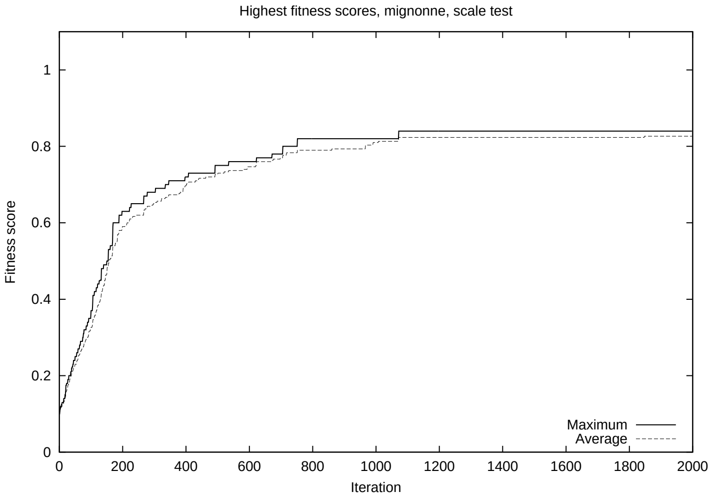


Figure D.5: Maximum and average of best fitness scores for effect of scale test

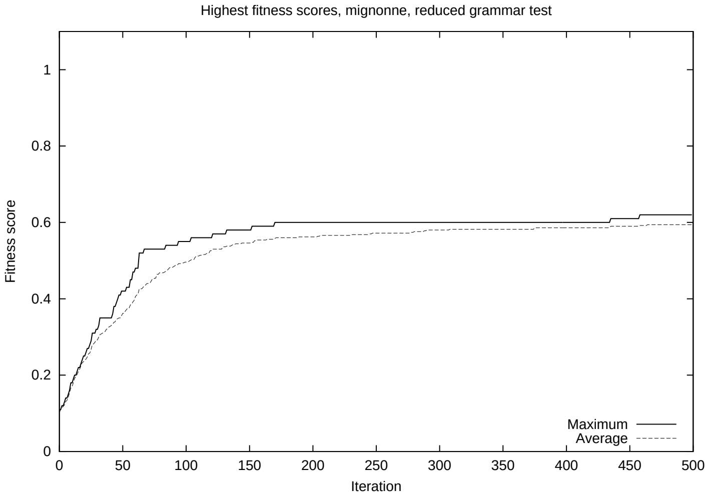


Figure D.6: Maximum and average of best fitness scores for reduced grammar test

Test	Min	Max	Mean	Std.Dev
$p_{mutation} = 0.75, p_{crossover} = 0.25$	0.46	0.90	0.71	0.14
$p_{mutation} = 0.50, p_{crossover} = 0.50$	0.60	0.96	0.80	0.13
$p_{mutation} = 0.25, p_{crossover} = 0.75$	0.52	0.75	0.67	0.09

Table D.7: Summary statistics of best fitness scores from final populations for crossover operator test

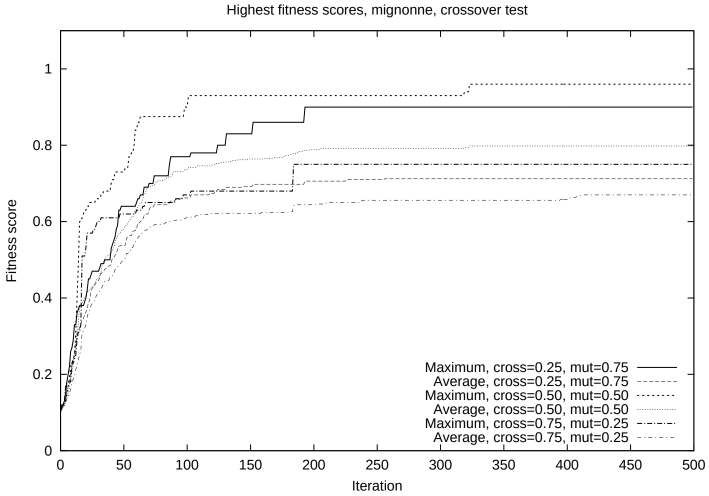


Figure D.7: Maximum and average of best fitness scores for crossover operator test

D.1.7 Crossover operator test

Details for this test are given in Section 8.3.7. We ran this test three times, assigning $p_{mutation}$ the values of 0.25, 0.5, and 0.75 (and thus assigning $p_{crossover}$ 0.75, 0.5, and 0.25). Each test was run five times. Table D.7 shows a summary of the results obtained from these runs. We take the best fitness scores from the last populations (i.e. the population at the end of the last iteration) of the runs, and show the minimum, maximum, mean and standard deviation values for these scores.

Figure D.7 plots the maximum and average of the best fitness scores of the three tests throughout the search process.

Test	Min	Max	Mean	Std.Dev
Weighting 1				
Elitist 20%	0.49	0.81	0.72	0.10
Elitist 40%	0.71	0.99	0.79	0.08
Weighting 2				
Elitist 20%	0.74	0.93	0.85	0.06
Elitist 40%	0.79	0.93	0.87	0.04
Weighting 3				
Elitist 20%	0.73	0.90	0.80	0.05
Elitist 40%	0.74	0.93	0.83	0.07

Table D.8: Summary statistics of best fitness scores from final populations for S_{target} =`lionhalf`, initial tactical NLG test

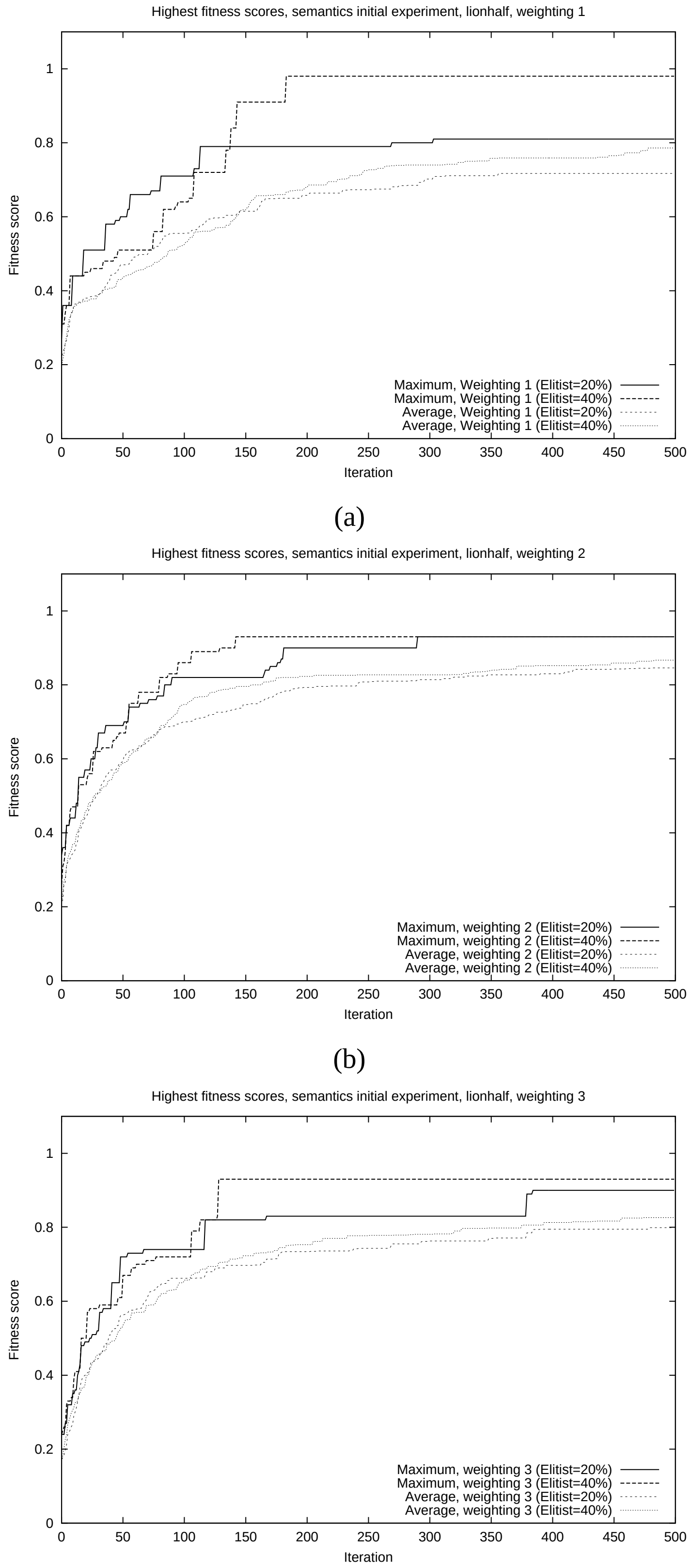
D.2 McGonagall as tactical NLG component

D.2.1 Initial tactical NLG test

Details for this test are given in Section 8.4.1. As there are three factors being varied in this test, i.e. target semantics, weighting scheme, and elitist ratio, we conduct twelve separate tests: S_{target} =`lionhalf` with weighting schemes 1, 2, and 3, and S_{target} =`lion` with weighting schemes 1, 2, and 3, all of which are run with elitist ratios of 20% and 40%. Each individual test is run ten times, and each run lasts for 500 iterations.

