Algebraic decoder specification: coupling formal-language theory and statistical machine translation

Matthias Büchse
matthias@buech.se

January 2015

Dissertation zur Erlangung des akademischen Grades Doktor rerum naturalium (Dr. rer. nat.)

vorgelegt an der	Technischen Universität Dresden Fakultät Informatik
eingereicht von	Dipl.-Inf. Matthias Büchse * 1983-08-12 in Köthen (Anhalt)
eingereicht am	2014-08-05
verteidigt am	2014-12-18
begutachtet durch	Prof. Dr.-Ing. habil. Heiko Vogler Technische Universität Dresden Prof. Dr. rer. nat. Alexander Koller Universität Potsdam

Abstract

The specification of a decoder, i.e., a program that translates sentences from one natural language into another, is an intricate process, driven by the application and lacking a canonical methodology. The practical nature of decoder development inhibits the transfer of knowledge between theory and application, which is unfortunate because many contemporary decoders are in fact related to formal-language theory. This thesis proposes an algebraic framework where a decoder is specified by an expression built from a fixed set of operations. As yet, this framework accommodates contemporary syntax-based decoders, it spans two levels of abstraction, and, primarily, it encourages mutual stimulation between the theory of weighted tree automata and the application.

Contents

List of Tables

List of Figures

List of Algorithms

1 Introduction

In statistical machine translation, a decoder is a mapping that is used to automatically translate sentences from one natural language into another. It goes without saying that such a mapping is typically very intricate. Therefore it is commonly specified on different levels of abstraction, which range from prose to equations to computer programs with hundreds of thousands of lines (see Fig. 1.1). Decoder development is mainly driven by the application, and the viability of a decoder has to be evaluated on real-world data. As a result, advances are usually due to practitioners; ad-hoc methods abound; and experience trumps codified knowledge, which presents a significant entry threshold for novices. In the absence of a canonical methodology, the refinement process, i.e., going from one level of abstraction to the next one, is particularly involved; intermediate levels are routinely being neglected; and the relationship between specifications on adjacent levels is informal at best. In the end, the intricate and practical nature of decoder development inhibits the transfer of knowledge between theory and application. This situation is unfortunate because many contemporary decoders are in fact related to formal-language theory.

Any effort to provide a method to mitigate this situation should pursue the long-term objectives shown in Tab. 1.1 (left column). This thesis seeks to take a first step towards such a method, with a high priority on Objective (d). To this end, this thesis proposes an algebraic framework where a decoder is specified by an expression built from a fixed set of operations. In the present form, the framework achieves the objectives to the degree shown in Tab. 1.1 (right column). These achievements rest on the three main contributions of this thesis, which comprise

1. the input product and the output product of a weighted synchronous context-free tree grammar and a weighted tree automaton (Ch. 3),

2. generic binarization of weighted grammars (Ch. 4), and

3. determinization of weighted tree automata using factorizations (Ch. 5).

We[1] proceed as follows. In the subsequent sections, we first review current approaches to decoder specification. Second, we introduce a preliminary version of the

[1] Throughout this work, “we” refers to the group of people consisting of the author and the reader.

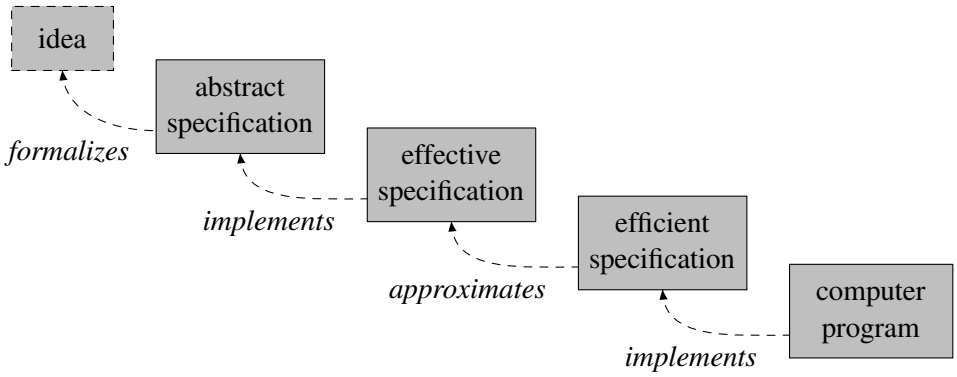


Figure 1.1: Decoder specifications on different levels of abstraction.

	long-term objective: any method should ...	*achievement:* the present framework ...
(a)	be versatile enough to accommodate the state of the art	accommodates contemporary syntax-based decoders
(b)	facilitate the refinement process (the cascade in Fig. 1.1)	permits refining each operation in isolation; as yet it only treats the "abstract" and the "effective" level
(c)	include formal relationships between adjacent levels of abstraction	guarantees equivalence between the two levels
(d)	encourage mutual stimulation between theory and application	is an interface to theory involving weighted tree automata and related devices; it asks for both exploiting and developing said theory
(e)	be easy to learn and to maintain	is difficult, because it incorporates many advanced concepts

Table 1.1: Long-term objectives vs. achievements.

proposed framework. Third, we review the three main contributions. We conclude this chapter with a brief overview of related work. In Ch. 2, we recall basic notions from formal-language theory and algebra, and we introduce the notation that we will use in the remaining chapters. Chapters 3–5 are dedicated to the three main contributions of this thesis. These chapters rely on Ch. 2, but are otherwise self-contained. Finally, Ch. 6 concludes this thesis; in particular, we revisit the achievements from Tab. 1.1, we consider the full version of the framework, and we discuss potential improvements of the proposed framework as well as open problems.

1.1 Decoder specification

The aim of *machine translation* is to use computers to automatically translate texts from one natural language to another, for example from French into English. Following a tradition established in [21], we will use this language pair as a proxy for any given language pair. In *statistical machine translation (SMT)*, translation rules are inferred automatically from a large body of existing translations, called a *parallel corpus*. Parallel corpora are readily available for many language pairs; e.g., the proceedings of the European parliament constitute several parallel corpora [109].

In the context of SMT, a *decoder* is a mapping

$$\mathbb{D}\colon \Omega \to E^F \,,$$

where Ω is a set called *parameter space*, E is the set of all English sentences, F is the set of all French sentences, and E^F denotes the set of all mappings from F to E. The problem of computing $\mathbb{D}(\omega)(f)$ for given ω and f is called *decoding*.

The process of devising a decoder is called *modelling*. In order to translate with a decoder, one first needs to fix a "good" element ω of Ω, given a parallel corpus $c \in (E \times F)^*$. This process is called *training*. Some training methods are guided by fundamental principles, others by heuristics and intuition. Ultimately, whether ω is indeed good is up to empirical *evaluation*; for this, we apply $\mathbb{D}(\omega)$ to previously unseen sentences, and we evaluate the resulting translations, either manually or automatically by comparing them with reference translations.

An introduction into SMT is given in [106, 122, 110, 173]; here we focus on how to build a decoder. Ideally, we follow a "refinement cascade", thereby specifying two decoders $\mathbb{D}_0$ and $\mathbb{D}$. This cascade consists of five specifications (cf. Fig. 1.1):

1. the *idea*, i.e., a description in prose based on examples;
2. the *abstract specification*, i.e., a mathematical description of a decoder $\mathbb{D}_0$ that is not necessarily constructive;

3. the *effective specification*, i.e., a constructive mathematical description of $\mathbb{D}_0$ that is not concerned with time and space limitations;

4. the *efficient specification*, i.e., a mathematical description of a decoder $\mathbb{D}$ that is efficient and approximates $\mathbb{D}_0$;

5. the *computer program* that implements $\mathbb{D}$.

In reality, (3) is usually omitted, and (4) is often fragmentary and presented in an operational style. Then (5) becomes the definitive specification of $\mathbb{D}$.

It is safe to say that (1) and (2) are well suited for a casual conversation and for formal reasoning, respectively. Since the empirical evaluation is based on (5), the question arises whether it permits any conclusions concerning the viability of (1) and (2); otherwise the conversation and the reasoning could be considered futile. Fortunately, for certain cases $\mathbb{D}_0$ and $\mathbb{D}$ coincide on real-world data [36]. In other cases, we assume that the transition from $\mathbb{D}_0$ to $\mathbb{D}$ introduces more "unhappy accidents" than "happy accidents", i.e., on average $\mathbb{D}_0$ is better than $\mathbb{D}$. Again, in certain cases, this assumption is backed by empirical evidence [39, Sec. 6.2] [164, Sec. 7].

1.2 Hierarchical phrases

Let us now illustrate the conventional specification of a decoder by means of a contemporary example, namely Hiero [38, 42, 39]. We begin with the underlying idea: hierarchical phrases. In order to translate an input sentence such as

"die katze ließ er frei" (German for "he freed the cat")

into English, we first segment it into phrases, and these phrases into subphrases, and so on. If we indicate (sub)phrases by square brackets, then we may obtain

[[die katze] ließ [er] frei] ,

where the whole sentence is a phrase that contains two subphrases. Second, we translate individual subphrases:

$$\text{die katze} \rightsquigarrow \text{the cat} , \qquad \text{er} \rightsquigarrow \text{he} , \qquad x_1 \text{ ließ } x_2 \text{ frei} \rightsquigarrow x_2 \text{ freed } x_1 . \tag{1.1}$$

Finally, we produce an English translation by composing the English subphrases to

"he freed the cat" .

ρ_1: $S \to \alpha_1(NP)$ $\alpha_1 = \langle x_1 \text{ ließ er frei}, \text{he freed } x_1 \rangle$
ρ_2: $S \to \alpha_2(PPER)$ $\alpha_2 = \langle \text{die katze ließ } x_1 \text{ frei}, x_1 \text{ let the cat out} \rangle$
ρ_3: $S \to \alpha_3(PPER, NP)$ $\alpha_3 = \langle x_1 \text{ ließ } x_2 \text{ frei}, x_1 \text{ freed } x_2 \rangle$
ρ_4: $S \to \alpha_4(PPER, NP)$ $\alpha_4 = \langle x_2 \text{ ließ } x_1 \text{ frei}, x_1 \text{ freed } x_2 \rangle$
ρ_5: $PPER \to \alpha_5$ $\alpha_5 = \langle \text{er}, \text{he} \rangle$
ρ_6: $NP \to \alpha_6$ $\alpha_6 = \langle \text{die katze}, \text{the cat} \rangle$

Figure 1.2: An SCFG for German-English SMT; the initial state is S.

We will see that the segmentation into subphrases as well as their translation can be captured by a finite set of rules that resemble (1.1).

It is possible that other segmentations and other translations are also valid, such as

[die katze ließ [er] frei] and die katze ließ x_1 frei $\rightsquigarrow x_1$ let the cat out ,

respectively. In this example, we obtain a different translation, namely

"he let the cat out" ,

but this need not be the case. At any rate, we want to output a single translation. Therefore we assign a real number, called *score*, to each way of segmenting and translating. Then we can either choose the way with the highest score and output the corresponding translation; or we can aggregate the scores of all ways that lead to the same translation and output the translation with highest aggregate score.

This concludes our account of the idea behind Hiero, and we proceed to the abstract specification. For this, we first formalize the aforementioned rules by means of *synchronous context-free grammars (SCFGs)*. This formalism first appeared in [119] by the name of syntax-directed transduction, and its viability for SMT was first shown via Hiero.

Let Σ be an alphabet. An *SCFG G over Σ* is a triple (Q, R, q_0) where Q is a finite set (of *states*), $q_0 \in Q$ is called *initial state*, and R is a finite set of *rules* of the form

$$q \to \langle w_1, w_2 \rangle (q_1, \ldots, q_k) ,$$

where $q, q_1, \ldots, q_k \in Q$ and w_i is a string over Σ and the variables $x_1, \ldots, x_k$ such that x_j occurs exactly once for every $j \in \{1, \ldots, k\}$. We call k the *rank* of the rule. For our example, we might use the SCFG G shown in Fig. 1.2, where $\Sigma = \{\text{ließ}, \text{er}, \text{frei}, \text{he}, \text{freed}, \ldots\}$.

Technically, we do not distinguish between Σ^*, E, and F; the latter two are merely more mnemonic. An SCFG G over Σ represents a set of pairs of strings in Σ^* by means

of two concepts: *abstract syntax trees (ASTs)* and *center trees*. Intuitively, an AST encodes a derivation of the grammar, and a center tree encodes the information about the derived string pair. The corresponding French and English strings are extracted from a center tree via mappings h_1 and h_2, respectively.

Let us now make these concepts more precise. For this, let Γ be an alphabet. We denote the set of all *trees over* Γ by T_Γ; it is the smallest set $T \subseteq (\Gamma \cup \{(,)\} \cup \{,\})^*$ such that $\gamma(t_1, \ldots, t_k) \in T$ for every $k \in \mathbb{N}$, $\gamma \in \Gamma$, and $t_1, \ldots, t_k \in T$. For every state q we define the *set* $D^q(G)$ *of q-ASTs of* G as follows. The family $(D^q(G) \mid q \in Q)$ is the smallest family $(D^q \mid q \in Q)$ with $D^q = \{\rho(d_1, \ldots, d_k) \mid \rho \in R, \exists q_1, \ldots, q_k, \alpha\colon \rho = (q \to \alpha(q_1, \ldots, q_k)), d_j \in D^{q_j}\}$. Let Γ be the set of all $\langle w_1, w_2 \rangle$ that occur in R, and let $\pi_\Gamma\colon T_R \to T_\Gamma$ be the mapping that replaces each label $q \to \alpha(q_1, \ldots, q_k)$ by α. Then a *center tree* is a tree over Γ that is obtained from an element of $D^{q_0}(G)$ by applying π_Γ. In our example, $\rho_4(\rho_5, \rho_6)$ is an S-AST and $\alpha_4(\alpha_5, \alpha_6)$ is a center tree.

We define $h_1, h_2\colon T_\Gamma \to \Sigma^*$ recursively by letting $h_i(\langle w_1, w_2 \rangle(t_1, \ldots, t_k))$ be the string obtained from w_i by replacing every occurrence of x_j by $h_i(t_j)$. For instance,

$$h_1(\alpha_4(\alpha_5, \alpha_6)) = h_1(\alpha_6) \text{ ließ } h_1(\alpha_5) \text{ frei} = \text{die katze ließ er frei} = h_1(\alpha_1(\alpha_6)) \,,$$
$$h_2(\alpha_4(\alpha_5, \alpha_6)) = h_2(\alpha_5) \text{ freed } h_2(\alpha_6) = \text{he freed the cat} = h_2(\alpha_1(\alpha_6)) \,.$$

Let $f \in \Sigma^*$. Then the informal process of segmenting f into phrases corresponds to finding a center tree t such that $h_1(t) = f$, and the informal process of translating the subphrases corresponds to computing $h_2(t)$. As stated above, there may be several center trees t with $h_1(t) = f$, and we intend to use scores in order to choose a translation. Now we turn to the formalization of these scores.

To this end, we consider an approach that is almost universally applied, namely *linear models* [152, 173]. Here we assign to each tree over R a linear combination of feature values for this tree. A *feature* is a mapping $\phi\colon T_R \to \bar{\mathbb{R}}$, where $\bar{\mathbb{R}} = \mathbb{R} \cup \{-\infty, \infty\}$. It is up to the engineer to devise suitable features. For the sake of simplicity, we focus on three of Hiero's seven features:

- We assume that we have a *probability assignment for* G, i.e., a mapping $\mu\colon R \to [0, 1]$; we extend μ to T_R and we define the feature ϕ_μ by letting

  $$\mu(\rho(d_1, \ldots, d_k)) = \mu(d_1) \cdots \mu(d_k) \cdot \mu(\rho) \,, \qquad \phi_\mu(d) = \log \mu(d) \,,$$

  where we assume that $\log 0 = -\infty$.

- Likewise, we assume that we have a probability distribution P_{LM} on E, called a *language model*; and we define the feature ϕ_{LM} by

  $$\phi_{\text{LM}}(d) = \log P_{\text{LM}}(h_2(\pi_\Gamma(d))) \,.$$

- We also count the number of words in the English string, i.e., we let

$$\phi_{\#}(d) = |h_2(\pi_\Gamma(d))| \,.$$

The first and the second feature can be regarded as scoring the faithfulness and the fluency of the translation, respectively [99, Sec. 25.3]. For Hiero, P_{LM} is an n-gram model. It is beyond the scope of this text to explain how such a language model is obtained (for details, see [99, Ch. 4]); suffice it to say that it can be simulated by a deterministic weighted string automaton [4, Sec. 4].

We note that a sequence $\phi_1, \ldots, \phi_m$ of features uniquely determines a mapping $\Phi\colon T_R \to \bar{\mathbb{R}}^m$ by

$$\Phi(d) = \begin{pmatrix} \phi_1(d) \\ \vdots \\ \phi_m(d) \end{pmatrix} \,,$$

and vice versa. We call Φ a *representation mapping (of dimension m)*.

Let $\theta \in \mathbb{R}^m$, which we call the *feature weight vector*. This vector contains the coefficients for our linear combination. Let $d \in T_R$. Then the *score of* d is $\Phi(d) \cdot \theta$, where $\cdot$ is the operation known variably as the dot product, scalar product, or inner product, i.e., $\Phi(d) \cdot \theta = \sum_{j\colon 1 \le j \le m} \phi_j(d) \cdot \theta_j$.

Finally, we arrive at the following abstract specification of Hiero:

$$\begin{aligned}
&\Omega = \{(G, \mu, \theta) \mid G \text{ is an SCFG, } \mu \text{ is a probability assignment for } G, \theta \in \mathbb{R}^3\} \,, \\
&\mathbb{D}_0\colon \Omega \to E^F,\ \mathbb{D}_0(G, \mu, \theta)\colon \\
&\qquad f \mapsto h_2\big(\pi_\Gamma\big(\mathrm{argmax}_{d \in D^{q_0}(G)\colon h_1(\pi_\Gamma(d)) = f}\ \Phi_{G,\mu}(d) \cdot \theta\big)\big) \,,
\end{aligned}$$

where $\Phi_{G,\mu}$ is the representation mapping consisting of ϕ_μ, ϕ_{LM}, and $\phi_{\#}$; and argmax is defined as follows. For every set X, we let $\mathrm{argmax}_X\colon \bar{\mathbb{R}}^X \to X$ be a partial mapping such that $\mathrm{argmax}_X(f)$ is a member of $\{x' \mid \forall x\colon f(x') \ge f(x)\}$ if that set is nonempty, and $\mathrm{argmax}_X(f)$ is undefined otherwise. Instead of $\mathrm{argmax}_X(f)$, we write $\mathrm{argmax}_{x \in X}\, f(x)$, and we usually silently assume that it is defined. Note that argmax_X is not uniquely determined; we stipulate that the above "let" fixed exactly the mapping that the (fictitious, potential) implementation of choice provides.

The structure of the parameter space Ω is in part motivated by the training method that is used with Hiero. For the sake of completeness, let us briefly sketch this method. Recall that training amounts to determining a specific triple $(G, \mu, \theta) \in \Omega$, given a parallel corpus $c \in (E \times F)^*$. First, we split c into two parts c_1 and c_2. Second, we perform *rule extraction*, i.e., we use a simple heuristic based on automatically induced word

alignments [151] to determine G and μ from c_1; for details, see [39, Secs. 3.2 and 4.3]. Finally, we determine the vector θ. To this end, let $c_2 = (e_1, f_1), \ldots, (e_l, f_l)$. Simply put, we select

$$\hat{\theta} = \operatorname{argmin}_{\theta \in \mathbb{R}^3} \sum_{j\colon 1 \leq j \leq l} L(\mathbb{D}_0(G, \mu, \theta)(f_j), e_j) \,, \tag{1.2}$$

where $L\colon E \times E \to \mathbb{R}$ is a mapping, called a *loss function*, and $L(e'_j, e_j)$ is our loss when f_j is translated to e'_j instead of the reference translation e_j. A typical loss function employed for Hiero is based on the BLEU score [153]. When this loss function is used, then (1.2) is called *minimum error-rate training (MERT)* [150]. Other loss functions are also common; for details see [84].

This concludes our account of the abstract specification. For the sake of brevity, we only summarize the effective specification. By means of a weighted deductive parsing system [89], Fig. 8 of [39] provides a weighted hypergraph that finitely encodes the mapping that maps each AST in $\{d \mid d \in D^{q_0}(G), h_1(\pi_\Gamma(d)) = f\}$ to its score. In principle, the highest-scoring AST can now be found by solving a shortest-path problem on the hypergraph. For this we may use standard algorithms such as Knuth's algorithm [108, 142] and the like [94, 28].

As indicated in [39, Sec. 6.2], the decoder $\mathbb{D}_0$ is not practical, for decoding a sentence with $\mathbb{D}_0$ takes too much time. The efficient specification therefore describes a decoder $\mathbb{D}$ that differs from $\mathbb{D}_0$ in two respects: (a) the parameter space is restricted to a certain subclass of SCFGs, and (b) the search for the highest-scoring AST is performed approximately. The algorithm that performs this approximate search is a variant of the aforementioned shortest-path algorithms, and it is dubbed *cube pruning*. Like said algorithms, cube pruning explores the weighted hypergraph, but it enforces a (user-defined) limit concerning the number of visited nodes [39, Sec. 5.3.4]. Recently, an exact alternative to cube pruning has been described [164].

In more recent years, variants of Hiero have been investigated [13, 121] that do not choose the highest-scoring AST, but the best translation, as follows:

$$\begin{aligned} &\mathbb{D}'_0\colon \Omega \to E^F,\ \mathbb{D}'_0(G, \mu, \theta)\colon \\ &\qquad f \mapsto \operatorname{argmax}_{e \in E} \sum_{d \in D^{q_0}(G)\colon h_1(\pi_\Gamma(d)) = f, h_2(\pi_\Gamma(d)) = e} \exp(\Phi_{G,\mu}(d) \cdot \theta) \,, \end{aligned}$$

where $\exp$ is the natural exponential function, i.e., $\exp(x) = e^x$, and we assume that $\exp(-\infty) = 0$.

Decoding with $\mathbb{D}'_0$ is known to be NP hard [121, 172, 35]. Therefore, the efficient specification describes a different decoder that approximates $\mathbb{D}'_0$. The decoder of [13, Sec. 3.3] uses a method that might be dubbed *beam search*. Like cube pruning, it explores a hyphergraph – albeit a slightly different one – with a limited-memory restriction. In contrast, the decoder of [121] uses a technique called *variational decoding*. Here we use the same hypergraph as for cube pruning, and we compute the *inside*

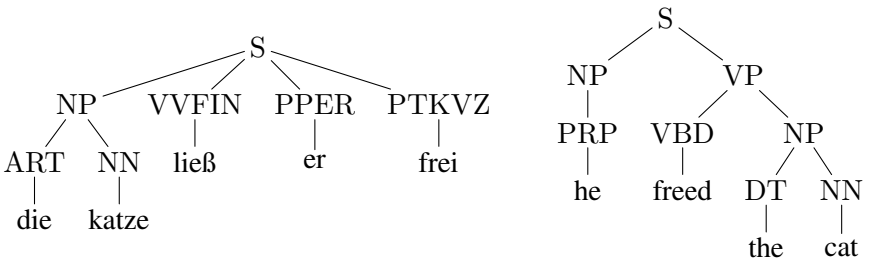


Figure 1.3: Constituent trees.

weight and *outside weight* of each node [121, 8, 118, 157]. Using these weights, we determine an n-gram language model that is "as close as possible" to the mapping

$$e \mapsto \tfrac{1}{Z} \cdot \textstyle\sum_{d \in D^{q_0}(G):\ h_1(\pi_\Gamma(d))=f, h_2(\pi_\Gamma(d))=e} \exp(\Phi_{G,\mu}(d) \cdot \theta) \ ,$$

where Z is the normalization constant $\sum_{d \in D^{q_0}(G):\ h_1(\pi_\Gamma(d))=f} \exp(\Phi_{G,\mu}(d) \cdot \theta)$. Since the n-gram language model is basically a particular deterministic weighted string automaton, the highest-weighted string can be found easily using standard shortest-path algorithms such as Dijkstra's algorithm [55, 70]. We note that variational decoding is very similar in spirit to [143].

At this point, it should be evident that we cover substantial distances when we refine a conventional specification, going from an "argmax formula" on the one end to various algorithms that work on a weighted hypergraph on the other end. This kind of refinement requires a great deal of mental labor, and it is hard to see how the two ends are related.

1.3 Explicit syntax

In recent years, decoders have been investigated that go beyond hierarchical phrases by considering explicit syntax information in the form of constituent trees, such as those in Fig. 1.3. These decoders are based on formalisms similar to SCFGs, such as

- tree-to-string transducer (yXTT) [79, 96, 90],
- synchronous tree-substitution grammar (STSG) [62],
- synchronous tree-sequence-substitution grammar [183, 130],
- synchronous tree-adjoining grammar (STAG) [171, 1],

- synchronous tree-insertion grammar (STIG) [167, 148, 149, 147, 51, 50],
- extended multi-bottom-up tree transducer (MBOT) [20, 65].

While yXTTs employ explicit syntax information on the source or target side only, the other formalisms do so on both sides. For the rule extraction of these formalisms, we use a parallel corpus that contains constituent trees instead of sentences. There are two principal advantages of explicit syntax information:

- Rule extraction is linguistically more informed due to the constituent trees in the training data [79, 40].
- We can use the constituent trees generated by the grammar to define more sophisticated features [41].

