UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Interactive narrative generation using computational verb theory

by

Marius Smit

Submitted in partial fulfilment of the requirements for the degree
Master of Science (Computer Science)
in the Faculty of Engineering, Built Environment and Information Technology
University of Pretoria, Pretoria

June 2009

Publication data:

Marius Smit. Interactive narrative generation using computational verb theory. Master's dissertation, University of Pretoria, Department of Computer Science, Pretoria, South Africa, June 2009.

Electronic, hyperlinked versions of this dissertation are available online, as Adobe PDF files, at:

http://cirg.cs.up.ac.za/

http://upetd.up.ac.za/UPeTD.htm

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Interactive Narrative Generation using Computational Verb Theory

by

Marius Smit
E-mail: marius.smit84@gmail.com

Abstract

Interactive narrative extends traditional story-telling techniques by enabling previously passive observers to become active participants in the narrative events that unfold. A variety of approaches have attempted to construct such interactive narrative spaces and reconcile the goals of interactivity and dramatic story-telling. With the advent of the linguistic variable in 1972, a means was established for modelling natural language words and phrases mathematically and computationally. Over the past decade, the computational verb, first introduced in 1997, has been developed as a mathematical means of modelling natural language verbs in terms of dynamic systems, and vice versa. Computational verb theory extends the initial concept of the linguistic variable beyond being able to model adjectives, nouns, and passive states, into the realm of actions as denoted by natural language verbs. This thesis presents the framework and implementation of a system that generates interactive narrative spaces from narrative text. The concept of interactive narrative is introduced and recent developments in the area of interactive narrative are discussed. Secondly, a brief history of the development of the linguistic variable and the computational verb are provided. With the context of the computational verb (interactive) narrative generation (CVTNG) system presented, the underlying theoretical principles of the system are established. The CVTNG system principles are described in terms of fuzzy set, computational verb, and constraint satisfaction theory. The fuzzy set, computational verb, and constraint satisfaction principles are organised according to a CVTNG architecture. The CVTNG architecture is then described in terms of its subsystems, structures, algorithms, and interfaces. Each CVTNG system component

is related to the overall design considerations and goals. A prototype of the CVTNG system is implemented and tested against a suite of natural language sentences. The behaviour and performance of the CVTNG system prototype are discussed in relation to the CVTNG system's design principles. Results are calculated and stored as variable values that are dynamically and generically associated with representational means, specifically computer graphics, to illustrate the generation of interactive narrative spaces. Plans for future work are discussed to show the immense development potential of this application. The thesis concludes that the CVTNG system provides a solid and extendable base for the intuitive generation of interactive narrative spaces from narrative text, computational verb models, and freely associated media.

Keywords: computational verb theory, fuzzy sets, interactive narrative, computational linguistics, constraint satisfaction problems, fuzzy constraint satisfaction problems, computer graphics.

Supervisor : Prof. AP Engelbrecht
Department : Department of Computer Science
Degree : Master of Science

Contents

List of Figures

List of Algorithms

List of Tables

Chapter 1

Introduction

This chapter introduces the fields related to this work in terms of their applications and background. The subject of this thesis is related to several fields through the latter's applications and background. The contributions of the current work are related to the fields of research introduced. The objectives of this thesis and an outline of the content are also set out.

1.1 Motivation and related work

Since the advent of computers in the 20th century and the conception of the first text-based adventure game, *Hunt the Wumpus* (1972), and the better–known *Zork I* (1977-1979), interactive fiction, also known as interactive narrative, has become an active area of interest in the fields of computational linguistics, artificial intelligence (AI), and computer graphics. Early works of interactive fiction were completely word-based in terms of their input and output. As the quality of computer graphics and sound improved and the popularity of arcade (video-based) games grew during the 1980s, word-based narrative and interactive narrative became more disconnected [58]. This trend grew into the modern-day standard of graphically constructing and representing narrative within interactive fiction [36].

The fuzzy set [127], with its associated fuzzy logic, was introduced by Lotfi Zadeh in 1965. Fuzzy sets provide a generalised means of representing approximate concepts,

often expressed only in terms of words. Linguistic variables [128] [129] build upon the power of fuzzy sets to represent approximate concepts and allow for the quantitative interpretation of word types such as nouns, adjectives, and adverbs. Fuzzy logic [129] enables logical reasoning by means of the fuzzy set models of nouns, adjectives, and adverbs as a generalised extension of boolean and multi–valued logics. Over the last 30 years fuzzy sets and their associated fuzzy logic and fuzzy systems have remained an active area of interest. Fuzzy sets, fuzzy logic, and fuzzy systems have been integrated into multiple scientific research fields which deal with concepts that are inherently imprecise or difficult to quantify. Applications of fuzzy set theory include control theory [25], data analysis [8], and AI [50].

Yang introduced the computational verb as a means of solving engineering and dynamics problems by modelling them in relation to natural language verbs [102] [104] [107]. Computational verb theory (CVT) extends fuzzy set theory to make verbs in natural language measurable, quantifiable, and therefore computable. Fuzzy sets and linguistic variables model nouns and their associated adjectives (noun phrases). CVT models verbs, verb phrases, and their associated adverbs. It has been applied to the solution of logical paradoxes [108], fuzzy dynamic systems [113], control theory [109] [116] [111], image processing and understanding [115] [119], and the modelling of stock markets [118] [121]. Multiple commercial image recognition and financial modelling software products are built on computational verb principles.

This work models natural language nouns, adjectives, verbs, and adverbs in terms of CVT. CVT models for natural language verbs are constructed, and the CVT models are subsequently evaluated to determine values that parameterise interactive narrative space generation.

Constraint satisfaction problem (CSP) theory is applied within the scope of this thesis and its associated computational verb theory interactive narrative generation (CVTNG) prototype. Constraint satisfaction was first introduced by Huffman [44] and Clowes [18] in the context of line labelling in visual scenes. The determination of the valid labellings, given a set of constraints, forms the basis of the CSP. Ruttkay [79] defined the fuzzy constraint satisfaction problem (FCSP) as a relaxation of the classic CSP by applying the concept of fuzzy sets to CSPs. Fuzzy constraints are formed by measuring the degree

to which a value tuple assignment corresponds to a constraint as a real number on the interval $[0, 1]$. Fuzzy constraint satisfaction is applied within this work as a strategy for the approximate resolution of the variable values present in models formed from conflicting natural language statements.

1.2 Contribution of the current work to the related fields of interest

The fields of fuzzy set theory, CVT, CSP theory, and computational linguistics are applied and combined within this work in a strategy that allows for the translation of narrative text to a representational means commonly used within interactive narrative. The narrative space is generated by parameterising transformations on representational means with values obtained from the evaluation of CVT models. The computational verb models formed for natural language words are solved as a system of linear equations. If such a system cannot be solved using linear resolution methods such as Gaussian elimination, fuzzy constraint satisfaction techniques are applied to obtain an approximate solution.

This thesis draws from and forms part of the field of computational linguistics [68]. It describes an architecture that allows for computational linguistics algorithms to be incorporated into the procedure that forms word groupings. The groupings formed using computational linguistics algorithms are converted to the aforementioned computational verb models. Furthermore, this thesis presents a novel application of computational linguistics to the field of interactive narrative, by serving not only as an interaction input to an interactive narrative system, but as a creation tool for the interactive space.

The computational verb models dynamically constructed for natural language words are organised according to hierarchies of interpretive contexts. This organisation of computational verb models into hierarchies of interpretive contexts is a novel scheme for organising and combining the interpretive contexts of computational verb sets. This work presents a crisp variable resolution strategy for the resolution of the system of equations that results from the formation of computational verb models for natural language words. An alternative fuzzified variable resolution strategy is provided in cases

where a single unique solution to a system of equations that results from computational verb models does not exist. The fuzzy variable resolution strategy presents a novel scheme for determining approximate values of conflicting evolving function equations [106] in the field of CVT.

The variable values obtained from either resolution strategy are transferred to a dynamically generated interface that enables the free association of representational media such as computer graphics, sound, and software agents with the system. The present work offers novel applications of CVT and computational linguistics to interactive narrative space creation, and presents a framework for the future development of CVT narrative generation systems. It also presents the results of tests performed successfully on a prototype implemented within the presented framework against a series of visual narrative interpretation examples.

1.3 Thesis outline

Chapter 2 summarises recent work in the field of interactive narrative that highlights their approach and contribution within the visual paradigm of interactive narrative construction and representation. Visually modelled interactive narrative systems are contrasted with a word-based interactive narrative generation system to illustrate the advantage of a system that enables natural language to interactive narrative space translation via the re-use of representational means such as graphics, sound, and AI story agents.

Chapter 3 sets out the definitions of the computational verb, fuzzy set, and constraint satisfaction theory concepts related to the principles applied within this work and its associated interactive narrative generation system. Chapter 4 draws on the background provided in chapter 3 to state the theoretical principles that enable natural language sentences to be translated to alternative representational media such as graphics, sound, and AI agents.

The interactive narrative generation systems architecture is discussed in detail in chapter 5. Details of the parsing, variable resolution, and representation interfacing subsystems are provided and discussed with regard to their structure, relationships, and design considerations.

Chapter 6 tests a prototype of the implemented architecture against a series of natural language sentences. The sentences tested against the prototype represent a selection of scenarios that a natural language translator such as the interactive narrative generation system presented in this work should be able to handle in a robust and adaptable manner. The tested examples serve to illustrate the ability of a computational verb-based system to model natural language sentences dynamically as systems of equations obtained from computational verb models. The tested examples in chapter 6 also demonstrate the ability of the interactive narrative generation system's architecture to interface the results dynamically and freely to representational media such as graphics, sound, and AI agents.

Chapter 7 summarises the content of this thesis and presents conclusions arising out of the work presented. The chapter also outlines plans for future work on the theoretical, architectural, and application components of this work.

Chapter 2

Related work

This chapter presents a summary of current work in the field of interactive narrative. The discussion of interactive narrative takes place in light of recent research developments, the application of interactive narrative to other areas of research and the application of fuzzy set theory to interactive narrative. A comparison is presented between the current visual paradigms of interactive narrative construction and the word-based narrative construction paradigm proposed in this work. This comparison serves to highlight the unique advantages of a word-based narrative construction paradigm.

2.1 Interactive narrative

This section discusses work in the field of interactive narrative related to the interactive narrative concepts applied in this work. Recent research trends in the field of interactive narrative are discussed in section 2.1.1. The application of interactive narrative to other fields of research interest is discussed in section 2.1.2; of fuzzy set theory to interactive narrative in section 2.1.3. This thesis presents a novel application of computational verb theory (CVT) to interactive narrative, and therefore no discussion of the application of CVT to interactive narrative is provided, as no such existing applications exist.

2.1.1 Research trends in interactive narrative

The main focus of interactive narrative centres on the problem of balancing the conflicting goals of interactivity, and the need to communicate a dramatic plot [94]. Recent work in the field addresses these issues using a variety of approaches that draw on the fields of computer science (artificial intelligence (AI) and multi–agent systems) and the arts (literature and drama).

One area of focus is the visual modelling of dynamic and branching story arcs, portrayed using media such as video and sound. "*Agent Stories*" [13] enables writers of cinematic narrative to structure a non-linear narrative visually, and then link the structure to text or video–based outputs for computational execution and output of the defined structure. "StorySpace" [96] presents a visual interface for representing collections of new media, such as audio and video recordings, in a way that conveys narrative within a classroom situation. Cavazza *et al* [16] have presented a system that relates multi–modal input (speech, gesture) to a defined narrative structure to be interpreted as the overall plot progresses.

Other systems approach the creation of interactive narrative more directly, representing the interactive narrative in real time by using a computer graphics engine and AI story agents [61] [62] [75] [126]. One of the first and most influential systems of this nature is the Oz system [61]. It fuses the goals of believable agents (characters that act in an emotionally believable way) and interactive drama (the dramatic development of a story arc). The objective of believable agents is achieved by developing story agents that are believable in terms of their portrayal, behaviour, and interaction with the user. The objective of presenting interactive drama is achieved through the use of a "drama manager" that controls the flow of narrative from one plot point to another. The Mimesis system [75] [126] integrates existing commercial 3D graphics technology with a narrative engine that controls story arc by structuring player interaction and system reaction. The narrative engine allows for the progression of a story plot in a meaningful and affecting way. Timegraph [46] attempts to integrate narrative into virtual reality representations, by allowing changes in the environment over time and in accordance with user interaction. *Facade* [62] builds on the technologies of the Oz project [61] to realise an interactive fiction game fully. Autonomous agents behave in accordance with

the plot structure and natural language inputs from the user to create an interactive drama. A notable point of contrast between existing interactive narrative systems and the interactive narrative generation system presented in this work is the auxiliary role of text within existing interactive narrative systems. Text is used within the majority of interactive narrative systems as an annotation or a means of interaction, not as the driver for interactive narrative creation.

Another active area of interest within the field of interactive narrative involves the concept of narrative mediation [76]. This is a technique in which interactive narrative system components such as AI story agents are used to guide the user should the user's actions threaten the integrity of the plot. Young *et al* [76] look more formally at the branching story arcs within Oz, Mimesis, and *Facade* and then apply the concept of linear narrative generation [63] by reconstructing the non-linear narratives of the Oz, Mimesis, and *Facade* systems in terms of multiple linear narratives. The narrative mediation scheme proposed by Young *et al* allows non-linear narrative to leverage some of the advantages of linear narrative. The "*Death Kitchen*" [57] interactive narrative scenario extends the principles of narrative mediation by story agents (non-user characters), as in Oz, Mimesis, and *Facade*, to the virtual environment itself. The milieu is therefore transformed into a plot shaping–device.

In contrast to the top–down approach of the systems presented above, *FearNot!* [6] follows a bottom-up approach by creating an environment for emergent narratives by employing "affectively driven" autonomous agents. The emergent narrative of the *FearNot!* system is then applied to educating young children against bullying behaviour. Steiner *et al* [88] attempt not to adapt narrative according to user interaction (narrative mediation), but try instead to reshape the way in which the narrative is conveyed. The way the narrative is conveyed is dynamically adjusted to user interaction to ensure that the user is aware of the unfolding plot, thus preserving the narrative structure.

The main areas of research interest within interactive narrative have been presented in terms of recent work. The section that follows briefly discusses the applications of interactive narrative to other fields of research such as occupational therapy [21] and education [96] [6].

2.1.2 Applications of interactive narrative

Interactive narrative clearly has entertainment potential, as seen in *Facade* [62] and commercial interactive fiction games such as *Gabriel Knight*™and *Monkey Island*™. *FearNot!* [6] and StorySpace [96] both show how interactive narrative can be used within educational contexts. Robertson *et al* [36] show how the creation of interactive narrative can serve as an educational process. A study was conducted in which children constructed stories using the editing tools of a commercial computer game. Robertson *et al* observed that the activity encouraged creativity and the acquisition of technical skills in the children who participated. Davis *et al* [21] presented *TouchStory*, a game that presents simple sequences of symbolic narratives called t-stories to encourage the learning of narrative concepts in children with autism.

The state of the art in interactive narrative and examples of applications have been presented. The next section presents related work in which fuzzy set theory was implemented in the creation of interactive narrative systems or subsystems.

2.1.3 Fuzzy set theory applied to interactive narrative

Fuzzy set theory has found a wide range of uses in computer game AI [73], which in turn serves as a form of interactive narrative. Fuzzy set theory has been applied directly to interactive narrative in the form of fuzzy cognitive maps [5]. The fuzzification of cognitive maps allows the behaviours and emotions of autonomous agents [22] [70] to be modelled as fuzzy cognitive maps [49], which have also been applied to narrative control as an alternative to hierarchal task networks [15]. The fuzzified narrative control system allows for dynamic story branching in interactive narrative systems that require real–time feedback [14]. Su *et al* [89] have implemented a multiple input, multiple output fuzzy logic hierarchy for categorising story characters and story arcs. The categorisations are used to determine the posturing and timing of character animations in interactive narrative systems.

2.1.4 Summary

This section discussed related work in the field of interactive narrative. Applications of interactive narrative to other fields of interest were presented. Examples were given of fuzzy set theory as applied to interactive narrative. The section that follows contrasts current paradigms for the creation of interactive narrative with a paradigm that models interactive narrative in terms of natural language words.

2.2 A word-centred paradigm

This section discusses the advantages of a top–down word-based approach to constructing interactive narrative spaces. The interactive narrative generation system described in this thesis generates interactive narrative depictions from natural language sentences. A top–down approach to narrative generation is contrasted with the current bottom-up approach to interactive narrative creation that directly constructs interactive narrative spaces using narrative depictions such as graphics, sound, and AI agents.

Systems that follow the approach of modelling in terms of narrative depictions usually relegate narrative text to the secondary role of representing dialogue in the absence of sound [13]. Interactive narrative systems such as *Facade* [62] utilise natural language text as an input to the system. The system presented in this work uses natural language text as the basic modelling unit for describing interactive narrative spaces. A word-centric approach presents unique advantages to narrative generation systems as presented in the sections that follow.

Section 2.2.1 argues that narrative text is a natural means of representing shared human experiences as conveyed in narrative. Section 2.2.2 shows that narrative text is a concise means of representing narrative. Section 2.2.3 describes how a system that generates narrative depictions from narrative text allows for the reinterpretation of a fixed narrative to convey a new message. Section 2.2.4 argues that a system which generates interactive narrative depictions from narrative text, opens up interactive narrative creation to a wider audience than is possible using bottom-up interactive narrative construction.

2.2.1 Language captures shared human experiences

Natural language has been the basic means of communication between human beings for thousands of years. Before the invention of writing, oral accounts served as a traditional means of propagating knowledge from one generation to the next. With the invention of writing and the subsequent invention of the printing press, natural language became the world wide standard for capturing and preserving knowledge. Narrative is a natural fit for language, since language itself evolved for the very purpose of transferring narrative from one individual to another.

The value of visually modelling abstract concepts is undeniable for designing complex systems, as can be seen by the success of modelling languages such as unified modelling language (UML) [32]. It seems counter–intuitive, however, to apply visual models that serve to capture abstract systemic knowledge to something as innately human and emotional as narrative. If interactive narrative spaces are constructed from systemic visual models such as UML, the value of thousands of years of collective tacit knowledge captured within natural languages is lost.

To illustrate the value of the tacit knowledge captured within natural language words, consider the sentence "Tom loves Katie". The sentence "Tom loves Katie" implies a host of complex relationships between the narrative objects "Tom" and "Katie". If the "love" relationship were to be modelled visually, knowledge of visual model constructs such as relationship arrows and object icons would be required to arrive at a correct interpretation. If the "love" relationship is modelled by the word "love" itself, its interpretation is implicit, as we think, learn, and reason about such emotional concepts through words.

The overhead of relating narrative concepts using visual and programmatic modelling tools is identified by Robertson *et al* [36] in a study that applied such tools to encourage the creation of narrative. The study indicated that children spent more time getting to grips with the Neverwinter Nights^{TM}level editor used to construct their stories, than telling the stories themselves. This editor uses graphical modelling tools and a scripting engine to construct interactive narrative spaces. A word-driven system that leverages the implicit understanding of natural language words would alleviate this problem while still providing the encouraging immediate visual feedback from generated interactive narrative spaces.

2.2.2 Concise narrative representation

A second distinct advantage of generating interactive narrative spaces from natural language sentences is that words are concise models for narrative concepts. A single sentence can represent a large volume of visual narrative information. Consider the visualisation of the word "forest". When visualised this single word designates thousands of images of trees, shrubs, and animals compiled into a single narrative concept. A system that generates these images from this single word would be able to encapsulate vast amounts of narrative depictions within a single–word model. If an adjective such as "dark" were also modelled, the attributes that it would encapsulate could be applied to the narrative depictions encapsulated within the noun "forest" by the simple natural language phrase "dark forest". Finally, if a verb such as "burn" were modelled the actions and occurrences it encapsulates could easily be applied to the narrative depictions encapsulated in the word "forest" by stating the simple sentence "The dark forest burns.". The complex visual narrative depiction of a burning forest can therefore be quite simply stated in a short sentence.

2.2.3 Reinterpretation

A third advantage of a system that generates interactive narrative depictions from natural language sentences stems from the fact that natural language as narrative construction medium affords us the luxury of reinterpretation. If the characters "Tom" and "Katie" are directly depicted using fixed images of a boy and a girl, our interpretation of the narrative surrounding them is narrowed by those depictions. If a paradigm of modelling narrative using natural language sentences is followed, the narrative and its representation may be decoupled. As an example, the natural language sentence "Tom loves Katie" can be visually represented as "Tom" and "Katie" being badgers instead of human beings. The alternative visual representation can be used to switch the theme of the narrative from teenage romance to children's fable. The narrative structure behind the events that unfold is preserved, but interpreted in an entirely new way.

2.2.4 Accessible

Natural language provides an easy way for a large majority of human beings to convey narrative. An interactive narrative creation system that generates interactive narrative depictions from natural language sentences, allows a wider audience to participate in the creation of interactive narrative. Since existing graphical models, sounds, and AI behaviours can be re-used in such a generation system, a user would not require in depth knowledge of how the depictions themselves are constructed. The construction of narrative depictions such as graphical models or software agents is typically a technically complex task. Using a system that generates interactive narrative from existing narrative depictions and natural language sentences, all that would be required for the creation of interactive narrative, is the ability to convey narrative in natural language sentences. In this way a widely accessible narrative creation system is achieved.

2.2.5 Summary

This section discussed the advantages of a word-based interactive narrative generation paradigm. Language was shown to be a natural fit in modelling narrative concepts, and words were shown to be concise models of narrative concepts. Narrative structure can be decoupled from narrative representation in a word-based interactive narrative generation paradigm which allows for narrative to be reinterpreted. Interactive narrative generation driven by natural language and the re–use of existing narrative depictions was shown to be an accessible method of interactive narrative creation. This section discussed the contrasts between a word-based narrative generation paradigm and one that seeks to construct interactive narrative directly from narrative depictions and abstract visual and programmatic models. The section that follows indicates how interactive narrative generation and direct interactive narrative creation may be combined to build full fledged interactive narrative systems.

2.3 Integrating narrative space generation with existing interactive narrative systems

The line separating a word-based paradigm for the generation of interactive narrative spaces, being idealistic and fanciful, and a practical reality, is an ability to model natural languages successfully in computable terms. Natural language is filled with vagueness and ambiguity, and relies on the collective experience of human beings for its correct interpretation. The construction of computable word models is therefore a difficult task. The initial objective of a word-based paradigm for interactive narrative creation is therefore to complement rather than replace other interactive narrative systems of the kind specified in section 2.1.

As an example of complementing interactive narrative systems with an interactive narrative generation system, consider the interactive drama game *Facade* [62]. Natural language character descriptions could be used to generate AI scripts for agent behaviours in as much detail as possible. The AI scripts could then be fleshed out to create a believable interactive fiction experience. Doing so would enable creators of interactive dramas such as *Facade* to create a greater variety of characters and scenarios more efficiently.

A top–down approach to narrative generation [76] [85] requires a large amount of narrative representation content to be created and recreated to account for the different story permutations that could result from user interaction. The generation of content from natural language sub-plot descriptions would allow users of a system such as Mimesis [126] to produce rapidly narrative representations corresponding to the plot permutations. For emergent interactive narrative scenarios such as *FearNot!* [6], narrative language models could efficiently describe subsequently generated agents and scenarios. The generated content could be realised and combined in a variety of ways, leading to an explosion of narrative possibility.

2.4 Summary

This chapter presented related work in the field of interactive narrative, and presented applications of interactive narrative in other fields of interest. It also described related work that applies fuzzy set theory principles to interactive narrative spaces. A word-centred interactive narrative generation approach was contrasted with approaches that either directly construct interactive narrative space by building narrative depictions or that model interactive narrative spaces in some other way. The high–level advantages of a system that allows for the generation of an interactive narrative space from natural language words were discussed. Finally, the integration potential of interactive narrative generation and existing interactive narrative systems was discussed.

Chapter 3

Fuzzy set theory, computational verb theory, and constraint satisfaction problem definitions

This chapter states the fuzzy set theory, computational verb theory, and constraint satisfaction problem definitions referenced in this work. The definitions are provided for self-containment and as a reference for the theoretical principles, architecture, and results chapters that follow. Those definitions not referenced elsewhere within this work are also provided here to allow for a sequential line of reasoning that leads to the definitions used throughout the work. Readers familiar with the definitions in the fields of fuzzy set theory, computational verb theory (CVT), and constraint satisfaction problems (CSPs) may skip the relevant subsections. References to the definitions in this chapter are provided in the chapters that follow, where applicable.

Section 3.1 contains fuzzy set theory definitions by defining crisp variables, extending the definitions to fuzzy variables and defining linguistic variables. Section 3.2 contains definitions related to computational verb theory. Section 3.3 contains the definitions for a constraint satisfaction problem (CSP) and a fuzzy constraint satisfaction problem (FCSP).

3.1 Fuzzy set theory definitions

This section contains the fuzzy set theory definitions utilised within this work. Crisp variable definitions are provided in section 3.1.1. Fuzzy sets are defined, and crisp variable definitions are extended in terms of fuzzy set theory to fuzzy variable definitions in section 3.1.2. Linguistic hedges are defined as modifying functions of the membership functions related to fuzzy variables in section 3.1.2. Section 3.1.2 also defines linguistic variables in terms of fuzzy variables.

3.1.1 Variable definitions

This subsection contains definitions related to crisp variables and to the relationhips between crisp variables. Variables, restrictions, projections, and interactivity between crisp variables are defined. The definitions are as follows:

Definition 3.1 (Crisp variable): Zadeh [128] characterises a variable as a triple $(X, U, R(X; u))$ in which X is the name of the variable, U is the universe of discourse (a finite or infinite set), u is a generic name for elements of U, and $R(X; u)$ is a subset of U which represents a restriction on the values of u imposed by X. Associated with every variable is an assignment equation,

$$x = u : R(X), \tag{3.1}$$

which can also be given as

$$x = u; u \in R(X). \tag{3.2}$$

This represents the assignment of a value u to x under the restriction $R(X)$.

More generally, $X = (X_1, X_2, \ldots, X_n)$ denotes an n-tuple of n variables $X_1, X_2, \ldots, X_n$ with universes of discourse $U_1, U_2, \ldots, U_n$. $(X_1, X_2, \ldots, X_n)$ is called an n-ary composite or joint variable. The universe of discourse for X is the Cartesian product

$$U_1 \times U_2 \times \ldots \times U_n. \tag{3.3}$$

The restriction, $R(X_1, X_2, \ldots, X_n)$, is an n-ary relation in $U_1, U_2, \ldots U_n$. This restriction

may be defined by a characteristic function, $\mu_R : U_1 \times U_2 \times \ldots \times U_n \rightarrow \{0, 1\}$, where

$$\mu(u_1, ..., u_n) = \begin{cases} 1 & \text{if}(u_1, ..., u_n) \in R(X_1, ..., X_n) \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

and u_i is the generic name for elements of $U_1, ..., U_n$. The associated n-ary assignment takes the form

$$x_1, \ldots, x_n = (u_1, \ldots, u_n) : R(X_1), \ldots, R(X_n), \tag{3.5}$$

which is interpreted as

$$x_i = u_i; i = 1, \ldots, n. \tag{3.6}$$

$u_1, \ldots, u_n \in R(X_1, \ldots, X_n)$ is a restriction. x_i for $i = 1, 2, \ldots, n$ denotes generic names for values of X_i.

Definition 3.2 (Marginal restriction): Zadeh [128] defines the restriction, $R(X_1, \ldots, X_n)$, imposed by $(X_1, \ldots, X_n)$, as inducing marginal restrictions, $R(X_{i_1}, \ldots, X_{i_k})$, imposed by composite variables of the form $(X_{i_1}, \ldots, X_{i_k})$, where the index sequence, $q = (i_1, \ldots, i_k)$, is a subsequence of index sequence, $(1, \ldots, n)$. In effect, $R(X_{i_1}, \ldots, X_{i_k})$ is the smallest restriction imposed by $(X_{i_1}, \ldots, X_{i_k})$ that satisfies the implication

$$(u_1, \ldots, u_n) \in R(X_1, \ldots, X_n) \Rightarrow (u_{i_1}, \ldots, u_{i_k}) \in R(X_{i_1}, \ldots, X_{i_k}). \tag{3.7}$$

Therefore a given k-tuple, $u_{(q)} \stackrel{\Delta}{=} (u_{i_1}, \ldots, u_{i_k})$, is an element of $R(X_{i_1}, \ldots, X_{i_k})$ if and only if there exists an n-tuple, $u \stackrel{\Delta}{=} (u_1, \ldots, u_n) \in R(X_1, \ldots, X_n)$, whose i_1-th, $\ldots$, i_k-th components are equal to $u_{i_1}, \ldots, u_{i_k}$ respectively. Expressed in terms of the characteristic functions of $R(X_1, \ldots, X_n)$ and $R(X_{i_1}, \ldots, X_{i_k})$, the restriction is given by:

$$\mu_{R(X_{i_1},...,X_{i_k})}(u_{i_1}, \ldots, u_{i_k}) = \bigvee_{u_{(q')}} \mu_{R(X_1,...,X_n)}(u_1, \ldots, u_n). \tag{3.8}$$

q' is the complement of index sequence (the indices that are not in the sequence), $q = (i_1, \ldots, i_k)$, relative to $(1, \ldots, n)$. $u_{(q')}$ is the complement of k-tuple, $u_{(q)} \stackrel{\Delta}{=} (u_{i_1}, \ldots, u_{i_k})$, relative to the n-tuple, $u \stackrel{\Delta}{=} (u_1, \ldots, u_n)$. $X_{(q)} \stackrel{\Delta}{=} (X_{i_1}, \ldots, X_{i_k})$. $\bigvee_{u_{(q')}}$ denotes the supremum of its operand over the u's that are in $u_{(q')}$.

Definition 3.3 (Projection of a variable): Zadeh [128] defines the characteristic function of the projection of $R(X_1, \ldots, X_n)$ on $U_{i_1} \times \ldots \times U_{i_k}$ as

$$R(X_{i_1}, \ldots, X_{i_k}) = ProjR(X_1, \ldots, X_n) \text{ on } U_{i_1} \times \ldots \times U_{i_k}, \tag{3.9}$$

where $i_1, \ldots, i_k$ denotes a subsequence of $1, 2, \ldots, n$. More simply,

$$R(X_{i_1}, \ldots, X_{i_k}) = P_q R(X_1, \ldots, X_n), \tag{3.10}$$

where P_q denotes the operation of projection on $U_{i_1}, \ldots, U_{i_k}$ with $q = (i_1, \ldots, i_k)$.

Definition 3.4 (Conditioned restriction): For the restriction, $R(X_1, \ldots, X_n)$, viewed as a relation in $U_1 \times \ldots \times U_n$, let $q' = (j_1, \ldots, j_m)$ denote the index sequence complementary to $q = (i_1, \ldots, i_k)$ and let $R(X_{i_1}, \ldots, X_{i_k} | u_{j_1}, \ldots, u_{j_m})$ denote a restriction in $U_{i_1} \times \ldots \times U_{i_k}$ which is conditioned on $u_{j_1}, \ldots, u_{j_m}$. The characteristic function of this conditioned restriction is defined by

$$\mu_{R(X_{i_1},\ldots,X_{i_k}|u_{j_1},\ldots,u_{j_m})}(u_{i_1}, \ldots, u_{i_k}) = \mu_{R(X_1,\ldots,X_n)}(u_1, \ldots, u_n), \tag{3.11}$$

on the understanding that the arguments, $u_{j_1}, \ldots, u_{j_m}$, are treated as parameters when specified on the right-hand side of equation (3.11). If the definition above is taken into consideration, the projection of the restriction, $R(X_1, \ldots, X_n)$, onto $U_{i_1} \times \ldots \times U_{i_k}$ can be expressed as

$$P_q R(X_1, \ldots, X_n) = \bigcup_{u(q')} R(X_{i_1}, \ldots, X_{i_k} | u_{j_1}, \ldots, u_{j_m}), \tag{3.12}$$

where $\bigcup_{u(q')}$ denotes the union of the family of restrictions $R(X_{i_1}, \ldots, X_{i_k} | u_{j_1}, \ldots, u_{j_m})$ parameterised by $u_{(q')} \triangleq (u_{j_1}, \ldots, u_{j_m})$.

Definition 3.5 (Non-interactive and interactive variables): Variables $X_1, \ldots, X_n$ are non-interactive variables under $R(X_1, \ldots, X_n)$ if and only if $R(X_1, \ldots, X_n)$ is separable, that is,

$$R(X_1, \ldots, X_n) = R(X_1) \times \ldots \times R(X_n), \tag{3.13}$$

where for $i = 1, \ldots, n$

$$\begin{aligned} R(X_i) &= Proj R(X_1, \ldots, X_n) \text{ on } U_i \\ &= \cup_{u_{(q')}} R(X_i | u_{(q')}), \end{aligned} \tag{3.14}$$

with $u_{(q)} \triangleq u_i$ and $u_{(q')} \triangleq$ complement of u_i in $(u_1, \ldots, u_n)$ [128].

If the restriction, $R(X_1, \ldots, X_n)$, over the variables $X_1, \ldots, X_n$ is not separable, then $X_1, \ldots, X_n$ are interactive variables. For interactive variables $X_1, \ldots, X_n$ a sequence of n unary assignment equations (refer to equation (3.6)) takes the form

$$\begin{aligned} x_1 &= u_1 : R(X_1), \\ x_2 &= u_2 : R(X_2 | u_1) \\ &\vdots \\ x_n &= u_n : R(X_n | u_1, \ldots, u_{n-1}), \end{aligned} \tag{3.15}$$

where $u_i : R(X_i | u_1, \ldots, u_{i-1})$ denotes the conditioned restriction induced on u_i conditioned on $u_1, \ldots, u_{i-1}$. The characteristic function of the conditioned restriction is given by

$$\mu_{R(X_i | u_1, \ldots, u_{i-1})}(u_i) = \mu_{R(X_1, \ldots, X_i)}(u_1, \ldots, u_i), \tag{3.16}$$

on the understanding that the arguments, $u_1, \ldots, u_{i-1}$, are treated as parameters when specified on the right-hand side of equation (3.16).

3.1.2 Fuzzy variable definitions

This subsection extends the crisp variable definitions of the preceding section in terms of fuzzy set theory. Definitions for a fuzzy set, fuzzy variable, linguistic hedge, and linguistic variable are provided.

Definition 3.6 (Fuzzy set): A fuzzy set A in a space of points $\mathbf{X}$, where $\mathbf{x}$ is a generic member of $\mathbf{X}$, is characterised by a membership or characteristic function $\mu_A(\mathbf{x})$ that associates every point in $\mathbf{X}$ with a real-valued number on the interval $[0, 1]$ [127]. The value of $\mu_A(\mathbf{x})$ represents the grade of membership of $\mathbf{x}$ in A.

Definition 3.7 (Fuzzy variable): A fuzzy variable is characterised by a triple $(X, U, R(X; u))$, where X is the name of the variable, U is the universe of discourse (a finite or infinite set) of X, u is a generic name for the elements of U, $R(X; u)$ is a fuzzy subset of U which represents a fuzzy restriction on the values of u by X, and u denotes the non-restricted non-fuzzy base variable for X [129]. Associated with the fuzzy variable is an assignment equation,

$$x = u : R(x), \tag{3.17}$$

which represents an assignment of x to a value u subjected to the restriction $R(X)$. The degree to which the restriction is satisfied is called the compatibility of u with $R(X)$, denoted by $c(u)$, and is defined as

$$c(u) \triangleq \mu_{R(X)}(u); u \in U, \tag{3.18}$$

where $\mu_{R(X)}$ is the grade of membership of u in the restriction $R(X)$.

More generally, if $X_1, \ldots, X_n$ are variables in $U_1, \ldots, U_n$ respectively, then $X \triangleq (X_1, \ldots, X_n)$ is an n-ary composite or joint variable in $U_1 \times \ldots \times U_n$. The corresponding n-ary assignment relation is

$$(x_1, \ldots, x_n) = (u_1, \ldots, u_n) : R(X_1, \ldots, X_n), \tag{3.19}$$

where $x_i, i = 1, \ldots, n$ is a generic name for the values of X_i, u_i is a generic name for the elements of U_i and $R(X) \triangleq R(X_1, \ldots, X_n)$ is an n-ary fuzzy relation in U which represents the restriction imposed by $X \triangleq (X_1, \ldots, X_n)$. The compatibility of $(u_1, \ldots, u_n)$ with $R(X_1, \ldots, X_n)$ is defined by

$$c(u_1, \ldots, u_n) = \mu_{R(X)}(u_1, \ldots, u_n), \tag{3.20}$$

where $\mu_{R(X)}$ is the membership function of the restriction on $u \triangleq (u_1, \ldots, u_n)$.

Definition 3.8 (Linguistic hedge): A linguistic hedge is defined as a function $f(\cdot)$ that acts upon a fuzzy set A to modify its associated membership function according to the definition of f [130] [85]. A concentration is given by

$$CON(A) \triangleq g(A) = A^{\eta}; \eta \geq 2 \tag{3.21}$$

whereas a dilation is given by

$$DIL(A) \triangleq g(A) = A^{\eta}; \eta \in (0,1). \tag{3.22}$$

Both concentrations and dilations are examples of linguistic hedges.

Definition 3.9 (Linguistic variable): A linguistic variable is characterised as a quintuple ($\mathcal{L}$, $\mathcal{T}(\mathcal{L})$, U, G, M). $\mathcal{L}$ is the name of the variable. $\mathcal{T}(\mathcal{L})$ denotes the term set of $\mathcal{L}$, the set of names of linguistic values for $\mathcal{L}$. Each linguistic value is a fuzzy variable denoted generically by X and ranges over a universe of discourse U associated with a base variable u. G is a syntactic rule (usually in the form of a grammar) for generating the names, X, of values of $\mathcal{L}$. M is a semantic rule for associating with each X its meaning, $M(X)$, which is a fuzzy subset of U [129].

The meaning, $M(X)$, of a term, X, is defined to be the restriction, $R(X)$, on the base variable, u, which is imposed by the fuzzy variable named X. Thus

$$M(X) \triangleq R(X), \tag{3.23}$$

on the understanding that $R(X)$ and thus $M(X)$ may be viewed as a fuzzy subset of U carrying the name X.

Assignment in the case of a linguistic variable (X) assumes the form

$$\begin{aligned} X &= \text{Term in } \mathscr{T}(\mathcal{L}), \\ &= \text{name generated by G.} \end{aligned} \tag{3.24}$$

Equation (3.24) implies that the meaning assigned to X is

$$M(X) = R(\text{Term in } \mathscr{T}(\mathcal{L})). \tag{3.25}$$

3.2 Computational verb theory definitions

This section provides an introduction to computational verb theory (CVT) and presents the CVT definitions referenced within this work. A short history of the development of CVT is presented in section 3.2.1. Computational verb definitions are provided in

section 3.2.2. Section 3.2.3 defines a computational verb BE-transformation. Section 3.2.4 provides definitions related to computational verb sentences. Section 3.2.5 provides definitions related to computational verb sets.

3.2.1 The development of computational verb theory

Computational verbs were first introduced as a means of solving engineering and dynamics problems by modelling them in terms of natural language verbs [102] [104] [107]. Computational verb concepts were further developed [101] [102] [106] [112] [107] into a new design paradigm called "computational verb theory".

Computational verb-based logical operations between verbs were defined by Yang in [103]. Computational verb logics were applied to form computational verb-based reasoning systems in [111] [104]. Verb sets and verb numbers which are computational verb-based generalisations of their respective mathematical concepts were introduced in [105] [110]. Decision trees were extended to computational verb decision trees in [123]. The link between fuzzy membership functions and computational verb collapses was studied in [122].

The relationship between computational verbs and natural language was studied in [114] [111]. The relationship between natural language adverbs and verbs as they relate to computational verbs were defined in [100].

A theory for the unification of fuzzy and computational verb theories as the basis for machine cognition was presented and studied in [117] [111]. The issues of implementing cognition and simulating human emotion using CVT were presented in [120].

Zhang *et al* [131] have described an alternative way of computing verb similarity using neural networks that more closely follows human thought processes. Yang *et al* [125] have provided a measure of ambiguity between computational verbs by using Shannon entropy [84]. Other work has been done in the field of CVT, but only the work applicable in the context of this thesis is listed here.

3.2.2 Computational verb definitions

This section defines evolving systems, trajectories of evolving systems, and equilibrium points. The definition of an evolving system is applied in the definition of a computational verb. A simplified definition of a computational verb is supplied. Basic computational verbs are categorised as static verbs, focus verbs, centre verbs, and node verbs.

Definition 3.10 (Dynamic system): An nth-order continuous-time dynamic system is represented by a state equation [106]:

$$\dot{x} = f(x); x(t_0) = x_0, \tag{3.26}$$

where

$$\dot{x} \triangleq \frac{\partial x}{\partial t}. \tag{3.27}$$

$x(t) \in \mathbb{R}^n$ is the state of the dynamic system at a time, t. $f : \mathbb{R}^n \mapsto \mathbb{R}^n$ is a vector field that associates every point in $\mathbb{R}^n$ with a tangent vector that describes the change in the dynamic system from that point. The solution to equation (3.26) is written as $\phi_t(x_0)$.

Definition 3.11 (Trajectory): The family of mappings $\phi_t : \mathbb{R}^n \mapsto \mathbb{R}^n$ is called a "flow" [106]. A flow has the properties:

$$\phi t_1 + t_2 = \phi_{t_1} \circ \phi_{t_2}; \phi_0(x) = x. \tag{3.28}$$

The set of points $\{\phi_t(x_0) | -\infty < t < +\infty\}$ is called "the trajectory through x_0". The trajectory through x_0 is calculated by setting x_0 as the initial condition and obtaining a solution to the dynamic system $\phi_t(x_0)$. The solution to the dynamic system obtained, $\phi_t(x_0)$, is evaluated for the time values specified for the trajectory to obtain the trajectory values.

Definition 3.12 (Equilibrium point): An equilibrium point (critical point) of an nth-order continuous-time dynamic system (refer to definition 3.10) is a constant solution of the equation (3.26). The constant solution of an equilibrium point is given by

$$x^* = \phi_t(x^*) \tag{3.29}$$

for all time, t.

Definition 3.13 (Computational verb): Yang [111] defines a computational verb, $\mathcal{V}$, in terms of three dynamic systems (refer to definition 3.10) called the "evolving system", the "inner system", and the "outer system". The inner system is a dynamic system that models the inner complexities of a human brain or another complex system. An example of such a complex system is a thought in a human brain. The outer system is a dynamic system that models the observation of a complex system. An example is the outer system modelling the thought as understood by another person when the first person communicates the thought to that other person using natural language words. The evolving system is a dynamic system that models a complex system as a whole. The evolving system encapsulates both the workings of the inner system and the observation of the inner system in terms of the outer system.

The full definition of a computational verb, $\mathcal{V}$, is given in terms of the evolving, inner, and outer systems as follows:

1. The evolving system, $\mathscr{E}_{\mathcal{V}}$, summarises the computational verb, $\mathcal{V}$, as a whole, giving

$$\mathscr{E}_{\mathcal{V}} = [\mathscr{T}, \mathscr{X}_s, \mathscr{X}_p, \mathcal{I}_{\mathcal{V}}, \mathcal{F}_{\mathcal{V}}]. \tag{3.30}$$

   $\mathscr{T} \subset \mathbb{R}$ is the time interval associated with the system. $\mathscr{X}_s \subset \mathbb{R}^3$ represents the physical space for the system as a whole. $\mathscr{X}_p \subset \mathbb{R}^k$ is the space of reasoning for the system defined in terms of real-valued functions in the space $\mathbb{R}^k$. $\mathcal{I}_{\mathcal{V}}$ is the inner system. $\mathcal{F}_{\mathcal{V}}$ is the outer system.

2. The inner system, $\mathcal{I}_{\mathcal{V}}$, is given as

$$\mathcal{I}_{\mathcal{V}} : \mathscr{T} \times \mathscr{X}_s \times \mathscr{X}_p \mapsto \mathscr{T} \times \mathscr{X}_s \times \mathscr{X}_p \tag{3.31}$$

   and is used to model the inner workings of a complex system.

3. The outer system, $\mathcal{F}_{\mathcal{V}}$, is given as

$$\mathcal{F}_{\mathcal{V}} : \mathscr{T} \times \mathscr{X}_s \times \mathscr{X}_p \mapsto \mathscr{T}' \times \mathscr{X}_s' \times \mathscr{X}_p' \tag{3.32}$$

   and is used to model the observation of the complex (inner) system. $\mathscr{T}' \subset \mathbb{R}$ is the perception of time in the system. $\mathscr{X}_s' \subset \mathbb{R}^3$ is the perception of the physical space the system is located in. $\mathscr{X}_p' \subset \mathbb{R}^l$ is the space of reasoning for the observed system in terms of real-valued functions in the space, $\mathbb{R}^l$.

Definition 3.14 (Simplified computational verb definition): A simplified definition of a computational verb, $\mathcal{V}$, used in engineering applications is given in terms of an evolving function [120] [122] [123] [124] [125]

$$\mathcal{E}_{\mathcal{V}} : T \times \Omega \rightarrow \Omega, \tag{3.33}$$

where $T \subseteq \mathbb{R}$ and $\Omega \subseteq \mathbb{R}^n$ are the time and universe of discourse for the evolving function respectively. A computational verb, $\mathcal{V}$, defined within a discrete space is given by the evolving function

$$\mathcal{E}_{\mathcal{V}} : T \times \Omega \rightarrow \Omega, \tag{3.34}$$

where $T \subseteq \mathbb{Z}$ are the discrete time values for the evolving function $\mathcal{E}_{\mathcal{V}}$.

This thesis uses this simplified definition of a computational verb. The evolving function, $\mathcal{E}_{\mathcal{V}}$, represents the state of the dynamic systems modelled within a computational verb (refer to definition 3.13) at a time, t. This thesis models natural language verbs in terms of basic computational verbs. Basic computational verbs represent the trajectories (refer to definition 3.11) between equilibrium points or around equilibrium points (refer to definition 3.12). The basic computational verb types presented in this thesis are as follows:

Definition 3.15 (Static computational verb): A static (computational) verb has no dynamics [106]. Every initial condition is an equilibrium point (refer to definition 3.12) in the dynamic system(s) associated with the computational verb. An example of this type of basic verb is the word "be".

Definition 3.16 (Node computational verb): A node (computational) verb is generated in the vicinity of a node (equilibrium point) [106]. A node verb represents a motion either towards or away from an equilibrium point. As an example, consider the verb phrase "jumps onto", which represents the motion of an object towards an equilibrium point. The equilibrium point for the example represents the top of an object where the universe of discourse for the dynamic system is 3D space.

Definition 3.17 (Focus computational verb): A focus (computational) verb is generated in the vicinity of a focus (equilibrium point) [106]. A focus verb represents

either an indirect motion away from an equilibrium point or an indirect motion towards an equilibrium point. An example of a focus verb is the verb "drawn" in the sentence "The moth is drawn to the flame.", if the verb "drawn" is modelled in terms of the indirect motion of the moth towards the position of the flame.

Definition 3.18 (Centre computational verb): A centre (computational) verb is generated in the vicinity of a centre (equilibrium point) [106]. A centre verb represents a motion that keeps an object in the vicinity of an equilibrium point. The motion does not push the object away from the equilibrium point or pull the object towards the equilibrium point. An example of a centre verb is the verb "circle".

3.2.3 Computational verb BE-transformation

This subsection provides the definition of a BE-transformation.

Definition 3.19 (BE-transformation): A verb statement, $\mathcal{S}$, is given by:

$$\mathcal{S} : [\mathcal{N}_s, \mathscr{V}, \mathcal{N}_o], \tag{3.35}$$

where $\mathcal{S}$, $\mathcal{N}_s$, $\mathscr{V}$, and $\mathcal{N}_o$ denote the verb statement, the optional subject noun, the verb, and the optional object noun respectively.

A BE-transformation is given by

$$[\mathcal{N}_s, \mathscr{V}, \mathcal{N}_o] \text{"BE word"} \tag{3.36}$$

with grammatical and contextual adjustment [107]. The "word" can be any noun, adjective or equivalence. The BE-transformation transforms the verb statement $\mathcal{S}$ into a BE-statement $\mathcal{S}'$ given by

$$\mathcal{S}' : [\mathcal{N}_s, \mathscr{V}, \mathcal{N}_o] \text{ BE "word"}. \tag{3.37}$$

The "word" accounts for the differing contexts and interpretations that could possibly be associated with a verb statement.

Computational verbs are interpreted in different contexts by calculating the degree of "BE word" for the computational verb, $\mathcal{V}$ [107]. The degree of "BE word" is denoted

by $\tau_{word}(\mathcal{V})$; it is calculated by applying the transformation function, Φ^I_{word}, to the inner system of a computational verb, $\mathcal{V}$ (refer to equation (3.31)), and by applying the transformation function, Φ^O_{word}, to the outer system of a computational verb, $\mathcal{V}$ (refer to equation (3.32)). The result of the transformation of an evolving system, $\mathscr{E}_{\mathcal{V}}$, by the transformation functions, Φ^I_{word} and Φ^O_{word}, is a function that calculates the degree of "BE word".

If the inner system of a computational verb, $\mathcal{V}$, is known, the transformation, Φ^I_{word}, is applied to the inner system of the computational verb, $\mathcal{V}$, and the degree of "BE word", $\tau_{word}(\mathcal{V})$, is given by

$$\tau_{word}(\mathcal{V}) = \Phi^I_{word} : \mathscr{T} \times \mathscr{X}_s \times \mathscr{X}_p \mapsto \mathbb{R}^m. \tag{3.38}$$

$\mathscr{T} \subseteq \mathbb{R}$, $\mathscr{X}_s \subseteq \mathbb{R}^3$, and $\mathscr{X}_p \subseteq \mathbb{R}^k$ are defined as for equation (3.31).

If the outer system of a computational verb, $\mathcal{V}$, is known, the transformation, Φ^O_{word}, is applied to the outer system of the computational verb, $\mathcal{V}$, and the degree of "BE word", $\tau_{word}(\mathcal{V})$, is given by

$$\tau_{word}(\mathcal{V}) = \Phi^O_{word} : \mathscr{T}' \times \mathscr{X}'_s \times \mathscr{X}'_p \mapsto \mathbb{R}^m. \tag{3.39}$$

$\mathscr{T}' \subseteq \mathbb{R}$, $\mathscr{X}'_s \subseteq \mathbb{R}^3$, and $\mathscr{X}'_p \subseteq \mathbb{R}^l$ are defined as for equation (3.32). $\mathbb{R}^m$ is the universe of discourse for the function that calculates the degree of "BE word".

If the sentence, "The man ages.", is interpreted in the context "old", the degree of "BE old" is calculated by applying the transformations, Φ^I_{old} and Φ^O_{old}, to the evolving system of the computational verb "ages". If the evolving function, $\mathcal{E}_{ages}$, that represents the evolving system of the computational verb "ages" (refer to definition 3.14) is given by

$$x = \mathcal{E}_{\mathcal{V}}(t) = t; t \in (0, 80], \tag{3.40}$$

a BE-transformation for the phrase "BE old" is given by

$$\mu(x) = \frac{x}{80}; x \in (0, 80]. \tag{3.41}$$

$\mu(x)$ is the membership function of a fuzzy set "old" with a base variable, x, in years. The time, t, in years of the evolving system is represented as a real-valued number on the interval $(0, 80]$. The evolving function, $\mathcal{E}_{ages}$, evaluates to a larger value as the time value,

t, that is passed to the evolving function, $\mathcal{E}_{ages}$, increases (refer to equation (3.40)). The value of the evolving function, $\mathcal{E}_{ages}$, is stored in the base variable, x. If the value of x increases to 80, the degree of "BE old", as determined by the membership function, $\mu(x)$, increases to 1.0. The BE-transformation for the phrase "BE old" therefore corresponds to the perception of the man growing older as his age increases to 80. The man growing old is modelled in turn by the computational verb "ages" and the corresponding evolving function, $\mathcal{E}_{ages}$.

BE-transformations allow different computational verbs expressed in different contexts of interpretation to be combined into a single logical system for combined logical operations and reasoning. The single logical system used is typically a fuzzy logic system.

3.2.4 Computational verb sentence definitions

This subsection defines both simple atomic verb sentences and molecular verb sentences.

Definition 3.20 (Atomic verb sentence): An atomic verb sentence consists of an original process-symbol (the verb word that corresponds to a computational verb) and an optional subject and/or object (noun) [107].

Definition 3.21 (Molecular verb sentence): A molecular or composite verb sentence consists of atomic verb sentences, connectives, and adverbials, written in different sentences and modes [107].

3.2.5 Verb set definitions

This section provides definitions related to verb sets by defining verb elements, computational verb collapses, and computational verb sets.

Definition 3.22 (Computational verb element): A (computational) verb element is denoted by the ordered pair, ($\mathscr{C}$, $\mathscr{S}$), and consists of a verb-statement, $\mathcal{S}$, interpreted in a context, $\mathscr{C}$. The "birth time" – which is the start of time interval associated with verb–element, ($\mathscr{C}$,$\mathcal{S}$), – is denoted by T_0. The "death time" – which is the end of the

time interval associated with the verb–element, $(\mathscr{C},\mathcal{S})$, – is denoted by $T_0 + T_C$. $T_{\mathscr{C}}$ is the size of the time interval associated with the verb–element, $(\mathscr{C},\mathcal{S})$ [107].

Definition 3.23 (Collapse of a computational verb statement): If a (computational) verb statement has the structure "verb + statement", a collapse of the computational verb statement, $\mathcal{S}$, is given by "BE + statement" with grammatical and contextual adjustments [107]. The computational verb collapse of verb statement, $\mathcal{S}$, is denoted by $\mathcal{C}(\mathcal{S})$.

Computational verb collapses map verb sentences to their end states. BE-transformations transform verb sentences (statements) into static structures such as fuzzy sets for unified logic operations between different verb sentences (statements) in different contexts of interpretation. The end state of a verb sentence (statement) is a language statement that represents the end state of the action named by the verb. The sample sentence "He walks to school." has an end state signified by the statement "at school". An example of a computational verb collapse for the verb sentence (statement) "He walks to school." in the context of interpretation "location" is given by the statement "BE at school". The computational verb collapse, "BE at school", can be expressed in terms of a crisp value, a crisp set, or a fuzzy set.

Definition 3.24 (Computational verb set): A computational verb set, $\mathscr{S}_{\mathcal{S}} \subset \mathcal{U}$, is deduced from a computational verb sentence (statement), $\mathcal{S}$, and a crisp or a fuzzy set, $\mathscr{S}$. The set, $\mathscr{S}$, is a set of collapses for the computational verb sentence (statement), $\mathcal{S}$. The verb set, $\mathscr{S}_{\mathcal{S}}$, is given by

$$\mathscr{S}_{\mathcal{S}} = \{(\mathscr{C},\mathcal{S})|\{\mathcal{C}(\mathcal{F}_{\mathcal{S}}(t,\mathscr{C}))\} = \mathscr{S}; (\mathscr{C},\mathcal{S}) \in \mathcal{U}; t \in [0,T_{\mathcal{C}}]\}. \tag{3.42}$$

$T_{\mathscr{C}}$ is the lifetime of the verb sentence $\mathcal{S}$ in context $\mathscr{C}$. $\mathcal{C}(\mathcal{F}_{\mathcal{S}}(t,\mathscr{C}))$ is the collapse of the verb element, $(\mathscr{C},\mathcal{S})$ (refer to definitions 3.22 and 3.23). $\mathcal{C}(\cdot)$ is a mapping function that maps a verb sentence (statement) to a crisp value, crisp set, or a fuzzy set that represents the collapse of the verb sentence (statement). $\{\mathcal{C}(\mathcal{F}_{\mathcal{S}}(t,\mathscr{C}))\}$ is the set of collapses of the outer observable system of $\mathcal{S}$ in context set, $\{\mathscr{C}\}$ [107]. The verb collapses are performed in terms of the outer system, $\mathcal{F}_{\mathcal{S}}$, of the verb sentence (statement), $\mathcal{S}$. The outer system is used for the verb collapse because a verb collapse corresponds to an end state, and

the system that is collapsed must be observable to determine the end state (refer to definition 3.13 and equation (3.32)).

3.3 Constraint satisfaction problem definitions

This section presents the definition of CSPs. After fuzzy constraints have been introduced, CSPs are relaxed in terms of fuzzy constraints to define fuzzy constraint satisfaction problems (FCSPs).

Definition 3.25 (Constraint satisfaction problem): A CSP [91] is defined as a triple (Z,D,C), where Z is a finite set of variables $x_1, \ldots, x_n$. D is a function that maps every variable in Z to a finite set of objects of arbitrary type, $D : Z \mapsto$ finite set of objects (any type). D_{x_i} denotes the set of objects mapped by D for variable x_i. The objects then form the values of x_i, and D_{x_i} represents the domain of x_i. C is a finite (possibly empty) set of constraints on an arbitrary subset of variables in Z. In other words, Z is a set of compound labels, where a label is the assignment of a value to a variable and a compound label is the simultaneous assignment of n values to n variables. The triple and therefore the CSP are denoted by csp(P), which is read as "P is a CSP".

Definition 3.26 (Fuzzy constraint satisfaction problem): An FCSP is defined as in definition 3.25, but the constraints are defined as fuzzy [79]. A fuzzy constraint is a mapping from a domain, $D = D_1 \times \ldots \times D_k$ (corresponding to the domains of the variables referred to by the constraint), to a real number on the interval [0, 1]. If C_i is a fuzzy constraint, the number $c(v_1, \ldots, v_k)$ denotes how well the variable value tuple, $(v_1, \ldots, v_k)$, satisfies the constraint, C_i. The number $c(v_1, \ldots, v_k)$ corresponds to the membership function of a fuzzy set, as in definition 3.6.

3.4 Summary

This chapter focused on the definitions from fuzzy set theory, computational verb theory, and constraint satisfaction problems that are applied in this work. The definitions in this

chapter are provided to ensure that this work is self-contained; they serve as a reference for the chapters that follow. In all instances where a definition is applicable reference will be made to this chapter.

Chapter 4

Computational verb theory and constraint satisfaction problem models for nouns, adjectives, verbs, and adverbs

This chapter sets out to show how computational verb theory and constraint satisfaction problem theory are applied in this work to model natural language nouns, adjectives, verbs, and adverbs. These theoretical models form the basis of a system for interactive narrative space generation from natural language words. A generic computational verb model for adjectives and verbs is presented in section 4.1. The notion of interpretive context, as applied to the computational verb-based word model presented in section 4.1, is defined in section 4.2. Section 4.3 describes the fuzzy set theory and computational verb theory models for nouns, adjectives, verbs, and adverbs. The computational verb models are related to the generic computational verb model of section 4.1. Crisp and fuzzy variable resolution strategies for determining variable values from the constructed computational verb-based word models are described in section 4.4.

4.1 A generic computational verb model for adjectives and verbs

Yang [106] states that when natural language is used to model changes of measured features over time, i.e. a dynamic process, three steps are involved:

- The continuous dynamic process is segemented into parts that have similar dynamics.
- A verb statement is assigned to every segmented piece of the dynamics.
- Verb statements in the form of a sentence or paragraph that encompasses the dynamic process in its entirety are connected.

A system that generates interactive narrative space from narrative text provided as input, however, requires a reversed process from the one listed above. In such a system, narrative text is provided as input and is related to previously constructed computational verb models. The computational verb models are in turn evaluated to provide values that parameterise the transformation of representational features. The transformed representational features serve to depict the narrative text in an alternative representation such as graphics, sounds, and the behaviour of artificial intelligence (AI) agents. An interactive narrative space is thus generated from natural language text.

A generic computational verb model is defined to allow the translation of narrative text to interactive narrative space by means of computational verb theory (CVT). The generic computational verb model enables evolving function word models to be dynamically constructed using the generic evolving function model as a template. The constructed evolving function models are used to model natural language words in terms of computational verb theory. The simplified definition of a computational verb defines a computational verb in terms of an evolving function, $\mathcal{E}_{\mathcal{V}}$ (refer to definition 3.14). This work models computational verbs in terms of the simplified definition of a computational verb and *ipso facto* models computational verbs in terms of evolving functions. Natural language sentences are modelled as verb sentences (statements) (refer to equation (3.35)). Each natural language sentence is modelled as a verb sentence in a specific interpretive

context, $\mathscr{C}$. The pairing of a context of interpretation, $\mathscr{C}$, and a verb sentence, $\mathcal{S}$, forms a verb element (refer to definition 3.22).

A time interval, $T = [T_0, T_0 + T_{\mathcal{C}}]$, is associated with a verb element. The time interval, T, corresponds to a segment of a dynamic system (refer to definition 3.10) that is modelled by a computational verb, $\mathcal{V}$. The computational verb, $\mathcal{V}$, combined with optional subject, N_s, and object, N_o, nouns form the verb sentence, $\mathcal{S}$. An evolving function model, $\mathcal{E}_{\mathcal{V}}$, of a computational verb, $\mathcal{V}$, by definition adopts the time interval, T, of the verb element, $(\mathcal{C}, \mathcal{S})$, in this work. A birth time

$$T_b = T_0 \tag{4.1}$$

and a death time

$$T_d = T_0 + T_{\mathcal{C}} \tag{4.2}$$

are therefore associated with every evolving function model. $T_{\mathscr{C}}$ is the lifetime of verb element, $(\mathcal{C}, \mathcal{S})$ (refer to definition 3.22).

This work models an evolving function, $\mathcal{E}_{\mathcal{V}}$, in terms of the following generic equation:

$$\mathcal{E}_{\mathcal{V}} = \alpha_1 f_1(t) + \ldots + \alpha_n f_n(t); t \in [T_b, T_d], T_b \in \mathbb{R}^+, T_d \in \mathbb{R}^+, \alpha_i \in \mathbb{R}. \tag{4.3}$$

The real-valued numbers, $\alpha_1, \ldots, \alpha_n$, are scaling values of the amplitudes of functions $f_1(t), \ldots, f_n(t)$ and are called "amplitude values" for short. The time value, t, is a non-negative real number on the interval $[T_b, T_d]$. T_b and T_d indicate the birth and death times of the evolving function model, $\mathcal{E}_{\mathcal{V}}$, as defined in equations (4.1) and (4.2).

This work focuses on evolving functions that are specified over a single time interval or repeat over a single time interval. An evolving function that repeats over a fixed interval, T, has the form:

$$\mathcal{E}_{\mathcal{V}} = \alpha_1 f_1(t - T_{b_j}) + \ldots + \alpha_n f_n(t - T_{b_j}); t \in [T_{b_j}, T_{d_j}], T_{b_j} \in \mathbb{R}^+, T_{d_j} \in \mathbb{R}^+, \alpha_i \in \mathbb{R}. \tag{4.4}$$

The evolving function has the form of equation (4.4) from the first repetition of the evaluation of the evolving function over the time interval, T, onwards. The indices, $j = 1, \ldots, m$, label the instances of the repetition of the evolving function, $\mathcal{E}_{\mathcal{V}}$, over intervals of the size of $T_{\mathscr{C}}$. Each time interval, T_j, has a birth time of $T_{b_j} = (j-1)T_{\mathscr{C}}$ and a death time, $T_{d_j} = jT_{\mathscr{C}}$. The first instance of the evolving function, $\mathcal{E}_{\mathcal{V}}$, over the

time interval, T, corresponds to repetition index, $j = 1$. The first repetition of $\mathcal{E}_{\mathcal{V}}$ in the form of equation (4.4) has the form of equation (4.3).

Equation (4.3) allows an evolving function, $\mathcal{E}_{\mathcal{V}}$, to be expressed as the sum of functions, $f_1(t), \ldots, f_n(t)$, scaled according to amplitude values, $\alpha_1, \ldots, \alpha_n$. Specific choices of functions, $f_1(t), \ldots, f_n(t)$, allow an evolving function, $\mathcal{E}_{\mathcal{V}}$, to be expressed in terms of the basic computational verb types. The basic computational verb types are static verbs (see definition 3.15), node verbs (see definition 3.16), centre verbs (see definition 3.17), and focus verbs (see definition 3.18). The specification of the amplitude values, $\alpha_1, \ldots, \alpha_n$, allows the chosen functions, $f_1, \ldots, f_n$, to be scaled according to the specific dynamics (changes over time) modelled for the interpretation of a word in a context.

The function, f_{grow}, that forms part of the evolving function model of the sentence "The tree grows." is, for example, defined by the equation

$$f_{grow} = 1 - e^{at}; a < 0, t \in \mathbb{R}^{+}. \tag{4.5}$$

If the growth of the tree is modelled as a node computational verb which is then scaled by an amplitude value, $\alpha = 3.0$, that represents the full-grown height of the tree, the evolving function model for the sample sentence "The tree grows." in the interpretive context "height" is given by

$$\mathcal{E}_{\mathcal{V}} = (3.0)(1 - e^{at}); a < 0, \alpha \in \mathbb{R}, t \in \mathbb{R}. \tag{4.6}$$

Equation (4.6) is specified in the form of equation (4.3), with $f_1 = f_{grow}$ and $\alpha = 3.0$.

4.2 Context as applied to computational verb-based word models

Natural language defines context as the part of text (sentence, paragraph) that surrounds a word or passage and enables the meaning of the word to be determined [2]. The notion of interpretive context applied to the computational verb-based word models presented in this work is specific. An interpretive context is defined to denote a single interpretation of the relationship between two words in a sentence. Adjectives and verbs in an interpretive

context are defined in relation to nouns. Adverbs in a context of interpretation are defined in relation to verbs and adjectives.

The objective of the present work is to translate natural language sentences into an interactive narrative space. Interactive narrative space is realised by multiple representational media such as graphics, sound, and the behaviour of AI agents. Multiple interpretations of a sentence in terms of these representations are therefore required, each of them defined in a context of interpretation. A further consideration for the definition of an interpretive context is that a single interpretation of a sentence can be further subdivided according to its constituent parts. The constituent parts of an interpretive context, as defined by the models of this work, refer to the dynamics modelled by computational verbs, and the variables whose values are determined by those dynamics.

The first division of an interpretive context subdivides an evolving function (refer to equation (4.3)) to model every term that corresponds to a generic function, f_i, in a separate context, which is labelled a subcontext. The second division of an interpretive context, labelled as a partial context, subdivides the amplitude value, α_i, that corresponds to a generic function, f_i, of a term in equation (4.3). The partial contexts, C'_i, express the amplitude value, α_i, of a term in equation (4.3) as a sum of values.

Equation (4.3) is expressed in terms of partial contexts by substituting amplitude values, α_i, with sums of values, $\alpha_{i_1} + \ldots + \alpha_{i_n}$, which yields the equation

$$\mathcal{E}_{\mathcal{V}} = (\alpha_{1_1}+\ldots+\alpha_{1_k})f_1(t)+\ldots+(\alpha_{n_1}+\ldots+\alpha_{n_m})f_n(t); t \in [T_{b_j}, T_{d_j}], T_{b_j} \in \mathbb{R}^+, T_{d_j} \in \mathbb{R}^+, \alpha_{i_j} \in \mathbb{R}. \tag{4.7}$$

Every α_{i_j} is modelled in a separate partial context. A partial context is related to a single subcontext.

For the computational verb-based word models that follow, interpretive context pairs the computational verb models with variables whose values are determined by the computational verb models. The variable values within an interpretive context therefore serve as measurements of the associated computational verb model. A computational verb model is measured by substituting a time value into the evolving function defined for an interpretive context. The measurement value is then stored within the variable associated with the computational verb model for the word in a context of interpretation. In the scope of this work, the variable that stores a measurement value is called an

"attribute variable". The computational verb model can be further subdivided in terms of its function terms and amplitude values as defined for the subcontexts and partial contexts described above.

The final function of a context of interpretation is to pair a word with the interactive narrative space representation which serves to depict the word in the context of interpretation. Interpretive context therefore groups computational verb models that describe the dynamics implied by a word with a representational means in an interactive narrative space. The measurement values that result from the computational verb models are then used to parameterise the transformation of the representational means. The transformed representational means serves to convey the dynamics of a word, for a context of interpretation, within the interactive narrative space. In this way the translation of narrative text to interactive narrative space is completed.

4.3 Computational verb theory models for nouns, adjectives, verbs, and adverbs

This section defines natural language nouns, adjectives, verbs, and adverbs in terms of the CVT set out in chapter 3. Models are constructed from CVT principles that represent a noun, an adjective, a verb, or an adverb in a context of interpretation (refer to section 4.2). Section 4.3.1 defines nouns in terms of natural language, fuzzy set theory, CVT, and the models implemented in this work. Adjectives are defined in section 4.3.2 in terms of natural language, fuzzy set theory, CVT, and the computational verb models of this work. Verbs are defined in terms of natural language and CVT in section 4.3.3. Adverbs are defined in terms of natural language, fuzzy set theory, and CVT in section 4.3.4. The definition of adverbs in terms of CVT is then used to construct a model for adverbs as applied in the present work.

4.3.1 Nouns

Natural language defines a noun as a part of speech that names a person, place, thing, quality or (performed) action and functions as the subject or object of a verb [2]. Fuzzy

set theory models nouns in terms of linguistic variables (refer to definition 3.9) where the noun in a context of interpretation is associated with a linguistic variable, $\mathcal{L}$. Linguistic variables are assigned linguistic values, typically in the form of adjectives, from a term set, $\mathcal{T}(\mathcal{L})$, as given by equation (3.24). CVT treats nouns as the subject or object of an atomic (refer definition 3.20) or molecular (refer to definition 3.21) verb sentence.

The present work models nouns in terms of their relationship to adjectives and verbs. A noun groups the values that result from the measurements obtained from computational verb models related to adjectives, verbs, and adverbs at a specific point in time. The measurement values from computational verb models are stored in variables called "attribute variables". In relation to an adjective or a verb, a noun uniquely identifies and groups the attribute variables defined for the computational verb models of an associated adjective or verb within a context of interpretation. A noun therefore uniquely labels and identifies a group of attribute variables.

On a separate note, every noun defined in a context of interpretation is associated with a narrative representation. Narrative representations are not dependent on CVT models implemented for the interpretive context and serve to represent the noun within the generated interactive narrative space. The values stored within attribute variables are used to parameterise transformations of narrative representations. Narrative representations and attribute variables are grouped according to unique instances of nouns and the contexts of interpretation considered for a specific scenario. The transformed narrative representations serve to depict the textual narrative within the interactive narrative space by a representational artifact such as a graphic, sound, or the behaviour of an AI agent.