Tables D.8 and D.9 show a summary of the results obtained from these runs for S_{target} =`lionhalf` and `lion` respectively. For each test, we take the best fitness scores from the last populations (i.e. the population at the end of the last iteration) of the ten test runs. We show the minimum, maximum, mean and standard deviation values for these best scores.

Figures D.8 and D.9 plot the maximum and average of the best fitness scores of the runs for each test throughout the search process for `lionhalf` and `lion` respectively. Parts (a), (b), and (c) of these figures show these scores for the three different weighting schemes.



Highest fitness scores, semantics initial experiment, lionhalf, weighting 1

(a)

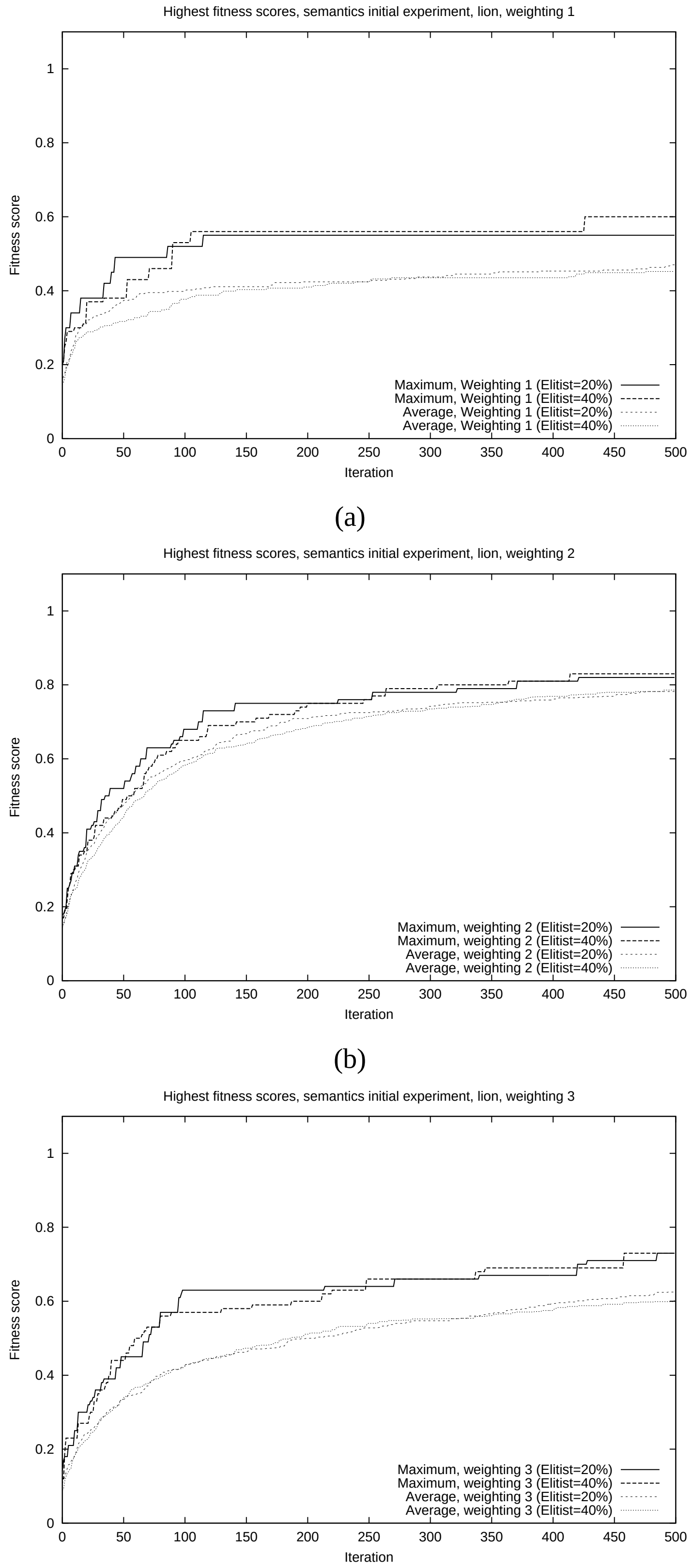
Highest fitness scores, semantics initial experiment, lionhalf, weighting 2

(b)

Highest fitness scores, semantics initial experiment, lionhalf, weighting 3

(c)

Figure D.8: Maximum and average of best fitness scores for (a) weighting 1, (b) weighting 2, and (c) weighting 3, S_{target} = `lionhalf`, initial tactical NLG test



(a)

(b)

(c)

Figure D.9: Maximum and average of best fitness scores for (a) weighting 1, (b) weighting 2, and (c) weighting 3, $S_{target} = \texttt{lion}$, initial tactical NLG test

Test	Min	Max	Mean	Std.Dev
Weighting 1				
Elitist 20%	0.39	0.55	0.47	0.05
Elitist 40%	0.34	0.60	0.45	0.08
Weighting 2				
Elitist 20%	0.74	0.82	0.78	0.03
Elitist 40%	0.71	0.83	0.79	0.03
Weighting 3				
Elitist 20%	0.45	0.73	0.63	0.09
Elitist 40%	0.28	0.73	0.60	0.12

Table D.9: Summary statistics of best fitness scores from final populations for S_{target} = `lion`, initial tactical NLG test

D.2.2 Smart operators test

Details for this test are given in Section 8.4.2. As in the previous section, we conduct twelve separate tests: S_{target} =`lionhalf` with weighting schemes 1, 2, and 3, and S_{target} =`lion` with weighting schemes 1, 2, and 3, all of which are run with elitist ratios of 20% and 40%.

Tables D.10 and D.11 show a summary of the results obtained from these runs for S_{target} =`lionhalf` and `lion` respectively. For each test, we take the best fitness scores from the last populations (i.e. the population at the end of the last iteration) of the ten test runs. We show the minimum, maximum, mean and standard deviation values for these best scores.

Figures D.10 and D.11 plot the maximum and average of the best fitness scores of the runs for each test throughout the search process for `lionhalf` and `lion` respectively. Parts (a), (b), and (c) of these figures show these scores for the three different weighting schemes.