Let us elucidate the syntax-based approach by means of an example decoder that is based on STSGs. If we replace $\alpha_1, \ldots, \alpha_6$ in Fig. 1.2 by the values given in Fig. 1.4, then we obtain an STSG. The rules now have the form $q \to \langle t_1, t_2 \rangle(q_1, \ldots, q_k)$, where t_i is a tree over $\Sigma \cup \{x_1, \ldots, x_k\}$, and each variable occurs exactly once in t_i. The notions of a probability assignment, an AST, and a center tree carry over to this new setting. We define the mappings $h_1, h_2 \colon T_\Gamma \to T_\Sigma$ as for SCFGs, only that we perform the variable replacement in a tree instead of a string. For instance, if we denote the trees of Fig. 1.3 by t_1 (left tree) and t_2 (right tree), then

$$h_i(\alpha_4(\alpha_5, \alpha_6)) = t_i = h_i(\alpha_1(\alpha_6)) \; .$$

Moreover, we define the *yield mapping* $\mathrm{yd} \colon T_\Sigma \to \Sigma^*$ as follows. Let $t \in T_\Sigma$, $t = \sigma(t_1, \ldots, t_k)$. If $k = 0$, then $\mathrm{yd}(t) = \sigma$. Otherwise, $\mathrm{yd}(t) = \mathrm{yd}(t_1) \cdots \mathrm{yd}(t_k)$. Continuing the example, we have that

$$\mathrm{yd}(t_1) = \text{die katze ließ er frei} \; , \qquad \mathrm{yd}(t_2) = \text{he freed the cat} \; .$$

One feature that capitalizes on syntax information is the following, we call the *parsing feature*. We assume that we have the conditional probability $P(t \mid f)$ for every constituent tree t and foreign sentence f with $\mathrm{yd}(t) = f$. It is outside the scope of this text to explain how these probabilities are determined; this task is the subject of statistical natural-language parsing [99, Ch. 14]. Suffice it to say that said probabilities are usually represented finitely using formalisms akin to probabilistic context-free grammars; for more details, see [155, 156, 47, 12, 37]. Then the parsing feature is

$$\phi_{\mathrm{P}}(d) = \log P(h_1(\pi_\Gamma(d)) \mid \mathrm{yd}(h_1(\pi_\Gamma(d)))) \; .$$

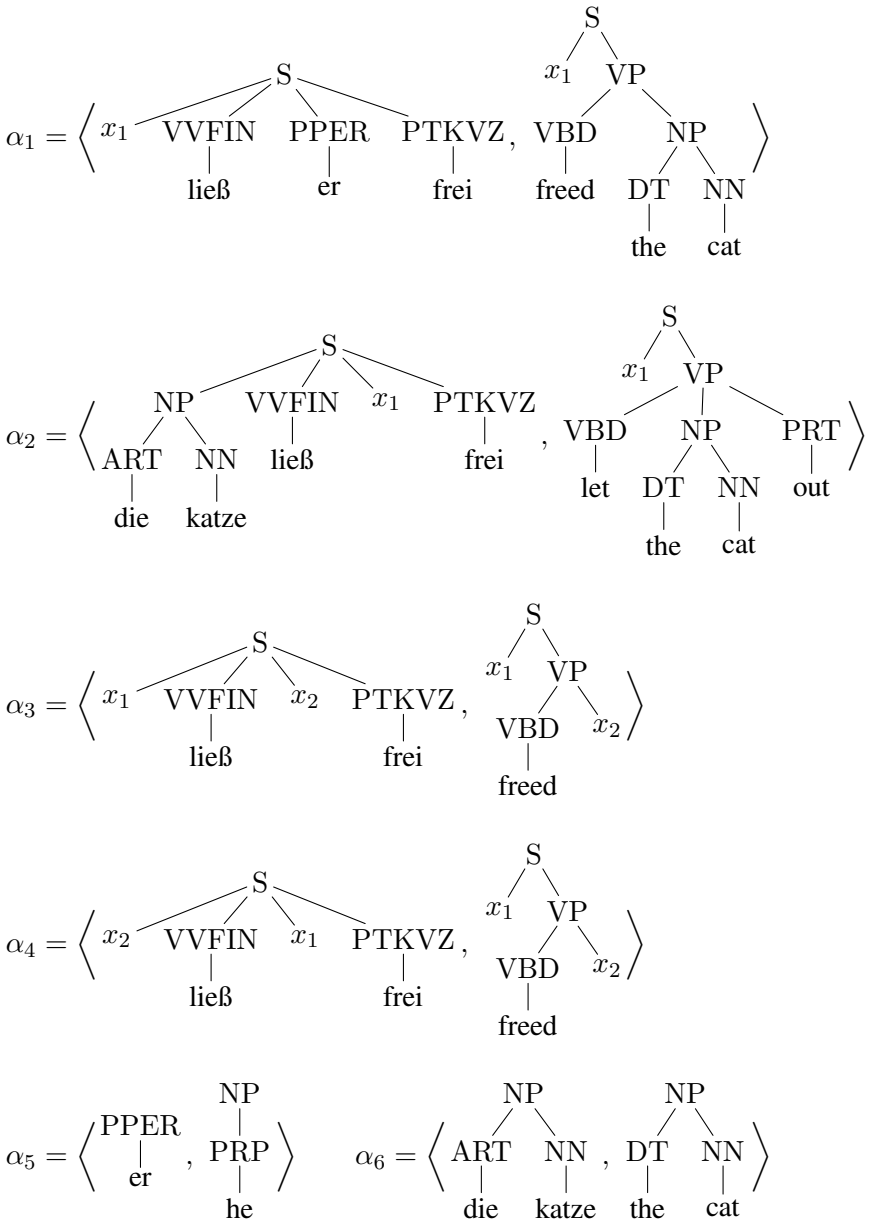


Figure 1.4: Tree pairs for an STSG.

This feature and variants thereof have been used successfully in [96, 137, 93].

Now we can define our example decoder as follows. We let

$$\Omega = \{(G, \mu, \theta) \mid G \text{ is an STSG}, \mu \text{ is a probability assignment for } G, \theta \in \mathbb{R}^3\} ,$$
$$\mathbb{D}_0 \colon \Omega \to E^F,\ \mathbb{D}_0(G, \mu, \theta) \colon$$
$$f \mapsto \mathrm{yd}(h_2(\pi_\Gamma(\mathrm{argmax}_{d \in D^{q_0}(G) \colon\ \mathrm{yd}(h_1(\pi_\Gamma(d))) = f}\ \Phi_{G,\mu}(d) \cdot \theta))) ,$$

where $\Phi_{G,\mu}$ is the representation mapping that consists of the three features ϕ_μ, ϕ_{LM} (adapted to the STSG case via yd), and ϕ_{P}.

1.4 The algebraic framework, preliminary version

The algebraic framework is essentially a collection of operations, and it allows us to define $\mathbb{D}(\omega)(f)$ as an expression over these operations, ω, and f. In order to keep the exposition simple, we only consider a preliminary version of the framework; the full version follows in Sec. 6.1. As a foundation, we utilize the notions of a weighted string language, a weighted tree language, and a weighted tree transformation [61, 57]. For the weight domain, we utilize the concept of a commutative semiring [91, 87].

A *(commutative) semiring* $\mathcal{S}$ is an algebraic structure consisting of a set S, called *domain*, two binary operations $+$ and $\cdot$ on S, called *addition* and *multiplication*, respectively, and neutral elements $0, 1 \in S$ for addition and multiplication, respectively. Furthermore, there are certain requirements that the operations be "well behaved"; for the purposes of this introduction, however, it is sufficient to imagine a commutative semiring as "a field without subtraction and division". For instance, the nonnegative reals $\mathbb{R}^{\geq 0}$, extended by ∞, with conventional addition and multiplication constitute the semiring Real. Another example is the *arctic semiring* Arct, where the domain is $\overline{\mathbb{R}}$, the operations are maximum for addition and (conventional) addition for multiplication, and the neutral elements are $-\infty$ and 0, respectively. A semiring is *complete* if, roughly speaking, infinite sums are defined. The two aforementioned semirings are complete. For a formal definition of semirings and complete semirings, see Sec. 2.3.2.

Let $\mathcal{S}$ be a commutative semiring and Σ an alphabet. A *weighted string language φ over Σ and $\mathcal{S}$* is a mapping $\varphi \colon \Sigma^* \to S$, a *weighted tree language φ over Σ and $\mathcal{S}$* is a mapping $\varphi \colon T_\Sigma \to S$, and a *weighted tree transformation τ over Σ and $\mathcal{S}$* is a mapping $\tau \colon T_\Sigma \times T_\Sigma \to S$. We abbreviate the corresponding sets as follows:

$$\mathcal{K} = S^{\Sigma^*} , \qquad \mathcal{L} = S^{T_\Sigma} , \qquad \mathcal{T} = S^{T_\Sigma \times T_\Sigma} .$$

We define the *string injection* $1.$, the *language yield* Yd, the *inverse language yield* Yd^{-1}, the *Hadamard product* $\odot$, the *input product* $\triangleleft$, the *output product* $\triangleright$, the *output*

$$
\begin{array}{rll}
1.\colon \Sigma^* \to \mathcal{K}\,, & (1.w)(w') = \text{if } w = w' \text{ then } 1 \text{ else } 0\,, & \\
\mathrm{Yd}\colon \mathcal{L} \to \mathcal{K}\,, & \mathrm{Yd}(\varphi)(w) = \sum_{t\colon\ \mathrm{yd}(t)=w} \varphi(t)\,, & (*) \\
\mathrm{Yd}^{-1}\colon \mathcal{K} \to \mathcal{L}\,, & \mathrm{Yd}^{-1}(\varphi)(t) = \varphi(\mathrm{yd}(t))\,, & \\
\odot\colon \mathcal{L} \times \mathcal{L} \to \mathcal{L}\,, & (\varphi_1 \odot \varphi_2)(t) = \varphi_1(t) \cdot \varphi_2(t)\,, & \\
\lhd\colon \mathcal{L} \times \mathcal{T} \to \mathcal{T}\,, & (\varphi \lhd \tau)(s,t) = \varphi(s) \cdot \tau(s,t)\,, & \\
\rhd\colon \mathcal{T} \times \mathcal{L} \to \mathcal{T}\,, & (\tau \rhd \varphi)(s,t) = \tau(s,t) \cdot \varphi(t)\,, & \\
\pi_2\colon \mathcal{T} \to \mathcal{L}\,, & \pi_2(\tau)(t) = \sum_s \tau(s,t)\,, & (\dagger) \\
\mathrm{best}\colon S^I \to I\,, & \mathrm{best}(\varphi) = \mathrm{argmax}_{i \in I}\, \varphi(i)\,. & (\ddagger)
\end{array}
$$

restrictions:
- $(*)$ $\mathcal{S}$ complete or $\{t \mid \mathrm{yd}(t) = w, \varphi(t) \neq 0\}$ finite
- $(\dagger)$ $\mathcal{S}$ complete or $\{s \mid \tau(s,t) \neq 0\}$ finite
- $(\ddagger)$ I set, $\mathcal{S} \in \{\mathrm{Real}, \mathrm{Arct}\}$

Figure 1.5: Operations of the algebraic framework.

projection π_2, and the *best-index operation* best as shown in Fig. 1.5. These operations constitute the preliminary version of the algebraic framework.

In order to illustrate the framework, we devise an alternative specification of $\mathbb{D}_0$ of Sec. 1.3. For this, let $\mathcal{S} = \mathrm{Arct}$, G an STSG, μ a probability assignment for G, $\theta \in \mathbb{R}^3$, and $\theta = (\theta_1, \theta_2, \theta_3)$. We claim that

$$\mathbb{D}_0(G, \mu, \theta)(f) = \mathrm{best}(\mathrm{Yd}(\pi_2\big((\mathrm{Yd}^{-1}(1.f) \odot \varphi_{\mathrm{P}}) \lhd \tau \rhd \mathrm{Yd}^{-1}(\varphi_{\mathrm{LM}})\big)))\,, \quad (1.3)$$

where $\tau \in \mathcal{T}$, $\varphi_{\mathrm{LM}} \in \mathcal{K}$, and $\varphi_{\mathrm{P}} \in \mathcal{L}$ with

$$
\begin{aligned}
\tau(t_1, t_2) &= \max\{\theta_1 \cdot \log \mu(d) \mid d \in D^{q_0}(G), h_i(\pi_\Gamma(d)) = t_i\}\,, \\
\varphi_{\mathrm{LM}}(e) &= \theta_2 \cdot \log P_{\mathrm{LM}}(e)\,, \\
\varphi_{\mathrm{P}}(t) &= \theta_3 \cdot \log P(t \mid \mathrm{yd}(t))\,.
\end{aligned}
$$

In order to prove the claim, we derive

$$
\begin{aligned}
&\mathbb{D}_0(G, \mu, \theta)(f) \\
&= \mathrm{yd}(h_2(\pi_\Gamma\big(\mathrm{argmax}_{d \in D^{q_0}(G)\colon\ \mathrm{yd}(h_1(\pi_\Gamma(d)))=f} \\
&\qquad \theta_1 \cdot \phi_\mu(d) + \theta_2 \cdot \phi_{\mathrm{LM}}(d) + \theta_3 \cdot \phi_{\mathrm{P}}(d)\big))) \\
&= \mathrm{argmax}_{w \in \Sigma^*}\, \mathrm{max}_{t \in T_\Sigma\colon\ \mathrm{yd}(t)=w}\, \mathrm{max}_{s \in T_\Sigma\colon\ \mathrm{yd}(s)=f} \\
&\qquad \mathrm{max}_{d \in D^{q_0}(G)\colon\ h_1(\pi_\Gamma(d))=s, h_2(\pi_\Gamma(d))=t}\, \theta_1 \cdot \phi_\mu(d) + \theta_2 \cdot \phi_{\mathrm{LM}}(d) + \theta_3 \cdot \phi_{\mathrm{P}}(d)
\end{aligned}
$$

$$
\begin{aligned}
&= \operatorname{argmax}_{w \in \Sigma^*} \max_{t \in T_\Sigma \colon \mathrm{yd}(t)=w} \max_{s \in T_\Sigma \colon \mathrm{yd}(s)=f} \\
&\qquad \varphi_{\mathrm{P}}(s) + \tau(s,t) + \varphi_{\mathrm{LM}}(\mathrm{yd}(t)) \\
&= \operatorname{argmax}_{w \in \Sigma^*} \max_{t \in T_\Sigma \colon \mathrm{yd}(t)=w} \max_{s \in T_\Sigma} \\
&\qquad (1.f)(\mathrm{yd}(s)) + \varphi_{\mathrm{P}}(s) + \tau(s,t) + \varphi_{\mathrm{LM}}(\mathrm{yd}(t)) \\
&= \mathrm{best}(\mathrm{Yd}(\pi_2\big((\mathrm{Yd}^{-1}(1.f) \odot \varphi_{\mathrm{P}}) \lhd \tau \rhd \mathrm{Yd}^{-1}(\varphi_{\mathrm{LM}})\big))).
\end{aligned}
$$

Note that, although we continue to use 1. to denote the string injection, the semiring 1 and the semiring 0 in our case are 0 and $-\infty$, respectively.

In the following, we unveil that the preliminary framework already exhibits the achievements from Tab. 1.1. As for (a), we already managed to specify a state-of-the-art decoder, apart from the limited repertory of features. As for (b)–(e), we proceed as follows. We will define subclasses of $\mathcal{K}$, $\mathcal{L}$, and $\mathcal{T}$ that correspond to certain formalisms such as weighted string automaton (WSA, [165, 11, 115, 166]), weighted tree automaton (WTA, cf. Sec. 2.4), or weighted context-free grammar (WCFG, [89, 46, 154]); in particular, we will define the notion of a weighted STSG (WSTSG, [74, 129]). Then we will gather established results about settings in which the aforementioned subclasses are effectively closed under the operations in Fig. 1.5. Finally, we will argue that the objects τ, φ_{LM}, and φ_{P} each belong to one of the new subclasses. At that point, it will become clear that (1.3) is already effective as it is (cf. (b), (c)), that we exploit the theory a great deal (cf. (d)), and that many concepts are involved (cf. (e)).

A *WSTSG G over Σ and $\mathcal{S}$* is a quadruple (Q, R, μ, q_0) where (Q, R, q_0) is an STSG over Σ and $\mu \colon R \to S$ is called *weight assignment*. The objects $D^q(G)$, Γ, π_Γ, and h_1 and h_2 are defined as for (Q, R, q_0). We define the mapping $\langle . \rangle_\mu \colon T_R \to S$ inductively by letting $\langle \rho(d_1, \ldots, d_k) \rangle_\mu = \langle d_1 \rangle_\mu \cdots \langle d_k \rangle_\mu \cdot \mu(\rho)$. Finally, the *meaning* $[\![G]\!]$ *of* G is the weighted tree transformation with

$$
[\![G]\!](t_1, t_2) = \textstyle\sum_{d \in D^{q_0}(G) \colon h_i(\pi_\Gamma(d))=t_i} \langle d \rangle_\mu \, .
$$

For this definition to be sound, we require one of two conditions: (i) $\mathcal{S}$ is complete or (ii) the index set of sum is finite for every (t_1, t_2). We can satisfy Condition (ii) by requiring that G be *productive* (cf. [74]), i.e., for every $\langle t_1, t_2 \rangle \in \Gamma$, we have $t_1, t_2 \neq x_1$. This is a common requirement (cf. [39, Sec. 3.2]). We note that a WSTSG is a particular WTA with an alternative meaning assigned to it.

We define the following classes:

$$
\begin{aligned}
\mathcal{K}_{\mathrm{Rec}} &= \{\varphi \mid \varphi \in \mathcal{K}, \varphi \text{ is the meaning of some WSA}\} \, , \\
\mathcal{K}_{\mathrm{CF}} &= \{\varphi \mid \varphi \in \mathcal{K}, \varphi \text{ is the meaning of some WCFG}\} \, , \\
\mathcal{L}_{\mathrm{Rec}} &= \{\varphi \mid \varphi \in \mathcal{L}, \varphi \text{ is the meaning of some WTA}\} \, , \\
\mathcal{T}_{\mathrm{STSG}} &= \{\tau \mid \tau \in \mathcal{T}, \tau \text{ is the meaning of some WSTSG}\} \, .
\end{aligned}
$$

operation	closure/restrictions	publications	complexity
1.	$\Sigma^* \to \mathcal{K}_{\text{Rec}}$	[11, 168]	$O(n)$
Yd	$\mathcal{L}_{\text{Rec}} \to \mathcal{K}_{\text{CF}}$	[176, 71]	$O(r)$
Yd^{-1}	$\mathcal{K}_{\text{Rec}} \to \mathcal{L}_{\text{Rec}}$	[132]	$O(p^k)$
$\odot$	$\mathcal{L}_{\text{Rec}} \times \mathcal{L}_{\text{Rec}} \to \mathcal{L}_{\text{Rec}}$	[15, Cor. 3.9]	$O(r_1 \cdot r_2)$
$\triangleleft$	$\mathcal{L}_{\text{Rec}} \times \mathcal{T}_{\text{STSG}} \to \mathcal{T}_{\text{STSG}}$	[128]	$O(r_2 \cdot p_1^{k_2})$
$\triangleright$	$\mathcal{T}_{\text{STSG}} \times \mathcal{L}_{\text{Rec}} \to \mathcal{T}_{\text{STSG}}$	[128]	$O(r_1 \cdot p_2^{k_1})$
π_2	$\mathcal{T}_{\text{STSG}} \to \mathcal{L}_{\text{Rec}}$	[75]	$O(r)$
best	$\mathcal{K}_{\text{CF}} \to \Sigma^*$, (†)	[108, 94, 28]	$O(r \cdot \log p)$
best	$\mathcal{L}_{\text{Rec}} \to T_\Sigma$, (†)	[108, 94, 28]	$O(r \cdot \log p)$
best	$\mathcal{T}_{\text{STSG}} \to T_\Sigma \times T_\Sigma$, (†)	[108, 94, 28]	$O(r \cdot \log p)$
best	$\mathcal{K}_{\text{Rec}} \to \Sigma^*$, (‡)	[134, 94], (Thm. 5.5.3)	(?)
best	$\mathcal{L}_{\text{Rec}} \to T_\Sigma$, (‡)	[134, 94], (Thm. 5.5.3)	(?)
Yd	$\mathcal{L}_{\text{CF}} \to \mathcal{K}_{\text{Mac}}$	[71]	$O(r)$
$\triangleleft$	$\mathcal{L}_{\text{Rec}} \times \mathcal{T}_{\text{SCFTG}} \to \mathcal{T}_{\text{SCFTG}}$	(Thm. 3.3.3)	$O(r_2 \cdot p_1^{c_2 \cdot k_2})$
$\triangleright$	$\mathcal{T}_{\text{SCFTG}} \times \mathcal{L}_{\text{Rec}} \to \mathcal{T}_{\text{SCFTG}}$	(Thm. 3.3.3)	$O(r_1 \cdot p_2^{c_1 \cdot k_1})$
π_2	$\mathcal{T}_{\text{SCFTG}} \to \mathcal{L}_{\text{CF}}$	(conjecture)	$O(r)$
best	$\mathcal{K}_{\text{Mac}} \to \Sigma^*$, (†)	(conjecture)	$O(r \cdot \log p)$
$\triangleleft$	$\mathcal{L}_{\text{Rec}} \times \mathcal{T}_{\text{STSG}} \to \mathcal{T}_{\text{STSG}}$, $(*)$	(Thm. 4.5.10, Sec. 4.6.5)	$O(r_2 \cdot p_1^3)$
$\triangleright$	$\mathcal{T}_{\text{STSG}} \times \mathcal{L}_{\text{Rec}} \to \mathcal{T}_{\text{STSG}}$, $(*)$	(Thm. 4.5.10, Sec. 4.6.5)	$O(r_1 \cdot p_2^3)$

legend:
n . . . length of the string
p . . . number of states
r . . . number of transitions/rules
k . . . maximal rank of a transition/rule
c . . . grammar-dependent constant

index 1, 2: first or second argument
(†) $\mathcal{S} = \text{Arct}$ and
either CFG/WTA acyclic [94] *or* weights negative [108]
(‡) $\mathcal{S} = \text{Real}$ and WTA unambiguous or acyclic [134, 94]
$(*)$ WSTSG rule-by-rule binarizable

Table 1.2: Computability of operations, with worst-case complexity.

The first section of Tab. 1.2 lists the closure results for our operations. Be advised that each entry corresponds to an algorithm; for instance, as implied by the table, [15] presents an algorithm that expects WTA M_1 and M_2 and outputs a WTA M with $[\![M]\!] = [\![M_1]\!] \odot [\![M_2]\!]$. We note that constructing a WSA for $1.f$ is straightforward, but it is beyond the scope of this text. Suffice it to say that, in the terminology of rational series [11, 165], $1.f$ is a polynomial; hence, it is rational and, thus, recognizable [168], which is tantamount to $1.f \in \mathcal{K}_{\text{Rec}}$. Furthermore, we note that the second conjunct in (†) guarantees that a best element exists. Technically, this condition should be incorporated into our subclasses $\mathcal{K}_{\text{CF}}$ and $\mathcal{L}_{\text{Rec}}$, or additional classes should be introduced, but we refrain from these complications. In practice, where unbounded translations are not in demand, it is often acceptable to simply *make* the CFG or the WTA in question acyclic, e.g., by removing transitions or "intersecting" it with a finite language.

We argue that (i) $\tau \in \mathcal{T}_{\text{STSG}}$, (ii) $\varphi_{\text{LM}} \in \mathcal{K}_{\text{Rec}}$, and (iii) $\varphi_{\text{P}} \in \mathcal{L}_{\text{Rec}}$. For (i), let G' be the WSTSG over Σ and Arct that is obtained from the STSG G by using the weight assignment μ' with $\mu'(\rho) = \theta_1 \cdot \log \mu(\rho)$. Then it is easy to verify that $[\![G']\!] = \tau$. For (ii) and (iii), we use that any n-gram model can be equivalently represented by a deterministic WSA M over Real [4, Sec. 4], and like [137, 93] we assume that the parsing probabilities are represented by a PCFG, which can be viewed as a bottom-up deterministic WTA M' over Real. Since deterministic devices do not employ the addition of the semiring, we can transfer them to the arctic semiring by applying $\log$ to each transition weight. (We will treat this construction more thoroughly in Sec. 6.1.) We transform the resulting WSA and WTA into a WSA for φ_{LM} and a WTA for φ_{P} by multiplying each transition weight by θ_2 and θ_3, respectively.

At this point, we can evaluate the expression on the right-hand side of (1.3) by composing the algorithms referred to in the table and applying the composite algorithm to the objects that we constructed for (i)–(iii). Put differently, (1.3) is effective.

So far, we only employed the framework to *rephrase* the definition of an existing decoder. Correspondingly, we had to prove (1.3). Now it is time to use the framework according to its purpose – to *specify* a decoder. We let $\mathcal{S} = \mathrm{Arct}$, and we define

$$\begin{aligned} \mathbb{D}_1 \colon \mathcal{T}_{\text{STSG}} \times \mathcal{K}_{\text{Rec}} \times \mathcal{L}_{\text{Rec}} \to E^F,\ \mathbb{D}_1(\tau, \varphi, \varphi') \colon \\ f \mapsto \mathrm{best}(\mathrm{Yd}(\pi_2\big((\mathrm{Yd}^{-1}(1.f) \odot \varphi') \lhd \tau \rhd \mathrm{Yd}^{-1}(\varphi)\big))) \,. \end{aligned} \tag{1.4}$$

Since we defined $\mathbb{D}_1$ "from scratch", we were able to define $\mathbb{D}_1(\tau, \varphi, \varphi')(f)$ by an expression over the operations and τ, φ, φ', and f. Cosmetic details aside, $\mathbb{D}_1$ and $\mathbb{D}_0$ are very similar; the principal difference is the absence of a feature weight vector in $\mathbb{D}_1$. Technically, we might assume that the feature weights are already present in τ, φ, and φ'. However, the training procedure usually determines the feature weight vector in a dedicated step, and it is at least debatable whether the training procedure should be

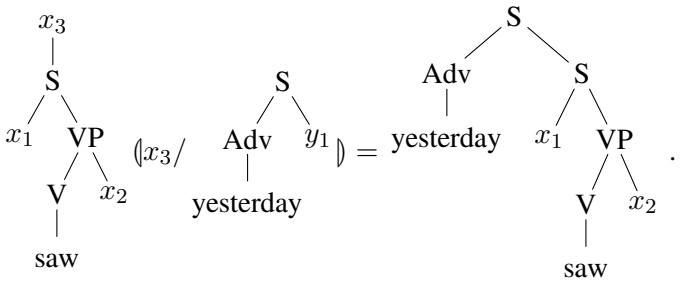


Figure 1.6: Applying second-order substitution.

burdened with the task of incorporating the feature weight vector into τ, φ, and φ'. The bottom line is that $\mathbb{D}_1$ lacks feature weights.

We end this section by discussing what differentiates the preliminary version of algebraic framework from the full version of Sec. 6.1. In the preliminary version, all operations act on the same semiring, and best basically forces us to choose Arct. Recall that our construction of G' was somewhat monolithic, as it already incorporated θ_1. This begs the question whether we have to provide a similar construction every time we modify (1.3), and the answer is probably yes. In the definition of τ, we apply the $\log$ and the multiplication with θ_1 on the level of individual ASTs, and this level is not exposed to the meaning of an WSTSG over the arctic semiring; it is "blurred" by the $\max$. More precisely, since multiplication does not distribute over $\max$ (consider $\theta_1 < 0$), we cannot "pull" this multiplication "out" of the $\max$, where it would be exposed. In other words, it is not possible to describe the integration of θ_1 as an operation on $\mathcal{T}$. This is the reason why $\mathbb{D}_1$ does not include feature weights. The full version of the framework permits "switching" the semiring via semiring homomorphisms. Then, using the multiset semiring, we are able to describe, i.a., the integration of θ_1 as an operation on $\mathcal{T}$. On the whole, this relieves us of the burden of constructing grammars, as in (i)–(iii).

1.5 Main contributions

1.5.1 Input product and output product of a weighted synchronous context-free tree grammar and a weighted tree automaton

So far, we have dealt with decoders based on SCFGs or on STSGs. Recently it has been suggested that SCFGs, STSGs, and yXTTs are not well suited to capture all phenomena that we encounter in real-world parallel corpora, and that STIGs and STAGs, among others, are better suited in that respect [170, 101, 83, 100].

These two formalisms are more powerful than the former three because they include an operation called second-order substitution. Roughly speaking, *second-order substitution* allows us to replace an occurrence of a variable x that has k successors, where $k > 0$ is permitted. The tree that we plug in for x usually contains the variables $y_1, \ldots, y_k$, and y_j is replaced by the jth successor of the occurrence of x. Figure 1.6 shows an example where we replace x_3; for a formal definition, see Sec. 2.2.2.