4.3.2 Adjectives

Natural language defines an adjective as the part of speech that qualifies a noun by limiting, qualifying, or specifying the noun [2]. Fuzzy theory models an adjective as a linguistic value assigned to a linguistic variable, $\mathcal{L}$, which is associated with a noun (refer to definition 3.9) in a fixed context of interpretation. Every linguistic value is represented by a fuzzy variable, X (refer to definition 3.7), that ranges over a universe of discourse, U. The base variable of the universe of discourse, U, is denoted by u. A semantic rule,

$M(X)$, assigns a meaning to every linguistic value, $\mathcal{L}$, in the form of a fuzzy variable, X. By definition, the semantic meaning, $M(X)$, is equal to the restriction on the fuzzy variable, X (see equation (3.25)), denoted by $R(X)$. According to the definition of a fuzzy variable, the restriction, $R(X)$, has a membership function, $c(u)$, in terms of the base variable, u.

As an example, consider the phrase "pink apple" that contains the noun "apple" and the adjective "pink". The noun "apple" in fuzzy set theory may be associated with a linguistic variable "Red" for the fixed interpretive context of the similarity in colour of an object to the colour red. The adjective "pink" represents a linguistic value in the form of a fuzzy variable.

The fuzzy variable is called *pink* and ranges over a universe of discourse, $U = [0, 100]$. The universe of discourse, U, represents the redness of an object as a percentage. Let u be a base variable of the universe of discourse, U. The meaning of the fuzzy variable, *pink*, denoted by $M(pink)$, is defined to be the restriction on the variable, *pink*, denoted by $R(pink)$ (refer to equation (3.25)). The restriction has an associated membership function, $c_{pink}(u) \in [0, 1]$, in terms of the base variable, u.

For the adjective "pink" this membership function can be chosen as a function that returns the highest membership degree to the value, $u = 50$ (if we accept that the percentage of red in the colour pink is 50%). If the fuzzy variable assignment were to be defuzzified by choosing the percentage value with the highest membership degree to the restriction associated with the fuzzy variable, *pink*, a value of 50 would be assigned to the linguistic variable "Red" [93].

CVT models adjectives as computational verbs whose dynamics have been lost [107]. The evolving system of the computational verb (see definition 3.14) is therefore no longer producing a change in variable values over time. A computational verb whose dynamics have been lost is called a "dead" verb, one that has already reached its end state. An end state for a verb sentence (refer to definitions 3.20 and 3.21) can be described by a verb statement such as "BE + statement". According to the definition of a computational verb collapse (refer to definition 3.23), a statement such as "BE + statement" can be expressed in terms of a crisp or a fuzzy set.

This work distinguishes two cases when modelling adjectives:

- The first case represents adjectives that are classified in natural language as an adjective and describe a feature that does not change over time. Adjectives which describe a feature that does not change over time are modelled in terms of CVT by a static verb (see definition 3.15). An example of such an adjective is the word "red", which represents a static state over time. This first case of adjective usage represents an adjective viewed in terms of CVT.

- The second distinguished case represents those adjectives that are classified as adjectives in natural language, but which describe a feature that changes over time. An example of such an adjective is the word "bleeding" [2]. The adjective "bleeding" describes a feature that changes over time but is classified in natural language as an adjective. CVT considers such an adjective as a verb that is typically modelled by a centre verb [106] (see definition 3.18).

This work handles the two cases of adjective usage by specifying an adjective in terms of a verb within a context of interpretation. An adjective within a context of interpretation specifies an amplitude value, α, as defined for the equation (4.3) and a verb whose evolving function model serves as the model for the word "BE" in the "BE + statement" phrase. The "BE + statement" phrase expresses an adjective in terms of the end state of a verb sentence within CVT [107]. This work allows any verb to be chosen to model the word "BE" and as a result the statement "BE + statement" is generalised to the statement "verb + statement", as is the case for a computational verb collapse (refer to definition 3.23). The generalisation of "BE + statement" to "verb + statement" allows a word classified as an adjective in natural language to be treated as a verb in terms of CVT. This generalisation enables the second usage case of an adjective to be modelled in terms of a computational verb while retaining the natural language word-type classification.

An adjective is modelled within a context of interpretation by specifying an amplitude value, α, a verb word, and an attribute variable. The function, f, of the evolving function model for the specified verb in a context of interpretation is combined with the amplitude value, α, to form a single term of equation (4.3). The specific instance of equation (4.3), formed by choosing an amplitude value, $alpha$, and a function, f, serves to model the adjective. A verb that repeats over a time interval may also be chosen, and in such a

case the adjective is modelled by a single term of equation (4.4). A value retrieved from substituting a specific time value, t, into the constructed model is stored in the specified attribute variable. The attribute variable is unique to the adjective–noun pair for the narrative text sentence modelled in a single context of interpretation.

4.3.3 Verbs

Natural language defines a verb as the part of speech that expresses actions, occurrences or states of existence [2]. Fuzzy logic models natural language in terms of linguistic variables (refer to definition 3.9) and focuses on the relationship between adjectives and nouns [129] as modified by adverbs in terms of linguistic hedges (refer to definition 3.8). Fuzzy set theory models are not based on dynamics (changes over time) and can therefore not model actions as labelled by verbs. On the other hand, CVT does model words in terms of their dynamics, that is, the changes in state over time that they describe and is therefore able to model natural language verbs.

This thesis models a verb by means of CVT by specifying an evolving function in terms of amplitude values and functions and attribute variables that receive values from the constructed evolving functions. A single permutation of functions and amplitude values defines an evolving function in the form of equation (4.3). The defined evolving function models the dynamics of a verb in a single context of interpretation. The chosen attribute variable stores the measurement values obtained from the constructed evolving function at a point in time. The attribute variable is grouped with the evolving function in an interpretive context.

The evolving function that models the dynamics of a verb within a context of interpretation is further subdivided into subcontexts. Every subcontext of interpretation (refer to section 4.2) relates to a term of the evolving function equation (4.3). The subcontext enables the term to be named according to the role the term plays in the interpretation of the dynamics of a verb. As an example, consider the sentence “The frog jumps off the rock.”. The sentence can be modelled by an evolving function of the form:

$$\mathcal{E}_{\mathcal{V}} = \alpha_1(1.0) + \alpha_2(1 - e^{bt}); b < 0, \alpha_i \in \mathbb{R}, t \in \mathbb{R}^+. \tag{4.8}$$

The evolving function presented in equation (4.8) is subdivided according to the

models in this work by defining the components of each term in a separate subcontext. The first term relates to the starting position of the frog and may be modelled in a subcontext labelled as "Start". The second term models the change in position of the frog and may be modelled in a subcontext labelled "Change". This allows a complex evolving function model to be decomposed into subcontexts. Each subcontext relates to a subcomponent of the dynamics of a verb. The subcontexts may then be named in non-mathematical terminology to ease further the modelling and understanding of the dynamics of verbs.

The evolving function of equation (4.8) may be further subdivided in accordance with the models presented in this work, by specifying the amplitude values, α_i, as sums of multiple values and/or variables. Partial contexts specified for the subcontext related to a term of the evolving function equation (4.8) allow an amplitude value, α_i, to be expressed in terms of multiple attribute variables and/or real-valued numbers as a sum. As an example, consider again the sentence "The frog jumps off the rock.". The initial height of the frog above the ground is modelled within the first term of the evolving function of equation (4.8) in a subcontext labelled as "Start". The initial height of the frog above the ground may be expressed as a sum of the height of the rock and the height of the frog. The "Start" subcontext may thus be further subdivided into two partial contexts related to the height of the rock and the frog respectively. Each partial amplitude value, α_{i_j}, of equation (4.7) is thus modelled in a suitably labelled partial context such as "Other" for the height of the rock and "Self" for the height of the frog.

The final component of a verb model within this work is the specification of attribute variables. An attribute variable may be specified in two roles for a verb model within a context of interpretation. The first role that an attribute variable may play in a verb model is that of the storage variable. An attribute variable may store a measurement value obtained from the constructed evolving function model of a verb in a context of interpretation. A measurement value is obtained by evaluating the constructed evolving function at a specific time.

The second role that an attribute variable may play in a verb model for a context of interpretation is that of an amplitude value, α_i. For an amplitude value, α_i, specified in terms of an attribute variable, V, the constant, κ, is introduced to serve as a scaling

factor for the attribute variable, V. The amplitude value, α_i, can therefore be expressed as

$$\alpha_i = \kappa V. \tag{4.9}$$

An amplitude value, α_i, expressed as a sum of values defined within partial contexts, may be defined in terms of attribute variables, $V_1, \ldots, V_n$. An amplitude value, α_i, defined over multiple partial contexts is expressed as

$$\alpha_i = \kappa_1 V_1 + \ldots + \kappa_n V_n; \kappa_i \in \mathbb{R}. \tag{4.10}$$

Note that every term, $\kappa_i V_i$, in equation (4.10) may be substituted for a fixed amplitude value, α_{i_j}.

4.3.4 Adverbs

Natural language defines an adverb as the part of speech that modifies another verb, adjective or adverb [2]. Fuzzy set theory defines adverbs in terms of linguistic hedges [130] (refer to definition 3.8). Linguistic hedges are functions that manipulate the membership functions of fuzzy sets. Examples of such functions are concentrators, as in equation (3.21), and dilators as in equation (3.22). CVT defines adverbs as modifiers of computational verbs and therefore as modifiers of their associated evolving functions [100].

Yang identifies three types of adverb within CVT, namely those that modify time, space, and perception respectively. The transformation performed by an adverb of time, $\beta_{\mathcal{T}}$, on the evolving system of a computational verb, $\mathcal{V}$, (refer to definition 3.13) is represented by the evolving system [100]:

$$\Phi_\beta \circ \mathscr{E}_{\mathcal{V}} = [\mathscr{T}, \mathscr{X}_s, \mathscr{X}_p, \mathcal{I}_{\mathcal{V}}, \mathcal{F}_{\mathcal{V}}, \Phi_\beta] \tag{4.11}$$

with $\mathscr{T}, \mathscr{X}_s, \mathscr{X}_p, \mathcal{I}_{\mathcal{V}}$ and $\mathcal{F}_{\mathcal{V}}$ defined as for equation (3.30) and $\Phi_\beta \circ \mathscr{E}_{\mathcal{V}}$ read as the mapping performed by the modifying system, Φ_β, on the evolving system, $\mathscr{E}_{\mathcal{V}}$. The modifying system is given by

$$\Phi_\beta : \mathscr{T} \mapsto \mathscr{T}^\Phi \tag{4.12}$$

$\mathscr{T}$ represents the time interval for the evolving system, $\mathscr{E}_{\mathcal{V}}$. $\mathscr{T}^{\Phi}$ represents a transformed time interval formed by applying the modifying system, Φ_{β}, to the evolving system of the computational verb, $\mathcal{V}$ (refer to definition 3.13).

The mapping performed by the modifying system, φ_{β}, on the outer observable system, $\mathcal{F}_{\mathcal{V}}$, of the evolving system, $\mathscr{E}_{\mathcal{V}}$, (refer to definition 3.13) is given by

$$\varphi_{\beta} \circ \mathcal{F}_{\mathcal{V}} : \mathscr{T}' \times \mathscr{X}_s' \times \mathscr{X}_p' \mapsto \mathscr{T}^{\varphi} \times \mathscr{X}_s' \times \mathscr{X}_p'. \tag{4.13}$$

$\mathscr{T}', \mathscr{X}_s', \mathscr{X}_p'$ are defined as for equation (3.32). $\mathscr{T}^{\varphi}$ represents a transformed perceived time interval formed by applying the modifying system, φ_{β}, to the outer observable system, $\mathcal{F}_{\mathcal{V}}$. $\varphi_{\beta} \circ \mathcal{F}_{\mathcal{V}}$ is read as the mapping performed on the outer system, $\mathcal{F}_{\mathcal{V}}$, by the modifying system, φ_{β}. The modifying system is given by

$$\varphi_{\beta} : \mathscr{T}' \mapsto \mathscr{T}^{\varphi}. \tag{4.14}$$

The transformation performed by an adverb of space, $\beta_{\mathscr{X}_s}$, on the evolving system of a computational verb, $\mathcal{V}$, (refer to definition 3.13) is represented by the evolving system [100]:

$$\Phi_{\beta} \circ \mathscr{E}_{\mathcal{V}} = [\mathscr{T}, \mathscr{X}_s, \mathscr{X}_p, \mathcal{I}_{\mathcal{V}}, \mathcal{F}_{\mathcal{V}}, \Phi_{\beta}], \tag{4.15}$$

with $\mathscr{T}, \mathscr{X}_s, \mathscr{X}_p, \mathcal{I}_{\mathcal{V}}$ and $\mathcal{F}_{\mathcal{V}}$ defined as for equation (3.30) and $\Phi_{\beta} \circ \mathscr{E}_{\mathcal{V}}$ read as the mapping performed by the modifying system, Φ_{β}, on the evolving system. The modifying system is given by

$$\Phi_{\beta} : \mathscr{X}_s \mapsto \mathscr{X}_s^{\Phi}. \tag{4.16}$$

$\mathscr{X}_s$ represents the regular physical space for the evolving system. $\mathscr{X}_s^{\Phi}$ represents the transformed physical space formed by applying the modifying system, Φ_{β}, to the evolving system of computational verb, $\mathcal{V}$ (refer to definition 3.13).

The mapping performed by the modifying system, φ_{β}, on the outer observable system, $\mathcal{F}_{\mathcal{V}}$, (refer to definition 3.13) is given by

$$\varphi_{\beta} \circ \mathcal{F}_{\mathcal{V}} : \mathscr{T}' \times \mathscr{X}_s' \times \mathscr{X}_p' \mapsto \mathscr{T}' \times \mathscr{X}_s^{\varphi} \times \mathscr{X}_p'. \tag{4.17}$$

$\mathscr{T}', \mathscr{X}_s', \mathscr{X}_p'$ are defined as for equation (3.32). $\mathscr{X}_s^{\varphi}$ represents a transformed perceived physical space formed by applying the modifying system, φ_{β}, to the outer observable

system, $\mathcal{F}_{\mathcal{V}}$ (refer to definition 3.13). $\varphi_{\beta} \circ \mathcal{F}_{\mathcal{V}}$ is read as the mapping performed on the outer system, $\mathcal{F}_{\mathcal{V}}$, by the modifying system, φ_{β}. The modifying system is given by

$$\varphi_{\beta} : \mathscr{X}_s' \mapsto \mathscr{X}_s^{\varphi}. \tag{4.18}$$

Adverbs that modify the space of perception of a computational verb are more complex in CVT. Adverbs that alter the perception space perform transformations on the evolving function associated with a computational verb (refer to definition 3.14) [120]. Yang [120] generalises adverbs that modify the space of perception for a computational verb into two cases. Adverbs of perception may modify the amplitude (first case) and/or frequency (second case) of the evolving function of a computational verb (refer to definition 3.14).

1. An adverb, β, that acts as a modifier of the amplitude of an evolving function, $\mathcal{E}_{\mathcal{V}}$, where $\mathcal{E}_{\mathcal{V}}$ is defined in a fuzzy space, can be applied to the fuzzy state of $\mathcal{E}_{\mathcal{V}}$ at a point in time, t, yielding

   $$\mathcal{E}_{\mathcal{V}}(t) \circ \Psi = \mu(x) \circ \Psi. \tag{4.19}$$

   $\mu(x)$ is a membership function (in terms of base variable x) that represents the end state of $\mathcal{E}_{\mathcal{V}}$ as given by a computational verb collapse (refer to definition 3.23). Ψ represents a modifying function applied to the end state of the computational verb, $\mathcal{V}$. An example of a transformation function that may be chosen as Ψ is a concentrator (refer to equation (3.21)). An example of an adverb, β, that corresponds to the choice of a concentrator as the transformation function of an adverb, Ψ, is the adverb "very".

   An adverb, β, that acts as a modifier of the amplitude of an evolving function, $\mathcal{E}_{\mathcal{V}}$, where $\mathcal{E}_{\mathcal{V}}$ is defined in a crisp space and the crisp states are later fuzzified, can be applied to the amplitude of $\mathcal{E}_{\mathcal{V}}$ yielding

   $$(\mathcal{E}_{\mathcal{V}} \circ \Psi) \circ \mu = \mathcal{E}_{\mathcal{V} \circ \Psi}(t) \circ \mu. \tag{4.20}$$

   $\mathcal{E}_{\mathcal{V} \circ \Psi}$ represents the transformed evolving function formed by the application of the transformation function, Ψ, to the amplitude value, α, of $\mathcal{E}_{\mathcal{V}}$. μ is the membership function that serves to fuzzify the crisp state of the evolving function, $\mathcal{E}_{\mathcal{V}}$. $\mathcal{E}_{\mathcal{V} \circ \Psi}(t) \circ \mu$

represents the fuzzification of a crisp state of the transformed evolving function at a point in time, t.

2. An adverb, β, that acts as a modifier of the perceived time of a computational verb, $\mathcal{V}$, modifies the phase of the associated evolving function, $\mathcal{E}_{\mathcal{V}}$, yielding

$$\mathcal{E}_{\mathcal{V}} \circ \Psi = \mathcal{E}_{\mathcal{V}}(\Psi \circ t). \tag{4.21}$$

Ψ represents the modifying function applied to the time parameter, t, of the evolving function. An example of a choice for Ψ is

$$\Psi \circ t = \Psi(t) = \delta t \tag{4.22}$$

where $\delta \in \mathbb{R}$ scales the time value.

This work treats natural language adverbs as adverbs that modify the space of perception for a computational verb, $\mathcal{V}$. Adverbs are modelled to modify the amplitude and/or phase of the evolving function of a computational verb (refer to definition 3.14). The mapping on the amplitude values of an evolving function, $\mathcal{E}_{\mathcal{V}}$, in the form of equation (4.3) is similar to that of equation (4.20). The transformation function, Ψ, (as for equation (4.20)) is applied directly to the $\mathcal{E}_{\mathcal{V}}$ that serves as the model for a verb or an adjective. The state of an evolving function at a time, t, is not fuzzified within the scope of this work as was the case for the evolving function of equation (4.20).

An adverb, β, that acts as a modifier of the amplitude of a constructed evolving function (refer to (4.3)) is defined to perform a mapping on the amplitude values of the individual terms. The amplitude values modified according to a specified transformation function, Ψ, are determined by the context of interpretation (see section 4.2) that the adverb is defined in. An adverb defined in a normal context of interpretation specifies a transformation, Ψ, that is applied to the each amplitude value, α_i, of the evolving function equation.

An adverb defined in a subcontext (refer to section 4.2), specifies a transformation function, Ψ, that is applied to the single amplitude value, α_i, defined within that subcontext. A subcontext whose amplitude value is defined in terms of partial contexts (refer to section 4.2), specifies a transformation function, Ψ, which is applied to all amplitude

values, α_{i_j} (refer to equation (4.7)). The amplitude values, α_{i_j}, are specified in partial contexts. The partial contexts relate to the subcontext that the adverb, β, is defined in.

An adverb defined in a partial context specifies a transformation function, Ψ, applied to a single amplitude value, α_{i_j}. The amplitude value, α_{i_j}, transformed by the specified transformation function, Ψ, corresponds to the same partial context that the adverb, β, is defined in.

An adverb, β, that acts as a modifier of time is defined as a transformation of the time value (phase) associated with the evolving function of a computational verb, $\mathcal{V}$. An adverb defined in a context of interpretation specifies a transformation function, Ψ, that modifies the time value passed to the evolving function, $\mathcal{E}_{\mathcal{V}}$ (refer to equation (4.21)), as a parameter. The evolving function, $\mathcal{E}_{\mathcal{V}}$, is specified in the same context of interpretation as the adverb, β.

For an adverb–verb or adverb–adjective pair, the evolving function, $\mathcal{E}_{\mathcal{V}_i}$, defined in the context of interpretation, $\mathscr{C}_i$, is transformed by the transformation function, Ψ_i, that is also specified in the context of interpretation, $\mathscr{C}_i$. A transformed evolving function, $\mathcal{E}^{\Psi}_{\mathcal{V}_i}$, is obtained from the transformation and evaluated to obtain a measurement value at a specific time, t. The measurement values calculated from all transformed (and non-transformed) evolving functions, specified over all applicable contexts of interpretation, are used to parameterise the transformation of narrative representation media. The transformed narrative media (graphics, sounds, AI scripts) serve to convey the effect of the adverbs in the generated interactive narrative space.

4.3.5 Summary

This section defined nouns, adjectives, verbs, and adverbs in terms of natural language, fuzzy set theory, CVT, and the word models described in this work. Nouns were defined as a grouping of attribute variables and narrative representations within a context of interpretation. Adjectives were defined in terms of natural language, fuzzy set theory, and CVT and modelled according to the computational verb models presented in section 4.1. Verbs were defined in terms of natural language and CVT, and the models for natural language verbs in this work were presented in accordance with the computational verb models described in section 4.1. Adverbs were defined in terms of natural language,

fuzzy set theory, and CVT, and the adverb models in this work were related to CVT models.

4.4 Variable resolution procedures

This section deals with variable resolution in the light of the inconsistencies and ambiguities introduced when natural language sentences are modelled mathematically. Section 4.4.1 examines the considerations for a robust variable value resolution procedure. The resolution procedure handles variable assignments that result from the evaluation of natural language models.

Section 4.4.2 describes a crisp variable resolution procedure. The crisp variable resolution procedure treats the variable value assignments that result from the evaluation of the computational verb models (see section 4.3) as a system of linear equations. Procedures for the resolution of the resulting system of linear equations are presented.

Section 4.4.3 presents a fuzzy variable resolution procedure. Variable assignments resulting from the computational verb word models presented in section 4.3 are modelled as fuzzy constraints. A literature study of constraint satisfaction problems (CSPs) in relation to the variable resolution procedures described is reported on. Fuzzy constraint satisfaction problems (FCSPs) are introduced as a relaxation of CSPs that allows for approximate solutions to inconsistent systems of linear equations. To conclude, the fuzzy constraint satisfaction procedures in this work are described.

4.4.1 Considerations for variable resolution

A variable value resolution procedure for assigning measurement values to attribute variables has to consider three important factors. The formation of evolving function equations from the substitution of time values into evolving function models of adjectives and verbs, the assignment of measurement values to interactive attribute variables (refer to definition 3.5), and conflicting assignments of measurement values to attribute variables are all discussed under separate headings in this section 4.4.1.3.

4.4.1.1 The formation of evolving function equations

The computational verb models presented for adjectives and verbs were modelled according to the evolving function models defined in section 4.1. To obtain measurements from the computational verb models, time values are substituted into the evolving function that corresponds to a verb or an adjective within a context of interpretation (refer to sections 4.3.2 and 4.3.3). Before the time values are substituted, the evolving functions are modified according to adverb transformation functions as presented in section 4.3.4. The adverb transformation functions correspond to the adverbs associated with an adjective or verb in natural language text and defined in a corresponding context of interpretation (refer to section 4.2).

The measurement values obtained from the substitution of time values into evolving function models are assigned to attribute variables. Every attribute variable is specific to an instance of a noun word in the natural language text modelled. Every adjective–noun or verb–noun pair corresponds to a single assignment of an attribute variable for a single context of interpretation. The evolving function that produces an attribute variable is further subdivided according to subcontexts and partial contexts (refer to section 4.2).

For an adjective, the value assigned by an evolving function measurement to an attribute variable is simple. The measurement value assigned depends on the specified amplitude value and a chosen function. The chosen function corresponds to the single term evolving function that represents the choice of verb for the adjective (refer to section 4.3.2).

For a verb, the value assigned by an evolving function measurement is more complex (refer to section 4.3.3). The measurement value depends on an evolving function specified over multiple terms with multiple amplitude values and multiple chosen functions. The amplitude values may be comprised of a sum of multiple values. A verb may also have both a subject and an object noun. A verb with a subject noun may have its amplitude values expressed in terms of attribute variables grouped with another noun (refer to section 4.3.1). An attribute variable may represent a single amplitude value (refer to equation (4.3)) or form part of the sum that expresses an amplitude value over multiple partial contexts (refer to equation (4.7)).

4.4.1.2 Interactive variable assignments

An attribute variable variable used as an amplitude value of an evolving function may be used concurrently to store the measurement value of another evolving function. The second evolving function may be used to model an adjective–noun or verb–noun pair in a context of interpretation. This creates a dependency between an attribute variable that receives a measurement value from an evolving function and the attribute variable that may represent the whole or part of an amplitude value in another evolving function.

The dependency between the two attribute variables means that any restriction placed on both attribute variables cannot be separated into two separate restrictions. A restriction placed on the possible values of one attribute variable will affect the value of the other attribute variable. Thus the assignment of measurement values to attribute variables should be treated as for interactive variables (refer to definition 3.5). For multiple interdependent measurement value to attribute variable assignments that result from multiple noun–adjective and noun–verb pairs modelled in multiple contexts of interpretation, assignment should be treated as for equation (3.15).

As an example, consider the sentences “The rock is small. The frog jumps onto the rock.”. The attribute variable variable that receives a value from the evolving function model of “jumps onto” in the context of height may be “Height”. The attribute variable that receives a value from the evolving function model of “is small” in the context of height may also be “Height”. Each attribute variable “Height” is unique to the nouns “frog” and “rock” respectively (refer to section 4.3.1). For the example sentences, the amplitude value of “jumps onto” may in part be determined by the attribute variable “Height” of the noun “rock”. Thus a dependency exists between the two instances of the attribute variable “Height” of the nouns “frog” and “rock”. An assignment to the attribute variable “Height” of either the nouns “frog” or “rock” should therefore be handled as an interactive variable assignment in the variable value resolution procedure (refer to definition 3.5).

4.4.1.3 Inconsistent variable assignments

Separate noun–adjective and noun–verb pairs may assign a measurement value obtained from the evaluation of their corresponding evolving function models to the same at-

tribute variable. The attribute variable may also be grouped with the same instance of a noun. The assignments of differing measurement values to the same instance of an attribute variable may result either from separate assignments in the same context of interpretation or from assignments in different contexts of interpretation. Both cases have the result that an instance of an attribute variable receives multiple measurement value assignments. If such assignments were to be represented as a system of equations, the resulting system would have no unique solution and would therefore be inconsistent.

As an example, consider the sentences "The car is to the left of the house. The car is to the right of the house.". The attribute variable "Position" may represent the position of the car in a context of interpretation "Placement". The attribute variable "Position" will be assigned a measurement value retrieved by evaluating the evolving functions that correspond to the verb phrases "is to the left" and "is to the right" in the context of interpretation "Placement" at a specific point in time. Two different measurement values are thus assigned to the attribute variable "Position". The same instance of the noun "car" is implied because there are no other adjustments such as adjectives to distinguish the two instances of the noun "car" and no additional context is provided. Different measurement values are therefore assigned to the same instance of the attribute variable "Position" grouped with the single instance of the noun "car" (refer to section 4.3.1). A system of equations that represents the assignment of the measurement values to the attribute variable "Placement" would therefore prove to be inconsistent as no single unique value for the instance of the attribute variable "Position" exists.

4.4.1.4 Summary

This section presented important considerations for a variable value resolution procedure that handles measurement value to attribute variable assignments. The measurement value assignments occur when time values are substituted into the evolving function models of adjectives and verbs in this work. The formation of evolving function equations was discussed in terms of evolving function models for adjectives and verbs. The assignment of measurement values to interactive attribute variables was discussed. Interactive attribute variables resulted from the use of attribute variables, grouped under object nouns, in the amplitude values of the evolving functions models of verbs in the

present work. The occurrence of conflicting measurement value assignments to an attribute variable was presented; the value assignments resulted from conflicting natural language statements modelled within the computational verb models of this work. The measurement value assignments retrieved when evaluating the models of conflicting natural language statements result in inconsistent systems of equations that have no precise unique solution.

A robust procedure for variable resolution should address the stated considerations. The sections that follow set out the details of two candidate attribute variable value resolution procedures: a crisp attribute variable measurement value assignment procedure and a fuzzy attribute variable measurement value assignment procedure.

4.4.2 Crisp procedure for assigning measurement values to attribute variables

A series of examples is presented in the form of diagrams to simplify the explanation of a crisp procedure for assigning measurement values to attribute variables. Each example corresponds to a consideration for an attribute variable value resolution system as presented in section 5.1.

The diagrams that represent the examples depict nouns as circles, the associated attribute variables, V_i, for a context of interpretation (refer to section 4.2 as rounded squares, and amplitude values, α_i, (refer to sections 4.3.2 and 4.3.3) as squares.

Solid lines in the example diagrams depict the association of an attribute variable with a noun in an adjective–noun or verb–noun pair for a context of interpretation (refer to section 4.3.2 and 4.3.3). Solid lines are labelled by the corresponding context (refer to section 4.2) which groups the attribute variable with an evolving function. The evolving function is evaluated to obtain a measurement value that is stored in the attribute variable.

Solid arrows in the example diagrams depict the choice of an amplitude value specified within a context, subcontext or partial context for an adjective–noun or verb–noun(s) pair. The solid arrows are labelled by the corresponding context, subcontext, or partial context which defines the choice of amplitude value if the amplitude value is defined within a context of interpretation, subcontext, or partial context. Amplitude values

that are set to the default values specified for a context, subcontext or partial context are labelled by the word "default". The context, subcontext or partial context that defines the default amplitude value is provided in parenthesis. A default value obtained from a representational element associated with the noun for a context of interpretation (refer to section 4.3.1) is labelled by the word "default". The noun associated with the narrative representation for the context of interpretation is provided in parenthesis. A dotted arrow indicates an amplitude value specified in a partial context of a subcontext (refer to section 4.2).

Figure 4.1 depicts the relationship between the noun "fox" and the adjectives "quick" and "brown" modelled in the interpretive contexts of "colour" and "attribute". An example of a natural language phrase that corresponds to the relationships is "... the quick brown fox ...". The context of interpretation "colour" is modelled in terms of the classification of perceived colours of objects into numbered categories. The category numbered as 2 is assigned to the colour labelled "brown" for this example. The category number is specified in an amplitude value node. The amplitude value is specified within the context "colour" for the adjective "brown". The context of interpretation "attribute" is modelled in terms of physical measurements of character attributes. The amplitude value 10 may be interpreted as the maximum speed of a fox in metres per second. The amplitude value 10 is assigned as a default retrieved from the representational element associated with the noun "fox" in the context of interpretation "attribute".

The crisp variable value resolution procedure presented in this work handles the example shown in figure 4.1 by dynamically constructing evolving functions. The evolving functions constructed are in the form of equation (4.3). The simple function

$$f_1(t) = 1.0; t \in \mathbb{R}^+ \tag{4.23}$$

may be chosen as the function, f_1, of the evolving function model of both adjectives. The adjectives "brown" and "quick" are thus modelled as static computational verbs (refer to section 4.3.2).

The evolving function models for the adjectives "brown" and "quick" are specified in their respective contexts of interpretation "colour" and "attribute"(refer to section 4.2). The chosen function, f_1, and the amplitude values of figure 4.1 are substituted into the generic evolving function equation (4.3). The substitution of the chosen function and

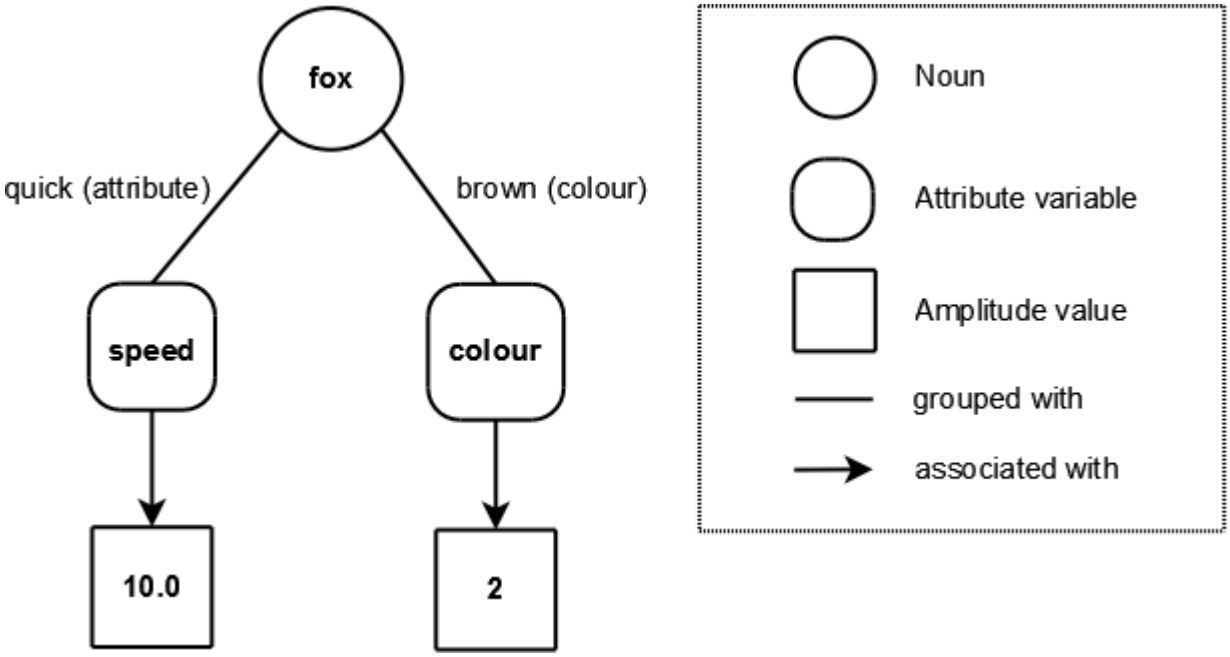

Figure 4.1: The association of attribute variables with amplitude values in contexts of interpretation

amplitude values produces evolving functions that correspond to the adjectives "brown" and "quick" for the contexts of interpretation "colour" and "attribute" respectively. The evolving functions produced are evaluated at a specific time, t, to obtain measurement values that are stored in the attribute variables "colour" and "speed". In this manner the attribute variable resolution procedure is completed.

Figure 4.2 shows the association of two attribute variables "Position (Y)" and "Height" with the nouns "fox" and "dog" in a context of interpretation "space". An example of an English sentence that may relate to such associations is "The fox jumps over the big dog.". The attribute variable "Position (Y)" is interpreted as a measure of the height of an object above the ground. The value of attribute variable "Position (Y)" is retrieved from the evolving function model of a verb phrase "jumps over" in the context of interpretation "space". The attribute variable "Height" is interpreted as a measure of the vertical size of an object labelled by an associated noun. The attribute variable "Height" is assigned to the value "1.0" by the evolving function model for the adjective "big" in the context of interpretation "space". The value "1.0" is interpreted as a size in metres.

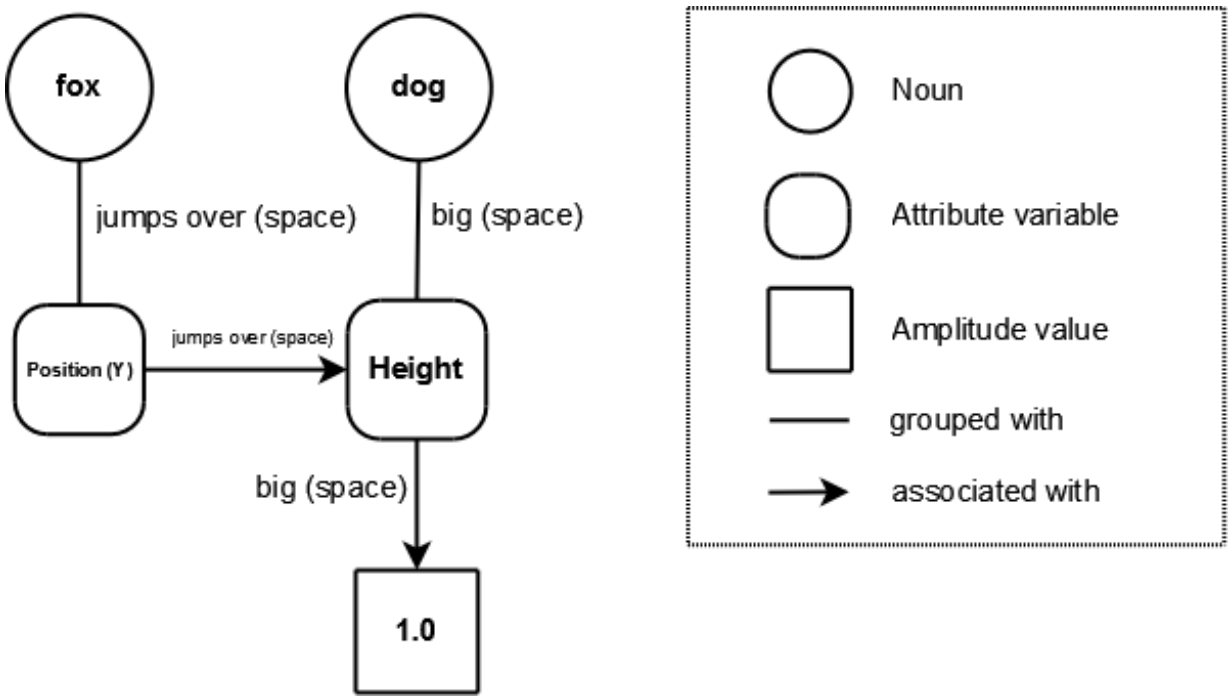

Figure 4.2: Interactive attribute variables

The attribute variable "Position (Y)" is associated with the attribute variable "Height" by choosing the value of "Height" as the amplitude value of a term in the evolving function model (refer to equation (4.3)). The evolving function term is modelled for the verb phrase "jumps over" in a subcontext of the context of interpretation "space". The value of attribute variable "Position (Y)" therefore depends on the value of attribute variable "Height". The attribute variables "Position (Y)" and "Height" are interactive as given by definition 3.5. The assignment of measurement values to the interactive attribute variables "Position (Y)" and "Height" should therefore be handled as for equation (3.15).

To allow for the assignment of measurement values to interactive attribute variables, the crisp variable resolution procedure forms graphs similar to the diagram shown in figure 4.2. A graph, G_i, is formed for every attribute variable, V_i, that should be assigned a measurement value. Every G_i is traversed to solve for the unknown attribute variable values and amplitude values. The traversal continues until an amplitude value, α_i, is reached. The amplitude value is substituted into the evolving function equation (refer

to equation (4.3)) of its associated attribute variable V_j. If all amplitude values for the evolving function, $\mathcal{E}_{V_j}$, associated with V_j are known, $\mathcal{E}_{V_j}$ is evaluated for a time value, t. The measurement value obtained from $\mathcal{E}_{V_j}$ is propagated through the connected nodes of the graph until an unknown attribute variable or amplitude value is once again reached.

For the example illustrated in figure 4.2, a graph, G_1, will be constructed and traversed for the attribute variable "Position (Y)" of the noun "fox". The traversal of G_1 reaches the amplitude node "1.0" and its value is propagated to the associated attribute variable node "Height". The amplitude value "1.0" is substituted into the evolving function, $\mathcal{E}_{\text{be}}$, associated with the attribute variable "Height" by the adjective "big" in the context of interpretation "space". The evolving function, $\mathcal{E}_{\text{be}}$, is defined by a choice of a function f_1 and the determined amplitude value α_2. $\mathcal{E}_{\text{be}}$ is evaluated for a time value, t, to obtain a measurement value that is stored in the attribute variable "Height".

The value of attribute variable "Height" is propagated to the attribute variable "Position (Y)" by substituting the measurement value of $\mathcal{E}_{\text{be}}$ at a time, t, as an amplitude value, α_1, into the evolving function $\mathcal{E}_{\text{jumps over}}$. The evolving function, $\mathcal{E}_{\text{jumps over}}$, is associated with the attribute variable "Position (Y)" in a subcontext of the context of interpretation "space". The evolving function, $\mathcal{E}_{\text{jumps over}}$, is defined by the choice of a function,

$$f_2(t) = \sin t, t \in [0, \pi] \tag{4.24}$$

and the determined amplitude value, α_1. $\mathcal{E}_{\text{jumps over}}$ is evaluated a time, t, and a measurement value is obtained for the attribute variable "Position (Y)". The crisp variable resolution procedure is complete and all the attribute variables have been assigned measurement values.

If a traversal of a G_i reaches an attribute variable, V_k, that has no associated amplitude value, α_k, a default value is retrieved from the context, subcontext or partial context of interpretation, if available. If no default value is specified in the context, subcontext or partial context of interpretation, a default value is retrieved from the narrative representation associated with the noun in the main context of interpretation (refer to section 4.3.1). If no default value can be retrieved from the associated narrative representation, both α_k and V_k remain unresolved. If the traversal of G_i forms a cycle, the unresolved amplitude values and attribute variables in the cycle are left unresolved. The traversal

of the G_i continues from the node where the cycle started, along another edge.

Figure 4.3 shows the association of an attribute variable "Position (Y)" with the noun "fox" in a context of interpretation "space". The attribute variable is associated with amplitude value "Height", which in turn is associated with the nouns "fox" and "dog", and with the attribute variable "Position (Y)" of the noun "dog". The attribute variables are interpreted as for figure 4.2. The respective amplitude values in terms of the attribute variables "Height" and "Position (Y)" are defined in the partial contexts "Self", "Other", and "Reference". The partial contexts are associated with a subcontext of a context of interpretation. The instances of the attribute variable "Height" are assigned default values from the respective narrative representations associated with the noun "fox" and "dog" in the context of interpretation. The attribute variable "Position (Y)" is assigned a default value specified within the partial context "Reference".

For the example presented in figure 4.3, the crisp variable resolution procedure constructs a graph, G_1, for the attribute variable "Position (Y)" associated with the noun "fox" in the context of interpretation "space". The graph G_1 is traversed and the node for attribute variable "Height" associated with the attribute variable "Position (Y)" of the noun "fox" is reached. The association exists because the attribute variable "Height" grouped with the noun "fox" is specified as an amplitude value, α_{1_1}, in the partial context "Self". "Self" in turn is defined as a partial context related to a subcontext for the context of interpretation "space".

The attribute variable "Height" associated with the noun "fox" has no other measurement value assigned by an evolving function in a context of interpretation. A default value is therefore retrieved from the narrative representation of the noun "fox" in the context of interpretation "space". The default value of "0.5" is substituted as an amplitude value, α_{1_1}, into an evolving function, $\mathcal{E}_{\text{jumps over}}$, which has the form of equation (4.7). $\mathcal{E}_{\text{jumps over}}$ is the evolving function associated with the attribute variable "Position (Y)" in the context of interpretation "space".

The traversal of G_1 continues until all amplitude values $\alpha_{1_1}, \alpha_{1_2}, \alpha_{1_3}$ are substituted into the evolving function equation $\mathcal{E}_{\text{jumps over}}$. Scaling factors of $\kappa_1, \kappa_2, \kappa_3 = 1.0$ defined in the partial contexts "Self","Other", and "Reference" are substituted into $\mathcal{E}_{\text{jumps over}}$. The scaled amplitude values of $\mathcal{E}_{\text{jumps over}}$ are calculated and the dy-

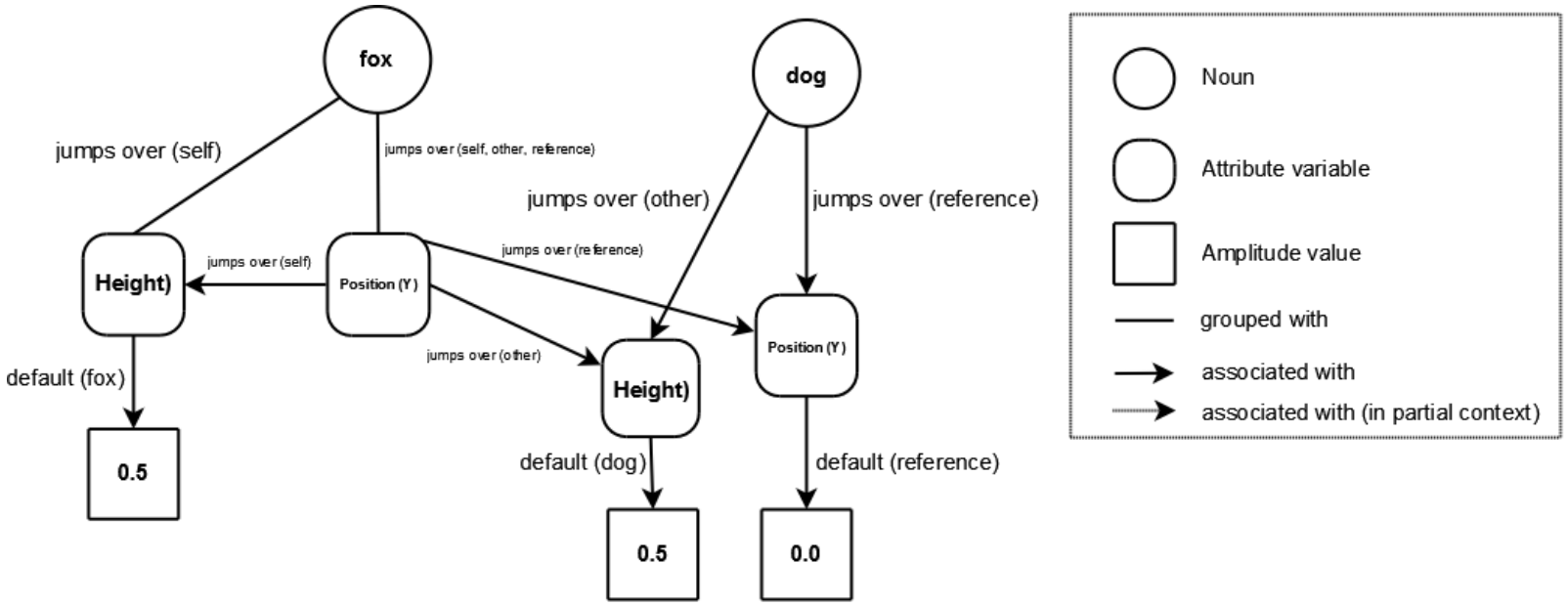

Figure 4.3: The association of multiple partial amplitude values with an attribute variable over multiple partial contexts

namic construction of the evolving function for the verb–noun pairings is complete. $\mathcal{E}_{\text{jumps over}}$ is evaluated at a time, t, to obtain a measurement value for the attribute variable "Position (Y)" of the noun "fox". The values of the instances of the attribute variable "Height", associated with the nouns "dog" and 'fox", and attribute variable "Position (Y)" associated with the noun "dog" are all determined from the substitution of default values. The default values are defined within their respective partial contexts or within the narrative representations associated with the nouns "fox" and "dog" in the context of interpretation "space". All attribute variables have been assigned measurement values and the crisp variable resolution procedure is complete.

If the graph, G_i, of every attribute variable, V_i, has been traversed and the related attribute variable values calculated, the first step of the crisp variable resolution process is complete. Unresolved attribute variables, V_k, may remain. The V_k may not be resolved to a value due to default values for amplitude values that were not specified as explained for figure 4.2. The V_k may also not be resolved to a value due to a cycle in the attribute variable relationships of graph, G_k, as explained for figure 4.2.

The equation formed for an unresolved attribute variable, V_k, is obtained by substituting all the known attribute variable values and amplitude values (at the completion of the graph traversal) into the evolving function equation associated with the attribute variable, V_k, for the context of interpretation. The evolving function equation takes the form of equation (4.3). The known amplitude values are substituted and the chosen generic functions, f_i, as defined in their respective subcontexts are evaluated for a time value t. The substitution of known amplitude and generic function measurement values into an assignment equation for the attribute variable, V_k, results in the linear equation

$$V_k = F_1\kappa_1\alpha_1 + \ldots + F_n\kappa_n\alpha_n + K; \kappa_i \in \mathbb{R}, K \in \mathbb{R}, F_i \in \mathbb{R}. \tag{4.25}$$

$F_1, \ldots, F_n$ are real-valued numbers obtained from the substitution of a time value, t, into the generic functions, $f_1, \ldots, f_n$. $\alpha_1, \ldots, \alpha_n$ are unresolved amplitude values defined within subcontexts or partial contexts. $\kappa_1, \ldots, \kappa_n$ are constant scaling factors defined within subcontexts or partial contexts. K is a real-valued number that represents the sum of terms in equation (4.27) whose amplitude values are known. The value of a term in equation (4.27) with a known amplitude value is calculated from the product of the scaling factors, generic function value, and the amplitude value.

Let

$$c_i = F_i \kappa_i \tag{4.26}$$

then equation (4.25) can be stated more simply as

$$V_k = c_1 \alpha_1 + \ldots c_n \alpha_n + K; c_i \in \mathbb{R}, K \in \mathbb{R}. \tag{4.27}$$

If the evolving function associated with the unresolved attribute variable, V_k, has subcontexts and those subcontexts have related partial contexts then the evolving functions takes the form of equation (4.7). The known amplitude values are substituted and the chosen generic functions, f_i, are evaluated to produce a linear equation of the form

$$V_k = F_1 \kappa_1 \left(\alpha_{1_1} + \ldots + \alpha_{1_k}\right) + \ldots + F_n \kappa_n \left(\alpha_{n_1} + \ldots + \alpha_{n_m}\right) + K; \kappa_i \in \mathbb{R}, K \in \mathbb{R}, F_i \in \mathbb{R} \tag{4.28}$$

which is stated more simply as

$$V_k = c_1 \left(\alpha_{1_1} + \ldots + \alpha_{1_k}\right) + \ldots + c_n \left(\alpha_{n_1} + \ldots + \alpha_{n_m}\right) + K; c_i \in \mathbb{R}, K \in \mathbb{R}. \tag{4.29}$$

The α_{i_j} are unresolved amplitude values expressed in terms of attribute variables as defined within the respective partial contexts (refer to equation (4.7)). All resolved amplitude values are added to the term K.

The equations formed for the unresolved attribute variables combine to form a system of linear equations. The crisp variable value resolution process solves the resulting system of linear equations to obtain the remaining attribute variable measurement values. An equation associated with an attribute variable, V_k, is rewritten to isolate V_k and any other term with an unresolved attribute variable on the left-hand side of the equation. The real-valued term, K, remains on the right-hand side of the equation. The attribute variable V_k within the rewritten form of equation (4.27) is viewed as having a coefficient value $c_0 = 1.0$. An equation associated with an attribute variable, V_k, which has the form of equation (4.29) is distributively multiplied, yielding

$$c_0 V_k - c_1 \alpha_{1_1} + \ldots + c_p \alpha_{n_m} = K; c_i \in \mathbb{R}, K \in \mathbb{R}. \tag{4.30}$$

The coefficients, c_i, of terms on the left hand sides of the rewritten equations associated with the unresolved attribute variables are grouped into a coefficient matrix

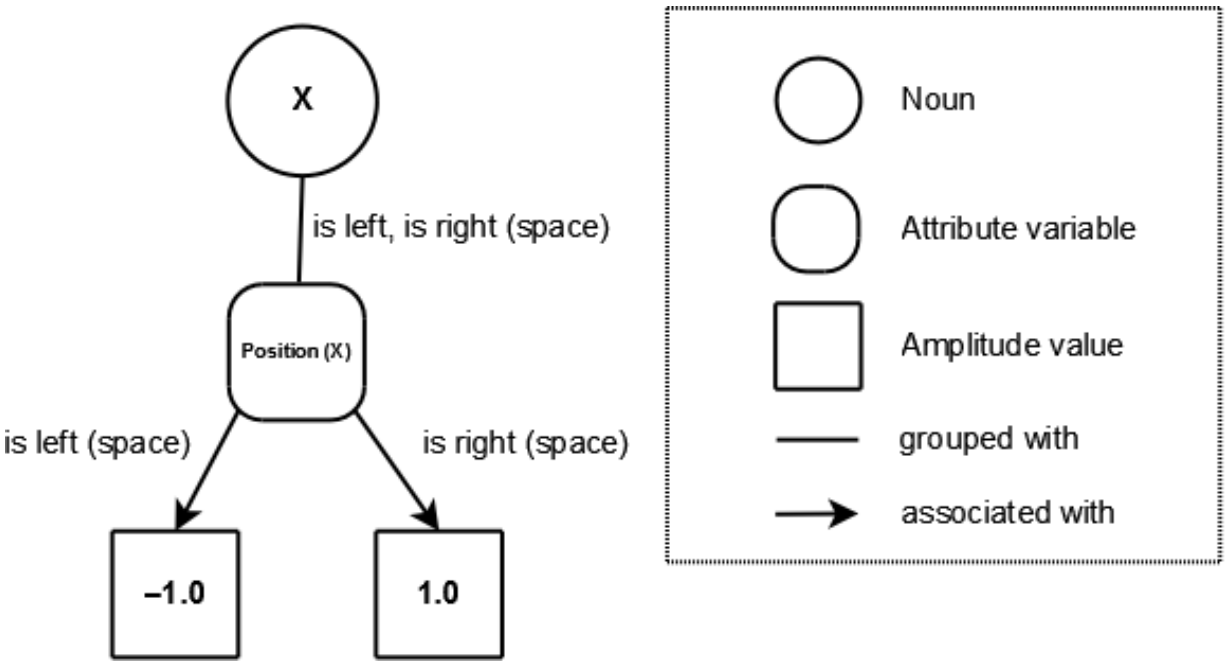

Figure 4.4: Multiple measurement value assignments to the same attribute variable (inconsistency)

A. An augmented matrix, $[A|K]$, is formed by adding the K values on the right-hand side of the equations related to the V_k as a column vector to the coefficient matrix A. The augmented matrix, $[A|K]$, is reduced by the process of Gaussian elimination. If the process of Gaussian elimination reduces $[A|K]$ to upper triangular form, a single unique solution of measurement values for the attribute variables is obtained by the process of back substitution. The values in the right hand column of $[A|K]$ are substituted into equations corresponding to the reduced form of $[A|K]$. The substitution starts at the bottom row and substitutes attribute variable values obtained into the equation corresponding to the row above. The process is repeated until the first row of the reduced augmented matrix, $[A|K]$, is reached and the equation corresponding to the first row is solved. The measurement values of the previously unresolved attribute variables are obtained and the crisp variable resolution procedure is complete.

If the process of Gaussian elimination cannot reduce the augmented matrix, $[A|K]$, to upper triangular form, a single unique solution of attribute variables cannot be found. If $[A|K]$ has more unknown attribute variables than equations, only a general solution of

$[A|K]$ can be found and the attribute variables have an infinite number of measurement value solutions. A linear system of equations with more unknown attribute variables than equations is formed when the evolving function related to an attribute variable is expressed in terms of amplitude values that are equal to unknown attribute variable values. The attribute variable values are unknown because they are either not set by other evolving function models of adjectives and/or verbs or do not have default values specified.

Figure 4.4 shows the association of amplitude values "1.0" and "−1.0" with the attribute variable "Position (X)". The amplitude values correspond to the adjectives "left" and "right" in the context of interpretation "space". Attribute variable "Position (X)" is interpreted as the position of an object on the x-axis of a 3D space. The amplitude values of "−1.0" and "1.0" are interpreted as being left and right of the centre of the x-axis in a 3D space. The chosen generic function, f_i, for the evolving functions that model the adjectives "left" and "right" in the context of interpretation "space", is f_1, as given by equation (4.23). The adjectives "left" and "right" are therefore modelled in terms of static computational verbs.

The crisp variable resolution process constructs and traverses the graph, G_1, to determine the measurement value for the attribute variable "Position (X)". Two evolving functions equations are formed from the substitution of amplitude values "1.0" and "−1.0" respectively. A unique measurement value cannot be obtained by evaluating the constructed evolving function equations a time value, t, since there are two separate evolving functions that result in two measurement value assignments to "Position (Y)". If equations in the form of equation (4.27) are formed, the following system of linear equations results:

$$\begin{aligned} V_{\text{“Position (Y)”}} &= -1 \\ V_{\text{“Position (Y)”}} &= 1. \end{aligned} \tag{4.31}$$

The equations are represented in an augmented matrix in the form of $[A|K]$ as

$$\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$$

.

If the matrix above is reduced by the Gaussian elimination procedure, the inconsistency $0 = 2$ results when the first row is subtracted from the second. The crisp variable resolution procedure can therefore not obtain a unique solution for the attribute variable "Position (Y)". The crisp variable resolution procedure can also not obtain a unique solution in the case of a linear system that has an infinite number of solutions, as discussed for figure 4.3 above. The section that follows details a fuzzy attribute variable resolution procedure. The fuzzy attribute variable resolution procedure determines approximate measurement value solutions for attribute variables if a unique solution cannot be obtained.

4.4.3 Fuzzy procedure for assigning approximate measurement values to attribute variables

This section describes a fuzzy resolution procedure for assigning measurement values to attribute variables. The equations that result from the evaluation of evolving function models for natural language nouns, adjectives, verbs, and adverbs (refer to sections 4.3.1 through 4.3.4) are interpreted as an fuzzy constraint satisfaction problem (FCSP). The formulation of the equations obtained by evaluating evolving function models at a specific time, t, in terms of fuzzy constraints, allows approximate solutions of measurement values to be obtained. Approximate solutions for measurement values are required when inconsistent linear equations result from modelling conflicting natural language statements in terms of the models presented in this work (refer to section 4.4.2). The formulation of conflicting evolving function equations as an FCSP presented in this section, is a novel approach for the determination of approximate values for conflicting evolving function equations in the field of CVT.

Section 4.4.3.1 introduces CSPs and explains how CSPs relate to the evolving function models of this work. Constraint satisfaction resolution algorithms are studied in section 4.4.3.2 to provide insight toward the types of algorithms that may be applied to the resolution of evolving function models formulated in terms of a CSP.

Section 4.4.3.3 introduces the FCSP as a means of relaxing the classic CSPs in cases where a precise and unique solution is not possible. CSP relaxation frameworks are discussed in section 4.4.3.4 to clarify the application of FCSPs within this work. Section

4.4.3.5 specifies the equations that result from the evaluation of the evolving function models of this work in terms of fuzzy constraints. FCSP resolution algorithms are studied in section 4.4.3.6 to provide insight toward the types of algorithms that may be applied in a fuzzy resolution strategy for the determination of measurement values of attribute variables.

The considerations for a robust fuzzy resolution procedure for assigning measurement values to attribute variables are stated in section 4.4.3.7. A genetic algorithm for the resolution of FCSPs is evaluated in terms of the considerations for a fuzzy resolution procedure for assigning measurement values to attribute variables.

Constraint satisfaction optimisation problems are shortly discussed and related to the FCSP approach of this work in section 4.4.3.8.

4.4.3.1 Constraint satisfaction problems

CSPs consider the valid assignment of values to a collection of variables that each have their own domain of valid values. The values of variables are further limited by a series of constraints that create relationships between the variables and the values they may take (refer to definition 3.25). Section 4.3.1 defined attribute variables within the models of this work as storage variables for measurement values obtained from evolving function models for adjectives and verbs (refer to sections 4.3.2 and 4.3.3). Attribute variables also play the role of variable amplitude values within evolving function models in this present work. Natural language sentences modelled according to the evolving function models of this work result in a series of interdependent attribute variable assignments, as shown in section 4.4.2.

An attribute variable is passed as a parameter for the generation of interactive narrative depictions such as computer graphics. This places a constraint on the values that an attribute variable may take for a specific context of interpretation. An example of such a constraint is the domain of $[0, 1]$ placed on colour values in the graphical rendering API, DirectX®. The assignment of measurement values to interdependent attribute variables enforces further constraints on the values that an attribute variable may take. The assignment of measurement values to attribute variables under the constraint of defined evolving functions can consequently be viewed as a CSP.

A study of CSPs and the algorithms applied to solve CSPs is of benefit to this work, as it provides insight into a method for determining correct measurement value assignments for attribute variables.

4.4.3.2 Constraint satisfaction problem resolution

CSPs are widely used to model a great variety of real-world problems [11] [78] [92]. A range of techniques has been proposed over years of research to solve the permutations in which a CSP may occur [52]. Backtracking algorithms are traditionally used to solve CSPs [54]. Such algorithms extend a partial solution to a full solution by continually assigning a new variable to a value in a way that it is consistent with regard to the other variables values assigned and the constraints imposed. A consistent variable assignment is an assignment where the permutation of concurrently assigned variables does not violate any of the problem's constraints. Alternative CSP resolution techniques have been built that utilise computational intelligence approaches such as evolutionary algorithms [20], ant colony optimisations [82] [86], particle swarm optimisations [83], neural networks [39], artificial immune systems [55], and stochastic search [53]. The most prevalent CSP resolution algorithms that utilise computational intelligence are evolutionary algorithms.

Evolutionary algorithms are often applied to solve CSPs, as they provide a global search alternative to the local searches performed by traditional backtracking algorithms and can therefore more easily avoid local optima, if local optima are present in a search space [67]. Evolutionary algorithms such as genetic algorithms can, however, still be trapped in local optima [87]. Evolutionary algorithms provide a means of integrating a global search strategy into a constraint satisfaction solver. Global search algorithms such as evolutionary algorithms are often used in conjunction with local search techniques and heuristics for the resolution of CSPs . A disadvantage of evolutionary algorithms in comparison to local search algorithms is that the former does not perform a complete search over the entirety of the search space [48].

The trade–off between a global search and a complete search drove some evolutionary algorithms applied to CSPs to incorporate classic backtracking techniques. Eiben *et al* [29] utilised local search heuristics in the mutation and crossover steps of a standard genetic algorithm to increase the algorithm's efficiency. Stuckey *et al* [90] applied

the min-conflicts heuristic within the mutation step of a genetic algorithm. The min-conflict's heuristic increases the genetic algorithm's efficiency by reducing the number of constraint checks performed.

Other evolutionary algorithms integrate local search techniques beyond the use of local search heuristics. Hybrid genetic algorithms include local search techniques within the framework of a genetic algorithm. Marchiori *et al* [60] apply a local search algorithm such as a hill climber to a set of candidate solutions as a preprocessing step. Standard genetic algorithm operators are then applied to the population of locally optimised solutions. Bahler *et al* [24] apply a hybrid genetic algorithm to evolve a population of hill climbers. The population of hill climber algorithms then iterates to find an optimal solution using the iterative descent method. Kanoh *et al* [48] reverse the process followed by Bahler *et al*. A population of solutions is evolved by a genetic algorithm. The best solution within the population is then set as the starting point of a min-conflict hill climbing search. Cebrián *et al* [17] follow an approach by which a genetic algorithm, within its genome population, evolves the parameters to a Greedy random adaptive search procedure (GRASP) algorithm. The genes of the optimal solution in the population of the genetic algorithm represent parameters that determine the order in which variables are solved for in the GRASP algorithm. The variables are assigned values from their domain in the order encoded within the fittest chromosome to form a solution.

Other approaches forgo the integration of classic local search procedures for a co-evolutionary approach to solving CSPs. Hasegawa *et al* [47] propose a genetic algorithm that evolves a population of "viruses" in parallel to a solution population. Traditional mutation and crossover reproduction techniques are replaced by reproduction techniques that mimic viral infection. Genes within a chromosome of the solution population are replaced with genes of an individual in the virus population. Baba *et al* [40] follow a similar co-evolutionary approach where a standard genetic algorithm population is evolved in conjunction with a population that represents effective schemata (patterns) within genetic information. The schemata are transcribed onto the chromosomes of the solution population for fitness determination and reproduction purposes. Paredis [71] presents a co-evolutionary approach where solutions and constraints co-evolve in a predator–prey relationship. Individuals from either population are selected for "encounters" in which

individuals from the solution population are awarded points for satisfying the constraint and individuals from the constraint population are rewarded should the constraint not be satisfied. Fitness is calculated by scoring these encounters in a life–time evaluation scheme. A life–time evaluation scheme keeps track of only the most recent encounters for fitness determination. The co-evolutionary approach of Paredis is a self-adaptive scheme that guides the search towards the tougher constraints. Tougher constraints are represented by the fitter individuals in the constraints population, and are therefore selected for encounters more frequently.

Some evolutionary algorithms are fine–tuned to a particular type of CSP. Bahler *et al* [23] have presented a micro–genetic algorithm that has a very small population of less than 20 individuals. The genetic operators applied to the population are feature–rich in their search heuristics. The search heuristics allow the algorithm to utilise fully the search potential of the smaller population. Some evolutionary algorithms are adapted to the CSP's structure. Riff [77] presents an algorithm that analyses the constraint network to adapt the evolutionary algorithm to the constraint network in a way that reduces the number of constraint checks performed. The reduced number of constraint checks increases the efficiency of the search. Marchiori [59] presents a preprocessing approach within the context of constraint specification languages. All constraints presented in a constraint language are processed into a single type of primitive constraint. The uniform simplified constraint allows for a uniform repair operation to be applied as the reproduction operator of an associated evolutionary algorithm. The simplified and uniform constraints increase the efficiency of the algorithm over standard evolutionary algorithms.

Eiben *et al* [19] have presented variations on the step–wise adaptation of weights algorithm (SAW), described in their earlier work. The SAW algorithm calculates a fitness function composed of the weighted sum of the constraint violations or incorrectly instantiated variables of an individual in the population. After a predetermined number of fitness evaluations, the weights of the fitness calculation are adjusted according to the incorrectly instantiated variables or constraint violations in the best individual of the population. The adjustment of weights for fitness calculation creates a self-adapting fitness function that drives the search towards solving the tougher constraints. Chromo-

somes within the population represent variable orderings, as is the case with the hybrid GRASP algorithm [17]. The chromosomes are decoded by assigning valid values from the domains of the associated variables. The variable assignments maintain consistency with other assigned variables and are assigned in the order encoded within the genes of the chromosome.

Craenen *et al* [20] have presented a comprehensive performance comparison of all of the algorithms above, based on an improved random binary CSP generation paradigm. The SAW algorithm is shown to be the most efficient and successful in obtaining solutions for the test suite. None of the genetic algorithms performed as well as a benchmark algorithm using standard CSP solution methods. Eiben *et al* [30] have presented a comparative study of the CCS, MID, and SAW algorithms as representatives of genetic algorithms with self-adapting fitness functions. Once again the SAW algorithm is shown to be the most robust. Cebrián *et al* [17] have shown that the hybrid GA-GRASP algorithm was more efficient than the SAW algorithm for the test suite studied.

The CSPs of this section and their corresponding resolution algorithms are not applied in this present work. An overview of CSPs has been provided because the standard CSP serves as a theoretical basis for the fuzzy CSP discussion that follows.

4.4.3.3 Fuzzy constraint satisfaction problems

CSPs that are based on real-world examples are often inconsistent and have no solution. Inconsistent CSPs may result because conflicting constraints are present within the real-world problem. Inconsistent CSPs are also formed when a problem is over–constrained and no possible value assignment can satisfy all of the constraints. Two main paradigms arose as a means of dealing with inconsistent CSPs.

Partial constraint satisfaction problems (PCSPs) were presented as a means of relaxing an inconsistent CSP to one that does have a solution [33]. An inconsistent CSP can be relaxed with a variety of approaches that range from the removal of constraints to an increase in the domain size of the variables. For each relaxation of the problem, a metric is defined that specifies the distance between the original CSP and the relaxed CSP. Resolution is performed via a standard branch-and-bound algorithm that relaxes constraints if a dead end is reached in the search.

FCSPs (refer to definition 3.26) were introduced as a means of relaxing inconsistent CSPs, by defining fuzzy constraints and then specifying an inconsistent CSP in terms of fuzzy constraints [79]. A fuzzy constraint associates a membership function with a constraint. The value of the membership function indicates the degree to which a permutation of variable values satisfies the associated constraint. FCSPs are a relaxation of the classic CSP because they allow for constraints to be satisfied to a certain degree. The classic CSP allows constraints to be satisfied either fully or not at all. Standard constraint satisfaction heuristics can still be applied to FCSPs and a resolution algorithm with a branch-and-bound approach is proposed. The proposed branch-and-bound algorithm for FCSPs is defined not to follow a search path that yields a variable value solution with a lower degree of membership to a fuzzy constraint than a variable value solution that has already been found.

4.4.3.4 Frameworks for constraint relaxation

A variety of approaches have been adopted to unify the concepts of partial, soft, and standard CSPs in a single framework. Pires *et al* [72] interpret fuzzy constraint satisfaction problems in terms of a finite set of satisfaction degrees that are tied to crisp constraints weighted in terms of their importance value. Guesgen [37] has hybridised the concepts of soft constraints representing preference values for variable value assignments, and as measures of constraint priority [27]. Soft constraint usage is hybridised by viewing soft constraints as certainty–qualified assertions in terms of possibilistic logic. Bistarelli *et al* [9] have proposed a framework by which multiple CSP types, such as classic, fuzzy, weighted, partial, and other similar types of CSP, are viewed in terms of a mathematical semi–ring structure. The set of the semi–ring represents a value associated with every variable value assignment in the CSP. An example of a value association is 0 for an unresolved constraint and 1 for a resolved constraint in the context of a classic CSP. The two operators defined for the semi–ring structure allow constraint combination and projection to be defined. The semi–ring classification of CSPs has been widely adopted by researchers as a generic means of classifying constraint relaxation [12], [54].

The scheme adopted for constraint relaxation in the face of over–constrained or inconsistent problems in the present work is that of fuzzy constraints, as presented in [37]

[79]. Fuzzy constraints are defined as a relaxation of a classic constraint (refer to definitions 3.25 and 3.26) that describes the preference value of one tuple of assigned variable values over another. The preference value is expressed as a real-valued number on the interval $[0, 1]$ and describes the degree to which the tuple of assigned variable values satisfies a given constraint. A preference value of 1 indicates that the constraint is fully satisfied and a preference value of 0 indicates that the constraint is not satisfied at all.

The application of fuzzy constraints in this work is analogous to that of Bahler *et al* [10]. They extend their relation of classic semantics to crisp constraint networks presented in earlier work in order to relate fuzzy semantics to fuzzy constraint networks. A set of interrelated constraints forms a constraint network. Constraints within a crisp constraint network are related to the sentences specified in a predicate language. The variables in the related CSP correspond to constant symbols in the predicate language. The assignment of variables to values within the context of the CSP (refer to definition 3.25) corresponds to the assignment of meaning to constant symbols in a predicate language. The crisp constraints are relaxed by defining constraints as fuzzy constraints with preference values for value assignments. The use of fuzzy constraints then allows fuzzy semantics to be related to fuzzy constraint networks.

The section that follows defines the computational verb models for natural language specified in this work in terms of an FCSP. The relationship between attribute variables and CSP variables is defined. The relationship between constraints in a constraint network and evolving function models is defined. The assignment of variable values in the CSP is related to the assignment of attribute variables to measurement values. The constructed constraints are relaxed in terms of a fuzzy CSP with fuzzy constraints to allow for approximate solutions to the CSP and therefore for approximate measurement value assignments to attribute variables.