D.2.3 PROTECTOR and SPUD-like operator test

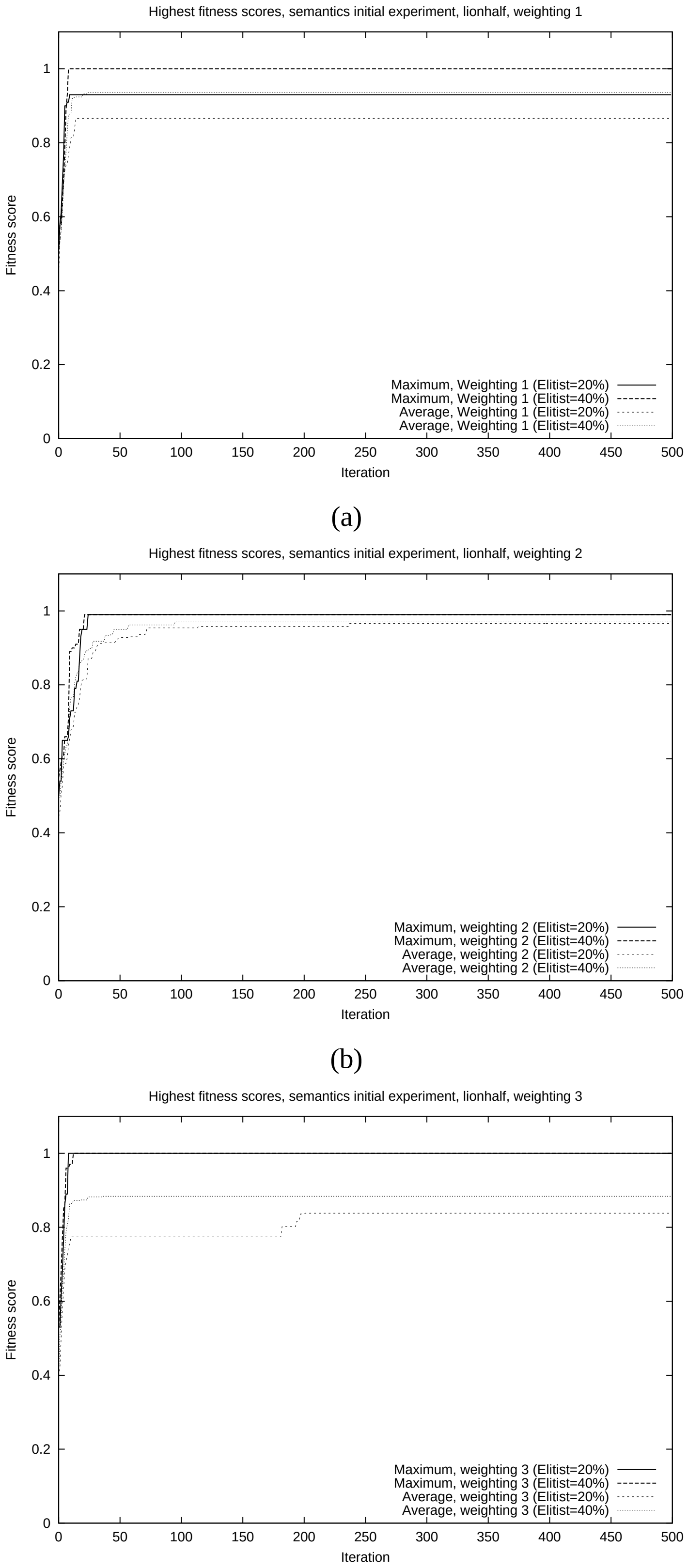
Details for this test are given in Section 8.4.3. The two tests for the PROTECTOR-like and SPUD-like operators were run ten times. Table D.12 shows a summary of the results obtained from these runs. For each test, we take the best fitness scores from the last populations (i.e. the population at the end of the last iteration) of the ten test runs. We show the minimum,

Test	Min	Max	Mean	Std.Dev
Weighting 1				
Elitist 20%	0.65	0.93	0.87	0.11
Elitist 40%	0.78	1.00	0.94	0.09
Weighting 2				
Elitist 20%	0.93	0.99	0.97	0.03
Elitist 40%	0.89	0.99	0.97	0.04
Weighting 3				
Elitist 20%	0.64	1.00	0.84	0.12
Elitist 40%	0.64	1.00	0.88	0.14

Table D.10: Summary statistics of best fitness scores from final populations for S_{target} =`lionhalf`, smart operators test

Test	Min	Max	Mean	Std.Dev
Weighting 1				
Elitist 20%	0.37	0.54	0.46	0.07
Elitist 40%	0.37	0.54	0.44	0.07
Weighting 2				
Elitist 20%	0.59	0.88	0.70	0.12
Elitist 40%	0.57	0.61	0.58	0.02
Weighting 3				
Elitist 20%	0.44	0.79	0.55	0.12
Elitist 40%	0.33	0.66	0.48	0.11

Table D.11: Summary statistics of best fitness scores from final populations for S_{target} =`lion`, smart operators test

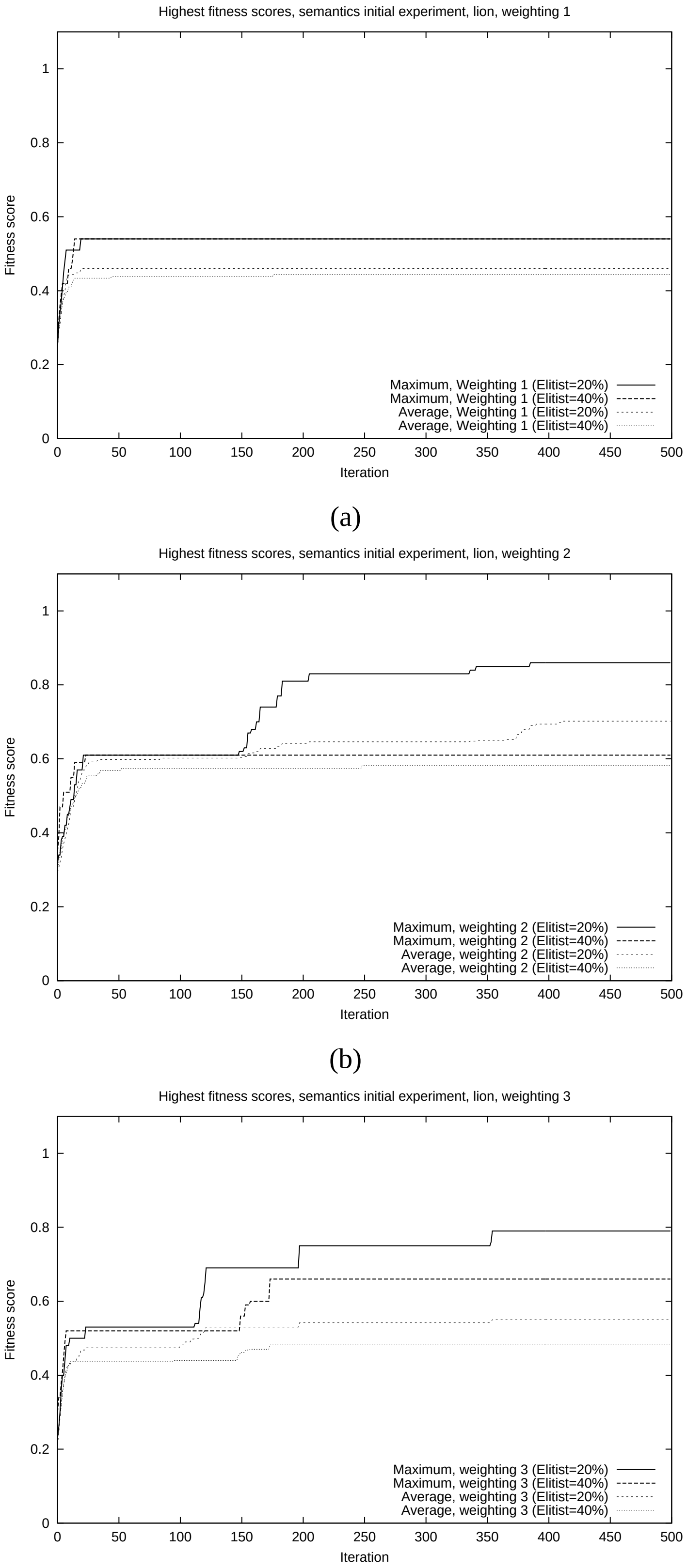


(a)

(b)

(c)

Figure D.10: Maximum and average of best fitness scores for (a) weighting 1, (b) weighting 2, and (c) weighting 3, $S_{target} = \texttt{lionhalf}$, smart operators test



(a)

(b)

(c)

Figure D.11: Maximum and average of best fitness scores for (a) weighting 1, (b) weighting 2, and (c) weighting 3, $S_{target} = \texttt{lion}$, smart operators test

Test	Min	Max	Mean	Std.Dev
PROTECTOR-like	0.60	0.85	0.78	0.09
SPUD-like	0.42	0.93	0.75	0.18

Table D.12: Summary statistics of best fitness scores from final populations for PROTECTOR and SPUD-like operator test

Test	Min	Max	Mean	Std.Dev
`haiku`	0.67	0.77	0.72	0.04
`limerick`	0.60	0.81	0.69	0.06
`mignonne`	0.61	0.78	0.69	0.04

Table D.13: Summary statistics of best fitness scores from final populations for initial poetry generation test

maximum, mean and standard deviation values for these best scores.