While STIGs do permit second-order substitution, they do so only in a limited fashion. In fact, they are weakly equivalent to yXTTs, which means that they have the same power for describing pairs of strings. To the author's knowledge, there are two decoders based on STIGs – [147] and [50] –, and they are fairly limited. More specifically, in the case of [147, Sec. 7.2.2], the variable arrangement in a rule has to follow a strict regime, and the decoder does not include a language-model feature. And in the case of [50, Sec. 4.3], decoding is accomplished by converting the STIG into a weakly equivalent yXTT. In this process, the explicit syntax information on the foreign side is lost. Consequently, this procedure is not suitable when we want to use the parsing feature.

Our algebraic framework is indifferent about the way in which we represent our weighted tree transformations – be it using an STSG, an STIG, or an STAG. Hence, we can readily use the framework to specify STIG- or STAG-based decoders; for instance, we can apply (1.3) also if G is an STAG. Crucially, this specification does not suffer from the limitations of the two above-mentioned decoders. There is one problem though: if we want our specification to be effective (let alone efficient), we can no longer rely on the first section of Tab. 1.2, because it mainly applies to $\mathcal{T}_{\text{STSG}}$.

In order to tackle this problem, we introduce further subsets of $\mathcal{K}$, $\mathcal{L}$, and $\mathcal{T}$ as follows, using the concepts of weighted context-free tree grammar (WCFTG, [19]), weighted synchronous context-free tree grammar (WSCFTG, cf. Sec. 3), and weighted macro grammar (WMG, called macro system in [71]):

$$\begin{aligned}
\mathcal{K}_{\text{Mac}} &= \{\varphi \mid \varphi \in \mathcal{K}, \varphi \text{ is the meaning of some WMG}\} \,, \\
\mathcal{L}_{\text{CF}} &= \{\varphi \mid \varphi \in \mathcal{L}, \varphi \text{ is the meaning of some WCFTG}\} \,, \\
\mathcal{T}_{\text{SCFTG}} &= \{\tau \mid \tau \in \mathcal{T}, \tau \text{ is the meaning of some WSCFTG}\} \,.
\end{aligned}$$

The class $\mathcal{T}_{\text{SCFTG}}$ subsumes the meanings of STAGs and STIGs [103].

The second section of Tab. 1.2 lists additional results concerning the computability of our operations. The results concerning the input and output product are taken from Ch. 3, and they constitute the first of the three main contributions of this thesis. To the author's knowledge, these results are novel, aside from the publications on which Ch. 3 is based. Therefore this contribution is crucial to underscore the viability of the algebraic framework for STIG-, STAG-, or WSCFTG-based decoders. As mentioned above, the framework does not impose the restrictions of current decoders.

1.5.2 Generic binarization of a weighted grammar

For the next main contribution of this thesis, let us turn to the matter of decoding complexity. As we can see from the first section of Tab. 1.2, the most expensive operations in decoding are the input and the output product. For both operations, the complexity is exponential in the maximal rank of any rule of the given WSTSG. The same complexity can be observed with established decoders, which is why they are only applied to grammars with maximal rank 2 (cf., e.g., [39, Sec. 3.2]) or to otherwise restricted grammars (cf., e.g., [147, Sec. 7.2.2]).

In view of these complexity considerations, it is a natural question whether we can transform a given grammar into an equivalent one where the maximal rank of any rule is bounded by a given constant; in particular, where it is bounded by 2. The latter kind of transformation is called *binarization*. It is well known that every CFG can be binarized [45], and that some SCFGs can not be binarized [2]. Hence, binarization procedures are in general partial. Here we shall focus on effective binarization procedures (how to construct a solution *in favorable cases*) rather than on purely existential statements (whether a solution exists).

The state of the art in binarization procedures is a rule-by-rule approach, where we replace each rule of rank greater than 2 by an equivalent collection of rules of rank at most 2, if possible. This approach has been applied to CFGs (for the Chomsky normal form) and to SCFGs [97]. On the other hand, binarization of yXTTs, STSGs, or WSCFTGs has – to the author's knowledge – not yet been investigated. As indicated by the third section of Tab. 1.2, having a binarization procedure for STSGs or WSCFTGs would underscore the viability of the algebraic framework, because it may improve the complexity. Of course, it is possible to try and construct several binarization procedures, one for yXTTs, one for STSGs, and one for WSCFTGs.

In contrast, the second main contribution of this thesis (Ch. 4) consists of (i) a generic rule-by-rule binarization procedure that can be tailored to many grammar formalisms by changing a parameter *at runtime* and (ii) considerations about the application to yXTTs and WSCFTGs (which subsume STSGs). The second item is crucial because said parameter is not trivial to come by, and moreover, it turns out that yXTTs and WSCFTGs do not lend themselves to binarization. As a remedy, we consider the (ad-hoc) formalisms of hedge-to-string transducers and weighted synchronous context-free hedge grammars, which encompass yXTTs and WSCFTGs, respectively.

1.5.3 Determinizing weighted tree automata using factorizations

In [134], it has been suggested that the translation quality can be improved by selecting the best English constituent tree instead of the best AST. On an abstract level, the

following decoders were compared (albeit for yXTT):

$$\Omega' = \{(G, \mu) \mid G \text{ is a productive STSG, } \mu \text{ is a probability assignment for } G\} ,$$
$$\mathbb{D}_1, \mathbb{D}_2 \colon \Omega' \to E^F ,$$
$$\mathbb{D}_1(G, \mu) \colon f \mapsto \mathrm{yd}(h_2(\pi_\Gamma(\mathrm{argmax}_{d \in D^{q_0}(G) \colon \mathrm{yd}(h_1(\pi_\Gamma(d))) = f} \mu(d)))) ,$$
$$\mathbb{D}_2(G, \mu) \colon f \mapsto \mathrm{yd}\big(\mathrm{argmax}_t \textstyle\sum_{d \in D^{q_0}(G) \colon \mathrm{yd}(h_1(\pi_\Gamma(d))) = f, h_2(\pi_\Gamma(d)) = t} \mu(d)\big) .$$

It turned out that $\mathbb{D}_2$ yields higher translation quality than $\mathbb{D}_1$ [134, Sec. 5.1].

In the algebraic framework, we obtain that

$$\mathrm{yd}(\mathrm{best}(\pi_2(\mathrm{Yd}^{-1}(1.f) \lhd [\![G']\!]))) = \begin{cases} \mathbb{D}_1(G, \mu) & \text{if } \mathcal{S} = (\bar{\mathbb{R}}, \max, \cdot, 0, 1), \\ \mathbb{D}_2(G, \mu) & \text{if } \mathcal{S} = \mathrm{Real}, \end{cases}$$

where G' is the WSTSG over Σ and $\mathcal{S}$ obtained from G by using the weight mapping μ. Since G' is productive, one can derive that the WTA for the output projection is acyclic.

Let us delve into how best is computed in both cases. The workhorse in this computation is a shortest-path algorithm for weighted hypergraphs [108, 94, 28], where a "path" corresponds to a run of the WTA, which is comparable to an AST of an STSG. Roughly speaking, the weight of a tree is the (semiring) sum of the weights of all runs on the tree. For $\mathbb{D}_1$, where the addition is $\max$, the highest possible weight of any tree coincides with the highest possible weight of any run, or: the "shortest" path. For $\mathbb{D}_2$, however, we can only exploit the shortest path if we make further assumptions concerning the given WTA. In fact, if the WTA is unambiguous – that is, for every tree, there is at most one run with non-zero weight –, then the highest possible weight of any tree again coincides with the highest possible weight of any run.

These considerations give rise to the question whether we can transform any given WTA into an equivalent WTA that is unambiguous. As in the case of binarization, we are interested in effective procedures that work in favorable cases rather than purely existential statements. Therefore, we turn to a related problem: transform a given WTA into an equivalent one that is *bottom-up deterministic*. This transformation is called *determinization*. Bottom-up determinism is a syntactic property that is easily decided in time linear in the number of transitions, and it implies the property of being unambiguous. It is well known that bottom-up deterministic WTA are strictly less powerful than WTA, so determinization procedures are partial.

In [134], the authors present a determinization procedure – albeit without proof – that applies to acyclic WTA over the nonnegative reals, and they put it in front of the shortest-path algorithm in order to compute best in Real. As in the case of $\mathbb{D}'_0$ of Sec. 1.2, decoding with $\mathbb{D}_2$ is NP hard, which is reflected in the complexity of the determinization procedure. Correspondingly, the authors state that determinization did

not finish in a reasonable amount of time for 26.7 % of their test sentences. Whenever the determinization procedure exceeded some fixed time limit, they fell back on an approximation method called *crunching*, where they determined the best tree by examining the 500 best runs of the WTA. Despite this occasional approximation, $\mathbb{D}_2$ produced better translations than both $\mathbb{D}_1$ and a version of $\mathbb{D}_2$ where determinization was completely replaced by crunching [134, Sec. 5.1].

Apart from best, determinization has another application in SMT, which is connected to the parsing feature and our argument for $\varphi_{\mathrm{P}} \in \mathcal{L}_{\mathrm{Rec}}$ in Sec. 1.4. Recall that this argument rested on the assumption that the parsing probabilities are represented by a PCFG. In contrast, modern-day parsers [155, 156] use an enriched formalism called PCFG with latent annotations (PCFG-LA). Like a PCFG, a PCFG-LA can be viewed as a WTA over the nonnegative reals; however, this WTA is far from being unambiguous. Clearly, if we are able to determinize this WTA, then we can again show that $\varphi_{\mathrm{P}} \in \mathcal{L}_{\mathrm{Rec}}$.

These two applications of determinization in SMT constitute a part of the motivation of the third and final main contribution of this thesis (Ch. 5): a determinization construction that generalizes and consolidates earlier work, including [134], which is thereby proved correct. However, it should be noted that the contribution is entirely theoretical, for it does not offer new use cases for SMT.

To be more specific, our construction generalizes [134] from the nonnegative reals to commutative semirings and [105] from WSA to WTA. The latter work requires that the semiring be extremal ($a + b \in \{a, b\}$) and that the WSA have a certain property called the twins property [44]. We transfer this property to the tree case, and we show that our construction applies with the same requirements. Moreover, we transfer results about the decidability of the twins property [5, 104] from the string case to the tree case.

1.6 Related work and bibliographic remarks

The algebraic framework proposed here draws inspiration from many sources and from ideas accumulated over time, and it is hard to trace them back to the origins. Therefore, the following account is most probably incomplete.

We defined our grammars in the spirit of bimorphisms [6]. The framework uses weighted tree languages and weighted tree transformations as the foundation, as opposed to WTA and WSTSG, respectively, which follows the idea that a specification should describe the "what" rather than the "how". This practice goes back to age-old notions such as a recognizable language or a rational language. Moreover, we used established operations such as the input product or the output projection.

From the perspective of universal algebra, the algebraic framework is essentially a many-sorted algebra [85], and the algorithms underlying Tab. 1.2 constitute a many-

sorted algebra as well, albeit with somewhat more fine-grained sorts. With suitable modifications, we may imagine that these two algebras have a common signature, and that the expression on the right-hand side of (1.4) is a term over that signature, where τ, φ, φ', and f are viewed as variables. By applying the corresponding homomorphism, we can interpret the term in either algebra, obtaining either a function that resembles $\mathbb{D}_0$ or an algorithm for computing said function.

It should be noted that conventional decoder specifications, such as the deductive system of [39, Fig. 8], do contain the automata-theoretic constructions for said operations, although in an implicit and interweaved manner, or adapted to special cases. By close inspection, a reader who is proficient in automata theory can "excavate" these operations.

A valuable source of information, certainly richer than the scant publications in SMT, is the program code of those decoders that are freely available, such as Moses [111], Joshua [120], or cdec [59]. A reader who is proficient in programming can learn a lot, in particular from cdec; for instance, when we view a synchronous grammar as a particular WTA, then a feature in cdec is merely a bottom-up deterministic WTA over the same alphabet, and feature weights are incorporated into said grammar via the Hadamard product – albeit approximately for complexity reasons.

For the sake of completeness, we note that even further variants of Hiero have been investigated, which choose neither the highest-scoring AST (cf. $\mathbb{D}_0$ in Sec. 1.2) nor the best translation (cf. $\mathbb{D}'_0$), but an English sentence that is similar (according to some similarity function) to many high-scoring translations. This approach is called *concensus decoding* [53, Sec. 2].

Algebraic decoder specification is not a new idea. For instance, Tiburon [135, 133] is a toolbox that allows to perform common operations on weighted tree transducers and weighted tree automata, such as Hadamard product, determinization, composition, application, and so on. Tiburon differs from our framework in three ways:

1. It focuses on automata and transducers rather than languages and transformations; therefore it is limited to the aforementioned devices.

2. It is limited to predefined semirings, most notably the tropical semiring and the nonnegative reals.

3. Next to a specification framework, it is primarily a computer program.

Another strand of research is concerned with interpreted regular tree grammars (or IRTGs, [112]; see also Sec. 4.2). Using the idea of initial-algebra semantics [86], this formalism unifies many common grammar formalisms, including CFGs, SCFGs, STSGs, etc. The IRTG framework differs from ours in three ways:

1. It is as yet unweighted. (Section 4.2 offers a weighted variant.)

2. It is not limited to tree languages or tree transformations.

3. Like Tiburon, it has a focus on manipulating grammars.

The IRTG framework is probably better viewed as a means of investigating grammar formalisms and their problems in a uniform way, rather than a specification framework. In Ch. 4, we will employ the IRTG framework in this spirit. We may also use IRTGs to produce new effectiveness results for our algebraic framework, comparable to Tab. 1.2. In fact, a precursor of our first main contribution has been described in this way [113]; cf. Sec. 3.1.

While the documentation for Tiburon and IRTGs describes critical operations for decoder specification, it remains vague and sketchy when it comes to the topic of actually specifying a state-of-the-art decoder.

Coincidentally, our notation for SCFGs is similar to the compact notation of [114] for linear context-free rewriting systems (LCFRSs). However, in contrast to SCFGs, LCFRSs do not treat the components in $\langle w_1, w_2 \rangle$ independently, and thus, there is a dedicated set of variables for each component. For instance, in the compact LCFRS notation, our rule ρ_3 could be written as

$$S \to \langle x_1 \text{ ließ } y_1 \text{ frei}, x_2 \text{ freed } y_2 \rangle (PPER, NP) \ .$$

Here the variants of x refer to the first successor ($PPER$) and the variants of y refer to the second successor (NP).

A promising alternative to STIGs and STAGs may be MBOTs. While they also exceed the power of STSGs, they are not based on second-order substitution. Instead, MBOTs permit specifying a sequence of trees on the English (target) side. In this regard, they can be viewed as "explicit-syntax versions" of synchronous linear context-free rewriting systems (SLCFRS) whose fanout on the source side is 1 [100].

It is the opinion of the author that the literature in both areas, formal-language theory and SMT, is somewhat unsatisfactory. In formal-language theory, the relevant sources span several decades, they follow varying notational conventions and varying approaches to semantics (such as term rewriting, fixpoint semantics, initial algebra semantics, etc.), and, on top of that, many texts are not available online, so that – even these days – the esteemed SMT practitioner has to plow through a library catalog, only to get acquainted with a topic that he is not necessarily fond of in the first place. It would be desirable to have a survey of modern formal-language theory, in particular, concerning semiring-weighted devices on strings and trees, that is available online and mentally accessible to practical and theoretical researchers alike.

In SMT, which admittedly is progressing rapidly, publications are often scant, ad-hoc, and particularly parsimonious when it comes to citations for established concepts; for instance, WCFGs are defined ad-hoc in [89, Sec. 2.3] (semiring-weighted) and [145, Sec. 2] (nonnegative reals), and neither publication cites a source. If this practice is due to the aforementioned obstacles concerning literature on formal-language theory, then an authoritative, comprehensive, yet plain survey of modern formal-language theory is all the more desirable.

2 Preliminaries

2.1 Mathematical foundations

Most concepts of this section can be found, e.g., in [179, Sec. 1.1, 1.3].

2.1.1 Sets, relations, mappings

By $\mathbb{N}$ we denote the set $\{0, 1, 2, \dots\}$ of nonnegative integers. We denote the empty set by $\emptyset$, set difference by $\setminus$, and the subset and the strict subset relations by $\subseteq$ and $\subset$, respectively. Let A and B be sets. Then B is a *partition of* A if $\emptyset \notin B$, $\bigcup_{b \in B} b = A$, and $b_1 \cap b_2 \neq \emptyset$ implies $b_1 = b_2$ for every $b_1, b_2 \in B$. The elements of a partition are also called *blocks*. The *powerset* $\mathcal{P}(A)$ *of* A is the set of all subsets of A; in particular, $\emptyset, A \in \mathcal{P}(A)$. If A is finite, then the *cardinality* $|A|$ *of* A is the number of elements of A. If $|A| = 1$, then we call A a *singleton*. We denote the *Cartesian product of* A *and* B by $A \times B$.

A *relation* R *from* A *into* B is a subset of $A \times B$. Let R be a relation from A into B. Instead of $(a, b) \in R$, we also write aRb. The *inverse* R^{-1} *of* R is the relation from B into A given by $R^{-1} = \{(b, a) \mid aRb\}$. Let C be a set and S a relation from B into C. The *relation product (or: composition)* $R; S$ *of* R *and* S is the relation from A into C defined by $R; S = \{(a, c) \mid \exists b\colon aRb, bSc\}$. Instead of $R; S$ (read "R, then S") we also write $S \circ R$ (read "S after R").

Let $A' \subseteq A$ and $B' \subseteq B$. By $R(A')$ we denote the set $\{b \mid \exists a \in A'\colon aRb\}$. The relation R is called

- *left-total on* A' if $A' \subseteq R^{-1}(B)$;
- *functional* if aRb and aRb' implies $b = b'$ for every $a \in A$ and $b, b' \in B$;
- *surjective on* B' if R^{-1} is left-total on B';
- *injective* if R^{-1} is functional;
- a *partial mapping from* A *into* B if it is functional; and
- a *mapping from* A *into* B if it is functional and left-total on A.

Let f be a partial mapping from A into B. We also say that f is *of type* $A \to B$, and instead of $f(\{a\}) = \{b\}$, we also write $f(a) = b$ or $a \mapsto b$. We call $f^{-1}(B)$ the *domain* $\operatorname{dom} f$ *of* f and $f(A)$ the *image of* f or *range of* f. For every $a \in \operatorname{dom} f$, we call $f(a)$ the *image of* a *(under* f*)* and we say that we *apply* f *to* a. Note that f is a partial mapping of type $A' \to B'$ iff $A' \supseteq \operatorname{dom} f$ and $B' \supseteq f(A)$; i.e., the type of f is not unique. If f is a mapping, then $\operatorname{dom} f = A$, and it is a mapping of type $A' \to B'$ iff $A' = A$ and $B' \supseteq f(A)$. We denote the fact that f is a mapping from A into B by $f\colon A \to B$. If we explicitly mention "partial mapping", then we may use the same notation in that sense as well.

Let $f\colon A \to B$. Clearly, f is surjective on $f(A)$. For every $b \in B$, we also write $f^{-1}(b)$ instead of $f^{-1}(\{b\})$, and we call $f^{-1}(b)$ the *preimage of* b *(under* f*)*. The *restriction* $f|_{A'}$ *of* f *to* A' is the mapping $f \cap (A' \times B)$. The mapping f is *bijective on* B' if it is injective and surjective on B'. If $B' = B$, then we omit the reference to B'. The set of all mappings of type $A \to B$ is denoted by B^A. Let $g\colon B \to C$. Then $f; g$ (alternatively, $g \circ f$) is a mapping from A to C with $a \mapsto g(f(a))$. Now let $g\colon C \to B$, $C \supseteq A$, and $g|_A = f$. Then g is *an extension of* f; note that f is already a partial mapping from C into B. We will sometimes *extend* f *to* C; formally, this means that we define an extension g of f, but instead of g, we will use the same symbol f. Naturally, since $f|_A$ is known, we will then only define $f|_{C \setminus A}$.

The *identity relation* id_A *on* A is defined by $id_A = \{(a, a) \mid a \in A\}$. Let $f\colon A \to A$. Then f is *idempotent* if $f = f \circ f$. For every $n \in \mathbb{N}$, we define n*-th iterate* f^n *of* f by letting $f^0 = id_A$ and $f^{n+1} = f^n \circ f$. An element $a \in A$ is called *fixpoint of* f of $f(a) = a$.

The set A is called *countably infinite* if there is a bijective mapping of type $A \to \mathbb{N}$, and it is *countable* if it is finite or countably infinite.

2.1.2 Families, sequences, and operations

Let I and A be sets. An I*-indexed family of elements of* A is a mapping a from I into A. Instead of "domain of a", we also call I the *index set of* a, and instead of $a(i)$ we write a_i. We denote the fact that a is an I-indexed family by $(a_i \mid i \in I)$. Note that this notation does not indicate A; in order to compensate, we usually state that $(a_i \mid i \in I)$ is a family of elements of A. We extend the Cartesian product to an arbitrary number of sets as follows. Let $(A_i \mid i \in I)$ be a family of sets. Then by $\times_i A_i$ we denote the set of all families $(a_i \mid i \in I)$ of elements of $\bigcup_i A_i$ with $a_i \in A_i$. If $I = \{1, 2\}$, then we identify the Cartesian product $A_1 \times A_2$ with $\times_i A_i$. If $I = \mathbb{N}$, then each element of $\times_i A_i$ is called a *sequence*, and we sometimes denote a sequence a by $(a_1, a_2, \dots)$. If $I = \{1, \dots, n\}$, then we denote $\times_i A_i$ by $A_1 \times \cdots \times A_n$, each element a in that set is called a *(finite) sequence (of length* n*)*, we denote a by $(a_1, \dots, a_n)$, and we have

$|a| = n$. We usually identify the sequence (a_1) with a_1. If $I = \emptyset$, then we observe that $\times_i A_i$ is a singleton, as there is only one mapping from $\emptyset$ into another set, namely $\emptyset$. In order to reduce confusion, we will denote this *empty sequence* by $()$ or ε.

A finite sequence of length n is also called an *n-tuple*; if $n = 2, 3, 4, 5$, then we also use the words pair, triple, quadruple, and quintuple, respectively. Let $f\colon A \to B$. We call the mapping f *n-ary* if there are $A_1, \ldots, A_n$ such that $A = A_1 \times \cdots \times A_n$; if $n = 0, 1, 2, 3$, then we also use the words nullary, unary, binary, and ternary, respectively. We usually write $f(a_1, \ldots, a_n)$ instead of $f((a_1, \ldots, a_n))$.

For every $n \in \mathbb{N}$, the *n-fold product A^n of A* is defined by $A^n = A_1 \times \cdots \times A_n$ with $A_i = A$. The *Kleene star A^* of A* is defined by $A^* = \bigcup_n A^n$. An *n-ary operation f on A is a mapping from A^n into A*. We often use symbols such as $+$ or $\cdot$ to denote binary operations, and then we use the infix notation $a + b$ instead of $+(a, b)$. A binary operation $\cdot$ is *associative* if $a_1 \cdot (a_2 \cdot a_3) = (a_1 \cdot a_2) \cdot a_3$, and it is *commutative* if $a_1 \cdot a_2 = a_2 \cdot a_1$. The *concatenation operation*, denoted by $\cdot$ or by juxtaposition, is the binary operation over A^* defined by $(a_1, \ldots, a_n) \cdot (b_1, \ldots, b_m) = (a_1, \ldots, a_n, b_1, \ldots, b_m)$. For every $w \in A^*$ and $n \in \mathbb{N}$ we define the *sequence iterate* w^n inductively by letting $w^0 = \varepsilon$ and $w^{n+1} = ww^n$.

An *alphabet* is a nonempty, finite set, and we call the elements of an alphabet *symbols*. Let Σ be an alphabet. We call each element of Σ^* a *string (over Σ)*, and instead of $(a_1, \ldots, a_n)$, we denote a string also by $a_1 \cdots a_n$. A *(string) language (over Σ)* is a subset $L \subseteq \Sigma^*$. We extend the concatenation operation to string languages by letting $L_1 \cdot L_2 = \{w_1 w_2 \mid w_1 \in L_1, w_2 \in L_2\}$. Note that $L \cdot \emptyset = \emptyset = \emptyset \cdot L$.

2.1.3 Orders and equivalence relations

A *binary relation R on A* is a relation from A into A. Let R be a binary relation on A. Let $A' \subseteq A$ and $a' \in A'$. Then *a' is an R-minimal element in A'* if aRa' implies $a \notin A'$ for every $a \in A$. We say that R is

- *reflexive* if $id_A \subseteq R$.
- *symmetric* if $R \subseteq R^{-1}$.
- *transitive* if $R; R \subseteq R$.
- *antisymmetric* if $R \cap R^{-1} \subseteq id_A$.
- *well founded* if every nonempty subset of A has an R-minimal element.
- a *(partial) order on A* if it is reflexive, antisymmetric, and transitive.
- an *equivalence relation on A* if it is reflexive, symmetric, and transitive.

We usually denote orders by variants of $\leq$ or $\sqsubseteq$, their inverses by $\geq$ or $\sqsupseteq$, respectively, and equivalence relations by variants of $\sim$ or $\equiv$. We will often use that the usual order $\leq$ on $\mathbb{N}$, i.e., $0 \leq 1 \leq 2 \leq \cdots$, is well founded.

Let $\leq$ be an order on A. Note that $\geq$ is an order on A as well. Two elements $a, b \in A$ are *comparable (by $\leq$)* if $a \leq b$ or $b \leq a$. Let $A' \subseteq A$. Then A' is called a *chain* if its elements are pairwise comparable. An element $a \in A$ is called *upper bound* of A' if $a' \leq a$ for every $a' \in A'$. An element $a \in A'$ is called *least element (in A')* if $a \leq a'$ for every $a' \in A'$. Note that each set has at most one least element. If the set of upper bounds of A' contains a least element a, then a is called the *supremum* $\sup A'$ *of* A'. The notions *lower bound* and *greatest element* are defined dually, with $\geq$ in place of $\leq$. If the set of lower bounds of A' contains a greatest element a, then a is called the *infimum* $\inf A'$ *of* A'. The order $\leq$ is *linear* or *total* if A is a chain. If $\leq$ is linear, the notions "least element" and "$\leq$-minimal element" coincide, as well as "greatest element" and "$\geq$-minimal element". The least element of a set A', if it exists, is denoted by $\min A'$. Likewise, the greatest element is denoted by $\max A'$. An *ω-chain* a is a sequence $a \in A^{\mathbb{N}}$ such that $a_i \leq a_{i+1}$ for every $i \in \mathbb{N}$. Let $a \in A^{\mathbb{N}}$. Recall that $a(\mathbb{N}) = \{a_i \mid i \in \mathbb{N}\}$. Instead of "upper bound of $a(\mathbb{N})$" and "supremum of $a(\mathbb{N})$", we say "upper bound of a" and "supremum of a", respectively. The order $\leq$ is *ω-complete* if A has a least element $\bot$ and every ω-chain has a supremum.