4.4.3.5 Evaluated evolving function equations as fuzzy constraints

Constructed evolving functions are evaluated as part of the crisp variable value resolution procedure for assigning measurement values to attribute variables, as presented in section 4.4.2. The equations formed from the evaluation of constructed evolving function equations are denoted by evaluated evolving function equations (EEFEs). The ampli-

tude values and attribute variables that are known after the evaluation of a formed evolving function at a time, t, are treated as constant values. Attribute variables are used either as storage variables or as amplitude values of constructed evolving functions (refer to section 4.3.3). Attribute variables that are unknown after the evaluation of the constructed evolving functions are treated as variables in subsequent measurement value determination steps. The evaluation of the evolving function models of this work at a time, t, therefore results in a system of equations in terms of the unknown attribute variables.

The system of equations that is formed from the EEFEs that still contain unknown attribute variables may be formulated as a CSP. The constraint variables are the unknown attribute variables remaining after the constructed evolving functions have been evaluated. The constraints themselves are the equations in terms of the constant values and the unknown attribute variables after the evaluation of the constructed evolving functions. The set of constraints related to EEFEs that contain unknown attribute variables form a CSP. The goal of the CSP is the assignment of measurement values to the attribute variables whose values are unknown in a consistent way. A consistent assignment of measurement values to attribute variables ensures that none of the constraints are violated. Each constraint corresponds to a single equation and therefore a violation occurs if the left-hand side of an EEFE is not equal to the right-hand side. The values on the left-hand side and the right-hand side of the EEFE are compared when a tuple of candidate measurement values is substituted into the corresponding unknown attribute variables present within the EEFE.

The CSP presented above may be formulated as an FCSP if the constraints are converted to fuzzy constraints (refer to definition 3.26). The challenge with modelling the system of equations obtained from EEFEs as an FCSP lies in formulating the equations that contain attribute variables whose values are unknown as membership functions of fuzzy constraints. The membership function of a fuzzy constraint assigns to a tuple of candidate measurement value assignments for attribute variables, a real-valued number on the interval $[0, 1]$. A membership value is calculated by the membership function. The membership value indicates how well a tuple of candidate measurement values assigned to the unknown attribute variables present in an EEFE satisfies the fuzzy constraint

associated with the EEFE.

One approach to solving the system of equations formed by a set of EEFEs that contain unknown attribute variables is to formulate the solution of the system of equations as an optimisation problem. The optimisation algorithm chosen to solve the optimisation problem would attempt to minimise the sum of absolute error values for all the equations that contain unknown attribute variables. An optimisation approach would, however, favour highly those equations that contain unknown attribute variables with larger domains and have larger coefficient values (refer to equation (4.27)). The majority of solutions obtained by an optimisation approach would therefore be skewed towards local optima. The local optima would be the solutions that minimise the error value of those equations that contain attribute variables with the largest domain size and equations that have the largest coefficient values. The attribute variables within the evolving function models of natural language sentences presented in this work do not have a uniform size. Solutions that favour local optima can be avoided by adjusting the error values calculated for an equation according to the domain size of the variables present and by the size of the coefficient values.

The local optima problem of an optimisation approach to solving EEFEs that contain unknown attribute variables is addressed to a large degree by solving the EEFEs that contain unknown attribute variables as an FCSP. Every fuzzy constraint in an FCSP has a degree of satisfaction represented on the uniform scale of a real number on the interval $[0, 1]$. If every EEFE that contains unknown attribute variables is formulated as a fuzzy constraint, the assignment of measurement values to the unknown attribute variables present within the EEFEs can be solved as an FCSP. The membership values of the fuzzy constraints are measured as real-valued numbers on the interval $[0, 1]$. The fuzzy constraints and their associated EEFEs are therefore treated uniformly and the bias towards EEFEs with larger coefficients and domain values is reduced.

The formulation of the assignment of measurement values to attribute variables as an FCSP requires the construction of membership functions for the fuzzy constraints. This work formulates the membership function of fuzzy constraints by calculating an error value. The error value represents the difference between the value calculated for an EEFE when a tuple of candidate measurement values is substituted, and the expected value of

the EEFE at a time, t. The error value is scaled according to the maximum error value possible when candidate measurement values are substituted into the unknown attribute variables of the EEFEs that correspond to the fuzzy constraints. The value used to scale a calculated error value is called the "scaling factor". The scaling factor maps an error value to a real-valued number on the interval $[0, 1]$.

The substitution of a measurement value tuple, $\overline{x}$, into an EEFE rewritten in the form of equation (4.30) yields

$$f(\overline{x}) + \epsilon = K. \tag{4.32}$$

$f(\overline{x})$ represents the substitution of the candidate measurement value tuple, $\overline{x}$, into the corresponding unknown attribute variables on the left-hand side of an EEFE that is rewritten in the form of equation (4.30). The left-hand side of the EEFE in the form of equation (4.30) is represented by the generic function, f. The error value, ϵ, represents the difference between f and the value, K, when the candidate measurement value tuple, $\overline{x}$, is substituted into an EEFE rewritten in the form of equation (4.30).

The isolation of the error value, ϵ, on the right-hand side of equation (4.32) yields

$$f(\overline{x}) - K = -\epsilon. \tag{4.33}$$

The absolute error value, $|\epsilon|$, is given by

$$|\epsilon| = f(\overline{x}) - K. \tag{4.34}$$

The calculated absolute error value, $|\epsilon|$, is scaled according to the calculated scaling factor, ε. The membership function, μ_C, of the fuzzy constraint, C, is therefore given by

$$\mu_C(\overline{x}) = \frac{|f(\overline{x}) - K|}{\varepsilon}. \tag{4.35}$$

The membership function, μ_C, calculates the degree to which the fuzzy constraint, C, is satisfied. The fuzzy constraint, C, corresponds to an EEFE that contains at least one unknown attribute variable.

The scaling factor, ε, is calculated by extending the evolving function models for natural language verbs presented in section 4.3.3. The evolving function models are extended to specify minimum, α^-, and maximum, α^+, values for the amplitude values, α_i, specified within an evolving function equation. The minimum, α^-, and maximum,

α^+, values for an amplitude value, α, are specified when the amplitude value is defined within a context of interpretation or subcontext. The minimum, $\alpha_{i_j}^-$, and maximum, $\alpha_{i_j}^+$, values for a partial amplitude value, α_{i_j}, are specified in a partial context. If a constant or a partial amplitude value is specified, the minimum and maximum values for the amplitude value should be defined as numbers equal to the constant amplitude value or partial amplitude value. If an attribute variable, V_i, is used as an amplitude value or a partial amplitude value, the minimum and maximum amplitude values or partial amplitude values specified define a domain for the attribute variable, V_i. The domain of an attribute variable used as an amplitude value in a context of interpretation, subcontext, or partial context is given by $V_i \in [\alpha^-, \alpha^+]$, $V_i \in [\alpha_i^-, \alpha_i^+]$, and $V_i \in [\alpha_{i_j}^-, \alpha^+ i_j]$ respectively. If no minimum and maximum values are specified for an amplitude or partial amplitude value, default minimum and maximum amplitude values are used. The default minimum and maximum amplitude values are the same for all evolving function models specified.

The minimum size, d^-, and the maximum size, d^+, of an EEFE in the form of equation (4.29) are calculated by the equations:

$$d^- = \left(\sum_{i=1}^{l} c_i \mathtt{d}_i^- \right) + K \tag{4.36}$$

$$d^+ = \left(\sum_{i=1}^{m} c_i \mathtt{d}_i^+ \right) + K \tag{4.37}$$

where K is the constant term on the right-hand side of an EEFE in the form of equation (4.29); $\mathtt{d}_i^-$ and $\mathtt{d}_i^+$ are the defined minimum and maximum values for an amplitude value, α_i^- and α_i^+, or partial amplitude value, $\alpha^- i_j$ and $\alpha^+ i_j$, on the right-hand side of an EEFE in the form of equation (4.29). The c_i are coefficient values on the right-hand side of an EEFE in the form of equation (4.29); they are calculated as the product of the value of a generic function, f_i, for a time value, t, that is $F_i = f_i(t)$, and a scaling value, κ (refer to equation (4.26)).

If the attribute variable, V_k, is defined as the storage variable (refer to section 4.3.3) for the evolving function that corresponds to an EEFE, the calculated minimum, d^-, and maximum, d^+, values of the EEFE define the domain of the attribute variable V_k. The domain of V_k is therefore given by $V_k \in [d^-, d^+]$.

The domain of an unknown attribute variable, V_i, in the context of a single EEFE that contains unknown amplitude values, is therefore calculated by equations (4.36) and (4.37) or defined as the values $\mathtt{d}_i^-$ and $\mathtt{d}_i^+$, if the attribute variable is used as a storage variable or an amplitude value, respectively. The attribute variable, V_i, may, however, be used in multiple EEFEs that contain unknown amplitude variables. The attribute variable, V_i, may also be used multiple times within a single EEFE that contains unknown amplitude values. The global domain for the attribute variable, V_i, is therefore required when the scaling factor, ε, of the error value, ϵ, is calculated.

This present work defines the global minimum value for an attribute variable, V_i, as

$$d_{V_i}^- = \min \mathcal{D}_{V_i}^-. \tag{4.38}$$

$d_{V_i}^-$ is the global minimum value for the attribute variable, V_i. $\mathcal{D}^-$ is an ordered set of all minimum values defined or calculated (refer to equations (4.36) and (4.37)) for the attribute variable, V_i. The min operation returns the smallest member of an ordered set. The global maximum value for an attribute variable, V_i, is defined as

$$d_{V_i}^+ = \max \mathcal{D}_{V_i}^+. \tag{4.39}$$

$d_{V_i}^+$ is the global maximum value for the attribute variable, V_i. $\mathcal{D}_{V_i}^+$ is an ordered set of all the maximum values defined or calculated for attribute variable, V_i. The max operation returns the largest member of an ordered set. The global domain of attribute variable, V_i, is therefore given by $V_i \in [d_{V_i}^-, d_{V_i}^+]$.

The candidate measurement values for unknown attribute variables in an EEFE are selected from the global domains determined for the respective attribute variables. The attribute variable, V_0, for example, is used as the storage variable of an EEFE. The attribute variables, $V_1, \ldots, V_n$, are used as amplitude values. The values of all attribute variables, $V_0, \ldots, V_n$, are unknown. If candidate measurement values are selected from the global domains of attribute variable, $V_0, \ldots, V_n$, and substituted into an EEFE in the form of equation (4.30), an error value, ϵ, is obtained (refer to equation (4.32)). The boundaries of the error value, ϵ, are obtained when the maximum value for the evolving function that corresponds to the EEFE is substituted as the value of the storage variable, V_0, while the minimum values from the global domains of the attribute variables,

$V_1, \ldots, V_n$, are substituted as amplitude values and vice versa. The boundary value, d_ϵ^+, is obtained for the maximum global storage variable value and minimum amplitude values and is given by

$$d_\epsilon^+ = d_{V_0}^+ - \left(\sum_{i=1}^{n} c_i d_{V_i}^- \right) - K. \tag{4.40}$$

The boundary value, d_ϵ^-, is obtained for the minimum global storage variable value and minimum global amplitude values and is given by

$$d_\epsilon^- = d_{V_0}^- - \left(\sum_{i=1}^{n} c_i d_{V_i}^+ \right) - K. \tag{4.41}$$

The scaling factor, ε, for the absolute error value, $|\epsilon|$, is simply calculated as

$$\varepsilon = |d_\epsilon^+ - d_\epsilon^-|. \tag{4.42}$$

The absolute error value, $|\epsilon|$, and the scaling factor, ε, are used to calculate the membership function value, $\mu(\overline{x})$, of a set of candidate measurement values, $\overline{x}$, to the fuzzy constraint, C (refer to equation (4.35)).

The accuracy of the scaling factor depends entirely on the maximum and minimum values specified for the amplitude values of an EEFE. All fuzzy constraint membership values are bounded on the interval $[0, 1]$. If the defined minimum and maximum amplitude values are not accurate, the calculated scaling factor, ε, of the error value, ϵ, cannot be calculated accurately. The membership values are calculated according to the absolute error value, $|\epsilon|$, scaled by the scaling factor, ε (refer to equation (4.35)). The membership values of value tuples to fuzzy constraints are consequently skewed if incorrect amplitude value domain sizes are specified. A set of EEFEs that contain unknown attribute variables forms a system of linear equations. The accuracy of a solution determined for the formed system of linear equations therefore depends on the accuracy of the defined amplitude value domain sizes.

The mapping scheme (specified in this section) between an EEFE and the membership function of a fuzzy constraint allows the system of equations formed from a set of EEFEs that contain unknown attribute variables to be formulated as an FCSP.

The section that follows discusses a series of techniques that have been applied to solving FCSPs. The FCSP resolution methods presented can be applied to solving a

system of equations formulated as an FCSP and are therefore of value to the present work.

4.4.3.6 Fuzzy constraint satisfaction problem resolution

Ruttkay [79] generalised classic CSP resolution techniques such as heuristic search to solve FCSPs. The heuristic search adopted to an FCSP is then replaced by a more efficient branch-and-bound search. Guesgen proposes a set of consistency algorithms for the resolution of FCSPs [37] [38]. An arc consistency algorithm is implemented as an iterative version of the forward-checking algorithm. The arc consistency algorithm shows an improvement over the fuzzy version of the AC3 arc consistency algorithm proposed by Dubois *et al* [26].

Hsu *et al* [43] have proposed an FCSP resolution algorithm based on adaptive level cuts. The algorithm attempts to reduce the amount of work done in the search by applying repair operations to a constraint. If the repair algorithm cannot find a solution to the current constraint which has a higher membership degree than the current alpha level-cut, the algorithm moves on to the next level-cut. An alpha level-cut refers to all fuzzy constraints with membership degrees that are higher than a defined constant [128]. The algorithm returns a solution once a level-cut has been found that supports all constraints to the degree proposed by an alpha level-cut. Candidate solutions are determined by a repair operation. Miguel *et al* [66] propose another repair approach in terms of a fuzzified version of the local changes algorithm [80]. The local changes algorithm allows for the solution of dynamic CSPs where constraints are added and removed over time. The fuzzy local changes algorithm divides variables into three groups (assigned and fixed, assigned and not fixed, unassigned). The fuzzy local changes algorithm then assigns the unassigned variables and repairs the non-fixed assigned variables to a previously achieved consistency level. The fuzzy local changes algorithm terminates when all variables have been assigned and the consistency has been maximised.

Kowalczyk [51] has proposed a genetic algorithm for solving FCSPs. The genetic algorithm performs an unconstrained optimisation of an objective function based on the resolution of fuzzy constraints. The algorithm uses a genetic algorithm's ability to optimise functions, by formulating an objective function that maximises the min-aggregation

of fuzzy constraint membership degrees (smallest membership degree in the set of fuzzy constraints). Each chromosome represents a tuple of candidate value assignments to the constraint satisfaction problem's variables. The candidate values are then substituted into the membership functions of fuzzy constraints to determine the membership degree of the value assignment to the fuzzy constraint. The fitness of the chromosome is then determined by the min-aggregation of the membership degrees and returns the membership degree of the least satisfied constraint as a single fitness value. The standard genetic algorithm operations of mutation and crossover are applied to chromosomes for reproduction. Each chromosome represents a candidate solution. The standard selection pressure mechanics of parent selection and elitism based on fitness levels are applied. The selection pressures ensure that only the fittest individuals, and therefore the most correct variable value assignments, are carried over to the next generation of the chromosome population.

4.4.3.7 Considerations for an FCSP resolution algorithm applied to the solution approximation of evaluated evolving function equations

Three vital requirements should be considered when choosing a resolution algorithm for the FCSPs formulated for EEFEs that contain unknown attribute variables:

- The first requirement for a resolution algorithm is that it is able to handle constraints of multiple variables. The constraints of the FCSP presented in section 4.4.3.5 are each related to an EEFE, which may contain multiple unknown attribute variables (refer to section 4.4.2). It has been shown that all CSPs can be rewritten into binary CSPs [7]. An algorithm based on binary restrictions involving two variables would therefore be limited only in terms of the overhead required in transforming a multiple variable CSP to a binary CSP.

- The second and more crucial requirement is the ability of the algorithm to operate on variables that have an infinite domain. Classic constraint satisfaction heuristics and algorithms, such as backtracking, are designed to operate on variables with a discrete domain. Algorithms that operate on variables with a discrete domain typically measure fitness in terms of the number of constraint violations. Algo-

rithms that operate on variables with a discrete domain also select values from the discrete domains of the constraint variables for candidate solutions.

If algorithms that typically operate on discrete variables are adopted to an FCSP, a metric is developed to calculate partial membership. The variable selection process remains a problem, however, as the algorithm would randomly select values from a continuous domain. The variable selection problem also occurs for genetic algorithms such as SAW [19] and the hybrid GA-GRASP algorithm [17], which evolve parameters that dictate how variable value selection from the corresponding discrete variable domains takes place.

The algorithm chosen for resolving the FCSPs in the present work should be able to represent and handle variables of continuous domain easily. Optimisation algorithms are better suited to CSPs that have variables of continuous domain, because error values are calculated in terms of distance metrics between a candidate solution and an optimal solution.

- A third requirement that results from the construction of evolving function models for natural language words is an ability to handle dynamic CSPs. Dynamic CSPs are produced when validity periods are attached to the evolving function models of natural language words. As a result, the evolving function model for a natural language word may vary over its defined time intervals and therefore alter the EEFEs produced. Altered EEFEs require changes to be made in the FCSP model produced to resolve the system of equations that corresponds to a set of EEFEs. A dynamic FCSP is produced as a result.

Schiex *et al* [81] propose an approach to solving dynamic CSPs that approximates the search space. The search algorithm approximates the search space by recording nogoods as it executes. Nogoods are value tuples that can never be altered to a consistent assignment of values for the variables of the current CSP (which is determined by the current set of constraints). If the search is repeated after constraints are added or removed, the search can ignore the recorded nogood values. Miguel *et al* [66] have proposed a fuzzy version of the local changes algorithm, as discussed in section 4.4.3.4. The stated approaches are not applied within the context of this work. The stated algorithms may,

however, form part of future work in adapting FCSP resolution algorithms to dynamic changes in the computational verb models of this work.

Based on the considerations listed above, this work adopts the approach of Kowalczyk [51], who applied a genetic algorithm to solving FCSPs, as discussed in section 4.4.3.4. The proposed genetic algorithm treats an FCSP as an optimisation problem. Genetic algorithms are well suited to optimisation problems and therefore present an efficient approach to solving FCSPs that are formulated as optimisation problems [35] [74]. Genetic algorithms are more generalised than classic CSP resolution algorithms such as backtracking; as a result, they can accommodate a wide range of variable types. Variables of infinite domain are easily encoded in a genetic algorithm as a chromosome with real-valued genes.

Optimisation algorithms such as particle swarm optimisation algorithms [28] are also effective at optimisation and are therefore able to handle multiple variables with continuous domains. A comparison between genetic and other optimisation algorithms is required that compares the performance of genetic algorithms to other optimisation algorithms in the context of FCSPs; however, such a comparison falls beyond the scope of this work.

The genetic algorithm of Kowalczyk retains the FCSP formulation of its search space. The pitfall of a solution skew to value tuples that produce optimal solutions for equations which have larger error values (refer to section 4.4.3.5) is therefore avoided. An FCSP approach handles all constraints and their associated equations in a uniform way. An FCSP formulation also allows for the prioritisation of constraints, which in turn enables an FCSP resolution algorithm to relax constraints in situations where the enforcement of a constraint prevents the FCSP resolution algorithm from obtaining a consistent solution for the remaining constraints.

4.4.3.8 Constraint satisfaction optimisation problems

A final consideration for the resolution of the system of equations produced by a set of EEFEs is to solve the system of equations in terms of a constraint satisfaction optimisation problem (CSOP) framework [91]. Constraint satisfaction optimisation problems extend the standard CSP by optimising an objective function defined over the constraint

variables. The optimised objective function is typically expressed in the form of either a weighted function or a cost function. CSOPs formulate constraints as either equality or inequality constraints. Equality constraints are a good fit for EEFEs.

EEFEs and the fuzzy constraints associated with them are formed dynamically, however, and no overarching objective is defined for the models of this work. The resolution of the system of equations related to a set of EEFEs can therefore not be adopted to a CSOP. The formulation of such a function in the context of a wider interpretation of the events described in a sentence — for example, in the context of a paragraph or story arc as described in [97] [98] [99] — is definitely useful. A CSOP approach to resolving the measurement values of attribute variables can also be applied to driving a specific interpretation of a set of natural language sentences. The interpretation may be determined by a greater context such as an overarching storyline. Alternatively, a set of natural language sentences can be interpreted in terms of an objective function, which is then specified as a weighted sum of attribute variables. No objective function has been formulated in this work, however, and the fuzzy resolution procedure for determining measurement values for attribute variables is formulated in terms of an FCSP instead.

4.4.3.9 Fuzzy variable resolution summary

A theoretical investigation of CSPs, FCSPs, and the resolution algorithms for CSPs and FCSPs was presented in the subsection above. Soft constraint hierarchies were discussed to illustrate how FCSPs are applied within this work. Fuzzy constraints were related to the equations that result when evolving function equations are evaluated (EEFEs) as part of the crisp variable resolution procedure (refer to section 4.4.2). A fuzzy variable resolution approach formulated in terms of an FCSP is applied to solving the system of equations related to a set of EEFEs. The fuzzy variable resolution procedure is applied in the case where the crisp variable resolution procedure shows the system of equations to be inconsistent. The fuzzy variable resolution approach presented in this subsection allows approximate solutions to be determined for the measurement values assigned to attribute variables. Approximate measurement values are required when conflicting natural language statements are modelled and the constructed evolving function equations prove to be inconsistent. The considerations regarding a robust FCSP resolution algo-

rithm in the context of this work were presented. A genetic algorithm for the resolution of FCSPs as presented by Kowalczyk [51] was chosen as the resolution procedure for FCSPs related to the system of equations produced by a set of EEFEs. Constraint satisfaction optimisation problems were investigated as an alternative means of solving the system of equations produced by a set of EEFEs.

4.4.4 Summary of variable resolution procedures

This section discussed the variable resolution procedures applied in this work. The variable resolution procedures have the objective of assigning measurement values to unknown attribute variables in the evolving function models constructed for natural language words in the scope of this work. The considerations for a variable resolution procedure were presented as the formation of evolving function equations, the assignment of interactive attribute variables, and the ability to assign accurate measurement values in the face of inconsistent evaluated evolving function equations. A crisp variable resolution was presented as a series of natural language examples converted to evolving function models. The crisp variable resolution procedure succeeds with regard to the first two considerations but is not able to handle inconsistent EEFEs. A fuzzy variable resolution procedure was proposed to formulate EEFEs in terms of fuzzy constraints and the resolution of the system of equations produced by a set of EEFEs as an FCSP. Resolution algorithms for FCSPs were studied and a genetic algorithm adapted for the resolution of FCSPs was chosen as the resolution method for FCSPs in this work.

4.5 Chapter summary

This chapter presented the computational verb and the CSP principles used to model natural language words within the scope of this work. A generic model for evolving functions was specified as a base for the computational verb models of adjectives and verbs. The concept of a context of interpretation was defined as applied within this work as a means of organising computational verb models for natural language words. A context of interpretation was subdivided into subcontexts and partial contexts in relation to the generic computational verb model defined. Nouns, adjectives, verbs, and adverbs

were defined in terms of fuzzy set theory, CVT, and the models in this work. Nouns, adjectives, verbs, and adverbs were specified in terms of the generic evolving function model. The evolving function models for nouns, adjectives, verbs, and adverbs were organised according to contexts of interpretation, subcontexts, and partial contexts. A crisp variable resolution procedure was presented with which to determine measurement values for attribute variables as defined within the models in this work. A fuzzy variable resolution procedure was investigated and formulated in terms of an FCSP. A genetic algorithm was chosen as the FCSP resolution procedure applied in this work.

The models presented in this chapter serve as the theoretical basis of the system for generating interactive narrative from natural language sentences by means of computational verb models. The system is discussed in detail in chapter 5. Chapter 5 also references this chapter regarding the application of the computational verb models defined in this chapter, where they are applicable to the interactive narrative generation system presented.

Chapter 5

The architecture of the computational verb theory interactive narrative generation system

This chapter describes the architecture of a computational verb theory interactive narrative generation (CVTNG) system in terms of its design principles, components, implementation, and usage possibilities. The challenge of generating interactive narrative from natural language sentences is stated in section 5.1 and is followed by a statement of design principles for the CVTNG system in section 5.2. The design principles provide a focus for the discussion of the architectural features of the CVTNG system.

The architectural features presented in this chapter are stated in terms of the computational verb-based word models of chapter 4. In section 5.3.2 the architectural features related to the creation of contexts of interpretation (refer to section 4.2) are discussed as a means of organising computational verb-based word models. The definition of computational verb-based word models for the supported word types in the CVTNG system (refer to sections 4.3.2 through 4.3.4) is discussed in section 5.4. In section 5.5 a parsing framework and algorithm are discussed as a means of translating natural language provided as a text input to predefined computational verb models within a defined context

of interpretation. The variable resolution procedures for assigning measurement values to attribute variables (refer to sections 4.4.2 and 4.4.3) are presented in section 5.6 in terms of the algorithms applied and the architectural features present in the CVTNG system. A representation subsystem is discussed in section 5.4.6 that serves to integrate the parsing and variable resolution subsystems of the CVTNG system with external representational media.

5.1 Problem statement

The problems studied in computational linguistics are inherent to the nature of natural language. Natural language is a knowledge representation medium that is implicit, inconsistent, and ambiguous [120] [125]:

- Natural language is implicit because it relies on collective human knowledge for correct interpretation. The implicit nature of natural language as a knowledge representation medium is difficult to encode in binary computers. Computational verb theory (CVT) seeks to address the complexities of processing natural language, by modelling natural language words and sentences in terms of the dynamics related to the natural language words and sentences.

- Natural language can be inconsistent because multiple definitions can be attributed to a single word or sentence. A sentence can also be structured in multiple ways and still have the same meaning. Multiple exceptions also exist within the rules of spelling and grammar. Language parsers need to be robust in the sense that they can handle all of the possible exceptions and variants in natural language in a generic and adaptable way.

- Natural language is ambiguous because multiple interpretations can be ascribed to a single word or sentence. Context is required to resolve the ambiguities of natural language and determine the true meaning of a natural language unit such as a word, a sentence or a paragraph. Natural language needs to be modelled in a way that allows multiple interpretations of a language unit yet with the correct interpretation determined by the current context.

A system that generates interactive narrative from natural language should also address the challenges posed by narrative, and especially interactive narrative. An interactive narrative generation system needs to identify the agents and areas in play. It should also detect the changes that occur in the agents and areas as time progresses. The CVTNG system therefore needs to allow changes over time in its narrative depictions. The CVTNG system is therefore parameterised by time. Interactive narrative adds the challenge of dynamic change, as introduced by human participants [75]. Consequently, the interactive narrative system needs to adapt the narrative depictions at interactive speeds. The CVTNG system has to be efficient and responsive in terms of its calculations to allow real-time feedback to user actions. Narrative text relies on the established contexts and relationships set up by preceding narrative for correct interpretation [97] [98] [99]. The CVTNG system has to keep track of such relationships, contexts, and statuses in a consistent and re-usable way.

5.2 Design principles of the CVTNG system

This section describes the design principles of the CVTNG system in view of the challenges that arise from creating interactive narrative from parsed natural language sentences. The design principles are to:

- encapsulate the complexities of modelling natural language and interactive narrative representation;
- represent interactive narrative depictions in a correct and consistent way;
- separate the concerns of the different users in the interactive narrative generation system;
- re-use artifacts such as word models and representations created inside or outside the system.

The subsections that follow, each address one of the design principles above and relate the design principles to the problem statement in section 5.1.

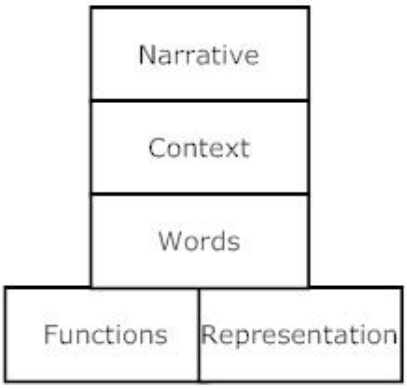

Figure 5.1: Levels of interaction with the CVTNG system

5.2.1 Encapsulation of the complexities in word models and interactive narrative generation

Figure 5.1 represents the approach the CVTNG system adopts to leverage and encapsulate the complexities of computational verb-based word models and narrative space generation.

The highest level of interaction with the system is the creation of textual narrative. A user who interacts with the system at the narrative text level requires only a working knowledge of natural language in order to realise interactive narrative spaces. The creation of narrative text is the most accessible and natural level of modelling in the CVTNG system: the majority of users are able to interact with the system at the level of interactive narrative.

The second level of interaction in the CVTNG system comprises the actions involved in building contexts of interpretation; they enable users to stipulate the contexts that narrative text is interpreted in. Context of interpretation is specified by users who know about the variables related to the observation of narrative text in a particular context of interpretation. Creators of contexts of interpretation also need to know about the relationship between observation variables and the parameters supplied to narrative representations. Narrative representations depict the narrative text for a specific context of interpretation within the generated interactive narrative space.

The third level of interaction in the CVTNG system is the creation of the word models themselves. Users who create word models define the computational verb models of the supported word types (refer to sections 4.3.1 through 4.3.4) within a context of interpretation. The creators of word models also stipulate the representation of nouns in terms of external media such as graphics, sound files, and artificial intelligence (AI) behaviours. The creators of word models should also know about evolving function models as presented in chapter 4.

The fourth and lowest level of interaction with the CVTNG system involves the construction of narrative depictions and the definition of mathematical functions. Narrative depictions are constructed in the form of computer graphics, sound files, and AI behaviours. Mathematical functions are used in evolving functions (refer to section 4.1) and transformation functions (refer to section 4.3.4). Users who construct narrative depictions need to have a thorough knowledge of the representational means such as computer graphics models, sound files, and AI scripts. Users who define mathematical models need to have a knowledge of dynamic systems, functions, and computational verb theory (CVT). The number of users who interact with the CVTNG system at this level is significantly smaller than the number of users who will engage it at the higher levels shown in figure 5.1. The reason for the lower number of users at the fourth level is the specialised mathematical and narrative representation knowledge required.

5.2.2 Correct and consistent representation of narrative depictions

The second design principle, and a high-priority one, is the accurate and consistent representation of parsed narrative. The CVTNG system has to identify correctly the narrative depictions that constitute the narrative space, the statuses of narrative depictions, and the relationships between narrative depictions. The system must then allow changes in the states and relationships of narrative depictions according to the narrative flow. The narrative flow is provided in the form of narrative text. The representation of the narrative space must remain consistent and continuous while changes in the narrative depictions that indicate narrative flow are applied. The narrative depictions must remain consistent in the face of changed inputs by the creator of the interactive narrative

space and the interactive narrative participants.

5.2.3 Separation of system user concerns

The third design principle of the CVTNG system is to decouple its subsystems in terms of the system architecture and user interaction. A creator of word models needs to have a knowledge of the computational verb models presented in section 4.1, but should not be required to define functions or build narrative representations in any way. Someone who is interested in constructing interactive narrative should not be confronted with the creation of computational verb-based word models and contexts of interpretation. A black box approach is followed throughout the architecture, meaning that every subsystem is encapsulated but is still able to interact freely with the other subsystems, in line with the overall CVTNG system design principles.

5.2.4 Re-usability of artifacts

All artifacts created within a subsystem such as a context of interpretation, an evolving function word model, or narrative depictions should be decoupled from other subsystems and therefore be fully interchangeable. A single mathematical function can therefore be used in multiple evolving function equations. A context of interpretation may be used for multiple evolving function word models. A graphical model may be associated with noun word definitions in multiple contexts. A unit of narrative text can be interpreted according to multiple alternative word definitions and contexts of interpretation. The examples listed should be possible within a system devised for interactive narrative generation from natural language in order to realise its modelling potential fully.

5.3 Context of interpretation architecture

This section presents the implementation of architectural features in the CVTNG system geared towards implementing the concept of a context of interpretation as presented in section 4.2. Section 5.3.1 sets out the implementation of contexts of interpretation (refer to section 4.2) within the architecture of the CVTNG system. The architectural

features of the CVTNG system by which contexts of interpretation are created and stored are presented in section 5.3.2. Variable sets and transformations are discussed as architectural features that act as interfaces between CVTNG subsystems in sections 5.3.3 and 5.3.4 respectively.

5.3.1 Application of a context of interpretation in the CVTNG architecture

Context serves as a means to organise and interpret natural language sentences. The CVTNG system uses context to group the separate meanings that may be associated with a natural language sentence, and then associates the grouped meanings with narrative depictions. The meanings of the sentences are modelled in terms of CVT. External media such as graphics, sound, and AI scripts are used as narrative depictions. The sentence fragment "a bad apple" can be visualised as a rotten apple if the context is that of a visual description. But the phrase "a bad apple" can also be interpreted as a person with a negative influence on others in the context of a character description. A context provides a point of view, a frame of mind, and a window through which to look at a natural language sentence and then attach a meaning to it.

The notion of context is central to the natural language models of the CVTNG system. Context of interpretation serves as a means of grouping narrative depictions with computational verb models that determine the parameters for their representation. Computational verb models are also subdivided in terms of their context of interpretation, subcontexts of a context of interpretation, and the partial contexts related to a subcontext. The subdivision of computational verb models in terms of context of interpretation allows the CVTNG system to break complex phenomena down into their constituent parts. The constituent parts of a computational verb model can also be re-used in other computational verb models.

Context of interpretation as a grouping mechanism within the CVTNG system allows the CVTNG system to separate usage concerns by grouping narrative depictions with computational verb word models (refer to section 5.2.3). The complexities of computational verb models are leveraged by subdividing evolving functions in terms of subcontexts and partial contexts (refer to section 5.2.1). In this way, context of interpretation

enables the CVTNG system to adhere to two of its main design principles.

5.3.2 Representation of context of interpretation within the CVTNG architecture

The CVTNG system stores contexts of interpretation as simple XML structures that contain tags for the attributes that describe a context of interpretation. XML was chosen as the storage format for contexts of interpretation for the following four reasons [1]:

- XML is a standardised and open format that is frequently used for communication between dissimilar systems, as is the case for the CVTNG subsystems.
- The XML structures are self-descriptive and word-based, and therefore align with the philosophy of the CVTNG system of modelling by words.
- XML structures allow for complex and nested structures such as hierarchies that are frequently used in the CVTNG system due to the subdivision of computational verb models in terms of contexts of interpretation, subcontexts, and partial contexts.
- XML structures can easily be modified to allow for an additive design approach to context and word model creation in the CVTNG system. New contexts can easily be added to existing word models and new word models can easily be specified for an existing context of interpretation.

XML as a storage format allows the CVTNG system to achieve the design goals of encapsulating and leveraging the complexities of computational verb models and narrative depictions (refer to section 5.2.1). XML also allows the re-use of artifacts created within the CVTNG subsystems due to its open, human-readable, and reinterpretable nature (refer to section 5.2.4).

The XML structure and the defined attributes of a CVTNG context of interpretation are:

- the name of the context,

- the name of the context this context forms part of in the case of a subcontext or a partial context,
- nested tags containing the properties defined for the context,
- and flags that indicate whether the context is a subcontext or a partial context.

A property is a structure within the CVTNG system that groups three system features:

- an attribute variable (refer to section 4.2);
- a transformation that describes how the attribute variable is applied to a narrative depiction;
- a default value for the attribute variable within the specified context of interpretation.

Attribute variables as defined within the computational verb models of chapter 4 are generalised to properties within the CVTNG architecture. This generalisation enables an attribute variable to be applied to a representational means in a variety of ways using different transformations. Each transformation specified within a property has a unique default value that corresponds to the transformation. For example, the words "skew" and "move" act on the vertices that constitute a graphical model when represented visually [56]. The effect of their respective computational verb models is, however, applied to the graphical model in different ways. The parameter values used to transform narrative depictions are therefore decoupled from the attribute variables of the computational verb models and can be applied in different ways to the narrative depictions. The use of properties and generic transformations is a basic implementation of the template method design pattern [34], which enables the CVTNG system to separate the user concerns of creating computational verb-based word models and narrative depictions. The template method design pattern also enables the re-use of context of interpretation for different narrative depictions. Adherence to these design principles allows the CVTNG system to integrate flexibly with representational media.

A context editor user interface within the CVTNG system, as shown in figure 5.2, allows users to create contexts of interpretation, subcontexts, and partial contexts without directly specifying the corresponding structures in the XML storage format. A user specifies a context of interpretation by entering

- the name of the context,
- the name of a related context of interpretation or a subcontext (if applicable),
- whether the context is a subcontext or a partial context,
- a name for the context property,
- the name of the attribute variable,
- a transformation by which the attribute variable is applied to the narrative depiction and,
- a default value for the attribute value within the defined context of interpretation.

An XML structure, as defined above, that contains the entered context attributes is then saved. The user modifies the stored XML structure by saving modified attribute values.

5.3.3 Variable sets

Variable sets are a CVTNG architecture feature that relates to the design principle of separating system user concerns. A variable set provides a means for representational media to integrate freely with the parsing and variable resolution subsystems. Variable sets serve as a protocol for the interaction of computational verb models and their associated attribute variables and external representational media such as computer graphics and sound. A variable set specifies the list of variables exposed by a narrative depiction to the evolving function models of the CVTNG. The narrative depictions utilise the variable name listed in a variable set within their internal representations. The name of the associated property (refer to section 5.3.2) is used as the name of the associated attribute

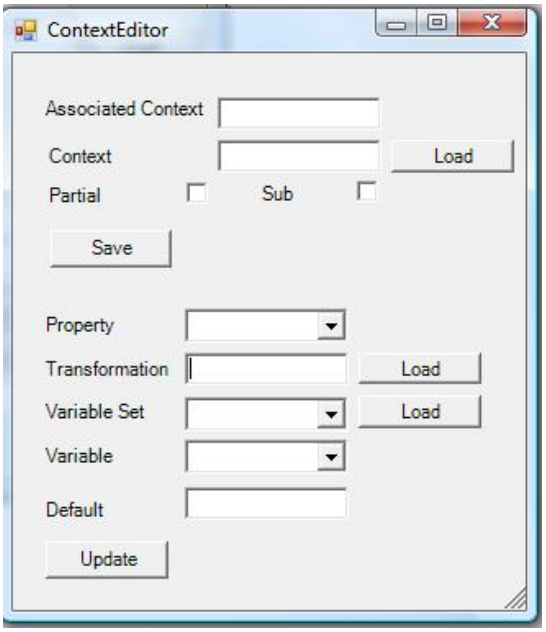

Figure 5.2: The CVTNG context editor

variable within the evolving function models of the CVTNG system. A property therefore defines a mapping between an attribute variable used within the computational verb models of the CVTNG system and a variable internal to a specific narrative depiction for a defined context of interpretation. Creators of representational media may therefore link their narrative depictions to the evolving function models of the CVTNG system by using variables in the variable set.

A variable list that contains a list of variables related to the commonly adjustable features of a narrative depiction allows the free integration of that type of narrative depiction to a context of interpretation. A graphical model could prescribe to a list of variables that specifies variables which parameterise the translation, scaling, and rotation of vertices that constitute a graphical model. A context of interpretation may define properties to map between its attribute variables and the internal variables specified within a graphical model variable list. Computational verb models defined within the defined context of interpretation are therefore able to interact with graphical models that use internal variables on the variable list or provide aliases to the variables on the variable list.

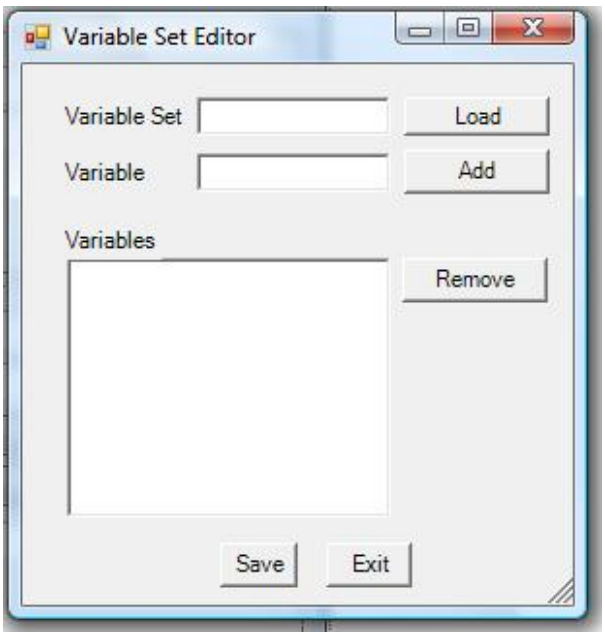

Figure 5.3: The CVTNG variable set editor

Variable sets are stored as simple XML files. Tags are specified with the variable set XML structure to annotate the name of the variable set as well as the names of the supported variables. The flat list of variable names does not fully utilise the nested nature of XML structures, but allows for additional metadata attributes to be specified. Specifying metadata and corresponding protocols will allow a full-fledged protocol to be specified for the interaction of CVTNG evolving function word models with representational media.

A variable set editor as shown in figure 5.3 was created within the CVTNG architecture to define variable sets without the XML storage format having to be manipulated directly. The variable set editor enables the creation and expansion of variable sets that integrate with the context-creation features of the CVTNG system (refer to section 5.3.2).

5.3.4 Transformations

Transformations allow the measurement value stored in an attribute variable to be applied to an external narrative depiction in a way that is sensible for the narrative de-

piction. A transformation on a graphical model may take the form of a transformation matrix applied to the vertices of the graphical model [56]. A transformation applied to an AI script may in turn take an entirely different form.

The encapsulation of transformations allows the creators of context of interpretation to apply evolving function models defined within the context of interpretation to a narrative depiction, without having to have knowledge of the internal representation of the narrative depiction. The creators of narrative depictions can in turn publish a set of transformations related to the internal variables of a variable list (refer to section 5.3.3) that the narrative depiction ascribes to. A protocol is therefore defined on either side of an interpretive context that allows the creators of narrative depictions to integrate with evolving function word models. The creators of evolving function word models in turn may apply their defined dynamics, as specified by evolving functions, to all narrative depictions that ascribe to the supported variable list and transformations.

Transformations are created in the CVTNG system as a separate code assembly (compiled library). The code for a transformation is specified in a class that implements a simple transformation function interface. The interface contains methods for initialising a transformation and for applying the transformation to a reference parameter altered according to the implemented transformation.

5.4 Representation of computational verb-based word models in the CVTNG system

This section describes the components of the CVTNG architecture used in the definition of computational verb-based word models within a defined context of interpretation (refer to section 5.3). The basics of word-in-context definitions are discussed and an overview of the supported natural language word types is given in section 5.4.1. Sections 5.4.2 to 5.4.5 provide an in-depth look at the definition of noun, adjective, verb, and adverb models in the CVTNG system. The models specified in the architecture relate to the theoretical word models presented in sections 4.3.1 through 4.3.4. The structure and generation of element classes are presented as a means of interfacing measurement values to narrative depictions in section 5.4.6. Section 5.4.7 contains a discussion of a

functions library used in the construction of evolving function models.

5.4.1 Word definition basics

The basic word types in the English language are as follows:

- *Nouns:* Natural language classifies nouns as the parts of speech that name a person, object, place, quality, or occurrence [2]. Nouns form the subjects and objects to which adjectives, verbs, and prepositions apply. In terms of CVT, nouns are seen as the passive centre in the English language [112].

- *Articles:* Articles in (English) grammar are the part of speech that specifies a noun (that is, makes a noun more specific). The articles in English are "a", "an", and "the" [2].

- *Pronouns:* Natural language defines pronouns as the part of speech which substitutes for a noun or noun phrase that names a person. A pronoun is substituted for a noun in the context of sentences that have previously introduced the person specified by the noun. An example of a pronoun substituting for a noun is "Jack is a boy. His favourite colour is green.". ("His" stands in the place of "Jack's".) [2]

- *Adjectives:* Natural language defines adjectives as the part of speech that modifies nouns and pronouns by limiting, qualifying, or specifying them [2].

- *Verbs:* Natural language defines a verb as the part of speech that expresses existence, action, or occurrence [2]. In terms of CVT, verbs form the active centre in the English language [112].

- *Prepositions:* Natural language defines a preposition as a word or phrase that precedes a noun or a pronoun (substantive) which relates the substantive to a verb [2]. Examples of prepositions are "at", "with", and "on". (In "The cat sits on the mat","on" relates "mat" to the verb "sits".)

- *Adverbs:* Natural language defines an adverb as the part of speech that modifies an adjective, a verb, or another adverb [2].

```
<Word>
 <Context>
  <Type>
   .
   . (context attributes for word type)
   .
  </Type>
 </Context>
</Word>
```

Figure 5.4: The CVTNG word definition storage structure (XML)

- *Conjunctions:* Natural language, in the context of grammar, defines conjunctions as the part of speech that serves to connect words, phrases, and sentences [2].
- *Numerals:* Natural language defines numerals as words or symbols that represent numbers [3].
- *Interjections:* Natural language defines an interjection as the part of speech that expresses emotion. Interjections are capable of standing alone [2]. ("Aha!","Never!", "You're joking!")

Nouns, adjectives, adverbs, and verbs serve as the theoretical core of the CVTNG system based on the theoretical models of chapter 4. In the CVTNG system, nouns are used to model the passive centre of language and they have adjectives as their modifiers or determiners; verbs form the active centre of language and they have adverbs as their modifiers. Articles and conjunctions are not modelled in the present work but are treated instead as modifiers of the parsing procedure. Pronouns are not discussed in this work but are treated similarly to nouns. Prepositions combine with verbs to form verb phrases and are therefore treated as verbs (refer to section 4.3.3) in this work. Numerals do not fall within the scope of this work but are treated in similar fashion to verb numbers [110]. Verb numbers are the CVT models for numbers and number words and are not discussed within the scope of the current work. Interjections can be conjectured to be modelled as self-perceptions of the internal system (refer to definition 3.13) that model an individual's emotional state. No such investigations are done within either the scope of this work or the prototype CVTNG system.

The CVTNG system constructs a computational verb-based word model in terms of defined contexts of interpretation (refer to section 5.3). A series of properties are

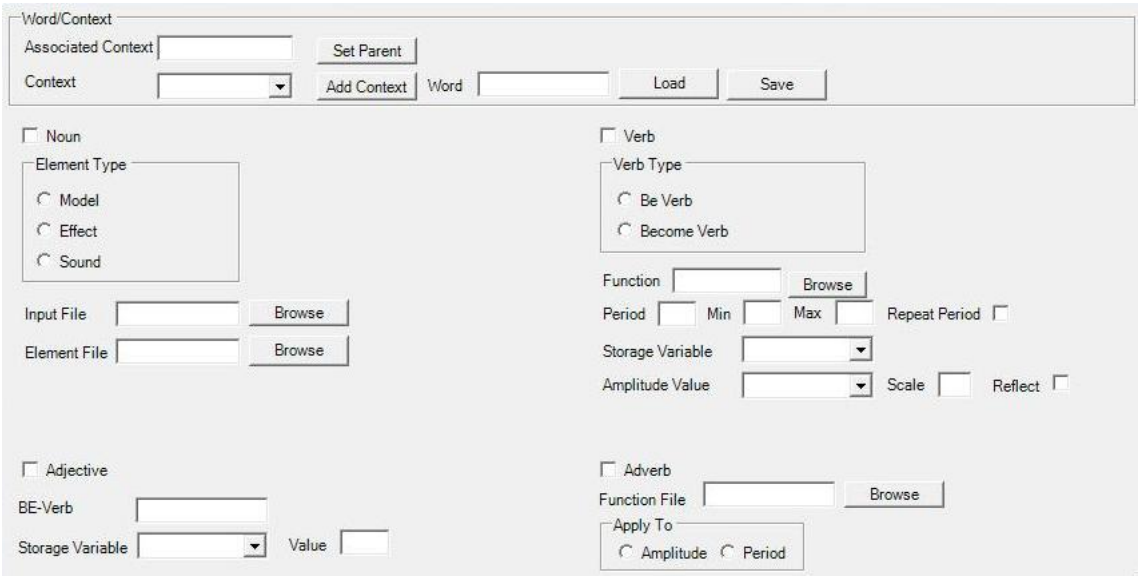

Figure 5.5: The CVTNG word editor

defined for a context of interpretation. A word model within a context of interpretation may utilise the defined properties as attribute variables. Evolving function models for adjectives and verbs are specified in terms of a context of interpretation's properties that are used as attributed variables and specified functions (refer to section 4.1). Nouns are defined within a context of interpretation by specifying a narrative depiction in the form of a class file that serves as an interface with the associated narrative depiction. Adverbs are specified in terms of mathematical functions within a context of interpretation.

Word models within a context of interpretation are stored within the CVTNG system as XML structures (refer to figure 5.4). Word models are subdivided according to contexts of interpretation, the subcontexts of a context of interpretation, and the partial contexts related to a subcontext. Word models are further subdivided according to word type. Within the XML structure of a word model a tag is created for every combination of word type and context of interpretation. The tag for a word type and context of interpretation pair contains tags that specify the attributes of the word within the associated context of interpretation and word-type combination. A word editor, as shown in figure 5.5, was created in the CVTNG architecture to create word models within a context of interpretation without the associated XML storage structure being created directly.

A central store called the "dictionary" is updated when a word model is created or modified in a context of interpretation. The dictionary serves as a central database of the word models that exist. The "dictionary" also stores the word types associated with the specified word models. It is stored as an XML structure.

The general aspects of word definitions have been presented. Sections 5.4.2 through 5.4.5 specify the specific architectural features of the supported word types. The attributes, XML storage structure, related word editor interface elements, and class models of the supported word types are described: the CVTNG architecture features related to noun models are presented in section 5.4.2; those related to adjectives are presented in section 5.4.3; the CVTNG architecture features related to verb and adverb models are presented in sections 5.4.4 and 5.4.5 respectively.

5.4.2 Nouns

A noun model within a context of interpretation as specified in the CVTNG system is specified by the following attributes: element type, input file, and output file.

- `Element type:` Categorises the associated representational means as a graphic, a sound file, or an AI script. Graphics, sound files, and AI scripts by no means represent a comprehensive list of interactive narrative depictions. The element type attribute, however, serves to group contexts during the generation of representational element interfaces.

- `Input file:` Contains the full path of a file that serves as the point of access for a narrative depiction associated with the noun within the context of interpretation. The word "tree", modelled as a noun within a visual context of interpretation, will point to a file containing the information of a graphical model as an example. The word "tree", in the context of sound, may point to a sound file of leaves blowing in the wind.

- `Output file:` Contains the full path of a file in which a generated class that implements the `Element` interface will be stored. The `Element` interface serves as an adapter [34] and allows the evolving function models of the CVTNG system to interface with the associated narrative depiction specified in the input file attribute.

The three listed attributes serve to represent a noun within a context of interpretation. The necessary data is stored to associate a noun with a narrative depiction in a context of interpretation. A reference is also stored in the output file attribute for the adapter that is generated to allow evolving function models of adjectives and verbs to interface with a specified narrative depiction. Noun models are created using the word editor illustrated in figure 5.5.

The noun "tree" can, for example, be specified in the "Visual" context of interpretation by specifying the following attribute values:

- `Element type:` Graphic
- `Input file:` The path of a graphical model of a tree.
- `Output file:` The path and file name of a file that a generated specification of the `Element` class is stored in. The specification of the `Element` interface is referred to as an element class within the CVTNG system architecture. The element class acts as an interface between the CVTNG system and the representational system. The representational system in the case of a graphical model may be a graphics engine that renders the graphical model specified in the `Input file` attribute.

Nouns are significantly more complex within the CVTNG code base than is suggested by their simple storage structures. Nouns serve as a grouping within the context of the the parsing and variable value resolution procedures of the CVTNG system. Nouns act as groupings of attribute variables as specified in section 4.3.1. Nouns also serve as collectors for the references to a narrative depiction and groups a noun word with its narrative depiction in a context of interpretation (refer to section 4.3.1).

Figure 5.6 shows a UML class diagram that represents the implementation of nouns within the CVTNG system. The flyweight design pattern [34] is applied through the introduction of `NounInstance` classes. Every instance of the `NounInstance` class stores the information unique to the particular instance of a noun resultant from parsing natural language. The `NounInstance` instances share common representational elements. The shared representational elements are stored within a shared instance of the `Noun` class to avoid duplication of large structures, such as graphical models that correspond to the representational elements, in memory. The `NounFactory` class controls the instantiation

of the `Noun` class. The `Vocabulary` class handles the parsing procedures and variable resolution procedures (refer to section 4.4.2 and 4.4.3) of the CVTNG system. If the `Vocabulary` class requests an instance of the `Noun` class, the `NounFactory` class checks whether an instance should be created and passes a reference of the `Noun` class instance to the `Vocabulary` class.

The structures of the representational element are created in memory and stored when an instance of the `Noun` class is created. The stored representational elements correspond to the narrative depictions related to the noun for the contexts of interpretation the noun is modelled in.

The phrase "the blue house is next to the red house" contains two instances of the noun "house". Each instance of "house" has a unique colour attribute as described by an adjective. The CVTNG system creates two instances of the `NounInstance` class when the sentence is parsed. These instances relate to the "blue" and "red" descriptions of the noun "house". The representational element associated with the noun "house" for the context of interpretation may be a graphical model of a house. A reference to the structures in memory that contain the graphical model of a house is stored within the `Noun` class instance. The two `NounInstance` class instances access the graphical model via the instance of the `Noun` class in an implementation of the flyweight design pattern [34].

The application of the flyweight design pattern allows the CVTNG system to store large structures such as the vertices of a graphical model only once in memory. The structure in memory that relates to the graphical model can still be accessed and re-used as required. The shared representation stored within the instance of the `Noun` class is then transformed as parameterised by attribute variables stored within every `NounInstance` class instance. The `NounInstance` class instances represent the unique occurrences of the noun in the narrative text; they therefore also store the attribute variables unique to every unique occurrence of a noun in the narrative text. A unique occurrence of a noun in the narrative text refers to a single use of a noun word in the narrative text to designate an object or group of objects that can be distinguished from other uses of the noun to designate other objects. The CVTNG system determines the ability to distinguish between unique occurrences of a noun.

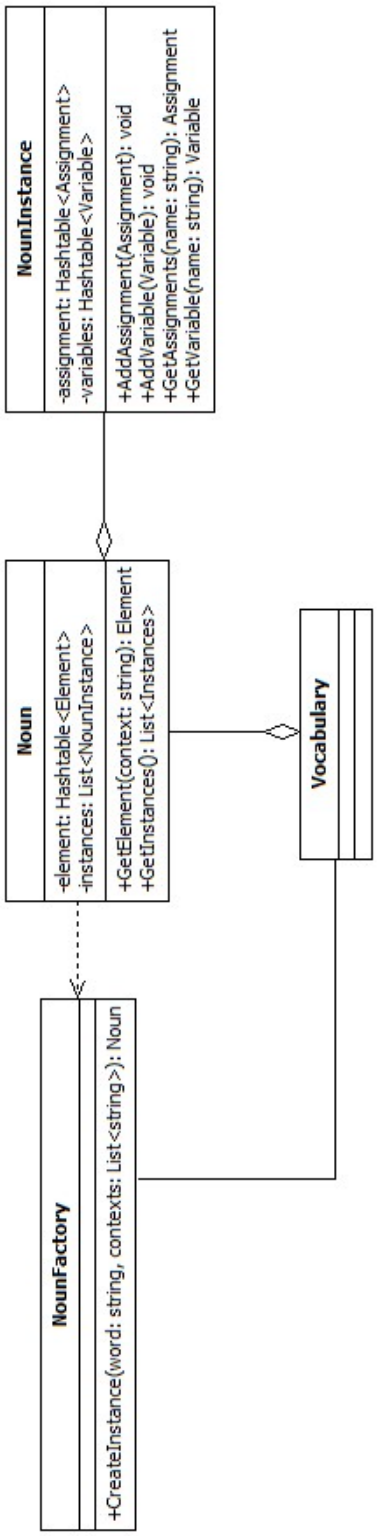

Figure 5.6: The CVTNG noun architecture

5.4.3 Adjectives

The CVTNG system stores an adjective modelled within a context of interpretation as the following set of attributes:

- `Property:` The property attribute stores the context property (refer to section 5.3) grouped with a noun as an attribute variable (refer to section 4.2). A unique instance of the attribute variable is created for every unique occurrence of an adjective–noun pair in the modelled narrative text.

- `BE-verb:` The BE-verb attribute contains a verb-based word definition specified in the same context of interpretation as the adjective model. The associated verb is flagged as a BE-verb and is implemented as a static verb (refer to definition 3.15). The BE-verb can also be modelled as another verb type if the modelled adjective is not an adjective as defined for CVT (refer to section 4.3.2). The generic function, f_i, as defined for equation (4.3), is inherited from the specified BE-verb. The maximum and minimum term sizes (refer to section 4.4.3.5) and the validity period of the verb (refer to definition 3.14) are also inherited.

- `Value:` The amplitude value, α, defined for the evolving function model of the adjective word (refer to section 4.3.2).

The attributes listed above are defined within every context of interpretation supported by the adjective. The adjective models within a context of interpretation are specified using the word editor shown in figure 5.5.

The adjective "red", for example, may be implemented within the context of interpretation "RedColouring" by specifying the following attribute values:

- `Property:` Red.

- `BE-verb`: "be".

- `Value:` 0.7

The attribute values specified define the evolving function model for the adjective "red" by specifying that the generic function, f, and the scaling value, κ, from the evolving

function model of the verb "be" should be used. The generic function value, $f(t)$, and scaling value, κ, are multiplied by the amplitude value, 0.7, to calculate a measurement value stored in the attribute variable "Red". The attribute variable "Red" corresponds to the property "Red" within evolving function models of the CVTNG system. If the generic function value, $f(t)$, and scaling value, κ, for the verb "be" are defined as $f = 1.0$ and $\kappa = 1.0$ respectively, the evolving function equation defined for the adjective "red" in the context of interpretation "RedColouring" is defined as:

$$Red = (0.7)\kappa f(t) = (0.7)(1.0)(1.0) = 0.7. \tag{5.1}$$

Figure 5.7 shows a UML class diagram of the implementation of adjectives in terms of the CVTNG code base. If the `Vocabulary` class requests an instance of the `Adjective` class, an instance of the `Adjective` class is created by the `AdjectiveFactory` class and a reference to the `Adjective` class instance is stored within the `Vocabulary` class. An instance of the `Adjective` class is created for every context of interpretation the adjective is modelled for. A `Verb` class instance is created for the BE-verb specified for an adjective within a specific context of interpretation. A reference to the created `Verb` class instance is stored within the `Adjective` class instance. The associated `Verb` class instance is not shown in figure 5.7.

An instance of the `Adjective` class that associates with an instance of the `NounInstance` class (an expanded class is shown) is created for every unique adjective–noun pair in the narrative text. The instance of the `NounInstance` class created for the unique adjective–noun pair stores the attribute variables specified in the evolving function model of the adjective in a specific context of interpretation. The CVTNG system determines unique occurrences of a noun in narrative text according to the adjectives associated with the noun in the narrative text. If the adjectives associated with a noun are identical to those of another adjective–noun pairing, the same `NounInstance` class references are returned to the `Vocabulary` class. The CVTNG class therefore considers nouns with identical associated adjectives as separate occurrences of the same noun instance.

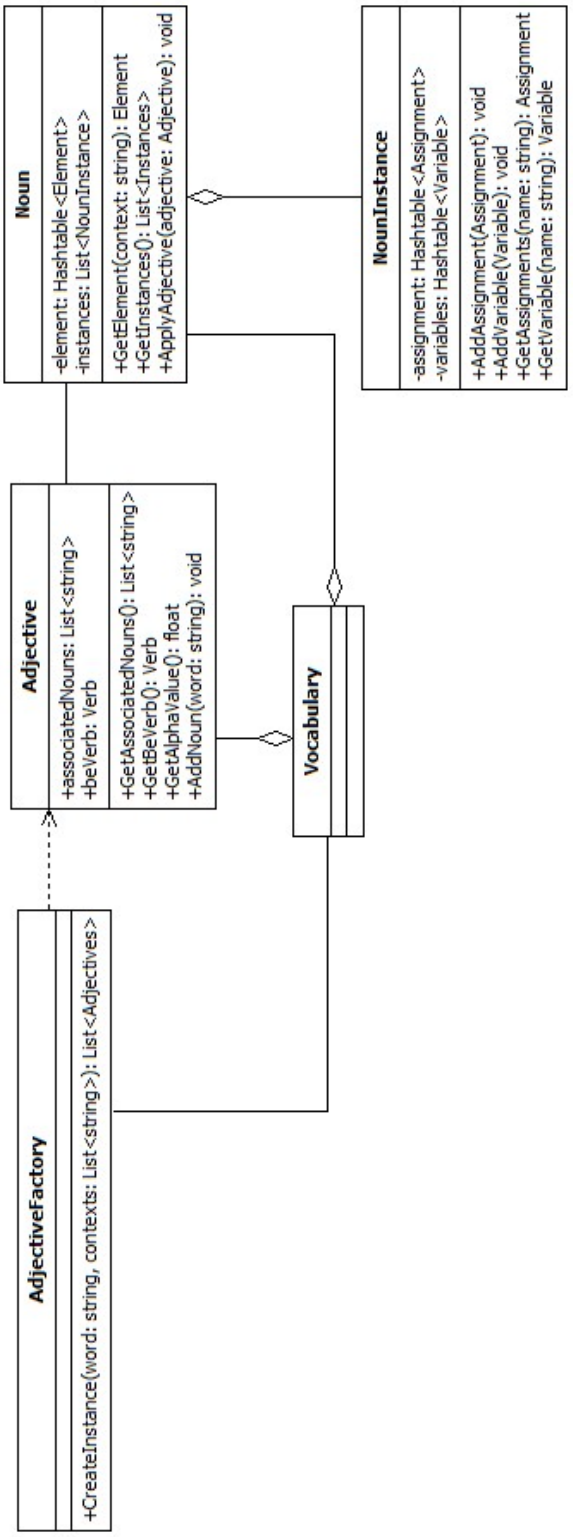

Figure 5.7: The CVTNG adjective architecture

5.4.4 Verbs

A verb modelled within a context of interpretation is stored in the CVTNG system as the following attributes:

- Verb type: The `Verb type` attribute indicates whether the verb model defined for the context of interpretation specifies a static computational verb (refer to definition 3.15) or a computational verb that models dynamics (changes over time).
- Subject property: The `Subject property` attribute stores the context property (refer to section 5.3) that is grouped with a subject noun as an attribute variable. The attribute variable is stored within an instance of the `NounInstance` class for the defined context of interpretation. The instance of the `NounInstance` class for the context of interpretation is determined by the adjectives associated with the noun in the narrative text (refer to section 5.4.3).
- Function: The `Function` attribute contains the path of a `Function` class. The `Function` class implements a function, f_1, as specified for the evolving function model of the verb within a context of interpretation (refer to equation (4.3)). The `Function` class implements a function, f_i, related to a term in the evolving function model of a verb (refer to equation (4.3)) when specified for a subcontext (refer to section 4.2).
- Object value: The `Object value` attribute stores the amplitude value, α, specified for the evolving function model of a verb within a context of interpretation (refer to equation (4.3)). The object value attribute stores an amplitude value, α_i, related to a term of the evolving function model of a verb (refer to equation (4.3)) when specified for a subcontext (refer to section 4.2). The object value attribute stores an amplitude value, α_{i_j}, related to an amplitude value defined as a sum (refer to equation (4.7)) when specified within a partial context.
- Period: The `Period` attributes stores the lifetime of the evolving function model of a verb within a context of interpretation, $T_{\mathscr{C}}$ (refer to definition 3.22).
- Infinite: The `Infinite` attribute is a flag that indicates whether the evolving function model of a verb within the context of interpretation should repeat over

a time interval, T. If the `Infinite` flag is set, the validity period of the evolving function as specified in the period attribute is infinite. The evolving function takes the form of equation (4.4) and it repeats over the time interval, T.

- Reflective: The `Reflective` attribute is a flag which indicates that the amplitude value specified in the object values attribute should be obtained from the subject noun in a verb sentence (refer to definition 3.20). The subject noun therefore acts as the object noun for the verb sentence.

- Maximum: The `Maximum` attribute stores an upper bound, α^+, for the amplitude value, α, of an evolving function in a context of interpretation as well as for the amplitude value, α_i, of a term in an evolving function when specified within a subcontext. The `Maximum` attribute also stores an upper bound, $\alpha_{i_j}^+$, for the partial amplitude value, α_{i_j}, if the attribute `Maximum` is specified for a partial context. The partial amplitude value, α_{i_j}, is summed with other partial amplitude values to obtain the amplitude of a term in an evolving function that is defined in the related subcontext.

- Minimum: The `Minimum` attribute stores a lower bound, α^-, for the amplitude value, α, of an evolving function in a context of interpretation as well as for the amplitude value, α_i, of a term in an evolving function when specified within a subcontext. The `Minimum` attribute also stores a lower bound, $\alpha_{i_j}^-$, for the partial amplitude value, α_{i_j}, if the `Minimum` attribute is specified for a partial context. The partial amplitude value, α_{i_j}, is summed with other partial amplitude values to obtain the amplitude value of a term in an evolving function that is defined in the related subcontext.

- Scale: The `Scale` attribute stores the scaling factor, κ, for the amplitude value, α, used in the evolving function model of a verb (refer to equation (4.3)) specified for a context of interpretation as well as for an amplitude value, α_i, of a term in an evolving function model of a verb when defined in a subcontext. The scale attribute stores the scaling factor, κ_{i_j}, of the partial amplitude value, α_{i_j}, when defined in a partial context (refer to equation (4.7)).

The attributes above are specified using the word editor shown in figure 5.5. The attributes `Verb type`, `Subject property`, `Period`, and `Infinite` are specified only within a context of interpretation. The attribute function is specified only within a context of interpretation or a subcontext. The remainder of the attributes listed above are specified in contexts of interpretation, the subcontext related to a context of interpretation, and the partial contexts related to a subcontext.

The evolving function model for the verb "jumps" may, for example, be defined within the context of interpretation "Position above ground" by specifying the following attributes:

- `Verb type:` Become. (The evolving function model specified is that of a dynamic computational verb.)
- `Subject property:` YPosition (The attribute variable chosen to store the measurement value calculated from the evolving function model is "YPosition", which refers to the height of an object above the ground.)
- `Function`: $\sin t$. (The generic function chosen for the evolving function model is $f(t) = \sin t$.)
- `Object value:` 1.0 (The evolving function model is scaled by the amplitude value 1.0.)
- `Period:` π (The evolving function model is specified for a period of size π.)
- `Infinite:` False. (The evolving function model should not repeatedly be evaluated.)
- `Reflective:` False. (The amplitude value is not retrieved from the subject noun.)
- `Scale:` 0.5. (The scaling factor for the evolving function model is 0.5.)
- `Maximum:` 0.5. (The maximum measurement value that can be obtained from the evolving function model is 0.5.)
- `Minimum:` 1.0. (The minimum measurement value that can be obtained from the evolving function model is 0.0.)

The evolving function model specified for the verb "jump" when the attribute values above are specified, is:

$$YPosition = \mathcal{E}_{jumps}(t) = (0.5)(1.0)\sin t = 0.5\sin t. \tag{5.2}$$

Figure 5.8 shows a UML diagram that represents the relationships between the `Verb` class in terms of the CVTNG code base. If the `Vocabulary` class requests an instance of the `Verb` class, the `VerbFactory` class creates an instance and returns a reference to the `Vocabulary` class. Attribute variables are created according to the evolving function models specified for a verb in the specified contexts of interpretation. The attribute variables are created for every unique grouping of a verb with a subject and/or object noun in the narrative text. The attribute variables are stored within the `NounInstances` class instances associated with the subject noun. An instance of the `Verb` class stores the evolving function associated with the verb for a context of interpretation. A unique instance of the `Verb` class is therefore created for every context of interpretation for every verb modelled in the narrative text.

The relationship between the specifications of a `Function` interface and the `Verb` class is not shown in figure 5.8. If an instance of the `Verb` class is created by the `VerbFactory` class, instances of specifications of the `Function` interface are created. The specifications of the `Function` interface correspond to the functions, f_i, defined for the evolving function models in the present work (refer to equation (4.3)). References to the `Function` interface specifications are stored in the instance of the associated `Verb` class.

Reflection is used to allow functions to be maintained within a separate library. A reference to an instance of the `Adverb` class may also be stored within an instance of the `Verb` class for a verb–adverb pair within the modelled narrative text.

5.4.5 Adverbs

An adverb modelled within a context of interpretation is stored in the CVTNG system, as the following list of attributes:

- Transformation function: The function attribute stores the path of a class that implements a specification of the `Function` interface. The `Function` interface

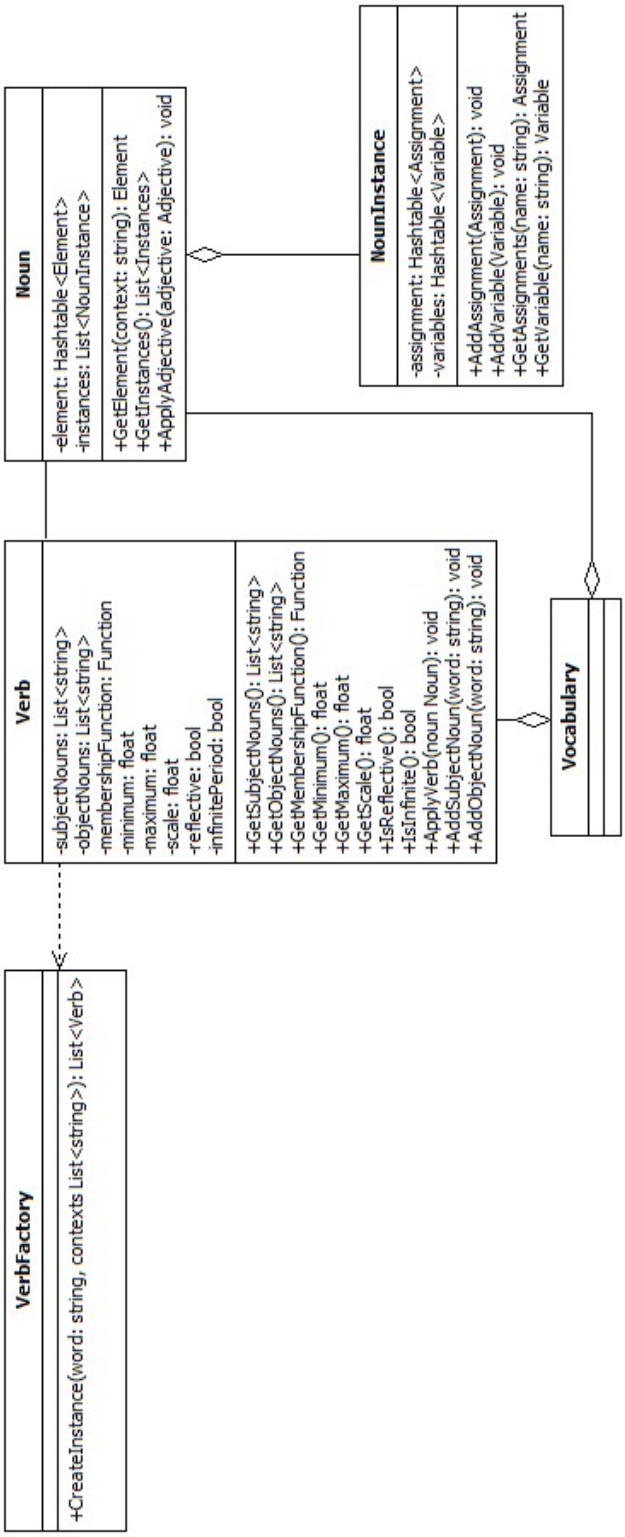

Figure 5.8: The CVTNG verb architecture

specification implements the transformation function, Ψ, as defined for equations (4.20) and (4.21).