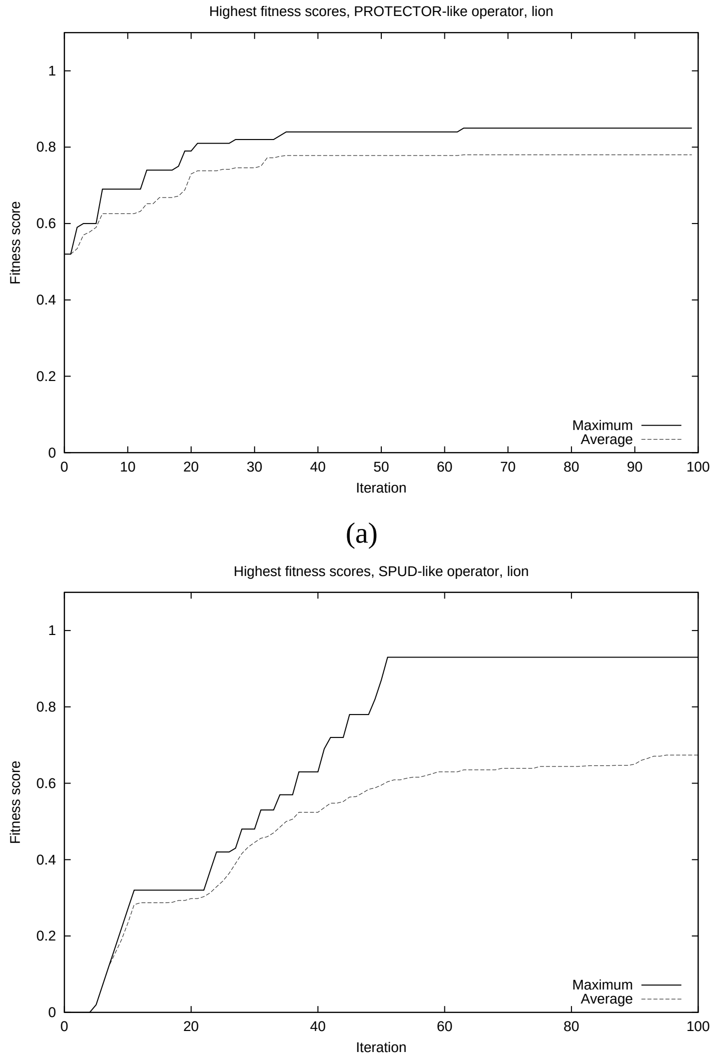
Figure D.12(a) and (b) plots the maximum and average of the best fitness scores of the runs for the tests using the PROTECTOR-like and SPUD-like operators respectively.

D.3 McGonagall as poetry generation system

D.3.1 Initial poetry generation test

Details for this test are given in Section 8.5.1. We ran this test three times, once for each target form. Each test was run five times. Table D.13 shows a summary of the results obtained from this test. For each target form, we take the best fitness scores from the last populations (i.e. the population at the end of the last iteration) of the runs. We show the minimum, maximum, mean and standard deviation values for these best scores.

Figure D.13 plots the maximum and average of the best fitness scores of the five runs for each test throughout the search process. Figure D.13(a), (b), and (c) show these scores for the `haiku`, `limerick`, and `mignonne` target forms respectively.



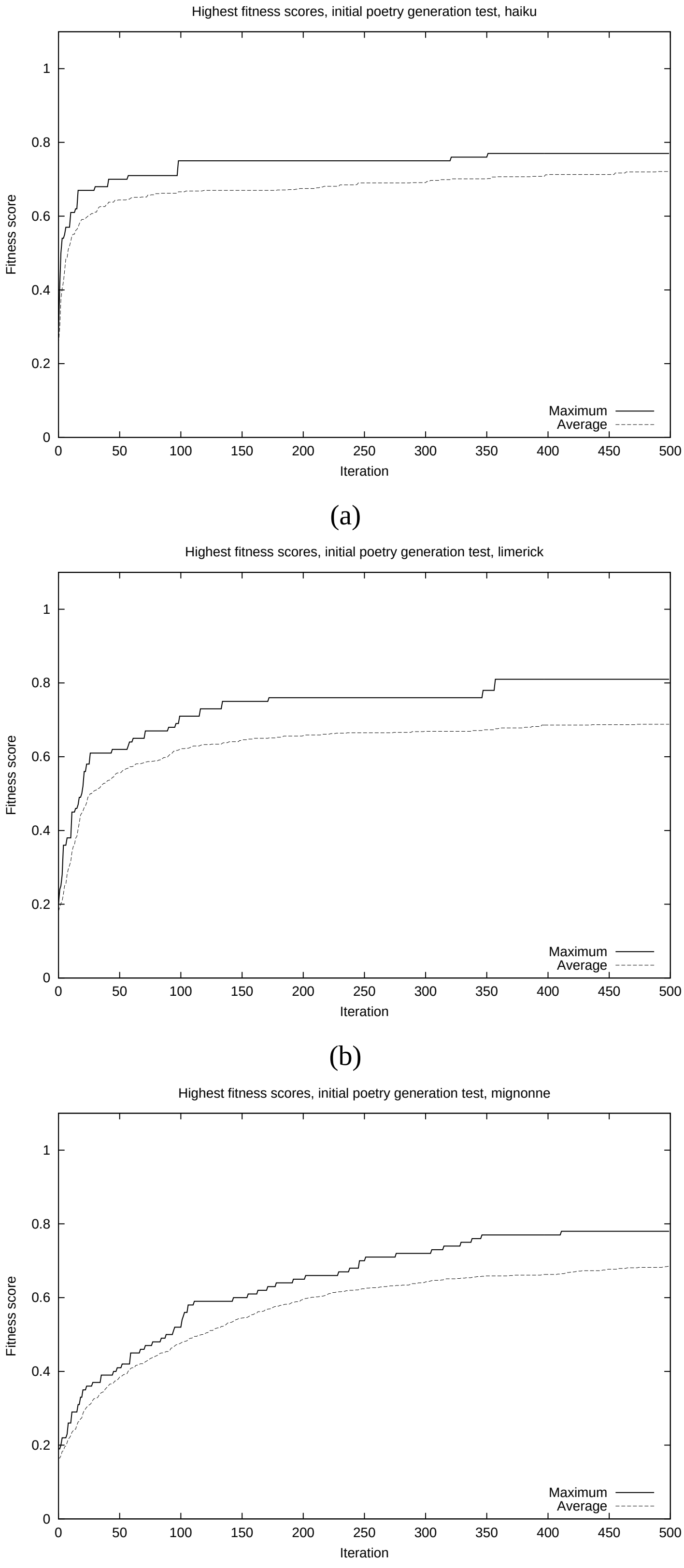
Highest fitness scores, PROTECTOR-like operator, lion

(a)

Highest fitness scores, SPUD-like operator, lion

(b)

Figure D.12: Maximum and average of best fitness scores for (a)PROTECTOR-like and (b)SPUD-like operator test



(a)

(b)

(c)

Figure D.13: Maximum and average of best fitness scores for F_{target} =(a)`haiku`, (b)`limerick` and (c)`mignonne`, initial poetry generation test

Test	Min	Max	Mean	Std.Dev
haiku	0.72	0.86	0.83	0.04
limerick	0.75	0.83	0.79	0.02
mignonne	0.55	0.80	0.66	0.10

Table D.14: Summary statistics of best fitness scores from final populations smart operators poetry generation test

Test	Min	Max	Mean	Std.Dev
haiku	0.66	0.90	0.81	0.09
limerick	0.66	0.82	0.72	0.06
mignonne	0.45	0.81	0.59	0.13

Table D.15: Summary statistics of best fitness scores from final populations for distribution of heuristics test

D.3.2 Smart operators poetry generation tests

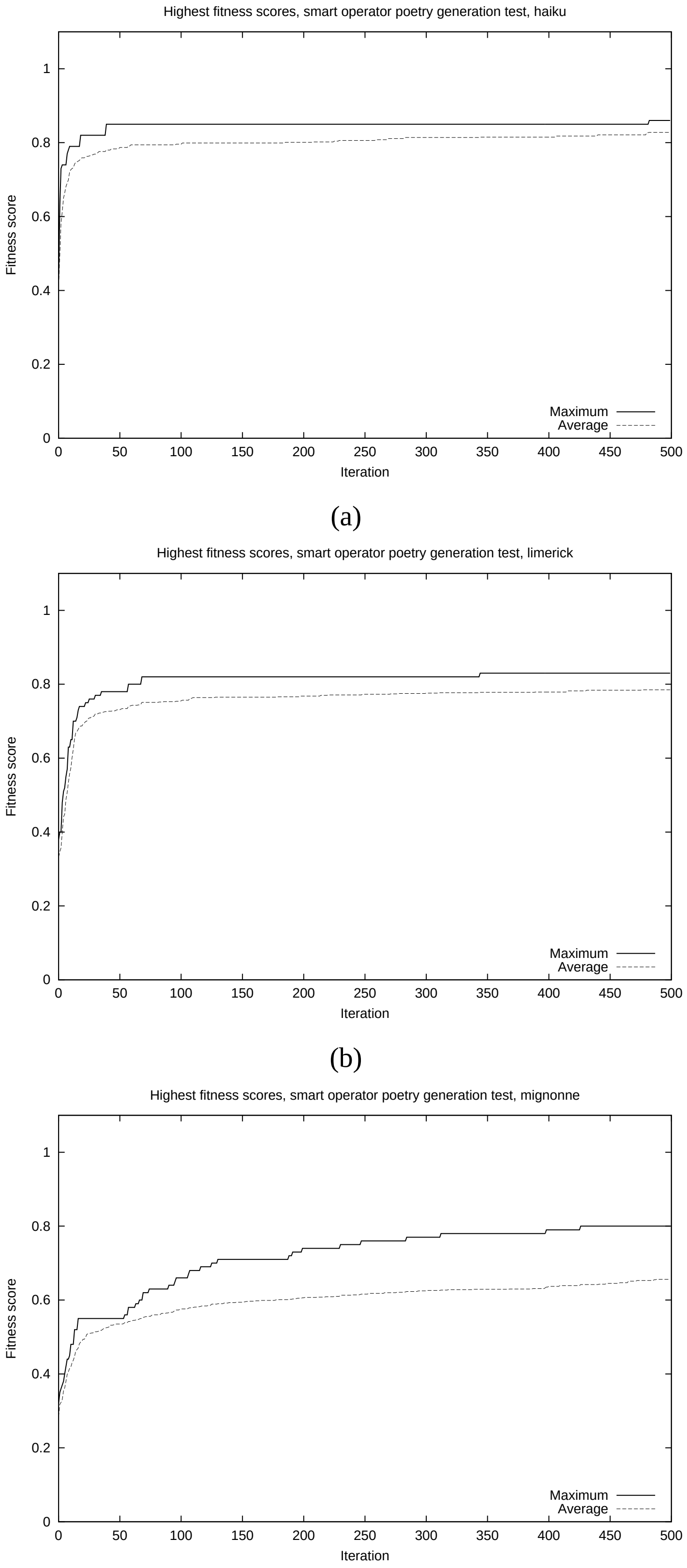
Details for this test are given in Section 8.5.2. We ran this test three times, once for each target form. Each test was run five times. Table D.14 shows a summary of the results obtained from this test. For each target form, we take the best fitness scores from the last populations (i.e. the population at the end of the last iteration) of the runs. We show the minimum, maximum, mean and standard deviation values for these best scores.