A *(partially) ordered set (poset)* is a pair $(A, \leq)$ where A is a set and $\leq$ is an order on A. A poset $(A, \leq)$ is a *linear*, *total*, or *ω-complete* poset if $\leq$ is *linear*, *total*, or *ω-complete*, respectively. We often identify $(A, \leq)$ and A. Let A and B be posets and $f\colon A \to B$. Then f is *monotone* if $a \leq a'$ implies $f(a) \leq f(a')$. Recall that $f \circ a = (f(a_i) \mid i \in \mathbb{N})$ for every $a \in A^{\mathbb{N}}$. Let A and B be ω-complete. The mapping f is *ω-continuous* if, for every ω-chain a, $\sup(f \circ a)$ is defined and $f(\sup a) = \sup(f \circ a)$. If f is ω-continuous, then it is monotone, for if $a \leq a'$, then $\sup\{f(a), f(a')\} = f(\sup\{a, a'\}) = f(a')$ and, hence, $f(a) \leq f(a')$. Consequently, $f \circ a$ is an ω-chain if a is an ω-chain. We observe that the composition of ω-continuous mappings is again ω-continuous.

The following theorem is sometimes called *fixpoint theorem*.

Theorem 2.1.1 ([115, Thm 3.1], [179, Sec. 1.5.2, Thm. 7]) *Let A be an ω-complete poset with least element $\bot$ and $f\colon A \to A$ an ω-continuous mapping. Then $(f^i(\bot) \mid i \in \mathbb{N})$ is an ω-chain and it has a least upper bound, which is the least fixpoint of f; i.e.,*

$$\min\{a \mid f(a) = a\} = \sup\{f^i(\bot) \mid i \in \mathbb{N}\}\,.$$

Let $\leq$ be an order on A and I a set. We extend $\leq$ *pointwise* to A^I by letting $a \leq a'$ if $a_i \leq a'_i$ for every $a, a' \in A^I$ and $i \in I$. Here we understand the word extend in

the same way as for mappings, in contrast to other established notions of extending an ordering that refer to adding pairs to the relation. If the order on A is ω-complete, then so is the extended order. We define the *lexicographic order* on $\mathbb{N}^*$, often denoted by $\leq$, by letting $w_1 \leq w_2$ if

1. $w_1 = \varepsilon$ or
2. there are $i_1, i_2 \in \mathbb{N}$ and $w_1', w_2' \in \mathbb{N}^*$ such that $w_1 = i_1 \cdot w_1'$, $w_2 = i_2 \cdot w_2'$, $i_1 \leq i_2$, and $i_1 = i_2$ implies $w_1' \leq w_2'$.

Let $\equiv$ be an equivalence relation on A. For every $a \in A$, we define the *equivalence class* $[a]_\equiv$ *represented by* a by letting $[a]_\equiv = \{a' \mid a \equiv a'\}$; and the *quotient set* $A/_\equiv$ *of* A *modulo* $\equiv$ is the partition of A defined by letting $A/_\equiv = \{[a]_\equiv \mid a \in A\}$. Conversely, every partition B of A gives rise to the equivalence relation on A that relates two elements precisely when they belong to the same block. If $B = A/_\equiv$, then this equivalence relation is again $\equiv$.

2.1.4 Bibliographic remarks

Our definitions of ω-complete order and ω-continuous mapping mainly follow [115, Sec. 2]. Other definitions are established as well, cf. [179, Sec. 1.5]. The latter author uses countable chains instead of ω-chains in his definitions, which is equivalent because every ω-chain has a supremum precisely when every countable chain has a supremum [179, Sec. 1.5, Prop. 3]. Moreover, his notion of ω-continuous also applies in the case that A and B are not ω-complete; correspondingly, he then only considers ω-chains a that have a supremum.

2.2 Trees

2.2.1 Unranked trees

Let Σ be an alphabet and V a set. We write $T_\Sigma(V)$ for the set of all well-formed expressions over Σ with variables V, i.e., the smallest set T such that (i) $V \subseteq T$ and (ii) for every $\sigma \in \Sigma$, $k \geq 0$, and $t_1, \ldots, t_k \in T$, we have $\sigma(t_1, \ldots, t_k) \in T$. Alternatively, we view $T_\Sigma(V)$ as the set of all (rooted, labeled, ordered, unranked) *trees over* Σ *indexed by* V, and draw them as usual. By T_Σ we abbreviate $T_\Sigma(\emptyset)$. We will often denote the tree $\sigma()$ just by σ. A *tree language over* Σ is a subset of T_Σ.

Let $t \in T_\Sigma(V)$. Next we define the *set* $\mathrm{pos}(t)$ *of positions of* t, the *height* $\mathrm{ht}(t)$ *of* t, the *rank* $\mathrm{rk}_t(w)$ *of the position* w *in* t, the *label* $t(w)$ *of* t *at* w, the *subtree* $t|_w$ *of* t *at* w, and the *tree* $t[t']_w$ *obtained from* t *by replacing the subtree at* w *by* t' [7, Def. 3.1.3].

To this end, we define two mappings $\mathrm{pos}\colon T_\Sigma(V) \to \mathcal{P}(\mathbb{N}^*)$ and $\mathrm{ht}\colon T_\Sigma(V) \to \mathbb{N}$; for every $t \in T_\Sigma(V)$, three mappings $\mathrm{rk}_t\colon \mathrm{pos}(t) \to \mathbb{N}$, $t(.)\colon \mathrm{pos}(t) \to \Sigma \cup V$, and $t|_.\colon \mathrm{pos}(t) \to T_\Sigma(V)$; and for every $t, t' \in T_\Sigma(V)$, the mapping $t[t']_.\colon \mathrm{pos}(t) \to T_\Sigma(V)$, by induction as follows. For every $v \in V$, we let

$$\mathrm{pos}(v) = \{\varepsilon\}\,, \quad \mathrm{ht}(v) = 1\,, \quad \mathrm{rk}_v(\varepsilon) = 0\,, \quad v(\varepsilon) = v\,, \quad v|_\varepsilon = v\,, \quad v[t']_\varepsilon = t'\,,$$

and for every $\sigma(t_1, \ldots, t_k) \in T_\Sigma(V)$, $i \in \{1, \ldots, k\}$, and $w \in \mathrm{pos}(t_i)$, we let

$$\begin{aligned}
\mathrm{pos}(\sigma(t_1, \ldots, t_k)) &= \{\varepsilon\} \cup \{iw \mid i \in \{1, \ldots, k\}, w \in \mathrm{pos}(t_i)\}\,,\\
\mathrm{ht}(\sigma(t_1, \ldots, t_k)) &= 1 + \max\{\mathrm{ht}(t_i) \mid i \in \{1, \ldots, k\}\}\,,\\
\mathrm{rk}_{\sigma(t_1,\ldots,t_k)}(\varepsilon) &= k\,, & \mathrm{rk}_{\sigma(t_1,\ldots,t_k)}(iw) &= \mathrm{rk}_{t_i}(w)\,,\\
\sigma(t_1, \ldots, t_k)(\varepsilon) &= \sigma\,, & \sigma(t_1, \ldots, t_k)(iw) &= t_i(w)\,,\\
\sigma(t_1, \ldots, t_k)|_\varepsilon &= \sigma(t_1, \ldots, t_k)\,, & \sigma(t_1, \ldots, t_k)|_{iw} &= t_i|_w\,,\\
\sigma(t_1, \ldots, t_k)[t']_\varepsilon &= t'\,, & \sigma(t_1, \ldots, t_k)[t']_{iw} &= \sigma(t'_1, \ldots, t'_k)\,,
\end{aligned}$$

where $t'_i = t_i[t']_w$, $t'_j = t_j$ for $j \neq i$, and we assume that $\max \emptyset = 0$. Sometimes we use the word *node* instead of position. The tree t is *binary* if $\mathrm{rk}_t(w) \leq 2$ for every $w \in \mathrm{pos}(t)$; and it is *suprabinary* otherwise. For each pair w_1, w_2 of positions, we say that w_1 *is above* w_2 if w_1 is a prefix of w_2, i.e., there is a $w \in \mathbb{N}^*$ with $w_2 = w_1 \cdot w$. Likewise, w_1 *is strictly above* w_2 if w_1 is above w_2 and $w_1 \neq w_2$.

Let $V' \subseteq \Sigma \cup V$. We say that t *is linear (nondeleting) in* V' if every element of V' occurs at most once (at least once) in t. Moreover, let W be a set and $t' \in T_\Sigma(W)$. We say that t *is a* V'*-prefix of* t' if there is a mapping κ from $\{w \mid t(w) \in V'\}$ into $T_\Sigma(W)$ such that t' is obtained from t by replacing the subtree at each $w \in \mathrm{dom}\,\kappa$ by $\kappa(w)$. If $V' = V$, we omit the reference to V', simply speaking of linear, nondeleting, and a prefix. We denote the set of all linear trees over Σ indexed by V by $T^{lin}_\Sigma(V)$, and we denote the set of all linear nondeleting trees over Σ indexed by V by $C_\Sigma(V)$. By C_Σ we abbreviate $C_\Sigma(\{z\})$, where z is a special symbol that does not occur in Σ. We call each element of C_Σ a *context (over* Σ*)*.

2.2.2 Substitution

Let $X = \{x_1, x_2, \ldots\}$ and $Y = \{y_1, y_2, \ldots\}$ be disjoint sets, whose elements we call *variables*. We let $X_k = \{x_1, \ldots, x_k\}$ and $Y_k = \{y_1, \ldots, y_k\}$ for every $k \geq 0$.

Let $V' \subseteq \Sigma \cup V \cup X \cup Y$ and $f\colon V' \to T_\Sigma(V)$. Then we define the mappings $f^\flat, f^{\flat\flat}\colon T_\Sigma(V) \to T_\Sigma(V)$, called *first-order substitution* and *second-order substitution*,

respectively, as follows. For every $v \in V$, we let

$$f^\flat(v) = f^{\flat\flat}(v) = \begin{cases} f(v) & \text{if } v \in V', \\ v & \text{if } v \notin V'. \end{cases}$$

For every $\sigma(t_1, \ldots, t_k) \in T_\Sigma(V)$, we let

$$f^\flat(\sigma(t_1, \ldots, t_k)) = \begin{cases} f(\sigma) & \text{if } \sigma \in V', \\ \sigma(f^\flat(t_1), \ldots, f^\flat(t_k)) & \text{if } \sigma \notin V'. \end{cases}$$

If $V' = \{v_1, \ldots, v_l\}$, then we also denote $f^\flat(t)$ by $t[v_1/f(v_1)] \cdots [v_l/f(v_l)]$. We let

$$f^{\flat\flat}(\sigma(t_1, \ldots, t_k)) = \begin{cases} f(\sigma)[y_1/f^{\flat\flat}(t_1)] \cdots [y_k/f^{\flat\flat}(t_k)] & \text{if } \sigma \in V', \\ \sigma(f^{\flat\flat}(t_1), \ldots, f^{\flat\flat}(t_k)) & \text{if } \sigma \notin V'. \end{cases}$$

If $V' = \{v_1, \ldots, v_l\}$, then we also denote $f^{\flat\flat}(t)$ by $t(\!|v_1/f(v_1)|\!) \cdots (\!|v_l/f(v_l)|\!)$.

Although second-order substitution is being performed in parallel, we may often imagine that we substitute the variables sequentially. This notion is made more precise in the following observation.

Observation 2.2.1 *Let $V_1, V_2 \subseteq \Sigma \cup V \cup X \cup Y$, $V_1 \cap V_2 = \emptyset$, $f_1 \colon V_1 \to T_\Sigma(V)$, $f_2 \colon V_2 \to T_\Sigma(V)$, and $f = f_1 \cup f_2$. Then $f \colon V_1 \cup V_2 \to T_\Sigma(V)$. If, for every $v_1 \in V_1$, the tree $f_1(v_1)$ does not contain occurrences of elements of V_2, then $f^{\flat\flat} = f_1^{\flat\flat}; f_2^{\flat\flat}$.*

Instead of $t[a/t']$ we also write $t' \cdot_a t$, and we omit the subscript a if $a = z$; recall that z is the special symbol that we use for contexts.

The following observation basically states that first-order substitution is "associative", e.g., we have that $t_3 \cdot (t_2 \cdot t_1) = (t_3 \cdot t_2) \cdot t_1$.

Observation 2.2.2 *Let $k, l \in \mathbb{N}$, $f \colon X_l \to T_\Delta(X)$, and $g \colon X_k \to T_\Delta(X_l)$. Then $f^\flat(g^\flat(t)) = (f^\flat \circ g)^\flat(t)$ for every $m \in \mathbb{N}$ and $t \in T_\Delta(X_k)$ with $|\mathrm{pos}(t)| \leq m$.*

2.2.3 Ranked trees

A *ranked alphabet* is a pair (Σ, rk) where Σ is an alphabet and $\mathrm{rk} \colon \Sigma \to \mathbb{N}$ assigns a natural number to each symbol, called its *arity* or *rank*. We write $\Sigma^{(k)}$ for the subset of all k-ary symbols of Σ. We denote the ranked alphabet by Σ as well. A ranked alphabet is *binary* if the arities do not exceed 2. Likewise, a symbol is *binary* if its arity is 2, and it is *suprabinary* if its rank exceeds 2. We also use $\sigma^{(k)}$ to denote that $\sigma \in \Sigma^{(k)}$, in

particular when specifying a ranked alphabet, e.g., $\Gamma = \{\alpha^{(0)}, \sigma^{(2)}\}$. We say that a tree $t \in T_\Sigma(V)$ is Σ-*ranked* if $t(w) \notin V$ implies $\mathrm{rk}_t(w) = \mathrm{rk}(t(w))$ for every $w \in \mathrm{pos}(t)$.

We will use the following convention: Σ usually denotes a "plain" alphabet (i.e., without ranks), while Γ, Δ, and their variants usually denote ranked alphabets. If Σ is a ranked alphabet, then we regard $T_\Sigma(V)$, T_Σ, $T_\Sigma^{lin}(V)$, $C_\Sigma(V)$, and C_Σ to be restricted to Σ-ranked trees. The same convention shall hold when we talk about tree languages over Σ.

2.3 Algebras and semirings

2.3.1 Algebras

Let Δ be a ranked alphabet. A Δ-*algebra* $\mathcal{A}$ is a pair $(A, .^\mathcal{A})$ where A is a nonempty set called *domain* and $.^\mathcal{A}$ maps each symbol $\delta \in \Delta$ with rank k to a k-ary operation $\delta^\mathcal{A}\colon A^k \to A$, which is also called the *realization of δ in $\mathcal{A}$*; $.^\mathcal{A}$ is called *realization mapping*. In the context of algebras, Δ is also called a (single-sorted) *signature*, and it can be viewed as an abstract data type, while a Δ-algebra can be viewed as its implementation.

Let $\mathcal{A}$ and $\mathcal{B}$ be Δ-algebras. A mapping $h\colon A \to B$ is a Δ-*homomorphism from $\mathcal{A}$ into $\mathcal{B}$* if

$$h(\delta^\mathcal{A}(a_1, \ldots, a_k)) = \delta^\mathcal{B}(h(a_1), \ldots, h(a_k))$$

holds for every k, $\delta \in \Delta^{(k)}$, and $a_1, \ldots, a_k \in A$. We write $h\colon \mathcal{A} \to \mathcal{B}$ to indicate that h is a Δ-homomorphism from $\mathcal{A}$ into $\mathcal{B}$. Note that the composition of two Δ-homomorphisms is a Δ-homomorphism. Let $h\colon \mathcal{A} \to \mathcal{B}$, $A' \subseteq A$, and $f\colon A' \to B$. If $h|_{A'} = f$, then h is *a homomorphic extension of f (with respect to $\mathcal{A}$)*.

The Δ-*term algebra $\mathcal{T}_\Delta(V)$ over V* has the domain $T_\Delta(V)$, and its operations are given by

$$\delta^{\mathcal{T}_\Delta(V)}(t_1, \ldots, t_k) = \delta(t_1, \ldots, t_k)$$

for every $k \in \mathbb{N}$, $\delta \in \Delta^{(k)}$, and $t_1, \ldots, t_k \in T_\Delta(V)$. It is well known that every mapping $f\colon V \to B$ has a unique homomorphic extension $f^\sharp$ with respect to $\mathcal{T}_\Delta(V)$ [179, Sec. 1.2, Thm. 4]; it is given by

$$f^\sharp(v) = f(v)\,, \qquad (v \in V)$$
$$f^\sharp(\delta(t_1, \ldots, t_k)) = \delta^\mathcal{B}(f^\sharp(t_1), \ldots, f^\sharp(t_k))\,. \qquad (\delta(t_1, \ldots, t_k) \in T_\Delta(V))$$

Let $l \in \mathbb{N}$ and $t \in T_\Delta(X_l)$. In the area of universal algebra, t is called a *term*. We define the *term function $t^\mathcal{B}\colon B^l \to B$ of t* by $t^\mathcal{B}(b_1, \ldots, b_l) = f^\sharp(t)$ with $f(x_j) = b_j$.

In particular, if $l = 0$, then we often omit the parentheses from $t^{\mathcal{B}}()$, and we view $t^{\mathcal{B}}$ as an element of B.

Observation 2.3.1 *Let $\mathcal{B}$ be a Δ-algebra, $k, l \in \mathbb{N}$, $f\colon X_k \to T_\Delta(X_l)$, and $g\colon X_l \to B$. For every $m \in \mathbb{N}$ and $t \in T_\Delta(X_k)$ we have that $|\mathrm{pos}(t)| \leq m$ implies $g^\sharp(f^\flat(t)) = (g^\sharp \circ f)^\sharp(t)$.*

Corollary 2.3.2 *Let $\mathcal{B}$ be a Δ-algebra, $k, l \in \mathbb{N}$, $g\colon X_l \to B$, $t \in T_\Delta(X_k)$, and $t_1, \ldots, t_k \in T_\Delta(X_l)$. Then $g^\sharp(t[x_1/t_1] \cdots [x_k/t_k]) = t^{\mathcal{B}}(g^\sharp(t_1), \ldots, g^\sharp(t_k))$. In particular, with $l = 0$, we have $(t[x_1/t_1] \cdots [x_k/t_k])^{\mathcal{B}} = t^{\mathcal{B}}(t_1^{\mathcal{B}}, \ldots, t_k^{\mathcal{B}})$.*

2.3.2 Semirings

A *monoid* is a Δ-algebra $\mathcal{S}$ with $\Delta = \{+^{(2)}, 0^{(0)}\}$ and carrier set S such that $+^{\mathcal{S}}$ is associative and $0^{\mathcal{S}}$ is neutral with respect to $+^{\mathcal{S}}$, i.e., (omitting the superscript $\mathcal{S}$)

$$s + 0 = s = 0 + s \; .$$

We represent $\mathcal{S}$ by the triple $(S, +^{\mathcal{S}}, 0^{\mathcal{S}})$. We call $\mathcal{S}$ *commutative* if $+^{\mathcal{S}}$ is commutative. A *monoid homorphism* is a Δ-homomorphism $h\colon \mathcal{A} \to \mathcal{B}$ such that $\mathcal{A}$ and $\mathcal{B}$ are monoids. A *semiring* [91, 87] is a Δ-algebra $\mathcal{S}$ with $\Delta = \{+^{(2)}, \cdot^{(2)}, 0^{(0)}, 1^{(0)}\}$ and carrier set S such that $(S, +^{\mathcal{S}}, 0^{\mathcal{S}})$ is a commutative monoid, called *additive monoid of $\mathcal{S}$*, $(S, \cdot^{\mathcal{S}}, 1^{\mathcal{S}})$ is a monoid, called *multiplicative monoid of $\mathcal{S}$*, and the following assertions hold (again omitting the superscript $\mathcal{S}$):

$$\begin{aligned} s_1 \cdot (s_2 + s_3) &= (s_1 \cdot s_2) + (s_1 \cdot s_3) \; , && (\cdot \text{ distributes over } + \text{ from the left}) \\ (s_1 + s_2) \cdot s_3 &= (s_1 \cdot s_3) + (s_2 \cdot s_3) \; , && (\cdot \text{ distributes over } + \text{ from the right}) \\ s_1 \cdot 0 &= 0 = 0 \cdot s_1 \; . && (\text{absorbing element of } \cdot) \end{aligned}$$

We represent $\mathcal{S}$ by the quintuple $(S, +^{\mathcal{S}}, \cdot^{\mathcal{S}}, 0^{\mathcal{S}}, 1^{\mathcal{S}})$. The operations $+^{\mathcal{S}}$ and $\cdot^{\mathcal{S}}$ are called the *addition* and the *multiplication* of $\mathcal{S}$, respectively. A *semiring homomorphism* is a Δ-homomorphism $h\colon \mathcal{A} \to \mathcal{B}$ such that $\mathcal{A}$ and $\mathcal{B}$ are semirings.

Let $\mathcal{S} = (S, +, \cdot, 0, 1)$ be a semiring. We define seven properties of $\mathcal{S}$ as follows.

- It is *commutative* if $\cdot$ is commutative.
- It is *zero-divisor free* if $s_1 \cdot s_2 = 0$ implies that $s_1 = 0$ or $s_2 = 0$.
- It is *zero-sum free* if $s_1 + s_2 = 0$ implies that $s_1 = 0 = s_2$.
- It is a *semifield* if it is commutative and it admits multiplicative inverses, i.e., for every $s \in S \setminus \{0\}$ there is a uniquely determined $s^{-1} \in S$ such that $s \cdot s^{-1} = 1$.

- It is *locally finite* if for every finite subset $S' \subseteq S$ the closure of S' under 0, 1, $+$, and $\cdot$ is finite; said closure is the smallest superset S'' of S' such that $0, 1 \in S''$ and $s_1, s_2 \in S''$ implies $s_1 + s_2, s_1 \cdot s_2 \in S''$.
- It is *extremal* if $s_1 + s_2 \in \{s_1, s_2\}$ for every $s_1, s_2 \in S$.
- It is *naturally ordered* if $(S, \leq)$ is an ordered set, where the binary relation $\leq$ on S is defined by $s_1 \leq s_2$ if there is an $s \in S$ with $s_1 + s = s_2$.

Example 2.3.3 We consider seven examples of semirings. To this end, let $\mathbb{R}_\infty^{\geq 0}$ denote the set of nonnegative reals extended by ∞ and let $\bar{\mathbb{R}} = \mathbb{R} \cup \{\infty, -\infty\}$.

1. The semiring $\mathrm{Real} = (\mathbb{R}_\infty^{\geq 0}, +, \cdot, 0, 1)$, where $\infty + r = \infty = r + \infty$ for every r, and $\infty \cdot r = \infty = r \cdot \infty$ for every r with $r \neq 0$;
2. the *arctic semiring* $\mathrm{Arct} = (\bar{\mathbb{R}}, \max, +, -\infty, 0)$ where $\max(\infty, r) = \infty = \max(r, \infty)$ for every r, and $\infty + r = \infty = r + \infty$ for every r with $r \neq -\infty$;
3. the *tropical semiring* $(\mathbb{R}_\infty^{\geq 0}, \min, +, \infty, 0)$;
4. the *Viterbi semiring* $([0, 1], \max, \cdot, 0, 1)$;
5. the *Boolean semiring* $(\mathbb{B}, \vee, \wedge, 0, 1)$ where $\mathbb{B} = \{0, 1\}$, and $\vee$ and $\wedge$ denote disjunction and conjunction, respectively;
6. the semifield $(\mathbb{R}^{\geq 0}, +, \cdot, 0, 1)$ of nonnegative real numbers;
7. the *formal-language semiring* $(\mathcal{P}(\Sigma^*), \cup, \cdot, \emptyset, \{\varepsilon\})$ over an alphabet Σ.

Semirings 1–7 are naturally ordered, zero-sum free, and zero-divisor free; 1–6 are commutative; 2–5 are extremal; and 5 is locally finite. □

Example 2.3.4 The mappings $\log$ and $\exp$ are monoid homomorphism from the multiplicative monoid of Real into the multiplicative monoid of Arct, and vice versa, respectively. □

Let I be a set. Then we refer to the elements in S^I also as *I-vectors over S*. For every $s \in S$ and $u \in S^I$, we define $s \cdot u \in S^I$ by $(s \cdot u)_i = s \cdot u_i$. Here and in general, we use family notation for vectors, i.e., u_i instead of $u(i)$. Moreover, $\mathcal{S}^I = (S^I, +, \odot, \tilde{0}, \tilde{1})$ is a semiring, where $\tilde{s}_i = s$, the operations $+$ and $\cdot$ are extended to S^I pointwise, i.e., $(u_1 + u_2)_i = (u_1)_i + (u_2)_i$ and $(u_1 \odot u_2)_i = (u_1)_i \cdot (u_2)_i$, and $\odot$ is called *Hadamard product*. If $\mathcal{S}$ is commutative (or zero-sum free, or extremal), then so is $\mathcal{S}^I$. However, $\mathcal{S}^I$ need not be zero-divisor free (or a semifield), even if $\mathcal{S}$ is zero-divisor free (or a semifield, respectively). Let $d \in \mathbb{N}$, $d \geq 1$. If $I = \{1, \ldots, d\}$, then we write $\mathcal{S}^d$ for $\mathcal{S}^I$.

Example 2.3.5 (Ex. 2.3.3 contd.) Let $d \in \mathbb{N}$, $d > 1$. We consider two semirings:

8. $\mathrm{Real}^d = ((\mathbb{R}_\infty^{\geq 0})^d, +, \odot, \tilde{0}, \tilde{1})$ and
9. $((\mathbb{R}_\infty^{\geq 0})^d, \min, \oplus, \tilde{\infty}, \tilde{0})$.

In contrast to Semiring 1, Semiring 8 is not zero-divisor free because

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \odot \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} . \qquad \square$$

Let Σ be an alphabet. A *weighted tree language (over Σ and $\mathcal{S}$)* is a mapping $\varphi\colon T_\Sigma \to S$. A *weighted tree transformation (over Σ and $\mathcal{S}$)* is a mapping $\tau\colon T_\Sigma \times T_\Sigma \to S$. If Σ is even a ranked alphabet, then T_Σ in this definition is understood to be restricted to Σ-ranked trees. Note that a weighted tree language over Σ and $\mathcal{S}$ is a T_Σ-vector over S; consequently, the Hadamard product applies to weighted tree languages over Σ and $\mathcal{S}$. Similar reasoning applies to weighted tree transformations.

2.3.3 Complete semirings

Now we turn to the problem of computing infinite sums in a semiring. We call $\mathcal{S}$ *complete* if it has an operation $\sum_I\colon S^I \to I$ for every index set I such that the following conditions are satisfied [115, Sec. 2]:

(i) $\sum_{i\in\emptyset} s_i = 0$, $\sum_{i\in\{j\}} s_i = s_j$, $\sum_{i\in\{j,k\}} s_i = s_j + s_k$ for $j \neq k$,

(ii) $\sum_{j\in J} \sum_{i\in I_j} s_i = \sum_{i\in I} s_i$ if $\bigcup_{j\in J} I_j = I$ and $I_j \cap I_{j'} = \emptyset$ for $j \neq j'$,

(iii) $\sum_{i\in I} s \cdot s_i = s \cdot \sum_{i\in I} s_i$, $\sum_{i\in I} s_i \cdot s = \left(\sum_{i\in I} s_i\right) \cdot s$.