- Apply to: The apply to attribute indicates whether the transformation function specified in the function attribute should be applied to the frequency or amplitude of an evolving function model of an associated verb.

The attributes above are specified using the word editor shown in figure 5.5.

The adverb "dark" may, for example, be specified within the "RedColouring" context of interpretation by specifying the following attributes:

- `Transformation function:` $\Psi_{half} = 0.5$. (The transformation function specified for the evolving function model is $\Psi_{half} = 0.5$.)
- `Apply to:` Amplitude. (The transformation function specified within the evolving function model for the adverb "dark" is an adverb of amplitude. An adverb of amplitude is applied to transform an amplitude value in an evolving function specified for an associated adjective or verb.)

If the adverb "dark" is defined by specifying the attribute values above and it is applied to the adjective "red' in the natural language phrase "dark red", the following evolving function model results:

$$Red = \mathcal{E}_{red}(t) = (0.7)\Psi_{half}(1.0)f_{one}(t) = (0.7)(0.5)(1.0)(1.0) = 0.35. \tag{5.3}$$

Figure 5.9 shows a UML diagram that represents adverbs in terms of CVTNG code base. If the `Vocabulary` class requests an instance of the `Adverb` class, the `AdverbFactory` class creates an instance of the `Adverb` class and returns a reference to the `Vocabulary` class. An instance of the `Adverb` class is created for every context of interpretation, subcontext, or partial context the `Adverb` is defined for. The class diagram in figure 5.9 shows the relationships between the `Adverb` class and the `Verb` and `Adjective` classes. A reference to the `Adverb` class is stored within a `Verb` class instance for a verb–adverb pair. A reference to the `Adverb` class is also stored within the `Verb` class instance created as the BE-verb of an `Adjective` class instance for an adjective–adverb pair. The relationship between an instance of the `Adverb` class and a

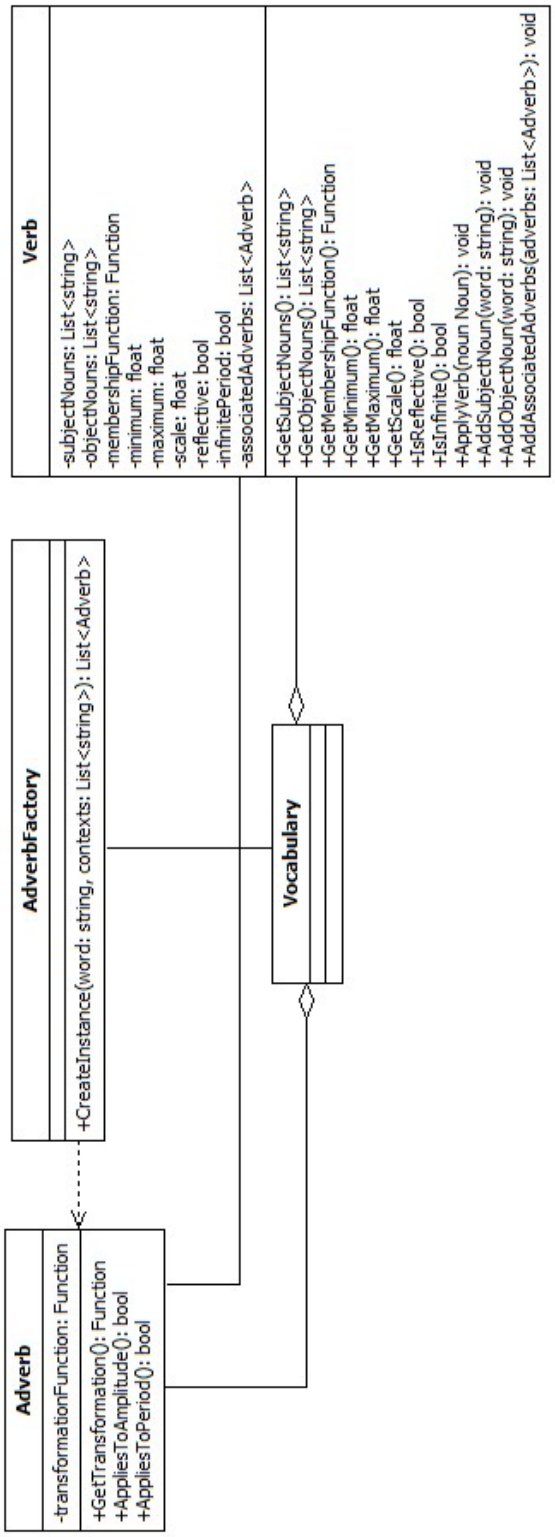

Figure 5.9: The CVTNG adverb architecture

class that serves as a specification of the `Function` interface is not shown. The specification of the `Function` interface serves as the transformation function applied to the evolving function of any associated verb.

5.4.6 Element classes

Element classes are generated classes that implement specifications of the `Element` interface within the CVTNG system. Element classes serve as adapters [34] between the measurement value resolution subsystem and the representation subsystem of the CVTNG system. The CVTNG system is decoupled from a specific representation system. A representation subsystem was, however, developed for the CVTNG system prototype related to this work. Accordingly, element classes are generated and compiled within a separate assembly. The assembly of element classes serves as a reference library for representational systems and the CVTNG measurement value resolution system. The main reasons that element classes are compiled into a separate assembly are:

- *Decoupling:* Representations are typically developed in an environment decoupled from the core CVTNG system. Interaction between a computational verb model in the CVTNG system and an associated representational means is achieved via an element class. The element class acts as an interface and allows a representational element to be handled by a separate representation system such as a graphics engine. The separate system can interact with the CVTNG system without changing the core CVTNG system or requiring a change in the separated representational system. The CVTNG system and the separate representational system are therefore independent with regard to change, though the two systems can still interact with each another via the element class stored in an element class assembly. The element class assembly acts as a library for both systems and also serves as a communication interface.

- *Integration possibilities:* A representation system used by the CVTNG system is decoupled from the evolving function models and measurement value resolution procedures. Third party representation systems such as graphics engines can therefore interact with the CVTNG system without additional adjustment.

The representation system used by the CVTNG system and the evolving function models and measurement value procedures of the CVTNG system are decoupled. Communication between representational elements and the measurement value resolution subsystem takes place through generated specifications of the `Element` interface. The `Element` interface specifies the following four methods that allow the interaction of a decoupled representation system with the CVTNG measurement value resolution system:

1. `Initialize()`: An implementation of the `Initialize()` method serves to initialise the representational element. An element class that implements the `Element` interface for a graphical model will use this method to load the graphical model into memory.

2. `InitVars()`: An implementation of the `InitVars()` method serves to initialise a series of transformations as parameterised by the default values of attribute variables. The default values of the attribute variables and the transformations are as specified within the properties of the specified contexts of interpretation (refer to section 5.3). If an attribute variable is therefore not changed by the evolving function word models of the modelled narrative text, the internal variable of the representational element that corresponds to the attribute variable (refer to section 5.3) will be transformed according to a default transformation. The default transformation serves to set the internal variable to its default state. A property "Position" not changed within the evolving function models of the modelled narrative text is, for example, initialised to its default value by multiplying a zero translation matrix with an array of vectors. The array of vectors is the internal variable of the representational element and stores the vertices of a graphical model. The zero translation matrix is initialised in an implementation of the `InitVars()` method.

3. `Transform()`: An implementation of the `Transform()` method serves to apply the transformations that correspond to all the properties of the specified contexts of interpretation. If an attribute variable is not changed by the evolving function models of the modelled narrative text, a default transformation as initialised in the `InitVars()` is applied to the internal variable. If an attribute variable is changed

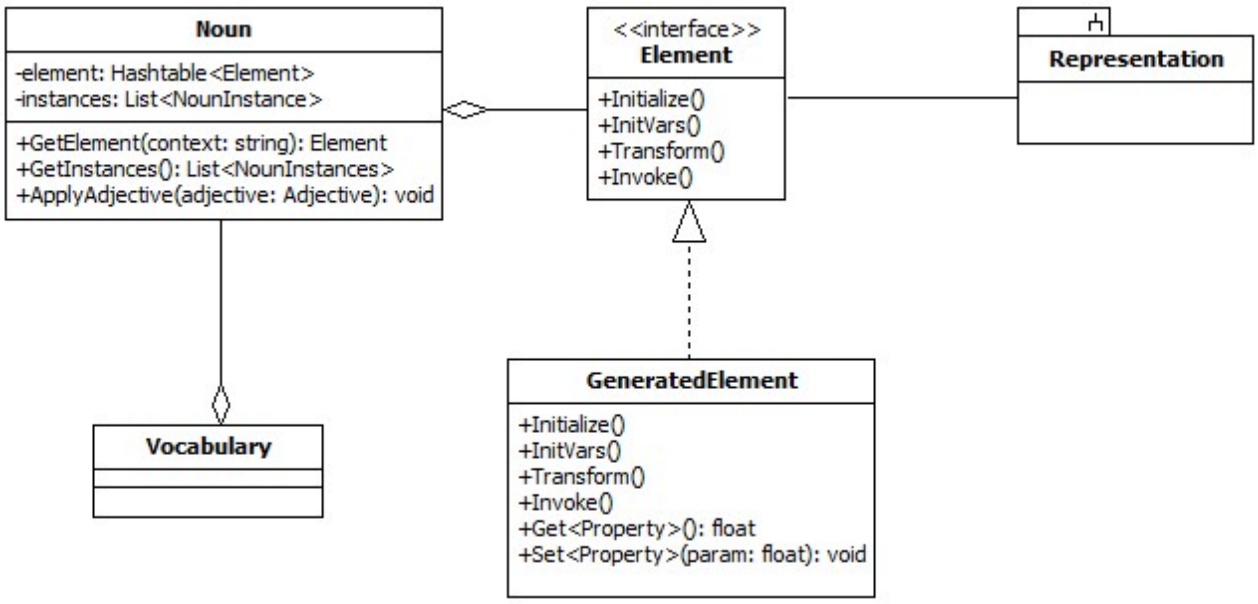

Figure 5.10: The CVTNG element file architecture

by the evolving function models of the modelled narrative text, the transformation applied to the internal variable of the associated representational element is initialised according to the attribute variable. The transformations applied to the internal variables of the representational elements serve to reflect the statuses and changes described in the narrative text. The statuses and changes described in the narrative text are in turn modelled by evolving function models.

4. `Invoke()`: An implementation of the `Invoke()` method serves to perform the representation of a representational element. A generated element class that corresponds to a representational element in the form of a graphical model would contain a series of calls to a graphics API. Such calls serve to render the graphical model onto a display.

A generator of element classes, which creates an element class according to the `Element` interface specified above, was implemented for the CVTNG prototype. An element class is created for every unique output file attribute specified within the contexts of interpretation a noun is defined in. A generated element class contains the methods

of the `Element` interface specified above applied to a list of supported properties (refer to section 5.3). The list of supported properties is formed from the combination of all properties specified within the contexts of interpretation a noun is modelled in.

The generator implemented within the CVTNG prototype generates a series of accessor methods that correspond to the supported list of properties. The accessor methods serve to initialise the transformations specified within the corresponding property of a context of interpretation by a measurement value passed to the accessor method as a parameter. A corresponding accessor method is also generated that returns the current parameter value of the transformation specified with the associated context property. The parameter value of a transformation in an element class is initialised by the default value specified within a property and is altered by measurement values retrieved from the variable value resolution subsystem of the CVTNG system. The accessor methods enable measurement values to be passed between the element class and the CVTNG system. The accessor methods also allow a separated representation system to access measurement values determined from the evolving function models of the CVTNG system.

Figure 5.10 shows the relationship between the `Element` interface and the corresponding classes of the associated CVTNG subsystems. A single instance of the noun class may have multiple references to instances of specifications of the `Element` interface because a single noun may have multiple related representational elements across multiple contexts of interpretation. A specific representational element in turn may be used by multiple nouns. Nouns serve as an access point to element classes and their corresponding representational elements.

5.4.7 Functions

Realisations of the `Function` interface are implemented in the CVTNG system within a separate assembly called the "function library". The `Function` interface and its specifications are separated from the other elements of the CVTNG system to allow mathematical functions used in the CVTNG system to be developed independently. If an instance of a specification of the `Function` interface is required, a reference is created by reflection. The function modelled within the `Function` interface specification is invoked by

a reflection method call. The separated function library enables the CVTNG system to adhere to the design principle of separating the user concerns of those users who specify mathematical functions for use in the CVTNG system from other users in the system.

The creation of functions in a separated library also allows the re-use of functions created for the function library by different evolving functions and transformation functions within the CVTNG system. The function library enables the CVTNG system to adhere to the design principle of the re-use of artifacts. The separation of the function library from the main CVTNG system also allows functions to be specified in a different programming language from the CVTNG system, if the functions can be compiled into a common runtime with the CVTNG system.

The `Function` interface consists of a single method called the `Evaluate` method. The `Evaluate` method receives an `object` value, which may be either a single value or a collection of values, and returns a single `double` value. The `double` value returned is the value the function evaluates to.

References to the `Function` interface are stored within instances of the `Verb` and `Adverb` classes. The `Function` references correspond to specifications of the `Function` interface that represent the functions, f_i, chosen for the terms of an evolving function (refer to equation (4.3)) within the context of an instance of the `Verb` class. The `Function` reference corresponds to a specification of the `Function` interface that represents the transformation function, Ψ, as given by equations (4.20) and (4.21) within the context of an instance of the `Adverb` class.

5.4.8 Summary of representation of computational verb-based models within the CVTNG architecture

This section has examined the architecture features of the CVTNG system that correspond to the creation of evolving function word models. A summary of word types in the English language has been provided and the word types have been related to their implementation within the `CVTNG` system. The four main word types used in the CVTNG system were identified as nouns, adjectives, verbs, and adverbs. The definition of evolving function word models within a context of interpretation has also been presented. The components of the defined evolving function word models were related to the mod-

els presented in chapter 4. XML was identified as the storage structure for word models. The XML storage format of word models was paired with a graphical user interface editor that allows the attributes of a word model to be captured and stored. Evolving function word models related to the main supported word types of nouns, adjectives, verbs, and adverbs were specified in terms of their XML storage structure, related word editor features, and their class representations within the CVTNG code base. Element classes were presented as generated specifications of the `Element` interface. The `Element` interface and its corresponding element classes allow the representational system used by the CVTNG system to be decoupled from its other subsystems. A function library was discussed that decouples the specification of mathematical function used within the CVTNG system from other subsystems of the CVTNG system.

The section that follows utilises the word models created by the architectural features of this section to parse natural language sentences to their corresponding evolving function models.

5.5 Parsing

This section presents the parsing subsystem of the CVTNG system that draws on the word definitions specified in section 5.4 to translate narrative text into evolving function models. The steps of the parsing procedure are presented in relation to the system design principles described in section 5.5.1. In section 5.5.2, the architecture of the parsing subsystem is presented in terms of corresponding classes in the CVTNG code base, the structure of the classes related to the parsing procedures, and the relationships between classes that form part of the parsing procedure. The concept of parsing bins as a key architectural component in the construction of a generic parsing framework is presented in section 5.5.3. The parsing algorithm is presented, studied, and critiqued in section 5.5.4.

5.5.1 Steps of the CVTNG parsing algorithm

The first objective of the CVTNG parsing subsystem and the first step of the CVTNG parsing procedure is to retrieve the narrative text provided as input. Narrative text may

be retrieved either by simple entry as text or by being read from a file. The retrieval of narrative text becomes more complicated if the narrative is provided in a less machine–friendly format such as voice. Voice recognition and other means by which voice is converted to narrative text are not discussed as they fall outside the scope of the present work.

The second step in the CVTNG parsing procedure is to divide the supplied narrative text into individual words and phrases and establish the relationships between the words and phrases. An example of a phrase that may be identified by the CVTNG parsing procedure is a verb–preposition pair such as "jumps onto". The CVTNG parsing subsystem divides sentences into words or phrases because the CVTNG system models natural language in terms of the computational verb model for a single word within a context of interpretation. Narrative text is divided into sentences and the sentences are divided into individual words or phrases.

The relationships between the words and phrases within a single sentence are then determined within the implemented CVTNG system prototype in the scope of a sentence. The unique instances of nouns, adjectives, verbs, and adverbs are determined over all sentences. Word instances are maintained over the scope of all narrative text to allow multiple verbs to act on the same object denoted by an instance of a noun. Sophisticated approaches should be followed for longer units of text such as the paragraphs, chapters, and volumes of books. Examples of such approaches are the subject-determination and point-of-view tracking algorithms of Wiebe [97] [98] [99] and the narrative-segmentation algorithms of Nakhimovsky *et al* [69]. The implemented CVTNG prototype does not make use of either point-of-view tracking or a narrative-segmentation algorithm in its current form.

The third and final step of the CVTNG parsing procedure is to translate words and word relationships to evolving function models. The CVTNG parsing system constructs evolving function models according to the word-in-context definitions created by the architectural features presented in section 5.4. The constructed word in context models correspond to the evolving function models for the supported word types as presented in sections 4.3.1 through 4.3.4.

The word-in-context definitions are applied to the words and word relationships iden-

tified in step one of the CVTNG parsing procedure. The words are constructed from their stored definitions according to the contexts of interpretation chosen by the user. The classes that correspond to the word types in the CVTNG code, as presented in sections 5.4.2 to 5.4.5, are instanced. The steps of the CVTNG parsing procedure are complete once the word definitions have been read from their XML storage structures and the appropriate classes in the CVTNG code base have been instanced to contain the relationships and evolving function models of the word in the supplied narrative text.

5.5.2 Parsing subsystem architecture

Figure 5.11 illustrates the architecture of the CVTNG parsing subsystem in terms of a UML diagram. At the centre of the parsing subsystem (and, in fact, the entire CVTNG system) lies the `Vocabulary` class. The `Vocabulary` class contains the methods that retrieve narrative text, converts the words in the narrative text to their corresponding evolving function models (refer to sections 4.3.1 through 4.3.4), and determines the measurement values that are assigned to the attribute variables of the evolving function models.

The `Vocabulary` class stores and manages the creation of instances of the `Noun`, `Adjective`, `Verb`, and `Adverb` classes. The word classes and the evolving function models and attribute variables stored within them are accessed by an associated representation system via the `Vocabulary` class.

The following properties are defined within the `Vocabulary` class to access words and the evolving function models and attribute variables stored within them:

1. *Nouns:* The `Nouns` property returns the list of instances of the `Noun`. Every instance of the `Noun` class in turn refers to unique instances of the `NounInstance` class that contain the attribute variables associated with the evolving function models of corresponding adjectives and nouns within the narrative text (refer to section 5.4.2).

2. *Verbs:* The `Verbs` property returns a hash table that maps a sentence number and context of interpretation name to a list of instances of the `Verb` class. The hash table mapping enables multiple instances of the `Verb` class to be created where

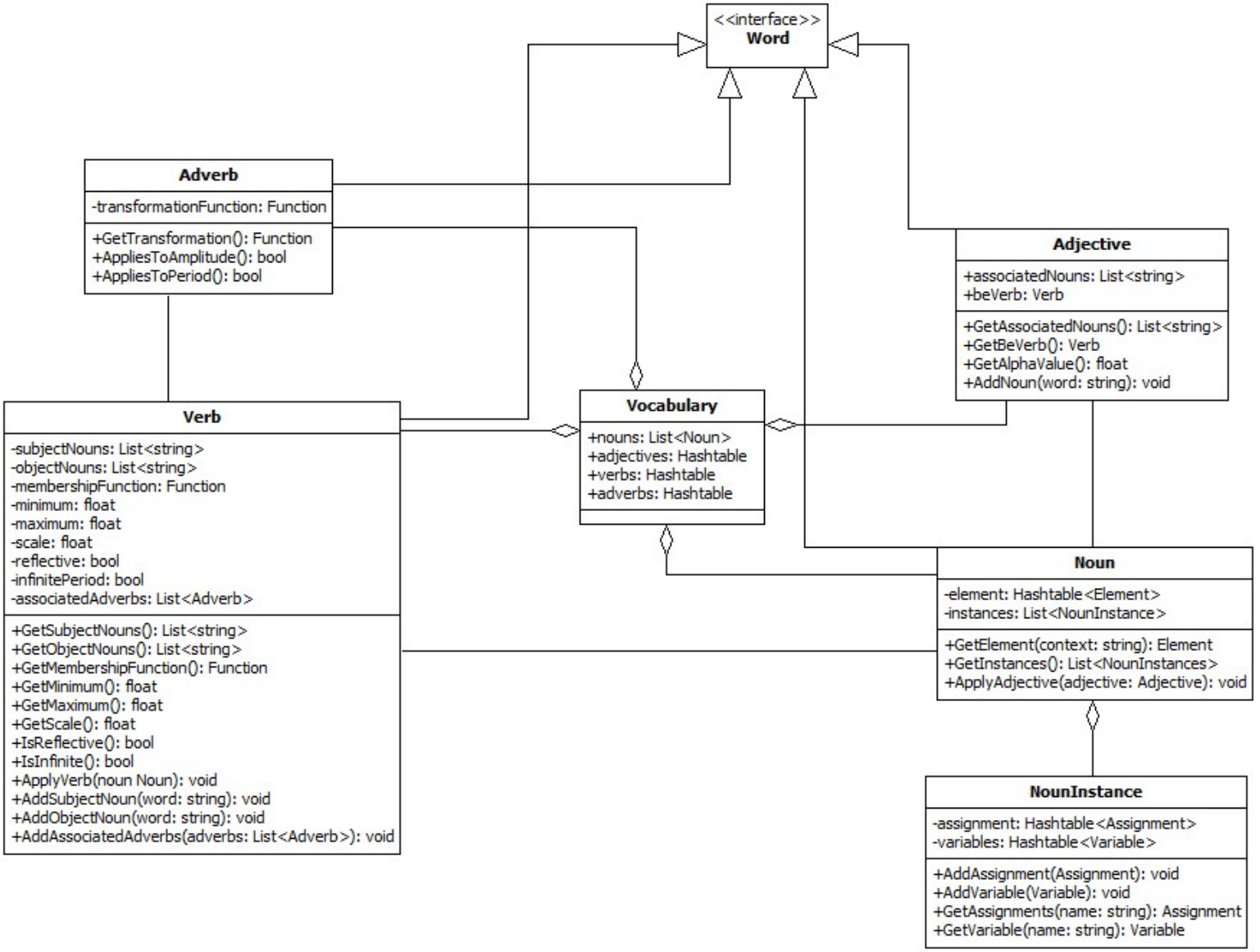

Figure 5.11: The CVTNG parsing architecture

each `Verb` class instance is unique to a sentence and context of interpretation. The mapping returns a list that enables multiple instances of the `Verb` class to be created. Each `Verb` class instance corresponds to the instances of a verb word in a single sentence defined in the same context of interpretation. The sentence "Timmy kicked the ball after Timmy kicked the dog." refers to two instances of the "Verb" class within a visual context of interpretation that both occur in the same sentence. Each instance of the `Verb` relates to the use of a verb in a sentence within a single context of interpretation (refer to section 5.4.4).

3. *Adjectives:* The `Adjectives` property returns a hash table that maps the name of a context of interpretation and a sentence number to a list of instances of the `Adjective` class. The hash table mapping allows multiple instances of the `Adjective` class to be created that are specific to certain sentences and contexts of interpretation. The returned list allows multiple instances of the `Adjective` class to be created within a single context of interpretation for a single sentence within the narrative text. Each instance of the `Adjective` class relates to the use of an adjective within a natural language sentence within a single context of interpretation (refer to section 5.4.3).

4. *Adverbs:* The `Adverbs` property returns a hash table that maps the name of a context of interpretation and a sentence number to a list of instances of the `Adverb` class. The hash table mapping enables multiple instances of the `Adverb` class to be created that are specific to certain sentences and contexts of interpretation. The returned list allows multiple instances of the `Adverb` class to be created in a single sentence for a specific context of interpretation. Each instance of the `Adverb` class relates to a verb–adverb or adjective–adverb pairing with a context of interpretation stored within the corresponding instance of the `Verb` class (refer to section 5.4.5).

The properties of the `Vocabulary` class listed above and the specifications of the `Word` class returned by the properties allow the `Vocabulary` class to manage all instances of the `Word` class created by the parsing procedures of the CVTNG system. The `Word` class is the base class for classes that contain word models such as the `Noun`,`Adjective`,`Verb`,

and `Adverb` classes. The instances of specifications of the `Word` class are stored in a robust manner that allows for multiple words, instances of words, and contexts of interpretation to be stored concurrently. The instances of specifications of the `Word` class can be accessed in a fast and simple manner by means of hash table mappings and list iterations.

Figure 5.11 illustrates the relationship between the `Word` class, the specifications of the `Word` class, and the `Vocabulary` class. The parsing procedure implements the Abstract Factory [34] design pattern by which a factory class relevant to a specification of the `Word` class is instanced. The factory class contains code that creates the required instance(s) of the `Word` class specification. The application of the Abstract Factory design pattern enables the parsing procedure of the CVTNG system to treat all words in a dynamic and generic fashion and makes it possible for the classes that contain the evolving function models of the supported word types to be changed and extended easily.

The `Vocabulary` class associates with the `ParseBin` interface and its specifications. Instances of specifications of the `ParseBin` interface provide a simplified means of parsing narrative text to evolving function models. The specifications of the `ParseBin` interface are also generic and allow the word types supported within the system and the parsing procedures to be easily extended.

The `ParseBin` interface forms the backbone of the CVTNG parsing procedure; the section that follows deals with the `ParseBin` interface in greater detail.

5.5.3 Parsing bins

Natural language is ambiguous, implicit, and inconsistent and as a result poses many problems for the parsing of algorithms regardless of their final objectives. The purpose of the parsing algorithm within the CVTNG system is to translate natural language sentences into specifications of the `Word` class. The specifications of the `Word` class contain the attribute variable and `Function` interface specifications that represent the evolving function models of the supported natural language word types, as presented in sections 4.3.1 to 4.3.4.

The complexities of natural language make it necessary to find a way of simplifying the parsing procedure of natural language words to evolving function models. A natural language sentence does not have a consistent structure and the relationships between

words vary according to the structure of each and every natural language sentence. A parsing algorithm that attempts to address the permutations and exceptions within a natural language sentence will quickly become highly branched and convoluted.

The CVTNG system addresses this problem by introducing the concept of parse bins, as represented by the `ParseBin` interface. The `ParseBin` interface is a generic means of encapsulating the logic that determines the relationships between words in a natural language sentence. A specification of the `ParseBin` interface groups words of similar type (hence the "Bin" suffix). The methods of the `ParseBin` interface as implemented within a specification class execute logic to determine whether another grouping of words within another instance of a specification of the `ParseBin` interface should be associated with the words grouped within the current specification instance of the `ParseBin` interface.

The `ParseBin` interface defines the following methods:

- `AddToBin(Word)`: Implementations of the `AddToBin()` method serve to add a specification of the `Word` class to the collection of `Word` instances of similar type stored in the `ParseBin` specification.

- `GetBinWords()`: Implementations of the `GetBinWords()` method serve to return the collection of `Word` specifications instances of similar type stored within the `ParseBin` specification.

- `GetBinType()`: Implementations of the `GetBinType()` method serve to return the word type of the `Word` specification instances stored in the `ParseBin` specification instance.

- `IsBinType(Word)`: The `IsBinType()` method is a predicate method that returns a "true" or "false" value which indicates whether the `Word` class parameter provided is of the same specification type as the collection of `Word` class instances stored within the `ParseBin` specification.

- `IsResolveType(Word)`: The `IsResolveType()` method is a predicate method that returns a "true" or "false" value which indicates whether the `Word` class parameter is a resolve type for the collection of `Word` class specification instances stored within the `ParseBin` specification. A `Word` class parameter of the resolve type

indicates that a relationship between the natural language word type of the `Word` class parameter and the natural language word type of the words stored within the `ParseBin` specification instance may be formed.

The `AdjectiveBin` specification of the `ParseBin` interface, for example, returns a "true" value for a parameter that is an instance of the `Noun` class when the `IsResolveType()` method is invoked. The `IsResolveType()` method, as implemented for the `AdjectiveParseBin` class, indicates that an association may be formed between an adjective and a noun.

The `IsResolveType()` method provides a generic test that determines whether two word types are allowed to form an association or not.

- `AddSubjectAssociations(List<Word>)`: The `AddSubjectAssociation()` method serves to add subject associations to the list of `Word` class specification instances stored within the current specification of the `ParseBin` interface. Subject associations indicate that the attribute variables associated with the `Word` class instances of the parameter act as storage variables. The storage variables are assigned measurement values from the evolving function models of the `Word` class specification instances stored within the current `ParseBin` interface specification.

  An instance of the `AdjectiveParseBin` specification of the `ParseBin` interface, for example, adds a single instance of the `Noun` class in a subject association to an instance of the `Adjective` class when the `AddSubjectAssociation()` method is called on the `AdjectiveParseBin` instance with a list containing the `Noun` instance as parameter. The attribute variable associated with an adjective model in a context of interpretation (refer to section 5.4.3) associated with the `Adjective` instance stored within the `AdjectiveParseBin` specification will be stored in a `NounInstance` class instance associated with the `Noun` instance passed as the parameter. The attribute variable stored within the `NounInstance` class instance will receive a measurement value from the evolving function model stored in the `Adjective` class instance for the same context of interpretation.

- `AddObjectAssociations(List<Word>)`: The `AddObjectAssociations()` method serves to add object associations to the list of `Word` class specification instances

stored within the current `ParseBin` specification instance. Object associations indicate that the attribute variables associated with the `Word` class specifications of the parameter will be treated as amplitude values (refer to section 4.3.3). The amplitude values are substituted into the evolving function models associated with the `Word` class specifications stored in the current `ParseBin` interface specification instance.

The `VerbParseBin` specification of the `ParseBin` interface, for example, adds an instance of the `Noun` class in an object association to the `Verb` class instance stored within the `VerbParseBin` instance when the `AddObjectAssociations()` method is called with a list containing the `Noun` class instance as parameter.

The attribute variables stored within a `NounInstance` associated with the instance of the `Noun` class will be substituted into the evolving function model stored within the `Verb` class instance for the same context of interpretation.

- `AddMetaData(object)`: The `AddMetaData()` method provides a means of specifying additional data for use in the parsing procedure that is not related to the word in a narrative text sentence. An example of such additional data is the contextual information from the sentences or paragraphs that surround a narrative text sentence. The contextual information may influence the parsing procedures defined in the other methods of the `ParseBin` interface for a specification instance.

  The `AddMetaData()` method is not used by any specifications of the `ParseBin` interface in the context of the present work. Additional data will, however, be required for any algorithms that use an existing context and additional information to parse words in narrative text, as is the case in the work of Wiebe [97] [98] [99] and Nakhimovsky *et al* [69].

- `IsResolved()`: The `IsResolved()` method is a predicate method that indicates whether the current instance of a specification of the `ParseBin` interface has been fully processed in terms of associations with other word groupings. If the associations between an instance of a specification of the `ParseBin` and other specification instances of the `ParseBin` interface have been determined, the current `ParseBin`

interface specification is not considered for further association checks by means of the `IsResolveType()` method.

The `ParseBin` interface facilitates the creation of complex parsing procedures in a simplified and uncluttered manner by isolating the parsing behaviours associated with a word type. The `ParseBin` interface and its associated specifications enable the parsing procedure of the CVTNG system to satisfy the overall system goal of separating user concerns. The `ParseBin` interface enables creators of parsing algorithms to redefine existing parsing procedures without affecting evolving function word models and narrative depictions.

The `ParseBin` interface also enables the CVTNG system to satisfy the design principle of conquering complexity. The complexities of language parsing are encapsulated in the specifications of the `ParseBin` interface and users who are interested in defining evolving function models and narrative depictions need not know the details of the parsing procedure used.

5.5.4 Parsing procedure

Algorithm 1 states the parsing procedure of the CVTNG system. The parsing algorithm of the CVTNG system divides narrative text into sentences and sentences into words. It groups words of similar type and determines the subject and object associations between the groupings and these are then resolved to evolving function models according to stored word-in-context definitions (refer to section 5.4.2 through 5.4.5).

Evolving function models are the output of the CVTNG parsing procedure for narrative text provided as input. If all the evolving function models for all the words in the narrative text are formed for all the specified contexts of interpretation, the CVTNG parsing procedure is complete.

- The narrative text is subdivided into sentences on line 1. Sentences are delimited by checking for full stop "." characters. The division of sentences can be extended, however, to subdivide sentences when conjunctions or end-of-sentence indicators are encountered. The CVTNG parsing algorithm treats narrative text sentences as atomic verb sentences (refer to definition 3.20).

Algorithm 1 The CVTNG parsing algorithm

```
for all sentences in input do
  for all words in sentence do
    if current bin = ∅ then
      current bin = bin of word type
      add word to bin
    else
      if current bin = type of word then
        add word to bin
      else
        for all open bins do
          if current bin is of bin resolve type then
            set current bin as object association to bin
            close bin
          end if
        end for
        add bin to open bin list
        current bin = bin of word type
        add word to bin
        for all open bins do
          if current bin is of bin resolve type then
            set bin as subject association to current bin
            if bin is resolved then
              close bin
            end if
          end if
        end for
      end if
    end if
  end for
end for
```

- A check is performed on line 3 to determine whether an instance of a specification of the `ParseBin` interface has been created within the scope of the current atomic sentence.

- A new instance of a specification of the `ParseBin` interface is created on line 4; this corresponds to the type of the word in the narrative text. A single word in the narrative text may correspond to multiple word types. The word type is determined by the structure of the sentence. The word "wave", for example, may refer to a noun that names a body of water or to a motion of the hand dependent on the context of the word's use and the sentence structure. Rules are specified within implementations of the `IsBinType()` method of the `ParseBin` interface to determine a word's type according to sentence structure and context.

  The current CVTNG parsing procedure uses the first word type specified for a narrative text word in a context of interpretation (refer to section 5.4.1). The current parsing algorithm is therefore consistent only for the interpretation of a word defined for a single word type within a context of interpretation. The extension of the `IsBinType()` and `AddMetaData()` methods of `ParseBin` specifications will make more complex word-type determination procedures possible. A forward check is performed to determine whether a verb or a preposition is part of a verb phrase. If a group of words is determined to form a verb phrase, the group is treated as a single verb.

- An instance of a specification of the `Word` class such as a `Noun`, a `Verb`, an `Adverb`, or an `Adjective` instance is added on line 5 to the `ParseBin` interface specification. The `Word` class specification is added by calling the `AddToBin()` method of the `ParseBin` interface specification (refer to section 5.5.3). The `Word` class specification instance is created by obtaining an appropriate instance of the `AbstractFactory` class. The instance of the `AbstractFactory` class specification is used to create either an instance or multiple instances of the associated `Word` class specification. The `Word` class specification instances are created according to the specified contexts of interpretation and the stored word definitions described in sections 5.4.2 through 5.4.5.

- On line 7 a check is performed via the `IsBinType()` method of the `ParseBin` interface specification to determine whether the word type of the `Word` class instance provided as a parameter matches that of the collection of `Word` class specification instances stored within the `ParseBin` specification instance.

- An iteration is performed on line 10 over all `ParseBin` specification instances within the scope of the current sentence that are awaiting an object association. The `ParseBin` specification instances are therefore not resolved, as is determined by a call to their corresponding `IsResolved()` method. For the remaining unresolved `ParseBin` specification instances, the iteration is repeated at the end of the sentence. If the current `ParseBin` specification instance is of the resolve type of a `ParseBin` specification instance in the iteration, the current `ParseBin` specification instance is added as an object association to the unresolved `ParseBin` specification instance in the iteration. A `ParseBin` specification instance is added as an object association to another `ParseBin` specification instance by means of the `AddObjectAssociations()` method.

- The resolve type of a `ParseBin` specification instance is compared with the word type of a `Word` specification instance on line 11. The comparison is performed via the `IsResolveType()` method of the `ParseBin` interface. The comparison method can therefore vary between `ParseBin` specifications. The resolve types for the supported word types are implemented in the current CVTNG prototype, as follows:

  - Adjective: Noun.
  - Adverb: Adjective, Verb, Adverb.
  - Verb: Noun.

- A list of `Word` specification instances is added as a subject association to the `Word` specification instances contained within a `ParseBin` specification instance on line 21. The `NounInstance` class instances are added as subject associations by means of the `AddSubjectAssociations()` method, which is implemented as discussed in section 5.5.3.

The CVTNG parsing procedure pre-processes the narrative text sentences supplied as input to the algorithm above by grouping prepositions with verbs on the word string level. The grouped prepositions and verbs are treated as a single verb phrase such as "is left of" or "is above". The grouping action also sorts the verb and preposition pairs into an adverb–verb–preposition order. The narrative text string "is far above", for example, is preprocessed by the CVTNG parsing procedure to read "far is above". The adverb "far" is then added as a subject association to the verb phrase "is above".

The parsing procedure of the CVTNG system prototype is designed to parse active–tense sentences stated in the positive (all statements are deemed to be true) to an atomic verb sentence form (refer to definition 3.20). The implementation of the template method design pattern [34] by means of the `ParseBin` interface makes possible the procedures by which word groupings and associations are determined to be redefined. The redefined grouping and association checks will allow the CVTNG parsing procedure to handle other sentence types such as sentences stated in the passive voice.

The CVTNG parsing procedure can be adapted to handle passive voice sentences by setting a "passive" flag via the `AddMetaData()` method of the `ParseBin` interface. The `AddMetaData()` method may be invoked to set the "passive" flag when a past tense form of the word "be", such as "was", and a verb in the perfect tense, such as "kicked", are added to the same `ParseBin` specification instance. The "passive" flag can be checked in the logic of the `AddSubjectAssociations()` and `AddObjectAssociations()` methods of the `VerbBin` class and the behaviour of the `AddSubjectAssociations()` and `AddObjectAssociations()` methods can be altered accordingly. The behaviour of the `AddSubjectAssociations()` and `AddObjectAssociations()` methods should change to reverse the subject and object associations formed between the `VerbBin` instance and other `ParseBin` specification instances in accordance with the English grammar rules for passive voice sentences. The implementation of such behaviour was not necessary within the scope of the CVTNG prototype and was therefore not implemented.

As stated in section 5.5.3, the `ParseBin` interface makes it possible for the behaviour of the CVTNG parsing procedure to be freely extended.

The completion of the CVTNG parsing procedure, as formulated in algorithm 1 above, results in the creation of `Word` specification class instances that contain evolving

function models specified in terms of attribute variables and functions. The attribute variables are stored within instances of the `NounInstance` class in accordance with the grouping function of nouns within the evolving function models of this work (refer to section 4.3.1). The functions are represented by instances of specifications of the `Function` interface and are stored as references within instances of the `Verb` class in accordance with the evolving function models of adjectives and verbs within this work (refer to sections 4.3.2 and 4.3.3). Functions are also stored within instances of the `Adverb` class for functions specified as transformation functions according to the evolving function models of this work (refer to section 4.3.4).

The relationships between word groupings are stored as references between specifications of the `Word` class. An instance of the `Verb` class stores references to associated instances of the `Noun` class. The associated instances of the `Noun` class correspond to the subject and object nouns of the verb within a narrative text sentence for a context of interpretation. An instance of the `Adjective` class stores references to `Verb` class instances according to the evolving function word models of section 4.3.2 and the class architecture discussed in section 5.4.3. An instance of the `Adjective` class also stores a reference to an instance of the `Noun` class. The instance of the `Noun` class relates to the noun in the adjective–noun pair within the supplied narrative text for a context of interpretation (refer to section 4.3.2). A reference to an `Adverb` class instance is stored within an instance of the `Verb` class for an adverb–verb or adverb–adjective pair in a narrative text sentence for a context of interpretation (refer to section 4.3.4). The `Verb` class instance that stores the `Adverb` class instance relates to the BE-verb specified for an adjective or to a verb directly.

5.5.5 Sequencing

A story is defined as "An account or a recital of an event, or series of events ..." [2]. The CVTNG system represents textual narrative in terms of media such as graphics and sounds to depict a story. A story is successfully related if a "... series of events ..." is depicted correctly using the alternative media that constitute the interactive narrative space. Every action or event denoted by a verb in the narrative text should therefore be ordered with regard to the other actions and events related to other verbs in the

narrative text. The correct ordering of events allows the narrative text to represent the story correctly in the interactive narrative space.

An observer of an interactive narrative space is immersed if discrete actions are distinguishable and the individual actions are continuous over the period they occur in. Immersion is achieved within the CVTNG system by a sequencing algorithm that ensures the correct depiction of actions. The CVTNG sequencing algorithm must ensure that a series of distinguishable actions are represented within the interactive narrative space as a series of distinguishable changes in the representation. The algorithm must also ensure that each individual action is continuously and smoothly represented in the interactive narrative space. An action represented by a sound must not skip when played and an action represented by a graphical animation must be represented by a smooth animation without large jumps between frames.

The CVTNG sequencing algorithm must sequence actions in a way that not only allows immersion, but also integrates easily with the existing evolving function models (refer to sections 4.3.1 through 4.3.4) and measurement value resolution procedures (refer to sections 4.4.2 and 4.4.3). The algorithm easily integrates with the existing evolving function models and measurement value resolution procedures of the CVTNG system by representing the sequencing of actions as transformations on the time period (refer to definition 3.14) of evolving function models. The evolving function models correspond to a verb over contexts of interpretation (refer to section 4.3.3). The verb in turn represents an action within the narrative text.

Temporal reasoning is a field of research that formalises the notion of time to provide a means of representing temporal knowledge and of reasoning according to temporal logic [95]. Allen [4] has modelled time in terms of temporal intervals organised in a hierarchy. The temporal intervals correspond to actions or events that are related to one another and so Allen proposes a relative view of time as opposed to an absolute view. An absolute view of time disconnects the dimension of time from the events or actions that occur over a stretch of time [95]. The temporal intervals that correspond to related actions or events are arranged by means of reference intervals. Temporal intervals that do not act as reference intervals are arranged to occur before, after, or during a reference interval, or to overlap the reference interval in some way. Allen's temporal interval hierarchy

approach and other similar approaches allow for complex action or event sequences to be modelled. Temporal models can easily become complicated and require resolution schemes such as a constraint satisfaction problem (CSP) formulation [4].

A full-fledged temporal reasoning algorithm falls beyond the scope of this work and an algorithm is proposed that forms a sequence of terminate events and actions that may occur in tandem with indefinite actions or events. An interval within the context of the CVTNG algorithm refers to the time interval, T, of an evolving function model of a verb within a context of interpretation. A verb modelled within the CVTNG system (refer to section 4.3.3) is associated with multiple intervals over multiple contexts of interpretation. The intervals related to a verb are ordered to start once all of the terminate intervals associated with the previous verb have ended. Infinite intervals or evolving function models that repeat over a fixed interval (refer to equation (4.4)) are not considered when the end of a verb's intervals is determined. Evolving function models that have infinite or repeated intervals may therefore be evaluated in tandem with evolving function models that have a fixed period and no repetition. The CVTNG sequencing algorithm allows only evolving function models with terminate intervals to be executed simultaneously if the evolving function models correspond to the same verb over different contexts of interpretation.

The CVTNG sequencing algorithm is executed in two distinct phases. The detailed steps of the phases of the CVTNG sequencing algorithm are presented in algorithms 2 and 3 respectively. The first phase of the CVTNG algorithm, as presented in algorithm 2, serves to group the evolving function models associated with verbs in a context of interpretation. The evolving function models are the result of the parsing procedure presented in section 5.5. Evolving function models whose evaluation starts at the same time receive the same number from an integer sequence. The number of evolving function models that have a terminate period and are associated with the verbs grouped by a sequence number are also stored.

The second phase of the CVTNG sequencing algorithm is executed during the representation of the interactive space for a specific time, t. The verbs that correspond to the current sequence number are obtained and the evaluation of the evolving function models associated with the verbs are started. The evolving functions are evaluated to obtain

Algorithm 2 The CVTNG narrative sequencing algorithm (Phase one)

```
Current sequence number = 1
for all Verbs in narrative text do
  for all Evolving function models in context of interpretation do
    if Period of evolving function model is terminate then
      current verb is terminate
      increase number of terminate evolving functions for current sequence number
    end if
    if current verb is terminate then
      increase current sequence number
    end if
    assign current sequence number to verb
    store number of terminate evolving functions for current sequence number
    N(terminate)
  end for
end for
```

measurement values to be passed to the representation system for the transformation of the interactive narrative space. The birth time, T_b, and death time, T_d, for an evolving function are recorded when the evaluation of the evolving function is started (refer to equation (4.3)).

If the current time, t, is greater than the death time determined for an evolving function and the evaluation of the evolving function was started in a previous iteration of the sequencing algorithm, the evaluation of the evolving function ends. If, therefore, an evolving function has a period of 0 seconds, the evolving function will be evaluated for a minimum period. The length of the minimum period is determined by the actual time that passes between the start of the evaluation of the evolving function and the next iteration of phase two of the CVTNG sequencing algorithm.

A sequence number is initialised to a value of one as for the first phase of the CVTNG sequencing algorithm. The evolving functions of the verbs associated with sequence number one by the CVTNG sequencing algorithm are therefore evaluated first. If the

Algorithm 3 The CVTNG narrative sequencing algorithm (Phase two)

if interactive narrative space is initialised **then**
  Current sequence number = 1
end if
At refresh of interactive narrative space for time t
Obtain verbs for current sequence number
for all Verbs of sequence number **do**
  for all Evolving function models of verb in context of interpretation **do**
    if Evaluation of evolving function model not started **then**
      Start evaluation of evolving function model
      Evolving function birth time, T_b, = current time t
      Evolving function death time, T_d, = current time t + evolving function period T_x
    else
      if $t > T_d$ **then**
        Increase N(ended) (Terminate evolving functions ended)
      end if
    end if
  end for
end for
if N(ended) = N(terminate) **then**
  Increase current sequence number
end if

evaluation of all evolving functions of terminate length has ended, the sequence number is increased. If the second phase of the CVTNG is executed again, the evaluation of the evolving functions associated with the verbs grouped with the new sequence number is started. The execution continues until all assigned sequence numbers assigned in the first phase of the CVTNG sequencing algorithm are used by the second phase of the CVTNG sequencing algorithm. The evaluation of evolving functions of infinite or repeated period continues indefinitely.

5.6 CVTNG system procedures for assigning measurement values to attribute variables

The section describes the CVTNG subsystem dedicated to assigning measurement values to the attribute variables found within the evolving function models of natural language words presented in sections 4.3.1 to 4.3.4. The evolving function models were implemented as part of the CVTNG system in sections 5.4.2 through 5.4.5. The evolving function models are formed as a result of the CVTNG parsing procedure presented in section 5.5. The measurement values obtained by the procedures presented in this section are used to generate an interactive narrative space.

The objectives and challenges of the subsystem for assigning measurement values to attribute variables are stated in section 5.6.1. The architecture of the subsystem attribute variable measurement value resolution system in relation to other subsystems within the CVTNG system is presented in section 5.6.2. The procedure for assigning measurement values to attribute variables in the CVTNG system is presented in section 5.6.3.

5.6.1 Objectives of the CVTNG subsystem for assigning measurement values to attribute variables

The CVTNG parsing procedure as described in section 5.5.4 produces a collection of evolving function models that correspond to the words of the narrative text provided as input. The evolving functions correspond to the models specified for the words (refer

to sections 5.4.2 through 5.4.5) within the specified contexts of interpretation (refer to section 5.3). The evolving function models are stored in terms of attribute variables (refer to section 4.3.1) and functions (refer to section 4.3.3). The attribute variables are stored within instances of the `NounInstance` class (refer to section 5.4.2). The functions are implemented as specifications of the `Function` interface (refer to section 5.4.7) and are stored within instances of the `Verb` and `Adverb` classes (refer to sections 5.4.4 and 5.4.5).

The CVTNG subsystem for assigning measurement values to attribute variables receives the output of the CVTNG parsing subsystem as input. The objectives of the CVTNG subsystem are:

- to combine attribute variables and functions into the evolving function models;
- to evaluate the formed evolving function equations for a specified time value to obtain measurement values for the attribute variables;
- to process the unsolved attribute variables into a format applicable to a variable solution algorithm (Gauss-Jordan or genetic algorithm);
- to execute the measurement value assignment algorithm to obtain measurement values for previously unresolved attribute variable values.

The attribute variable solution procedure chosen for the determination of unknown attribute variable values depends on the system of equations formed when evolving function models are evaluated for a specific time value. If a system of equations cannot be solved to a single exact solution by means of the Gauss–Jordan reduction algorithm, the equations are formulated as fuzzy constraints and resolved using a genetic algorithm that attempts to obtain the most accurate approximate solution. The approach of the variable resolution procedures presented is guided by the overall system goal of consistent and accurate representation of narrative depictions.

5.6.2 Attribute variable resolution architecture

The `Vocabulary` class is central to the process of assigning measurement values to attribute variables (refer to figure 5.11). The steps of the attribute variable resolution

procedure are implemented within the `Resolve()` method of the `Vocabulary` class. The measurement values determined by the attribute variable resolution procedure are stored within instances of the `NounInstance` class. The instances of the `NounInstance` class contain the attribute variables assigned the determined measurement values. The measurement values are stored as value assignments to the attribute variables stored within the `NounInstance` class instances.

The representation system for the interactive narrative spaces utilises the `Vocabulary` class as an access point to the evolving function models of the CVTNG system. The `Vocabulary` class is initialised and provided with an input of narrative text. The parsing procedure (refer to section 5.5) is applied to the narrative text to produce evolving function models for the word in the narrative text within the specified contexts of interpretation.

The variable resolution procedures are applied to determine the measurement values for all attribute variables at a specific point in time. The representation system is able to access the determined measurement values of attribute variables by means of specifications of the `Element` interface. The specifications of the `Element` interface contain methods that allow the measurement values to be passed to the specification of the `Element` interface (refer to section 5.4.6). The measurement values are used within the specification of the `Element` interface to parameterise transformations applied to a narrative depiction when the `Transform` method of the `Element` interface is called.

The narrative depiction is stored either within the instance of the `Element` interface specification or in an external representation system. The narrative depiction is specified within the context of interpretation model for the nouns present within the narrative text. The narrative depiction is realised within the interactive narrative system when the `Invoke` method is called as transformed by the transformations stored within the specification of the `Element` interface and parameterised by the measurement values retrieved from the CVTNG system. The representation of the transformed narrative depiction within the interactive narrative space completes the generation of interactive narrative space from narrative text.

5.6.3 Attribute variable resolution procedure

This section presents the implementation of attribute variable resolution procedures within the CVTNG system. The attribute variable resolution procedure serves to obtain measurement values for the attribute variables present in evolving function equations formed from the CVTNG parsing procedures presented in section 5.5.4. The CVTNG attribute variable resolution procedure attempts to obtain a unique and precise assignment of measurement values to all attribute variables. The precise and unique measurement value assignment is obtained by means of a crisp variable resolution procedure, as initially presented in section 4.4.2. If a unique and precise measurement value assignment cannot be obtained for all attribute variables, an approximate measurement value assignment for attribute variables is obtained. The approximate measurement value assignment for attribute variables is obtained by means of a fuzzy variable resolution procedure as initially presented in section 4.4.3.

The main steps of the CVTNG procedure for assigning measurement values to attribute steps are:

- substituting default amplitude values into evolving function equations;
- evaluating formed evolving function equations at a specific point in time;
- determining a precise and unique assignment of measurement values to attribute variables;
- determining an approximate assignment of measurement values to attribute variables should no precise and unique solution be available.

Section 5.6.3.1 presents a recursive substitution algorithm for substituting default amplitude values to form evolving function equations. The algorithm evaluates the formed evolving functions equations at a specific time to obtain measurement value assignments for attribute variables. The measurement values are recursively substituted into other evolving function equations in order to obtain further measurement values until all evolving function equations are evaluated.

Section 5.6.3.2 presents the procedures within the CVTNG system that serve to process evaluated evolving function equations which contain attribute variables that

were not assigned measurement values by the recursive substitution algorithm presented in section 5.6.3.1. The procedures serve to process the unresolved evolving function equations into a coefficient matrix to be reduced by a Gaussian elimination algorithm presented in section 5.6.3.3.

A genetic algorithm is described in section 5.6.3.4; it treats unresolved evolving function equations as a series of fuzzy constraints to obtain an approximate measurement value assignment for attribute variables. The attribute variables involved are those not assigned measurement values by the algorithms presented in sections 5.6.3.1 and 5.6.3.3.

The measurement values obtained for attribute variables by means of the procedures that follow are passed as parameters to specifications of the `Element` interface. The specifications of the `Element` interface transform their associated narrative depictions according to the supplied measurement values. The representation system depicts the transformed narrative depictions in order to complete the transformation of narrative text to interactive narrative space.

5.6.3.1 Recursive substitution

This section discusses the CVTNG recursive substitution algorithm (refer to algorithm 4) in terms of the

- relation to theoretical components of the crisp variable resolution procedure presented in section 4.4.2;
- implementation of the recursive substitution algorithm within the CVTNG system architecture;
- behaviour of the recursive substitution algorithm when applied to an evolving function model specified in a context of interpretation (refer to section 4.2);
- behaviour of the recursive substitution algorithm when applied to an evolving function model containing multiple terms specified in subcontexts (refer to section 4.2);
- behaviour of the recursive substitution algorithm when applied to an evolving function model containing amplitude values specified in multiple partial contexts

Algorithm 4 The CVTNG recursive variable value substitution algorithm

1: **for all** nouns class instances **do**
2: **for all** instances of noun **do**
3: **for all** attribute variables, V_i, stored in a noun instance **do**
4: **if** attribute variable, V_i, is already assigned a measurement value **then**
5: set attribute variable value as unknown
6: return to line 3 for next attribute variable
7: **end if**
8: **if** measurement value of V_i is assigned by evolving function model $\mathcal{E}$ **then**
9: PerformSubstitutions(V_i)
10: **if** all amplitude values α_i and partial amplitude values α_{i_j} of $\mathcal{E}$ are known **then**
11: apply transformation function, Ψ_T, to time value, t, (refer to equation 4.21), if applicable
12: $V_i = \mathcal{E}(t)$
13: **end if**
14: **end if**
15: **end for**
16: **end for**
17: **end for**

(refer to section 4.2);

- determination of time values for the recursive substitution algorithm;
- treatment of adverbs of time;
- treatments of adverbs of amplitude;
- procedure for the substitution of default values;
- scenarios that lead to attribute variables not being assigned measurement values by the recursive substitution algorithm.

Algorithm 5 The PerformSubstitutions() procedure of the CVTNG recursive substitution algorithm

1: **for all** amplitude values α_i or partial amplitude values α_{i_j} in $\mathcal{E}$ **do**
2: **if** amplitude value, α_i, or partial amplitude value $\alpha_{i_j} = V_j$ **then**
3: **if** unknown attribute variable, V_j, not in processed list **then**
4: add variable, V_j, to processed list
5: return to line 4 of algorithm 4 for attribute variable V_j
6: **else**
7: {cycle in attribute variable relationships detected}
8: flag attribute variable, V_i, as unknown
9: exit procedure PerformSubstitutions()
10: **end if**
11: **if** amplitude value, α_i, or partial amplitude value α_{i_j} = constant value $a \in \mathtt{R}$ **then**
12: $\alpha_i = a$ {or α_{i_j}}
13: **end if**
14: **if** context has default amplitude value $a_{\mathscr{C}} \in \mathtt{R}$ **then**
15: $\alpha_i = a_{\mathscr{C}}$ {or α_{i_j}}
16: **end if**
17: **if** representational element associated with noun in context has default measurement value $a_r \in \mathtt{R}$ **then**
18: $\alpha_i = a_r$ {or α_{i_j}}
19: **end if**
20: **if** amplitude value, α_i, or partial amplitude value, α_{i_j}, are unknown **then**
21: calculate generic function value $f_i(t)$
22: set measurement value of attribute variable, V_i, as unknown
23: **else**
24: apply transformation function, $\Psi_{\mathcal{A}}$, to amplitude value, α_i, or partial amplitude value α_{i_j} (refer to equation 4.20), if applicable
25: **end if**
26: **end if**
27: **end for**

Theoretical components of the recursive substitution algorithm: Algorithm 4 sets out the implementation of the crisp variable resolution procedure presented in section 4.4.2 and illustrated in figures 4.1 to 4.4 in terms of the CVTNG system. The evolving function models of natural language words are specified in the CVTNG system in terms of contexts of interpretation (refer to section 4.2). An evolving function word model for an adjective, verb, or adverb is specified in terms of a word pair in a subject or object relation for a specific context of interpretation (refer to sections 4.3.2, 4.3.3, and 4.3.4). The evolving function word models are specified in terms of defined functions, amplitude values, and attribute variables. An amplitude value may be specified either as a fixed real-valued number or in terms of an attribute variable. The generic functions are functions over time implemented in an external library as specifications of the `Function` interface (refer to section 5.4.7).

The combinations of attribute variables, generic functions, and amplitude values to specify evolving function models for word pairs create a graph of attribute variable dependencies as discussed in section 4.4.2 and illustrated in figures 4.1 through 4.4. The depth-first traversal described in section 4.4.2 is implemented in the CVTNG system as the recursive substitution algorithm stated in algorithm 4.

CVTNG architecture components of the recursive substitution algorithm: The algorithm iterates over all evolving function models specified for the words of the narrative text within the specified contexts of interpretation. The evolving functions are stored as relationships in instances of the `NounInstance` class. The instances of the `NounInstance` class are grouped as references within an instance of the `Noun` class. The algorithm therefore iterates over all instances of the `Noun` class at line 1 of algorithm 4. The instances of the `Noun` class are created by the parsing algorithm stated in section 5.5.4. The algorithm iterates over the references of the `NounInstance` stored within the instance of the `Noun` class at line 2 of algorithm 4. The algorithm iterates over all the attribute variable relationships stored within a specific instance of the `NounInstance` class at line 3 of algorithm 4.

Recursive substitution algorithm applied to a word definition in a context of interpretation: An evolving function model specified for a word in a context of

interpretation is defined in terms of a single attribute variable, V_1, that receives a measurement value calculated by an evolving function. The evolving function is specified in terms of a single function, f_1, and an amplitude value, α_1. The amplitude value, α_1, is specified in terms of a real-valued number or another attribute variable. The function, f_1, is evaluated for a specific time, t, and multiplied by the amplitude value, α_1, if known. The product of $f_1(t)$ and α_1 is assigned to the original attribute variable, V_1, as a measurement value. If the amplitude value is specified in terms of another attribute variable, V_2, the algorithm is recursively called for the attribute variable, V_2, resuming at line 4 of algorithm 4. The attribute variable, V_2, is added to a list of processed attribute variables. V_2 is added to a list of processed attribute variables to prevent the repeated execution of the algorithm for an attribute variable. The repeated execution of the algorithm for an attribute variable is caused by a cycle in the graph of attribute variable relationships. The graph of attribute variable relationships, G_1, is formed for the evolving function models of the word in the specified contexts of interpretation (refer to section 4.4.2). If the attribute variable, V_2, is assigned a measurement value, the measurement value is propagated up the recursive call chain within the substitution algorithm. The measurement value would therefore be substituted as an amplitude value, α_1, and allow the measurement value of the original attribute variable, V_1, to be calculated.

Recursive substitution algorithm applied to word definition over multiple subcontexts: An evolving function model specified in a context of interpretation that is subdivided in terms of multiple subcontexts is defined by an attribute variable, V_1, that receives a measurement value calculated by an evolving function of multiple terms. Each term of the evolving function is specified in terms of a generic function f_i, and an amplitude value, α_i, specified within a subcontext related to the context of interpretation the evolving function model is specified for. The algorithm iterates over all of the terms of the evolving function model and calculates a term value by evaluating the generic function, f_i, for a specific time value, t, and multiplying $f_i(t)$ by the amplitude value, α_i. The calculated term value is added to a constant value, K (refer to section 4.4.2). If all term values are calculated, the value, K, is assigned as the measurement value of the attribute variable, V_1. If an amplitude value, α_i, is specified in terms of other

attribute variables, V_k, the substitution algorithm is recursively called for each variable resuming at line 4 of algorithm 4. If the measurement value for the attribute variables, V_k, is calculated the measurement values are propagated through the chain of recursive calls within the substitution algorithm. The measurement values are substituted as the amplitude values, α_i, which allow the term values to be calculated and summed and therefore the measurement value of attribute variable, V_1, to be calculated.

Recursive substitution algorithm applied to word definitions specified over subcontexts divided into multiple partial contexts: If the recursive substitution algorithm reaches an attribute variable that receives a measurement value from an evolving function equation whose amplitude values are specified as a sum of multiple amplitude values, the algorithm iterates over all amplitude values at line 1 of algorithm 5 and sums the individual amplitude values, α_{i_j}, to obtain a single amplitude value, α_i. The individual amplitude values, α_{i_j}, are specified within partial contexts related to a subcontext of a context of interpretation. The single calculated amplitude value, α_i, is multiplied by a value obtained by evaluating a function, f_i, at a specific time, t. The product of the calculated α_i and f_i constitutes a measurement value assigned to an attribute variable or a term value summed with other term values to determine a measurement value assigned to an attribute variable, V_1. If a partial amplitude value, α_{i_j}, is specified as another attribute variable, V_k, the algorithm is recursively called for the attribute variable, V_k. If the measurement value for V_k is determined, the measurement value is propagated as a partial amplitude value to the evolving function associated with attribute variable, V_1. If all amplitude values are determined for the evolving function equation associated with attribute variable, V_1, in a specific context of interpretation, the term values are calculated, and summed to obtain a measurement value that is assigned to the attribute variable, V_1.

Determination of time values in the recursive substitution algorithm: The generic functions, f_i, specified for evolving function models defined within contexts of interpretation and subcontexts of interpretation have time, t, as a parameter. The value of t, passed to a generic function, f_i, is determined by a time value retrieved from the representation system, the sequencing algorithm, the period of evaluation defined for the

evolving function, and adverbs of time (refer to section 4.3.4).

A time value is obtained from the independent time system of the representation system that interfaces with the CVTNG subsystem for the determination of measurement values (refer to section 5.6.2). The CVTNG sequencing algorithm determines whether an evolving function model should be evaluated based on the calculated birth time and death time of the evolving function model (refer to section 5.5.5).

An evolving function model defined within a terminate period of evaluation is specified in the form of equation (4.3). An evolving function model defined to repeat over a terminate period is specified in the form of equation (4.4). An evolving function defined to have an infinite period is specified in the form of equation (4.3) with no death time specified. The transformed time value is passed to an evolving function equation of the appropriate form and a value is calculated. The calculated value is scaled by an amplitude value, α_i, to determine a measurement value or term values that are summed to determine a measurement value.

Treatment of adverbs of time within the recursive substitution algorithm: If an evolving function model is determined to be active and the parsing procedure has determined that the adjective or verb associated with the evolving function model is associated with an adverb defined as an adverb of time (refer to sections 4.3.4 and 5.4.5), the provided time value is transformed by the transformation function defined for the adverb within the same context of interpretation as the adjective or verb (refer to equation (4.21)).

Treatment of adverbs of amplitude within the recursive substitution algorithm: If an adjective or a verb is associated with an adverb of amplitude (refer to sections 4.3.4 and 5.4.5), an amplitude value is transformed by the transformation function of the adverb, if the amplitude value is specified in the same context of interpretation, subcontext, or partial context as the amplitude value. The transformation is performed at line 12 of algorithm 4 and line 24 of algorithm 5 (refer to equation (4.20)). The value of the transformation function, $\Psi_{\mathcal{A}}$, is calculated with the amplitude value, α_i or α_{i_j}, that forms part of the evolving function model of an adjective or a verb as input. The value calculated for the transformation function, $\Psi_{\mathcal{A}}$, represents a transformed ampli-

tude value. The transformed amplitude value is subsequently used within an evolving function model to calculate a measurement value.

Substitution of default measurement values within the recursive substitution algorithm: Algorithm 4 performs a depth-first traversal over a graph structure, G, formed from the relationships between attribute variables as determined by the parsing procedure (refer to section 5.5.4) of the CVTNG system. If the algorithm reaches an attribute variable that is not associated with any other node in the graph of attribute variable relationships, the attribute variable is assigned a default value. An attribute variable is not associated with another node in the graph, G, of attribute variable relationships if it is not assigned a measurement value by an evolving function. The evolving function in turn is specified in terms of functions, f_i, and amplitude values, α_i, within a context of interpretation. The functions and amplitude values, if specified, form edges and nodes respectively within the graph, G.

If the CVTNG recursive substitution algorithm determines that an attribute variable is not assigned a measurement value, the recursive substitution algorithm attempts to assign the attribute variable a default measurement value. The algorithm attempts to retrieve a default measurement value specified for the attribute variable from the definition of the context of interpretation (refer to section 5.3) at line 15 of algorithm 5.

If no default measurement value is specified within the context of interpretation, subcontext, or partial context the attribute variable is specified in, the recursive substitution algorithm of the CVTNG system attempts to retrieve a measurement value from the narrative depiction of the associated noun that the attribute variable is grouped under (refer to section 4.3.1) at line 18 of algorithm 5.

The narrative depiction is the narrative depiction specified for the noun within the same context of interpretation as the attribute variable whose default measurement value is assigned.

If no default measurement value is determined for an attribute variable, the attribute variable is marked as "unknown" and the recursive call of the recursive substitution algorithm is ended. The recursive substitution algorithm returns to the level of the call stack that invoked the algorithm for the attribute variable determined to be unknown.

If the unknown attribute variable causes the amplitude value of a term in an evolving function equation to be unknown, the associated generic function, f_i, is calculated and stored. The attribute variable assigned a measurement value by the evolving function is also marked as unknown and propagated up the recursive call chain. The unknown measurement values for attribute variables are therefore propagated through the attribute variable graph, G.

Scenarios that cause the recursive substitution algorithm to assign no measurement value to an attribute variable: Three scenarios may arise during the execution of the recursive substitution algorithm for assigning measurement values to attribute variables that render the algorithm unable to assign a measurement value to an attribute variable. The CVTNG recursive substitution algorithm for assigning measurement values to attribute variables is unable to assign a measurement value to an attribute variable if the attribute variables is:

- used as an amplitude value in an evolving function, but is not assigned a measurement value by another evolving function and no default can be obtained;
- part of a cycle of attribute references within the graph of attribute variable relations formed by the parsing procedure (refer to section 5.5.4);
- assigned a measurement value by multiple evolving function models.

The first exception arises when an attribute variable is not assigned a measurement value by an evolving function and no default value was specified within the context of interpretation that the attribute variable is specified in or within the narrative depiction connected to an associated noun within the same context of interpretation. The measurement value of the attribute variable remains unknown after the conclusion of the recursive substitution algorithm. Any attribute variable related to the attribute variable whose measurement value is unknown will also not be assigned a measurement value as detailed in the previous subsection on the substitution of default values.

The second exception arises when attribute variables form a cycle due to the nature of their relationships within the graph of attribute variable relationships formed by the parsing procedure (refer to section 5.5.4). If the recursive substitution procedure

is called for an attribute variable, the attribute variable is added to a processed list at line 4 of algorithm 5. If the attribute variable is assigned as an amplitude value in the same recursive call chain, the attribute variable is part of a cycle in the graph of attribute variable relationships. The CVTNG recursive substitution algorithm for assigning measurement values to attribute variables determines cycles in attribute variable relationships by checking whether an attribute variable is part of the processed list at line 3 of algorithm 5. If the attribute variable is not part of the processed list, the algorithm is recursively invoked for the attribute variable. If the attribute variable is part of the processed list, the cycle is avoided and no measurement value is assigned to the attribute variable. As a result, the algorithm returns up the call chain and the unknown measurement value is propagated through the graph of attribute variable relationships. If an evolving function term has an unknown amplitude value due to an attribute variable whose measurement value is unknown, the function value is calculated and stored. The measurement value of the attribute variable that receives a measurement value from the evolving function with unknown amplitude values remains unknown.

The third and final exception occurs if an attribute variable is assigned measurement values by multiple evolving function models. The evolving function models may relate to different verb or adjective words with the same determined subject and/or object nouns or from the same verb or adjective word within different contexts of interpretation. If the recursive substitution algorithm for assigning measurement values to attribute values determines at line 4 of algorithm 4 that an attribute variable has already been assigned a measurement value, and the current assignment has not resulted from an evolving function specified within the same word instance and context of interpretation, then the measurement value of the assigned attribute variable is set as unknown.

The different evolving functions that assign measurement value to the attribute variable are recorded for further processing by the fuzzy algorithm for assigning approximate measurement values to attribute variables.

5.6.3.2 Equations from assignments

Four possibilities exist for assigning a measurement value to an attribute variable upon completion of the recursive substitution algorithm of the CVTNG system for assigning

measurement values to attribute variables (refer to algorithm 4). The attribute variable may have:

- no measurement value assignment;
- a single measurement value assignment in the form of a real-valued number;
- a measurement value assignment from an evolving function equation that contains unknown amplitude values, calculated function values, and calculated terms summed to a constant, K;
- multiple measurement value assignments in the form of real-valued numbers or evolving function with unknown amplitude values.

An attribute variable that has no measurement value assignment is the result of an attribute variable used as an amplitude value in an evolving function equation that is not assigned a measurement value. The attribute variable is not assigned a measurement value if:

- the parsing of the natural language text does not associate the attribute variable with an evolving function model;
- the attribute variable is not assigned a default value within the context of interpretation in which the attribute variable is specified (refer to section 5.3);
- the narrative depiction of the subject or object noun associated with the adjective or verb whose evolving function contains the attribute variable specifies no default value for the attribute variable in its element class (refer to section 5.4.6).