Figure D.14 plots the maximum and average of the best fitness scores of the five runs for each test throughout the search process. Figure D.14(a), (b), and (c) show these scores for the `haiku`, `limerick`, and `mignonne` target forms respectively.

D.3.3 Distribution of heuristics test

Details for this test are given in Section 8.5.3. We ran this test three times, once for each target form. Each test was run five times. Table D.15 shows a summary of the results obtained from this test. For each target form, we take the best fitness scores from the last populations (i.e. the population at the end of the last iteration) of the runs. We show the minimum, maximum, mean and standard deviation values for these best scores.

Figure D.15 plots the maximum and average of the best fitness scores of the five runs for each



(a)

(b)

(c)

Figure D.14: Maximum and average of best fitness scores for F_{target} =(a)`haiku`, (b)`limerick` and (c)`mignonne`, smart operators poetry generation test

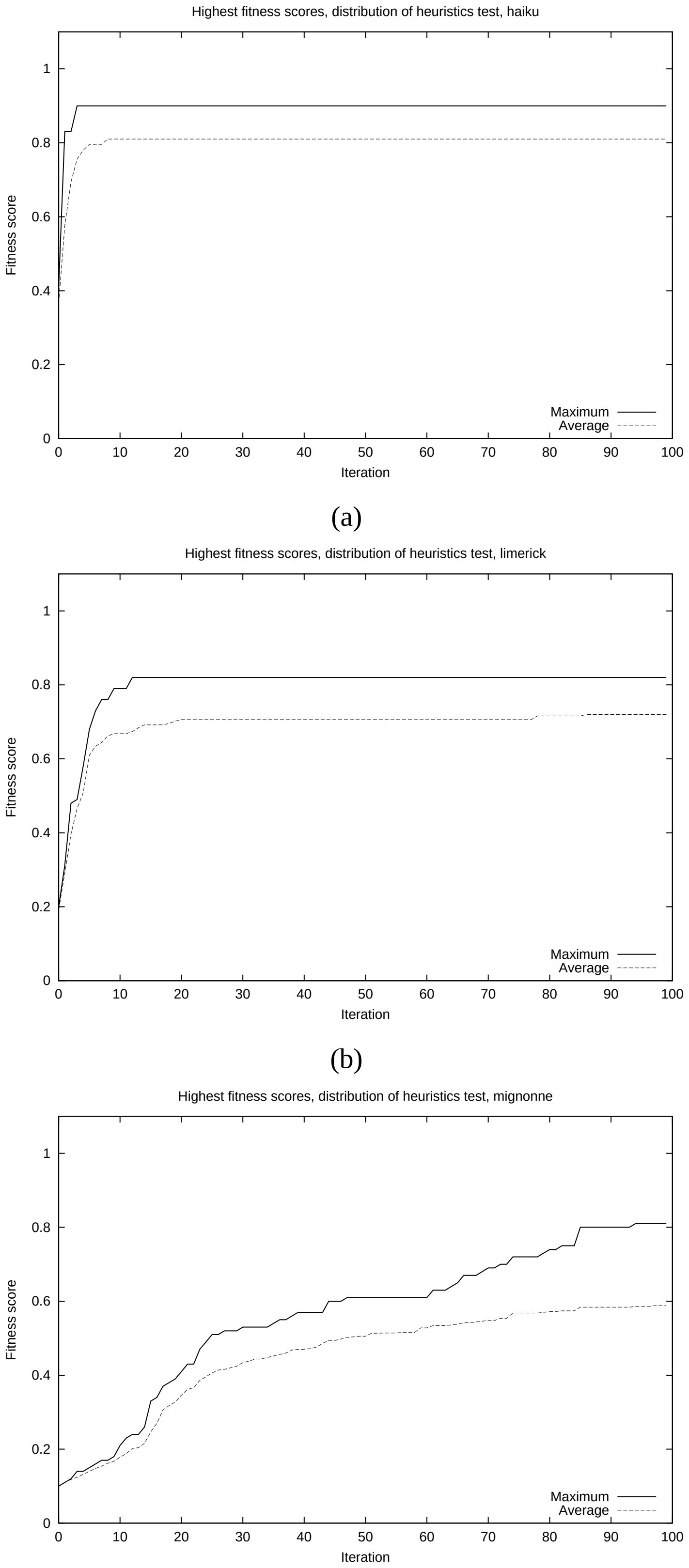
Test	Min	Max	Mean	Std.Dev
`relativity`	0.46	0.69	0.58	0.07
`relativity1`	0.80	0.82	0.81	0.01
`relativity2`	0.50	0.66	0.62	0.07
`relativity3`	0.59	0.78	0.70	0.07
`relativity4`	0.69	0.86	0.76	0.07

Table D.16: Summary statistics of best fitness scores from final populations, 'relativity' limerick generation test

test throughout the search process. Figure D.15(a), (b), and (c) show these scores for the `haiku`, `limerick`, and `mignonne` target forms respectively.

D.3.4 Line-by-line generation

Details for this test are given in Section 8.5.4. We ran this test five times, once for the entire limerick and once for each individual line. Each test was run ten times. Table D.16 shows a summary of the results obtained from this test. For each target form, we take the best fitness scores from the last populations (i.e. the population at the end of the last iteration) of the runs. We show the minimum, maximum, mean and standard deviation values for these best scores.



(a)

(b)

(c)

Figure D.15: Maximum and average of best fitness scores for F_{target} =(a)haiku, (b)limerick and (c)mignonne, distribution of heuristics test

Bibliography

Aamodt, A. and Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–59.

Angeline, P. J. (1996). Genetic programming's continued evolution. In Angeline, P. J. and Kinnear, K. E., editors, *Advances in Genetic Programming*, volume 2, pages 89–110. MIT Press, Cambridge, USA.

Appelt, D. E. (1985). *Planning English Sentences*. Cambridge University Press, Cambridge, UK.

Attridge, D. (1995). *Poetic Rhythm: an Introduction*. Cambridge University Press.

Bäck, T., Fogel, D., and Michalewicz, Z., editors (1997). *Handbook of Evolutionary Computation*. Oxford University Press and Institute of Physics Publishing.

Bailey, P. (1999). A reader-based model of story generation; or 'stories: they're not what you expected'. In *Proceedings of the AISB'99 Symposium on Creative Language: Humour and Stories*, pages 36–45, Edinburgh, UK. AISB.

Bailey, R. W. (1974). Computer-assisted poetry: the writing machine is for everybody. In Mitchell, J. L., editor, *Computers in the Humanities*, pages 283–295. Edinburgh University Press, Edinburgh, UK.

Baker, J. E. (1987). Reducing bias and inefficiency in the selection algorithm. In Grefenstette, J. J., editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 14–21, Cambridge, USA. Lawrence Erlbaum Associates.

Bangalore, S. and Rambow, O. (2000a). Exploiting a probabilistic hierarchical model for generation. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, Saarbrücken, Germany.