Then we call $\sum_I$ *infinitary sum operation*. Roughly, the three conditions mean that (i) the infinitary sum extends the finite sum, (ii) it is associative and commutative, and (iii) it satisfies the distributivity laws. A semiring homomorphism from a complete semiring into a complete semiring is *complete* if it also preserves the infinite sums.

Let J be a set. If $\mathcal{S}$ is complete, then so is $\mathcal{S}^J$, where $(\sum_{i\in I} u_i)_j = \sum_{i\in I} (u_i)_j$.

The semiring $\mathcal{S}$ is *ω-continuous* if it is complete, naturally ordered, and

$$\forall n\colon \sum_{i\in\{0,\ldots,n\}} a_i \leq c \implies \sum_{i\in\mathbb{N}} a_i \leq c$$

for every $a \in A^{\mathbb{N}}$ and $c \in \mathbb{N}$.

Example 2.3.6 (Ex. 2.3.5 contd.) Semirings 1–5 and 7–9 are ω-continuous; following [115, Ex. 2.2], the infinite sums are defined by

$$\textstyle\sum_{i\in I} s_i = \sup\{\sum_{i\in E} s_i \mid E \subseteq I, E \text{ finite}\} \ .$$

It can be desirable to have both an infinitary sum operation (as in Semiring 1, but not in Semiring 6) and multiplicative inverses (as in Semiring 6, but not in Semiring 1). However, the only element in Semiring 1 that lacks a multiplicative inverse is ∞. Consequently, as long as we avoid "∞^{-1}", we may utilize Semiring 1. □

Theorem 2.3.7 ([115, Thm. 2.3]) *Let $\mathcal{S}$ be ω-continuous. Then, for every $s \in S^{\mathbb{N}}$,*

$$\sup\{\textstyle\sum_{i\in\{0,\dots,n\}} s_i \mid n \in \mathbb{N}\} = \sum_{i\in\mathbb{N}} s_i \ .$$

Theorem 2.3.8 ([115, Thm. 3.2, Thm. 3.3]) *Let $\mathcal{S}$ be ω-continuous. Then S is an ω-complete poset and addition and multiplication are ω-continuous.*

2.4 Weighted tree automata

Let Σ be an alphabet and $\mathcal{S}$ a semiring. A weighted tree automaton [71] over Σ and $\mathcal{S}$ is a finite-state machine that represents a weighted tree language. It assigns a weight to every tree based on weighted transitions. The following formal definitions deviate a little from the literature; the interested reader will find more about the deviations and the rationale behind them at the end of this section.

2.4.1 Syntax

Formally, a *weighted tree automaton M over Σ and $\mathcal{S}$*, for short: *WTA (over Σ and $\mathcal{S}$)*, is a tuple (Q, R, μ, ν) where

- Q is a nonempty, finite set (of *states*),
- $R \subseteq Q^* \times \Sigma \times Q$ is a finite set (of *transitions* or *(transition) rules*),
- $\mu\colon R \to S$ is the *weight assignment*, and
- $\nu\colon Q \to S$ is the *root-weight mapping*.

In the following, let $M = (Q, R, \mu, \nu)$ be a WTA over Σ and $\mathcal{S}$. For a transition $(q_1 \cdots q_k, \sigma, q)$, we call σ its *terminal symbol* and k its *rank*, so that R can be viewed as a ranked alphabet. If Σ is a ranked alphabet, then we require that the rank of any transition coincide with the rank of its terminal symbol. For every $q \in Q$, we denote by $R|_q$ the set of all transitions whose third component is q.

We define four properties of M as follows:

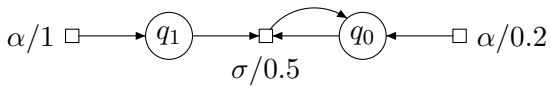


Figure 2.1: Visualization of the WTA of Ex. 2.4.1.

- It is *classical* if Σ is a ranked alphabet and $R = \bigcup_k Q^k \times \Sigma^{(k)} \times Q$. Then we denote M by (Q, μ, ν).
- It is *bottom-up deterministic* (*bu-det*) if the set

  $$\{q \mid (q_1 \cdots q_k, \sigma, q) \in R, \mu(q_1 \cdots q_k, \sigma, q) \neq 0\}$$

  has at most one element for every $\sigma \in \Sigma$, $k \in \mathbb{N}$, and $q_1, \ldots, q_k \in Q$.
- It is a *(finite) tree automaton (FTA)* if $\mathcal{S}$ is the Boolean semiring and $\mu(\rho) = 1$ for every $\rho \in R$. Then we denote M by (Q, R, F) where $F = \nu^{-1}(1)$.
- It is in *root-state form* if there is a $q_0 \in Q$ such that $\nu_{q_0} = 1$ and $\nu_q = 0$ for $q \neq q_0$. In such a case, we call q_0 the *root state* of M, and we denote M by (Q, R, μ, q_0) or by (Q, R, q_0) if it is an FTA.

We note that a (weighted) tree automaton in root-state form is a (weighted) regular tree grammar over Σ and $\mathcal{S}$ in normal form [3].

Example 2.4.1 Let $\mathcal{S}$ be the Viterbi semiring $([0,1], \max, \cdot, 0, 1)$, $\Gamma = \{\sigma^{(2)}, \alpha^{(0)}\}$, and $M = (Q, R, \mu, \nu)$ the WTA over Γ and $\mathcal{S}$ where

- $Q = \{q_0, q_1\}$,
- $\nu_{q_0} = 1$, $\nu_{q_1} = 0$, and
- R and μ are given by the following list:

$$(\varepsilon, \alpha, q_1) \mapsto 1\ , \qquad (\varepsilon, \alpha, q_0) \mapsto 0.2\ , \qquad (q_1 q_0, \sigma, q_0) \mapsto 0.5\ .$$

An equivalent representation of R and μ is the hypergraph visualized in Fig. 2.1; each node in the hypergraph (drawn as circle) corresponds to a state, and each hyperedge (drawn as box with arbitrarily many ingoing arcs and exactly one outgoing arc) represents a transition. Ingoing arcs of a hyperedge are meant to be read counter-clockwise, starting from the outgoing arc.

The WTA M is not bu-det because we have that $\mu(\varepsilon, \alpha, q_1), \mu(\varepsilon, \alpha, q_0) \neq 0$. Furthermore, it is in root-state form; its root state is q_0. An equivalent representation of R and μ in the spirit of weighted regular tree grammars is the following:

$$\begin{array}{lcl} q_0 \to \sigma(q_1, q_0) & \# & 0.5 \\ q_0 \to \alpha & \# & 0.2 \\ q_1 \to \alpha & \# & 1 \end{array}$$

□

2.4.2 Semantics

Now we define the weighted tree language $[\![M]\!]$ recognized by M. To this end, we employ the approach of *run semantics*. Roughly speaking, a run d is a tree over R with the following property: if a node w is labeled $(q_1 \cdots q_k, \sigma, q)$, then the node wj is labeled by a transition in $R|_{q_j}$. We say that d is a run for the tree t that is obtained from d by projecting each label to the second component. Furthermore, the weight of d is the product of $\mu(d(w))$ over all positions $w \in \mathrm{pos}(d)$. In order to compute the weight of a tree t, we consider each run d for t, multiply its weight by the root weight $\nu(q)$ if $d(\varepsilon)$ is in $R|_q$, and sum up over all weights thus obtained.

Now we formalize the notions of a run and its weight. For our proofs, we will need runs and their weights to be as easily composable and decomposable as trees and contexts. Therefore, we will consider trees indexed by semiring elements and even Q-vectors over S. Moreover, we will consider each state a (trivial) run as well; this will enable us to speak about "partial runs".

Formally, we define the mappings

$$\begin{aligned} &\pi_Q \colon T_R(Q \cup (S^Q \times Q)) \to Q \,, && \text{(\textit{root state})} \\ &\pi_\Sigma \colon T_R(Q \cup (S^Q \times Q)) \to T_\Sigma(Q \cup S^Q) \text{ , and} && \text{(\textit{terminal tree})} \\ &\langle . \rangle_\mu \colon T_R(Q \cup S \cup (S^Q \times Q)) \to S && \text{(\textit{weight})} \end{aligned}$$

as follows. We let $\langle s \rangle_\mu = s$ for every $s \in S$. For every $q \in Q$, $(u, q) \in S^Q \times Q$, and $d \in T_R(Q \cup (S^Q \times Q))$ with $d = \rho(d_1, \ldots, d_k)$ and $\rho \in R|_q$, respectively, we let

$$\begin{array}{lll} \pi_Q(q) = q \,, & \pi_Q((u,q)) = q \,, & \pi_Q(d) = q \,, \\ \pi_\Sigma(q) = q \,, & \pi_\Sigma((u,q)) = u \,, & \pi_\Sigma(d) = \sigma(\pi_\Sigma(d_1), \ldots, \pi_\Sigma(d_k)) \,, \\ \langle q \rangle_\mu = 1 \,, & \langle (u,q) \rangle_\mu = u(q) \,, & \langle d \rangle_\mu = \langle d_1 \rangle_\mu \cdots \langle d_k \rangle_\mu \cdot \mu(\rho) \,. \end{array}$$

The set $D(M)$ of *runs of* M is the smallest subset D of $T_R(Q \cup (S^Q \times Q))$ such that $Q \cup (S^Q \times Q) \subseteq D$ and $\rho(d_1, \ldots, d_k) \in D$ for every $\rho \in R$, $\rho = (q_1 \cdots q_k, \sigma, q)$, and sequence $d_1, \ldots, d_k \in D$ with $\pi_Q(d_j) = q_j$.

Let $d \in D(M)$, $q \in Q$, and $t \in T_\Sigma(Q \cup S^Q)$. We define five properties of d.

- It is *proper* if $d \in T_R(Q)$. We denote the set of all proper runs by $D_{\text{pr}}(M)$.
- It is *complete* if $d \in T_R$. We denote the set of all complete runs by $D_{\text{co}}(M)$.
- It is *a partial run on* t if $\pi_\Sigma(d)$ is a Q-prefix of t.
- It is *a run on* t if $\pi_\Sigma(d) = t$. We denote the set of all runs on t by $D(M, t)$.
- It is a *q-run* if $\pi_Q(d) = q$. We use a superscript q to indicate that a set of runs is restricted to q-runs; this gives rise to the sets $D^q(M)$, $D^q_{\text{pr}}(M)$, $D^q_{\text{co}}(M)$, and $D^q(M, t)$.

We define the mapping $[\![.]\!]_M \colon T_\Sigma(S^Q) \to S^Q$ by

$$[\![t]\!]_M(q) = \textstyle\sum_{d \in D^q(M,t)} \langle d \rangle_\mu \ .$$

We will often omit the subscripts μ and M from $\langle . \rangle_\mu$ and $[\![.]\!]_M$, respectively. It will be clear from the context which M or μ is meant, respectively; either some WTA M is fixed throughout a section or paragraph, or we compute $\langle d \rangle$ for a run $d \in D(M)$, and then the quantification of d indicates μ.

The *(weighted) meaning* $[\![M]\!]$ *of* M is the weighted tree language over Σ and $\mathcal{S}$ with

$$[\![M]\!](t) \mapsto \textstyle\sum_{q \in Q} [\![t]\!]_q \cdot \nu_q \ .$$

A weighted tree language φ over Σ and $\mathcal{S}$ is called *recognizable* if it is the meaning of some WTA M over Σ and $\mathcal{S}$. We say that two WTA M and M' over Σ and $\mathcal{S}$ are *equivalent* if $[\![M]\!] = [\![M']\!]$. The *language* $L(M)$ *of* M is the tree language defined by

$$L(M) = \{t \mid t \in T_\Sigma, \exists q \in Q \colon D^q(M, t) \neq \emptyset, \nu_q \neq 0\} \ .$$

If M is an FTA, then $L(M) = [\![M]\!]^{-1}(1)$. A tree language L is *recognizable* if there is an FTA M such that $L = L(M)$.

Example 2.4.2 (Ex. 2.4.1 contd.) We show the mappings $[\![.]\!]_M$ and $[\![M]\!]$. For notational convenience, we will write the elements of S^Q as column vectors, where the first row is the q_1-component. By elementary computation, we obtain

$$[\![\alpha]\!] = \begin{pmatrix} 1 \\ 0.2 \end{pmatrix} \ , \qquad [\![\sigma(\alpha, \alpha)]\!] = \begin{pmatrix} 0 \\ [\![\alpha]\!]_{q_1} \cdot [\![\alpha]\!]_{q_0} \cdot 0.5 \end{pmatrix} = \begin{pmatrix} 0 \\ 0.1 \end{pmatrix} \ .$$

Now we form a general hypothesis. To this end, we define the family $(t_n \mid n \in \mathbb{N})$ of trees in T_Γ by letting $t_0 = \alpha$ and $t_{n+1} = \sigma(\alpha, t_n)$. It is easy to prove by induction on t that

$$[\![t]\!]_{q_0} = \begin{cases} 0.2 \cdot 0.5^n & \text{if } t = t_n \ , \\ 0 & \text{otherwise} \ . \end{cases}$$

By the nature of ν, we obtain $[\![M]\!](t) = [\![t]\!]_{q_0}$. □

The WTA M is *acyclic* if, for every $q \in Q$, $d \in D^q_{\mathrm{pr}}(M)$, and $w \in \mathrm{pos}(d)$, $d(w) = q$ implies $w = \varepsilon$. It is *unambiguous* if, for every $t \in T_\Sigma$, there is at most one run $d \in D(M,t)$ with $\langle d \rangle \neq 0$. We observe the following.

Observation 2.4.3 *If M is bu-det, then it is unambiguous. Consequently, for every $t \in T_\Sigma$, there is at most one $q \in Q$ with $[\![t]\!]_q \neq 0$.*

The following observation follows from the distributivity law of the semiring.

Observation 2.4.4 *Let $t \in T_\Sigma(S^Q)$ and $t = \sigma(t_1, \ldots, t_k)$. Then*

$$[\![t]\!] = [\![\sigma([\![t_1]\!], \ldots, [\![t_k]\!])]\!] \,.$$

If Σ is a ranked alphabet, then we may define the *Σ-algebra $\mathcal{M}$ associated with M* as the algebra with the carrier set S^Q and, for every $k \in \mathbb{N}$ and $\sigma \in \Sigma^{(k)}$,

$$\sigma^{\mathcal{M}}(u_1, \ldots, u_k) = [\![\sigma(u_1, \ldots, u_k)]\!] \,.$$

In the approach of *initial-algebra semantics* [86], $t^{\mathcal{M}}$ is used in lieu of $[\![t]\!]$ to define $[\![M]\!](t)$. By Obs. 2.4.4, we obtain that $[\![t]\!] = t^{\mathcal{M}}$. This means that run semantics and initial-algebra semantics coincide, which is a known fact [77, Sec. 3.2].

The first statement of the next observation follows from Obs. 2.4.4 (and vice versa), while the second statement is a direct consequence of the definition of $\langle . \rangle_\mu$.

Observation 2.4.5 *Let $t \in T_\Sigma(S^Q)$, $t' \in T_\Sigma(S^Q \cup \{z\})$, $q \in Q$, $d \in D^q(M,t)$, and $d' \in D(M, q \cdot t')$. Then*

$$[\![[\![t]\!] \cdot t']\!] = [\![t \cdot t']\!] \qquad \textit{and} \qquad \langle \langle d \rangle \cdot_q d' \rangle = \langle d \cdot_q d' \rangle \,.$$

2.4.3 Order on runs

Next we define a "prefix" order $\sqsubseteq$ on the set $D_{\mathrm{pr}}(M)$ of proper runs. We begin by illustrating the idea behind $\sqsubseteq$ by means of an example.

Example 2.4.6 (Ex. 2.4.1 contd.) Two runs d_1, d_2 are in the $\sqsubseteq$ relation if d_2 can be obtained from d_1 by simultaneously replacing arbitrarily many occurrences of states, each by a corresponding run, for instance:

$$\begin{aligned} q_0 &\sqsubseteq (q_1 q_0, \sigma, q_0)\big(q_1, q_0\big) \\ &\sqsubseteq (q_1 q_0, \sigma, q_0)\big((\varepsilon, \alpha, q_1), q_0\big) \\ &\sqsubseteq (q_1 q_0, \sigma, q_0)\big((\varepsilon, \alpha, q_1), (\varepsilon, \alpha, q_0)\big) \,. \end{aligned}$$

The set $D_{\mathrm{pr}}(M)$ and the relation $\sqsubseteq$ bear some resemblance to the set of sentential forms and the transitive, reflexive closure of the derivation relation of a context-free grammar, respectively; the main difference is that runs are more akin to abstract syntax trees than to derivation trees. □

Now we make the notion precise. First we define the family $(\wedge_q \mid q \in Q)$, where $\wedge_q$ is a binary operation on the set $D^q_{\mathrm{pr}}(M)$, inductively as follows. Let $q \in Q$ and $d, d' \in D^q_{\mathrm{pr}}(M)$. Then

$$d \wedge_q d' = \begin{cases} q & \text{if } d(\varepsilon) \neq d'(\varepsilon) \text{ or } d(\varepsilon) \notin R, \\ d(\varepsilon)\big(d|_1 \wedge_{q_1} d'|_1, \ldots, & \text{if } d(\varepsilon) = d'(\varepsilon) = (q_1 \cdots q_k, \sigma, q). \\ \qquad d|_k \wedge_{q_k} d'|_k\big) & \end{cases}$$

We define the binary relation $\sqsubseteq_p$ on the set $D^q_{\mathrm{pr}}(M)$ by letting $d \sqsubseteq_q d'$ iff $d = d \wedge_q d'$, and we define the binary relation $\sqsubseteq$ on $D_{\mathrm{pr}}(M)$ by letting $\sqsubseteq = \bigcup_{q \in Q} \sqsubseteq_q$. It is easy to verify that $\sqsubseteq$ is a partial order.

2.4.4 Properness

We call M *proper* if $\sum_{\rho \in R|_q} \mu(\rho) = 1$ for every $q \in Q$.

Lemma 2.4.7 *Let $\mathcal{S}$ be ω-continuous. Then S^Q is an ω-complete poset and there is an ω-continuous mapping $F\colon S^Q \to S^Q$ such that*

$$\textstyle\sum_{t \in T_\Sigma} [\![t]\!] = \sup\{F^n(\tilde{0}) \mid n \in \mathbb{N}\} .$$

PROOF. By Thm. 2.3.8, S is an ω-complete poset and addition and multiplication are ω-continuous. Then S^Q is again an ω-complete poset. We define the family $(T_i \mid i \in \mathbb{N})$ by letting $T_i = T_\Sigma \cap \mathrm{ht}^{-1}(i)$, and we define $F\colon S^Q \to S^Q$ by letting

$$F(u)_q = \textstyle\sum_{(q_1 \cdots q_k, \sigma, q) \in R|_q} u_{q_1} \cdots u_{q_k} \cdot \mu(q_1 \cdots q_k, \sigma, q) .$$

Since (a) the semiring operations are ω-continuous, (b) the composition of ω-continuous mappings is again ω-continuous, and (c) the supremum of a sequence of vectors can be computed pointwise, we have that F is ω-continuous as well. Moreover, using distributivity, it is straightforward to show by induction on n that

$$F^n(\tilde{0}) = \textstyle\sum_{i \in \{0,\ldots,n\}} \sum_{t \in T_i} [\![t]\!] .$$

Finally, we derive

$$\begin{aligned}\textstyle\sum_{t\in T_\Sigma} [\![t]\!] &= \textstyle\sum_{i\in\mathbb{N}} \sum_{t\in T_i} [\![t]\!] && \text{(infinitary sum operation)}\\ &= \sup\{\textstyle\sum_{i\in\{0,\dots,n\}} \sum_{t\in T_i} [\![t]\!] \mid n\in\mathbb{N}\} && \text{(Thm. 2.3.7)}\\ &= \sup\{F^n(\tilde{0}) \mid n\in\mathbb{N}\}\ . && \blacksquare\end{aligned}$$

Next, we use Lm. 2.4.7 to show that, if M is proper, then $\sum_{t\in T_\Sigma} [\![t]\!] \leq \tilde{1}$. We note that the mapping F from said lemma can be used to approximate this sum: for this, we compute $F^1(\tilde{0}), F^2(\tilde{0}), \dots$ until convergence or until some designated amount of time is up. Details about this approximation and an alternative method that converges quicker are discussed in [72].

Lemma 2.4.8 *Let M be proper and $\mathcal{S}$ ω-continuous. Then $\sum_{t\in T_\Sigma} [\![t]\!] \leq \tilde{1}$.*

PROOF. Let F be the mapping from Lm. 2.4.7. It is easy to see that $\tilde{1}$ is a fixpoint of F. We derive

$$\begin{aligned}\textstyle\sum_{t\in T_\Sigma} [\![t]\!] &= \sup\{F^n(\tilde{0}) \mid n\in\mathbb{N}\} && \text{(Lm. 2.4.7)}\\ &= \min\{u \mid u = F(u)\} && \text{(Thm. 2.1.1)}\\ &\leq \tilde{1}\ . && \text{(least fixpoint)}\\ & && \blacksquare\end{aligned}$$

2.4.5 Root-state form, trimness

The root-state form is a normal form, i.e., we have the following lemma.

Lemma 2.4.9 ([3, Prop. 3.1], [16, Lm. 6.1.1]) *For every WTA M there is a WTA M' in root-state form such that M and M' are equivalent; however, bottom-up determinism is not preserved.*

PROOF. Let $M = (Q, R, \mu, \nu)$. We let $f \notin Q$ and $M' = (Q \cup \{f\}, R \cup R', \mu', f)$ where

- $R' = \{(q_1 \cdots q_k, \sigma, f) \mid \exists q\colon (q_1 \cdots q_k, \sigma, q) \in R\}$,
- μ' coincides with μ on R, and
- $\mu'(q_1 \cdots q_k, \sigma, f) = \sum_{q\in Q} \mu(q_1 \cdots q_k, \sigma, q) \cdot \nu_q$.

Clearly, this construction does not preserve bottom-up determinism. In fact, this preservation is impossible [16, Lm. 6.1.3]. ∎

Let $M = (Q, R, \mu, \nu)$ be a WTA. A state $q \in Q$ is *reachable* if there are $q_0 \in Q$, $d \in D^{q_0}(M)$, and $w \in \mathrm{pos}(d)$ with $\nu(q_0) \neq 0$ and $d(w) = q$; and it is *productive* if $D^q_{\mathrm{co}}(M)$ is nonempty. A transition $\rho \in R$ is *useful* if there are $q_0 \in Q$, $d \in D^{q_0}_{\mathrm{co}}(M)$, and $w \in \mathrm{pos}(d)$ with $\nu(q_0) \neq 0$ and $d(w) = \rho$; otherwise ρ is *useless*. The WTA M is *trim* if every state is reachable and productive. We note that then every transition of M is useful. Converting a WTA into an equivalent trim WTA is called *trimming* or *reducing*.

Lemma 2.4.10 *Let $M = (Q, R, \mu, \nu)$ be a WTA and $L(M) \neq \emptyset$. Then there is effectively an equivalent trim WTA M' such that M' is in root-state form if so is M.*

PROOF. The reduction proceeds as known from context-free grammars. First, we determine the set of productive states as follows. We let $Q_0 = \emptyset$ and $Q_{n+1} = \{q \mid \exists (q_1 \cdots q_k, \sigma, q) \in R \colon q_1, \ldots, q_k \in Q_n\}$. Then, for every n, $Q_n \subseteq Q_{n+1}$ and $Q_n \subseteq Q$. Hence, $Q_{n+1} = Q_n$ for some n. We let $Q' = Q_n$. One can show that Q' is the set of productive states. We note that $Q' \neq \emptyset$ because $L(M) \neq \emptyset$. Second, we determine the set of reachable states of the WTA obtained from M by removing any state that is not in Q'. To this end, we let $P_0 = \emptyset$ and $P_{n+1} = \{q_0 \mid q_0 \in Q', \nu(q_0) \neq 0\} \cup \{q_i \mid \exists (q_1 \cdots q_k, \sigma, q) \in R \colon q_1, \ldots, q_k \in Q', q \in P_n, 1 \leq i \leq k\}$. Again, we find that $P_{n+1} = P_n$ for some n, and that P_n is the desired set. We construct $M' = (P_n, R', \mu|_{R'}, \nu|_{P_n})$ where R' contains exactly the transitions from R that only use states from P_n. It is easy to see that M' is trim and that $[\![M]\!] = [\![M']\!]$. ∎

2.4.6 Bibliographic remarks

There is an established theory of WTA with varying weight domains, namely fields [10], commutative semirings [3], continuous semirings [71, 116], m-monoids [116, 117, 125, 73, 175, 76], or tree-valuation monoids [56]. An overview of WTA over semirings is given in [77].

A WTA (over a semiring) from the literature is a classical WTA by our definition. When Σ is a ranked alphabet, the generative capacity of WTA and classical WTA is the same. We deviate from the literature by including the set R of transitions; this has four advantages, as we can more easily

- define a subclass of WTA by restricting the syntax of transitions; in this way, we will define weighted synchronous context-free tree grammars in Ch. 3,
- consider constructions that replace a single transition at a time; namely, the binarization procedure of Ch. 4,
- consider a notion of usability of a state that does not depend on $\mathcal{S}$, and

- handle unranked trees (with bounded rank).

Note that there is a variant of weighted tree automata that allows specifying weighted tree languages with unbounded rank [58, Sec. 3].

A potential fifth advantage of explicit transitions may emerge in the context of *training*, i.e., estimating transition weights from data. As in the case of probabilistic context-free grammars [145, Sec. 6], it is conceivable that the data naturally suggest a set of transitions, and focusing on this set may reduce the training effort. In fact, the state-splitting method of [155] may be construed as WTA training, and the method uses explicit transitions.

In the literature, a run is usually defined with respect to a tree t, namely as a mapping from the set $\mathrm{pos}(t)$ of its positions into the set Q of states. We deviate from the literature by defining a run (roughly) as a tree with labels in $R \cup Q \cup S^Q$; this has three advantages, as we can

- consider partial runs and the "prefix" order $\sqsubseteq$ on runs, which we will need in Chs. 3 and 4,
- use tree manipulation functions, from which we will benefit a great deal in Ch. 5, and
- compute the weight of a run using a homomorphism.

3 Input product and output product of a weighted synchronous context-free tree grammar and a weighted tree automaton

This chapter is a considerably expanded and revised version of [33, 32].

3.1 Introduction

Given a weighted tree tansformation τ over Σ and $\mathcal{S}$ and a weighted tree language φ over Σ and $\mathcal{S}$, the *input product* $\varphi \lhd \tau$ *of* φ *and* τ and the *output product* $\tau \rhd \varphi$ *of* φ *and* τ are the weighted tree transformations over Σ and $\mathcal{S}$ defined by [128]

$$\varphi \lhd \tau \colon (s,t) \mapsto \varphi(s) \cdot \tau(s,t) \qquad \text{and} \qquad \tau \rhd \varphi \colon (s,t) \mapsto \tau(s,t) \cdot \varphi(t) \ .$$

In the following, for the sake of brevity, we restrict our attention to the input product. The same ideas apply to the output product.