The CVTNG system for assigning measurement values to attribute variables assigns no equation to attribute variables that have no measurement value assignment. The attribute variables are variables within equations associated with other attribute variables.

An attribute variable assigned to a single measurement value in the form of a real-valued number is resolved and considered as a constant in further attribute variable resolution steps. The real-valued number assigned as the measurement value may be the result of

- a single-termed evolving function specified in a context of interpretation;
- a sum of term values, K, calculated from known function evaluations at a point in time, $f_i(t)$, and known amplitude values, α_i, for an evolving function specified over multiple terms in multiple subcontexts and partial contexts;
- a default value specified within a context of interpretation or the element class of an associated narrative depiction.

An attribute variable, V_k, assigned a measurement value by an evolving function, $\mathcal{E}_{V_k}$, that contains undetermined amplitude values, α_i, is processed by the CVTNG subsystem for assigning measurement values to attribute variables into an assignment equation. The left-hand side of the equation consists of the assigned attribute variable, V_k. The right-hand side of the equation consists of coefficient values, unknown attribute variables, and a constant value, K. The coefficient values, c_i, are determined by multiplying function values, $F_i = f_i(t)$, in the original evolving function, $\mathcal{E}_{V_k}$, with scaling values, κ_i (refer to equation (4.29)), for terms with undetermined amplitude values in the form of unknown attribute variables. Known amplitude values are multiplied by function values, F_i, and scaling values, κ_i, and summed to obtain a constant value, K (refer to equation (4.29)).

An evolving function specified within a context of interpretation only may contain a single unknown amplitude value whilst an evolving function specified over multiple subcontexts may contain multiple unknown attribute variables. The formed assignment equation may therefore have multiple unknown variables on the right-hand side. An amplitude value specified as a sum of amplitude values within multiple partial contexts that relate to a subcontext is determined by summing the specified amplitude values. If a partial amplitude value, α_{i_j}, is unknown, the function value, $F_i = f_i(t)$, for a specific time of evaluation, t, is multiplied by all of the specified partial amplitude values, and the scaling value, κ_i, to obtain multiple terms on the right-hand side of the formed assignment equation. The known partial amplitude values are multiplied by the scaling value, κ_i, and function value, F_i, of the term and summed to the constant value, K, of the formed assignment equation. The unknown partial amplitude values become unknown variables on the right-hand side of the assignment equation. The unknown variables that correspond to unknown partial amplitude values have the product of the scaling value,

κ_i, and function value, F_i, that are specified for the i^{th} term in the evolving function, $\mathcal{E}_{V_k}$, as coefficients, c_i.

The formed assignment equations that correspond to evolving functions with unknown amplitude values have the form of equation (4.29). The CVTNG subsystem for assigning measurement values to attribute variables rearranges the formed assignment equations to isolate the constant, K, on the right-hand side. A coefficient value of $c_0 = 1.0$ is multiplied with the attribute variable, V_k. The rearranged assignment equation has the form of equation (4.30).

A coefficient matrix, A, is formed in memory for the terms on the left-hand side of all the rearranged assignment equations formed from all the evolving functions containing unknown amplitude values. A column is formed in the coefficient matrix, A, for every unknown variable that corresponds to an unknown amplitude value within an evolving function. A row is formed in the coefficient matrix, A, for every rearranged assignment equation that corresponds to an evolving function with unknown amplitude values. The coefficient values are placed into the columns that match the unknown variables for the rearranged assignment equation for which the row is formed. The constant value, K, on the right-hand side of every equation is grouped into a column vector. The formed column vector is added as a column to the coefficient matrix, A, to form an augmented matrix, $[A|K]$. The augmented matrix, $[A|K]$, is supplied as input to a Gaussian elimination algorithm, to be reduced to upper triangular form.

An attribute variable determined by the parsing procedure (refer to section 5.5) to be assigned multiple measurement values by multiple evolving functions is not assigned a specific measurement value and remains unresolved. The attribute variable assignments to measurement values determined as single real-valued numbers are processed by the CVTNG subsystem for assigning measurement values to attribute variables into assignment equations. The assignment equations have the attribute variable, V_k, on the left-hand side and real-valued numbers as constant values, K, on the right-hand side. The attribute variable assignment(s) in the form of a measurement value determined by an evolving function equation with unknown amplitude values are processed into assignment equations as specified above. The assignment equations are once again rearranged to have unknown variables, V_i, with coefficient values, c_i, on the left-hand

side and a constant value, K, on the right-hand side. Multiple assignment equations are therefore formed for the attribute variable assigned multiple measurement values. The equations are not processed into a coefficient matrix as the application of a matrix reduction algorithm will show inconsistencies (refer to section 4.4.2).

The assignment equations are processed by a fuzzy algorithm for assigning measurement values to attribute variables that calculates an approximate measurement value assigned to the attribute variable.

5.6.3.3 Gaussian elimination

Algorithm 6 The Gaussian elimination algorithm

1: **for all** rows i of matrix **M do**
2: **for all** columns j in matrix row $\mathbf{M}[i]$ **do**
3: **if** $\mathbf{M}[i,j] = 0$ **then**
4: Swap matrix row $\mathbf{M}[i]$ with matrix row $\mathbf{M}[k]$ that has non-zero entry of value $\mathbf{p}$ in $\mathbf{M}[k,j]$
5: **end if**
6: **for all** non-zero matrix rows $\mathbf{M}[l]$ of value $\mathbf{r}$ below matrix row $\mathbf{M}[i]$ **do**
7: {Obtain a zero value in $\mathbf{M}[l,j]$}
8: matrix row $\mathbf{M}[l]$ = matrix row $\mathbf{M}[l]$ + $-\mathbf{r}/\mathbf{p}$(matrix row $\mathbf{M}[i]$)
9: **end for**
10: **end for**
11: **end for**

Algorithm 6 specifies a Gaussian elimination algorithm that is implemented as part of the CVTNG subsystem for assigning measurement values to attribute variables. The Gaussian elimination algorithm is applied to an augmented matrix, $[A|K]$, formed by the attribute variable resolution procedures of the CVTNG system (refer to section 5.6.3.2) to reduce the augmented matrix, $[A|K]$, to upper triangular form (also called row–echelon form). If an upper triangular form of $[A|K]$ is obtained after the Gaussian elimination algorithm completes itself, the variables that correspond to the columns are determined by solving the equations that correspond to the upper triangular form of

$[A|K]$. The process of back substitution starts with the equation that corresponds to the bottom row of the upper triangular form of $[A|K]$ and substitutes the value in the final (augmented) column as the value of the variable that corresponds to the second to last column. The value determined for the variable that corresponds to the second to last column is substituted into an equation that corresponds to the row above the bottom row in the upper triangular form of $[A|K]$. The value of the third to last column is determined from the formed equation. The process continues until the top of the upper triangular form of $[A|K]$ is reached an all variable values are determined.

The Gaussian elimination algorithm iterates over the rows of a matrix from the top row to the bottom row. The algorithm maintains a counter, j, called the "pivot column" which corresponds to the column on the diagonal for the current matrix row, i. The algorithm exchanges one row for another with a non-zero value, $\mathbf{p}$, in the pivot column, if required. Once a non-zero value, $\mathbf{p}$, is found in the pivot column, j, for the current row, i, the algorithm proceeds to reduce all other values in the pivot column, j, that are below the current row, i, to zero values. The algorithm reduces a value, $\mathbf{r}$, below the current row in the pivot column to zero by subtracting a multiple of the current row, i, from the row below the current row, l. The multiple is calculated as $\frac{\mathbf{r}}{\mathbf{p}}$. If no pivot value is found for the current pivot counter, j, the algorithm proceeds to the next pivot value. The matrix does not reduce to upper triangular form if a pivot is not found for every value of the pivot counter, j. The algorithm terminates when the final row and a pivot value equal to the number of columns minus one are reached.

If the Gaussian elimination algorithm is unable to reduce the augmented matrix, $[A|K]$, to upper triangular form, the crisp procedure for assigning measurement values to attribute variables is unable to determine unique and accurate measurement values for all attribute variables (refer to section 4.4.2). If the reduced form of $[A|K]$ contains rows that have only zero values, an infinite number of solutions exists for the assignment equations that correspond to the augmented coefficient matrix $[A|K]$. If the reduction process of the matrix $[A|K]$ shows an inconsistency, no unique and accurate solution to the system of assignment equations exists. An inconsistency occurs when all the columns of a row have zero values, except for the augmented column that contains a non-zero value. The row therefore corresponds to an equation that states $0 = K$, where K is

a non-zero value and the left-hand side could as a result not possibly be equal to the left-hand side.

An inconsistency arises when multiple measurement values are assigned to the same attribute variable. The CVTNG system avoids the formation of the augmented matrix, $[A|K]$, if multiple values are assigned to a single attribute variable as no solution can be found using the Gaussian elimination algorithm.

The next section specifies an algorithm that treats the system of assignment equations as a fuzzy constraint satisfaction problem (FCSP) in cases where no accurate and unique assignment of measurement values to attribute variables can be found using the process of recursive substitution (refer to algorithm 4) followed by coefficient matrix formation and Gaussian elimination (refer to algorithm 6) with back substitution. The FCSP is resolved to obtain an approximate assignment of measurement values to attribute variables.

5.6.3.4 Genetic algorithm for fuzzy constraint satisfaction

Algorithm 7 The genetic algorithm for the resolution of fuzzy constraints

1: Initialise the population
2: Evaluate fitness of individuals
3: **repeat**
4: Select fittest individuals from the population
5: Perform crossover to produce offspring
6: Perform mutation on offspring
7: Calculate fitness of offspring
8: Replace least fit individuals with offspring
9: **until** termination condition
10: Return the fittest individual as the result

Algorithm 7 sets out the steps of a genetic algorithm implemented within the CVTNG system as a resolution method for a fuzzy constraint satisfaction formulation of attribute variable assignment equations. The attribute variable assignment equations are formed by the evaluation of evolving function models constructed for words according to word-

in-context definitions (refer to sections 5.5.4 and 5.6.3.1 respectively). The specific assignment equations solved as FCSPs relate to attribute variables whose values cannot be determined by the recursive substitution (refer to section 5.6.3.1) and Gaussian elimination algorithms (refer to section 5.6.3.3). The recursive substitution and Gaussian elimination algorithm are implemented as part of the crisp variable resolution procedure (refer to section 4.4.2) of the CVTNG subsystem for assigning measurement values to attribute variables.

FCSPs were chosen as the formulation of evaluated evolving function equations (EEFEs) containing unknown variable values for three main reasons (refer to section 4.4.3):

- FCSPs are not skewed, in terms of their solutions, towards equations containing large coefficients and variables, as is the case for standard optimisation problems.
- FCSPs provide a means by which priority can be attached to certain fuzzy constraints, and therefore to certain equations, evolving functions, contexts of interpretation, and words. The prioritisation of words and contexts of interpretation does not form part of the CVTNG system and therefore does not fall within the scope of this work.
- Fuzzy constraint satisfaction problems are a good fit for the resolution of evolving functions that operate in a fuzzy space [122] [113]. However, evolving functions defined within a fuzzy space do not form part of the current implementation of the CVTNG system and therefore do not fall within the scope of the present work.

A genetic algorithm was chosen to solve the FCSP constructed from EEFEs because (refer to section 4.4.3):

- FCSPs are solved as optimisation problems and genetic algorithms are proven at optimisation;
- genetic algorithms are able to represent constraints of multiple variables easily;
- genetic algorithms can easily handle variables of infinite domains as chromosomes with real-valued genes;

- genetic algorithms can be adopted to dynamic CSPs.

Algorithm 7 represents candidate solutions for assigning measurement values to attribute variables in terms of chromosomes. A chromosome is a vector of real-valued numbers within the scope of algorithm 7. A population of chromosomes is created by the CVTNG subsystem for assigning measurement values to attribute variables by iterating over all the attribute variables present in EEFEs that contain unknown amplitude values (refer to section 5.6.3.2). If an attribute variable of unknown value, V_i, is present within an EEFE, and is encountered for the first time, a new gene, g_i, is added to the chromosomes of the population of candidate solutions in algorithm 7. If the set of EEFEs that contain attribute variables of unknown value contains the unique attribute variables, $V_1, \ldots, V_k$, every chromosome contains the genes, $g_1, \ldots, g_k$ where gene, g_i, corresponds to attribute variable, V_i. An assignment of values to genes, $g_1, \ldots, g_k$, therefore corresponds to an assignment of measurement values to attribute variables, $V_1, \ldots, V_k$.

A genetic algorithm is able to optimise a solution by operating on a population of chromosomes that contain within them candidate solutions as indicated above. The genetic algorithm enhances the population by applying a process of elitism by which new chromosomes are introduced into the population by reproduction operators and only the fittest individuals are retained for the next generation. The genetic algorithm therefore requires a means of calculating the fitness of a chromosome in the population to determine the chromosomes selected for reproduction and to determine the chromosomes retained in the population for the next generation. The fitness of a chromosome in algorithm 7 is calculated as a min-aggregate of membership degrees to the fuzzy constraints that the algorithm is attempting to satisfy [51].

The first step towards calculating the membership degree of a fuzzy constraint is to substitute the values of the genes, $g_1, \ldots, g_k$, into a function, f. The function, f, represents the left-hand side of an EEFE rearranged to isolate the constant value, K, on the right-hand side of the equation, as is the case for equation (4.30). The constant value, K, is then subtracted from the value calculated by substituting the gene values, $g_1, \ldots, g_k$, as a value tuple, $\overline{x}$, into the function representing the left-hand side of the EEFE, f. $|f(\overline{x}) - K|$ represents an absolute error value, $|\epsilon|$, for the value tuple substituted, $\overline{x}$. The absolute error value, $|\epsilon|$, is scaled by dividing the absolute error value by a scaling factor,

ε. The global domains of attribute variables, $V_1, \ldots, V_k$, that correspond to the genes, $g_1, \ldots, g_k$, are determined and substituted into equations (4.40) and (4.41) to calculate the bounds of the error value, ϵ. The scaling factor, ε, is subsequently calculated from equation (4.42) as the size of the domain for the error value, ϵ. The membership degree to the fuzzy constraint that corresponds to the EEFE is equal to the absolute error value, $|\epsilon|$, scaled according to the scaling factor, ε (refer to equation (4.35)).

The first step to calculating the fitness of a chromosome is to determine the membership degree of all fuzzy constraints present within the FCSP. The fuzzy constraints of the FCSP correspond to the EEFEs that contain variables whose values could not be determined by crisp variable resolution procedures (refer to section 5.6.3.2). The membership degrees are calculated according to the values of the genes in the chromosome, as explained above. The membership value of the fuzzy restriction with the lowest membership degree is used as the fitness value of the chromosome. The min-aggregation obtained for fuzzy constraint membership degrees therefore forms the objective function that the genetic algorithm attempts to optimise.

Figure 5.12 illustrates the relationship between EEFEs, the genes of chromosomes in algorithm 7, the domain sizes of evolving function terms, the constant value, K, within EEFEs, and the calculation of the membership function values of fuzzy constraints.

Algorithm 7 calculates the fitness of all chromosomes in the population as described above and selects the chromosomes that are used for reproduction operations by means of a roulette wheel selection scheme. Roulette wheel selection builds a range that is the sum of the fitness of all chromosome values and then partitions the range according to the percentage of a specific chromosome's fitness in relation to the total fitness value. A chromosome whose fitness constitutes 10% of a total fitness of 100 and is the first chromosome to be assigned a partition will be related to the values 1 through 10, for example. The next chromosome partitioned that constitutes only 4% of the total fitness value will be related to the values 11 to 14. A uniform random number is selected and if the uniform random number falls within the partition associated with a chromosome, that chromosome is selected for reproduction operations. The roulette wheel selection scheme has a high selection pressure that leads to the rapid conversion of chromosome gene values to the same or similar values.

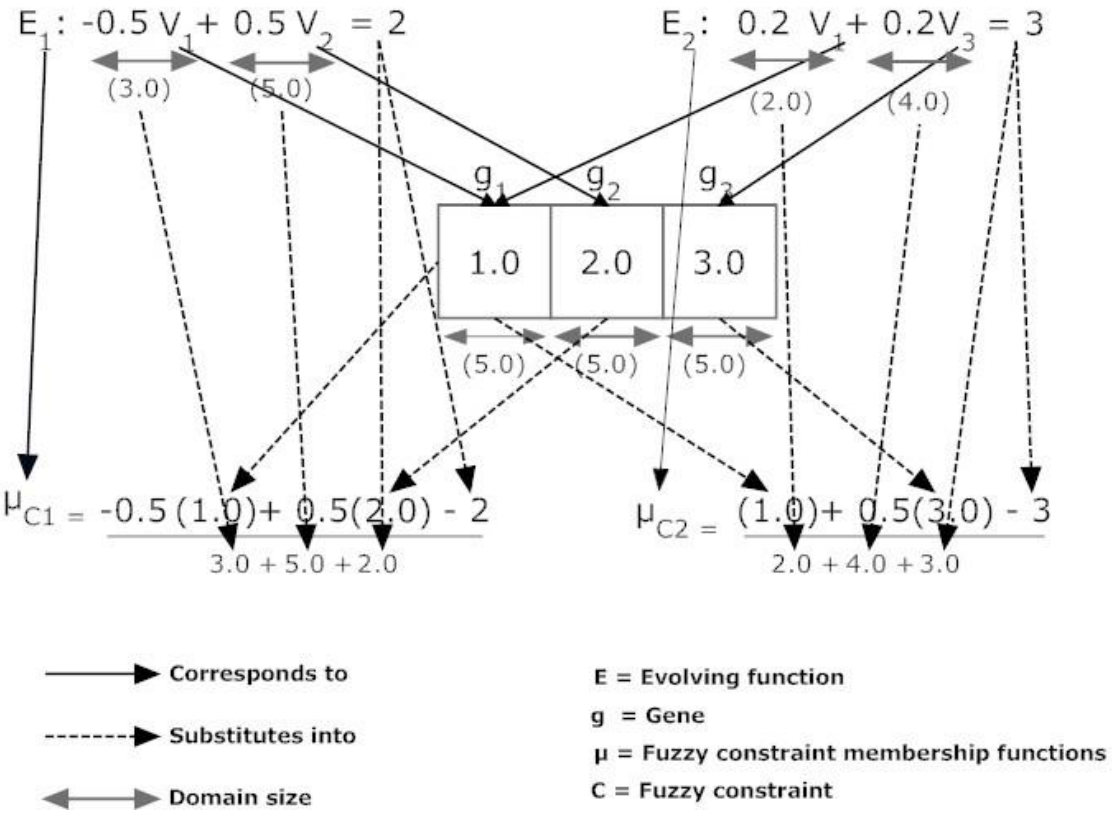

Figure 5.12: The relationships between EEFEs, chromosomes, and fuzzy membership functions

The genetic algorithm for resolving fuzzy constraint satisfaction problems proposed by Kowalczyk uses a binary gene coding with uniform crossover and mutation operators for reproduction [51]. Algorithm 7 uses real-coded genes because a real-coded genetic algorithm:

- has lower computational overhead because no encoding and decoding operations are required to convert binary representations of variable values in the genetic algorithm to real-valued numbers used as measurement values in the CVTNG system and vice versa [42];
- has been shown to be more accurate than binary encoded genetic algorithms for optimising functions on large continuous and possibly infinite intervals as is the case for evolving functions [42];

- has been shown to be more efficient for optimising functions over large continuous and possibly infinite intervals [45] [65];
- is a more natural representation of problem variables, and crossover and mutation operators specific to the problem can more easily be engineered [42].

The real-valued coding of genes in algorithm 7 requires alternative mutation and crossover operators that are suited to real-coded genes. The blended crossover (BLX-$\boldsymbol{\alpha}$) operation initially proposed by Eshelman *et al* [31] was selected as the crossover operator. The BLX-$\boldsymbol{\alpha}$ operator is a proven and popular crossover operator for real-coded genetic algorithms [41] [42] that allows for the parameterisation of a genetic algorithm's exploitation and exploration behaviour over a continuous interval. The BLX-$\boldsymbol{\alpha}$ crossover operation selects from two genes, g_i^1 and g_i^2, of parent chromosomes, p_1 and p_2, respectively, a gene value, g_i^3. The gene value g_i^3 forms part of the offspring chromosome p_3. The value selected for g_i^3 is a uniform random value from the interval $[g^- - I \cdot \boldsymbol{\alpha}, g^+ + I \cdot \boldsymbol{\alpha}]$, where $g^- = \min(g_i^1, g_i^2)$, $g^+ = \max(g_i^1, g_i^2)$, and $I = g^+ - g^-$. A crossover probability of 1.0 was chosen to allow the exploitation and exploration behaviour of algorithm 7 to be fully determined by the value $\boldsymbol{\alpha}$.

A value of $\boldsymbol{\alpha} = 0.5$ provides a balance in the levels of exploitation and exploration for a real-coded genetic algorithm that uses the BLX-$\boldsymbol{\alpha}$ operator. An $\boldsymbol{\alpha}$ higher than 0.5 causes a higher level of exploration in the genetic algorithm so that a larger portion of the search space is covered. An $\boldsymbol{\alpha}$ value lower than 0.5 causes a higher level of exploitation within the associated genetic algorithm and the chromosomes within the population more quickly converge to an optimum for the objective function of the genetic algorithm. The optimum obtained by the genetic algorithm with a high exploitation level may, however, not be the global optimum for the objective function.

A high rate of convergence to a local optimum within the chromosome population of algorithm 7 is not a completely undesirable feature when the nature of the underlying FCSP is considered.

Algorithm 7 is invoked within the CVTNG system when approximate measurement value assignments are required for attribute variables in EEFEs. Approximate measurement value assignments are required because no unique and exact measurement value assignments exist as determined by the crisp procedures for assigning measurement values

to attribute variables (refer to sections 4.4.2 and 5.6.3.1). An approximate measurement value that represents some aggregation of the multiple measurement values assigned to an attribute variable is therefore a favourable measurement value assignment. A gene value assigned by the BLX-$\boldsymbol{\alpha}$ crossover operator with an $\boldsymbol{\alpha}$ value lower than 0.5 has a higher probability of falling between the gene values of its two parents. The values assigned to the genes of offspring will therefore tend to move towards the centre of the search space, where more aggregated solutions reside. An attribute variable assigned no measurement value by an evolving function or default is not affected by the measurement value assigned and can therefore take any value within the search space that leads to the highest min-aggregation of membership degrees to the fuzzy constraints related to undetermined EEFEs. A high level of exploration within the search space is therefore not necessarily required for the attribute variables processed by algorithm 7.

The value of $\boldsymbol{\alpha}$ is supplied as an input to the CVTNG subsystem for assigning measurement values to attribute variables. The $\boldsymbol{\alpha}$ used for BLX-$\boldsymbol{\alpha}$ crossover was experimentally configured for CVTNG system applications within the scope of this work.

Random mutation [64] was chosen as the mutation operator for algorithm 7. Random mutation changes the value of a gene, g_i, by selecting a uniform random number from the domain of the associated attribute variable V_i.

Herrera *et al* [41] have shown that random mutation achieves good results within the first 100–500 generations of a genetic algorithm applied to object function optimisation. The reason for the large number of favourable results within the initial generations is ascribed to a high exploration rate. The ability to reach good solutions within a low number of generations is important in the CVTNG subsystem for assigning measurement values to attribute variables because the subsystem typically interfaces with a representation system that requires updated measurement values within real time.

The mutation rate of algorithm 7 is provided as an input to the CVTNG subsystem for assigning measurement values to attribute variables. The mutation rate was experimentally configured for the applications of the CVTNG system within the scope of this work, but typically kept lower than 0.1 as rapid convergence to optima was favoured above the retrieval of a globally optimal solution.

The initialisation and mutation of genes in algorithm 7 require a means by which the

domain of a gene, d_{g_i}, can be determined. The domain size of a gene, g_i, associated with an attribute variable, V_i, where V_i is used as an amplitude value, α_i, has the predefined size, $\alpha_i^+ - \alpha_i^-$. The domain size of a gene, g_i, associated with an attribute variable, V_i, where V_i is used as partial amplitude value, α_{i_j}, has the predefined size, $\alpha_{i_j}^+ - \alpha_{i_j}^-$. The domain of a gene, g_i, is defined by the `Minimum` and `Maximum` values specified in the context of interpretation, the subcontext, or the partial context that defines an attribute variable as an amplitude value or a partial amplitude value (refer to section 5.4.4). If no values are specified for the `Minimum` and `Maximum` attributes associated with an attribute variable used as an amplitude value, a global default domain size is used for the attribute variable, V_i, and the associated gene, g_i.

The domain size of a gene, g_k, associated with an attribute variable, V_k, where V_k stores a measurement value from an evolving function, is calculated as a step in the scaling factor calculations of a fuzzy restriction. The minimum and maximum values for the domain of, g_k, are calculated by equations (4.41) and (4.40) respectively. If a gene, g_i, is associated with an attribute variable, V_j, that occurs in multiple EEFEs, the largest domain size determined for attribute variable, V_j, is used (refer to equations (4.38) and (4.39)). The domain size of attribute variable, V_j, may be determined as for an attribute variable used as an amplitude value or a storage variable for a value determined from an evolving function. The gene values of chromosomes in algorithm 7 are selected once the domain sizes of all attribute variables, $V_1, \ldots, V_n$, associated with genes, $g_1, \ldots, g_n$, have been determined. If a gene is selected for mutation, a uniform random value from the domain of the associated attribute variable, V_i, is selected as the new mutated value of the gene, g_i.

Algorithm 7 executes within the fuzzy procedure for assigning measurement values to attribute variables until the fitness of the fittest chromosome in the population changes by less than a predetermined value, p_{var}, or reaches a maximum number of iterations, p_{gen}. The values of p_{var} and p_{gen} are provided as input to the CVTNG subsystem for assigning measurement values to attribute variables. The values of p_{var} and p_{gen} are configured experimentally for optimal results within the applications of the CVTNG system presented in this work. The values of genes, g_i, of the fittest chromosome are set as the approximate measurement values of the associated attribute variables, V_i. The

substitution of the gene values of the fittest chromosome, as determined by algorithm 7 as approximate measurement values of the associated unresolved attribute variables, completes the fuzzy procedure for assigning approximate measurement values to attribute variables.

5.7 Chapter summary

This chapter presented the architectural features of the CVTNG system implemented for the generation of interactive narrative space from natural language sentences. The CVTNG system utilises the computational verb-based evolving function models of natural language words presented in chapter 4 to obtain measurement values that parameterise the generation of interactive narrative space. The interactive narrative space is realised in terms of external media such as graphics, sound files, and AI behaviours transformed according to parameter values obtained from evolving function model measurement values.

The challenges posed to the creation of a computational model for natural language were stated, because natural language is an implicit, inconsistent, and ambiguous medium of knowledge transfer that is difficult to model computationally. The design principles of the CVTNG architecture were presented in the light of the challenges to modelling natural language as the encapsulation of complexity in evolving function models, the correct and consistent depiction of narrative representations, the separation of system user concerns, and the re-use of artifacts created within the system.

The architectural components of the CVTNG system that relate to a context of interpretation as presented in chapter 4 were described. The application of context of interpretation as a grouping mechanism for evolving function models was stated. The representation of contexts of interpretation within the CVTNG system was stated in terms of their storage as XML files, their representative classes in the CVTNG code base, and the related interface features. Transformations were presented as a subcomponent of contexts of interpretation as a means whereby the results determined from the evolving function models of words are transferred to the interactive narrative space.

The architectural components of the CVTNG system that relate to the evolving

function models of words presented in chapter 4 were discussed. The grouping of evolving function models for words in terms of contexts of interpretation was presented.

The subdivision of the evolving function models of words was presented. Subcontexts serve to subdivide evolving function models among function terms. Partial contexts serve to expand single amplitude values into a sum of partial amplitude values.

The XML storage structure and class representation of nouns, adjectives, verbs, and adverbs were presented. Element classes were presented as an architectural feature that serve as an adapter between the evolving function models of the CVTNG system and the representational system used to realise the interactive narrative space. The element classes are generated to perform transformations on an associated representational feature as parameterised by measurement values retrieved from the evolving function models of the CVTNG system. The function interface was presented as a subcomponent utilised within evolving function models of the CVTNG system. The function interface allows for the independent and generic implementation of functions utilised within the evolving function models of words.

The architectural features and algorithms of the CVTNG parsing system were presented. The system receives narrative text as input and constructs evolving function models according to the evolving function models defined for the words in the narrative texts and the contexts of interpretation specified. The steps in the parsing procedure were presented as the subdivision of narrative text in terms of words, the determination of the subject and object relationships between words, and the construction of evolving function models based on word relationships and the evolving function models defined for words. The architectural features of the CVTNG parsing system were presented in terms of storage of the relationships between specifications of the `Word` class for the word types supported. Parse bins implemented in terms of the `ParseBin` interface were presented as an implementation of the template method design pattern that allows the intricacies of language parsing to be captured within implementations of the `ParseBin` interface to define the parsing behaviour followed for a specific word type.

The parsing procedure implemented within the CVTNG parsing subsystem was presented as an algorithm that groups words of similar types, determines the relationships between different word groupings, and constructs evolving function models for the rela-

tionships between word groupings. A sequencing algorithm was discussed that sequences terminate actions as described by verbs and adjectives that have evolving functions of a terminate period. The sequencing algorithm ensures that terminate actions are initiated in the sequence specified within the narrative text provided to the parsing procedure as input.

The CVTNG subsystem for assigning measurement values to attribute variables was presented. The objectives of the procedures for assigning measurement values to attribute values were presented as the formation of evolving function equations, the evaluation of evolving function equations to obtain measurement values, process the evolving function equations that contain unresolved attribute variables into a format applicable to a variable resolution algorithm, and execute the variable resolution algorithm applicable to obtain measurement values for the remaining unresolved attribute variables.

The architectural features of the CVTNG system related to assigning measurement values to attribute variables were presented in relation to the representation and parsing subsystems. The steps in the procedure for assigning measurement values to attribute variables were set out as the substitution of default values for amplitude values in the formed evolving functions, the evaluation of formed evolving function models by means of a recursive substitution algorithm, the formation and reduction of a coefficient matrix representing unresolved attribute variables to determine their exact and unique measurement values, and the execution of a genetic algorithm to determine approximate measurement values for remaining unresolved attribute variables.

The details were presented of the implementation of the recursive substitution algorithm for the evaluation of evolving functions, as was the procedure for forming a coefficient matrix. The coefficient matrix relates to attribute variables that remain unresolved after the recursive substitution algorithm is executed. The details were given of a Gaussian elimination algorithm that reduces the formed coefficient matrix to upper triangular form. The reduction of the coefficient to upper triangular form makes it possible to determine measurement values for the unresolved attribute variables related to the columns of the coefficient matrix.

The details of a genetic algorithm were presented: it allows for the determination of approximate measurement values for attribute variables. The algorithm is executed for

attribute variables that cannot be assigned exact measurement values due to conflicting measurement value assignments, the absence of defaulted or determined amplitude values in evolving functions, and cyclical relationships between attribute variables.

The architectural features of the CVTNG system have been discussed in terms of their design considerations, storage structures, class representations, relationships, and integration with external components.

Chapter 6 presents and discusses empirical results determined by the application of the CVTNG system to a series of English natural language sentences to generate an interactive narrative space.

Chapter 6

Empirical results of applying the CVTNG system to natural language sentences

The backdrop, goals, theoretical principals, and architecture for a system that enables interactive narrative space to be generated from narrative text has been presented. This chapter tests the computational verb theory interactive narrative generation (CVTNG) system against a series of natural language sentences to empirically demonstrate that the system is able to generate interactive narrative space from natural language sentences by means of computational verb theory (CVT) models and freely associated media. The chapter provides basic examples of natural language sentences stating objects in static, periodic, and dynamic relationships and describes the actions of the CVTNG system for every example. The examples are then extended by introducing modifiers in the form of adverbs, inconsistencies, and examples on a larger scale. The extension of the illustrative examples serves to prove empirically that the CVTNG system is a robust tool for the generation of interactive narrative space from natural language sentences.

Section 6.1 presents natural language sentences that relate objects in static relationships. The sentences are translated to a visual representation of the static relationships stated to exist between the objects in the natural language sentences provided as input. The static relationships between objects are modelled as static computational verbs (refer

to definition 3.15).

The implementation of evolving function models for the supported word types (nouns, adjectives, verbs, adverbs) are described in terms of their grouping and subdivision according to contexts of interpretation, subcontexts, and partial contexts. The behaviour of the parsing subsystem, the system for assigning measurement values to attribute variables, and interaction with the representation system are described.

The static examples are extended by the addition of adverbs to the natural language sentences to show the effect of modifiers on the computational verb models of the associated natural language words. Inconsistencies are introduced to the natural language sentences to illustrate how approximate measurement values are determined within the subsystem for assigning measurement values to attribute variables whose exact measurement value cannot be determined.

Section 6.2 presents natural language sentences that relate objects in relationships which repeat over a period of time. The sentences are translated to a visual interpretation that varies over time to reflect the stated relationships between the object in the provided natural language sentences. The periodic actions described by the natural language sentences are modelled as centre computational verbs (refer to definition 3.18). The grouping and subdivision of the evolving function models for the supported word types are stated in terms of contexts of interpretation, subcontexts, and partial contexts. The actions of the parsing subsystem, the subsystem for assigning measurement values to attribute variables, and the interaction of the CVTNG system with the representation system are described once again. The periodic examples are extended by the addition of adverbs to the associated natural language sentences that illustrate the effect of modifiers on the evolving function models of periodic actions. Inconsistencies are added to the natural language sentences that describe periodic actions to illustrate the determination of approximate measurement values for attribute variables that cannot be assigned exact measurement values.

Section 6.3 presents natural language statements that describe dynamic actions and relationships between objects over time. The sentences are translated to a visual representation of the changes in the objects over time as described in the provided natural language sentences. The dynamic actions described by the natural language sentences

are modelled as node computational verbs (refer to definition 3.16). The grouping and subdivision of the evolving function models for the supported word types are stated according to contexts of interpretation, subcontexts, and partial contexts. The actions of the parsing subsystem, the subsystem for assigning measurement values to attribute variables, and the interaction between the CVTNG system and the representation system are discussed in relation to the dynamic examples. The examples are once again extended by the introduction of adverbs to the natural language sentences to illustrate the effect of modifiers on the associated evolving function models of the dynamic actions described. Inconsistencies are introduced into the natural language sentences that describe dynamic actions in order to illustrate how approximate measurement values are determined for attribute values that cannot be assigned exact measurement values.

6.1 Static example

The first example applies the CVTNG system to the simple natural language sentence, "The red cube is on the blue cube.". The sentence is modelled in terms of static computational verbs (refer to definition 3.15) within the CVTNG system that relate to the adjectives "red" and "blue" and the verb phrase "is on". The words are represented in the interactive narrative space by 3D model associated with the noun "cube".

Section 6.1.1 states the contexts of interpretation, subcontexts, and partial contexts defined for the static examples presented in this section. The contexts are defined in terms of property definitions (refer to section 5.3). The context definitions are stated in terms of context, attribute variable definitions, and function definitions. Section 6.1.2 states the evolving function models of the words "red", "blue", and the verb phrase "is on" in terms of context of interpretation, subcontexts, and partial contexts. The actions of the parsing procedure as implemented in the CVTNG parsing subsystem are described in section 6.1.3. The action of the subsystem for assigning measurement values to attribute variables are described in section 6.1.4. The interaction between the evolving function models of the CVTNG system and the representation system are described in section 6.1.5. In section 6.1.6, adverbs are introduced to the sample sentences of this section to illustrate the effect of modifiers on the evolving function models of the

static relationships described. Section 6.1.7 introduces inconsistent natural language statements in order to describe how approximate measurement values are assigned to attribute variables whose exact measurement values cannot be determined.

6.1.1 Contexts

The sample sentences in this section are modelled in five contexts of interpretation:

- The first is labelled "SpatialX" and serves to model the displacement of an object in a 3D visual space along the x-axis.
- The context of interpretation labelled as "SpatialY" serves to model the displacement of an object in a 3D visual space along the y-axis.
- The "GreenColour", "BlueColour", and "RedColour" contexts of interpretation models the colour of an object in a visual space with regard to the green, blue, and red colour channels respectively. These channels correspond to the representation of colour within graphics rendering APIs such as ® DirectX ®.

The "SpatialX" and "SpatialY" contexts have a single related subcontext labelled "Start" as the evolving function models of this section are implemented as static computational verbs with a single function term.

The "Start" subcontext is so labelled because the function term specified in the subcontext models the starting position of an object in a visual space. The examples of this section are static and no additional terms are required in the evolving function models.

The subcontext "Start" has six related partial contexts labelled "SelfX", "SelfY", "ReferenceX", "ReferenceY", "OtherX", and "OtherY". The "SelfX" and "SelfY" partial contexts serve to model partial amplitude values related to the width and depth of an object that forms the subject in a natural language sentence. The "ReferenceX" and "ReferenceY" partial contexts serve to model partial amplitude values related to the point of reference for the displacement of an object along the x-axis and y-axis of a visual space. The "OtherX" and "OtherY" partial contexts serve to model partial amplitude values related to the width and height of an object that forms the object in a natural

language sentence. The contexts of interpretation, subcontexts, and partial contexts specified above allow the visual displacement relationship between objects in a visual space to be modelled in terms of evolving functions.

The objects whose relationships are described by the evolving function models are the subject and object nouns of a natural language sentence. The visual displacement relationships modelled between the objects are specified by natural language adjectives or verbs.

The contexts are implemented by specifying context relationships and properties. A context relationship is defined as the relationship between a context of interpretation and its subcontexts, and also between a subcontext and the partial contexts related to it (refer to section 5.3). The properties defined within a context of interpretation, a subcontext, or a partial context are specified in terms of an attribute variable and an associated representation transformation (refer to section 5.3.4).

The attribute variables defined within context properties are utilised in the construction of evolving function models by the parsing subsystem. Attribute variables are used within the CVTNG subsystem for assigning measurement values to attribute variables in order to store measurement values calculated from the evolving function models of adjectives and verbs. Attribute variables also store amplitude values, if the attribute variable in question is used as an amplitude value, α, in an evolving function model.

Transformations provide a means by which the measurement values determined from evolving function models are transferred to narrative representations. The transformations are implemented as specifications of the `Transformation` interface in a separated assembly.

Figure 6.1 illustrates the relationships between the contexts defined above and the properties defined for the contexts of interpretation, subcontexts, and partial contexts. Contexts of interpretation, subcontexts, and partial contexts are shown as squares. Properties defined for the contexts, subcontexts, and partial contexts are shown as rounded squares. The contexts, subcontexts, and partial contexts were implemented as discussed above and illustrated in figure 6.1. The next section describes the implementation of evolving function models for the words of the example sentences within the defined contexts, subcontexts, and partial contexts.

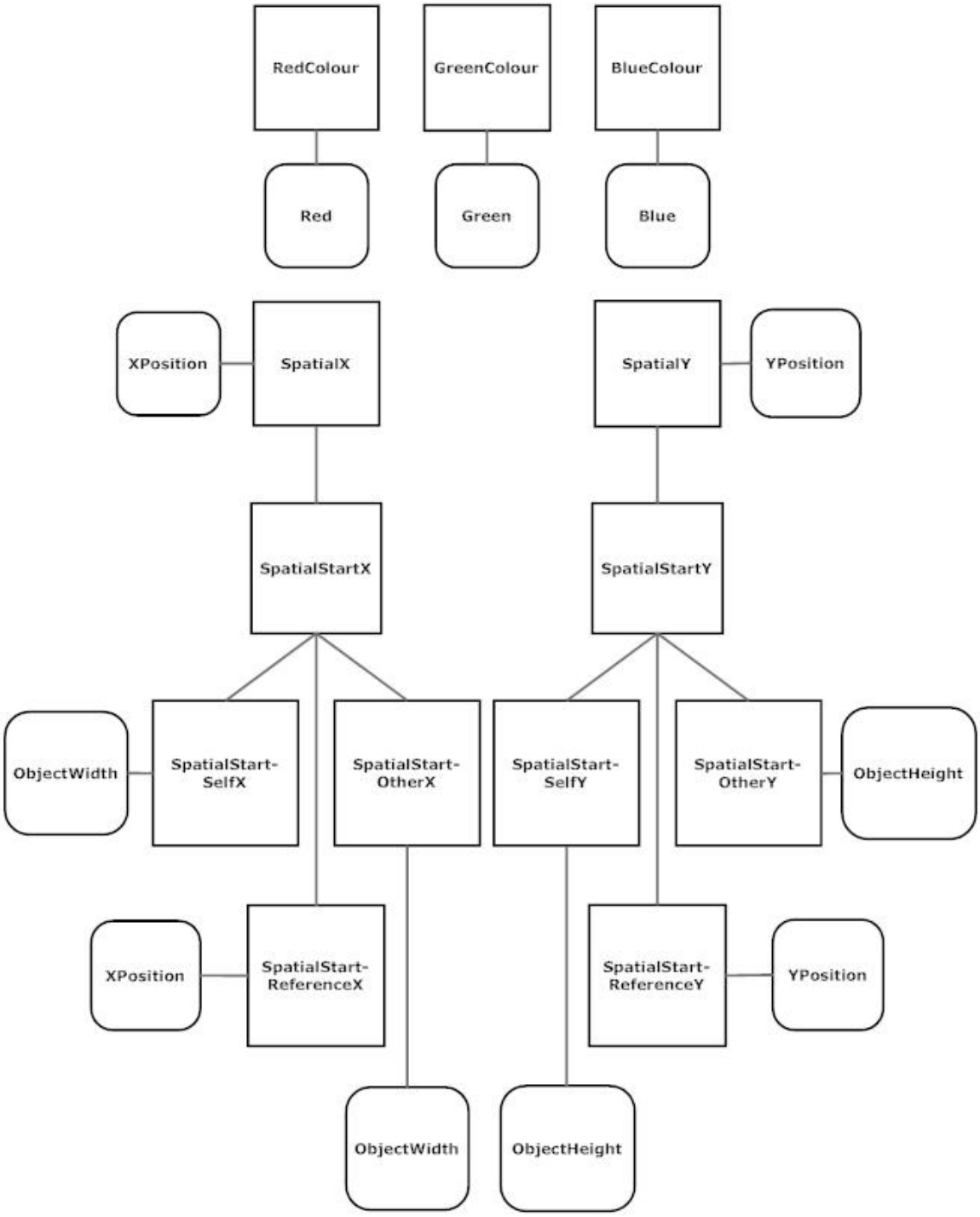

Figure 6.1: The contexts, subcontexts, partial contexts, and properties specified for the grouping and subdivision of evolving function models for the sample sentence, "The red cube is on the blue cube."

6.1.2 Words

This section sets out the evolving function models implemented for the static sample sentences in this section. The evolving function models for the supported word types are also presented. The article "the" is not supported by the CVTNG system and is therefore not treated within the scope of the present work. The article "the" does not affect the relationships between the objects in the natural language sentences that are presented as examples and is consequently not considered further.

Table 6.1 summarises the definitions of the words "Red" and "Blue" and the verb phrase "is on" in the contexts, subcontexts, and partial contexts presented in section 6.1.1:

- The "Word" column lists the individual natural language words whose evolving function model components are listed in the remaining columns.
- The "Context" column lists the context of interpretation, the subcontext, or the partial context in which the evolving function components specified in the remaining columns are defined. The abbreviation "SS" is short form of "SpatialStart" for the context names stated in table 6.1.
- The "Storage variable" column lists the attribute variable (if applicable) that stores the measurement value obtained from an evolving function model whose details are specified in the remaining columns of table 6.1 for the same table row.
- The "BE-verb" column lists a verb word (if applicable) whose evolving function model is used as the evolving function model of the associated adjective listed in the "Word" column of the same table row.
- The "Scaling factor" column lists the scaling value, κ, applicable to the evolving function term whose components are specified on the same table row in the remaining columns.
- The "Generic function" column lists the generic function, f, that forms part of a term specified for an evolving function model. The term of the evolving function

Word	Context	Storage variable	BE-verb	Scaling factor κ	Generic function f	Amplitude value α	Transformation function $\Psi(x)$
be	RedChannel			$\kappa = 1.0$	$f_{one}(t) = 1$		
	BlueChannel			$\kappa = 1.0$	$f_{one}(t) = 1$		
	GreenChannel			$\kappa = 1.0$	$f_{one}(t) = 1$		
red	RedChannel	Red	be			0.7	
blue	BlueChannel	Blue	be			0.7	
is on	SpatialY	YPosition					
	SpatialStartY				$f_{one}(t) = 1$		
	SSReferenceY			$\kappa_1 = 1.0$		YPosition	
	SSSelfY			$\kappa_2 = 0.5$		Height**	
	SSOtherY			$\kappa_3 = 0.5$		Height	
	SpatialX	XPosition		$\kappa = 1.0$	$f_{one}(t) = 1$	XPosition	
dark	RedChannel						$\Psi_{half}(t) = 0.5x$
	BlueChannel						$\Psi_{half}(t) = 0.5x$
	GreenChannel						$\Psi_{half}(t) = 0.5x$
light	RedChannel						$\Psi_{two}(t) = 2.0x$
	BlueChannel						$\Psi_{two}(t) = 2.0x$
	GreenChannel						$\Psi_{two}(t) = 2.0x$
is above	SpatialY	YPosition					
	SpatialStartY				$f_{one}(t) = 1$		
	SSReferenceY			$\kappa_1 = 1.0$		YPosition	
	SSSelfY			$\kappa_2 = 1.0$		Height**	
	SSOtherY			$\kappa_3 = 1.0$		Height	
	SpatialX	XPosition		$\kappa = 1.0$	$f_{one}(t) = 1$	XPosition	
far	SpatialX						$\Psi_{oneandhalf} = 1.5x$
	SpatialY						$\Psi_{oneandhalf} = 1.5x$
is to the left of	SpatialX	XPosition					
	SpatialStartX				$f_{one}(t) = 1$		
	SSReferenceX	XPosition		$\kappa_1 = 1.0$		XPosition	
	SSSelfX	XPosition		$\kappa_2 = -1.0$		Width**	
	SSOtherX	XPosition		$\kappa_3 = -1.0$		Width	
	SpatialY	YPosition		$\kappa = 1.0$	$f_{one}(t) = 1$	YPosition	
is to the right of	SpatialX	XPosition					
	SpatialStartX				$f_{one}(t) = 1$		
	SSReferenceX			$\kappa_1 = 1.0$		XPosition	
	SSSelfX			$\kappa_2 = 1.0$		Width**	
	SSOtherX			$\kappa_3 = 1.0$		Width	
	ReferenceY	YPosition		$\kappa = 1.0$	$f_{one}(t) = 1$	YPosition	

Table 6.1: Static word definitions

model corresponds to the context of interpretation or subcontext listed on the same table row as the generic function, f.

- The "Amplitude value" column lists an amplitude value, α, specified as an amplitude value or a partial amplitude value for an evolving function model term. The evolving function term corresponds to the subcontext or partial context listed in the "Context" column on the same table row as the amplitude value, α.

- The "Transformation function" column lists a transformation function, Ψ, that corresponds to the adverb word specified in the "Word" column of the same table row and the context of interpretation, subcontext, or partial context listed in the "Context" column of the same table row. The transformation function, Ψ, is applied to the generic value, x, supplied in the form of an amplitude or time value.

The annotation "**" indicates an amplitude value, α, specified within the evolving function model of a verb whose subject and object noun are the same noun word. If a default value is retrieved for the amplitude value, α, from a representational element, the default value is obtained from the representational element associated with a subject noun (refer to section 5.6.3.1). The subject noun corresponds to the verb whose evolving function model components are listed in the same table row as the amplitude value, α, that has the annotation "**". The annotation "**" is specified by the attribute `Reflective` within the CVTNG system (refer to section 5.4.4).

The evolving function word models specified in table 6.1 were implemented with the "Word Editor" user interface component of the CVTNG system presented in section 5.4.1. The functions stated in table 6.1 were implemented as specifications of the `Function` interface in the function assembly described in section 5.4.7.

The amplitude values, α_i, that are tagged with the "**" symbol in the variable definitions tables of this chapter are defined as reflective amplitude values in their respective evolving function models (refer to section 5.4.4). The amplitude values are defined as reflective amplitude values by setting the `Reflective` property when the amplitude value is specified in a context of interpretation, a subcontext, or a partial context. If an amplitude value is flagged as a reflective amplitude value, the `Reflective` flag serves as an indication that the subject noun of the associated verb word also serves as the object

noun of the verb word in the narrative text. If a default value is required for an amplitude value flagged as a reflective amplitude value and no default is specified within the context of interpretation, the default value is retrieved from the narrative depiction associated with the subject noun of the atomic verb sentence (refer to definition 3.20).

The noun "cube" was associated with a representational element in the form of a 3D graphical model in the "SpatialX","SpatialY", "RedChannel", "GreenChannel", and "BlueChannel" contexts of interpretation (refer to section 5.4.2). A single element class was generated to correspond to the noun "cube" that served as the access point of the CVTNG system to the representation system. The `Inialize` and `Invoke` methods of the `Element` interface were implemented in the generated element class to define and display the vertices of a 3D cube, and in this way, serve as a representational element in the interactive narrative space. The remaining methods of the `Element` class were implemented by code generated as part of the generation of the element class.

6.1.3 Parsing

The parsing procedure specified by algorithm 1 processes the words of the sentence "The red cube is on the blue cube." sequentially. The parsing procedure, as applied to every word modelled in the CVTNG system, is described and the parsing procedure detailed in algorithm 1 is referenced throughout the steps detailed:

- The adjective "red" is implemented as a static computational verb in the "RedChannel" context. The parsing algorithm creates an instance of the `Adjective` class (refer to section 5.4.3) and stores a reference to the instance of the `Adjective` class within an `AdjectiveParseBin` specification of the `ParseBin` (refer to section 5.5.3) interface.

- The second word processed in sequential order by the parsing procedure is the noun "cube". The noun "cube" is defined within the "SpatialX", "SpatialY", "RedChannel", "GreenChannel", and "BlueChannel" contexts of interpretation. The word "cube" is determined to be of a noun type and therefore not similar to the words of the adjective type stored within the instance of the `AdjectiveParseBin` class. The instance of the `AdjectiveParseBin` is flagged as "closed".

An instance of the `Noun` class is created and a reference to the instance of the `Noun` class is stored in a newly created instance of the `NounParseBin` specification of the `ParseBin` interface. An instance of the `NounInstance` class is created and a reference is stored within the instance of the `Noun` class. A generated element class is instanced and stored within the instance of the `Noun` class for every context of interpretation the noun is defined in. The contexts of interpretation stated above share the same element file and consequently multiple references to the same element file are stored within the newly created instance of the `Noun` class.

The instance of the `NounParseBin` class is determined by the parsing procedure to be of the resolve type for the `AdjectiveParseBin` class (refer to section 5.5.3). A reference to the `NounInstance` class instance is stored as a subject relation to the instance of the `Adjective` class stored within the `AdjectiveParseBin` instance. The instance of the `NounInstance` class is stored within the instance of the `Noun` class which in turn is stored in the `NounParseBin` instance.

- The next iteration of the parsing procedure encounters the word "is" and determines that no definition for the word is specified (refer to table 6.1). The parsing procedure scans forward and finds the word "on". The parsing procedure determines that the verb phrase "is on" is modelled as a verb within the contexts of interpretation "SpatialX" and "SpatialY" respectively. The evolving function models defined for the verb phrase "is on" within the contexts of interpretation "SpatialX" and "SpatialY" are created and stored within newly created instances of the `Verb` class. The instances of the `Verb` class relate to the evolving function models specified in the "SpatialX" and "SpatialY" contexts of interpretation specifically. The function, f_{one}, used in the subcontexts "SpatialStartX" and "SpatialStartY" and implemented as a specification of the `Function` interface, is instanced. References to the specification instance of the `Function` interface that implements the function, f_{one}, are stored within the instances of the `Verb` class that relate to the contexts "SpatialX" and "SpatialY" respectively.

  The parsing procedure determines that the instance of the `Verb` class is not of the word type stored within the `NounParseBin` specification instance of the `ParseBin`

interface. The `NounParseBin` instance is flagged as "closed" and an instance of the `VerbParseBin` specification of the `ParseBin` interface is created. References to the `Verb` class instances are stored in the newly created instance of the `VerbParseBin` class. The `VerbParseBin` is determined to be of the resolve type of the `NounParseBin` instance. A reference to the instance of the `Noun` class stored within the `NounParseBin` instance is added as a subject relation in the `Verb` class instance stored within the `VerbParseBin` instance.

- The parsing procedure reaches the second adjective of the sample sentence, "blue", defined in the context of interpretation "BlueChannel". An instance of the `Adjective` class is created that stores the evolving function model defined in the context of interpretation "BlueChannel". The instance of the `Adjective` class is determined not to be of the word type stored in the open `VerbParseBin` instance. The `VerbParseBin` class instance is not flagged as closed because verbs may have both subject and object associations. An instance of the `AdjectiveParseBin` class is created and marked as the current parsing bin.

- The second instance of the noun "cube" is reached by the parsing procedure. A new instance of the `NounInstance` class is created and a reference is stored in the original instance of the `Noun` class. A new instance of the `NounParseBin` class is created and the newly created instance of the `NounInstance` class is stored within the `NounParseBin` instance. The parsing procedure determines that the `NounParseBin` instance is of the resolve type of the `VerbParseBin` awaiting an object association. A reference to the newly created instance of the `NounInstance` class is added as an object relation to the instance of the `Verb` class stored within the `VerbParseBin` instance. The parsing procedure also determines that the `NounParseBin` instance is of the resolve type of the `AdjectiveParseBin` instance. The instance of the `NounInstance` class stored within the instance of the `NounParseBin` class is added as a subject association to the `Adjective` instance stored within the `AdjectiveParseBin` instance.

Figure 6.2 illustrates the relationships that exist between parse bins at the completion of the parsing procedure. The final step of the parsing procedure constructs the evolving

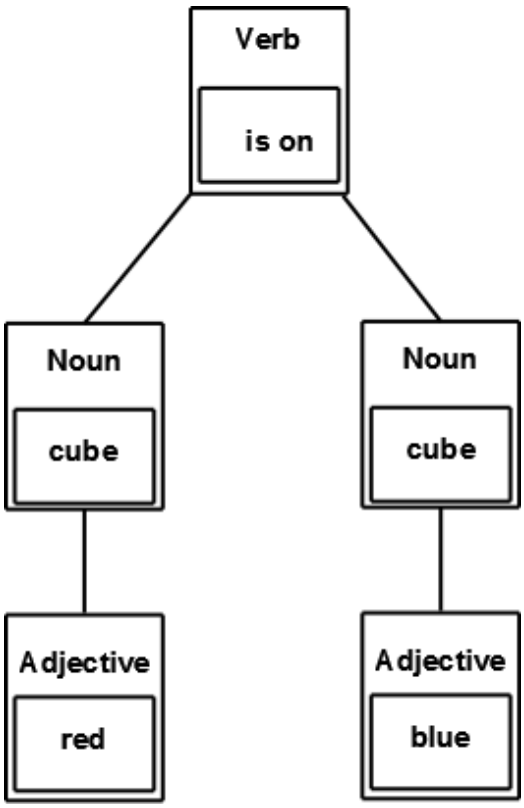

Figure 6.2: The parse bin relationships produced for the sample sentence, "The red cube is on the blue cube."

function models illustrated in figure 6.3. Attribute variables specified in the definition of the evolving function models illustrated are stored within instances of the `NounInstance` class. Functions specified in the definition of the evolving function models illustrated are stored as references to specifications instances of the `Function` interface. The references to specifications of the `Function` interface are stored in instances of the `Verb` class created during the parsing procedure.

A relationship between attribute variables "YPosition(Red)" and "ObjectHeight(Blue)" is created by specifying the attribute variable, "ObjectHeight(Blue)", as an amplitude value in the evolving function of the verb phrase "is on". The evolving function model of the verb phrase "is on" assigns a measurement value to the attribute variable "YPosition(Red)". The relationship between the attribute variables "YPosition(Red)" and "ObjectHeight(Blue)" is stored in the `NounInstance` class instance that stores the at-

tribute variable "YPosition(Red)". Other attribute variable relationships created by the formation of the specified evolving function models are stored in a similar fashion.

The parsing procedure completes with the formation of the evolving function models instanced and stored as described above. The CVTNG system passes the structure illustrated in figure 6.3 to the CVTNG subsystem for assigning measurement values to attribute variables.

6.1.4 Variable resolution

The CVTNG subsystem for assigning measurement values to attribute variables receives the structure illustrated in figure 6.3 as input and is tasked with the evaluation of the formed evolving function equations for a specific time value. The function, f_1, the amplitude values, $\alpha = 0.7$, and the scaling factor, $\kappa = 1.0$, are substituted into the template evolving function equation (refer to equation (4.3)) to obtain evolving function equations for assigning measurement values to the attribute variables "Red" and "Blue" in the "RedChannel" and "BlueChannel" contexts of interpretation respectively.

The evolving function equations are:

$$\begin{aligned} Red(Red) &= f_{one}(t)\kappa\alpha \\ &= (1.0)(1.0)(0.7) \\ &= 0.7 \\ Blue(Blue) &= f_{one}(t)\kappa\alpha \\ &= (1.0)(1.0)(0.7) \\ &= 0.7. \end{aligned} \tag{6.1}$$

The annotation (Blue) refers to an attribute variable stored in the `NounInstance` class instance associated with the noun "cube" described by the adjective "blue" in the natural language text. The annotation (Red) refers to an attribute variable stored in the `NounInstance` class instance associated with the noun "cube" that is described by the adjective "red" in the natural language text. $f_{one} = 1.0$ is a generic function used in the evolving function equations associated with the adjectives "red" and "blue". The

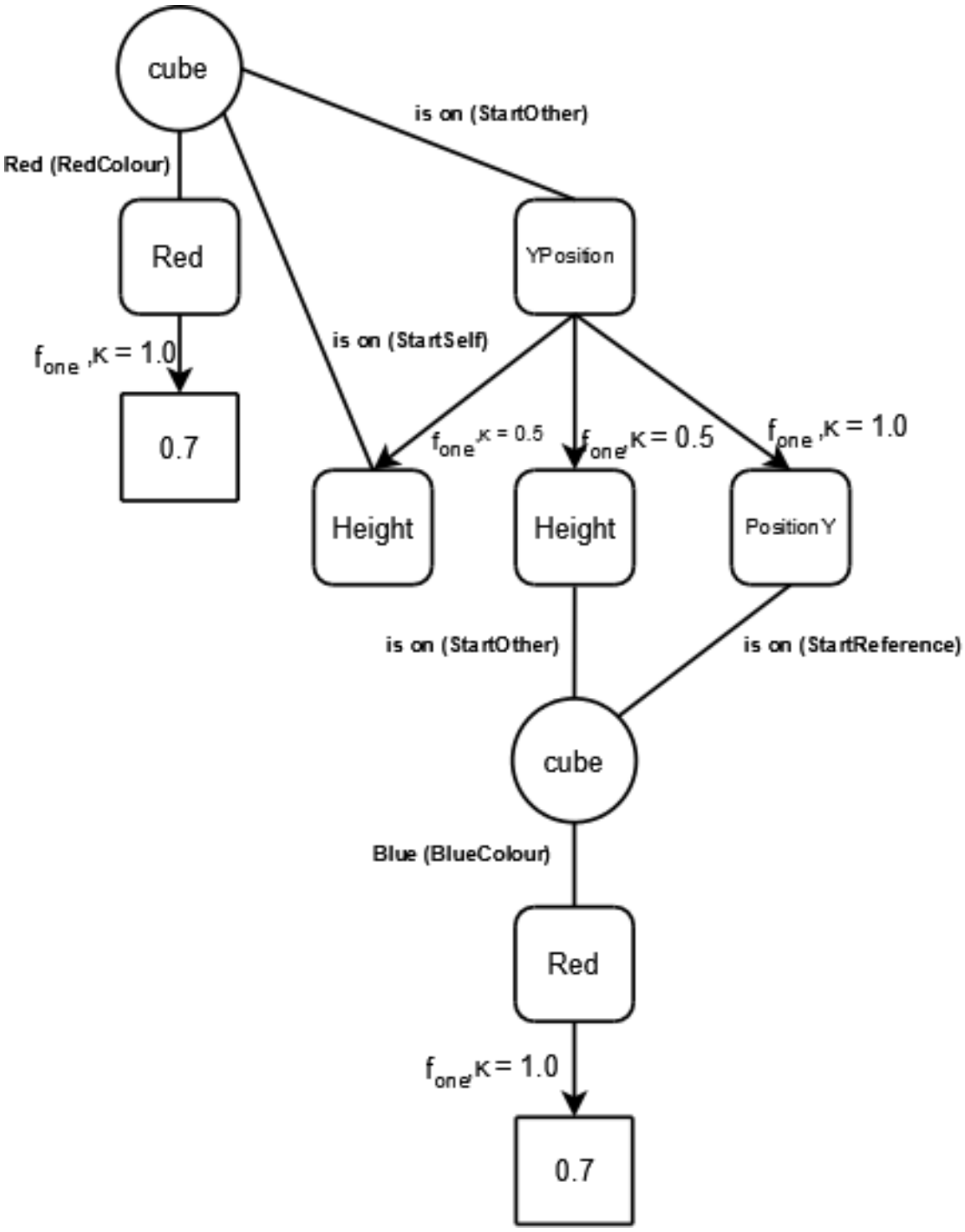

Figure 6.3: The attribute variable relationships formed by resolving the parse bins in figure 6.2

generic function, f_{one}, is as defined for equation (4.3) and specified in table 6.1. $\kappa = 1.0$ is a scaling factor as defined for equation (4.3) and specified in table 6.1. $\alpha = 0.7$ is an amplitude value that serves to scale the evolving function equations associated with the adjectives "red" and "blue". The amplitude value, α, is as defined for equation (4.3) and specified in table 6.1.

The attribute variable "YPosition(Red)" is assigned a measurement value by an evolving function equation defined in the context of interpretation "SpatialY", the sub-context "SpatialStartY", and the partial contexts "SpatialStartSelfY", "SpatialStartOtherY", and "SpatialStartReferenceY". The functions, partial amplitude values, and scaling factors defined for the evolving function in table 6.1 are substituted into the template evolving function equation with partial amplitude values (refer to equation 4.7) and the evolving function:

$$\begin{aligned} YPosition(Red) &= \kappa_1 YPosition(Blue) f_{one}(t) \\ &+ \kappa_2 ObjectHeight(Red) f_{one}(t) \\ &+ \kappa_3 ObjectHeight(Blue) f_{one}(t) \\ &= ((1.0)(0.0) + (0.5)(1.0) + (0.5)(1.0))\,(1.0) \\ &= 1.0 \end{aligned} \tag{6.2}$$

is obtained.

Algorithm 4 is executed and the attribute variables in the evolving functions equations (6.1) and (6.2) are processed and resolved recursively. The attribute variables "Red(Red)" and "Blue(Blue)" are both assigned a measurement value of "0.7" that is obtained by evaluating the function, f_{one}, and multiplying the obtained values of 1.0 with the scaling factor, $\kappa = 1.0$, and the amplitude value, $\alpha = 0.7$ specified (refer to equation (6.1)).

The recursive substitution algorithm performs a depth-first traversal over the graph of attribute variable relationships formed for the evolving function equation related to the verb phrase "is on". The traversal reaches the leaf nodes associated with the attribute variables "YPosition(Blue)", "Height(Red)", and "Height(Blue)". The attribute variables associated with the leaf nodes are not assigned measurement values by another evolving function and are therefore assigned default values. The default values are spec-

ified within the partial contexts that specify "YPosition(Blue)", "ObjectHeight(Red)", and "ObjectHeight(Blue)" as partial amplitude values for equation (6.2). The attribute variables "YPosition(Blue)", "ObjectHeight(Red)", and "ObjectHeight(Blue)" are assigned default measurement values of 0.0, 1.0, and 1.0 respectively. The recursive call that visits each of the leaf nodes terminates and the substituted partial amplitude values are multiplied by their respective scaling values, $\kappa_1, \kappa_2, \kappa_3$, and summed to obtain an amplitude value for evolving function equation (6.2). The evolving function is evaluated by multiplying the obtained amplitude values with the evaluated function value, $f_{one}(t) = 1.0$, and a measurement value of "1.0" is obtained for the attribute variable "YPosition(Red)".

The evolving function equations related to the sample sentence "The red cube is on the blue cube." are simple and the matrix formation (refer to section 5.6.3.2) and Gaussian elimination (refer to section 5.6.3.3) steps of the crisp procedure for assigning measurement values to attribute variables are not required. The determined measurement values for the attribute variables obtained from the substitution of default measurement values and the calculation of measurement values from evolving function equations are passed as parameters to the representation system.

6.1.5 Representation

The measurement values obtained by the CVTNG subsystem for assigning measurement values to attribute variables allows the representation system to transform the narrative depictions according to the measurement values obtained. The transformed interactive narrative space serves to depict the narrative text within the interactive narrative space.

The first step performed for representing natural language text in an interactive narrative space is the initialisation of the representational means. It is initialised by calling the `Initialize` method of the `Element` interface (refer to section 5.4.6). The `Initialize` method is implemented in the element class generated for the noun "cube" as specified in the contexts of interpretation "RedChannel", "GreenChannel", "BlueChannel", "SpatialX", and "SpatialY". The `Initialize` method was implemented to initialise a series of vertices and buffers associated with rendering a 3D cube in the Microsoft ® DirectX ® graphics rendering API. The `Initialize` method is invoked once and the

initialised buffers and vertices are stored within the instance of the generated element class for re-use by the representation system. If an external representation system is used, the appropriate buffers and vertices are stored within the external representation system and the instance of the generated element class stores the appropriate references to the external representation system.

The next step towards representing narrative text in the interactive narrative space is performed by invoking the `InitVars` method of the generated element class. The `InitVars` method serves to initialise the transformations applied to the representational means. The transformations serve to transform the narrative depiction associated with the generated element class. The transformed narrative depictions serves to depict the narrative text provided as input to the CVTNG system within the interactive narrative space. The transformations are initialised according to default parameter values specified for the properties that group the transformations with attribute variables in the definition of a context of interpretation (refer to section 5.3). If the actions of the CVTNG subsystem for assigning measurement values to attribute variables does not alter the measurement value of an attribute variable, the transformation defaulted in the `InitVars` method is applied to the narrative depiction associated with the generated element class.

The third step towards representing narrative text in the interactive narrative space passes measurement values obtained from the CVTNG subsystem for assigning measurement values to attribute variables to transformations that are applied to the narrative depictions that constitute the interactive narrative space. The measurement values are transferred by calling the necessary accessor methods created as part of the generation of the element class. The accessor methods receive the real-valued numbers representing measurement values as an input and initialises a transformation according to the input received. The accessor method called to pass a measurement value to the element class corresponds to a property of a context of interpretation, subcontext, or partial context. The specific property is determined to be the property that specifies the attribute variable whose measurement value is passed as a parameter to the accessor method (refer to section 5.3). The transformation initialised by the parameter value received from the passed measurement value is defined in the same context property as the attribute

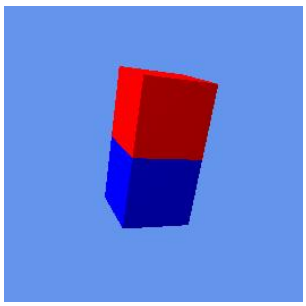

Figure 6.4: The visual output produced for the sample sentence, "The red cube is on the blue cube."

variable. If a transformation is initialised by a measurement value passed as a parameter to an accessor method, the default transformation initialised in the `InitVars` method is no longer applied to the narrative depiction.

The fourth step towards representing narrative text in the interactive narrative space transforms the narrative depiction related to an element class according to the transformations stored in the element class. The transformations are performed by invoking the `Transform` method of the `Element` interface (refer to section 5.4.6). The transformations act on the underlying storage structure of a representational medium.

The final step for the representation of narrative text in the interactive narrative space invokes the transformed narrative depiction to realise the narrative text within the interactive narrative space. The narrative depiction is invoked by calling the `Invoke` method implemented in the generated element class. The program code specified for the `Invoke` method issues the necessary commands to the Microsoft ® DirectX ® graphics rendering API, which renders the transformed graphical model of a cube to a graphical display. Figure 6.4 illustrates the graphical output produced when sample sentence, "The red cube is on the blue cube.", is processed by the CVTNG system prototype.

6.1.6 Adverbs

This section describes the actions of the CVTNG when the CVTNG system is applied to a version of the sample sentence, "The red cube is on the blue cube.", altered to include adverbs of amplitude (refer to sections 4.3.4 and 5.4.5). The sample sentence, "The red

cube is on the blue cube.", is changed to the sentence "The dark red cube is far above the light blue cube.". The altered sample sentence includes the adverbs "dark", "far", and "light".

Table 6.1 specifies the details of the models defined for the adverbs "dark", "light", and "far" in the "RedChannel", "BlueChannel", and "GreenChannel" contexts of interpretation and in the partial contexts, "SpatialStartSelfY" and "SpatialStartOtherY". The definition of the verb phrase "is above" in the contexts of interpretation "SpatialX" and "SpatialY", the subcontexts "SpatialStartX" and "SpatialStartY", and the partial contexts "SpatialStartSelfX", "SpatialStartOtherX", "SpatialStartReferenceX", "SpatialStartSelfY", "SpatialStartOtherY", and "SpatialStartReferenceY" partial contexts is provided in table 6.1. The additional word definitions were specified using the "Word Editor" CVTNG system interface component presented in section 5.4.1.

The verb phrase "is above" was substituted for the verb phrase "is on" for the example presented in this section because the phrase "far is on" does not make sense linguistically. The CVTNG system will still process the statement, however, since no language based restrictions are placed on the natural language sentences provided as input to the CVTNG system. The sentence, "The red cube is far on the blue cube", produces a graphical rendering of a red cube slightly above a blue cube when provided as input to the CVTNG system. Checks for linguistic correctness do not fall within the scope of this work and all natural language sentences are processed in the same way.