Bangalore, S. and Rambow, O. (2000b). Using TAG, a tree model, and a language model for generation. In *Proceedings of the Fifth Workshop on Tree Adjoining Grammars (TAG+ 5)*, Paris, France.

Beeferman, D. (1996). The rhythm of lexical stress in prose. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, Santa Cruz, UK. Association for Computational Linguistics.

Beeferman, D. (1998). Lexical discovery with an enriched semantic network. In Harabagiu, S., editor, *Use of WordNet in Natural Language Processing Systems: Proceedings of the Conference*, pages 135–141. Association for Computational Linguistics, Somerset, USA.

Belloc, J. H. P. (1991). *The bad child's book of beasts*. Jonathan Cape, London, UK.

Binsted, K. (1996). *Machine Humour: An Implemented Model of Puns*. PhD thesis, Department of Artificial Intelligence, University of Edinburgh, Edinburgh, UK.

Bod, R. and Scha, R. (1997). Corpus-based methods in language and speech processing. In Young, S. and Bloothooft, G., editors, *Data-Oriented Language Processing: An Overview*, pages 137–173. Kluwer Academic Publishers, Boston, USA.

Boden, M. A. (1990). *The Creative Mind: Myths and Mechanisms*. Weidenfeld and Nicolson, London, UK.

Bouayad-Agha, N., Scott, D., and Power, R. (2000). Integrating content and style in documents: a case study of patient information leaflets. *Information Design Journal*, 9(2-3):161–176.

Brew, C. (1992). Letting the cat out of the bag: Generation for shake-and-bake MT. In *Proceedings of the 14th International Conference on Computational Linguistics*, pages 29–34, Nantes, France.

Bunke, H. and Shearer, K. (1998). A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*, 19:255–259.

Cahill, L. and Reape, M. (1999). Component tasks in applied NLG systems. Technical Report ITRI-99-05, Information Technology Research Institute, University of Brighton, Brighton, UK.

Candito, M.-H. and Kahane, S. (1998). Can the TAG derivation tree represent a semantic graph? an answer in the light of meaning-text theory. In *Proceedings of the Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4)*, pages 21–24, Philadelphia, USA.

Carroll, J. A., Copestake, A., Flickinger, D., and Poznański, V. (1999). An efficient chart generator for (semi-)lexicalist grammars. In *Proceedings of the 7th European Workshop on Natural Language Generation*, pages 86–95, Toulouse, France.

Cheng, H. (2002). *Modelling Aggregation Motivated Interactions in Descriptive Text Generation*. PhD thesis, Division of Informatics, University of Edinburgh, Edinburgh, UK.

Copestake, A., Flickinger, D., Malouf, R., Riehemann, S., and Sag, I. A. (1995). Translation using minimal recursion semantics. In *Proceedings of the Sixth International Conference on Theoretical and Methodological Issues in Machine Translation (TMI95)*, Leuven, Belgium.

Dale, R. and Haddock, N. (1991). Content determination in the generation of referring expressions. *Computational Intelligence*, 7(4):252–265.

Daly, A. (1984). *Animal Poems*. Ladybird Books, Loughborough, UK.

Davis, L. and Steenstrup, M. (1987). Genetic algorithms and simulated annealing: An overview. In Davis, L., editor, *Genetic Algorithms and Simulated Annealing*, pages 1–11. Pitman, London, UK.

Dawkins, R. (1991). *The Blind Watchmaker*. Penguin, London, UK.

de Kleer, J. (1986). An assumption-based truth maintenance system. *Artificial Intelligence*, 28(2):127–162.

De Smedt, K., Horacek, H., and Zock, M. (1996). Architectures for natural language generation: Problems and perspectives. In Adorni, G. and Zock, M., editors, *Trends in Natural Language Generation: An Artificial Intelligence Perspective*, number 1036 in Springer Lecture Notes in Artificial Intelligence, pages 17–46. Springer-Verlag, Berlin, Germany.

Diaz-Agudo, B., Gervás, P., and González-Calero, P. (2002). Poetry generation in COLIBRI. In *Proceedings of the 6th European Conference on Case Based Reasoning (ECCBR 2002)*, Aberdeen, UK.

Dowty, D. R., Wall, R. E., and Peters, S. (1981). *Introduction to Montague Semantics*. Reidel, Dordrecht, The Netherlands.

Dras, M. (1999). *Tree Adjoining Grammar and the Reluctant Paraphrasing of Text*. PhD thesis, Macquarie University, Australia.

Eddy, B. (2002). Towards balancing conciseness, readability and salience: an integrated architecture. In *Proceedings of the Second International Natural Language Generation Conference*, pages 173–178, Harriman, USA.

Eddy, B., Bental, D., and Cawsey, A. (2001). An algorithm for efficiently generating summary paragraphs using tree-adjoining grammar. In *Proceedings of the 8th European Workshop On Natural Language Generation*, Toulouse, France. Association for Computational Linguistics.

Evans, R., Piwek, P., and Cahill, L. (2002). What is NLG? In *Proceedings of the Second International Natural Language Generation Conference*, pages 144–151, Harriman, USA.

Falkenhainer, B., Forbus, K. D., and Gentner, D. (1989). The structure-mapping engine: Algorithm and examples. *Artificial Intelligence*, 41:1–63.

Fellbaum, C., editor (1998). *WordNet: An Electronic Lexical Database*. MIT Press.

Fogel, D. B. (1995). *Evolutionary Computation: Towards a New Philosophy of Machine Intelligence*. IEEE Press, New York, USA.

Fonseca, C. M. and Fleming, P. J. (1995). An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16.

Foo, N., Garner, B. J., Rao, A., and Tsui, E. (1992). Semantic distance in conceptual graphs. In Gerhotz, L., editor, *Current Directions in Conceptual Structure Research*, pages 149–154. Ellis Horwood.

Gardent, C. (2002). Generating minimal definite descriptions. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, USA. ACL.

Gazdar, G. and Mellish, C. (1989). *Natural Language Processing in PROLOG: an Introduction to Computational Linguistics*. Addison-Wesley.

Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2):155–170.

Gervás, P. (2000). WASP: Evaluation of different strategies for the automatic generation of spanish verse. In *Proceedings of the AISB'00 Symposium on Creative and Cultural Aspects and Applications of AI and Cognitive Science*, Birmingham, UK. AISB.

Gervás, P. (2001). An expert system for the composition of formal spanish poetry. *Journal of Knowledge-Based Systems*, 14(3-4):181–188.

Gervás, P. (2002). Exploring quantitative evaluations of the creativity of automatic poets. In *Proceedings of the 2nd. Workshop on Creative Systems, Approaches to Creativity in Artificial Intelligence and Cognitive Science, 15th European Conference on Artificial Intelligence (ECAI 2002)*, Lyon, France.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, USA.

Goldberg, E., Driedger, N., and Kittredge, R. I. (1994). Using natural-language processing to produce weather forecasts. *IEEE Expert*, 9(2):45–53.

Grice, H. (1975). Logic and conversation. In Cole, P. and Morgan, J., editors, *Syntax and Semantics Vol. 3: Speech Acts*, pages 41–58. Academic Press, New York, USA.

Grosz, B. J., Joshi, A. K., and Weinstein, S. (1995). Centering: a framework for modelling the local coherence of discourse. *Computational Linguistics*, 21(2):203–226.

Gruber, H. and Davis, S. (1988). Inching our way up mount olympus: The evolving systems approach to creative thinking. In Sternberg, R. J., editor, *The Nature of Creativity*, pages 243–269. Cambridge University Press, New York, USA.

Hartman, C. O. (1996). *Virtual Muse: Experiments in Computer Poetry*. Wesleyan University Press.

Hayes, B. (1995). *Metrical Stress Theory: Principles and Case Studies*. University of Chicago Press.

Hobbs, J. (1985). Ontological promiscuity. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, pages 61–69, Chicago, USA. The Association for Computational Linguistics.

Hobbs, J. R. (1978). Resolving pronoun references. *Lingua*, 44(4):311–338.

Hobsbaum, P. (1996). *Metre, Rhythm, and Verse Form*. Routledge.

Hofstadter, D. R., editor (1997). *Le Ton Beau de Marot: In Praise of the Music of Language*. Basic Books, New York, USA.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press, Ann Arbor, USA.

Holmes, D. I. (1998). The evolution of stylometry in humanities scholarship. *Literary and Linguistic Computing*, 13(3):111–117.

Hovy, E. (1990). Unresolved issues in paragraph planning. In Dale, R., Mellish, C., and Zock, M., editors, *Current Research in Natural Language Generation*, pages 17–45. Academic Press, London, UK.