If $\mathcal{C}$ is a class of weighted tree transformations over Σ and $\mathcal{S}$, we may ask whether $\mathcal{C}$ is effectively closed under input product (with recognizable weighted tree languages over Σ and $\mathcal{S}$). In other words, if $\tau \in \mathcal{C}$ and φ is recognizable, we ask whether $\varphi \lhd \tau$ is again in $\mathcal{C}$, and how to construct it. As argued in Sec. 1.5.1, such a closure result allows for an effective algebraic decoder specification, e.g., if $\mathcal{C}$ is the class of STAG meanings.

Let us review what is known about closure under input product, with a focus on classes that contain STAG meanings. As a preparation, we relate the input product to composition of two weighted tree transformations on the one hand and to the Hadamard product of two weighted tree languages (also known as weighted intersection [145]) on the other hand. For this, we define the mapping $f \colon S^{T_\Sigma} \to S^{T_\Sigma \times T_\Sigma}$ by letting $f(\varphi)(t,t) = \varphi(t)$ and $f(\varphi)(s,t) = 0$ for $s \neq t$.

Given another weighted tree transformation τ' over Σ and $\mathcal{S}$, the *composition* $\tau \diamond \tau'$ *of* τ *and* τ' is the weighted tree transformation over Σ and $\mathcal{S}$ defined by [75, Sec. 2.6]

$$\tau \diamond \tau' \colon (s,t) \mapsto \textstyle\sum_u \tau(s,u) \cdot \tau'(u,t) \ ,$$

publication	type* F C A I	grammar formalism	product with	semiring restriction	remarks
[161]	✓✓ · ·	CFTG	FTA	Boolean	
[144]	· ✓ · ·	WLIG	WTA	nonnegative reals	
[127, Sec. 7.2]	· ✓ · ·	STAG	FTA	Boolean	(1)
[113]	· ✓✓✓	STAG	tree	Boolean	(2)
[146]	✓✓ · ·	SCFTG	FTA	Boolean	
(this chapter)	✓✓✓ ·	WSCFTG	WTA	commutative	

* Formal closure result, Construction, Algorithm, Implementation.
(1) represents STAG as XTT with explicit substitution
(2) represents STAG as IRTG

Table 3.1: Results towards closure under Hadamard/input/output product.

where we assume that $\mathcal{S}$ is complete. Then we observe that

$$\varphi \lhd \tau = f(\varphi) \diamond \tau \qquad \text{and} \qquad \tau \diamond f(\varphi) = \tau \rhd \varphi \,.$$

If the class $\mathcal{C}$ contains $f(\varphi)$ for every recognizable φ, then closure of $\mathcal{C}$ under composition implies closure of $\mathcal{C}$ under both input and output product. For instance, the class of meanings of extended multi-bottom-up tree transducers is closed under composition, whereas the class of STAG meanings is not [66]. Consequently, it is worthwile to consider the input product as a topic of its own.

Given another weighted tree language φ' over Σ and $\mathcal{S}$, we observe that

$$f(\varphi) \rhd \varphi' = f(\varphi \odot \varphi') = \varphi \lhd f(\varphi') \,;$$

i.e., the input product can be used to compute the Hadamard product. Conversely, the input product can be viewed as a simple generalization of the Hadamard product. In fact, the corresponding constructions can be highly similar; roughly speaking, the input-product construction is merely an intersection construction that accounts for an additional, yet "uninvolved" component. This intuitive observation is substantiated formally in the framework of interpreted regular tree grammars (IRTGs, cf. [112]).

Table 3.1 lists existing results with respect to the Hadamard product and the input product. The first entry is the seminal result in this area: the class of context-free tree languages is closed under intersection with regular tree languages. In [144] it is shown that the class of meanings of weighted linear indexed grammars (WLIGs) is

closed under Hadamard product with recognizable weighted tree languages. WLIGs are equivalent to tree-adjoining grammars (TAGs). It is not clear how this result can be transferred to the synchronous setting. Providing a corresponding construction, [127] and [113] indicate that the class of meanings of STAGs is closed under input and output product with recognizable tree languages and singleton tree languages, respectively. The work [146] extends this result to the class of meanings of synchronous context-free tree grammars (SCFTGs), which contains all STAG meanings.

In this chapter, we show that the class of meanings of weighted synchronous context-free tree grammars (WSCFTGs) is closed under input and output product with recognizable weighted tree languages. Moreover, we show that this closure is effective by means of a product construction, and we provide an Earley-like algorithm [60] for computing at least the useful rules of said construction. As argued in [89] and [99, Sec. 13.4], algorithms such as Earley's are often used for parsing, which is, ultimately, an application of the input product [145]. We note that WSCFTGs subsume many grammar formalisms mentioned in Ch. 1, such as STSGs, STIGs, and STAGs [102]. WSCFTGs provide additional expressive power, whose relevance to SMT has already been underscored in [146]; in particular, the authors cite recent findings concerning lexicalization of tree-adjoining grammars [131].

We proceed as follows. First, we define WSCFTGs in terms of particular WTA, and we define the WSCFTG meaning of these WTA (Sec. 3.2). Second, we prove our closure result (Sec. 3.3). More specifically, given an WSCFTG G and a WTA M, we construct a WSCFTG $M \triangleleft G$ with $[\![M \triangleleft G]\!] = [\![M]\!] \triangleleft [\![G]\!]$. We prove our closure result (Thm. 3.3.3) by showing a stronger statement: we relate the runs of G and M on the one hand to the runs of $M \triangleleft G$ on the other (Lm. 3.3.2). Roughly speaking, our result implies that, if M is unambiguous, then the n best runs of $M \triangleleft G$ correspond to the n best runs of G, when adjusted according to the input product. Third, we derive the Earley-like algorithm for computing at least the useful rules of $M \triangleleft G$, and we indicate that the algorithm is correct (Sec. 3.4).

We end this chapter with a conclusion, discussion, and outlook (Sec. 3.5).

3.2 Weighted synchronous context-free tree grammars

For the remainder of this chapter, let $\mathcal{S}$ be a commutative semiring.

Let Σ be an alphabet and $l, m, r_1, \ldots, r_l \in \mathbb{N}$. By $C_\Sigma(m, r_1, \ldots, r_l)$ we denote the set of all unranked trees t over $\Sigma \cup X_l \cup Y_m$ that are linear and nondeleting in $X_l \cup Y_m$ such that (i) $t(w) = x_j$ implies $\mathrm{rk}_t(w) = r_j$ and (ii) $t(w) = y_j$ implies $\mathrm{rk}_t(w) = 0$. Note that $C_\Sigma(m) = C_\Sigma(Y_m)$ and $C_\Sigma(0) = T_\Sigma$.

A *weighted synchronous context-free tree grammar (WSCFTG) G over Σ and $\mathcal{S}$* is a

tuple (Q, R, μ, ν) where

- Q is a ranked alphabet with $Q^{(0)} \neq \emptyset$,
- R is a finite set of triples $(q_1 \cdots q_l, \langle \zeta \zeta' \rangle, q)$ where
  - $q, q_1, \ldots, q_l \in Q$ and
  - $\zeta, \zeta' \in C_\Sigma(\mathrm{rk}(q), \mathrm{rk}(q_1), \ldots, \mathrm{rk}(q_l))$,
- $\mu \colon R \to S$, and
- $\nu \colon Q^{(0)} \to S$.

Let $G = (Q, R, \mu, \nu)$ be a WSCFTG over Σ and $\mathcal{S}$, and let Γ be the ranked alphabet with

$$\Gamma = \{\alpha^{(l)} \mid \exists q_1, \ldots, q_l, q \colon (q_1 \cdots q_l, \alpha, q) \in R\} \,.$$

Then G can be viewed as a WTA $G' = (Q, R, \mu, \nu')$ over Γ and $\mathcal{S}$ where ν' extends ν by mapping every element of $Q \setminus Q^{(0)}$ to 0. In the following, we will identify G and G'. We will also employ the notation $G = (Q, R, \mu, q_0)$ if G is in root-state form. For a transition $\rho \in R$, $\rho = (q_1 \cdots q_l, \langle \zeta \zeta' \rangle, q)$, we call ζ and ζ' the *input tree* and *output tree*, respectively.

Example 3.2.1 We consider the WSCFTG $G = (Q, R, \mu, q_1)$ over Σ and Real, where

- $\Sigma = \{\mathrm{S}, \mathrm{NP}, \mathrm{VP}, \ldots\}$,
- $Q^{(0)} = \{q_1, q_2, q_3\}$ and $Q^{(1)} = \{f\}$,
- R and μ are given in Fig. 3.1; for every transition $\rho \in R$, $\rho = (q_1 \cdots q_l, \alpha, q)$, the figure contains a line $q \to \alpha(q_1, \ldots, q_l) \;\#\; \mu(\rho)$, which is preceded by a shorthand for ρ. □

Next we will define the WSCFTG meaning of G, which is a weighted tree transformation over Σ and $\mathcal{S}$. We do so in the spirit of bimorphisms [6], where the WTA G specifies the weighted center language $[\![G]\!]$, and we define two embedded tree homomorphisms h_1 and h_2, which retrieve from a center tree the derived input tree and output tree, respectively.

Let $\zeta \in C_\Sigma(m, r_1, \ldots, r_l)$. We define the mapping $\zeta^I \colon T_\Sigma(Y)^l \to T_\Sigma(Y)$ by

$$\zeta^I(t_1, \ldots, t_l) = \zeta(\!|x_l/t_l|\!) \cdots (\!|x_1/t_1|\!) \,.$$

As mentioned in Obs. 2.2.1, we may imagine that we substitute the variables sequentially. This lets us specify the type for intermediate results, as follows.

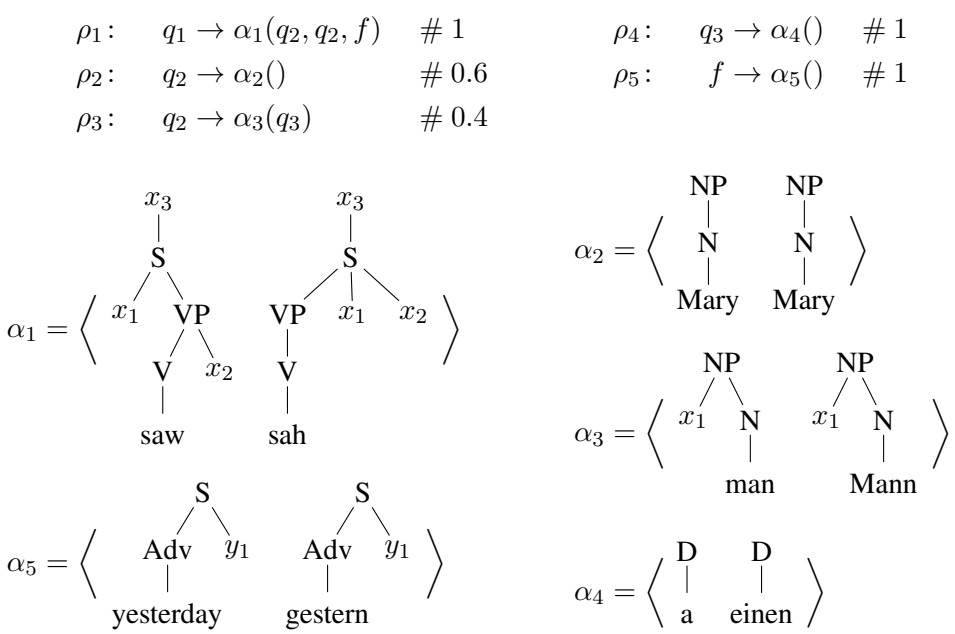


Figure 3.1: WSCFTG with initial state q_1 (adapted from [98, Fig. 2.4]).

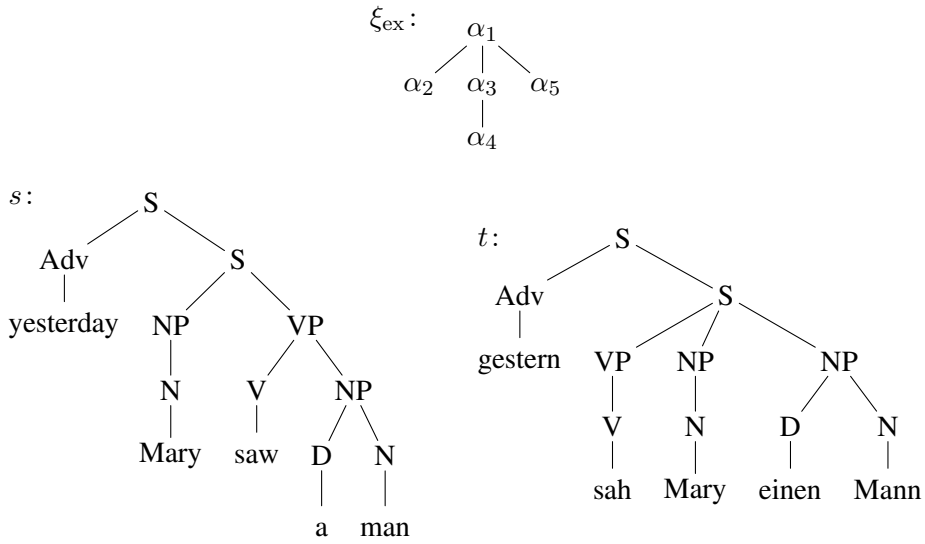


Figure 3.2: Center tree ξ_{ex}, input tree $s = h_1(\xi_{\text{ex}})$, output tree $t = h_2(\xi_{\text{ex}})$.

Observation 3.2.2 *Let $\zeta \in C_\Sigma(m, r_1, \ldots, r_l)$, $t_1 \in C_\Sigma(r_1), \ldots, t_l \in C_\Sigma(r_l)$, and $j \in \{0, \ldots, l\}$. Then $\zeta(\!|x_l/t_l|\!) \cdots (\!|x_{j+1}/t_{j+1}|\!) \in C_\Sigma(m, r_1, \ldots, r_j)$.*

We define the *embedded tree homomorphisms* $h_1, h_2 \colon T_\Gamma \to T_\Sigma(Y)$ by

$$h_i\big(\langle\zeta_1\zeta_2\rangle(\xi_1, \ldots, \xi_k)\big) = \zeta_i^I\big(h_i(\xi_1), \ldots, h_i(\xi_k)\big)$$

for every $\langle\zeta_1\zeta_2\rangle(\xi_1, \ldots, \xi_k)$ in T_Γ. We call the trees in $L(G)$ *center trees*. For every center tree ξ, we call $(h_1(\xi), h_2(\xi))$ the *derived tree pair for* ξ.

Example 3.2.3 (Ex. 3.2.1 contd.) Let ξ_{ex} and s be the trees from Fig. 3.2. We show that $h_1(\xi_{\text{ex}}) = s$ and $[\![G]\!](\xi_{\text{ex}}) = 0.24$. We begin with the former. To this end, let ζ be the input tree of ρ_1.

First, we apply the definitions of h_1 and ζ^I, and we introduce abbreviations:

$$h_1(\xi_{\text{ex}}) = \zeta(\!|x_3/\underbrace{h_1(\alpha_5)}_{t_3}|\!)(\!|x_2/\underbrace{h_1(\alpha_3(\alpha_4))}_{t_2}|\!)(\!|x_1/\underbrace{h_1(\alpha_2)}_{t_1}|\!) \,.$$

Second, we perform an auxiliary computation for t_1, t_2, and t_3; we derive

$$\begin{aligned}
t_1 &= h_1(\alpha_2) = \text{NP}(\text{N}(\text{Mary})) \,, \\
t_2 &= h_1(\alpha_3(\alpha_4)) = \text{NP}(x_1, \text{N}(\text{man}))(\!|x_1/h_1(\alpha_4)|\!) = \text{NP}(\text{D}(a), \text{N}(\text{man})) \,, \\
t_3 &= h_1(\alpha_5) = \text{S}(\text{Adv}(\text{yesterday}), y_1) \,.
\end{aligned}$$

Finally, we derive

$$\begin{aligned}
h_1(\xi_{\text{ex}}) &= x_3(\text{S}(x_1, \text{VP}(\text{V}(\text{saw}), x_2)))(\!|x_3/t_3|\!)(\!|x_2/t_2|\!)(\!|x_1/t_1|\!) \\
&= \text{S}(\text{Adv}(\text{yesterday}), \text{S}(x_1, \text{VP}(\text{V}(\text{saw}), x_2)))(\!|x_2/t_2|\!)(\!|x_1/t_1|\!) \\
&= \text{S}(\text{Adv}(\text{yesterday}), \text{S}(x_1, \text{VP}(\text{V}(\text{saw}), t_2)))(\!|x_1/t_1|\!) \\
&= \text{S}(\text{Adv}(\text{yesterday}), \text{S}(t_1, \text{VP}(\text{V}(\text{saw}), t_2))) = s \,.
\end{aligned}$$

We can show in a similar fashion that $h_2(\xi_{\text{ex}}) = t$, where t is also given in Fig. 3.2.

Now we show that $[\![G]\!](\xi_{\text{ex}}) = 0.24$. It is easy to see that

$$\rho_1(\rho_2, \rho_3(\rho_4), \rho_5)$$

is the only q_1-run on ξ_{ex}. Let us call this run d. We derive

$$\begin{aligned}
[\![G]\!](\xi_{\text{ex}}) = \langle d\rangle &= \mu(\rho_2) \cdot \mu(\rho_4) \cdot \mu(\rho_3) \cdot \mu(\rho_5) \cdot \mu(\rho_1) \\
&= 0.6 \cdot 1 \cdot 0.4 \cdot 1 \cdot 1 = 0.24 \,.
\end{aligned}$$

□

WSCFTGs are "type safe" in the following sense. Let $\zeta \in C_\Sigma(m, r_1, \ldots, r_l)$. A tree $\xi \in T_\Gamma$, $\xi = \langle\zeta\zeta'\rangle(\xi_1, \ldots, \xi_l)$, is called *type conformant* if ξ_ι is type conformant and $h_i(\xi_\iota) \in C_\Sigma(r_\iota)$. The following lemma comprises our type-safety statement.

Lemma 3.2.4 *Let $\zeta \in C_\Sigma(m, r_1, \ldots, r_l)$, $\xi \in T_\Gamma$, and $\xi = \langle\zeta\zeta'\rangle(\xi_1, \ldots, \xi_l)$. If ξ is type conformant, then $h_i(\xi) \in C_\Sigma(m)$. If $D(G, \xi) \neq \emptyset$, then ξ is type conformant.*

PROOF. The first statement follows from Obs. 3.2.2.

For the second statement, we prove the following statement by induction on n. For every $n \in \mathbb{N}$ and $\xi \in T_\Gamma$, if $|\mathrm{pos}(\xi)| \leq n$ and $D(G, \xi) \neq \emptyset$, then ξ is type conformant. For the induction base ($n = 0$), there is nothing to show. For the induction step ("$n \to n + 1$"), we let $\xi \in T_\Gamma$ with $|\mathrm{pos}(\xi)| \leq n + 1$ and $D(G, \xi) \neq \emptyset$. Then there are $\langle\zeta\zeta'\rangle \in \Gamma$, $\xi_1, \ldots, \xi_l \in T_\Gamma$, $d \in D(G, \xi)$, and $\rho = (q_1 \cdots q_l, \langle\zeta\zeta'\rangle, q)$ such that $\xi = \langle\zeta\zeta'\rangle(\xi_1, \ldots, \xi_l)$ and $d(\varepsilon) = \rho$. Then $\zeta \in C_\Sigma(\mathrm{rk}(q), \mathrm{rk}(q_1), \ldots, \mathrm{rk}(q_l))$ and the input tree ζ_ι of $\xi_\iota(\varepsilon)$ is in $C_\Sigma(\mathrm{rk}(q_\iota), r'_1, \ldots, r'_{l'})$ for some l' and $r'_1, \ldots, r'_{l'}$. By the induction hypothesis, $\xi_1, \ldots, \xi_l$ are type conformant, and by the first statement of the lemma, we obtain that $h_i(\xi_\iota) \in C_\Sigma(\mathrm{rk}(q_\iota))$. ∎

Corollary 3.2.5 *For every $\xi \in T_\Gamma$ with $[\![G]\!](\xi) \neq 0$, we have that $h_i(\xi) \in T_\Sigma$.*

PROOF. Since $[\![G]\!](\xi) \neq 0$, we have that $D^q(G, \xi) \neq \emptyset$ for some $q \in Q^{(0)}$. By Lm. 3.2.4, $h_i(\xi) \in C_\Sigma(0)$, that is, $h_i(\xi) \in T_\Sigma$. ∎

Finally we define the WSCFTG meaning of G. For this, we call G *admissible* if (i) $\mathcal{S}$ is complete or (ii) $\{\xi \mid \xi \in L(G), \forall i\colon h_i(\xi) = t_i\}$ is finite for every $(t_1, t_2) \in T_\Sigma \times T_\Sigma$. Let G be admissible. The *(WSCFTG) meaning $[\![G]\!]'$ of G* is the weighted tree transformation over Σ and $\mathcal{S}$ with

$$[\![G]\!]'(s, t) = \textstyle\sum_{\xi \in h_1^{-1}(s) \cap h_2^{-1}(t)} [\![G]\!](\xi) \ .$$

We can satisfy Condition (ii) by requiring that G be *productive*. This property has been discussed, e.g., in [74]. In our setting, it amounts to

$$\langle\zeta_1, \zeta_2\rangle \in \Gamma \implies \zeta_i \notin \{x_1, x_1(y_1)\} \ ,$$

which is easily tested for by looking at the transitions of G.

In the following, we usually omit the prime from $[\![G]\!]'$; there is no confusion with the WTA meaning because of the different type.

Example 3.2.6 (Ex. 3.2.3 contd.) Let ξ_{ex}, s, and t be the trees from Fig. 3.1. It is easy to see that $h_1^{-1}(s) \cap h_2^{-1}(t) = \{\xi_{\mathrm{ex}}\}$, and we compute

$$[\![G]\!](s, t) = [\![G]\!](\xi_{\mathrm{ex}}) = 0.24 \ .$$

As we will see in Ex. 3.3.4, we can use the input and output product to compute $[\![G]\!](s, t)$ algorithmically. □

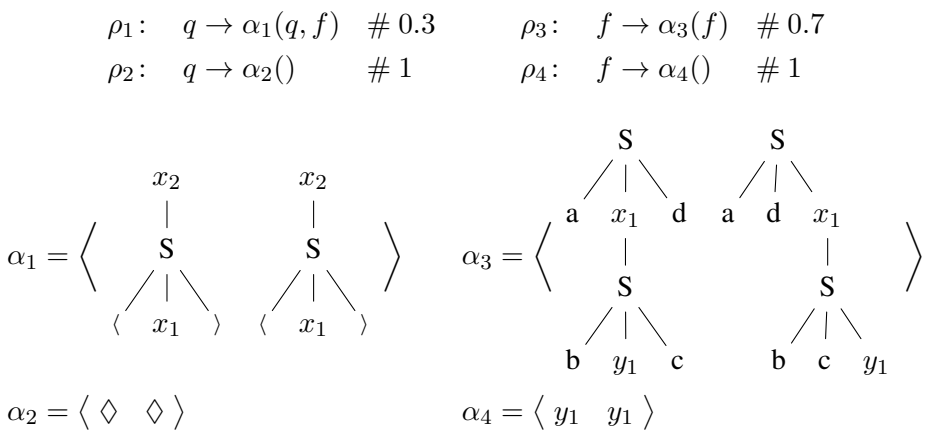


Figure 3.3: WSCFTG G from Ex. 3.3.1 (adapted from [98, Ex. 2.2]).

3.3 Closure under input and output product

3.3.1 Example and objective

Example 3.3.1 We consider the WSCFTG G given in Fig. 3.3 in the same style as in Ex. 3.2.1, where q is the root state and f is a unary state. Figure 3.4 shows the shape of the center trees of G and a concrete derived tree pair. The weight of a center tree of the given shape is $0.3^k \cdot 0.7^{n_1+\cdots+n_k}$, and the derived tree pair for such a center tree corresponds to the following pair of strings:

$$\left(\mathrm{a}^{n_1}\mathrm{b}^{n_1}\langle\cdots\langle\mathrm{a}^{n_k}\mathrm{b}^{n_k}\langle\Diamond\rangle\mathrm{c}^{n_k}\mathrm{d}^{n_k}\rangle\cdots\rangle\mathrm{c}^{n_1}\mathrm{d}^{n_1},\ (\mathrm{ad})^{n_1}(\mathrm{bc})^{n_1}\langle\cdots\langle(\mathrm{ad})^{n_k}(\mathrm{bc})^{n_k}\langle\Diamond\rangle^k\right).$$

Furthermore, we consider the WTA $M = (P, R_M, \mu_M, 0)$ with root state 0 shown in Fig. 3.5(a). The states 0 and 1 recognize backbones of even and odd lengths, respectively. Then $[\![M]\!]$ maps trees of the form shown in Fig. 3.5(b), where the unlabeled nodes may carry any label in $\{\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}, \langle, \rangle\}$, to the weight $0.5^{2n} \cdot 0.2^{4n}$ if the number of occurrences of S is $2n$. Every other tree is mapped to 0.

The input product $[\![M]\!] \lhd [\![G]\!]$ maps pairs like in Fig. 3.4(c) to $0.5^{2n} \cdot 0.2^{4n} \cdot 0.3^k \cdot 0.7^{n_1+\cdots+n_k}$ if $2n = k + 2(n_1 + \cdots + n_k)$. Every other pair is mapped to 0.