The parsing procedure produces the parse bin relationships illustrated in figure 6.5 when applied to the sentence, "The dark red cube is far above the light blue cube.". Figure 6.5 shows adverb parse bins containing the words "dark", "light", and "far". The adverb parse bins are associated parse bins that contain the adjectives "red" and "blue" and the verb phrase "is above". The evolving function models formed from the parse bin relationships illustrated in figure 6.5 are illustrated in figure 6.6. The relationships between the adverb parse bins and the parse bins containing the adjectives and verb phrase of the sample sentence are modelled as transformation functions (refer to sections 4.3.4 and 5.4.5). The adverbs "dark", "light", and "far" are defined as adverbs of amplitude (refer to sections 4.3.4 and 5.4.5). The transformation functions associated with the adverbs in the sample sentence are therefore applied to the evolving function

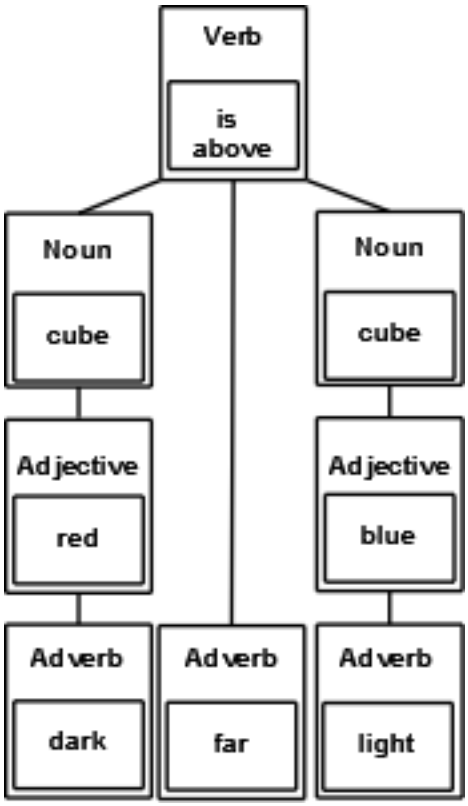

Figure 6.5: The parse bin relationships produced for the sample sentence, "The dark red cube is far above the light blue cube."

amplitude values specified in the same context, subcontext, or partial context that the adverbs are defined in.

The following evolving functions are produced by the CVTNG parsing procedure for the sample sentence, "The dark red cube is far above the light blue cube.":

$$\begin{aligned}
Red(Red) &= f_{one}(t)\,(\Psi_{half} \circ 0.7)\,\kappa \\
&= (1.0)(0.5)(0.7)(1.0) \\
&= 0.35 \\
Blue(Blue) &= f_{one}(t)\,(\Psi_{two} \circ 0.7)\,\kappa \\
&= (1.0)(2.0)(0.7)(1.0) \\
&= 1.4 \\
YPosition(Red) &= f_{one}(t)[\kappa_1 YPosition(Blue) \\
&+ (\Psi_{oneandhalf} \circ \kappa_2 ObjectHeight(Blue)) \\
&+ (\Psi_{oneandhalf} \circ \kappa_3 ObjectHeight(Red))] \\
&= (1.0)\,((1.0)(0.0) + (1.5)(1.0)(1.0) + (1.5)(1.0)(1.0)) \\
&= 3.0
\end{aligned} \tag{6.3}$$

The transformation functions Ψ_{half}, $\Psi_{oneandhalf}$, and Ψ_{two} are defined as in equation (4.20) and specified in table 6.1. The composition operation, $\circ$, was performed by calculating the values of the transformation functions Ψ_{half}, $\Psi_{oneandhalf}$, and Ψ_{two} with the respective amplitude values that the transformation functions apply to as input. All other symbols of the equations above are defined as for equations (6.1) and (6.2).

The measurement values assigned to the attribute variables "Red(Red)", "Blue(Blue)", and "YPosition(Red)" and the default measurement values substituted for the attribute variables "YPosition(Blue)", "ObjectHeight(Blue)", and "ObjectHeight(Red)" are determined by the recursive substitution procedure. The operation of the recursive substitution procedure is the same as that described for equation (6.2) with the exception that the default measurement values of the attribute variables "ObjectHeight(Blue)", and "ObjectHeight(Red)" are transformed by the transformation function, $\Psi_{oneandhalf}$. The default amplitude value specified for the evolving functions of the adjectives "red" and

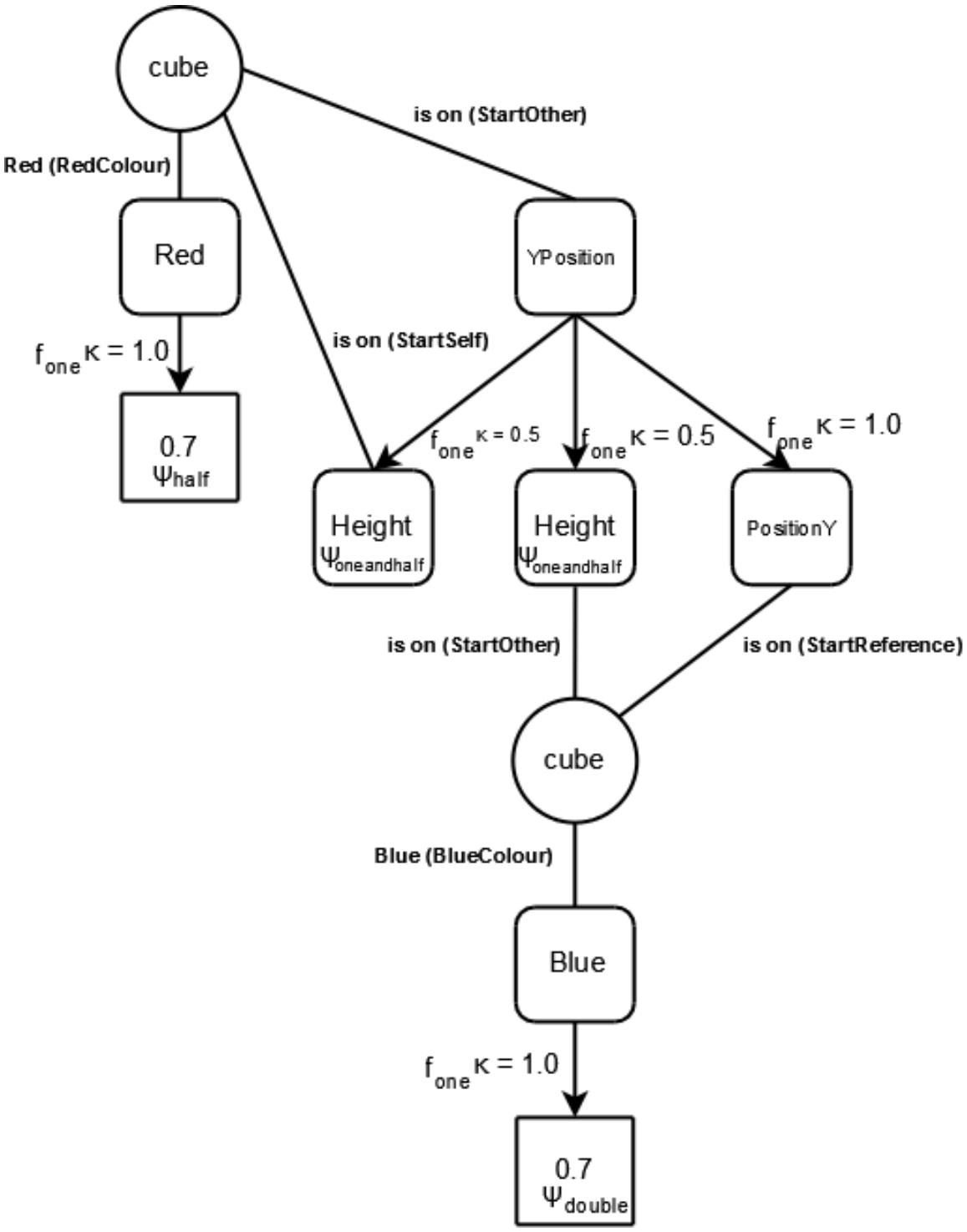

Figure 6.6: The attribute variable relationships produced by resolving the parse bins in figure 6.5

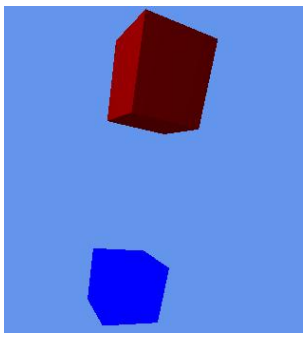

Figure 6.7: The visual output produced for the sample sentence, "The dark red cube is far above the light blue cube."

"blue" are transformed by the transformation functions, Ψ_{half} and Ψ_{two}, respectively before the measurement value from the associated evolving function equations are calculated as specified for equation (6.2). The maximum colour value within the Microsoft ® DirectX ® API is 1.0. A measurement value assignment of 1.4 to the blue colour component of a representational element has the same effect as an assignment of 1.0.

The adverbs introduced have no effect on the generated element class. The measurement values obtained from the evolving function models specified by the equations above are transferred to the representation system in the same way as the measurement values obtained in section 6.1.5. The graphical output that results when the CVTNG system processes the sample sentence "The dark red cube is far above the blue cube." is shown in figure 6.7.

6.1.7 Inconsistency handling

This section describes the operation of the CVTNG system when applied to the sample sentences, "The red cube is to the left of the blue cube. The red cube is to the right of the blue cube.". The discussion of the example serves to illustrate the handling of inconsistent natural language statements that describe static relationships between objects. The sentences are once again represented visually in an interactive narrative space. The inconsistency is introduced by the conflicting natural language statements regarding the position of the red cube. The definition of the verb phrases "is to the left of" and "is to the right of" in the contexts of interpretation "SpatialX" and "Spa-

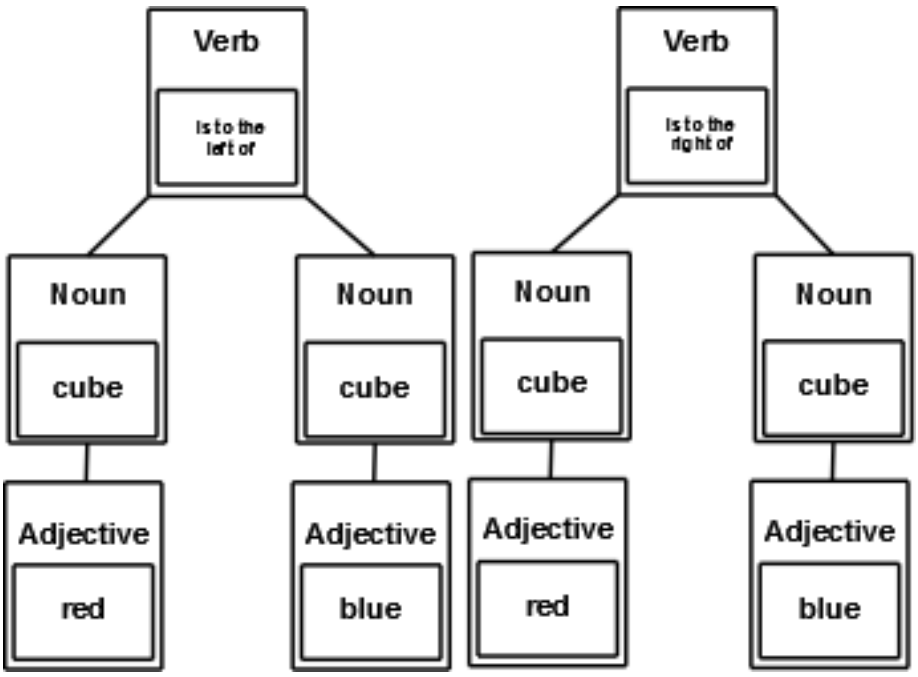

Figure 6.8: The parse bin relationships produced for the sample sentences, "The red cube is to the left of the blue cube. The red cube is to the right of the blue cube."

tialY", the subcontext "SpatialStartX" and "SpatialStartY", and the partial contexts "SpatialStartSelfX", "SpatialStartSelfY", "SpatialStartOtherX", "SpatialStartOtherY", "SpatialStartReferenceX", and "SpatialStartReferenceY" is stated in table 6.1.

The parsing algorithm (refer to algorithm 1) processes the sample sentences in sequence. The words in each sentence are also parsed in sequential order. The parsing algorithm produces the parse bin relationships illustrated in figure 6.8, which shows two clear subgraphs related to the two sentences of the example. Figure 6.9 illustrates the evolving function models formed from the parse bin relationships shown in figure 6.8 in terms of attribute variable relationships. Figure 6.9 clearly shows two separate measurement value assignments to the attribute variable, "XPosition(Red)".

If the crisp strategy for assigning measurement values to attribute variables is applied to the sample sentences, two evolving function equations are produced that assign measurement values to the attribute variable, "XPosition(Red)". The evolving function

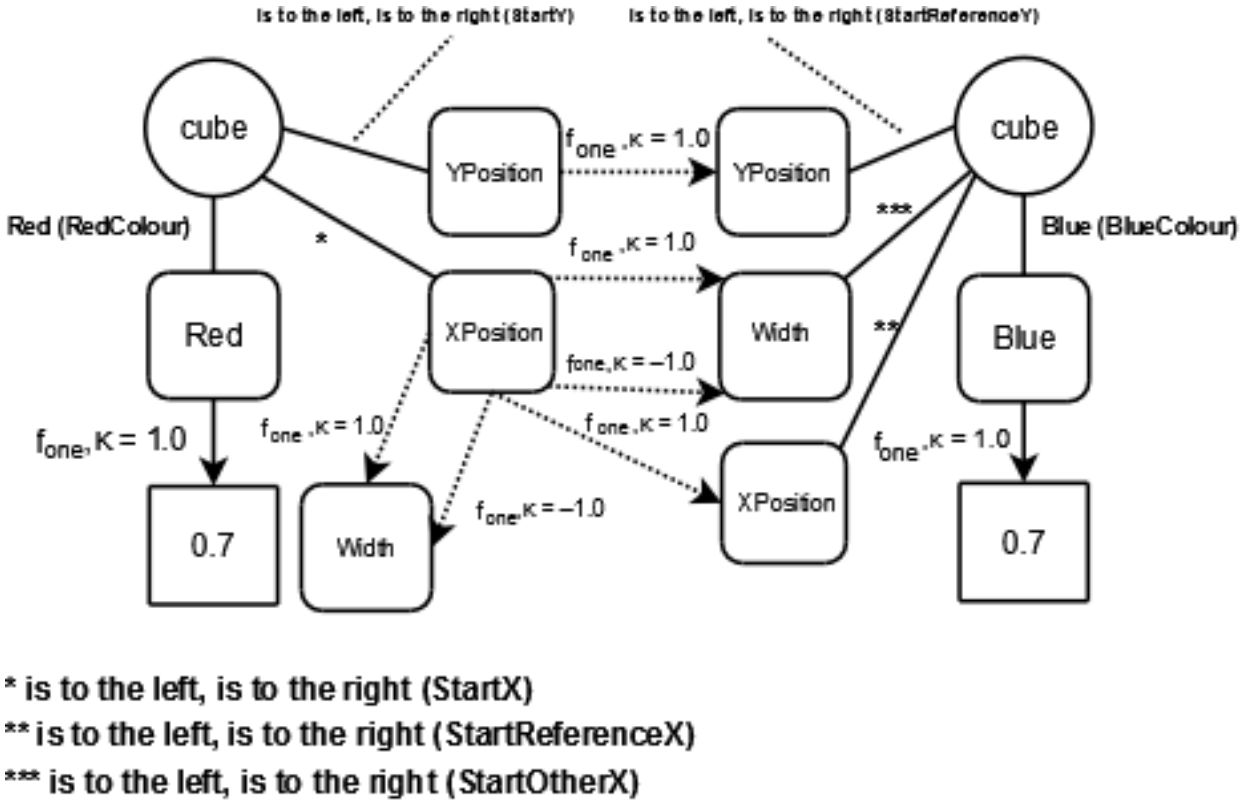

Figure 6.9: The attribute variable relationships produced by resolving the parse bins in figure 6.8

equations formed for the sample sentences are:

$$\begin{aligned}
Red(Red) &= f_{one}\kappa 0.7 \\
Blue(Blue) &= f_{one}\kappa 0.7 \\
XPosition(Red) &= f_{one}\kappa_1 XPosition(Blue) + f_{one}(\kappa_2 Width(Blue) + \kappa_3 Width(Red)) \\
XPosition(Red) &= f_{one}\kappa_1 XPosition(Blue) - f_{one}(\kappa_2 Width(Blue) + \kappa_3 Width(Red)) \\
YPosition(Red) &= f_{one}\kappa(YPosition(Blue)) \\
YPosition(Red) &= f_{one}\kappa(YPosition(Blue)) \\
. & \qquad (6.4)
\end{aligned}$$

The annotations (Blue) and (Red), the function, f_{one}, and the scaling values, $\kappa, \kappa_1, \kappa_2, \kappa_3$, are defined as they were for equations (6.1) and (6.2).

The recursive substitution algorithm produces the following set of measurement value assignments when executed for the evolving function equations above:

$$\begin{aligned}
Red(Red) &= (1.0)(1.0)(0.7) \\
&= 0.7 \\
Blue(Blue) &= (1.0)(1.0)(0.7) \\
&= 0.7 \\
XPosition(Red) &= (1.0)(1.0)(0.0) + (1.0)(1.0)(1.0) + (1.0)(1.0)(1.0) \\
&= 2.0 \\
XPosition(Red) &= (1.0)(1.0)(0.0) - (1.0)(1.0)(1.0) - (1.0)(1.0)(1.0) \\
&= -2.0 \\
YPosition(Red) &= (1.0)(1.0)(0.0) \\
&= 0.0
\end{aligned} \tag{6.5}$$

The measurement value assignments above show that two separate measurement values of 2.0 and -2.0 are assigned to the attribute variable, "XPosition(Red)". The CVTNG subsystem for assigning measurement values to attribute variables therefore determines that the fuzzy procedure for assigning measurement values to attribute variables is to be used for assigning an approximated measurement value to the attribute variable, "XPosition(Red)". The measurement value assignments of the other attribute variables are correctly determined and not considered further.

The chromosomes of algorithm 7 used in the fuzzy procedure for assigning measurement values to attribute variables are formed by adding a gene to the chromosomes for every variable not assigned an exact and a unique measurement value. The chromosomes of the population of algorithm 7, as applied to the single unknown attribute variable, "XPosition(Red)", have a single gene that corresponds to the attribute variable, "XPosition(Red)".

A set of fuzzy constraints is formed for a subset of the evaluated evolving function equations (EEFEs) (refer to equation (6.5)) which involve attribute variables that are

not assigned measurement values. The fuzzy constraints formed are:

$$\begin{aligned} C_1 : XPosition(Red) &= 2.0 \\ C_2 : XPosition(Red) &= -2.0 \end{aligned} \tag{6.6}$$

$C_i, i = (1, 2)$ are fuzzy constraints formed for the EEFEs related to the attribute variable "XPosition(Red). The membership degrees of the C_i, $\mu_{C_i}(\overline{x})$, are calculated by substituting the right-hand side of the C_i (in this case 2.0 and -2.0) as the constant value, K, and the gene values of a chromosome as the value tuple, $\overline{x}$, into equation (4.35). The scaling factors, d_1 and d_2, were calculated by equation (4.42) as 4.0. The domain of attribute variable "XPosition(Red)" was determined as specified in section 4.4.3.5 as $[-2.0, 2.0]$.

Algorithm 7 was executed for the fuzzy constraints, C_1 and C_2, with a population of constructed chromosomes that contain a single gene, g_1, related to the attribute variable, "XPosition(Red)". A population size of 16 was determined experimentally to provide optimal results averaged over 30 iterations of algorithm 7 for population sizes ranging from 2 to 20. The crossover probability of 1.0 and the mutation probability of 0.1 specified in section 5.6.3.4 were retained. The value, $\boldsymbol{\alpha}$, used by the BLX-$\boldsymbol{\alpha}$ crossover operator was experimentally determined to provide optimal solutions at $\boldsymbol{\alpha} = 0.7$. The value of $\boldsymbol{\alpha}$ was experimentally determined over 30 repetitions of algorithm 7 for $\boldsymbol{\alpha}$ values between 0.0 and 1.0 in increments of 0.1. The optimal population size and blending factor were determined in tandem and the experiments were repeated 30 times for every combination of blending factor and population size. Algorithm 7 was set to terminate after 500 generations ($p_{gen} = 500$) to allow comparison between different runs of the algorithm and also for the experimental determination of the other parameters of algorithm 7.

Figure 6.10 illustrates the accuracy of algorithm 7 as measured in terms of the fitness of the fittest individual in the population. A fitness value of 0.5 is the maximum fitness attainable in the case of an inconsistent system such as the one presented in this example.

The fittest chromosome in the initial population had a fairly high fitness value and the fittest chromosome over the subsequent generations rapidly reached a near-optimal solution. The averaged fitness of the entire chromosome population was not initialised

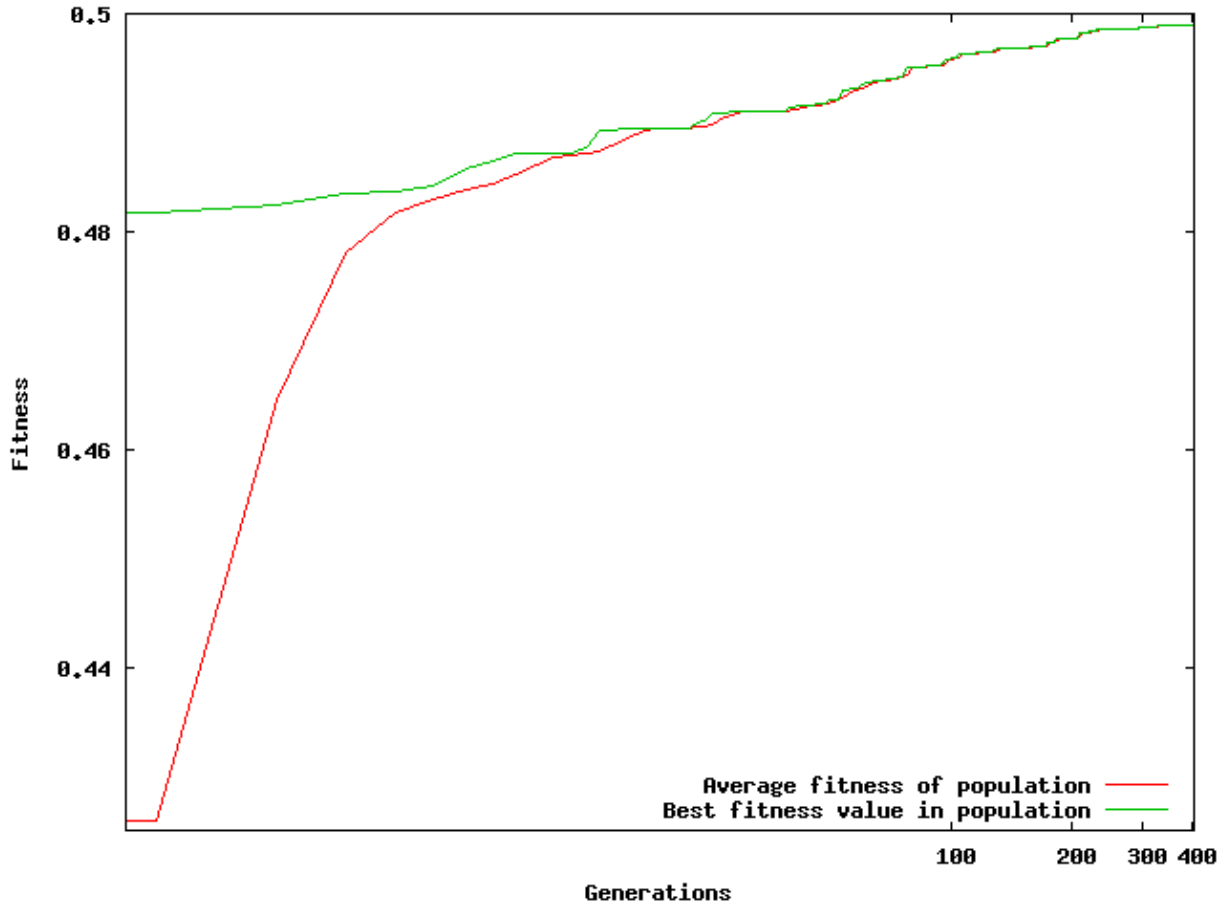

Figure 6.10: The solution accuracy of the genetic algorithm for fuzzy constraint satisfaction when applied to the sample sentences, "The red cube is to the left of the blue cube. The red cube is to the right of the blue cube."

as high as the fittest chromosome which shows a fair exploration of the search space. The averaged and best fitness values quickly converge, however, which shows that the population very quickly becomes homogeneous.

The visual output for a near-optimal approximate measurement value assignment to the attribute variable "XPosition(Red)" is displayed in figure 6.11. The approximate measurement value was determined by algorithm 7 when applied to the sample sentences, "The red cube is to the left of the blue cube. The red cube is to the right of the blue cube.".

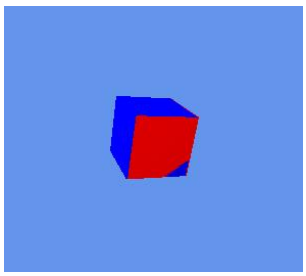

Figure 6.11: The visual output produced for the sample sentences, "The red cube is to the left of the blue cube. The red cube is to the right of the blue cube."

6.2 Periodic example

This section applies the CVTNG system to a series of natural language sentences that describe the actions of objects which repeat over time. The repetitive actions described by adjectives and verbs are modelled as centre computational verbs (refer to definition 3.19). The examples serve to illustrate the CVTNG system's handling of evolving function models parameterised by time.

Section 6.2.1 states the contexts of interpretation, subcontexts, and partial contexts that group and subdivide the evolving function models defined for the sample sentences of this section. The evolving function models of the adjectives and verbs of the sample sentences are stated.

The actions of the CVTNG system on sentences which describe actions that repeat over a time interval are described in section 6.2.2. The actions of the parsing subsystem, the subsystem for assigning measurement values to attribute variables, and the interaction between the CVTNG system and the representation system are described.

The effect of adverbs of amplitude and adverbs of period on natural language sentences that describe actions that repeat over a period of time is described in section 6.2.3. The modified evolving function models are presented and the actions of the parsing, measurement value assignment, and representations systems are described.

The effect of inconsistencies in natural language statements that describe periodic actions over time are examined in section 6.2.4. The evolving function models of the sample sentences that contain inconsistencies are presented. Finally, the actions of the CVTNG subsystem are described for the examples that contain inconsistencies.

6.2.1 Modeling

Here the sample sentence "The green rotating cube orbits the yellow cube." is examined. The sentence ascribes the periodic adjective, "rotating", to an instance of the noun "cube" and describes a periodic action by means of the word "orbit". The example enables us to examine the actions of the CVTNG system on adjectives and verbs of a periodical nature. The adjectives and verbs are applied to the noun "cube" represented as a 3D model to convey the actions and properties of the adjectives and verbs within the interactive narrative space.

Table 6.2 specifies the definition details of the implementation of the adjectives "yellow" and "green" in the contexts of interpretation "RedChannel" and "GreenChannel". The definition details of the adjective "rotating" in the context of interpretation "SpatialX" are specified as are those of the verb "orbit" for the "SpatialX" and "SpatialZ" contexts of interpretation. The column definitions of table 6.2 are the same as those for table 6.1. The column "Time Interval", if applicable, of table 6.2 lists the time interval of the evolving function model for the word listed in the "Word" column as modelled in the context of interpretation listed in the "Context" column. The applicable context of interpretation and natural language word are listed in the same table row as the time interval specified.

The annotation "*" specified for the "Time interval" column of table 6.2 indicates a time interval of an evolving function model defined to repeat over the specified time interval and is specified by the `Infinite` attribute in the CVTNG system (refer to section 5.4.4).

Figure 6.12 illustrates the new contexts of interpretation, subcontexts, and partial contexts introduced for the examples of this section. The attribute variables specified for the newly introduced contexts of interpretation, subcontexts, and partial contexts are also shown. The attribute variables are either assigned measurement values by evolving functions in the related contexts of interpretation or are used as attribute variables in related subcontexts and partial contexts.

The "SpatialX" and "SpatialZ" contexts of interpretation are related to the "SpatialStartX", "SpatialStartZ", "SpatialChangeX", and "SpatialChangeZ" subcontexts that serve to define the terms of the evolving functions related to the verb "orbits". The

Word	Context	Attribute variable	BE-verb	Scaling factor κ	Generic function f	Amplitude value α	Transformation function $\Psi(x)$	Time interval
be(circle)	SpatialX			$\kappa = 1.0$	$f_{XDiv2Pi}(t) = \frac{t}{2\pi}$			$0 < t < 2\pi*$
	SpatialZ			$\kappa = 1.0$	$f_{XDiv2Pi}(t) = \frac{t}{2\pi}$			$0 < t < 2\pi*$
rotating	SpatialX	YRotation	be(circle)			2π		
green	GreenChannel	Green	be			0.7		
yellow	RedChannel	Red	be			0.7		
	GreenChannel	Green	be			0.7		
rotates	SpatialX	YRotation		$\kappa = 1.0$	$f_{XDiv2Pi}(t) = \frac{t}{2\pi}$	2π		$0 < t < 2\pi*$
orbits	SpatialX	XPosition						
	SpatialStartX				$f_{one}(t) = 1$			
	SSReferenceX			$\kappa_1 = 1.0$		XPosition		
	SSOtherX			$\kappa_2 = -1.0$		Width		
	SSSelfX			$\kappa_3 = -1.0$		Width**		
	SpatialChangeX				$f_{halfcoshalf}(t) = \frac{-cos(t)}{2} + \frac{1}{2}$			
	SCOtherX			$\kappa_4 = 2.0$		Width		
	SCSelfX			$\kappa_5 = 2.0$		Width**		
	SpatialZ	ZPosition						
	SpatialStartZ				$f_{one}(t) = 1$			
	SSReferenceZ			$\kappa_1 = 1.0$		ZPosition		
	SSOtherZ			$\kappa_2 = -1.0$		Depth		
	SSSelfZ			$\kappa_3 = -1.0$		Depth**		
	SpatialZ	ZPosition						
	SpatialChangeZ				$f_{halfsinhalf}(t) = \frac{\sin(t)}{2} + \frac{1}{2}$			
	SCOtherZ			$\kappa_4 = 2.0$		Depth		
	SCSelfZ			$\kappa_5 = 2.0$		Depth**		
clockwise	SpatialX						$\Psi_{minone}(x) = -1.0x$	
	SpatialZ						$\Psi_{minone}(x) = -1.0x$	
anticlockwise	SpatialX						$\Psi_{one}(x) = 1.0x$	
	SpatialZ						$\Psi_{one}(x) = 1.0x$	
slowly	SpatialX						$\Psi_{half}(x) = 0.5x$	
	SpatialZ						$\Psi_{half}(x) = 0.5x$	
quickly	SpatialX						$\Psi_{two}(x) = 2.0x$	
	SpatialZ						$\Psi_{two}(x) = 2.0x$	

Table 6.2: Periodic word definitions

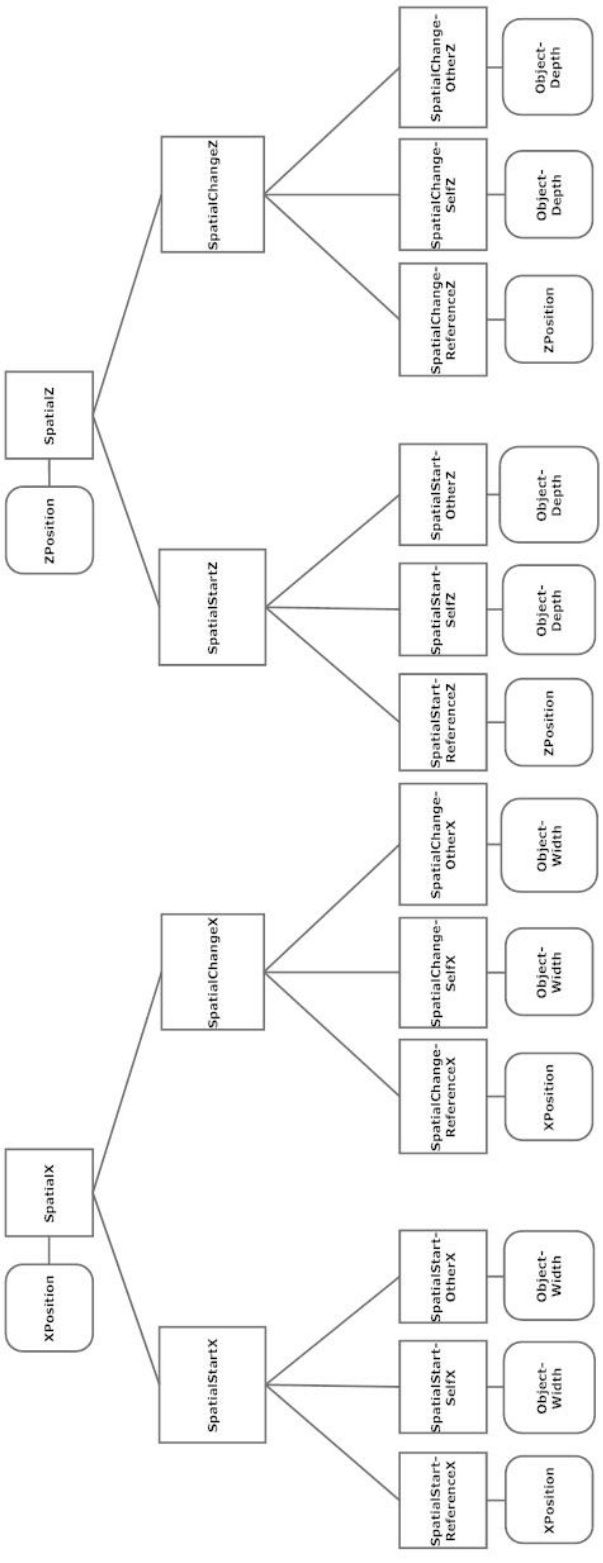

Figure 6.12: Additional contexts of interpretation, subcontexts, and partial contexts introduced for the computational verb models of the sample sentence, "The green rotating cube orbits the yellow cube."

subcontexts “SpatialChangeX” and “SpatialChangeZ” serve to model the terms of the evolving function equations that in turn model the change in the position of an object along the x-axis and the z-axis respectively.

The partial contexts “SpatialStartSelfX”, “SpatialStartOtherX”, and “SpatialStartReferenceX” are related to the subcontext “SpatialStartX” and serve to model the partial amplitude values related to the width of a subject noun, the width of an object noun, and the position of an object noun respectively. The partial contexts “SpatialStartSelfZ”, “SpatialStartOtherZ”, and “SpatialStartReferenceZ” are related to the subcontext “SpatialStartZ” and serve to model the partial amplitude values related to the depth of a subject noun, the depth of an object noun, and the position of an object noun respectively. The partial contexts “SpatialChangeSelfZ”, “SpatialChangeOtherZ”, “SpatialChangeReferenceZ”, “SpatialChangeSelfX”, “SpatialChangeOtherX”, and “SpatialChangeReferenceX” are similar to the partial contexts related to the “SpatialStartX” subcontexts, but serve to model partial amplitude values related to the displacement of an object along the z-axis in a 3D space.

The contexts of interpretation, subcontexts, and partial contexts stated above were implemented in the “Context Editor” interface component of the CVTNG system (refer to section 5.3). The words of the sample sentence specified in table 6.2 were implemented within the stated contexts of interpretation, subcontexts, and partial contexts in the “Word Editor” interface component of the CVTNG system. The noun “cube” was implemented in the newly introduced contexts of interpretation, subcontexts, and partial contexts illustrated in figure 6.12. The element class associated with the noun “cube” was regenerated to accommodate the additional properties (refer to section 5.3.2) specified in the contexts of interpretation, subcontexts, and partial contexts illustrated in figure 6.12.

A `YRotation` property which corresponds to a rotation around the y-axis (horizontally) of the coordinate system within the Microsoft® DirectX® graphics rendering API was implemented in the “SpatialX” context of interpretation and the “SpatialStartX” subcontext. A corresponding rotation transformation was implemented within the transformation library (refer to section 5.3.4). The rotation transformation is defined as a matrix multiplied with the vectors that contain the vertices of a 3D graphical model.

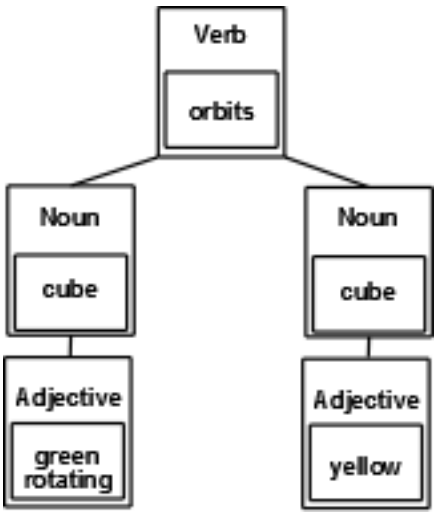

Figure 6.13: The parse bin relationships produced for the sample sentence, "The green rotating cube orbits the yellow cube."

The transformed vertices correspond to a transformed version of the graphical model rotated around the y-axis by a specified degree equal to the measurement value of the associated attribute variable. The associated attribute variable "YRotation" is specified within the `YRotation` property.

6.2.2 Parsing, resolution, and representation

The actions of the CVTNG parsing subsystem applied to the sample sentence "The green rotating cube orbits the yellow cube." are similar to the sequence of actions described in section 6.1.3. The parse bin relationships illustrated in figure 6.13 are produced. The attribute variable relationships of the evolving function models formed from the parse bin relationships illustrated in figure 6.13 are shown in figure 6.14.

The difference between the evolving function models of section 6.1 and those defined for "The green rotating cube orbits the yellow cube." are that the evolving function models of the latter evaluate to different values over time. The evolving function equations produced for the sample sentence, "The green rotating cube orbits the yellow cube.",

are:

$$\begin{aligned}
Green(Green) &= f_{one}(t)\kappa_1(0.7) \\
Red(Yellow) &= f_{one}(t)\kappa_1(0.7) \\
Green(Yellow) &= f_{one}(t)\kappa_1(0.7) \\
YRotation(Green) &= f_{Linear}(t)\kappa_1 2\pi \\
XPosition(Green) &= f_{one}(t)\kappa_1 Width(Green) + f_{one}(t)\kappa_2 Width(Yellow) \\
&+ f_{MinusHalfCosPlusHalf}(t)\kappa_3 Width(Green) \\
&+ f_{MinusHalfCosPlusHalf}(t)\kappa_4 Width(Yellow) \\
&+ f_{one}(t)\kappa_5 YPosition(Yellow) \\
ZPosition(Green) &= f_{one}(t)\kappa_1 Depth(Green) + f_{one}(t)\kappa_2 Depth(Yellow) \\
&+ f_{HalfSinPlusHalf}(t)\kappa_3 Depth(Green) \\
&+ f_{HalfSinPlusHalf}(t)\kappa_4 Depth(Yellow) \\
&+ f_{one}(t)\kappa_5 ZPosition(Yellow)
\end{aligned} \quad . \tag{6.7}$$

The annotation, (Green), refers to the instance of the noun "cube" described by the adjective "green". The annotation, (Yellow), refers to the instance of the noun "cube" described by the adjective "yellow". $f_{one}(t)$, f_{Linear}, $f_{MinusHalfCosPlusHalf}$, and $f_{HalfSinPlusHalf}$ are generic functions, as defined for equation (4.3), and their are specified in table 6.2. The $\kappa_i \in \mathbb{R}$ are scalars as defined for equation (4.3) and the values of the scalars are specified in table 6.2. t designates the current value of a timer in seconds. $t = 0$ corresponds to the first iteration of the procedures for assigning measurement values to attribute variables.

The evolving function equations (refer to equation (6.7)) for the sample sentence, "The green rotating cube orbits the yellow cube.", in contrast to the evolving function equations of the sample sentence, "The red cube is on the blue cube.", produce different measurement values over time. The sample sentence, "The red cube is on the blue cube.", was modelled in terms of static computational verbs (refer to definition 3.15) while the sample sentence of this section, "The green rotating cube orbits the yellow cube.", was modelled in terms of centre computational verbs (refer to definition 3.19).

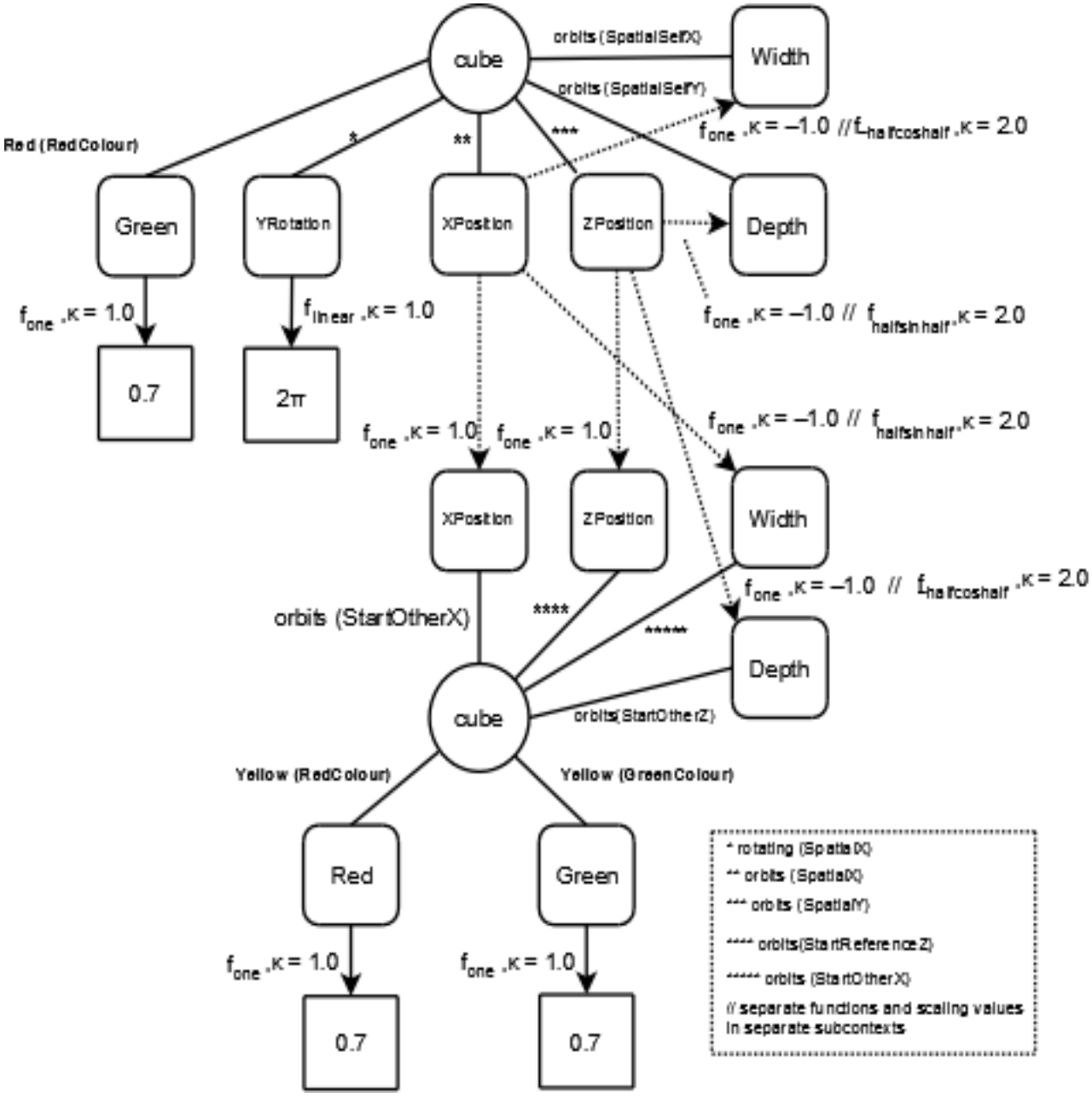

Figure 6.14: The attribute variable relationships produced by resolving the parse bins in figure 6.13

The generic functions $f_{HalfSinPlusHalf}$, $f_{MinusHalfCosPlusHalf}$, and f_{Linear} evaluate to different values for different time values, however, and the measurement values obtained by the crisp procedure for assigning measurement values to attribute variables (refer to section 4.4.2) therefore produces different measurement values over a period of time. The measurement values are assigned to the corresponding attribute variables. The equations stated in equation (6.7) are accordingly left unevaluated within this discussion, because the equations may evaluate to different measurement values for different values of the time parameter, t.

The CVTNG subsystem for assigning measurement values to attribute variables executes the recursive substitution algorithm (refer to section 5.6.3.1) for a specific time value and obtains the measurement values for the attribute variables "Green(Green)", "Red(Yellow)", "Green(Yellow)", "YRotation(Green)"," XPosition(Green)", and "ZPosition(Green)".

The measurement values for a specific instance of the noun "cube" (Green or Yellow) are passed to the element class generated for the noun "cube". The transformations applied to the vertices of the 3D model of a cube are instanced as parameterised by the measurement values obtained. The transformations are instanced by the obtained measurement values when the appropriate accessor methods of the element class generated (refer to section 5.4.6) are called.

The instanced transformations are applied to the vertices of the 3D model of the cube when the `Transformation` method of the generated element class is invoked. The transformed vertices of the cube are rendered by calls to the Microsoft® DirectX® API as implemented in the `Invoke` method of the element class generated. The transformed image of the cube represents the natural language noun "cube" within the interactive narrative space as described by the natural language adjectives and verbs of the sample sentence, "The green rotating cube orbits the yellow cube.", at a specific point in time. The steps above are performed for both instances of the noun "cube" and therefore two separate cubes are rendered to the graphical display.

The measurement values obtained from the evolving function equations (refer to equation (6.7)) vary over time and so the transformations applied to the vertices of the cube models will also vary over time. The sequence of images rendered for the varying

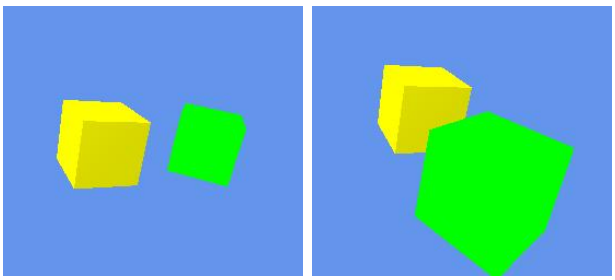

Figure 6.15: The visual output produced for the sample sentence, "The green rotating cube orbits the yellow cube.", for time values, $t = \frac{\pi}{2}, t = \pi$

transformations on the graphical models of the cubes serves to represent the natural language adjectives and verbs of the periodic sample sentence, "The green rotating cube orbits the yellow cube.", over a time period. Figure 6.15 depicts the visual representation of the sentence "The green rotating cube orbits the yellow cube." for time values of $t = \pi/2$ and $t = \pi$ respectively. The images form part of an animation which shows that the green cube rotates around its own y-axis while it simultaneously orbits around the yellow cube.

6.2.3 Adverbs

This section examines the actions of the CVTNG system when applied to the sample sentence, "The green quickly rotating cube slowly orbits the yellow cube.". The sample sentence extends the sample sentence, "The green rotating cube orbits the yellow cube.", with the addition of the natural language adverbs "quickly" and "slowly" that are implemented as adverbs of period within the CVTNG system (refer to sections 4.3.4 and 5.4.5).

Table 6.2 contains the implementation details of the adverbs "quickly" and "slowly" within the "SpatialX" and "SpatialZ" contexts of interpretation. The transformation functions Ψ_{half} and Ψ_{two} used within the evolving function models of the adverbs "quickly" and "slowly" were implemented as specifications of the `Function` interface (refer to section 5.4.7). The specifications of the `Function` interface that correspond to the transformation functions Ψ_{half} and Ψ_{two} were implemented as part of the function library described in section 5.4.7.

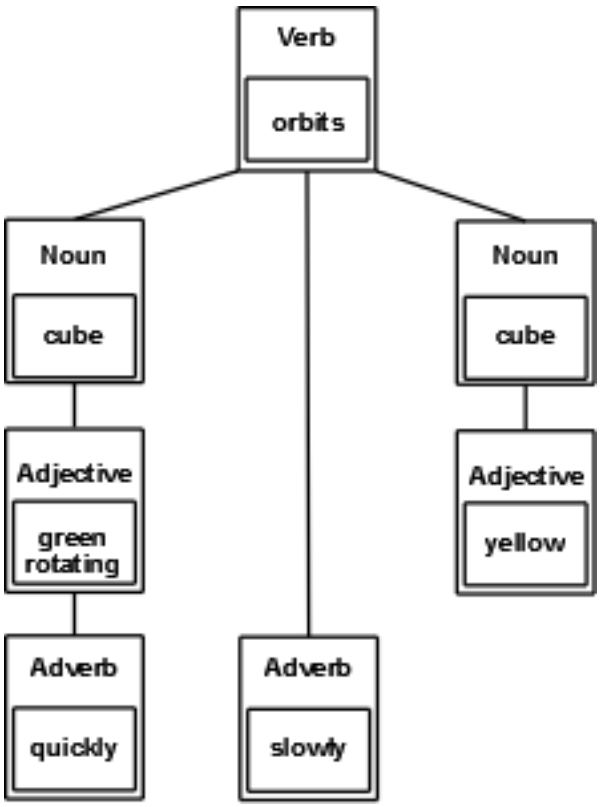

Figure 6.16: The parse bin relationships produced for the sample sentence, "The green quickly rotating cube slowly orbits the yellow cube."

The actions of the CVTNG parsing subsystem on the sample sentence, "The green quickly rotating cube slowly orbits the yellow cube.", are similar to those described in section 6.1.3. The parse bin relationships produced for the sample sentence, "The green quickly rotating cube slowly orbits the yellow cube.", are shown in figure 6.16. The attribute variable relationships of the evolving function models formed from the parse bin relationships shown in figure 6.16 are similar to those shown in figure 6.14 and are therefore not shown. Figure 6.16 shows that the adverbs "quickly" and "slowly" are associated with the adjective "rotating" and the verb "orbits" respectively. The transformation functions, Ψ_{two} and Ψ_{half}, are applied to the time value, t, passed as parameter to the generic functions, f_i, specified for the evolving function models of the adjective "quickly" and the verb "orbits" (refer to section 4.3.4 and equation (4.21)). The adverbs "quickly" and "slowly" were not defined in the contexts of interpretation

"RedChannel" and "GreenChannel" and therefore the transformation functions, Ψ_{two} and Ψ_{half}, are not applied to the generic functions specified in the evolving function models of the adjectives "green" and "yellow".

The CVTNG subsystem for assigning measurement values to attribute variables forms a series of evolving function equations that correspond to the parse bin relationships illustrated in figure 6.16. The evolving function equations are formed from the evolving function models specified for the natural language words of the sample sentence, "The green quickly rotating cube slowly orbits the yellow cube.". The evolving function models defined are specified in table 6.2 for the applicable contexts of interpretation, subcontexts, and partial contexts.

The evolving function equations formed for the parse bin relationships from the defined evolving function models are:

$$\begin{aligned}
Green(Green) &= f_{one}(t)\kappa_1(0.7) \\
Red(Yellow) &= f_{one}(t)\kappa_1(0.7) \\
Green(Yellow) &= f_{one}(t)\kappa_1(0.7) \\
YRotation(Green) &= f_{Linear}(\Psi_{Double}(t) \circ t)\kappa_1 2\pi \\
XPosition(Green) &= f_{one}(t)\kappa_1 Width(Green) \\
&+ f_{one}(t)\kappa_2 Width(Yellow) \\
&+ f_{MinusHalfCosPlusHalf}(\Psi_{Half}(t) \circ t)\kappa_3 Width(Green) \\
&+ f_{MinusHalfCosPlusHalf}(\Psi_{Half}(t) \circ t)\kappa_4 Width(Yellow) \\
&+ f_{one}(t)\kappa_5 YPosition(Yellow) \\
YPosition(Green) &= f_{one}(t)\kappa_1 Depth(Green) \\
&+ f_{one}(t)\kappa_2 Depth(Yellow) \\
&+ f_{HalfSinPlusHalf}(\Psi_{Half}(t) \circ t)\kappa_3 Depth(Green) \\
&+ f_{HalfSinPlusHalf}(\Psi_{Half}(t) \circ t)\kappa_4 Depth(Yellow) \\
&+ f_{one}(t)\kappa_5 ZPosition(Yellow)
\end{aligned} \tag{6.8}$$

The annotations (Green) and (Yellow), the generic functions f_{one}, f_{Linear}, $f_{MinusHalfCosPlusHalf}$, and $f_{HalfSinPlusHalf}$, and the scaling factors, κ_i, are defined as for

equation (6.7). The transformation functions, Ψ_{Half} and Ψ_{Double}, are applied to the period of the generic functions, f_{one}, f_{Linear}, $f_{MinusHalfCosPlusHalf}$, and $f_{HalfSinPlusHalf}$. The transformation functions are applied to the generic functions, f_i, defined in the same context of interpretation or subcontext as the transformation functions. The transformations functions are applied to the generic functions, f_i, as specified by equation (4.21) for adverbs of period. The details of the transformation functions, Ψ_{Half} and Ψ_{Double}, are stated in table 6.2.

The evolving function equations for the sample sentence, "The green quickly rotating cube slowly orbits the yellow cube.", are similar to those of the sample sentence, "The green rotating cube orbits the yellow cube.". The difference between the two sets of evolving function equations is that the evolving function equations of the former sample sentence are transformed by adverbs of period. The time parameters of the generic functions, f_i, specified in the evolving function equations that assign measurement values to the attribute variables "YRotation(Green)", "XPosition(Green)", and "YPosition(Green)" are transformed by the transformation functions, Ψ_{Half} and Ψ_{Double}. The transformations are applied by means of the composition operation, $\circ$, performed by evaluating the transformation functions, Ψ_{Half} and Ψ_{Double}, for the time value, t, provided as input to obtain a transformed time value.

The transformed time values are used to calculate the values of the generic functions, f_{one}, f_{Linear}, $f_{LinearDown}$, $f_{MinusHalfCosPlusHalf}$, and $f_{HalfSinPlusHalf}$. The calculated values of the generic functions are used when the measurement values of the evolving function equations that correspond to the sample sentence, "The green quickly rotating cube slowly orbits the yellow cube.", are determined by the recursive substitution algorithm (refer to section 5.6.3.1). The remainder of the linear procedure for assigning measurement values to attribute variables occurs as for the sample sentence, "The green rotating cube orbits the yellow cube.". The measurement values determined by the linear procedure for assigning measurement values to attribute variables are passed to a representation system by means of a generated element class. The measurement values are transferred as described for the sample sentence in section 6.2.

Figure 6.17 shows the visual representation of the sentence, "The green quickly rotating cube slowly orbits the yellow cube.", for time values of $t = \pi/2$ and $t = \pi$. The

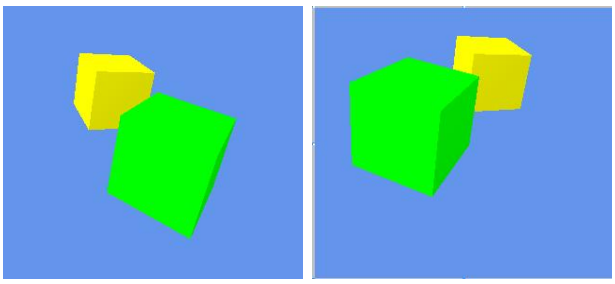

Figure 6.17: The visual output produced for the sample sentence, "The green quickly rotating cube slowly orbits the yellow cube.", for time values $t = \frac{\pi}{2}, t = \pi$

images shown in figure 6.17 can be compared with those shown in figure 6.15 to observe the effect of the adverbs of time in this sample sentence.

6.2.4 Inconsistency handling

This section describes the actions of the CVTNG system when applied to natural language sentences which describe actions that repeat over a period of time where the periodic actions described are in conflict with one another. The sample sentences, "The red cube slowly rotates clockwise. The red cube quickly rotates anticlockwise.", are examined. Table 6.2 states the evolving function model details of the adverbs "clockwise" and "anticlockwise" in the "SpatialX" context of interpretation. The transformation function, $\Psi_{MinusOne}$, was implemented as a specification of the `Function` interface in the function library (refer to section 5.4.7).

The actions of the CVTNG parsing system are similar to those described in section 6.1.3 and the parse bin relationships illustrated in figure 6.18 are produced. The attribute variable relationships of the evolving function equations that correspond to the sample sentences, "The red cube slowly rotates clockwise. The red cube quickly rotates anticlockwise.", are illustrated in figure 6.19. The attribute variable relationships illustrated in figure 6.19 are produced from the evolving function models specified in table 6.2 and the parse bin relationships shown in figure 6.18.

Figure 6.18 shows that the adverbs "clockwise" and "anticlockwise" are related to two instances of the verb "rotate". The instances of the noun "cube" in the two sample sentences cannot be distinguished from each other by their associated adjectives (refer

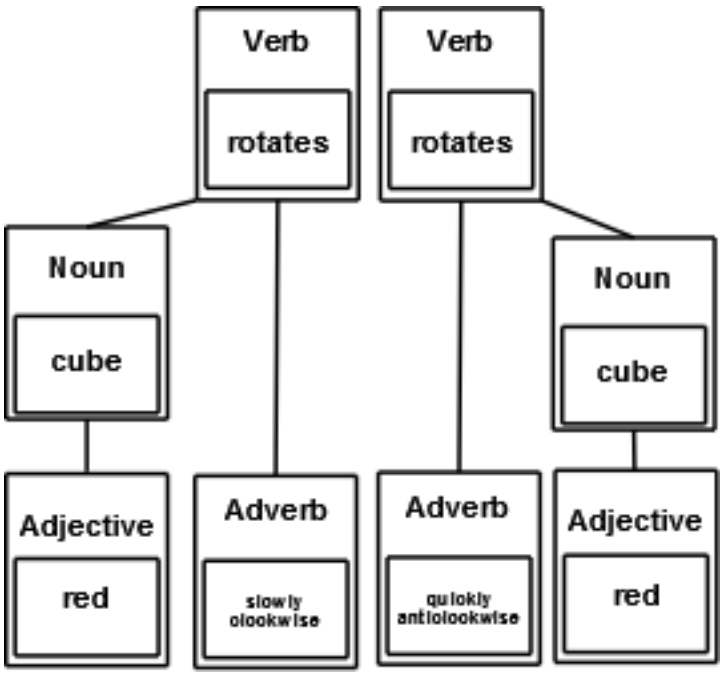

Figure 6.18: The parse bin relationships produced for the sample sentences, "The red cube slowly rotates clockwise. The red cube quickly rotates anticlockwise."

to section 5.4.3). The attribute variable relationships illustrated in figure 6.19 therefore clearly show two separate measurement value assignments to the attribute variable, "YRotation(Red)", by the evolving functions that correspond to the two instances of the verb, "rotate". The evolving functions that correspond to instances of the verb "rotate" are transformed by the transformation functions associated with the adverbs, "clockwise" and "anticlockwise", respectively.

The evolving function equations produced by the parsing subsystem of the CVTNG system, when applied to the sample sentences, "The red cube slowly rotates clockwise. The red cube quickly rotates anticlockwise.", are:

$$\begin{aligned} Red(Red) &= 0.7 \\ YRotation(Red) &= f_{Linear}(\Psi_{half} \circ t)\kappa_1(\Psi_{minusone} \circ 2\pi) \\ YRotation(Red) &= f_{Linear}(\Psi_{Double} \circ t)\kappa_1(\Psi_{one} \circ 2\pi) \\ . \end{aligned} \tag{6.9}$$

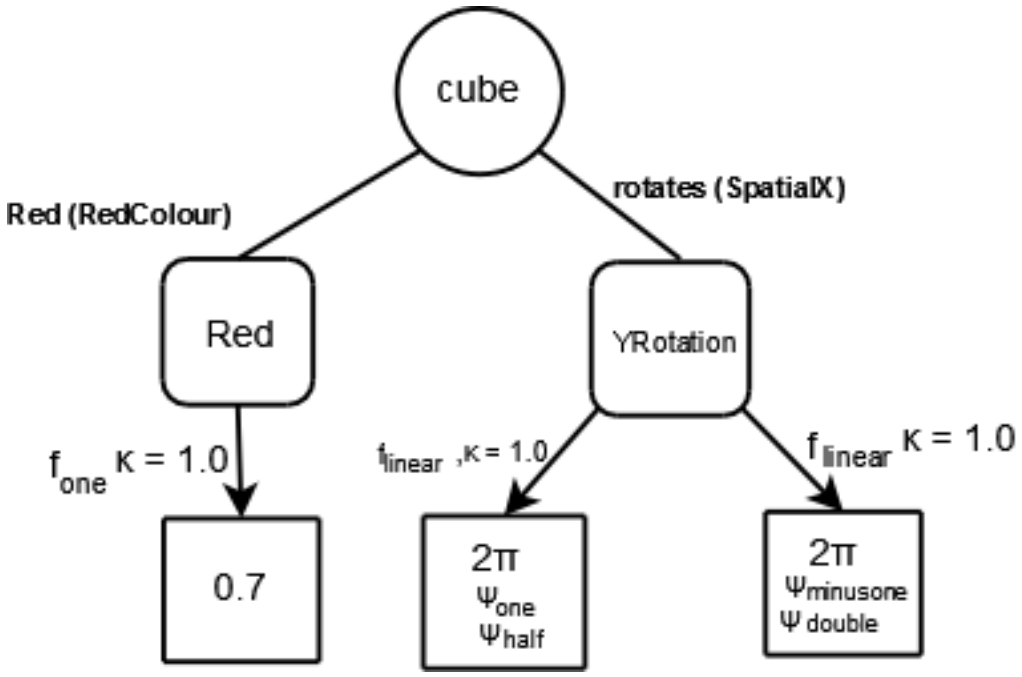

Figure 6.19: The attribute variable relationships produced by resolving the parse bins of 6.18

The annotation (Red) is as defined for equation (6.1). The generic function, f_{Linear}, the transformation functions Ψ_{Half} and Ψ_{Double}, the scaling values, κ_i, and the time parameter, t, are defined as for equation (6.8).

The evolving function equations (refer to equation (6.9)) clearly show two conflicting measurement value assignments to the attribute variable "YRotation(Red)". The linear procedure for assigning measurement values to attribute variables is therefore unable to assign a single exact measurement value to the attribute variable, "YRotation(Red)". The two evolving function equations that contain the attribute variable, "YRotation(Red)", are therefore formulated as fuzzy constraints in the fuzzy procedure for assigning approximate measurement values to attribute variables.

The actions of the fuzzy procedure for assigning approximate measurement values to attribute variables are similar to those presented in section 6.1.7. The chromosomes in the population of algorithm 7 have a single gene that corresponds to the attribute variable, "YRotation(Red)". The constraints of the fuzzy constraint satisfaction problem (FCSP) resolved by algorithm 7 correspond to the evolving function equations that

assign measurement values to the attribute variable, "YRotation(Red)" (refer to equation (6.9)). A population size of 6 was experimentally determined to provide optimal fitness values averaged over 30 iterations of algorithm 7. A blending factor of $\boldsymbol{\alpha} = 0.4$ was experimentally determined to provide optimal solution values averaged over 30 iterations of algorithm 7.

The optimal blending factor and population size were determined in tandem for different permutations of population size and blending factor. The population sizes ranged between 2 and 20 in increments of 1 and the blending factor ranged between 0.0 and 1.0 in increments of 0.1. All fitness values were averaged over 30 iterations of algorithm 7 for a specific population size and blending factor combination. The mutation and crossover probabilities were fixed at 0.1 and 1.0 respectively, as was the case for the sample sentences, "The red cube is to the left of the blue cube. The red cube is to the right of the blue cube.", discussed in section 6.1.7.

Figure 6.20 shows the average and best averaged fitness values within the chromosome population of algorithm 7 over 500 generations. The averaged fitness values are calculated over 30 repetitions of algorithm 7 for the parameter values specified above. The experiment was repeated for time values of $t = \pi/2$ and $t = \pi$. The average and best fitness values within the chromosome population initialise to fairly high fitness values and quickly converge towards a near-optimal fitness value of 0.5 for both time values. The fitness value of 0.5 corresponds to a near-optimal measurement value assignment for the attribute variable "YRotation(Red)" due to the conflicting fuzzy constraints. The conflicting fuzzy constraints correspond to the conflicting measurement value assignments from the evolving function models of the sample sentences, "The red cube slowly rotates clockwise. The red cube quickly rotates anticlockwise." (refer to equation (6.9)).

The approximate measurement values that correspond to the averaged fitness values illustrated in figure 6.20 are passed to a representation system by means of a generated element class. The CVTNG subsystem for assigning measurement values to attribute variables and the representation system interact as described in section 6.2. Figure 6.21 illustrates the graphical output produced for time values $t = \pi/2$ and $t = \pi$ when the CVTNG system is applied to the sample sentences, "The red cube slowly rotates clockwise. The red cube quickly rotates anticlockwise.".

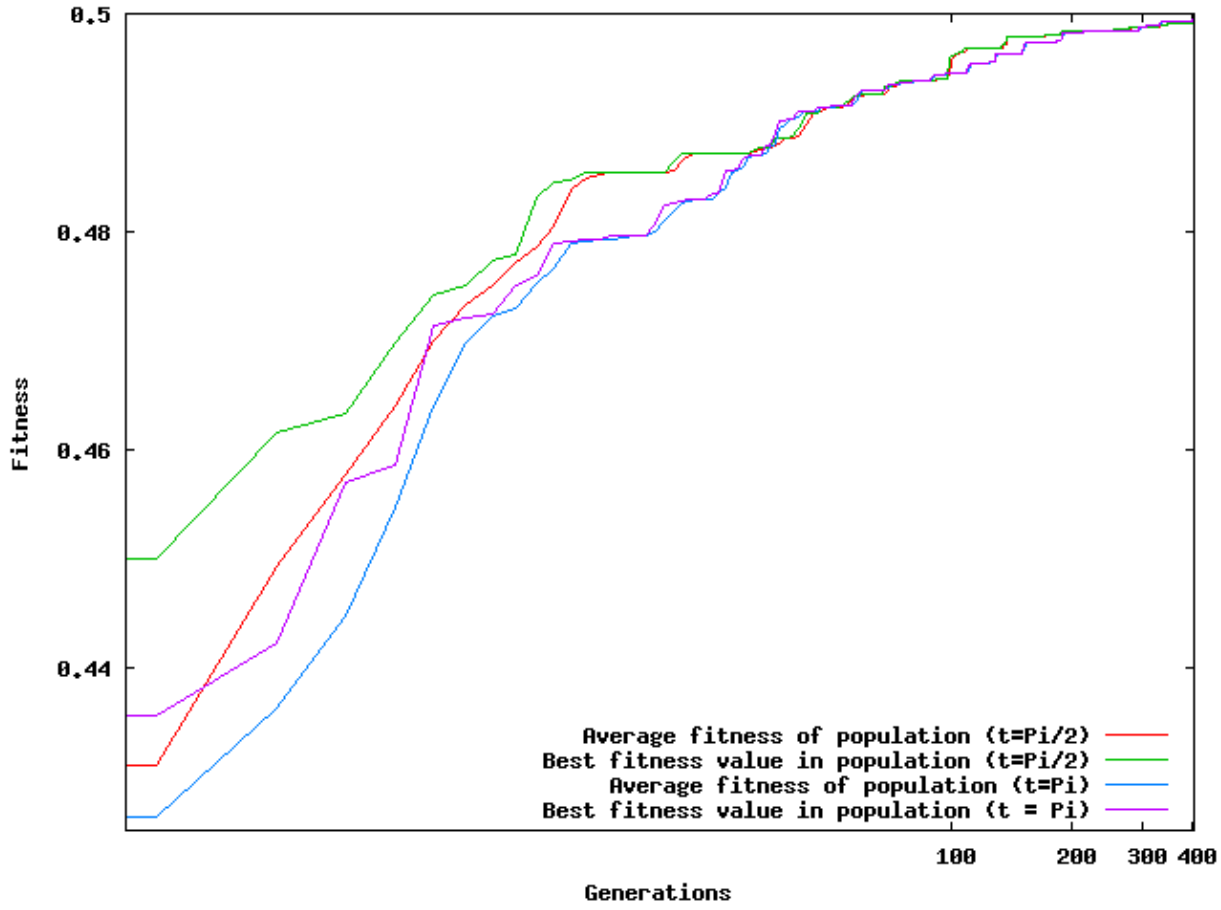

Figure 6.20: The solution accuracy of the genetic algorithm for fuzzy constraint satisfaction when applied to the sample sentences, "The red cube slowly rotates clockwise. The red cube quickly rotates anticlockwise."

6.3 Dynamic examples

This section discusses the actions of the CVTNG system when applied to represent visually natural language sentences that describe changes in the state of an object. The dynamic actions that correspond to verbs in natural language sentences are modelled according to node computational verbs (refer to definition 3.16). The nouns of the natural language sentences are modelled in terms of 3D graphical models. The adjectives of the sample sentence are modelled as static (refer to definition 3.15) or centre computational verbs (refer to definition 3.19). The adverbs of the natural language sentences presented

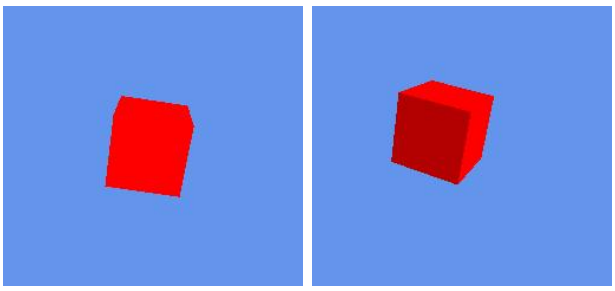

Figure 6.21: The visual output produced for the sample sentences, "The red cube slowly rotates clockwise. The red cube quickly rotates anticlockwise.", for time values $t = \pi/2$ and $t = \pi$

as examples are modelled as adverbs of amplitude and adverbs of period that correspond to the transformations of the amplitude and period of evolving functions defined in the same context of interpretation, subcontext, or partial context.

Section 6.3.1 discusses the evolving function models of the words of the sample sentence presented in this section. The actions of the sequencing algorithm introduced in section 5.5.5 as applied to the sample sentences of this section, are also discussed.

The actions of the parsing subsystem, the subsystem for the assigning measurement values to attribute variables, and the interaction between the CVTNG system and a representation system are discussed in section 6.3.2. The basic examples presented in section 6.3.1 are extended in section 6.3.3 by the introduction of adverbs of amplitude and adverbs of period to the evolving function models of verbs that describe changes in state. The actions of the CVTNG system are discussed in section 6.3.4, when the CVTNG system is applied to sample sentences that describes conflicting changes in state.