Johnson-Laird, P. N. (1988). *The Computer and the Mind: an Introduction to Cognitive Science*. Harvard University Press, Cambridge, USA.

Joshi, A. K. (1987). The relevance of tree adjoining grammars to generation. In Kempen, G., editor, *Natural Language Generation: New Results in Artificial Intellligence*, pages 233–252. Martinus Nijhoff Press, Dordrecht, The Netherlands.

Joshi, A. K. and Schabes, Y. (1991). Tree-adjoining grammars and lexicalized grammars. Technical Report IRCS-91-04, Institute for Research in Cognitive Science, University of Pennsylvania.

Joshi, A. K. and Schabes, Y. (1997). Tree-adjoining grammars. In Rozenberg, G. and Salomaa, A., editors, *Handbook of Formal Languages*, volume 3: Beyond Words, pages 69–123. Springer-Verlag, Berlin, Germany.

Jurafsky, D. S. and Martin, J. H. (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice-Hall.

Kallmeyer, L. (2002). Using an enriched TAG derivation structure as basis for semantics. In *Proceedings of the Sixth International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+6)*, pages 101–110, Universita di Venezia.

Kamal, H. (2002). *An ATMS-Based Architecture for Stylistics-Aware Text Generation*. PhD thesis, Division of Informatics, University of Edinburgh, Edinburgh, UK.

Kamp, H. and Reyle, U. (1993). *From Discourse to Logic: Introduction to Model-theoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Kantrowitz, M. and Bates, J. (1992). Integrated natural language systems. In Dale, R., Hovy, E., Rösner, D., and Stock, O., editors, *Aspects of Automated Natural Language Generation: Proceedings of the Sixth International Workshop on Natural Language Generation*, number 587 in Springer Lecture Notes in Artificial Intelligence, pages 13–28, Trento, Italy. Springer-Verlag.

Karamanis, N. (2001). Exploring entity-based coherence. In *Proceedings of the Fourth Annual CLUK Research Colloquium*, pages 18–26, University of Sheffield, UK.

Karamanis, N. and Manurung, H. M. (2002). Stochastic text structuring using the principle of continuity. In *Proceedings of the Second International Natural Language Generation Conference*, pages 81–88, Harriman, USA.

Kay, M. (1996). Chart generation. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 200–204, Santa Cruz, USA. ACL.

Kempe, V., Levy, R., and Graci, C. (2001). Neural networks as fitness evaluators in genetic algorithms: Simulating human creativity. In Moore, J. D. and Stenning, K., editors, *Proceedings of the 23rd Annual Conference of the Cognitive Science Society*, Edinburgh, UK. Cognitive Science Society, Lawrence Erlbaum Associates. Poster Session 1.

Kibble, R. and Power, R. (2000). An integrated framework for text planning and pronominalisation. In *Proceedings of the First International Conference on Natural Language Generation (INLG-2000)*, pages 77–84, Mitzpe Ramon, Israel.

Kilger, A. (1992). Realization of tree adjoining grammars with unification. Technical Report TM-92-08, DFKI, Saarbrücken, Germany.

Knight, K. and Hatzivassiloglou, V. (1995). Two-level, many-paths generation. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 252–260, Cambridge, USA.

Knott, A., Oberlander, J., O'Donnell, M., and Mellish, C. (2001). Beyond elaboration: The interaction of relations and focus in coherent text. In Sanders, T., Schilperoord, J., and

Spooren, W., editors, *Text Representation: Linguistic and Psycholinguistic Aspects*, pages 181–196. Benjamins, Amsterdam, The Netherlands.

Knowles, E., editor (1999). *The Oxford dictionary of quotations*. Oxford University Press, Oxford, UK, fifth edition.

Koller, A. and Striegnitz, K. (2002). Generation as dependency parsing. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*, Philadelphia, USA.

Koza, J. R. (1994). Introduction to genetic programming. In Kenneth E. Kinnear, J., editor, *Advances in Genetic Programming*, pages 21–42. MIT Press.

Kurzweil, R. (2001). Ray kurzweil's cybernetic poet. `http://www.kurzweilcyberart.com/poetry`.

Langkilde, I. and Knight, K. (1998). Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 704–710, Montreal, Canada.

Lear, E. (1947). *The Complete Nonsense of Edward Lear*. Faber and Faber, London.

Legman, G., editor (1974). *The Limerick: 1700 Examples, with Notes, Variants and Index*. Jupiter Books Ltd., London, UK.

Levin, S. R. (1962). *Linguistic Structures in Poetry*. Number 23 in Janua Linguarum. 's-Gravenhage.

Levy, R. P. (2001). A computational model of poetic creativity with neural network as measure of adaptive fitness. In Bento, C. and Cardoso, A., editors, *Proceedings of the Fourth International Conference on Case Based Reasoning (ICCBR'01) Workshop on Creative Systems: Approaches to Creativity in AI and Cognitive Science*, Vancouver, Canada.

Love, B. C. (2000). A computational level theory of similarity. In *Proceedings of the 22nd Annual Meeting of the Cognitive Science Society*, pages 316–321, Philadelphia, USA.

Luger, G. F. and Stubblefield, W. A. (1998). *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. Addison Wesley Longman, Inc., third edition.

Mann, W. C. and Thompson, S. A. (1987). Rhetorical structure theory: A framework for the analysis of texts. Technical Report RS-87-185, USC Information Science Institute, Marina Del Rey, UK.

Manning, C. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, USA.

Manurung, H. M. (1999). A chart generator for rhythm patterned text. In *Proceedings of the First International Workshop on Literature in Cognition and Computer*, Tokyo, Japan.

Manurung, H. M., Ritchie, G., and Thompson, H. (2000). Towards a computational model of poetry generation. In *Proceedings of the AISB'00 Symposium on Creative and Cultural Aspects and Applications of AI and Cognitive Science*, pages 79–86, Birmingham, UK. AISB.

Marcu, D. (1997). From local to global coherence: A bottom-up approach to text planning. In *Proceedings of AAAI-97*. American Association for Artificial Intelligence.

Markert, K. and Hahn, U. (2002). Understanding metonymies in discourse. *Artificial Intelligence*, 135(1-2):145–198.

Marzal, A. and Vidal, E. (1993). Computation of normalized edit distance and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):926–932.

Masterman, M. (1971). Computerized haiku. In Reichardt, J., editor, *Cybernetics, Art and Ideas*, pages 175–183. New York Graphic Society Ltd., Greenwich, UK.

McKeown, K. R. (1985). *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*. Cambridge University Press, Cambridge, UK.

McKeown, K. R. and Swartout, W. R. (1988). Language generation and explanation. In Zock, M. and Sabah, G., editors, *Advances in Natural Language Generation: An Interdisciplinary Perspective*.

Mellish, C. and Dale, R. (1998). Evaluation in the context of natural language generation. *Computer Speech and Language*, 12(4):349–373.

Mellish, C., Knott, A., Oberlander, J., and O'Donnell, M. (1998a). Experiments using stochastic search for text planning. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, Niagara-on-the-Lake, Canada.

Mellish, C., O'Donnell, M., Oberlander, J., and Knott, A. (1998b). An architecture for opportunistic text generation. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, Niagara-on-the-Lake, Canada.

Meteer, M. (1991). Bridging the generation gap between text planning and linguistic realisation. *Computational Intelligence*, 7(4):296–304.

Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, New York, USA, third edition.

Minsky, M. (1963). Steps towards artificial intelligence. In Feigenbaum, E. A. and Feldman, J., editors, *Computers and Thought*, pages 406–450. McGraw-Hill.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, USA.

Moore, J. D. and Paris, C. L. (1993). Planning text for advisory dialogues: Capturing intentional and rhetorical information. *Computational Linguistics*, 19:651–694.

Nicolov, N. (1998). *Approximate Text Generation from Non-Hierarchical Representations in a Declarative Framework*. PhD thesis, Department of Artificial Intelligence, University of Edinburgh, Edinburgh, UK.

Nicolov, N., Mellish, C., and Ritchie, G. (1995). Sentence generation from conceptual graphs. In *Proceedings of the International Conference on Conceptual Structures*, number 954 in Lecture Notes in Artificial Intelligence, Santa Cruz, USA. Springer-Verlag.

Nicolov, N., Mellish, C., and Ritchie, G. (1996). Approximate generation from non-hierarchical representations. In *Proceedings of the Eighth International Workshop on Natural Language Generation*, pages 31–40, Brighton, UK.

Nirenburg, S., Lesser, V., and Nyberg, E. (1989). Controlling a language generation planner. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 1524–1530, Detroit, USA.