Our aim is to to construct a WSCFTG $M \lhd G$ with $[\![M \lhd G]\!] = [\![M]\!] \lhd [\![G]\!]$. For our example, such a WSCFTG is shown in Fig. 3.6. The underlying idea is to incorporate the behavior of M into G. To this end, we augment the states of G by states of M, so that we have sufficient information to simulate M on the input tree of each rule of $M \lhd G$. We note that the input tree in α_3 contains exactly two nodes labeled S, so this

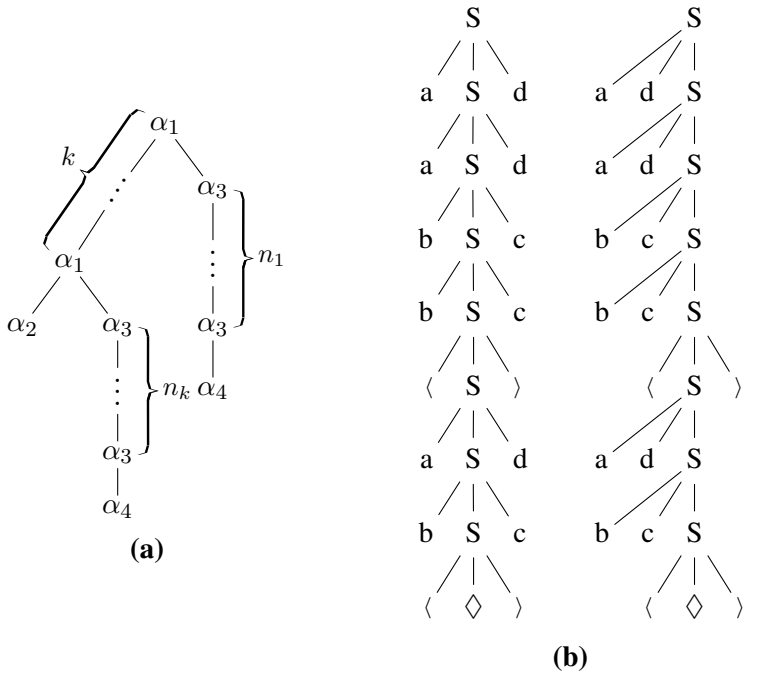
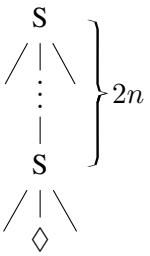


Figure 3.4: (a) Shape of center trees of G, where $k \in \mathbb{N}$ and $n_1, \ldots, n_k \in \mathbb{N}$.
(b) Derived tree pair for $k = 2$, $n_1 = 2$, and $n_2 = 1$.

$$(r1r, S, 0) \mapsto 0.5$$
$$(r0r, S, 1) \mapsto 0.5$$
$$(\varepsilon, \Diamond, 0) \mapsto 1$$
$$(\varepsilon, x, r) \mapsto 0.2 \qquad x \in \{\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}, \langle, \rangle\}$$

(a)



(b)

Figure 3.5: (a) WTA M from Ex. 3.3.1.
(b) Shape of trees with nonzero weight in $[\![M]\!]$, where $n \in \mathbb{N}$.

$$
\begin{aligned}
&(q,0,\varepsilon) \to \alpha_1\big((q,1,\varepsilon),(f,0,0)\big) && \#\ 0.3\cdot(0.5\cdot 0.2^2)\\
&(q,1,\varepsilon) \to \alpha_1\big((q,0,\varepsilon),(f,1,1)\big) && \#\ 0.3\cdot(0.5\cdot 0.2^2)\\
&(q,0,\varepsilon) \to \alpha_2() && \#\ 1.0\\
&(f,0,0) \to \alpha_3\big((f,1,1)\big) && \#\ 0.7\cdot(0.5^2\cdot 0.2^4)\\
&(f,1,1) \to \alpha_3\big((f,0,0)\big) && \#\ 0.7\cdot(0.5^2\cdot 0.2^4)\\
&(f,0,0) \to \alpha_4() && \#\ 1.0\\
&(f,1,1) \to \alpha_4() && \#\ 1.0
\end{aligned}
$$

Figure 3.6: WSCFTG $M \lhd G$ for Ex. 3.3.1.

tree does not affect the parity of the total number of S-labeled nodes. Hence, transitions of $M \lhd G$ with α_3 only contain the states $(f,0,0)$ and $(f,1,1)$, but not $(f,0,1)$ or $(f,1,0)$. Also note how these transitions alternate between said states. □

In the following, we show that the class of meanings of admissible WSCFTGs is effectively closed under input product with recognizable weighted tree languages. However, we do not prove this result directly. We rather consider a stronger statement. For this, let $M = (P, R_M, \mu_M, \nu_M)$ be a WTA over Σ and $\mathcal{S}$. Moreover, let $p \in P$, $m \in \mathbb{N}$, $s \in C_\Sigma(m)$, and $p' = (p_1, \ldots, p_m)$. Then we define

$$D^{(p,p')}(M,s) = D^p(M, s[y_1/p_1]\cdots[y_m/p_m])\ .$$

Recall that G is both an admissible WSCFTG over Σ and $\mathcal{S}$ and a WTA over Γ and $\mathcal{S}$.

Lemma 3.3.2 *There is effectively an admissible WSCFTG $M \lhd G = (Q', R', \mu', \nu')$ over Σ and $\mathcal{S}$ such that $M \lhd G$ is also a WTA over Γ and $\mathcal{S}$, $Q' = \bigcup_m Q^{(m)} \times P \times P^m$ with the ranks carried over from Q, $\nu'_{(q,p,\varepsilon)} = \nu_q \cdot (\nu_M)_p$, and the following holds. Let $\xi \in T_\Gamma$ be type conformant, $s = h_1(\xi)$, and $s \in C_\Sigma(m)$. Then there are families $(\equiv_{(p,p')} \mid p \in P, p' \in P^m)$ and $(\pi_{q'} \mid q' \in Q'^{(m)})$ such that*

- *$\equiv_{(p,p')}$ is an equivalence relation on $D^{(p,p')}(M,s)$,*
- *$\pi_{(q,p,p')}\colon D^{(q,p,p')}(M \lhd G, \xi) \to D^q(G,\xi) \times D^{(p,p')}(M,s)/_{\equiv_{(p,p')}}$ is bijective,*
- *$\pi_{q'}(d') = (d, D)$ implies $\langle d' \rangle = \langle d \rangle \cdot \sum_{e \in D} \langle e \rangle$.*

Let us consider some intuition for the lemma. As illustrated in Ex. 3.3.1, the construction of $M \lhd G$ involves "guessing" (and thus fixing) a state of M at various positions in the input tree. Roughly speaking, the equivalence relation $\equiv_{(p,p')}$ relates those

runs of M that coincide at exactly these "guessing" positions; that is, each equivalence class corresponds to one way of guessing these states. We will prove Lm. 3.3.2 in two steps. We will construct $M \lhd G$ in Sec. 3.3.2, and we will show the remaining part in Sec. 3.3.3. Now we show how the lemma implies our closure result.

Theorem 3.3.3 *Let G be an admissible WSCFTG and M a WTA, both over Σ and $\mathcal{S}$. There are admissible WSCFTGs $M \lhd G$ and $G \rhd M$ such that*

$$[\![M \lhd G]\!] = [\![M]\!] \lhd [\![G]\!] \qquad \textit{and} \qquad [\![G \rhd M]\!] = [\![G]\!] \rhd [\![M]\!] \,.$$

PROOF. Let $M = (P, R_M, \mu_M, \nu_M)$. For reasons of symmetry, we only prove the part of the theorem pertaining to $M \lhd G$. Let $M \lhd G = (Q', R', \mu', \nu')$ be the WSCFTG from Lm. 3.3.2. First, we show that $[\![M \lhd G]\!](\xi) = [\![M]\!](s) \cdot [\![G]\!](\xi)$ for every type-conformant tree $\xi \in T_\Gamma$ and $s = h_1(\xi)$. Using the families from said lemma, we derive

$$\begin{aligned}
&[\![M \lhd G]\!](\xi) = \textstyle\sum_{q' \in Q'^{(0)}} \big(\sum_{d' \in D^{q'}(M \lhd G, \xi)} \langle d' \rangle \big) \cdot \nu'_{q'} \\
&= \textstyle\sum_{q \in Q^{(0)}, p \in P} \big(\sum_{d \in D^q(G,\xi), D \in D^p(M,s)/_{\equiv_{(p,\varepsilon)}}} \langle d \rangle \cdot \sum_{e \in D} \langle e \rangle \big) \cdot \nu_q \cdot (\nu_M)_p \\
&\qquad\qquad (\pi_{q'} \text{ bijective}) \\
&= \textstyle\sum_{q \in Q^{(0)}, p \in P} \big(\sum_{d \in D^q(G,\xi), e \in D^p(M,s)} \langle d \rangle \cdot \langle e \rangle \big) \cdot \nu_q \cdot (\nu_M)_p \\
&\qquad\qquad \text{(distributivity, partition)} \\
&= \textstyle\sum_{q \in Q^{(0)}, p \in P} \big(\sum_{d \in D^q(G,\xi)} \langle d \rangle \cdot \sum_{e \in D^p(M,s)} \langle e \rangle \big) \cdot \nu_q \cdot (\nu_M)_p \qquad \text{(distributivity)} \\
&= \big(\textstyle\sum_{p \in P, e \in D^p(M,s)} \langle e \rangle \cdot (\nu_M)_p \big) \cdot \big(\sum_{q \in Q^{(0)}, d \in D^q(G,\xi)} \langle d \rangle \cdot \nu_q \big) \\
&\qquad\qquad \text{(commutativity, distributivity)} \\
&= [\![M]\!](s) \cdot [\![G]\!](\xi) \,. \qquad \text{(distributivity)}
\end{aligned}$$

Then, for every $s, t \in T_\Sigma$,

$$\begin{aligned}
[\![M \lhd G]\!](s,t) &= \textstyle\sum_{\xi \in h_1^{-1}(s) \cap h_2^{-1}(t)} [\![M \lhd G]\!](\xi) \\
&= \textstyle\sum_{\xi \in h_1^{-1}(s) \cap h_2^{-1}(t)} [\![M]\!](s) \cdot [\![G]\!](\xi) \qquad (\dagger) \\
&= [\![M]\!](s) \cdot \textstyle\sum_{\xi \in h_1^{-1}(s) \cap h_2^{-1}(t)} [\![G]\!](\xi) \qquad \text{(distributivity)} \\
&= [\![M]\!](s) \cdot [\![G]\!](s,t) = ([\![M]\!] \lhd [\![G]\!])(s,t) \,.
\end{aligned}$$

At (†), we use the statement that we derived first, together with Lm. 3.2.4. ∎

Example 3.3.4 (Ex. 3.2.3 contd.) We indicate how to use the input and output product to compute $[\![G]\!](s,t)$. We can easily construct WTA M_s and M_t over Σ and $\mathcal{S}$ such that

$$[\![M_s]\!](s) = 1 \,, \qquad [\![M_t]\!](t) = 1 \,, \qquad [\![M_s]\!](s') = 0 \,, \qquad [\![M_t]\!](t') = 0$$

for every $s' \neq s$ and $t' \neq t$. Then

$$\begin{aligned}
[\![G]\!](s,t) &= \textstyle\sum_{s',t'} ([\![M_s]\!] \lhd [\![G]\!] \rhd [\![M_t]\!])(s',t') \\
&= \textstyle\sum_{s',t'} [\![M_s \lhd G \rhd M_t]\!](s',t') && \text{(Thm. 3.3.3)} \\
&= \textstyle\sum_{\xi \in T_\Gamma} [\![M_s \lhd G \rhd M_t]\!](\xi) = \sum_{\xi \in T_\Gamma} [\![\xi]\!]_{M_s \lhd G \rhd M_t} \cdot \nu' \\
&= \big(\textstyle\sum_{\xi \in T_\Gamma} [\![\xi]\!]_{M_s \lhd G \rhd M_t}\big) \cdot \nu', && \text{(distributivity)}
\end{aligned}$$

where ν' is the root-weight mapping of $M_s \lhd G \rhd M_t$. For every WTA M over Γ and $\mathcal{S}$, we can compute $\sum_{\xi \in T_\Gamma} [\![\xi]\!]_M$ at least approximatively by using the fixpoint method that is indicated below Lm. 2.4.7. □

3.3.2 Constructing the product WSCFTG

Here we provide our construction of the WSCFTG $M \lhd G$, whose existence is postulated in Lm. 3.3.2. Let G be a WSCFTG and M a WTA, both over Σ and $\mathcal{S}$, with $G = (Q, R, \mu, \nu)$ and $M = (P, R_M, \mu_M, \nu_M)$.

First, we enrich M so that it can accept trees such as those that occur in Γ, that is, including variables. To this end, let $\zeta \in C_\Sigma(m, r_1, \ldots, r_l)$. A *(state) assignment for* ζ is a mapping θ that maps each variable x_ι that occurs in ζ to an element of $P \times P^{r_\iota}$ and each variable y_ι to an element of P. Finally, θ maps the special symbol $\diamond$ to an element of P. Then, for every assignment θ, we define the WTA $M\theta$ over $\Sigma \cup X_l \cup Y_m$ by $M\theta = (P, R_{M\theta}, \mu_{M\theta}, \theta(\diamond))$ where

- $R_{M\theta} = R_M \cup \{(p', x_\iota, p) \mid \theta(x_\iota) = (p, p')\} \cup \{(\varepsilon, y_\iota, p) \mid \theta(y_\iota) = p\}$,
- $\mu_{M\theta}(\rho) = \mu_M(\rho)$ if $\rho \in R_M$, and
- $\mu_{M\theta}(\rho) = 1$ if $\rho \notin R_M$.

Second, for every transition $\rho \in R$ and suitable θ, we let $M\theta$ "run" on the input tree of ρ. Formally, we define the *product WSCFTG* $M \lhd G$ *of* M *and* G by

$$M \lhd G = \big(\textstyle\bigcup_m Q^{(m)} \times P \times P^m, R', \mu', \nu'\big) ,$$

where the ranks of the states are carried over from Q, $\nu'_{(q,p,\varepsilon)} = \nu_q \cdot (\nu_M)_p$, and R' and μ' are defined as follows. Let $\rho \in R$, $\rho = (q_1 \cdots q_l, \langle \zeta\zeta' \rangle, q)$, $m = \mathrm{rk}(q)$, θ an assignment for ζ, and $p' = (\theta(y_1), \ldots, \theta(y_m))$. Then we let

$$\rho\theta = \Big(\big(q_1, \theta(x_1)\big) \cdots \big(q_l, \theta(x_l)\big),\ \langle \zeta\zeta' \rangle,\ \big(q, (\theta(\diamond), p')\big)\Big) ,$$

and $\rho\theta \in R'$ if $\zeta \in L(M\theta)$. Then its weight is given by

$$\mu'(\rho\theta) = \mu(\rho) \cdot [\![M\theta]\!](\zeta) \ .$$

This definition is sound because the mapping with $(\rho, \theta) \mapsto \rho\theta$ is injective. There are no further elements in R'. We note that we have identified $Q^{(m)} \times P \times P'$ (a set of triples) with $Q^{(m)} \times (P \times P')$ (a set of pairs whose second component is a pair). We will continue to do so.

We have that $|R'| \leq |R| \cdot |P|^C$ where $C = \max\{\mathrm{rk}(q_0) + \cdots + \mathrm{rk}(q_l) + l + 1 \mid (q_1 \cdots q_l, \alpha, q_0) \in R\}$. More specifically, the factors $|R|$ and $|P|^C$ are due to the choices of ρ and θ, respectively. An inspection of C suggests that, if we want to keep the cost for the input product low, then we should try to represent $[\![G]\!]$ with a WSCFTG whose states and transitions have as low rank as possible.

Example 3.3.5 (Ex. 3.3.1 contd.) In addition to the transitions shown in Fig. 3.6, the WSCFTG $M \lhd G$ also contains the following transitions:

$$\begin{aligned} (f,0,1) &\to \alpha_3\big((f,1,0)\big) \quad \# \, 0.7 \cdot (0.5^2 \cdot 0.2) \ , \\ (f,1,0) &\to \alpha_3\big((f,0,1)\big) \quad \# \, 0.7 \cdot (0.5^2 \cdot 0.2) \ . \end{aligned}$$

As argued in Ex. 3.3.1, applying ρ_3 does not affect the parity of the total number of S-labeled nodes. Hence, these transitions do not occur in any $(q, 0, \varepsilon)$-run of $M \lhd G$, and they can be discarded. □

3.3.3 Proof of Lemma 3.3.2

It is easy to see that $M \lhd G$ is admissible, because $L(M \lhd G) \subseteq L(G)$. We prove the remaining statement of the lemma by induction on the size of ξ. More precisely, we prove by induction on n that the following statement $P(n)$ holds for every n:

> $P(n)$: Let $\xi \in T_\Gamma$ be type conformant, $s = h_1(\xi)$, and $s \in C_\Sigma(m)$. If $|\mathrm{pos}(\xi)| \leq n$, then there are the families as postulated in the lemma.

For the induction base ($n = 0$), there is nothing to show. We show the induction step ($n \to n+1$). For this purpose, let $n \in \mathbb{N}$ such that $P(n)$ holds (the induction hypothesis). We show $P(n+1)$. To this end, let $\xi \in T_\Gamma$ be type conformant, $s = h_1(\xi)$, $s \in C_\Sigma(m)$, and $|\mathrm{pos}(\xi)| \leq n+1$. There are $\langle\zeta\zeta'\rangle \in \Gamma$ and $\xi_1, \ldots, \xi_l \in T_\Gamma$ such that $\xi = \langle\zeta\zeta'\rangle(\xi_1, \ldots, \xi_l)$. Clearly, ξ_ι is type conformant and $|\mathrm{pos}(\xi_\iota)| \leq n$.

By applying the induction hypothesis to $\xi_1, \ldots, \xi_l$, we obtain the families $\equiv_1$ and π_1 up to $\equiv_l$ and π_l, respectively. In the following, we will often omit the subscripts (p, p')

and q' from $\equiv_\iota$ and π_ι, respectively. Before we construct $\equiv$ and π, we introduce the following notion and lemma.

A *composition item* $\mathfrak{s}$ is a tuple $(\theta, e_0, e_1, \ldots, e_l)$ such that θ is an assignment for ζ, $e_0 \in D^{\theta(\diamond)}(M\theta, \zeta)$, and $e_\iota \in D^{\theta(x_\iota)}(M, h_1(\xi_\iota))$. The *composite* $[\![\mathfrak{s}]\!]$ *of* $\mathfrak{s}$ is defined by

$$[\![\mathfrak{s}]\!] = v'(e_0)^I(v(e_1), \ldots, v(e_l)) \,,$$

where $v(e_\iota)$ is obtained from e_ι by replacing every occurrence of a state in P by the label of $h(\xi_\iota)$ at the same position, and v' replaces every occurrence of a transition containing a variable x_ι or y_ι by the respective variable or by $\theta(y_\iota)$, respectively.

Lemma 3.3.6 *For every $p \in P$, $p' \in P^m$, and $e \in D^{(p,p')}(M, s)$, there is exactly one composition item $\mathfrak{s}$ with $[\![\mathfrak{s}]\!] = e$.*

PROOF. We prove the following statement by induction on l. For every $l \in \mathbb{N}$, $\zeta \in C_\Sigma(m, r_1, \ldots, r_l)$, $s_1, \ldots, s_l$ with $s_\iota \in C_\Sigma(r_\iota)$, and $e \in D^{(p,p')}(M, \zeta^I(s_1, \ldots, s_l))$, there is exactly one tuple $(\theta, e_0, e_1, \ldots, e_l)$ such that θ is an assignment for ζ, $e_0 \in D^{\theta(\diamond)}(M\theta, \zeta)$, $e_\iota \in D^{\theta(x_\iota)}(M, s_\iota)$, and $e = v'(e_0)^I(v(e_1), \ldots, v(e_l))$, where v' and v are defined as before.

We prove the induction base ($l = 0$). For this, we let $w_1, \ldots, w_m$ be the positions of $y_1, \ldots, y_m$ in ζ, respectively. Then we construct $p_1, \ldots, p_m$ by letting $p_\iota = e(w_\iota)$, $e_0 = e[(\varepsilon, y_1, p_1)]_{w_1} \cdots [(\varepsilon, y_m, p_m)]_{w_m}$, $\theta(\diamond) = \pi_P(e)$, and $\theta(y_\iota) = p_\iota$. It is easy to see by the definition of v' that (θ, e_0) is the only tuple with the desired property.

For the induction step ($l \to l+1$), we let $\zeta' = \zeta(\!|x_{l+1}/s_{l+1}|\!)$. By Obs. 3.2.2, we have that $\zeta' \in C_\Sigma(m, r_1, \ldots, r_l)$ and, thus,

$$\zeta^I(s_1, \ldots, s_l, s_{l+1}) = (\zeta')^I(s_1, \ldots, s_l) \,.$$

Hence, we can apply the induction hypothesis, obtaining the tuple $(\theta', e'_0, e'_1, \ldots, e'_l)$. Let $k = r_{l+1}$, w the position of x_{l+1} in ζ and $w_1, \ldots, w_k$ the positions of $y_1, \ldots, y_k$ in s_{l+1}, respectively. Then we construct $p, p_1, \ldots, p_k$ by letting $p = \pi_P(e'_0|_w)$ and $p_j = \pi_P(e'_0|_{ww_j})$, $\rho = (p_1 \cdots p_k, x_{l+1}, p)$, $e_0 = e'_0[\rho(e'_0|_{ww_1}, \ldots, e'_0|_{ww_k})]_w$, $e_\iota = e'_\iota$ for $\iota \in \{1, \ldots, l\}$, $e_{l+1} = (e'_0[p_1]_{ww_1} \cdots [p_k]_{ww_k})|_w$, and we let θ be obtained from θ' by adding the entry $x_{l+1} \mapsto (p, p_1 \cdots p_k)$. We derive

$$\begin{aligned} e &= v'(e'_0)^I(v(e'_1), \ldots, v(e'_l)) && \text{(induction hypothesis)} \\ &= v'(e_0(\!|\rho/v(e_{l+1})|\!))^I(v(e'_1), \ldots, v(e'_l)) \\ &= v'(e_0)^I(v(e_1), \ldots, v(e_l), v(e_{l+1})) \,. \end{aligned}$$

Again, it is easy to see that $(\theta, e_0, e_1, \ldots, e_{l+1})$ is the only tuple that has the desired property. ∎

Now we define $\equiv$. Let $e_1, e_2 \in D^{(p,p')}(M, s)$. We let $e_1 \equiv_{(p,p')} e_2$ iff there are composition items $(\theta_1, e_{1,0}, e_{1,1}, \ldots, e_{1,l})$ and $(\theta_2, e_{2,0}, e_{2,1}, \ldots, e_{2,l})$ with composites e_1 and e_2, respectively, such that $\theta_1 = \theta_2$ and $e_{1,\iota}\ (\equiv_\iota)_{\theta(x_\iota)}\ e_{2,\iota}$ for every $\iota \in \{1, \ldots, l\}$.

Finally, we define π. Let $d' \in D^{(q,p,p')}(M \lhd G, \xi)$, $d' = (\rho\theta)\big(d'_1, \ldots, d'_l\big)$, and let $\pi_1(d'_1) = (d_1, D_1), \ldots, \pi_l(d'_l) = (d_l, D_l)$. Note that $D_\iota \in D^{\theta(x_\iota)}(M, h_1(\xi_\iota))/_{\equiv_\iota}$. We let $\pi_{(q,p,p')}(d') = (d, D)$ where

$$\begin{aligned} d &= \rho(d_1, \ldots, d_l)\ , \\ D &= \{[\![(\theta, e_0, e_1, \ldots, e_l)]\!] \mid e_0 \in D^{\theta(\diamond)}(M\theta, \zeta), e_\iota \in D_\iota\}\ . \end{aligned}$$

We have to show that our definition of π is sound, i.e., that $(d, D) \in D^q(G, \xi) \times D^{(p,p')}(M, s)/_{\equiv_{(p,p')}}$. It is easy to see that $d \in D^q(G, \xi)$; and we focus on D. Since $\rho\theta \in R'$, we have that there is an $e_0 \in D^{\theta(\diamond)}(M\theta, \zeta)$. By the induction hypothesis, D_ι is an equivalence class and, thus, not empty. Hence, D is not empty either, and there is an $e \in D$. It remains to show that $e' \equiv e$ iff $e' \in D$ for every $e' \in D^{(p,p')}(M, s)$. This, however, is straightforward due to the definition of $\equiv_{(p,p')}$.

Lemma 3.3.7 *For every (q, p, p'), the mapping $\pi_{(q,p,p')}$ is injective.*

PROOF. Let $q' = (q, p, p')$ and $d'_1, d'_2 \in D^{(q,p,p')}(M \lhd G, \xi)$ such that $\pi_{q'}(d'_1) = \pi_{q'}(d'_2)$. We show that $d'_1 = d'_2$. To this end, let $\pi_{q'}(d'_1) = (d_1, D_1)$ and $\pi_{q'}(d'_2) = (d_2, D_2)$, and let

$$\begin{aligned} \pi_1(d'_1|_1) = (d_{1,1}, D_{1,1}), \quad \ldots, \quad \pi_l(d'_1|_l) = (d_{1,l}, D_{1,l})\ , \\ \pi_1(d'_2|_1) = (d_{2,1}, D_{2,1}), \quad \ldots, \quad \pi_l(d'_2|_l) = (d_{2,l}, D_{2,l})\ . \end{aligned}$$

We derive

$$\rho_1(d_{1,1}, \ldots, d_{1,l}) = d_1 = d_2 = \rho_2(d_{2,1}, \ldots, d_{2,l})\ ,$$

which implies

$$\rho_1 = \rho_2\ , \tag{3.1}$$

$$d_{1,\iota} = d_{2,\iota}\ . \tag{3.2}$$

Since D_1 is an equivalence class, it is nonempty, and there is an $e \in D_1$. Since $D_1 = D_2$, we also have $e \in D_2$. By definition, there are composition items $(\theta_1, e_{1,0}, \ldots, e_{1,l})$ and $(\theta_2, e_{2,0}, \ldots, e_{2,l})$ with the same composite e. By Lm. 3.3.6, these composition items coincide and, consequently,

$$\theta_1 = \theta_2\ . \tag{3.3}$$

Furthermore, we obtain that $D_{1,\iota}$ and $D_{2,\iota}$ share an element (denoted by both $e_{1,\iota}$ and $e_{2,\iota}$). Since these sets are equivalence classes, we conclude that

$$D_{1,\iota} = D_{2,\iota} \,. \tag{3.4}$$

By the induction hypothesis, $(\pi_\iota)_{q'}$ is injective for every q'. Thus, (3.2) and (3.4) imply $d'_1|_\iota = d'_2|_\iota$. By (3.1) and (3.3) we obtain $d'_1(\varepsilon) = d'_2(\varepsilon)$. ∎

Lemma 3.3.8 *For every (q, p, p'), the mapping $\pi_{(q,p,p')}$ is surjective.*

PROOF. Let $q' = (q, p, p')$ and $(d, D) \in D^q(G, \xi) \times D^{(p,p')}(M, s)/_{\equiv_{(p,p')}}$. We construct a $d' \in D^{q'}(M \lhd G, \xi)$ such that $\pi_{q'}(d') = (d, D)$. Since D is an equivalence class, it is nonempty, and there is an $e \in D$. By Lm. 3.3.6, there is a composition item $(\theta, e_0, e_1, \ldots, e_l)$ with composite e. By D_ι we denote the element of $D^{(p_\iota,p'_\iota)}(M, h_1(\xi_\iota))/_{\equiv_{(p_\iota,p'_\iota)}}$ that contains e_ι, where p_ι and p'_ι are read off from e_ι in the obvious way, i.e., such that $e_\iota \in D^{(p_\iota,p'_\iota)}(M, h_1(\xi_\iota))$. By the induction hypothesis, $(\pi_\iota)_{q'}$ is surjective for every q', and there are $d'_1, \ldots, d'_l$ such that $\pi_\iota(d'_\iota) = (d|_\iota, D_\iota)$. We construct $d' = (d(\varepsilon)\theta)\big(d'_1, \ldots, d'_l\big)$.