6.3.1 Modeling

This section describes the evolving function models of natural language verbs that describe changes in the state of objects. The objects correspond to the subject and object nouns of the natural language verbs in the sample sentences, "The red cube is to the left of the blue cube. The red cube moves onto the blue cube.", that are provided as input to the CVTNG system. The evolving function models of the natural language

verbs in the sample sentences, "The red cube is to the left of the blue cube. The red cube moves onto the blue cube.", are also described in terms of the CVTNG sequencing algorithm (refer to section 5.5.5). The CVTNG sequencing algorithm determines the time interval for the evolving function model of a natural language verb in relation to the time intervals of other natural language verbs. The natural language verbs whose time intervals are determined by the CVTNG sequencing algorithm are parts of natural language sentences provided as input to the CVTNG system.

The details of the evolving function models of the verb phrases "is left of" and "moves onto" used in the sample sentences, "The red cube is to the left of the blue cube. The red cube moves onto the blue cube.", are provided in table 6.3. The verb phrase "is left of" describes a static relationships between two objects named by its subject and object nouns. The verb phrase "is left of" is therefore modelled as a static computational verb (refer to definition 3.15) in the context of interpretation "SpatialX", the subcontexts "SpatialStartX", and the partial contexts "SpatialStartReferenceX", "SpatialStartSelfX", "SpatialStartOtherX".

The verb phrase, "is left of", is also modelled within the newly introduced "SequenceX" context of interpretation. The "SequenceX" context of interpretation serves as a grouping for evolving function models that describe the state and change of state for a point of reference along the x-axis of a 3D visual space. The point of reference modelled corresponds to the perceived starting point of a movement in a 3D visual space. The movement is a change of position for an object named by a natural language noun. The natural language noun serves as the subject and object noun of the natural language verb that names the movement. The movement is modelled by the evolving function model of the corresponding natural language verb.

The "SequenceY" context of interpretation described models the state and changes of state for a point of reference along the y-axis of a 3D space. The subcontexts "SequenceStartX" and "SequenceStartY" serve to group the component of an evolving function model that describes the start state of a point of reference along the x-axis and y-axis of a 3D visual space respectively. The subcontexts "SequenceChangeX", and "SequenceChangeY" serve to group the component of an evolving function model that describes the change in state of a point of reference along the x-axis and y-axis of a 3D

visual space respectively.

The "SequenceX" and "SequenceY" contexts of interpretation with their associated subcontexts and partial contexts allow a sequence of movements in a 3D space to be modelled. The "SequenceX" and "SequenceY" contexts of interpretation allow evolving function models to be specified for a point of reference along the x-axis and y-axis respectively. The measurement values determined by evolving function models specified within the "SequenceX" and "SequenceY" contexts of interpretation are stored in the attribute variables "StartXPosition" and "StartYPosition" respectively. The use of the attribute variables "StartPositionX" and "StartPositionY" as amplitude values or partial amplitude values in the evolving function models specified in the contexts of interpretation "SpatialX" and "SpatialY" allow movements to be modelled in relation to a point of reference determined by other movements. The evolving function models of verbs that model movements in a 3D visual space are therefore able to model sequences of movements. The sequences of movements are named by natural language verbs that occur in a set of related natural language sentences that are provided as input to the CVTNG system.

The verb phrase,"moves onto", is modelled in the contexts of interpretation "SpatialX", "SpatialY", "SequenceX", and "SequenceY". Figure 6.22 illustrates the contexts of interpretation, subcontexts, and partial contexts that the verb phrase "moves onto" is modelled in. The evolving function models for the verb phrases "is left of" and "moves onto" were specified using the "Word Editor" interface component of the CVTNG system (refer to section 5.4.1). The newly introduced contexts of interpretation, "SequenceX" and "SequenceY", along with their associated subcontexts and partial contexts were defined using the "Context editor" interface component of the CVTNG system (refer to section 5.3). The generic functions specified in table 6.3 were implemented as specifications of the `Function` interface in the function library (refer to section 5.4.7).

Table 6.3 indicates the time intervals associated with the evolving function models specified. The time intervals are specified in seconds and are either instant, terminate, or indefinite. The time intervals specified for the evolving function models are used by the sequencing algorithm (refer to algorithms 2 and 3) discussed in section 5.5.5 to determine the validity periods of the associated evolving function models. The validity period of

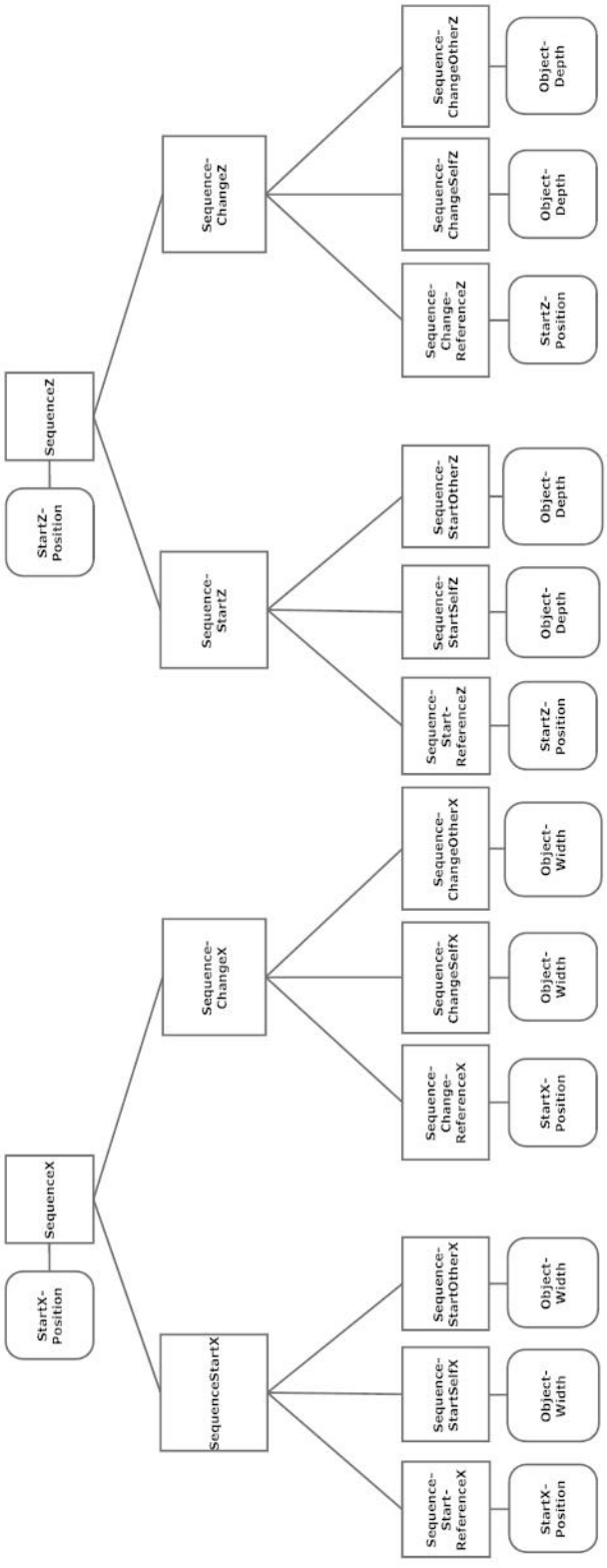

Figure 6.22: Additional contexts of interpretation, subcontexts, and partial contexts introduced for the sample sentences, "The red cube is to the left of the blue cube. The red cube moves onto the blue cube."

Word	Context	Attribute variable	Scaling factorκ	Generic function f	Amplitude value α	Transformation function $\Psi(x)$	Time interval
is to the left of	SpatialX	XPosition					$t = 0$
	SpatialStartX			$f_{one}(t) = 1$			$t = 0$
	SSReferenceX		$\kappa_1 = 1.0$		XPosition		$t = 0$
	SSSelfX		$\kappa_2 = -1.0$		Width**		$t = 0$
	SSOtherX		$\kappa_3 = -1.0$		Width		$t = 0$
	SpatialY	YPosition	$\kappa = 1.0$	$f_{one}(t) = 1$	YPosition		$t = 0$
	SequenceX	StartXPosition	$\kappa = 1.0$	$f_{one}(t) = 1$	XPosition**		$t = 0$
	SequenceY	StartYPosition	$\kappa = 1.0$	$f_{one}(t) = 1$	YPosition**		$t = 0$
moves onto	SequenceX	StartXPosition					$0 \leq t \leq 1$
	SequenceStartX		$\kappa_1 = 1.0$	$f_{stepdownat1}(t)$ $= 1, t < 1$ $= 0, t \geq 1$	StartXPosition**		
	SequenceChangeX		$\kappa_2 = 1.0$	$f_{stepupat1}(t)$ $= 0, t < 1$ $= 1, t \geq 1$	XPosition		
	SequenceY	StartYPosition					$0 \leq t \leq 1$
	SequenceStartY		$\kappa_1 = 1.0$	$f_{stepdownat1}(t)$ $= 1, t < 1$ $= 0, t \geq 1$	StartYPosition**		
	SequenceChangeY		$\kappa_2 = 1.0$	$f_{stepupat1}(t)$ $= 0, t < 1$ $= 1, t \geq 1$	YPosition		
	SpatialX	XPosition					$0 \leq t \leq 1$
	SpatialStartX			$f_{lineardown}(t)$ $= 1 - t$			
	SSReferenceX		$\kappa_1 = 1.0$		StartXPosition**		
	SpatialChangeX			$f_{linear}(t) = t$			
	SCReferenceX		$\kappa_1 = 1.0$		XPosition		
	SpatialY	YPosition					$0 \leq t \leq 1$
	SpatialStartY			$f_{lineardown}(t)$ $= 1 - t$			
	SSReferenceY		$\kappa_1 = 1.0$		StartYPosition**		
	SpatialChangeY			$f_{linear}(t) = t$			$0 \leq t \leq 1$
	SCReferenceY		$\kappa_1 = 1.0$		YPosition		
	SCOtherY		$\kappa_2 = 0.5$		Height		
	SCSelfY		$\kappa_3 = 0.5$		Height**		
moves left	SpatialX	XPosition	$\kappa = 1.0$	$f_{lineardown} = 1 - t$	1.0		$t \geq 0$
moves right	SpatialX	XPosition	$\kappa = 1.0$	$f_{linear} = 1 - t$	1.0		$t \geq 0$
slowly	SpatialX					$\Psi_{half}(x) = 0.5x$	
	SequenceX					$\Psi_{half}(x) = 0.5x$	
quickly	SpatialX					$\Psi_{two}(x) = 2.0x$	
	SequenceX					$\Psi_{two}(x) = 2.0x$	

Table 6.3: Dynamic word definitions

an evolving function model dictates the time values at which the evolving function is evaluated to retrieve measurement value for attribute variables.

Evolving function models with instant time intervals are specified by intervals of, $t = 0$, in table 6.3. Evolving function models that have an instant time interval are evaluated a single time within the CVTNG system. The measurement value retrieved from an evolving function model at a single time instant is stored within the associated attribute variable. The attribute variable retains the measurement value assigned by the evolving function model evaluated for the single time instant until the attribute variable is assigned a measurement value by another evolving function model.

An evolving function model that has an infinite time interval is specified in table 6.3 by the time period $t > 0$. Evolving function models that have an infinite time interval have a start time determined by the CVTNG sequencing algorithm (refer to section 5.5.5). An evolving function model with an infinite time interval is evaluated for all time values after the start time determined by the CVTNG sequencing algorithm.

Table 6.3 specifies a finite time interval for an associated evolving function model as $0 < t < x$. The value x is a real-valued number that specifies the size of the time interval specified for an associated evolving function model. An evolving function model with a finite time period has its start time determined by the CVTNG sequencing algorithm. An evolving function model with a finite time interval is evaluated from a start time determined by the CVTNG sequencing algorithm to determine a measurement value for the associated attribute variable. The evolving function model with a finite time period is evaluated up to an end time determined by the CVTNG sequencing algorithm. The CVTNG system evaluates an evolving function model with a finite time period at the start and end times determined by the CVTNG sequencing algorithm to ensure continuity in the associated representational media (refer to section 5.5.5). If the actual time, t, that an evolving function model with finite period is evaluated at is greater than the end time determined by the sequencing algorithm of the CVTNG system for that evolving function model, a modified time value, $t*$, is passed as a parameter to the evolving function model and it is equal to the end time determined for the evolving function model.

An evolving function model defined to be repeatedly evaluated over a specified time

interval is annotated with an asterisk (*). Evolving functions defined to be repeatedly executed over the same time interval take the form of equation (4.4). The actual time parameter, t, is adapted for evolving function models of the form of equation (4.4) to fall within the time interval specified for the evolving function model.

The time intervals specified for the evolving function models with a finite time interval (refer to table 6.3) are static. An evolving function model with a time interval of static size is always evaluated for the same time interval. The movement in a 3D visual space that is modelled by means of the evolving function model with a finite period of static size will therefore also always occur over the same time period. The evolving function models specified in table 6.3 and the CVTNG system will therefore model and subsequently represent actions over a fixed period of time. An action in the representative space will occur over the time interval specified for the associated evolving function regardless of the amplitude values that scale the associated evolving functions and therefore the representation of the actions in the 3D space. The visual representation of the verb phrase, "moves onto", will, for example, depict the movement of the visual representation of the subject noun onto the object noun within the representational space over the same period of time, regardless of the starting position of the visual depiction of the subject noun. The limitation of modelling evolving functions over static-sized intervals can be overcome by dynamically scaling the interval sizes of evolving function models based on external factors such as amplitude values. The dynamic determination of evolving function time intervals does not fall within the scope of this work.

6.3.2 Parsing, variable resolution, and representation

The actions of the CVTNG parsing procedure applied to the sample sentences, "The red cube is to the left of the blue cube. The red cube moves onto the blue cube.", are similar to those described in section 6.1.3. Figure 6.23 illustrates the parse bin relationships produced for the sample sentences, "The red cube is to the left of the blue cube. The red cube moves onto the blue cube.". The attribute variable relationships produced from the parse bin relationships illustrated in figure 6.23 and the evolving function models specified in table 6.3 are shown in figure 6.24. The edges that connect the attribute variables are not only annotated by the generic functions, f_i, and scaling factors, κ_i,

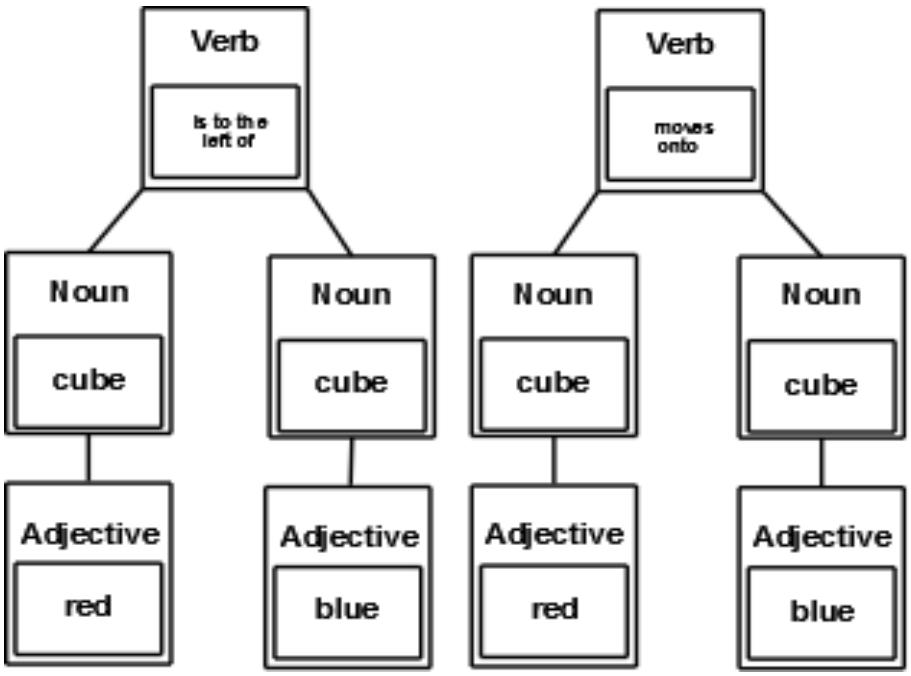

Figure 6.23: The parse bin relationships produced for the sample sentences, "The red cube is to the left of the blue cube. The red cube moves onto the blue cube."

associated with the attribute variables in the specified evolving function models. The edges are also annotated with a time interval that specifies the validity periods for the attribute variable relationship illustrated by a particular edge. The validity periods are determined by the sequencing algorithm within the CVTNG system (refer to section 5.5.5). The validity periods determined are based on the time intervals that are specified for the evolving function models containing the attribute variable relationships shown in figure 6.24.

The evolving function equations produced by the parsing subsystem of the CVTNG system when applied to the sample sentences,"The red cube is to the left of the blue cube. The red cube moves onto the blue cube.", are:

$$\begin{aligned} Red(Red) &= 0.7 f_{one} \kappa; t \geq 0 \\ Blue(Blue) &= 0.7 f_{one} \kappa; t \geq 0 \end{aligned} \tag{6.10}$$

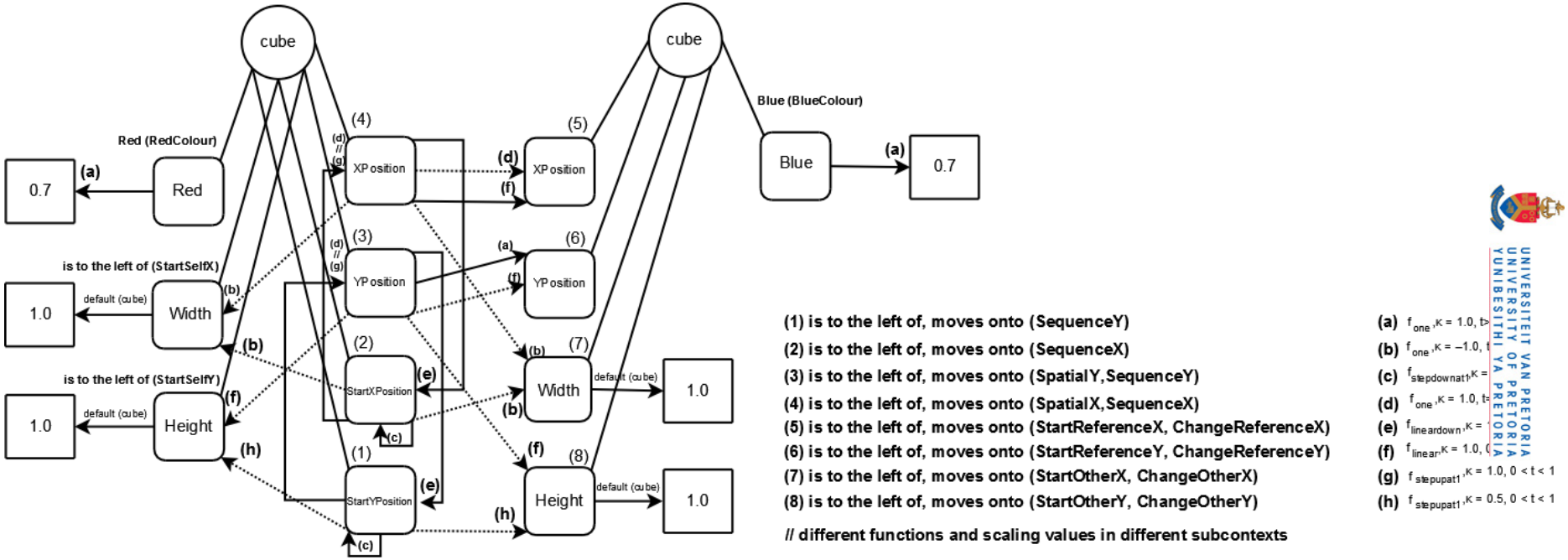

Figure 6.24: The attribute variable relationships produced by resolving the parse bins in figure 6.28

$$\begin{aligned}
XPosition(Red) &= f_{one}(t)\kappa_1 Width(R) + f_{one}(t)\kappa_2 Width(B) \\
&+ f_{one}(t)\kappa_3 XPosition(B), t = 0 \\
YPosition(Red) &= f_{one}(t)\kappa_1 YPosition(B), t = 0 \\
StartXPosition(Red) &= f_{one}(t)\kappa_1 XPosition(R), t = 0 \\
StartYPosition(Red) &= f_{one}(t)\kappa_1 YPosition(R), t = 0
\end{aligned} \tag{6.11}$$

$$\begin{aligned}
XPosition(Red) &= f_{LinearDown}(t)\kappa_1 StartXPosition(R) \\
&+ f_{Linear}(t)\kappa_2 XPosition(B), 0 < t < 1 \\
YPosition(Red) &= f_{LinearDown}(t)\kappa_1 StartYPosition(R) + f_{Linear}(t)\kappa_2 Height(B) \\
&+ f_{Linear}(t)\kappa_3 Height(R) + f_{Linear}(t)\kappa_4 YPosition(B), 0 < t < 1 \\
StartXPosition(Red) &= f_{StepUpAt1}(t)\kappa_1 XPosition(R) \\
&+ f_{StepDownAt1}(t)\kappa_2 StartXPosition(R), 0 < t < 1 \\
StartYPosition(Red) &= f_{StepUpAt1}(t)\kappa_1 YPosition(R) \\
&+ f_{StepDownAt1}(t)\kappa_2 StartYPosition(R), 0 < t < 1
\end{aligned} \tag{6.12}$$

.

The annotation "(R)" refers to the instance of the noun "cube" that is described by the adjective "red" in the natural language text provided as input to the CVTNG system. The annotation "(B)" refers to the instance of the noun "cube" that is described by the adjective "blue" in the natural language text provided as input to the CVTNG system. The generic functions, f_{one},f_{Linear}, $f_{LinearDown}$, $f_{StepUpAt1}$, and $f_{StepDownAt1}$ are defined as for equations (4.3) and (4.4). The details of the generic functions, f_{one},f_{Linear}, $f_{LinearDown}$, $f_{StepUpAt1}$, and $f_{StepDownAt1}$, are specified in table 6.3. The scaling factors, κ_i, are defined as they are for equations (4.3) and (4.4). The scaling factors, κ_i, have values as specified in table 6.3 for the evolving function models of the natural language words that the scaling factors correspond to. The time value, t, is a real-valued number provided by a clock in the CVTNG system.

If the evolving function model of a natural language word for a specific context of interpretation is defined to repeat over a specified time interval, the evolving function equation takes the form of equation (4.4) and the time value, t, is transformed to fall within the interval specified for the evolving function equation. The time value, t, is specified in seconds. The time value, t, has a value of $t = 0$ at the first iteration of the algorithms within the CVTNG subsystem for assigning measurement values to attribute variables.

The system of evolving function equations (refer to equations (6.10) through (6.12)) produced for the sample sentences, "The red cube is to the left of the blue cube. The red cube moves onto the blue cube.", is similar to the systems of evolving function equations produced for the static and periodic sample sentences of the previous sections. The system of evolving function equations given by equations (6.10) through (6.12) differs from the other systems of evolving function equations associated with previous examples, with respect to the validity period attached to the evolving function equations.

The set of evolving function equations given by equation (6.11) is instantaneous with regard to the time interval specified for them. The time interval determined for the set of evolving function equations given by equation (6.11) by the CVTNG sequencing algorithm is $t = 0$. The set of evolving function equations given by equation (6.11), therefore only holds for the first iteration of the algorithms within the CVTNG subsystem for assigning measurement values to attribute variables.

The set of evolving functions equations given by equation (6.12) have an associated time interval of static size. The set of evolving function equations given by equation (6.12) is determined by the CVTNG sequencing algorithm to be valid in the time interval $0 < t \leq 1$.

The set of the evolving function equations given by equation (6.10) are valid for all time values. The time interval determined for the set of evolving function equations given by equation (6.10) is $t \geq 0$ as determined by the CVTNG sequencing algorithm.

The algorithms within the CVTNG subsystem for assigning measurement values to attribute variables are executed once the valid evolving function models have been determined. The recursive substitution, Gaussian elimination, and genetic algorithm for the resolution of FCSPs are executed for the evolving functions determined to be

valid for the current time value. The evolving functions that are valid for a specific time value are those evolving functions whose time intervals contain the current time value. Evolving functions determined to repeat over a specific time interval are valid for all time values after start time determined by the CVTNG sequencing algorithm. Evolving function equations that repeat over a specified time interval take the form of equation (4.4) and therefore modify the time value to fall within the time period specified for the evolving function model.

Evolving function models that are defined to have a finite time interval are evaluated at least at the start time and then end time determined for the evolving function models by the CVTNG sequencing algorithm. The evolving function models for the attribute variables, "StartXPosition" and "StartYPosition", depend on the feature of the CVTNG system that evaluates an evolving function model at the determined start and end times. The attribute variables "StartXPosition" and "StartYPosition" specified within the "SequenceX" and "SequenceY" contexts of interpretation serve to model a point of reference for a movement along the x-axis and the y-axis with respect to the position of the point of reference along the x-axis and y-axis respectively. If the movement over the x-axis and the y-axis has concluded, the point of reference shifts and the position of the point of reference changes along both the x-axis and the y-axis. The end positions of a movement along the x-axis and y-axis become the new reference point for the next movement.

The instantaneous jump of a reference point when one observed movement transitions into another discernible observed movement is modelled by means of the generic functions, $f_{StepUpAt1}$ and $f_{StepDownAt1}$. The generic functions, $f_{StepUpAt1}$ and $f_{StepDownAt1}$, evaluate to values of 1 and 0 respectively at the end time defined for the evolving function model. The evolving function models associated with the attribute variables, "StartXPosition" and "StartYPosition", are defined to repeat over the interval $0 \geq t \leq 1$. If the generic functions, $f_{StepUpAt1}$ and $f_{StepDownAt1}$, are scaled by amplitude values equal to the starting position and current position of an object whose movement is modelled within another evolving function model, the reference point is modelled to shift from the starting position of the modelled movement to the current position of the modelled movement.

The generic function, $f_{StepUpAt1}$, evaluates to a value of 1 only at the end of the time

intervals associated with the evolving function models that assign measurement values to the attribute variables, "StartXPosition", and "StartYPosition". The inverse of the previous statement holds true for the generic function, $f_{StepDownAt1}$, which evaluates to 0 only at the end of the time intervals defined for the evolving function equations that assign measurement values to the attribute variables, "StartXPosition" and "StartYPosition". The point of reference for a movement therefore shifts to the current position of a movement only at the end of the time intervals specified for the evolving function equations defined in terms of the generic functions, $f_{StepUpAt1}$ and $f_{StepDownAt1}$. The point of reference for a movement is therefore modelled as shifting from the start point to the end point of the movement. The evolving function equations associated with the attribute variables, "StartXPosition" and "StartYPosition", evaluate to measurement values equal to the current position of a reference point along the x-axis and the y-axis if the time value is not equal to the determined end time of the respective evolving function models.

Figure 6.24 illustrates the cyclical relationships from the attribute variables, "StartXPosition" and "StartPosition", to themselves. The cyclical attribute variable relationships are created by the parsing subsystem of the CVTNG system when the sample sentences, "The red cube is to the left of the blue cube. The red cube moves onto the blue cube.", are processed. The recursive substitution algorithm cannot determine the value of the attribute variables, "StartXPosition" and "StartYPosition", because the cyclical call of the recursive substitution algorithm for the attribute variable relationships to themselves are pre-empted by the recursive substitution algorithm (refer to section 5.6.3.1). Termination of the algorithm when a cycle is detected in the attribute variable relationships prevents an endless loop from occurring in the recursive substitution algorithm. The measurement values of the attribute variables, "StartXPosition" and "StartYPosition", are therefore undetermined when the recursive substitution algorithm terminates. The evolving functions that assign measurement values to the attribute variables "XPosition" and "YPosition" use the attribute variables, "StartXPosition" and "StartYPosition", as amplitude values respectively. The recursive substitution algorithm can, as a result, not calculate the measurement value of the attribute variables, "XPosition" and "YPosition", from the evolving functions that use "StartXPosition"

and "StartYPosition" as amplitude values. The measurement values of the attribute variables "XPosition" and "YPosition" are therefore undetermined when the recursive substitution algorithm terminates.

The evolving function equations that contain the attribute variables "XPosition", "YPosition", "StartXPosition", and "StartYPosition" are evaluated for a specific time value to form evaluated evolving function equations (refer to section 5.6.3.2). The EEFEs are rewritten into a form that isolates the determined values as a single constant, K, on the right-hand side of the rewritten EEFE. Coefficient values are determined from the products of generic functions, f_i, that are evaluated at the specific time value and scalar values, k_i, specified for the terms of the corresponding evolving function models. The coefficients, c_i, on the left-hand side of the EEFEs are grouped into a coefficient matrix A. The rows of the coefficient matrix A correspond to the evolving function equations that contain attribute variables "XPosition", "YPosition", "StartXPosition", and "StartYPosition". The coefficient matrix, A, is augmented with a column vector, $\mathbf{K}$, of the constant, K, values on the right hand sides of the EEFEs that correspond to the attribute variables "XPosition", "YPosition", "StartXPosition", and "StartYPosition". The Gaussian elimination algorithm (refer to section 5.6.3.3) reduces the augmented matrix, $[A|\mathbf{K}]$, to upper triangular form. The measurement values of the attribute variables "XPosition", "YPosition", "StartXPosition", and "StartYPosition" are determined by back substitution (refer to section 5.6.3.3).

The measurement values determined for the attribute variables in figure 6.24 at a specific time value are passed as parameters to transformations applied to a representational element by means of an element class. The interaction between the subsystem for assigning measurement values to attribute variables and the representational elements associated with the noun "cube" occur as described in section 6.1.5. Figure 6.25 illustrates the visual output produced when the sample sentences, "The red cube is to the left of the blue cube. The red cube moves onto the blue cube.", are processed by the CVTNG system and the resulting measurement values are applied to a representational element in a 3D visual space. If the measurement–value calculation and the visual output are repeated for a sequence of time values, an animation is produced that initially shows the red cube to the left of the blue cube and then moves the red cube onto the

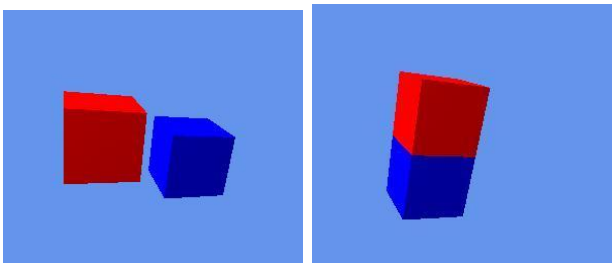

Figure 6.25: The visual output produced for the sample sentences, "The red cube is to the left of the blue cube. The red cube moves onto the blue cube.", for time values of $t = 0.5$ and $t = 1.0$

blue cube.

6.3.3 Adverbs

This section discusses the effect of adverbs on the behaviour of the CVTNG system when the CVTNG system is applied to natural language sentences that described changes in state. The adverbs are modelled as adverbs of period and adverbs of time in terms the computational verb models of the CVTNG system (refer to section 4.3.4). The sample sentences, "The red cube is to the left of the blue cube. The red cube moves onto the blue cube.", are extended by the introduction of the adverbs "far" and "quickly" to form the sample sentences, "The red cube is far to the left of the blue cube. The red cube quickly moves onto blue cube.".

The evolving function models for the adverbs "far" and "quickly" were retained and are specified in tables 6.2 and 6.3 respectively. The evolving function models were specified by means of the "Word editor" graphical user interface component of the CVTNG system. The generic functions utilised with the evolving function word models of the sample sentences were implemented within the `Function` library (refer to section 5.4.7).

The actions of the CVTNG parsing subsystem are similar to those described in section 6.1.3. The parse bin relationships illustrated in figure 6.26 are produced by the CVTNG parsing algorithm. The attribute variable relationships produced from the parse bin relationships shown in figure 6.26 and the evolving function models specified in table 6.3 are similar to those illustrated in figure 6.24. The evolving function models produced

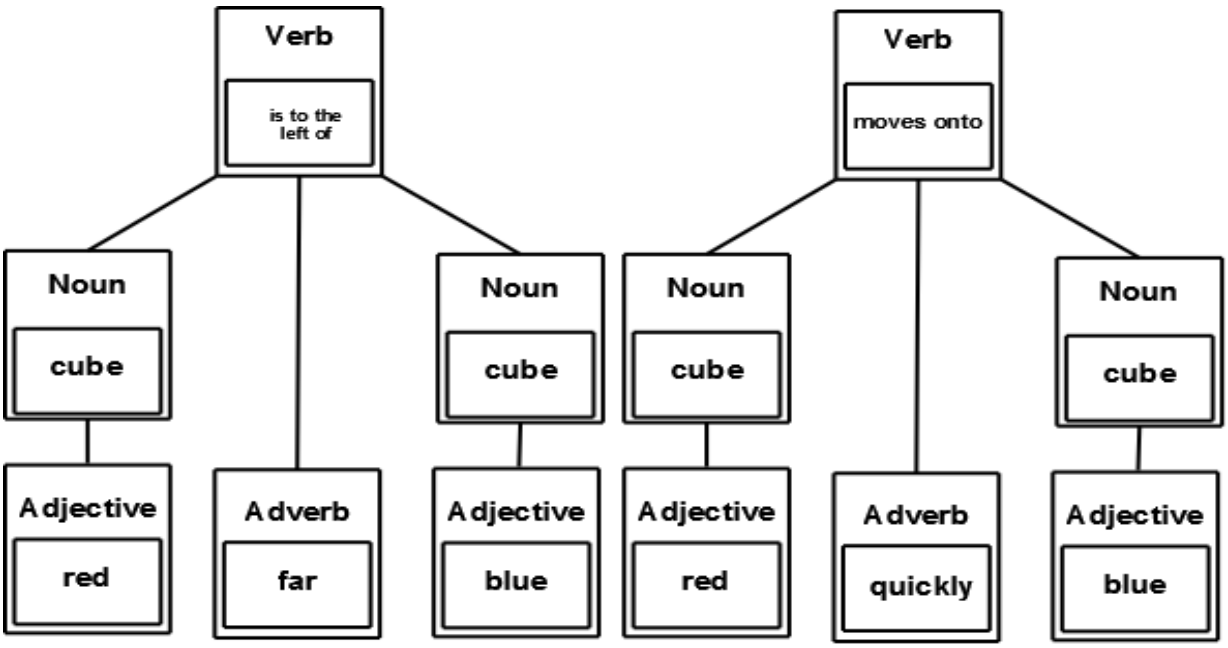

Figure 6.26: The parse bin relationships produced for the sample sentences, "The red cube is far to the left of the blue cube. The red cube quickly moves onto the blue cube."

for the sample sentences, "The red cube is far to the left of the blue cube. The red cube quickly moves onto blue cube.", differ from the evolving function models produced for the sample sentences, "The red cube is to the left of the blue cube. The red cube moves onto blue cube.", in terms of transformation functions applied to certain period values and amplitude values. The period of the generic functions, f_{Linear} and $f_{LinearDown}$, are transformed by the transformation function, Ψ_{Double}, specified within the evolving function model of the adverb, "quickly". The partial amplitude values, α_{1_1} and α_{1_2}, of the evolving function model for the verb phrase, "is left", are transformed by the transformation function, $\Psi_{OneAndHalf}$, specified within the evolving function model of the adverb, "far".

The CVTNG system produces the following set of evolving function equations when the evolving function models specified in table 6.3 are applied to model the natural language sentences "The red cube is far to the left of the blue cube. The red cube

quickly moves onto blue cube.":

$$
\begin{aligned}
Red(Red) &= 0.7, t \geq 0 \\
Blue(Blue) &= 0.7, t \geq 0 \\
XPosition(Red) &= f_{one}(t)\kappa_1(\Psi_{Double}(t) \circ Width(R)) \\
&+ f_{one}(t)\kappa_2(\Psi_{Double}(t) \circ Width(B)) \\
&+ f_{one}(t)\kappa_3 XPosition(B), t = 0 \\
YPosition(Red) &= f_{one}(t)\kappa_1 YPosition(B), t = 0 \\
StartXPosition(Red) &= f_{one}(t)\kappa_1(\Psi_{Double} \circ XPosition(R)), t = 0 \\
StartYPosition(Red) &= f_{one}(t)\kappa_1 YPosition(R), t = 0 \\
XPosition(Red) &= f_{LinearDown}(\Psi_{Double}(t) \circ t)\kappa_1 StartXPosition(R) \\
&+ f_{Linear}(\Psi_{Double}(t) \circ t)\kappa_2 XPosition(B), 0 < t < 1 \\
YPosition(Red) &= f_{LinearDown}(\Psi_{Double}(t) \circ t)\kappa_1 StartYPosition(R) \\
&+ f_{Linear}(\Psi_{Double}(t) \circ t)\kappa_2 Height(B) \\
&+ f_{Linear}(\Psi_{Double}(t) \circ t)\kappa_3 Height(R) \\
&+ f_{Linear}(\Psi_{Double}(t) \circ t)\kappa_4 YPosition(B), 0 < t < 1 \\
StartXPosition(Red) &= f_{StepUpAt1}(\Psi_{Double}(t) \circ t)\kappa_1 XPosition(R) \\
&+ f_{StepDownAt1}(\Psi_{Double}(t) \circ t)\kappa_2 StartXPosition(R), 0 < t < 1 \\
StartYPosition(Red) &= f_{StepUpAt1}(\Psi_{Double}(t) \circ t)\kappa_1 YPosition(R) \\
&+ f_{StepDownAt1}(\Psi_{Double}(t) \circ t)\kappa_2 StartYPosition(R), 0 < t < 1
\end{aligned}
\tag{6.13}
$$

The annotation "(R)" refers to the instance of the noun "cube" described by the adjective "red" in the natural language text. The annotation "(B)" refers to the instance of the noun "cube" described by the adjective "blue" in the natural language text. The generic functions f_{Linear},$f_{LinearDown}$,$f_{StepUpAt1}$, and $f_{StepDownAt1}$ are defined as in section 6.3.2. The transformation function, Ψ_{Double}, is used to transform time values and amplitude values when specified in the definition of an adverb of period (refer to equation 4.21) and an adverb of amplitude (refer to equation 4.20) respectively. The scaling values, κ_i, and the time value, t, are defined in section 6.1.4.

The actions of the CVTNG system for assigning measurement values to attribute variables for the sample sentences, "The red cube is far to the left of the blue cube. The red cube quickly moves onto blue cube.", are similar to those for the sample sentences, "The red cube is to the left of the blue cube. The red cube moves onto blue cube.", discussed in section 6.3.2. The time values passed to the generic functions, f_{Linear}, $f_{LinearDown}$, $f_{StepUpAt1}$, and $f_{StepDownAt1}$, are transformed by the transformation function, Ψ_{Double}. The transformations are performed by forming a composition of the transformation function, Ψ_{Double}, and the time value passed to an associated generic function. The composition operation is performed by evaluating the transformation function, Ψ_{Double}, for the time value passed to the respective associated generic functions, f_{Linear}, $f_{LinearDown}$, $f_{StepUpAt1}$, and $f_{StepDownAt1}$, to obtain a transformed time value.

If the generic function is defined over an infinite time interval, it is evaluated for double the current time value, t. The dynamics modelled by the evolving function with an infinite time interval therefore appear to occur at double the speed and as a result the transformation function, Ψ_{Double}, serves to model the adverb "quickly" when applied to the time value passed to a generic function. The generic function is specified as part of an evolving function model of an adjective or verb associated with the adverb "quickly".

If the evolving function model is specified to repeat over a specific time interval, it takes the form of equation (4.4). The time value, t, transformed by the transformation function, Ψ_{Double}, to become $2t$ is transformed by an evolving function in the form of equation (4.4) to fall within the time interval specified for the evolving function. If consecutive time values are $t = 1$, $t = 1.25$, $t = 1.5$, they are transformed by the transformation function, Ψ_{Double}, to $t = 2$, $t = 2.5$, and $t = 3$ respectively. If the evolving function to which the transformed time values, $t = 2$, $t = 2.5$, and $t = 3$, are passed to is defined to repeat over the interval, $0 \geq t \leq 1$, then the generic functions within the evolving function are calculated for transformed time values of $t = 0$, $t = 0.5$, and $t = 0$ respectively. The transformed time values of $t = 0$, $t = 0.5$, and $t = 0$ correspond to a full repetition of the evolving function over its defined time interval. The actual time values of $t = 1$, $t = 1.25$, $t = 1.5$, correspond only to half a repetition over the specified time interval if not transformed by function Ψ_{Double}. The transformation function, Ψ_{Double}, causes the evolving function defined to repeat over a specific time interval to repeat more

frequently over a specific time period. The dynamics modelled by an evolving function that is defined to repeat over a specific period therefore appear to repeat at double the speed and the transformation, Ψ_{Double}, once again serves to model the adverb "quickly".

If an evolving function model is specified for a interval of a fixed size, the effect of the transformation function, Ψ_{Double}, is similar to the effect on an evolving function defined to repeat over a specific time interval. The transformed time value, $2t$, more quickly reaches the end value defined for the evolving function model and therefore the dynamics modelled by the evolving function with a fixed time interval occur at double the speed. The transformation function, Ψ_{Double}, once again serves as a model for the adverb "quickly" when applied to an evolving function with a fixed time interval. The evolving function model defined for the verb phrase "moves onto" in table 6.3 serves as an example of an evolving function model defined over an interval with a fixed size.

The use of the transformation function, Ψ_{Double}, in the evolving function model of the adverb of amplitude (refer to equation 4.20) "far" is as described in section 6.1.6. The remainder of the processes within the CVTNG subsystem for assigning measurement values to attribute variables are the same as the actions described in section 6.3.2. The measurement values determined from the transformed evolving function models that correspond to the natural language sentences, "The red cube is far to the left of the blue cube. The red cube quickly moves onto blue cube", are passed to a representation system by means of the element class associated with the noun "cube". Figure 6.27 illustrates the visual output produced when the sample sentences are processed by the CVTNG system and the resulting measurement values are used to transform a representational means in a 3D visual space.

6.3.4 Inconsistency handling

This section examines the actions of the CVTNG system when natural language sentences are processed that describe changes in a conflicting or inconsistent manner. The actions of the CVTNG system are described for the sample sentences, "The red cube slowly moves left. The red cube quickly moves right.". The verb phrases, "moves left" and "moves right", are modelled as node computational verbs (refer to definition 3.16) implemented as evolving functions that have infinite validity periods. If the evolving

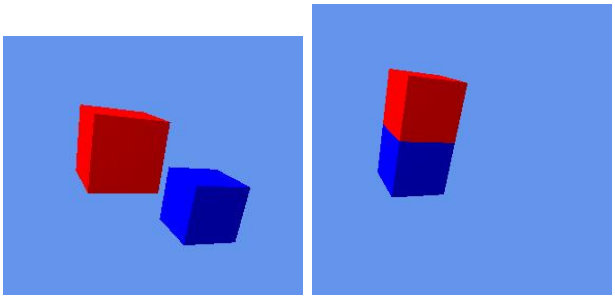

Figure 6.27: The visual output produced for the sample sentences, "The red cube is far to the left of the blue cube. The red cube quickly moves onto the blue cube." for time values of $t = 0.5$ and $t = 1.0$

function models of the verb phrases, "moves left" and "moves right", are implemented as evolving functions with a fixed validity interval, the CVTNG sequencing algorithm (refer to section 5.5.5) assigns consecutive validity intervals to the corresponding evolving function models. The evolving functions that correspond to the verb phrases "moves left" and "moves right" would therefore not assign conflicting measurement values to the same attribute variable, because they are not evaluated for the same time values. The visual representation of the noun "cube" would move to the left and then subsequently move to the right and no inconsistencies with regard to interpretation and representation would occur.

The verb phrases "moves left" and "moves right" were implemented in the context of interpretation, "SpatialX", and the subcontext, "SpatialChangeX", as specified in table 6.3. The word definitions were created by the "Word editor" graphical user interface component of the CVTNG system. The generic functions, f_{Linear} and $f_{LinearDown}$, are defined as in section 6.3.2. The transformation functions, Ψ_{Double} and Ψ_{Half}, are defined as in section 6.2.3. The functions were implemented as specifications of the `Function` interface (refer to section 5.4.7).

The actions of the CVTNG sequencing algorithm are as described in section 6.3.2. The evolving function models of the verb phrases "moves to the left" and "moves to the right" are defined as having infinite validity periods. The evolving function models that correspond to the verb phrases "moves left" and "moves right" are therefore evaluated for all time values that the natural language sentences, "The red cube slowly moves left.

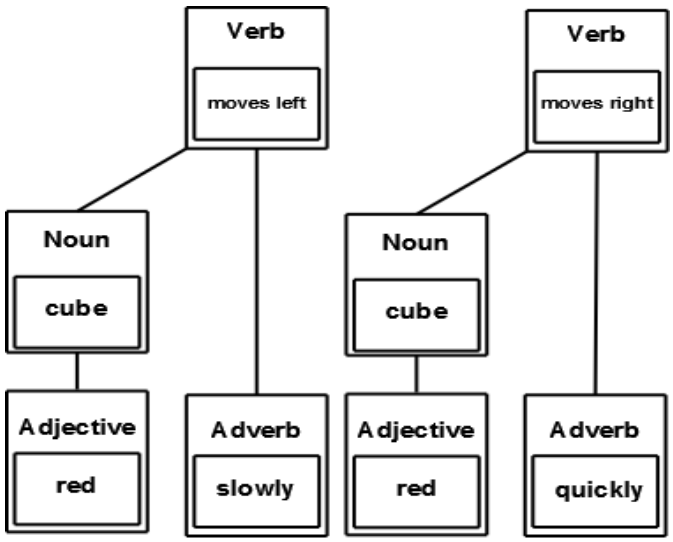

Figure 6.28: The parse bin relationships produced for the sample sentences, "The red cube slowly moves left. The red cube quickly moves right."

The red cube quickly moves right.", are processed for, by the CVTNG system.

The actions of the CVTNG parsing subsystem are similar to those described in section 6.1.3. The parse bin relationships illustrated in figure 6.28 are produced when the CVTNG parsing algorithm is applied to the natural language sentences "The red cube slowly moves left. The red cube quickly moves right.". The parse bin relationships illustrated in figure 6.28 are combined with the evolving function models specified in table 6.3 to form evolving function models that have the attribute variable relationships shown in figure 6.29.

The CVTNG parsing subsystem produces the following evolving function equations when applied to the natural language sentences, "The red cube slowly moves left. The red cube quickly moves right.":

$$\begin{aligned} Red(Red) &= 0.7 \\ XPosition(Red) &= f_{Linear}(t)\kappa_1(1.0); t \geq 0 \\ XPosition(Red) &= f_{LinearDown}(t)\kappa_1(1.0); t \geq 0 \end{aligned} \tag{6.14}$$

The annotations, "(Red)" and "(Blue)", the scaling values, κ_i, and the time value, t, are

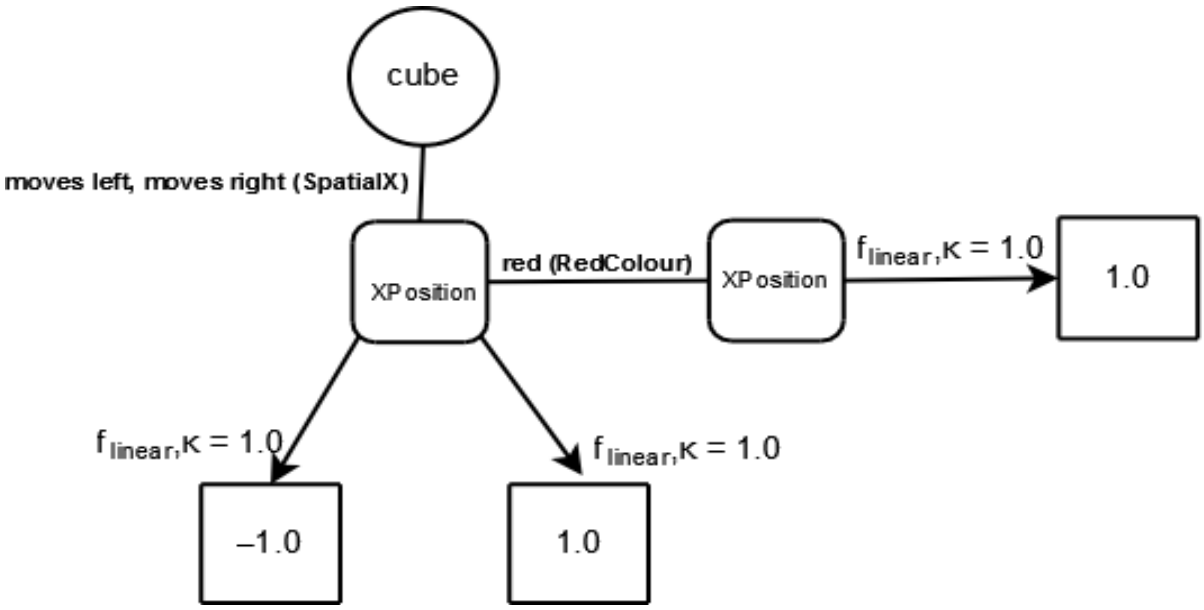

Figure 6.29: The attribute variable relationships produced by resolving the parse bins in figure 6.28

defined as in section 6.1.4. The generic functions, f_{Linear} and $f_{LinearDown}$, are defined as in section 6.3.2. The transformations functions, Ψ_{Double} and Ψ_{Half}, are defined as in section 6.2.3.

The evolving function models given by equation (6.14) clearly show that two conflicting measurement value assignments are made to the same instance of the "XPosition" attribute variable when the corresponding evolving functions are evaluated for a specific time value. The evolving functions that assign measurement values to the attribute variable ,"XPosition", are formulated as fuzzy constraints and so the fuzzy procedure for assigning measurement values to attribute variables is initiated.

The actions of the fuzzy procedure for assigning approximate measurement values to attribute variables were similar to those discussed in section 6.1.7 when applied to the sample sentences, "The red cube slowly moves left. The red cube quickly moves right.". The chromosomes of the genetic algorithm population (refer to algorithm 7) have a single gene that corresponds to the attribute variable, "XPosition(Red)". The population size for algorithm 7 was experimentally determined to provide optimal fitness values for a population of size 4. A blending factor of $\boldsymbol{\alpha} = 0.4$ was experimentally determined to

provide optimal fitness values for algorithm 7.

The optimal blending factor and population size were determined in tandem for different permutations of population size and blending factor. The population sizes ranged between 2 and 20 in increments of 1 and the blending factor ranged between 0.0 and 1.0 in increments of 0.1. All solution values were averaged over 30 iterations of algorithm 7 for a specific population size and blending factor combination. The mutation and crossover probabilities were fixed at 0.1 and 1.0 respectively as discussed in section 6.1.7.

Figure 6.30 shows the averaged and best fitness values for solutions determined by algorithm 7. The average and best fitness values are aggregated over 30 runs of algorithm 7. The aggregated fitness values were determined for time values of $t = 1.5$ and $t = 3.0$ respectively. The best and average fitness values of solutions determined by algorithm 7 are both shown to converge quickly to a near-optimal fitness value. The fuzzy constraints which correspond to the evolving functions that assign measurement values to the attribute variable, "XPosition", are in conflict and therefore only a fitness value of 0.5 is possible and optimal. The fitness of solutions determined for the time value, $t = 3.0$, start at a higher averaged fitness value but converge more slowly to a near-optimal fitness value than the fitness values of solutions determined for the time value, $t = 1.5$. The slower convergence can be ascribed to the larger measurement values assigned to the attribute variable, "XPosition", for the time value, $t = 3.0$. The larger assigned measurement values of 3.0 and -3.0 increase the domain size of the gene that corresponds to attribute variable "XPosition". The larger domain size of the gene constitutes a larger search space for algorithm 7 and a lower rate of fitness value convergence occurs for the chromosomes in the population.

The approximate measurement values determined by the fuzzy procedure for assigning measurement values to attribute variables are passed to a representational medium by means of an element class. The interaction between the representational element and the CVTNG subsystem for assigning measurement values to attribute variables is similar to the actions described in section 6.1.5. Figure 6.31 illustrates the visual output produced when the sample sentence, "The red cube slowly moves left. The red cube quickly moves right.", are processed by the CVTNG system and the approximate measurement

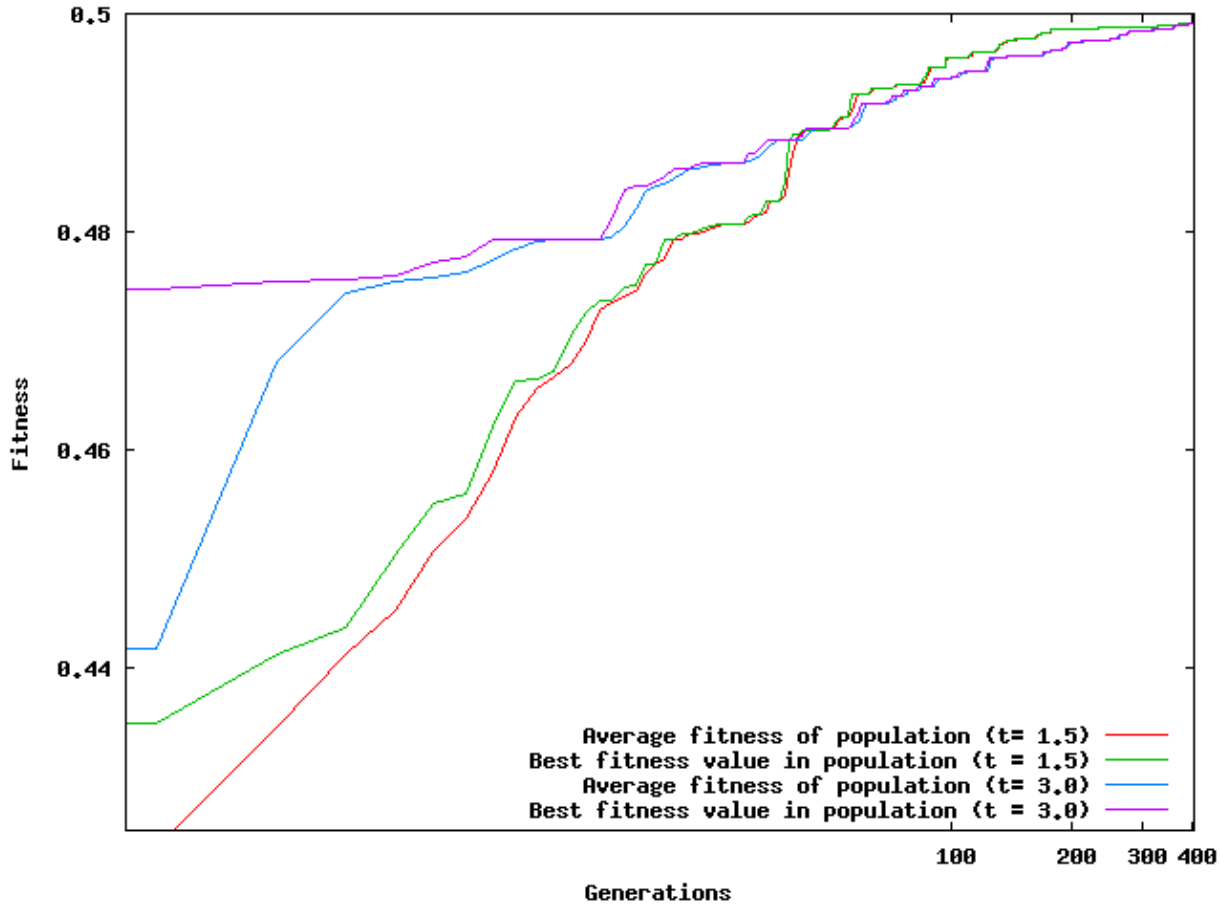

Figure 6.30: The solution accuracy of the genetic algorithm for the resolution of fuzzy constraints when applied to the sample sentences, "The red cube slowly moves left. The red cube quickly moves right."

values determined are used to transform a representational means associated with the noun "cube" in a 3D visual space. The visual output is illustrated for measurement values determined at time values of $t = 1.5$ and $t = 3$ respectively.

The visual output animates the slow movement of a 3D cube to the left, with slight deviation in position along the x-axis of the 3D space. The deviation in the overall pattern of movement is dependent on the accuracy and consistency of the solutions obtained by algorithm 7. The deviations are not desirable, because the correct and consistent representation of the interactive narrative space is one of the main goals of the CVTNG

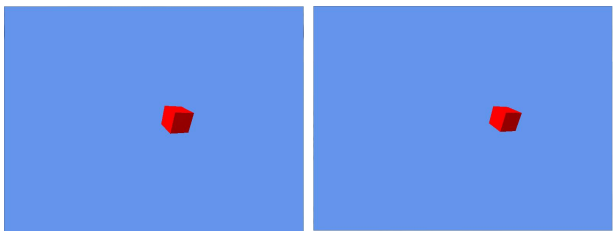

Figure 6.31: The visual output produced when interpreting "The red cube slowly moves left. The red cube quickly moves right." when interpreted in the "Visual" context hierarchy, at times $t = 1.5$ and $t = 3.0$

system (refer to section 5.2). The deviations can be addressed by utilising previously determined approximate measurement values when approximate measurement values are determined by the fuzzy procedure for assigning measurement values to attribute variables. The previously determined approximate measurement values can be compared to the approximate measurement value determined by the current iteration of algorithm 7, and adjusted to ensure consistency with previously obtained results. The near-optimal values obtained for the sample sentences, "The red cube slowly moves left. The red cube quickly moves right.", do not make the deviations visible. The introduction of large variable counts or large variable domains will, however, make any deviations more noticeable.

6.3.5 Chapter Summary

This chapter has set out and described the empirical results obtained when the CVTNG system was applied to a series of natural language sentences that describe objects in static, periodic, and dynamic relationships. The words of the natural language sentences were modelled in terms of computational verb models within the CVTNG system. The computational verb models were processed by the CVTNG system to obtain measurement values. The measurement values were passed to a representational means as parameters. These parameters were used to transform a representational means in an interactive narrative space. The transformed representational means served to represent the natural language sentences within the interactive narrative space and they therefore completed the narrative text to interactive narrative space translation.

The contexts of interpretation, subcontexts, and partial contexts used to group and subdivide the evolving function models of natural language words that describe static spatial relationships were discussed. The implementation details of the contexts of interpretation, subcontexts, and partial contexts were discussed in relation to CVTNG architecture and graphical user interface components. The evolving function models of the words in the natural language sentence, "The red cube is on the blue cube.", were discussed with respect to theoretical models and implementation details within the CVTNG system. The CVTNG implementation details for the evolving function models of the sample sentence, "The red cube is on the blue cube.", were discussed with respect to the architecture and CVTNG graphical user interface components of the CVTNG system.

The actions of the CVTNG parsing subsystem were discussed in relation to the sample sentence, "The red cube is on the blue cube.". The formation of parsing bins from the relationships between natural language words in this sample sentence were discussed as were the attribute variable relationships formed from the construction of evolving function models for the same sample sentence. The evolving function models are formed from the evolving function models specified for the words within the sample sentence, "The red cube is on the blue cube.", and the parse bin relationships formed by the parsing algorithm of the CVTNG parsing subsystem. The actions and behaviour of the CVTNG parsing subsystem discussed in relation to the sample sentence served as a reference for the discussion of the CVTNG parsing subsystem in relation to the other examples presented in the chapter.

The actions of the CVTNG subsystem for assigning measurement values to attribute variables were discussed in relation to the evolving function models for the sample sentence, "The red cube is on the blue cube.". The actions of the recursive substitution algorithm (refer to algorithm 5.6.3.1) were discussed in detail as an interation over a graph of attribute relationships with default measurement values substitutions and measurement value calculations as required. The measurement value calculations of the attribute variables in the evolving function models of the sample sentence were discussed in detail. The actions of the recursive substitution algorithm upon the evolving function models of the sample sentence served as a base reference for the accounts of the actions

of the recursive substitution algorithm on other examples in the chapter.

The interaction between the CVTNG subsystem for assigning measurement values to attribute variables and a representational system by means of a generated element class were discussed. An element class is a generated class that implements methods of the `Element` interface (refer to section 5.4.6). The `Element` interface facilitates the transformation of a representational means such as a graphical model or sound in an interactive narrative space according to measurement values obtained from evolving function models. The initialisation of a 3D graphical model that corresponds to the noun "cube" was discussed, as implemented in the `Initialize` method of the `Element` interface. The initialisation of default values to transform the graphical model (should no measurement values be assigned) was discussed, as implemented in the `InitVars` method of the `Element` interface. The transfer of measurement values to the element class was discussed, as implemented in the generated accessor methods of the element class. The accessor methods receive a measurement value as a parameter and reinitialise a transformation according to the measurement value parameter received. The application of transformations to the representational means was discussed as implemented in the `Transformation` method of the `Element` interface. The transformations applied were implemented as specifications of the `Transformation` interface in a separate assembly. The display of the transformed 3D model was discussed, as implemented in the `Invoke` method of the `Element` interface within the generated element class. The display of the transformed 3D model completes the translation of narrative text to computational verb models, and to depictions in an interactive narrative space. The discussion of the interactions between the CVTNG subsystem for assigning measurement values to attribute variables and the representational system served as a reference for discussions with respect to representation for the other examples presented in the chapter.

The actions of the CVTNG system were discussed when the sample sentence, "The dark red cube is far above the light blue cube.", is processed by the CVTNG system. This sample sentence contains adverbs that transform the evolving function models of associated adjectives and verbs. The contexts of interpretation, subcontexts, and partial contexts that group and subdivide the evolving function models of the words in the sample sentence were stated. The evolving function models of the sample sentence were

stated, as implemented within the specified contexts, subcontexts, and partial contexts. The effect of the adverbs introduced in the sample sentence were discussed in relation to the actions of the CVTNG parsing subsystem. The parse bin relationships and attribute variable relationships produced by the CVTNG parsing subsystem were presented. The amplitude value transformations performed on the evolving function models of the sample sentence were discussed. The amplitude value transformations correspond to the adverbs in the sample sentence. The amplitude values correspond to the evolving function models of the adjectives "red" and "blue" and the evolving function model of the verb phrase "is above". The representation of the sample sentence within a 3D visual space was briefly discussed and illustrated.

The sample sentences, "The red cube is to the left of the blue cube. The red cube is to the right of the blue cube.", were introduced to examine the actions of the CVTNG system when natural language that contain conflicting statements with regard to the static relationships between objects is processed. The required contexts of interpretation, subcontexts, and partial contexts were introduced and evolving function models were specified within the context, subcontext, and partial context groupings. The actions of the CVTNG parsing subsystem were discussed and the resulting parse bins, attribute variable relationships and evolving function equations were presented. The evolving function models clearly showed conflicting measurement value assignments to the attribute variable, "XPosition(Red)", from two distinct evolving functions. The evolving functions related to the verb phrases "is to the left of" and "is to the right of" respectively. The evolving functions that corresponded to the conflicting measurement value assignments to the "XPosition(Red)" attribute variable were formulated as fuzzy constraints. The fuzzy procedure for assigning approximate measurement values to attribute variables was discussed in terms of the genetic algorithm for the resolution of fuzzy constraints. The parameters of the genetic algorithm for the resolution of fuzzy constraints were experimentally determined, stated, and discussed in relation to the exploration and exploitation behaviour of algorithm 7. The optimal fitness values attained for solutions determined by algorithm 7 were stated and briefly discussed. The approximate measurement values that correspond to the optimal fitness values for algorithm 7 were interfaced to a representational element in the form of a 3D model, and the visual

output was illustrated.

The sample sentence, "The green rotating cube orbits the yellow cube.", was introduced to examine the actions of the CVTNG system when natural language sentences which describe relationships between objects in terms of actions that repeat over time are processed. Additional contexts, subcontexts, and partial contexts were introduced: these allowed evolving function models to be specified for changes in the position and rotation of objects along the x-axis, y-axis, and z-axis of a 3D space.

The evolving function models of the words in the sample sentence, "The green rotating cube orbits the yellow cube.", were stated. The actions of the CVTNG parsing subsystem were similar to those of previous examples and the parse bins and attribute variable relationships produced were presented. The evolving function models produced by the CVTNG parsing subsystem for the sample sentence differed from those of previous examples because the evolving function models were parameterised by time. The actions of the CVTNG subsystem for assigning measurement values to attribute variables were discussed to describe the calculation of measurement values at a specific point in time. The visual output produced for this sample sentence were shown for transformations according to measurement values determined at specific time values. Different measurement values that are calculated over a time interval result in different transformations of an associated representational means. The transformations applied to a 3D model of a cube over a period time, yields an animation of the periodic actions described within the sample sentence.

The sample sentence, "The green rotating cube orbits the yellow cube.", was then altered to the sample sentence, "The green quickly rotating cube slowly orbits the yellow cube.", in order to examine the effect of adverbs of period, "quickly" and "slowly", on the evolving function models of the adjective, "rotating", and the verb, "orbits". The transformations associated with the computational verb models of the adverbs "quickly" and "slowly" were discussed in terms of their computational verb models, their effect on the actions of the CVTNG parsing subsystem, and their effect on the actions of the CVTNG subsystem for assigning measurement values to attribute variables. The measurement values determined from the transformed evolving functions associated with the sample sentence were used to transform a representational element in the form of

a 3D model of a cube. The visual output of a 3D cube transformed by measurement values determined from the evolving function models of the new sample sentence was contrasted with the visual output produced for the sample sentence, "The green quickly rotating cube slowly orbits the yellow cube.".

Yet other sample sentences, "The red cube slowly rotates clockwise. The red cube quickly rotates anticlockwise.", were introduced to examine the behaviour of the CVTNG system when natural language sentences that describe inconsistent periodic actions are processed. The necessary contexts of interpretation, subcontexts, and partial contexts were stated. The implementation of computational verb models for the sample sentences, "The red cube slowly rotates clockwise. The red cube quickly rotates anticlockwise.", were stated. The actions of the CVTNG parsing subsystem and the recursive substitution algorithm were briefly discussed with reference to previous examples. The evolving function equations produced for these sample sentences included conflicting measurement values assignments to the attribute variable, "YRotation(Red)". The conflicting measurement value assignments corresponded to two instances of the verb "rotates" that were transformed by the adverbs "slowly" and "clockwise" and "quickly" and "anticlockwise" respectively. The evolving function models that correspond to the conflicting measurement value assignments of the attribute variable, "YRotation(Red)", were formulated as fuzzy constraints. The fuzzy constraints were processed by algorithm 7. The parameters for algorithm 7 as applied to these sample sentences were experimentally determined and stated. The fitness values of solutions obtained by algorithm 7 were stated and discussed. A 3D model of a cube was transformed according to the approximate measurement values determined by algorithm 7 for the sample sentences and the visual output for specific time values was provided.

The actions of the CVTNG system were examined when applied to a series of example natural language sentences that describe changes in the state of an object over time. The sample sentences, "The red cube is to the left of the blue cube. The red cube moves onto the blue cube.", were introduced and discussed with respect to evolving function models, the actions of the CVTNG parsing subsystem, and the CVTNG subsystem for assigning measurement values to attribute variables. Additional contexts of interpretation, subcontexts, and partial contexts were introduced to allow evolving

function models to be specified for a point of reference for a movement in a 3D space. The evolving function models associated with a point of reference were subsequently discussed in depth in relation to the actions of the CVTNG subsystem for assigning measurement values to attribute variables. The evolving function models specified for the sample sentences above included time interval information. The effect of the time interval information on evolving function models was discussed for the CVTNG sequencing algorithm, the CVTNG parsing subsystem, and the CVTNG subsystem for assigning measurement values to attribute variables.

The evolving function models of the sample sentences contain cyclical attribute variable references and therefore some measurement values for attribute variables were determined by the Gaussian elimination algorithm. The measurement values determined from the active evolving function models at a point in time were once again transferred to a representational system. The visual output produced for the sample sentences, "The red cube is to the left of the blue cube. The red cube moves onto the blue cube.", was illustrated.

The sample sentences, "The red cube is to the left of the blue cube. The red cube moves onto the blue cube.", were then modified to the sample sentences, "The red cube is far to the left of the blue cube. The red cube quickly moves onto the blue cube.", in order to examine the effect of adverbs of period and amplitude on the actions of the CVTNG system. The actions of the CVTNG parsing subsystem and the CVTNG subsystem for assigning measurement values to attribute variables were discussed in relation to the sample sentences, "The red cube is far to the left of the blue cube. The red cube quickly moves onto the blue cube.". The effect of the adverb of period, "quickly", on the time interval information of evolving functions with infinite and terminate interval sizes was discussed. The visual output produced from the measurement values calculated by the CVTNG system for these latest sample sentences was presented.

The actions of the CVTNG system were examined when natural language sentences that describe inconsistent changes in the state of an object are presented as input. Another set of sample sentences, "The red cube slowly moves left. The red cube quickly moves right.", was discussed in relation to its CVTNG evolving function models, the CVTNG parsing subsystem, and the CVTNG subsystem for assigning measurement

values to attribute variables. The evolving function models contained duplicate measurement value assignments to the attribute variable, "XPosition(Red)". The evolving functions that assigned measurement values to the attribute variable, "XPosition(Red)", were formulated as fuzzy constraints. The parameters and optimised fitness values for solutions attained by algorithm 7 were discussed in relation to the fuzzy constraints related to the attribute variable "XPosition(Red)". The visual output produced from the approximate measurement values calculated by the CVTNG system for the sample sentences, "The red cube slowly moves left. The red cube quickly moves right.", was presented.

In this chapter, the empirical results obtained when the CVTNG system was tested against a series of natural language sentences that describe relationships between objects, have been stated and discussed. The theoretical models, implementation, and empirical verification of the CVTNG system have been stated in their entirety. Chapter 7 summarises the content of this thesis with regard to the aspects presented, discusses the findings, and proposes future work based on the findings.

Chapter 7

Summary, conclusions, and future work

This chapter comprises a summary of the work presented in chapters 1 to 6, a number of conclusions reached with regard to the theoretical principles and architecture underlying interactive narrative generation using computational verb theory (CVT), and the empirical results drawn from the present study. It goes on to outline the possibilities for future work based on the conclusions reached. Section 7.1 consists of a brief summary of the contents of chapters 1 to 6 followed by a set of conclusions drawn from aspects of this work. Section 7.2 contains a number of topics for future work that the present writer considers necessary for improving and extending the theoretical concepts, system architecture, and application examples presented in this work.

7.1 Content, summary, and conclusions

Chapter 1 introduced the fields of interest and main concepts presented in this work. The fields of interactive narrative, computational verb theory, computational linguistics, and constraint satisfaction problems (CSPs) were introduced and the background to each field was briefly discussed. At the outset the concept of a system for the generation of interactive narrative from narrative text by means of computational verb theory as covered within the ambit of this work was presented. The novel contributions and applications

described in this work were stated in relation to the fields of interactive narrative, computational linguistics, computational verb theory, and constraint satisfaction problems. Chapter 1 concluded by providing an outline of the remainder of the thesis.