Oberlander, J. and Brew, C. (2000). Stochastic text generation. *Philosophical Transactions of the Royal Society of London, Series A*, 358:1373–1385.

O'Donnell, M., Mellish, C., Oberlander, J., and Knott, A. (2001). ILEX: An architecture for a dynamic hypertext generation system. *Natural Language Engineering*, 7(3):225–250.

Paiva, D. S. (1998). A survey of applied natural language generation systems. Technical Report

ITRI-98-03, Information Technology Research Institute, University of Brighton, Brighton, UK.

Pease, A., Winterstein, D., and Colton, S. (2001). Evaluating machine creativity. Technical Report EDI-INF-RR-0054, Division of Informatics, University of Edinburgh, Edinburgh, UK.

Pedersen, C. N. S. (1999). *Algorithms in Computational Biology*. PhD thesis, Department of Computer Science, University of Aarhus, Aarhus, Denmark.

Pérez y Pérez, R. and Sharples, M. (1999). Mexica: A computational model of the process of creative writing. In *Proceedings of the AISB'99 Symposium on Creative Language: Humour and Stories*, pages 46–51, Edinburgh, UK. AISB.

Pollard, C. and Sag, I. (1994). *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

Popowich, F. (1996). A chart generator for shake and bake machine translation. In McCalla, G., editor, *Proceedings of the 11th Biennial Conference of the Canadian Society for Computational Studies of Intelligence (AI '96)*, Toronto, Canada.

Power, R. (2000). Planning texts by constraint satisfaction. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING-2000)*, pages 642–648, Saarbrücken, Germany.

Prevost, S. (1996). An information structural approach for spoken language generation. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 294–301, Santa Cruz, USA. ACL.

Prince, A. and Smolensky, P. (1993). Optimality theory: Constraint interaction in generative grammar. Technical Report RuCCS-TR-2, Rutgers University Center for Cognitive Science, New Jersey, USA.

Quinn, A. (1982). *Figures of Speech: 60 Ways to Turn a Phrase*. Hermagoras Press.

Reiter, E. (1994). Has a consensus on NL generation appeared? and is it psycholinguistically plausible? In *Proceedings of the Seventh International Natural Language Generation Workshop*, pages 163–170, Kennebunkport, USA. Springer-Verlag.

Reiter, E. and Dale, R. (2000). *Building Natural Language Generation Systems*. Cambridge University Press, Cambridge, UK, first edition.

Reithinger, N. (1991). POPEL - a parallel and incremental natural language generation system. In Paris, C. L., Swartout, W. R., and Mann, W. C., editors, *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, pages 179–199. Kluwer Academic Publishers, Boston, USA.

Ristad, E. S. and Yianilos, P. N. (1998). Learning string edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(2):522–532.

Ritchie, G. (2001). Assessing creativity. In *Proceedings of the AISB'01 Symposium on Artificial Intelligence and Creativity in Arts and Science*, pages 3–11, York, UK. The Society for the Study of Artificial Intelligence and the Simulation of Behaviour.

Roberts, P. D. (1986). *How Poetry Works: The Elements of English Poetry*. Penguin Books.

Robin, J. (1994). Revision-based generation of natural language summaries providing historical background: corpus-based analysis, design, implementation and evaluation. Technical Report CUCS-034-94, Computer Science Department, Columbia University, New York, USA. Ph.D. thesis.

Rubaud, J., Lussonnal, P., and Braffort, P. (2000). ALAMO: Atelier de littérature assisté par la mathématique et les ordinateurs. `http://indy.culture.fr/alamo/rialt/pagaccalam.html`.

Rubinoff, R. (1992). Integrating text planning and linguistic choice by annotating linguistic structures. In Dale, R., Hovy, E., Rösner, D., and Stock, O., editors, *Aspects of Automated Natural Language Generation: Proceedings of the Sixth International Workshop on Natural Language Generation*, number 587 in Springer Lecture Notes in Artificial Intelligence, pages 45–56. Springer-Verlag, Trento, Italy.

Russell, S. J. and Norvig, P. (1995). *Artificial Intelligence: a Modern Approach*. Prentice Hall, New Jersey, USA.

Sankoff, D. and Kruskal, J. B., editors (1983). *Time warps, string edits and macromolecules: the theory and practice of sequence comparison*. Addison-Wesley, Reading, USA.

Sansom, P. (1994). *Writing Poems*. Bloodaxe Books.

Schabes, Y. (1990). *Mathematical and Computational Aspects of Lexicalized Grammars*. PhD thesis, Computer Science Department, University of Pennsylvania.

Schaffer, J. D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the International Conference on Genetic Algorithms and Their Applications*, pages 93–100, Pittsburgh, USA.

Schaffer, J. D. (1987). Some effects of selection procedures on hyperplane sampling by genetic algorithms. In Davis, L., editor, *Genetic Algorithms and Simulated Annealing*, pages 89–103. Pitman, London, UK.

Sharples, M. (1996). An account of writing as creative design. In Levy, C. M. and Ransdell, S. E., editors, *The Science of Writing: Theories, Methods, Individual Differences, and Applications*. Lawrence Erlbaum Associates, Mahwah, USA.

Sims, K. (1991). Artificial evolution for computer graphics. *Computer Graphics*, 25(4):319–328.

Sowa, J. F. (1984). *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley Publishing Company.

Spears, W. M. (1992). Crossover or mutation? In Whitley, D., editor, *Proceedings of the 2nd Foundations of Genetic Algorithms Workshop*, pages 221–237, San Mateo, USA. Morgan Kaufmann.

Steedman, M. (1996). *Surface Structure and Interpretation*. MIT Press, Cambridge, USA.

Stone, M. and Doran, C. (1996). Paying heed to collocations. In *Proceedings of the Eighth International Workshop on Natural Language Generation*, pages 91–100, Brighton, UK.

Stone, M. and Doran, C. (1997). Sentence planning as description using tree adjoining grammar. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 198–205, Madrid, Spain. The Association for Computational Linguistics.

Stone, M., Doran, C., Webber, B., Bleam, T., and Palmer, M. (2001). Microplanning with communicative intentions: The SPUD system. Technical Report TR-65, Rutgers University Center for Cognitive Science, New Jersey, USA.

Thompson, H. S. (1977). Strategy and tactics: A model for language production. In *Papers from the 13th Regional Meeting of the Chicago Linguistics Society*.

Thompson, H. S. (1980). Stress and salience in english: Theory and practice. Technical Report CSL-80-8, Xerox Palo Alto Research Centre.

Turner, S. R. (1994). *The Creative Process: A Computer Model of Storytelling and Creativity*. Lawrence Erlbaum Associates.

Tversky, A. (1977). Features of similarity. *Psychological Review*, 84(4):327–352.

van Mechelen, M. V. (1992). Computer poetry. `http://www.trinp.org/Poet/ComP/ComPoe.HTM`.

Varges, S. (2002). *Instance-based Natural Language Generation*. PhD thesis, Division of Informatics, University of Edinburgh, Edinburgh, UK.

Vijay-Shanker, K. and Joshi, A. K. (1988). Feature structure based tree adjoining grammars. In *Proceedings of 12th International Conference of Computational Linguistics*, pages 714–720, Budapest, Hungary.

Wanner, L. and Hovy, E. H. (1996). The HealthDoc sentence planner. In *Proceedings of the Eighth International Workshop on Natural Language Generation*, pages 1–10, Brighton, UK.

Ward, N. (1994). *A Connectionist Language Generator*. Ablex Series in Artificial Intelligence. Ablex Publishing, Norwood, USA.

Weide, R. L. (1996). Carnegie mellon university pronouncing dictionary. `http://www.speech.cs.cmu.edu/cgi-bin/cmudict`.

Whitelock, P. (1992). Shake-and-bake translation. In *Proceedings of the 14th International Conference on Computational Linguistics*, pages 610–616, Nantes, France.

Whitley, D., Rana, S., and Heckendorn, R. B. (1997). Island model genetic algorithms and linearly separable problems. In Corne, D. and Shapiro, J. L., editors, *Evolutionary Computing: AISB International Workshop, Manchester, UK, April 1997 (Selected Papers)*, number 1305 in Lecture Notes in Computer Science, pages 109–125. Springer-Verlag.

Wiggins, G. (2001). Towards a more precise characterisation of creativity in AI. In Bento, C. and Cardoso, A., editors, *Proceedings of the Fourth International Conference on Case Based Reasoning (ICCBR'01) Workshop on Creative Systems: Approaches to Creativity in AI and Cognitive Science*, Vancouver, Canada.

XTAG Research Group (2001). A lexicalized tree adjoining grammar for english. Technical Report IRCS-01-03, IRCS, University of Pennsylvania.