In chapter 2 research trends in the related field of interactive narrative were discussed, as was the application of fuzzy set theory to interactive narrative. In addition, the paradigm of a system that generates interactive narrative from narrative text was constrasted with the approaches of existing interactive narrative systems in order to highlight the advantages of a system able to generate interactive narrative from narrative text.

The discussion of interactive narrative in chapter 2 in particular showed that the interactive narrative generation system presented is both a novel approach towards the creation of interactive narrative and a complementary addition to existing interactive narrative systems. Narrative text was shown to be a natural and concise representation format for narrative. The generation of interactive narrative also makes it possible for multiple interpretations to be associated with the same narrative text. A system that generates interactive narrative from natural language text was put forward as being an accessible medium for interactive narrative generation, because a larger percentage of the population is able to phrase narrative in terms of natural language sentences than re-create narrative using complex modelling tools and programs.

Chapter 3 went on to state the fuzzy set theory, CVT, and constraint satisfaction problem (CSP) definitions that are applied within this work. The background and development of CVT was provided together with an outline of some existing applications of CVT. The definitions provided in chapter 3 served as an important reference for the chapters that followed.

Fuzzy set theory and the development of the linguistic variable provide a means whereby natural language nouns, adjectives, and adverbs that modify adjectives can be quantified and computed. CVT extends the possibilities of fuzzy set theory by offering a way for natural language verbs, and the adverbs that apply to them, to be quantified and computed. CVT models natural language verbs in terms of dynamic systems, and vice versa: if a dynamic system model can be constructed for a natural language verb, a CVT model can be constructed for the natural language verb. Following from this, a dynamic

system model can be constructed for any observable feature that can be measured over a period of time. CVT is therefore a very powerful tool for enabling many — and possibly all — natural language verbs to be modelled. Constraint satisfaction problems (CSPs) and their relaxation to fuzzy constraint satisfaction problems (FCSPs) provide a framework with which the solution of variables within multiple concurrent computational verb models can be framed and resolved.

Chapter 4 presented in detail the CVT and CSP models constructed and applied within this work. A computational verb model was described that models computational verbs in terms of generic evolving function models. The unique instances of evolving function models are constructed from a template evolving function model through choosing generic functions of time, amplitude values, and scaling values. The evolving function models were made specific to contexts of interpretation, and were further subdivided in terms of function terms and amplitude values within subcontexts and partial contexts respectively. The computational verb models of nouns, adjectives, verbs, and adverbs were presented and the role of each word type within the evolving function models described in this work was stated. In addition, the variable resolution procedures that can be applied to determine measurement values for attribute variables within the context of this work were discussed.

A crisp procedure for assigning exact measurement values to attribute variables was also discussed. The crisp variable resolution procedure was shown to be unable to provide exact crisp measurement value assignments in the face of conflicting natural language statements. As an alternative solution, a fuzzy procedure for assigning approximate measurement values to attribute variables was introduced and discussed. A background study of CSP and FCSP principles and resolution schemes was presented, followed by a discussion in which the evolving function models of this work were framed in terms of an FCSP. A genetic algorithm for resolving fuzzy constraints was chosen as the resolution method for the FCSP formulation of the evolving function models presented in this work.

These generic evolving function models are able to model basic computational verbs that are either defined over a single time interval or repeat over a single time interval. The majority of computational verbs can be constructed from basic computational verbs by applying transformations and the verb extension principle [106]. This principle combines

known computational verb models to form a computational verb model for computational verbs that are unknown [111]. A complex dynamic system related to a complex verb such as "dream" or "think" can be segmented into components that are modelled in terms of basic computational verbs. The generic evolving function models particular to this work can be extended to enable multiple evolving functions that relate to multiple time intervals to be specified. The multiple evolving functions and multiple time periods all relate to the same computational verb in a context of interpretation. Furthermore, if the proper evolving function is selected according to the current time value, the generic function models of this work are able to model the majority of complex computational verbs. Any extensions of evolving function models over multiple time periods falls beyond the scope of this work.

The crisp procedure for assigning measurement values to attribute variables presented is able to determine the measurement values of attribute variables that are present in evolving function equations, if the evolving function models do not assign conflicting measurement values to the same attribute variable. The inability of the crisp procedure for assigning measurement values to attribute variables to assign precise measurement values to attribute variables if conflicting measurement value assignments exist, necessitated a procedure that is able to assign approximate measurement values to attribute variables. The fuzzy procedure for assigning approximate measurement values to attribute variables presented is able to assign approximate measurement values to attribute variables present in evolving functions. The fuzzy procedure for the assignment of approximate measurement values to attribute variables formulates evaluated evolving function equations that contain unknown attribute variables as an FCSP. The genetic algorithm presented to solve the resulting FCSPs was able to provide near-optimal approximated values for the attribute variables present in the evolving function models of the sample sentences presented in this work. Appendix B presents further examples that test the scalability of the crisp and fuzzy procedures for assigning measurement values to attribute variables.

Chapter 5 presented the architecture and algorithms implemented in the computational verb theory interactive narrative generation (CVTNG) system for generating interactive narrative space from natural language sentences. The challenges presented

by the generation of interactive narrative from natural language sentences were stated in terms of the inconsistency, implicitness, and ambiguity of natural language. The overall system goals of the CVTNG system were presented as the encapsulation of model complexity, the correct and consistent representation of the narrative space, the separation of concerns for users that interact with the system in different ways, and the re-use of artifacts produced by the system in other interactive narrative spaces. The architectural features used to represent contexts of interpretation, subcontexts, and partial contexts within the CVTNG system were also described. In addition, the architectural features used to represent the evolving function models for words in the CVTNG system were described. The algorithms and class structure of the parsing subsystem of the CVTNG system were described in conjunction with the sequencing algorithm present within the CVTNG system. The crisp and fuzzy procedures for assigning measurement values to attribute variables were described in terms of the class structures and algorithms related to the respective procedures.

The architectural features presented for the CVTNG system combine to create a system that generates interactive narrative spaces from natural language sentences. The architectural components implemented realise the main system goal of narrative space generation whilst adhering to the system principles of encapsulation, separation of concerns, consistent representation, and the re-use of artifacts. The architectural features related to contexts of interpretation and word models allow the evolving function models presented in chapter 4 to be defined and subdivided in terms of contexts of interpretation, subcontexts, and partial contexts. The architectural features of the word models allow nouns, adjectives, verbs, and adverbs to be modelled in terms of computational verb models that are specific to a context of interpretation and a time period. The parsing algorithm presented translates natural language sentences into verb sentences by forming groupings of similar word types and determining the subject and object relationships between the word groupings. The sequencing algorithm presented allows the time periods associated with computational verb models to be sequenced so that a sequence of actions described by natural language sentences in narrative text can be modelled in terms of computational verb models with sequential validity periods. The architectural features described for the crisp procedure for assigning measurement values to attribute variables

enable the CVTNG system to calculate exact measurement values for attribute variables. If one or more attribute variables are assigned conflicting measurement values by a set of evolving function models, the architectural features of the CVTNG system that relate to the fuzzy procedure for assigning measurement values to attribute variables allow the CVTNG system to calculate approximate measurement values for the attribute variables in question.

Chapter 6 tested the CVTNG system and the evolving function models presented in this work against a series of natural language sentences that describe objects in static, periodic, and dynamic relationships. The static, periodic, and dynamic object relationships were modelled in terms of static (refer to definition 3.15), centre (refer to definition 3.18), and node (refer to definition 3.16) computational verbs respectively. The evolving function models for the computational verbs were presented for a context that interprets the sample sentence visually. The actions of the parsing subsystem and the actions of the crisp measurement value assignment procedure were described in relation to the evolving function models presented for the static, periodic, and dynamic examples. The static, periodic, and dynamic examples were extended in terms of adverbs of amplitude and period to examine the effect of the adverbs on the corresponding evolving function models, the parsing subystem, and procedures for assigning measurement values to attribute variables. Sentences that state conflicting static, periodic, and dynamic relationships were introduced to examine the calculation of approximate measurement values from the associated evolving function models. The parameters and performance of the genetic algorithm for resolving the FCSP used in the fuzzy procedure for assigning approximate measurement values to attribute variables was discussed. Three-dimensional models illustrated the interactive narrative space generated for all of the sample sentences presented. The actions of the CVTNG sequencing algorithm (refer to algorithm 2) were discussed in relation to the dynamic examples presented.

The empirical results obtained by testing the CVTNG system against natural language sentences that describe static, periodic, and dynamic relationships illustrate that the evolving function models presented in this work are able to model natural language words in terms of computational verbs. The actions of the parsing subsystem as illustrated prove that the CVTNG system is able to dynamically construct evolving function

models from predefined evolving function and context models of natural language words. The actions of the CVTNG crisp and fuzzy procedures for assigning measurement values to attribute variables prove that the CVTNG system is able to determine precise and approximate measurement values for attribute variables present in constructed evolving function models. The 3D models produced as output prove that the CVTNG system is able to generate interactive narrative spaces from natural language sentences.

This work presented generic computational verb models for nouns, adjectives,verbs, and adverbs. The supported word types were modelled in terms of evolving functions or as modifiers of evolving function models. A CVTNG system was presented that dynamically constructs evolving function models for natural language sentences provided as input. The evolving function models are constructed from stored evolving function definitions for words in specified contexts of interpretation. The evolving function models for a specific context are dynamically scaled in terms of amplitude and period according to the natural language sentences provided as input. The CVTNG system dynamically calculates the validity periods for the evolving functions and calculates measurement values from the valid evolving function models at a particular point in time. If inconsistencies exist in the measurement value assignments within the evaluated evolving function models, the CVTNG system is capable of calculating approximate measurement values results. The CVTNG system generates interface (element) classes according to the word definitions specified within the CVTNG system. The element classes allow measurement values calculated within the CVTNG system to be interfaced to an external representation system, as may be found in an interactive narrative space. The element classes transform the representational elements by transformations that are parameterised by the calculated measurement values. The transformed representational elements form the interactive narrative space generated from natural language text when invoked.

This work concludes by stating that the CVTNG system is a robust and extendable framework for generating interactive narrative spaces from natural language sentences by means of computational verb models. The section that follows examines key aspects of this work and proposes extensions and enchancements that will allow the evolving function models, systems, and algorithms presented in this work to be applied to a greater variety of natural language sentences, and to generate more complex interactive

narrative spaces.

7.2 Future work

This section examines aspects of the work presented and identifies research topics for future work. The research topics identified will expand the models, systems, and algorithms presented in this work to incorporate new types of natural language words and sentences, generate more complex interactive narrative spaces, and perform narrative text to interactive narrative space translation more efficiently and accurately. Section 7.2.1 examines the evolving function models presented in this work. It also proposes extensions that will allow complex computational verbs to be modelled within the CVTNG system. Section 7.2.2 examines the parsing subsystem of the CVTNG system and proposes research in the field of computational linguistics that will allow the CVTNG system to process more complex natural language sentences into verb sentences and evolving function models. Section 7.2.3 discusses the procedures for assigning measurement values to attribute variables presented in this work and proposes research that will compare the algorithms presented in this work against other constraint satisfaction and optimisation algorithms. Research that examines the formal aspects of a genetic algorithm applied to an FCSP as employed in this work is also proposed. The interaction of the CVTNG system with representations used typically in interactive narrative systems is examined in section 7.2.4 to highlight future applications of the CVTNG system in the fields of interactive narrative and education.

7.2.1 Word models

This work models natural language adjectives and verbs in terms of basic computational verb models. The basic computational verb models are expressed in terms of an evolving function that is specific to a context of interpretation. A time period is associated with the evolving function and the context of interpretation to form a verb element (refer to definition 3.22).

The verb elements formed within this work associate a single evolving function with a single time period within a context of interpretation. The evolving function models

adequately model basic verbs and adjectives as illustrated in chapter 6 and further illustrated in Appendix B. If the verb elements of this work are extended to allow multiple evolving functions with multiple validity periods to associate with a single computational verb in a context of interpretation, the extension principle of computational verbs states that more complex computational verb models can be formed from the recombination of the simpler models presented in this work [111]. If complex computational verb models are implemented within the CVTNG system, and the CVTNG is able to select the evolving function that corresponds to a time value from the list of evolving function models that segment the dynamics of a complex computational verb, the CVTNG system will be capable of modelling complex natural language verbs such as "think", "understand", and "dream".

The evolving function models presented in this work allow an amplitude value to be calculated as the sum of a collection of partial amplitude values. The partial amplitude value concept can be generalised in further work to specify a generic function, $\alpha_i = f(\alpha_{i_1}, \ldots, \alpha_{i_j})$, that allows an amplitude value, α_i, to be calculated from any mathematical combination of partial amplitude values. The generalisation is not required within the scope of this work and has not been implemented in the CVTNG system prototype. The generalisation would allow more complex amplitude scaling calculations to occur within the CVTNG system and therefore to more accurately scale the amplitude values of evolving function models for complex verbs and adjectives.

The evolving function models of this work are defined within a crisp space. If exact measurement values cannot be calculated for attribute variables, the evaluated evolving function equations that contain the attribute variables are treated as fuzzy constraints to allow approximate measurement values to be calculated for the attribute variables involved. The evolving function models of this work can be extended to a fuzzy space by associating a membership function, μ, with an evolving function, $\mathcal{E}_{\mathcal{V}}$. The membership function, μ, calculates the membership value of a measurement value to the fuzzy set, A, associated with μ. The measurement value is calculated from the evolving function, $\mathcal{E}_{\mathcal{V}}$. Extending the evolving function models in this work to a fuzzy space would allow adjectives and verbs to be modelled that have a measure of uncertainty such as "around" or "more or less". The fuzzy procedure for determining approximate measurement values

for attribute variables can be adjusted to accomodate evolving functions in a fuzzy space by using the membership function, μ, as the membership function of a fuzzy constraint. The fuzzy constraint in question corresponds to the evaluated evolving function equation of the evolving function, $\mathcal{E}_{\mathcal{V}}$. The extension of the fuzzy procedure for determining approximate measurement values for attribute variables would allow the CVTNG system to model and process natural language sentences which contain adjectives and verbs that have a measure of uncertainty.

7.2.2 Parsing

This work presented a parsing algorithm that forms verb sentences by grouping similar word types and determining the subject and object relationships between the word groupings. The parsing algorithm was presented in terms of a parse bin framework (refer to section 5.5.3) that abstracted the formation of word groupings and the determination of relationships among word groupings. The parse bin architecture allows the parsing operations of the CVTNG system to be extended easily in terms of new computational linguistics algorithms. New parsing operations and algorithms would allow the CVTNG system to process more complex natural language word relationships and sentences.

The current CVTNG parsing algorithm processes atomic verb sentences (refer to definition 3.20). Additional computational linguistics algorithms can be incorporated within the parsing behaviour of the CVTNG system as part of future work. The new algorithms can be implemented by means of the parse bin interface to allow more complex natural language sentences that contain conjunctions and other connectives to be processed into word groupings with subject and object relations.

The CVTNG parsing algorithm processes subject and object relationships individually. If multiple subject nouns are specified, as for the sample sentence, "The red cube and the blue cube are on the yellow cube.", the CVTNG parsing algorithm interprets the sentence as "The red cube is on the yellow cube. The blue cube is on the yellow cube." and a conflict occurs. The CVTNG parsing algorithm can process the sample sentence, "The red cube and the blue cube are on the yellow cube.", correctly if the noun phrase, "red cube and blue cube", is implemented as a single noun within the word models specified for the sample sentence. The relationships between the red and the blue

cube would be handled implicitly by the narrative depiction associated with the noun phrase, "red cube and blue cube". Future work on the CVTNG parsing subsystem may incorporate work such as that of Wiebe [97] to identify subjects within natural language sentences and dynamically construct collective nouns and noun groupings with adapted evolving function models and narrative depictions.

Improved subject determination algorithms will also improve the ability of the CVTNG parsing subsystem to detect unique occurrences of nouns. The examples presented within the main body of this work determined unique instances of nouns according to their associated adjectives. Appendix B provides examples of improved subject determination procedures that also factor in the adverbs that describe the adjectives associated with a noun when determining a unique instance. Subject determination algorithms from the field of computational linguistics will improve this aspect of the CVTNG system even further.

The current CVTNG parsing algorithm considers all contexts of interpretation active when the evolving function models for narrative text provided as input are formed. The parsing algorithm may be extended by point of view tracking algorithms such as that of Wiebe *et al* [98] to determine the point of view for a natural language sentence and select the applicable context (first, second, or third person) accordingly.

A future development of the CVTNG parsing subsystem that cannot be incorporated via the parse bin interface alone is the ability to alter the parsing process according to measurement values determined from the CVTNG subsystem for assigning measurement values to attribute variables. If the behaviour of the CVTNG parsing algorithm can be altered according to measurement values, conditional statements such as, "If the cube is on the blue cube, the cube is red.", and questions such as, "The cube is on the blue cube. What is the colour of the cube?", can be processed by the CVTNG system.

The final proposal for future work with regard to the expansion of the CVTNG parsing subsystem is the enhancement of the CVTNG sequencing algorithm. The current CVTNG sequencing algorithm does not allow terminate actions to occur simultaneously. Further research in the field of temporal reasoning needs to be performed to allow complex event sequencing to be performed for narrative text provided as input to the CVTNG system. Event sequencing algorithms such as that of Allen [4] can be adopted as the

sequencing algorithm of the CVTNG system, and the sequencing of time intervals can be handled as a CSP. A complex sequencing algorithm would allow the CVTNG system to handle concurrent actions as indicated by the example of the natural language word "while".

Another development of the CVTNG sequencing behaviour would be to allow adverbs of period to shift the time interval associated with an evolving function according to the tense of a natural language sentence and other indicators of time.

7.2.3 Measurement value assignment

The crisp and fuzzy procedures for the asssignment of measurement values to attribute variables allow the CVTNG system to determine exact measurement values for attribute variables and approximate measurement values if exact measurement values cannot be determined. The FCSP formulation of evaluated evolving function equations (EEFEs) that contain attribute variables whose measurement values cannot be determined allows the CVTNG system to attach priorities to the EEFEs [27]. If the priority values are factored into the determination of approximate measurement values, and certain EEFEs can be deprioritised, the CVTNG system would have the ability to "ignore" conflicting statements. The CVTNG subsystem for assigning measurement values to attribute variables would be able to calculate approximate measurement values as if either side of a conflicting statement were true. The sample sentences, "The red cube is to the left of the blue cube. The red cube is to the right of the blue cube.", could be processed as "The red cube is to the left of the blue cube". if the evolving function(s) associated with the second sentence were deprioritised.

A real-coded genetic algorithm is implemented in the CVTNG system to solve the FCSPs associated with EEFEs that contain attribute variables whose measurement values cannot be determined (refer to algorithm 7). A comparison of algorithm 7 to other optimisation algorithms such as particle swarm optimisation and stochastic search within the context of FCSP solution is required to determine the optimal strategy for assigning approximate measurement values to attribute variables in the CVTNG system.

The parameters of algorithm 7 were experimentally determined for optimal results within the scope of this work. A formal study of the parameters of the optimisation al-

gorithm chosen for the resolution of FCSPs with the CVTNG could form part of future work. If the exploration and exploitation behaviour of the chosen optimisation algorithm is predictable for all parameter settings, parameter values may be dynamically chosen to obtain optimal approximated measurement values at speeds that allow real-time interfacing to representational systems such as a graphics engine. Chapter 6 examined basic examples of conflicting statements and Appendix B provides more complex examples of conflicting statements. A full study of the scalability of algorithm 7 to FCSPs of high dimensionality should form part of future work.

7.2.4 Application of the CVTNG system

Section 5.4.6 presented element classes as a means whereby measurement values calculated by the CVTNG system can be transferred to external representations that are typically used in interactive narrative such as graphics, sound, and artificial intelligence (AI) agents. Future work will associate the CVTNG system with a standalone interactive narrative system to generate interactive narrative spaces that can be used by the standalone interactive narrative system. The application would allow interactive narrative spaces to be constructed for all facets of the standalone interactive narrative system by simply entering narrative text.

Another application of the CVTNG system could be realised if the input to the CVTNG system were to be simplified from the input of narrative text to the selection of available words through a simple interface such as a touch screen. The application would allow children to select word combinations and have the CVTNG render a visual depiction of the word combination.

The CVTNG system can achieve this by calculating measurement values for preset contexts of interpretation and interfacing the calculated measurement values to a rendering system. The rendering system subsequently draws the transformed images associated with the word combination. If the definition of words were furthermore simplified to the drawing of images for nouns and the specification of adjectives and verbs in terms of arrows rather than mathematical formulas, the system would allow children to construct their own words. The proposed system could be tested on children with varying levels of reading and spatial reasoning skills to determine whether the sentence and word

construction activities facilitate the development of reading and spatial reasoning skills.

Bibliography

[1] `http://www.w3.org/XML/`. Accessed on: June 1 2009.

[2] *The American Heritage Dictionary of the English Language, Fourth edition.* Houghton Mifflin Company, Boston, MA, USA, 2004.

[3] *The American Heritage Science Dictionary, Fourth edition.* Houghton Mifflin Company, Boston, MA, USA, 2004.

[4] J.F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, November 1983.

[5] R.M. Axelrod. *Structure of decision: the cognitive maps of political elites.* Princeton University Press, Princeton, NJ, USA, 1976.

[6] R. Aylett, S. Louchart, J. Dias, A. Paiva, M. Vala, S. Woods, and L. Hall. Unscripted narrative for affectively driven characters. *Computer Graphics and Applications, IEEE*, 26(3):42–52, May–June 2006.

[7] F. Bacchus and P. van Beek. On the conversion between non-binary constraint satisfaction problems. In *AAAI '98/IAAI '98: Proceedings of the fifteenth national/tenth conference on artificial intelligence/innovative applications of artificial intelligence, 1998*, pages 311–318, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence.

[8] H. Bandemer and W. Nather. *Fuzzy Data Analysis.* Kluwer Academic Publishing, Dordrecht, The Netherlands, 1992.

[9] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint satisfaction and optimization. *Journal of the ACM*, 44(2):201–236, March 1997.

[10] J. Bowen, R. Lai, and D. Bahler. Fuzzy semantics and fuzzy constraint networks. In *Proceedings of the IEEE international conference on fuzzy systems, 1992*, pages 1009–1016, March 1992.

[11] S.C. Brailsford, C.N. Potts, and B.M. Smith. Constraint satisfaction problems: Algorithms and applications. Technical Report 3, University of Southampton - Department of Accounting and Management Science, Essex, UK, December 1999.

[12] K. Britz. Satisfiability in semiring constraint satisfaction problems. In *SAICSIT '03: Proceedings of the 2003 annual research conference of the South African institute of computer scientists and information technologists on enablement through technology*, pages 173–177, Republic of South Africa, 2003. South African Institute for Computer Scientists and Information Technologists.

[13] K.M. Brooks. Programming narrative. *Proceedings of the 1997 IEEE symposium on visual languages, 1997*, pages 380–386, September 1997.

[14] Y. Cai, C. Miao, A. Tan, and Z. Shen. Fuzzy cognitive goal net for interactive storytelling plot design. In *ACE '06: Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology*, volume 266, New York, NY, USA, 2006. ACM.

[15] M. Cavazza, F. Charles, and S.J. Mead. Interacting with virtual characters in interactive storytelling. In *AAMAS '02: Proceedings of the first international joint conference on autonomous agents and multiagent systems*, pages 318–325, New York, NY, USA, 2002. ACM.

[16] M. Cavazza, F. Charles, S.J. Mead, O. Martin, X. Marichal, and A. Nandi. Multimodal acting in mixed reality interactive storytelling. *Multimedia, IEEE*, 11(3):30–39, July–September 2004.

[17] M. Cebrián and I. Dotú. Grasp – evolution for constraint satisfaction problems. In *GECCO '06: Proceedings of the 8th annual conference on genetic and evolutionary computation, 2006*, pages 531–538, New York, NY, USA, 2006. ACM.

[18] M.B. Clowes. On seeing things. *Artificial Intelligence*, 2:79–116, 1971.

[19] B.G.W. Craenen and A. E. Eiben. Stepwise adaption of weights with refinement and decay on constraint satisfaction problems. In *GECCO '01: Proceedings of the Genetic and evolutionary computation conference, 2001*, pages 291–298, July 2001.

[20] B.G.W. Craenen, A.E. Eiben, and J.I. van Hemert. Comparing evolutionary algorithms on binary constraint satisfaction problems. *IEEE Transactions on Evolutionary Computation*, 7(5):424–444, October 2003.

[21] M. Davis, N. Otero, K. Dautenhahn, C.L. Nehaniv, and S.D. Powell. Creating a software to promote understanding about narrative in children with autism: Reflecting on the design of feedback and opportunities to reason. In *ICDL 2007: Proceedings of the IEEE 6th international conference on Development and learning, 2007*, pages 64–69, July 2007.

[22] J.A. Dickerson and B. Kosko. Virtual worlds as fuzzy cognitive maps. In *Proceedings of the 1993 IEEE virtual reality annual international symposium, 1993*, pages 471–477, September 1993.

[23] G. Dozier, J. Bowen, and D. Bahler. Solving small and large scale constraint satisfaction problems using a heuristic-based microgenetic algorithm. In *Proceedings of the first IEEE conference on evolutionary computation, IEEE world congress on computational intelligence, 1994*, volume 1, pages 306–311, June 1994.

[24] G. Dozier, J. Bowen, and D. Bahler. Solving randomly generated constraint satisfaction problems using a micro-evolutionary hybrid that evolves a population of hill-climbers. *Proceedings of the IEEE international conference on evolutionary computation, 1995*, 2(29):614–619, November–December 1995.

[25] D. Driankov, H. Hellendoorn, and M. Reinfrank. *An introduction to fuzzy control*. Springer-Verlag New York, Inc., New York, NY, USA, 1993.

[26] D. Dubois, H. Fargier, and H. Prade. *Propagation de satisfaction de constraintes flexibles*. Kluwer Academic Publishing, Dordrecht, The Netherlands, 1993.

[27] D. Dubois, J. Lang, and H. Prade. Possibilistic logic. *Handbook of logic in artificial intelligence and logic programming: nonmonotonic reasoning and uncertain reasoning*, 3:439–513, 1994.

[28] R. C. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micromachine and human science, 1995*, pages 39–43, October 1995.

[29] A.E. Eiben, P.-E. Raue, and Z. Ruttkay. Solving constraint satisfaction problems using genetic algorithms. In *Proceedings of the first IEEE Conference on evolutionary computation, IEEE world congress on computational intelligence, 1994*, volume 2, pages 542–547, June 1994.

[30] A.E. Eiben, J.I. van Hemert, E. Marchiori, and A. G. Steenbeek. Solving binary constraint satisfaction problems using evolutionary algorithms with an adaptive fitness function. In *PPSN V: Proceedings of the 5th international conference on Parallel problem solving from nature*, pages 201–210, London, UK, 1998. Springer-Verlag.

[31] L.J. Eshelman and J. D. Schaffer. *Real-coded genetic algorithms and interval-schemata*. Morgan Kaufmann, San Mateo, CA, USA, 1993.

[32] M. Fowler and K. Scott. *UML distilled: applying the standard object modeling language*. Addison-Wesley Longman Ltd., Essex, UK, 1997.

[33] E.C. Freuder and R.J. Wallace. Partial constraint satisfaction. *Artificial Intelligence*, 58:21–70, December 1992.

[34] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

[35] M. Gen and R. Cheng. *Genetic Algorithms and Engineering Optimization*. Wiley-Interscience, Hoboken, NJ, USA, 2000.

[36] J. Good and J. Robertson. Children's narrative development through computer game authoring. In *IDC '04: Proceedings of the 2004 conference on interaction design and children*, pages 57–64, New York, NY, USA, 2004. ACM.

[37] H.W. Guesgen. A formal framework for weak constraint satisfaction based on fuzzy sets. In *Proceedings of the 1994 Second Australian and New Zealand conference on intelligent information systems, 1994*, pages 199–203, November–December 1994.

[38] H.W. Guesgen and A. Philpott. Heuristics for solving fuzzy constraint satisfaction problems. In *ANNES '95: Proceedings of the 2nd New Zealand international two-stream conference on artificial neural networks and expert systems, 1995*, pages 132–135, Washington, DC, USA, November 1995. IEEE Computer Society.

[39] S. Hamissi and M. Babes. A neural approach for solving the constraint satisfaction problem. In *Proceedings of the 2003 international conference on geometric modeling and graphics, 2003*, pages 96–103, July 2003.

[40] H. Handa, O. Katai, N. Baba, and T. Sawaragi. Solving constraint satisfaction problems by using coevolutionary genetic algorithms. In *Proceedings of the 1998 IEEE international conference on evolutionary computation, The 1998 IEEE world congress on computational intelligence, 1998*, pages 21–26, May 1998.

[41] F. Herrera and M. Lozano. Gradual distributed real-coded genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 4:43–63, 2000.

[42] F. Herrera, M. Lozano, and J.L. Verdegay. Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artificial Intelligence Review*, 12(4):265–319, 1998.

[43] P.L. Hsu and R. Lai. A generic algorithm for fuzzy constraint satisfaction problems. *Proceedings of the Joint 9th IFSA World Congress and 20th NAFIPS International Conference, 2001*, 2:768–772, July 2001.

[44] D.A. Huffman. Impossible objects as nonsense sentences. *Machine Intelligence*, 6:295–323, 1971.

[45] C.Z. Janikow and Z. Michalewicz. An experimental comparison of binary and floating point representations in genetic algorithms. In *Proceedings of the 4th International conference on genetic algorithms, 1991*, pages 31–36, San Francisco, CA, USA, 1991. Morgan Kauffman.

[46] S. Kang, H. Ko, and Y Choy. Temporal scene management for interactive virtual storytelling. *ICACT 2006: The 8th international conference on advanced communication technology, 2006.*, 2:967–972, February 2006.

[47] H. Kanoh, K. Hasegawa, M. Matsumoto, S. Nishihara, and N. Kato. Solving constraint satisfaction problems by a genetic algorithm adopting viral infection. In *IJSIS '96: Proceedings of the 1996 IEEE International joint symposia on intelligence and systems*, pages 67–73, Washington, DC, USA, November 1996. IEEE Computer Society.

[48] H. Kanoh, M. Matsumoto, and S. Nishihara. Genetic algorithms for constraint satisfaction problems. In *Proceedings of the IEEE international conference on systems, man, and cybernetics, 1995. Intelligent systems for the 21st century*, volume 1, pages 626–631, October 1995.

[49] B. Kosko. Fuzzy cognitive maps. *International Journal of Man-Machine Studies*, 24(1):65–75, 1986.

[50] B. Kosko. *Neural networks and fuzzy systems: a dynamical systems approach to machine intelligence*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1992.

[51] R. Kowalczyk. On solving fuzzy constraint satisfaction problems with genetic algorithms. In *Proceedings of the 1998 IEEE international conference on evolutionary computation, IEEE world congress on computational intelligence, 1998*, pages 758–762, May 1998.

[52] V. Kumar. Algorithms for constraint-satisfaction problems: A survey. *AI Magazine*, 13(1):32–44, 1992.

[53] B.C. Lam and H. Leung. Progressive stochastic search for solving constraint satisfaction problems. In *TAI '03: Proceedings of the 15th IEEE international conference on Tools with artificial intelligence, 2003*, pages 487–491, Washington, DC, USA, 2003. IEEE Computer Society.

[54] L. Leenen, T. Meyer, and A. Ghose. Relaxations of semiring constraint satisfaction problems. *Information Processing Letters*, 103(5):177–182, 2007.

[55] L. Leenen, T. Meyer, and A. Ghose. Relaxations of semiring constraint satisfaction problems. *Information Processing Letters*, 103(5):177–182, 2007.

[56] E. Lengyel. *Mathematics for 3D Game Programming and Computer Graphics, Second Edition*. Charles River Media, Inc., Rockland, MA, USA, 2003.

[57] J. Lugrin and M. Cavazza. AI-based world behaviour for emergent narratives. In *ACE '06: Proceedings of the 2006 ACM SIGCHI international conference on advances in computer entertainment technology*, page 25, New York, NY, USA, 2006. ACM.

[58] A.D. Malmud. Tech(xt)s [links between text and technology]. *IEEE Multimedia*, 7(4):6–9, October–December 2000.

[59] E. Marchiori. Combining constraint processing and genetic algorithms for constraint satisfaction problem. In *ICGA: Proceedings of the 7th International conference on genetic algorithms, 1997*, pages 330–337, San Francisco, CA, USA, 1997. Morgan Kauffman.

[60] E. Marchiori and A. Steenbeek. A genetic local search algorithm for random binary constraint satisfaction problems. In *SAC '00: Proceedings of the 2000 ACM symposium on applied computing, 2000*, pages 458–462, New York, NY, USA, 2000. ACM.

[61] M. Mateas. *Artificial Intelligence Today: Recent trends and developments - An Oz-Centric Review of Interactive Drama and Believable Agents*, volume 1600 of *Lecture Notes in Computer Science*. Springer Berlin, Berlin, Germany, 1999.

[62] M. Mateas and A. Stern. A behavior language for story-based believable agents. *IEEE Intelligent Systems*, 17(4):39–47, 2002.

[63] J.R. Meehan. *The metanovel: Writing Stories by Computer*. PhD thesis, Yale University, New Haven, CT, USA, 1976.

[64] Z. Michalewicz. *Genetic algorithms + data structures = evolution programs, second extended edition*. Springer-Verlag New York, Inc., New York, NY, USA, 1994.

[65] Z. Michalewicz. *Genetic algorithms + data structures = evolution programs, third edition*. Springer-Verlag, London, UK, 1996.

[66] I. Miguel and Q. Shen. Extending fcsp to support dynamically changing problems. In *FUZZ-IEEE '99: Proceedings of the 1999 IEEE international fuzzy systems conference, 1999*, volume 3, pages 1615–1620, Washington, DC, USA, 1999. IEEE Computer Society.

[67] S. Minton, A. Philips, M.D. Johnston, and P. Laird. Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, 58(1–3):161–205, 1992.

[68] R. Mitkov. *The Oxford Handbook of Computational Linguistics*. Oxford Handbooks in Linguistics. Oxford University Press, New York, NY, USA, March 2003.

[69] A. Nakhimovsky and W.J. Rapaport. Discontinuities in narratives. In *Proceedings of the 12th conference on Computational linguistics*, pages 465–470, Morristown, NJ, USA, 1988. Association for Computational Linguistics.

[70] M. Parenthoën P., Reignier, and J. Tisseau. Put fuzzy cognitive maps to work in virtual worlds. In *Proceedings of the 10th IEEE international conference on fuzzy Systems, 2001*, volume 1, pages 252–255, 2001.

[71] J. Paredis. Co-evolutionary constraint satisfaction. In *PPSN III: Proceedings of the international conference on evolutionary computation. The third conference on Parallel problem solving from nature, 1994*, pages 46–55, London, UK, 1994. Springer-Verlag.

[72] J.M. Pires and H. Prade. Logical analysis of fuzzy constraint satisfaction problems. In *Proceedings of the 7th IEEE international conference on fuzzy systems*, volume 1, pages 857–862, May 1998.

[73] S. Rabin. *AI Game Programming Wisdom*. Charles River Media, Inc., Rockland, MA, USA, 2002.

[74] S. Rajeev and C. S. Krishnamoorthy. Discrete optimization of structures using genetic algorithms. *Journal of Structural Engineering*, 118(5):1233–1250, 1992.

[75] M. Riedl, C. J. Saretto, and R.M. Young. Managing interaction between users and agents in a multi-agent storytelling environment. In *AAMAS '03: Proceedings of the second international joint conference on autonomous agents and multiagent systems, 2003*, pages 741–748, New York, NY, USA, 2003. ACM.

[76] M.O. Riedl and R.M. Young. From linear story generation to branching story graphs. *Computer Graphics and Applications, IEEE*, 26(3):23–31, May–June 2006.

[77] M. Riff. Evolutionary algorithms for constraint satisfaction problems. In *SCCC '98: Proceedings of the XVIII international conference of the Chilean computer science society, 1998*, pages 158–165, Washington, DC, USA, November 1998. IEEE Computer Society.

[78] P. Ross. Evolution of constraint satisfaction strategies in examination timetabling. In *GECCO '99: Proceedings of the Genetic and evolutionary computation conference, 1999*, pages 635–642, San Francisco, CA, USA, 1999. Morgan Kaufmann.

[79] Z. Ruttkay. Fuzzy constraint satisfaction. In *Proceedings of the 3rd IEEE international conference on fuzzy systems*, pages 1263–1268, June 1994.

[80] T. Schiex. Solution reuse in dynamic constraint satisfaction problems. In *Proceedings of the 12th national conference on artificial intelligence, 1994*, volume 1, pages 307–312, Menlo Park, CA, USA, 1994. American Association for Artificial Intelligence.

[81] T. Schiex and G. Verfaillie. Nogood recording for static and dynamic constraint satisfaction problems. In *TAI '93: Proceedings of the fifth international conference on Tools with artificial intelligence, 1993*, pages 48–55, November 1993.

[82] L. Schoofs and B. Naudts. Ant colonies are good at solving constraint satisfaction problems. In *Proceedings of the 2000 Congress on evolutionary computation, 2000*, volume 2, pages 1190–1195, Washington, DC, USA, July 2000. IEEE Computer Society.

[83] L. Schoofs and B. Naudts. Swarm intelligence on the binary constraint satisfaction problem. In *CEC '02: Proceedings of the 2002 Congress on evolutionary computation, 2002*, volume 2, pages 1444–1449, Washington, DC, USA, 2002. IEEE Computer Society.

[84] C.E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, July–October 1948.

[85] H. Shi, R.Ward, and N. Kharma. Expanding the definitions of linguistic hedges. In *Proceedings of the joint 9th IFSA world congress and 20th NAFIPS international conference, 2001*, volume 5, pages 2591–2595, July 2001.

[86] C. Solnon. Ants can solve constraint satisfaction problems. *IEEE Transactions on Evolutionary Computation*, 6(4):347–357, August 2002.

[87] M. Srinivas and L.M. Patnaik. Genetic algorithms: A survey. *Computer*, 27(6):17–26, June 1994.

[88] K.E. Steiner and J. Tomkins. Narrative event adaptation in virtual environments. In *IUI '04: Proceedings of the 9th international conference on Intelligent user interfaces*, pages 46–53, New York, NY, USA, 2004. ACM.

[89] W.P. Su, B. Pham, and A. Wardhani. High-level control posture of story characters based on personality and emotion. In *IE2005: Proceedings of the second Australasian conference on Interactive entertainment*, pages 179–186, Sydney, Australia, 2005. Creativity & Cognition Studios Press.

[90] V. Tam and P. Stuckey. An efficient heuristic-based evolutionary algorithm for solving constraint satisfaction problems. In *INTSYS '98: Proceedings of the IEEE international joint symposia on Intelligence and systems, 1998*, page 75, Washington, DC, USA, May 1998. IEEE Computer Society.

[91] E. Tsang. *Foundations of Constraint Satisfaction*. Academic Press Limited, Sandiego, CA, USA, 1993.

[92] R.J.M. Vaessens and E.H.L. Aarts. Job shop scheduling by local search. *INFORMS Journal on Computing*, 8:302–317, 1996.

[93] W. van Leekwijck and E.E. Kerre. Defuzzification: criteria and classification. *Fuzzy Sets and Systems*, 108(2):159–178, 1999.

[94] M. van Lent. Guest editor's introduction: Interactive narrative. *Computer Graphics and Applications, IEEE*, 26(3):20–21, May–June 2006.

[95] L. Vila. A survey on temporal reasoning in artificial intelligence. *AI Communications*, 7(1):4–28, 1994.

[96] B. Watson, J. Kim, T. McEneany, T. Moher, C. Hindo, L. Gomez, and S. Fransen. Storyspace: Technology supporting reflection, expression, and discourse in classroom narrative. *Computer Graphics and Applications, IEEE*, 24(2):13–15, March–April 2004.

[97] J.M. Wiebe. Identifying subjective characters in narrative. In *Proceedings of the 13th conference on Computational linguistics*, pages 401–406, Morristown, NJ, USA, 1990. Association for Computational Linguistics.

[98] J.M. Wiebe. Tracking point of view in narrative. *Computational Linguistics*, 20(2):233–287, 1994.

[99] J.M. Wiebe and W.J. Rapaport. A computational theory of perspective and reference in narrative. In *Proceedings of the 26th annual meeting on Association for Computational Linguistics*, pages 131–138, Morristown, NJ, USA, 1988. Association for Computational Linguistics.

[100] T. Yang. Computational verb systems: Adverbs and adverbials as modifiers of verbs. *Information Sciences*, 121(3–4):147–175, December 1999.

[101] T. Yang. Computational verb systems: Computing with verbs and applications. *International Journal of General Systems*, 28(1):1–36, 1999.

[102] T. Yang. Computational verb systems: Modeling with verbs and applications. *Information Sciences*, 117(3–4):147–175, 1999.

[103] T. Yang. Computational verb systems: Verb logic. *International Journal of Intelligent Systems*, 14(11):1071–1087, November 1999.

[104] T. Yang. Computational verb systems: A new paradigm for artificial intelligence. *Information Sciences*, 124(1–4):103–123, May 2000.

[105] T. Yang. Computational verb systems: Verb sets. *International Journal of General Systems*, 20(6):941–964, 2000.

[106] T. Yang. *Advances in Computational Verb Systems*. Nova Science Publishers, Inc., Huntington, NY, USA, May 2001.

[107] T. Yang. Computational verb systems: Computing with perceptions of dynamics. *Information Sciences*, 134(1–4):167–248, May 2001.

[108] T. Yang. Computational verb systems: The paradox of the liar. *International Journal of Intelligent Systems*, 16(9):1053–1067, September 2001.

[109] T. Yang. *Impulsive Control Theory*, volume 272 of *Lecture Notes in Control and Information Sciences*. Springer-Verlag, Berlin, Germany, August 2001.

[110] T. Yang. Verb numbers. *International Journal of Intelligent Systems*, 16(5):655–678, May 2001.

[111] T. Yang. *Computational Verb Theory: From Engineering, Dynamic Systems to Physical Linguistics*, volume 2 of *YangSky.com Monographs in Information Sciences*. Yangs Scientific Research Institute, Tucson, AZ, USA, October 2002.

[112] T. Yang. Computational verb systems: Verbs and dynamic systems. *International Journal of Computational Cognition*, 1(3):1–50, September 2003.

[113] T. Yang. *Fuzzy Dynamic Systems and Computational Verbs Represented by Fuzzy Mathematics*, volume 3 of *YangSky.com Monographs in Information Sciences*. Yang's Scientific Press, Tucson, AZ, USA, September 2003.

[114] T. Yang. *Physical Linguistics: Measurable Linguistics and Duality Between Universe and Cognition*, volume 5 of *YangSky.com Monographs in Information Sciences*. Yangs Scientific Press, Tucson, AZ, USA, December 2004.

[115] T. Yang. Applications of computational verbs to digital image processing. *International Journal of Computational Cognition*, 3(3):31–40, September 2005.

[116] T. Yang. Applications of computational verbs to the design of p-controllers. *International Journal of Computational Cognition*, 3(2):52–60, June 2005.

[117] T. Yang. Bridging the universe and the cognition. *International Journal of Computational Cognition*, 3(4):1–15, December 2005.

[118] T. Yang. Applications of computational verbs to cognitive models of stock markets. *International Journal of Computational Cognition*, 4(2):1–13, June 2006.

[119] T. Yang. Applications of computational verbs to effective and realtime image understanding. *International Journal of Computational Cognition*, 4(1):49–67, March 2006.

[120] T. Yang. Applications of computational verbs to feeling retrieval from texts. *International Journal of Computational Cognition*, 4(3):28–45, September 2006.

[121] T. Yang. Applications of computational verbs to the study of the effects of russell's annual index reconstitution on stock markets. *Internation Journal of Computational Cognition*, 4(3):1–8, September 2006.

[122] T. Yang. Bridging computational verbs and fuzzy membership functions using computational verb collapses. *International Journal of Computational Cognition*, 4(4):47–61, December 2006.

[123] T. Yang. Computational verb decision trees. *International Journal of Computational Cognition*, 4(4):34–46, December 2006.

[124] T. Yang. Distances and similarities of saturated computational verbs. *International Journal of Computational Cognition*, 4(4):62–77, December 2006.

[125] T. Yang. Measures of ambiguity of computational verbs based on computational verb collapses. *International Journal of Computational Cognition*, 5(4):1–12, December 2007.

[126] R.M. Young and M. Riedl. Towards an architecture for intelligent control of narrative in interactive virtual worlds. In *IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces*, pages 310–312, New York, NY, USA, 2003. ACM.

[127] L.A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[128] L.A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning, part 1. *Information Sciences*, 8(1):199–249, 1975.

[129] L.A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning, part 2. *Information Sciences*, 8(1):301–357, 1975.

[130] L.A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning, part 3. *Information Sciences*, 9(1):301–357, 1975.

[131] J. Zhang and M. Fei. Determination of verb similarity in computational verb theory. *International Journal of Computational Cognition*, 3(3):74–77, September 2005.

Acronyms

This appendix lists the acronyms used in this paper. The acronyms are listed alphabetically in bold text, with the meaning of each acronym given alongside. If the meaning of an acronym is explained by a definition or section within the thesis, a reference to the appropriate definition or section is made.

AI Artificial Intelligence is a branch of computer science that attempts to recreate human intelligence in machines.

CSP Constraint satisfaction problems formulate problems in terms of the assignment of values to variables without the violation of specified constraints (refer to definition 3.25 in chapter 3).

CVT Computational verb theory models natural language verbs in terms of dynamic systems and vice versa (refer to section).

CVTNG The computational verb theory interactive narrative generation system presented in this work accepts natural language sentences as inputs and generates interactive narrative spaces such as graphical models as output (refer to chapter 5).

EEFE Evaluated evolving function equations are formed when the evolving function models in this work are evaluated for a specific time value.

FCSP Fuzzy constraint satisfaction problems are a relaxation of CSPs that uses fuzzy constraints which have associated membership functions that indicate the degree to which a constraint is satisfied (refer to definition 3.26 of chapter 3).

Appendix A

Symbols

This chapter defines the symbols used throughout this work. The symbols are listed for the chapter the symbol first appeared in. A brief description of every symbol is given next to the symbol. If a definition or equation within the work defines the symbol further, a reference to the definition or equation is given. The symbols are listed in the order of their first appearance.

A.1 Chapter 3: Fuzzy set theory, computational verb theory, and constraint satisfaction problem definitions

X The name of a variable (crisp or fuzzy).

X_i The i^{th} variable name.

x The generic name for the value of a variable X.

x_i The generic name for the value of a variable X_i.

U The universe of discourse for a variable X.

U_i The universe of discourse for the i^{th} variable.

u The generic name for a member of the universe of discourse U.

u_i The generic name for a member of the universe of discourse U_i.

$R(X;u)$ The subset of U that represents the restriction of the values for u by variable X.

$R(X)$ The subset (restriction) of domain values associated with the variable named X.

μ_R The characteristic function associated with the restriction R (refer to equation (3.4)).

q A sequence of index numbers $1, \ldots, k$ that designates a subset in a sequence of variables or domains.

q' The index values not in the index sequence q.

$\bigvee$ An operator that returns the supremum of its operand (a set) over its subscript (also a set).

P_q An operator that designates the projection its operand (variables) over a subsequence of domains, $U_1, \ldots, U_k$, identified by the index sequence q.

$\bigcup$ An operator that returns the union of its operand and subscript, which is a set containing the elements of both.

A A fuzzy set (refer to definition 3.6).

$\mathbf{X}$ A space of points i.e. a discrete or continuous domain.

$\mathbf{x}$ The generic name for a member of $\mathbf{X}$.

c A function returning the compatibility of a domain value, u, with a restriction $R(X)$ (refer to equation (3.18)).

CON A linguistic hedge function that acts as a concentrator of a fuzzy set (refer to equation (3.21)).

DIL A linguistic hedge function that acts as a dilator of a fuzzy set (refer to equation (3.22)).

g A function used as a linguistic hedge function.

η A constant value used as an exponent in linguistic hedge functions.

$\mathcal{L}$ The name of a linguistic variable.

$\mathcal{T}(\mathcal{L})$ The term set for a linguistic variable $\mathcal{L}$.

G A syntactic rule that generates names for linguistic variables.

M A semantic rule that associates a meaning with a variable in terms of a restriction R (refer to equation (3.23)).

t A time value.

$\dot{x}$ The change in value of a variable, x, over time.

$x(t)$ The state of a dynamic system at a time value t.

f A generic mathematical function.

x_0 The initial value of a variable x in a dynamic system.

$\phi_t(\cdot)$ A function that returns the solution of a dynamic system (a function) for the specified initial value parameter.

x^* A critical point in a dynamic system that returns a constant solution to the dynamic system for all time values.

$\mathcal{V}$ A computational verb (refer to definition 3.13 of chapter 3).

$\mathscr{E}_{\mathcal{V}}$ The evolving system of a computational verb $\mathcal{V}$ (refer to equation (3.30)).

$\mathcal{I}_{\mathcal{V}}$ The inner system of a computational verb $\mathcal{V}$ (refer to equation (3.31)).

$\mathcal{F}_{\mathcal{V}}$ The outer system of a computational verb $\mathcal{V}$ (refer to equation (3.32)).

$\mathscr{T}$ The time interval associated with an inner system.

$\mathscr{X}_s$ The physical space of an inner system.

$\mathscr{X}_p$ The perception space of an inner system.

$\mathscr{T}'$ The perceived time interval associated with an outer system.

$\mathscr{X}_s'$ The perceived physical space of an outer system.

$\mathscr{X}_p'$ The perceived space of perception of an outer system.

$\mathcal{E}_{\mathcal{V}}$ The evolving function associated with the computational verb $\mathcal{V}$.

$\mathbb{T}$ The time domain associated with an evolving function.

Ω The universe of discourse for an evolving function.

$\mathcal{S}$ A verb statement (refer to equation (3.35)).

$\mathcal{N}_s$ A subject noun in a verb statement (refer to equation (3.35)).

$\mathcal{N}_o$ An object noun in a verb statement (refer to equation (3.35)).

$\tau_{word}(\mathcal{V})$ The degree of "BE word" for a computational verb $\mathcal{V}$.

Φ^I A transformation applied to the inner system of a computational verb.

Φ^O A transformation applied to the outer system of a computational verb.

$\mathscr{C}$ A context of interpretation.

T A time interval associated with a verb–element (refer to definition 3.22 of chapter 3).

T_0 The birth time of a verb element (refer to definition 3.22 of chapter 3).

$T_{\mathscr{C}}$ The lifetime of a verb element (refer to definition 3.22 of chapter 3).

$\mathcal{C}(\mathcal{S})$ The collapse, $\mathcal{C}$, of a computational verb sentence (statement) $\mathcal{S}$.

$\mathscr{S}$ A set of collapses for a computational verb sentence (statement) $\mathcal{S}$.

$\mathscr{S}_{\mathcal{S}}$ A computational verb set for the verb sentence (statement) $\mathcal{S}$ (refer to equation (3.42)).

$\mathcal{C}(\cdot)$ A collapsing system.

Z A finite set of variables in a constraint satisfaction problem (CSP).

D A function that maps a variable x in a CSP to a finite set of objects.

C A set of constraints in a CSP.

A.2 Chapter 4: Computational verb theory and constraint satisfaction problem models for nouns, adjectives, verbs, and adverbs

T_b The birth time of an evolving function (refer to equation (4.3).

T_d The death time of an evolving function (refer to equation (4.3).

T_{b_j} The birth time of the j^{th} repetition of an evolving function (refer to equation (4.4)).

T_{d_j} The death time of the j^{th} repetition of an evolving function (refer to equation (4.4)).

α A scaling value for the amplitude of a generic function, f, (amplitude value) used in an evolving function in the form of equation (4.3).

α_i The amplitude value of the i^{th} term in an evolving function.

α_{i_j} The j^{th} (partial) amplitude value summed to obtain the amplitude value of the i^{th} term of an evolving function in the form of equation (4.7).

V An attribute variable used as a storage variable or amplitude value in an evolving function.

V_i The i^{th} attribute variable used as an amplitude value in an evolving function.

V_{i_j} The j^{th} attribute variable used as a partial amplitude value for the i^{th} term of an evolving function.

κ The scaling value of an attribute variable, V, used as an amplitude value in an evolving function.

κ_i The scaling value of the i^{th} attribute variable, V_i, used as an amplitude value in an evolving function.

κ_{i_j} The scaling value of the j^{th} attribute variable used as a partial amplitude value for the i^{th} term of an evolving function.

β An adverb in the context of computational verb theory.

$\beta_{\mathcal{T}}$ An adverb of time.

$\beta_{\mathcal{A}}$ An adverb of amplitude.

Φ_β A modifying system (mapping) applied to the evolving system of a computational verb by the adverb β.

φ_β A modifying system (mapping) applied to the outer system of a computational verb by the adverb β.

Ψ A transformation(modifying) function applied to the period or amplitude of an evolving function by an adverb.

δ A scaling factor of a time value t.

G A graph, related to an attribute variable V, that consists of attribute variable nodes and attribute variable relationship edges (graph of attribute variable relationships).

G_i A graph of attribute variable relationships related to the i^{th} attribute variable V_i.

F_i The value obtained by evaluating the i^{th} generic function, f_i, for a time value, t, in an evolving function $\mathcal{E}_{\mathcal{V}}$.

K A real-valued constant number obtained by summing the known values in an evaluated evolving function equation (EEFE).

$\mathbf{K}$ A column vector of the K values in all EEFEs with unknown amplitude values.

c_i The coefficient of the i^{th} term in an EEFE that contains at least one unkown amplitude value (refer to equation (4.27)).

$\mathbf{A}$ A coefficient matrix formed from the coefficients, c_i, in EEFEs containing unknown amplitude values.

$[\mathbf{A}|K]$ An augmented matrix formed from the coefficient matrix, $\mathbf{A}$, and the column vector $\mathbf{K}$.

$\overline{x}$ A tuple of candidate values for variables in an fuzzy constraint satisfaction problem (FCSP).

ϵ An error value that corresponds to the subsitution of a candidate measurement value value tuple, $\overline{x}$, into the corresponding unknown attribute variables of an EEFE in the form of equation (4.30).

ε The calculated scaling factor for an error value ϵ.

C A (fuzzy) constraint in a constraint satisfaction problem (CSP) or FCSP.

μ_C The membership function associated with the fuzzy constraint C in an FCSP.

α^+ The defined maximum value for an amplitude value α.

α_i^+ The defined maximum value for the i^{th} amplitude value α_i.

$\alpha_{i_j}^+$ The defined maximum value for the j^{th} partial amplitude value summed to obtain the i^{th} α_i.

α^- The defined minimum value for an amplitude value α.

α_i^- The defined minimum value for the i^{th} amplitude value α_i.

$\alpha_{i_j}^-$ The defined minimum value for the j^{th} partial amplitude value summed to obtain the i^{th} α_i.

d^- The minimum size of an EEFE that contains attribute variables whose values are unknown.

d^+ The maximum size of an EEFE that contains attribute variables whose values are unknown.

$\mathtt{d}_i$ The defined minimum value for the i^{th} amplitude value or partial amplitude value on the r.h.s. of an EEFE in the form of equation (4.27) or equation (4.29).

$\mathtt{d}_i$ The defined maximum value for the i^{th} amplitude value or partial amplitude value on the r.h.s. of an EEFE in the form of equation (4.27) or equation (4.29).

$d_{V_i}^-$ The global minimum value of the attribute variable V_i.

$d_{V_i}^+$ The global maximum value of the attribute variable V_i.

min An operator that returns the smallest member of an ordered set.

max An operator that returns the largest member of an ordered set.

D^-V_i An ordered set of all the minimum values defined or calculated for the attribute variable V_i.

D^+V_i An ordered set of all the maximum values defined or calculated for the attribute variable V_i.

d_ϵ^+ The boundary for the error value, ϵ, that is obtained when the maximum storage variable value and minimum amplitude values are substituted into the unkown attribute variables of an EEFE in the form of equation (4.30).

d_ϵ^- The boundary for the error value, ϵ, that is obtained when the minimum storage variable value and maximum amplitude values are substituted into the unkown attribute variables of an EEFE in the form of equation (4.30).

a A constant value specified for an amplitude value.

$a_{\mathscr{C}}$ A default value for an mapllitude value specified in a context of interpretation, sub-context, or partial context $\mathscr{C}$.

a_r A default value for an amplitude value retrieved from a representational element.

Ψ_T A transformation applied to the period, T, of an evolving function, $\mathcal{E}_\mathcal{V}$, if an adverb of period is applied to the computational verb $\mathcal{V}$.

$\Psi_\mathcal{A}$ A transformation applied to an amplitude value in an evolving function, $\mathcal{E}_\mathcal{V}$, if an adverb of amplitude is applied to the computational verb $\mathcal{V}$.

$\mathbf{M}$ A generic name for a matrix.

$\mathbf{p}$ A non-zero value in the pivot column of a matrix.

$\mathbf{r}$ The value stored in a row–column combination of a matrix.

g A gene in a chromosome of a genetic algorithm.

g_i The i^{th} gene in a chromosome of a genetic algorithm.

g_i^j The value of the j^{th} gene of the i^{th} chromosome of a genetic algorithm.

p A chromosome in the population of a genetic algorithm.

p_i The i^{th} chromosome in the population of a genetic algorithm.

g^- The minimum value between two gene values chosen from parent chromosomes.

g^+ The maximum value between two gene values chosen from parent chromosomes.

I The difference between the maximum value, g^+, and minimum value, g^-, chosen from the genes of parent chromosomes.

$\boldsymbol{\alpha}$ The blending factor used by the BLX-$\boldsymbol{\alpha}$ crossover operation.

d_{g_i} The domain value of the i^{th} gene of a chromosome in a genetic algorithm.

p_{var} A predefined stoppage value for algorithm 7 based on the change in fitness between generations.

p_{gen} A predefined stoppage value for algorithm 7 based on the number of generations.

Appendix B

Additional examples

This appendix presents two extended examples that illustrate the ability of the computational verb theory interactive narrative generation (CVTNG) system to process and model natural language sentences that contain multiple nouns, adjectives applied to a single noun, adverbs applied to a single adjective or verb, verbs within a single sentence, and sequences with of multiple actions. Section B.1 presents a sample sentence that illustrates the ability of the CVTNG parsing subsystem (refer to section 5.5) and the CVTNG subsystem for assigning measurement values to attribute values (refer to section 5.6 to handle sentences more complex than the sample sentences presented in chapter 6. The sample sentence presented in section B.1 is changed to introduce inconsistencies to examine the solution accuracy of the genetic algorithm for the resolution of fuzzy constraint satisfaction problems (FCSPs) (refer to algorithm 7) when applied to an FCSP that contains multiple variables and fuzzy constraints. Section B.2 presents a dynamic sample sentence that illustrates the ability of the CVTNG parsing subsystem, sequencing algorithm, and subsystem for assignning measurement values to attribute variables to handle sample sentences that describe dynamic behaviours that are more complex than those of the sample sentences presented in chapter 6.

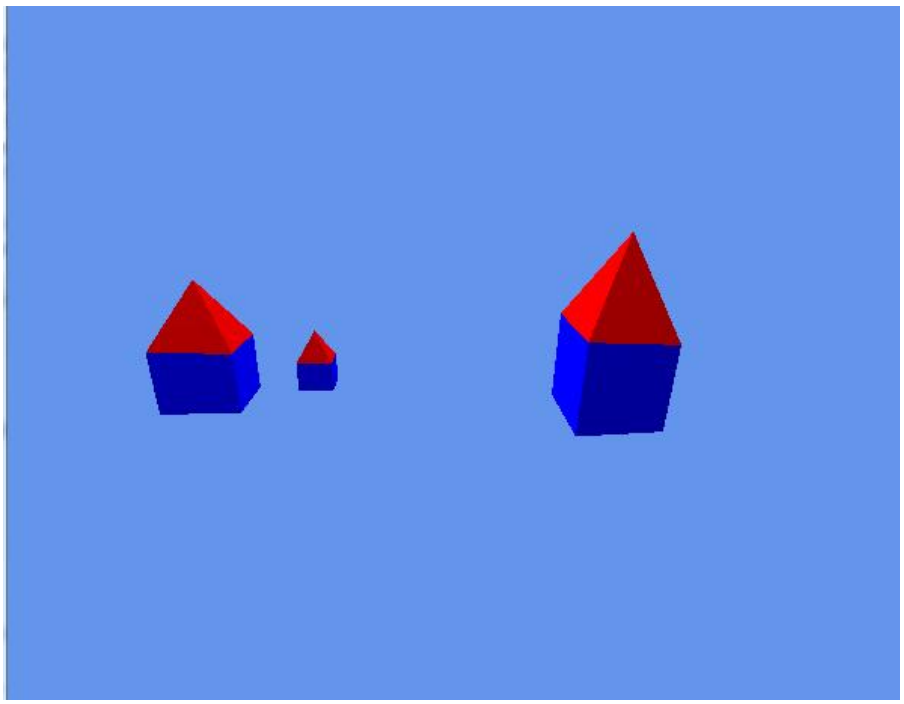

Figure B.1: The visual output produced for the sample sentences, "The red pyramid is on the blue cube. The small blue cube is very far to the left of the blue cube. The small red pyramid is on the small blue cube. The very small blue cube is to the right of the small blue cube. The very small red pyramid is on the very small blue cube."

B.1 Extended static example

Figure B.1 illustrates the visual output produced when the sample sentences, "The red pyramid is on the blue pyramid. The small blue cube is very far to the left of the blue cube. The small red pyramid is on the small blue pyramid. The very small blue cube is to the right of the small blue cube. The very small red pyramid is on the very small blue cube.", are processed by the CVTNG system and the measurement values obtained are passed as parameters to a representational system. The measurement values are passed as parameters to generated specifications of the `Element` interface that serve to transform 3D models according to the measurement value parameters received. The transformed 3D models are rendered to produce the output illustrated in figure B.2.

The parsing procedures presented in section 5.5 were extended to identify unique

noun instances not only by the adjectives that describe them (refer to section 5.4.3) but also by the adverbs that describe the adjectives in turn. The visual output produced for the sample sentence illustrates the ability of the CVTNG system to:

- apply the same evolving function models to different nouns with different representations;
- apply multiple adjectives to a single noun ("small' and "blue") and multiple adverbs to a single adjective ("very" and "far" to "is to the left of");
- correctly calculate measurement values for indirect relationships (the "very small blue cube" and "blue cube").

The sample sentences that correspond to figure B.1 may, for example, be altered by adding the sample sentences, "The small blue cube is very far to the right of the blue cube. The very small blue cube is to the left of the small blue cube.", when the sample sentences are provided as input to the CVTNG system. The newly introduced sample sentences create conflicting statements with regard to the positions of the "small blue" and "very small blue" instances of the noun "cube". The positions of the "small red" and "very small red" instances of the noun "pyramid" are also affected by the conflicting statements with regards to the position of the instances of the noun "cube" specified above. An FCSP with multiple fuzzy constraints and unknown variable values results.

Figure B.2 illustrates the average solution accuracy obtained over 30 iterations of algorithm 7 for optimal blending and population size values. A blending value of 0.7 was experimentally determined to provide optimal results in conjunction with a population size of 2. The population size and blending factor were determined in tandem and algorithm 7 was repeated 30 times for all population and blending factor combinations ranging from 2 to 20 and 0 to 1.0 respectively.

B.2 Extended dynamic example

Figure B.3 illustrates the output produced when the sample sentences, "The small blue cube is left of the blue cube. The very small red pyramid is left of the small blue cube.

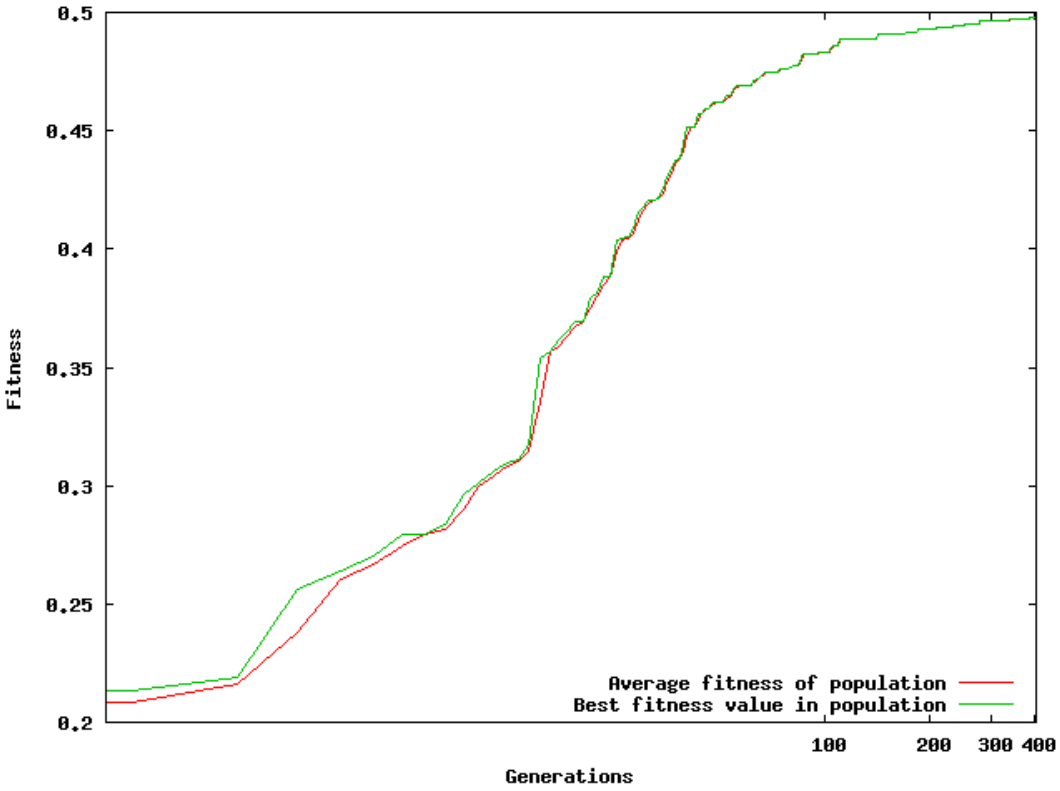

Figure B.2: The solution accuracy of the genetic algorithm for the resolution of fuzzy constraints applied to the sample sentences of section B.1 in addition to the sample sentences, "The small blue cube is very far to the right of the blue cube. The very small blue cube is to the left of the small blue cube."

The very small red pyramid grows and moves onto the small blue cube. The very small red pyramid stays on top of the small blue cube. The small blue cube grows and moves onto the blue cube.", are processed by the CVTNG system. The example illustrates the ability of the CVTNG system to:

- apply the same verb to different nouns that have different representations ("... pyramid grows ..." and "small blue cube grows ...");
- process multiple verbs in the same sentence that describe terminate actions ("The very small pyramid grows and moves onto the small blue cube.");
- maintain existing relationships while applying a dynamic action ("The very small

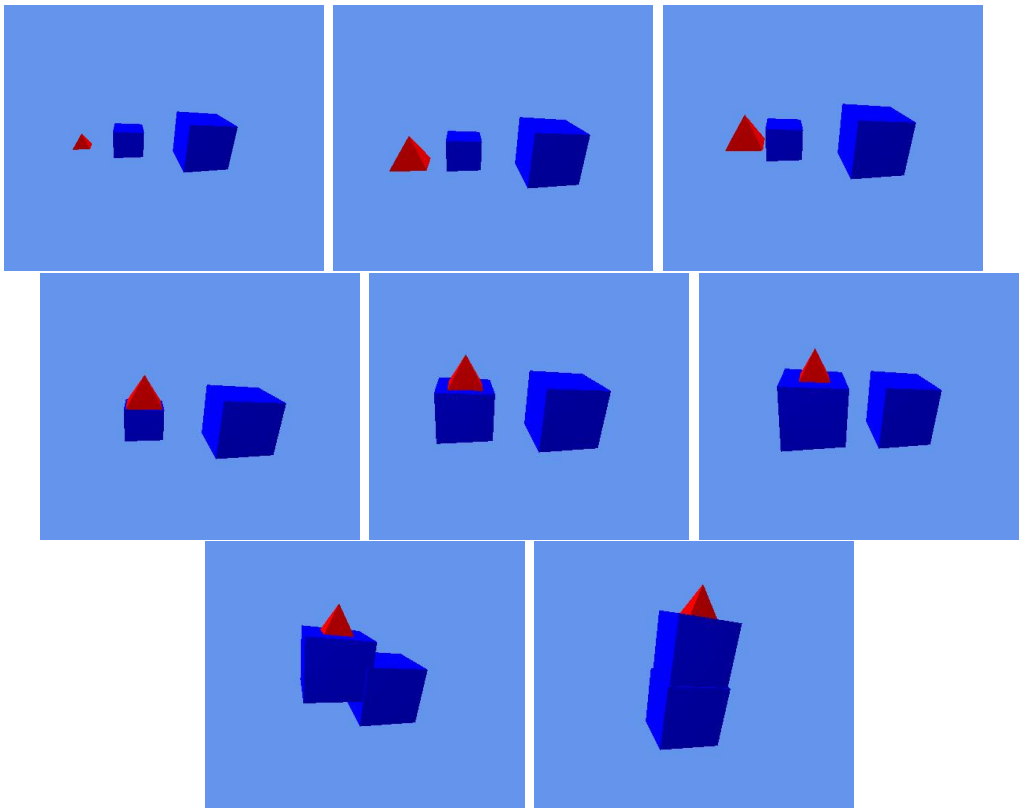

Figure B.3: The visual output produced for the sample sentences, "The small blue cube is left of the blue cube. The very small red pyramid is left of the small blue cube. The very small red pyramid grows and moves onto the small blue cube. The very small red pyramid stays on top of the small blue cube. The small blue cube grows and moves onto the blue cube.", for times values of $t = 0, t = 1, t = 1.5, t = 2.0, t = 2.5, t = 3.0, t = 3.5, t = 4.0$ in seconds (shown from left to right and top to bottom in time order stated)

red pyramid stays on top of the small blue cube. The small blue cube grows and moves onto the blue cube.").

If the CVTNG parsing algorithm is extended to use measurement values calculated by the CVTNG system to determine unique noun instances, the sample sentences would no longer need to repeatedly state the initial descriptives adjectives of a noun instance to uniquely identify the noun instance. The CVTNG would also be able to identify a noun based on its current state, for example, "The cube turns red. The red cube grows." would show a cube turning red and then becoming larger as the visual output. The feedback of measurement values to the CVTNG parsing subsystem is beyond the scope of this work.