We show that $\pi_{q'}(d') = (d, D)$. To this end, let $\pi_{q'}(d') = (d', D')$. Then $d = d'$ is straightforward to show, and we turn to the proof of $D = D'$. We observe that $e \in D$ (by assumption) and $e \in D'$ (by definition). Since D and D' are equivalence classes, the fact that they share an element (namely, e) implies that they are equal. ∎

Lemma 3.3.9 *Let $\pi_{(q,p,p')}(d') = (d, D)$. Then $\langle d' \rangle = \langle d \rangle \cdot \sum_{e \in D} \langle e \rangle$.*

PROOF. There are ρ and θ such that $d'(\varepsilon) = \rho\theta$. We let $\pi_1(d'|_1) = (d_1, D_1)$ up to $\pi_l(d'|_l) = (d_l, D_l)$. Then

$$\begin{aligned}
\langle d' \rangle &= \big(\textstyle\prod_\iota \langle d'|_\iota \rangle\big) \cdot \mu'(\rho\theta) \\
&= \big(\textstyle\prod_\iota \langle d_\iota \rangle \cdot \big(\sum_{e_\iota \in D_\iota} \langle e_\iota \rangle\big)\big) \cdot \mu(\rho) \cdot [\![M\theta]\!](\zeta) && \text{(Def. } \mu', \text{ induction hypothesis)} \\
&= \big(\textstyle\prod_\iota \langle d_\iota \rangle\big) \cdot \mu(\rho) \cdot [\![M\theta]\!](\zeta) \cdot \textstyle\prod_\iota \big(\sum_{e_\iota \in D_\iota} \langle e_\iota \rangle\big) && \text{(commutativity)} \\
&= \langle d \rangle \cdot \textstyle\sum_{e_0 \in D^{\theta(\diamond)}(M\theta,\zeta), e_1, \ldots, e_l \colon e_\iota \in D_\iota} \langle e_0 \rangle \cdot \textstyle\prod_\iota \langle e_\iota \rangle && \text{(distributivity)} \\
&= \langle d \rangle \cdot \textstyle\sum_{e_0 \in D^{\theta(\diamond)}(M\theta,\zeta), e_1, \ldots, e_l \colon e_\iota \in D_\iota} \langle v'(e_0)^I(v(e_1), \ldots, v(e_l)) \rangle_{\mu_M} \\
& && \text{(commutativity)} \\
&= \langle d \rangle \cdot \textstyle\sum_{e \in D} \langle e \rangle \,. && \qquad \blacksquare
\end{aligned}$$

3.4 An Earley-like algorithm for the input product

As shown in Ex. 3.3.5, the product WSCFTG $M \lhd G$ of a WTA M and a WSCFTG G may contain "useless" transitions, in the sense that they do not occur in the computation of $[\![M \lhd G]\!]$. In this section, we assume that M and G are in root-state form, and we consider a strategy for enumerating the transitions of $M \lhd G$ that attempts to avoid useless transitions. For this, we take inspiration from Earley's algorithm [60] for parsing with context-free grammars. Ultimately, this approach leads us to Alg. 3.1. Be advised that in the worst case, when $M \lhd G$ does not contain useless transitions, we still have to construct every transition.

3.4.1 Reasoning about useful transitions

It is possible to compute the set of useful transitions of $M \lhd G$: for example, we can reduce $M \lhd G$, as stated in Lm. 2.4.10; then the remaining transitions are useful. However, this procedure involves exploring the whole set of transitions of $M \lhd G$, which is exactly what we want to avoid. So we settle for an approximation, that is, we compute a superset of the set of useful transitions. For instance, we can employ the following simple observation.

Observation 3.4.1 *If a transition $(q_1 \cdots q_k, \alpha, q)$ is useful, then the states $q_1, \ldots, q_k$ and q are reachable.*

In other words, if we want to avoid computing useless transitions, then we might focus on transitions that only contain reachable states, which is reasonably simple.

In the remainder of this section, we will develop a more sophisticated approximation, which is inspired by Earley's algorithm. For this, we recall from Sec. 2.4.5 that the notion of a reachable state is defined in terms of the existence of a certain run of $M \lhd G$. We introduce the concept of a *base-item tree*, which generalizes the concept of a run. Roughly speaking, base-item trees incorporate the idea that, instead of treating a transition $\rho\theta$ of $M \lhd G$ as an atomic entity, we can construct it gradually by performing a depth-first left-to-right simulation of M on the input tree of ρ.

Example 3.4.2 (Ex. 3.3.5 contd.) Figure 3.7 shows a visualization of two base-item trees for $(q, 0)$. In general, a base-item tree δ for a pair $(q, p) \in Q \times P$ has one of four possible shapes: either it consists only of the state q, or the pair (q, p), or it is a complete (q, p, p')-run of $M \lhd G$ for some p', or its root is labeled by a base item.

A *base item* represents a partial construction of a transition of $M \lhd G$; and for its description we need the WTA M_ζ, which is obtained from M by adding all suitable transitions for the variables occurring in ζ, each with weight 1. A base item consists of

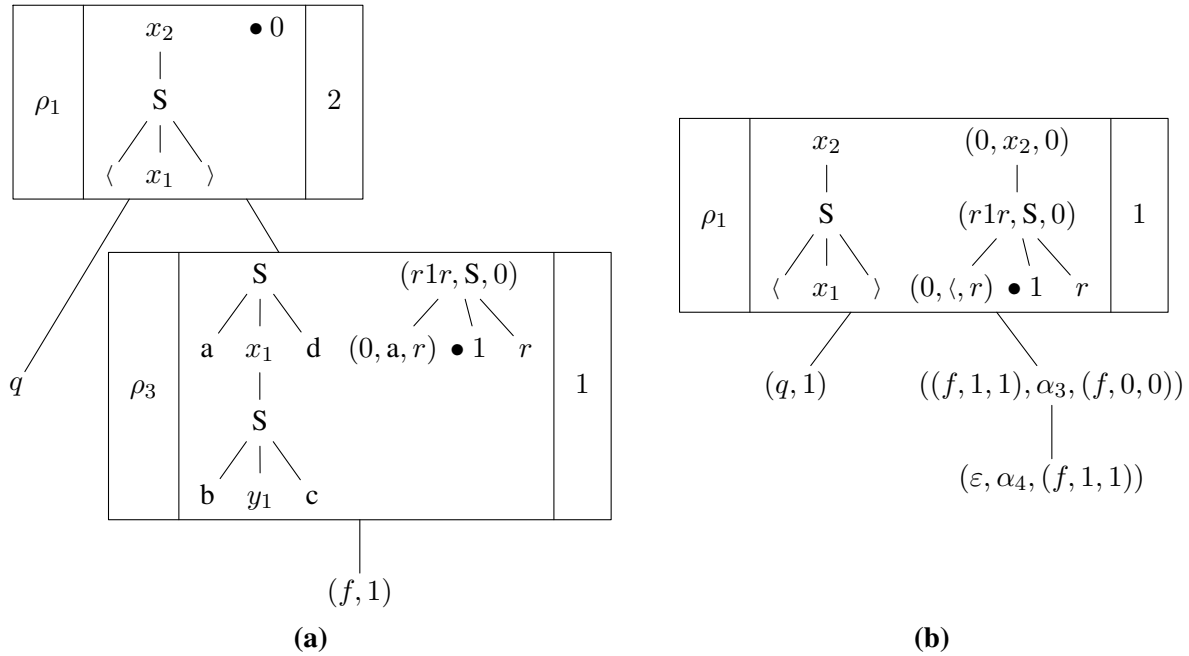


Figure 3.7: Two base-item trees of Ex. 3.4.2, where the base items are visualized as boxes.

a transition $\rho \in R|_q$, a partial p-run d of M_ζ on the input tree ζ of ρ, a "bullet position" in d, and an "active index" a. Each base item is visualized as a box whose contents are, from left to right, ρ, ζ, d, and a. The bullet position is visualized as a $\bullet$ in d. The run d is of a certain shape: all positions left of the bullet are labeled by a transition; all positions right of the bullet are labeled by a state. The active index a ranges from 0 to $\mathrm{rk}(\rho)$, and if it is positive, then the bullet is directly in front of the position labeled x_a in ζ; here we use that every position in d is also a position in ζ.

When the root of δ is labeled by a base item for some transition $(q_1 \cdots q_l, \alpha, q)$ of G, then it has l successors $\delta_1, \ldots, \delta_l$. If the active index a is positive, then δ_a is a base-item tree for (q_a, p_a) where p_a is the state in d directly behind the bullet. If $\iota \neq a$ and (p', x_ι, p) occurs in d, then δ_ι is a complete (q_ι, p, p')-run of $M \lhd G$. If $\iota \neq a$ and x_ι is right of the bullet, then δ_ι is just q_ι.

We note that the occurrences of base items in a base-item tree form a "spine": every such occurrence is either at the root or a successor of another such occurrence, and then it is the only successor that is labeled by a base item. □

Now we make the concepts of Ex. 3.4.2 precise. Let $\zeta \in C_\Sigma(m, r_1, \ldots, r_l)$. Then the WTA M_ζ over $\Sigma \cup X_l \cup Y_m$ and $\mathcal{S}$ is obtained from M by adding the following transitions with weight 1:

$$(p', x_\iota, p)\,, \qquad (1 \leq \iota \leq l, p \in P, p' \in P^{r_\iota})$$
$$(\varepsilon, y_\iota, p)\,. \qquad (1 \leq \iota \leq m, p \in P)$$

Let $d \in D(M_\zeta)$ be a partial run on ζ. We observe that, for every $v \in X_l \cup Y_m$, d contains at most one transition with the terminal symbol v. If it does contain such a transition, then we denote it by $d(v)$. If d does not contain the respective transition, then the notation $d(v)$ is not defined. Accordingly, whenever we employ the notation, we imply that the transition be contained.

In the following, when we reason about bullet positions, we will often use the lexicographic order on $\mathbb{N}^*$ (see Sec. 2.1.3) and denote it by $\leq$. Let Δ be an alphabet and V a set. For every tree $t \in T_\Delta(V)$, the *set* $\mathrm{bpos}(t)$ *of bullet positions of* t is defined by

$$\mathrm{bpos}(t) = \{(\varepsilon, 0), (\varepsilon, 1)\} \cup \{(1w, j) \mid w \in \mathrm{pos}(t), j \in \{0, \ldots, \mathrm{rk}_t(w)\}\}\,.$$

Example 3.4.3 We can imagine a bullet position (v, j) of t by means of a tree obtained from t by putting a special root symbol $\top$ on top and inserting exactly one occurrence of $\bullet$, as illustrated in Fig. 3.8. Then $v(j+1)$ is the position of the bullet in the modified tree, and a position $w \in \mathrm{pos}(t)$ is left of the bullet if $1w \leq vj$. □

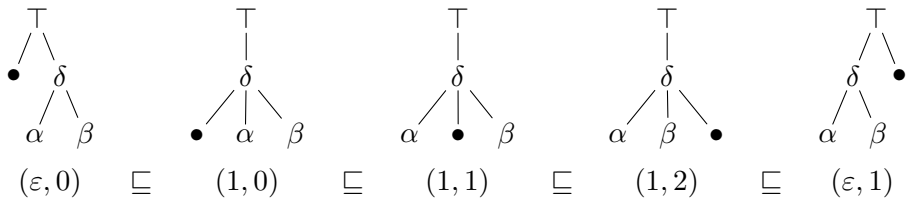


Figure 3.8: Viewing the bullet as a node in a variant of the tree $\delta(\alpha, \beta)$.

Let $\bullet \in \mathrm{bpos}(t)$ and $\bullet = (v, j)$. We define two unary predicates on $\mathrm{pos}(t)$, "is left of $\bullet$" and "is immediately right of $\bullet$", as follows. Let $w \in pos(t)$. Then we let

$$w \leq \bullet \iff 1w \leq vj \qquad \text{and} \qquad \bullet\, w \iff 1w = v(j+1)\ .$$

For every $\delta \in \Delta \cup V$ we use $\delta \leq \bullet$ to denote that there is a $w \in \mathrm{pos}(t)$ such that $t(w) = \delta$ and $w \leq \bullet$; and likewise for $\bullet\delta$.

Moreover, we define four partial mappings $\rightarrow$, $\downarrow$, $\leftarrow$, and $\uparrow$ of type $\mathrm{bpos}(t) \to \mathrm{bpos}(t)$ as follows.

1. If $j < \mathrm{rk}_{\top(t)}(v)$, then $\bullet\rightarrow = (v, j+1)$.
2. If $j < \mathrm{rk}_{\top(t)}(v)$, then $\bullet\downarrow = (v(j+1), 0)$.
3. If $j > 0$, then $\bullet\leftarrow = (v, j-1)$.
4. If $v = v'j'$ with $v' \in \mathbb{N}^*$ and $j' \in \mathbb{N}$, then $\bullet\uparrow = (v', j')$.

Finally, we define the order $\sqsubseteq$ on $\mathrm{bpos}(t)$, illustrated in Fig. 3.8, by

$$(v, j) \sqsubseteq (v', j') \quad \iff \quad v(j+1) \leq v'(j'+1)\ .$$

In order to facilitate our upcoming considerations, we introduce a new (pseudo) transition Ω. We let $\hat{q}_0 \notin Q$, $\hat{Q} = Q \cup \{\hat{q}_0^{(0)}\}$, and $\Omega = (q_0, \langle x_1 x_1 \rangle, \hat{q}_0)$. We note that $\Omega \notin R$. Nevertheless, we will transfer concepts defined for elements of R, such as the input tree, to Ω as well. This transfer is possible because a WSCFTG with the transition Ω exists, and said concepts do not depend on the assumption that $\Omega \notin R$.

A *base item* is a quadruple $(\rho, d, \bullet, a)$ such that

(i) $\rho \in R \cup \{\Omega\}$,

(ii) $d \in D(M_\zeta)$ is a partial run on the input tree ζ of ρ,

(iii) $\bullet \in \mathrm{bpos}(d)$ such that $d(w) \notin P$ iff $w \leq \bullet$ for every $w \in \mathrm{pos}(d)$,

(iv) $a \in \{0, \ldots, \mathrm{rk}(\rho)\}$ is called the *active index*; and if $a > 0$, then $\bullet x_a$.

We note that we abused notation in the last item. Since $\bullet \in \mathrm{bpos}(d)$, we also have $\bullet \in \mathrm{bpos}(\zeta)$, and we interpret $\bullet x_a$ in that sense. We will continue to do so. By $\mathbb{B}$ we denote the set of all base items. The same symbol $\mathbb{B}$ often denotes the Boolean semiring, but we will not use this semiring here. We make $\mathbb{B}$ a ranked alphabet by carrying over the rank from the first component of each base item. For every I and ι, we use $I \setminus \iota$ to denote that $I \in \mathbb{B}$, ι is the active index of I, and $\iota > 0$.

Recall that R' is the set of transitions of $M \lhd G$. For every $(q, p) \in \hat{Q} \times P$ the *set* $\mathbb{D}^{(q,p)}$ *of base-item trees for* (q, p) is a subset of $T_{R' \cup \mathbb{B}}(Q \cup (Q \times P))$, defined as follows. The family $(\mathbb{D}^{(q,p)} \mid (q, p) \in \hat{Q} \times P)$ is the smallest family $(D^{(q,p)} \mid (q, p) \in \hat{Q} \times P)$ such that for every $(q, p) \in \hat{Q} \times P$:

- if $q \in Q$, then $(q, p) \in D^{(q,p)}$,
- if $q \in Q^{(m)}$ and $p' \in P^m$, then $D_{\mathrm{co}}^{(q,p,p')}(M \lhd G) \subseteq D^{(q,p)}$,
- if $I \in \mathbb{B}$, $I = (\rho, d, \bullet, a)$, $\rho = (q_1 \cdots q_l, \langle \zeta \zeta' \rangle, q)$, $\pi_P(d) = p$, and, for every $\iota \in \{1, \ldots, l\}$,
  - if $x_\iota \leq \bullet$, then $\delta_\iota \in D_{\mathrm{co}}^{(q_\iota, d(x_\iota))}(M \lhd G)$;
  - if $\iota = a$, then $\delta_\iota \in D^{(q_\iota, p_\iota)}$ where p_ι is uniquely determined by $\bullet p_\iota$,
  - otherwise, $\delta_\iota = q_\iota$,

  then $I(\delta_1, \ldots, \delta_l) \in D^{(q,p)}$.

3.4.2 Item syntax and semantics

Based on the concept of a base-item tree, we can now define refined versions of notions such as "reachable" and "productive". Instead of giving each refined notion a proper name, we use *items*, i.e., syntactic representations of statements. Now we define the syntax and semantics of these items.

We begin with a few auxiliary definitions. Let $\zeta \in C_\Sigma(m, r_1, \ldots, r_l)$ and $w \in \mathrm{pos}(\zeta)$. A *$w$-assignment θ for ζ* is defined like an assignment for ζ, however, its domain consists only of the variables that occur in $\zeta|_w$. Let $d \in D(M_\zeta)$ be a partial run on ζ. We say that *θ and d agree* if $\theta(v) = d(v)$ for every symbol v in the domain of θ.

Let $\rho \in R$ and $\rho = (q_1 \cdots q_l, \langle \zeta \zeta' \rangle, q)$. Then we set $M_\rho = M_\zeta$, $\mathrm{pos}(\rho) = \mathrm{pos}(\zeta)$, and $\mathrm{rk}_\rho(w) = \mathrm{rk}_\zeta(w)$ for every $w \in \mathrm{pos}(\zeta)$. An *assignment for ρ* is an assignment for ζ, and likewise for w-assignments.

For every $i \in \{0, \ldots, 5\}$, we define the *set $\mathbb{I}_i$ of items of type i* as follows:

Ω	x_1	$\bullet$ 0	1

Figure 3.9: Root base item.

- $\mathbb{I}_0 = \{(q, p) \mid q \in Q, p \in P\}$,
- $\mathbb{I}_1 = \{[q, p, p'] \mid \exists m\colon q \in Q^{(m)}, p \in P, p' \in P^m\}$,
- $\mathbb{I}_2 = \{[\rho, w, p, \theta] \mid \rho \in R, w \in \mathrm{pos}(\rho), p \in P, \theta \text{ is a } w\text{-assignment for } \rho\}$,
- $\mathbb{I}_3 = \{[\rho, w, p] \mid \rho \in R, w \in \mathrm{pos}(\rho), p \in P\}$,
- $\mathbb{I}_4 = \{[\rho, w, j, p, p'] \mid \rho \in R, w \in \mathrm{pos}(\rho), \mathrm{rk}_\rho(w) = k, j \in \{0, \dots, k\}, p \in P, p' \in P^k\}$,
- $\mathbb{I}_5 = \{(\rho, w, p) \mid \rho \in R, w \in \mathrm{pos}(\rho), p \in P\}$.

The *set* $\mathbb{I}$ *of items* is $\bigcup_i \mathbb{I}_i$. Note that this is a disjoint union. We define the mapping $\mathit{type}\colon \mathbb{I} \to \{0, \dots, 5\}$ such that, for every $I \in \mathbb{I}$, $\mathit{type}(I)$ is the unique i with $I \in \mathbb{I}_i$.

We define the *models* relation as the smallest relation

$$\models \subseteq \{(\delta, \omega) \mid \delta \in \mathbb{D}^{(\hat{q}_0, p_0)}, \omega \in \mathrm{pos}(\delta)\} \times \mathbb{I}$$

such that for every assignment of the free variables the following statements hold:

$(\boldsymbol{\delta}, \boldsymbol{\omega}) \models (\boldsymbol{q}, \boldsymbol{p})$ if $\delta(\omega) \searrow \iota$ and $\delta(\omega\iota) = (q, p)$;

$(\boldsymbol{\delta}, \boldsymbol{\omega}) \models [\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{p'}]$ if $\delta(\omega) \searrow \iota$ and $\delta|_{\omega\iota} \in D_{\mathrm{co}}^{(q,p,p')}(M \lhd G)$;

$(\boldsymbol{\delta}, \boldsymbol{\omega}) \models [\boldsymbol{\rho}, \boldsymbol{w}, \boldsymbol{p}, \boldsymbol{\theta}]$ if $\delta(\omega) = (\rho, d, \bullet\!\!\rightarrow, 0)$, $\bullet w$, $d|_w \in D^p(M_\rho, \zeta|_w)$, and θ and d agree;

$(\boldsymbol{\delta}, \boldsymbol{\omega}) \models [\boldsymbol{\rho}, \boldsymbol{w}, \boldsymbol{p}]$ if $(\delta, \omega) \models [\rho, w, p, \theta]$;

$(\boldsymbol{\delta}, \boldsymbol{\omega}) \models [\boldsymbol{\rho}, \boldsymbol{w}, \boldsymbol{j}, \boldsymbol{p}, \boldsymbol{p'}]$ if $\delta(\omega) = (\rho, d, (1w, j), 0)$ and $d(w) = (p', \zeta(w), p)$;

$(\boldsymbol{\delta}, \boldsymbol{\omega}) \models (\boldsymbol{\rho}, \boldsymbol{w}, \boldsymbol{p})$ if $\delta(\omega) = (\rho, d, \bullet, 0)$, $\bullet w$, and $d(w) = p$.

We denote the models relation by $\models$. An item I is *valid* if $(\delta, \omega) \models I$ for some δ and ω; and it is *invalid* if it is not valid.

Example 3.4.4 (Ex. 3.4.2 contd.) Let δ_1 and δ_2 be obtained from the base-item trees shown in Fig. 3.7 (a) and (b), respectively, by putting the base item from Fig. 3.9 on top. Then we have, e.g.,

$$\begin{aligned}
&(\delta_1, 12) \models [\rho_3, \varepsilon, 1, 0, r1r] \,, && (\delta_2, 1) \models [\rho_1, 1, 1, 0, r1r] \,, \\
&(\delta_1, 12) \models [\rho_3, 1, r, \emptyset] \,, && (\delta_2, 1) \models [\rho_1, 11, r, \emptyset] \,, \\
&(\delta_1, 12) \models [\rho_3, 1, r] \,, && (\delta_2, 1) \models [\rho_1, 11, r] \,, \\
&(\delta_1, 12) \models (\rho_3, 2, 1) \,, && (\delta_2, 1) \models (\rho_1, 12, 1) \,, \\
&(\delta_1, 12) \models (f, 1) \,, && (\delta_2, 1) \models (q, 1) \,.
\end{aligned}$$

We compare our refined notion of reachability with the classical one; more specifically, we compare the two statements

(a) the item $(q, 1)$ is valid and (b) the state $(q, 1)$ is reachable .

The statements are true because (a) $(\delta_2, 1) \models (q, 1)$ and (b) $\rho_1'((q, 1), (f, 0, 0)) \in D^{(q,0)}(M \lhd G)$, where ρ_1' is the transition that stems from ρ_1 via the obvious state augmentation. In δ_2, we do not yet see whether the base item for ρ_1 can be extended to a transition such as ρ_1'. In this sense, (a) is a weaker statement than (b). On the other hand, we see in δ_2 that the state for x_2 is productive. In this sense, (a) is stronger than (b). □

We observe that both the set $\mathbb{D}^{(\hat{q}_0, p_0)}$ and the set $D^{(q_0,p_0)}(M \lhd G)$ can be viewed as superset approximations of $D_{\text{co}}^{(q_0,p_0)}(M \lhd G)$ – albeit with different degrees of sophistication. Hence, we can transfer Obs. 3.4.1 to our new setting, as follows.

Lemma 3.4.5 *If a transition $\rho\theta$ of $M \lhd G$ is useful, then $[\rho, \varepsilon, \theta(\diamond), \theta']$ is valid, where θ' is obtained from θ by removing $\diamond$ from its domain.*

PROOF. Let $\rho\theta$ be a useful transition of $M \lhd G$. Then there are $d' \in D_{\text{co}}^{(q_0,p_0)}(M \lhd G)$ and $\omega \in \text{pos}(d')$ such that $d'(\omega) = \rho\theta$. We will construct a δ such that $(\delta, 1\omega) \models [\rho, \varepsilon, \theta(\diamond), \theta']$. For this, we "prune" d', removing parts that are not licensed by the definition of a base-item tree.

To this end, let $\omega = (\omega_1, \ldots, \omega_n)$, $\rho_1, \ldots, \rho_{n+1} \in R$, and $\theta_1, \ldots, \theta_{n+1}$ state assignments such that $\rho_j\theta_j = d'(\omega_1 \cdots \omega_{j-1})$. Then there are $d_1, \ldots, d_{n+1}$ such that $d_j \in D^{\theta_j(\diamond)}(M\theta_j, \zeta_j)$ where ζ_j is the input tree of ρ_j. We define trees $\delta_1, \ldots, \delta_{n+1}$ inductively. To this end, let $j \in \{1, \ldots, n+1\}$ and $\rho_j = (q_1 \cdots q_l, \langle \zeta\zeta' \rangle, q)$. If $j = n+1$, then we let

$$\delta_j = (\rho_j, d_j, (\varepsilon, 1), 0)\big(d'|_{\omega 1}, \ldots, d'|_{\omega l}\big) \,.$$

Otherwise, we let $w = (w_1, \ldots, w_m)$ be the position in ζ labeled x_{ω_j}, and we define trees $\hat{d}_1, \ldots, \hat{d}_{m+1}$ inductively. For this, let $\hat{\jmath} \in \{1, \ldots, m+1\}$ and $d_j(w_1 \cdots w_{j-1}) = (p_1 \cdots p_k, \sigma, p)$. If $\hat{\jmath} = m+1$, then we let $\hat{d}_{\hat{\jmath}} = p$. Otherwise, we let

$$\hat{d}_{\hat{\jmath}} = (p_1 \cdots p_k, \sigma, p)\big(d'_1, \ldots, d'_k\big) \;,$$

where $d'_{j'} = d_j|_{w_1 \cdots w_{\hat{\jmath}-1} j'}$ if $j' < w_{\hat{\jmath}}$, it is $\hat{d}_{\hat{\jmath}+1}$ if $j' = \hat{\jmath}$, and it is $p_{j'}$ otherwise. Now we let

$$\delta_j = (\rho_j, \hat{d}_1, \bullet, \omega_j)\big(\delta'_1, \ldots, \delta'_l\big) \;,$$

where

- the bullet position $\bullet$ is determined by $\bullet w$, and
- δ'_ι is $d'|_{\omega_1 \cdots \omega_{j-1} \iota}$ if x_ι occurs in ζ strictly left of x_{ω_j}, it is δ_{j+1} if $\iota = \omega_j$, and it is q_ι otherwise.

We let $\delta = (\Omega, p_0, (\varepsilon, 0), 1)\big(\delta_1\big)$. We omit the proof of $(\delta, 1\omega) \models [\rho, \varepsilon, \theta(\diamond), \theta']$. ∎

3.4.3 Algorithm

Lemma 3.4.5 implies that, when we compute every valid item of the form $[\rho, \varepsilon, p, \theta]$, then we can read off a superset of the useful transitions of $M \lhd G$. This is the basic approach of our algorithm, Alg. 3.1.

The algorithm proceeds in two steps. In the first step (Lines 1–4), it computes the set of all valid items by means of the deductive system shown in Fig. 3.10. As usual, the deductive system consists of inference rules, each one being a syntactic representation of a conditional implication [89, 142]. Since there are only finitely many items, this process will terminate. Roughly speaking, the items drive a depth-first left-to-right simulation of M on the input trees of transitions of G. Items with round brackets are responsible for top-down traversal, and items with square brackets are responsible for horizontal and bottom-up traversal. In the second step (Lines 5–9), we use the items to construct transitions of $M \lhd G$, together with their weights.

Example 3.4.6 (Ex. 3.3.1 contd.) We demonstrate the inference rules of Fig. 3.10 by showing the generation of items for the transition ρ_3 of G. In Fig. 3.11 we show the input tree ζ of ρ_3 in bold-face letters and lines. On top of this syntactic structure, we have drawn another graph, consisting of items and arrows. Close to every position w of ζ, we have placed those items that involve w. The arrows show the dependencies between the items as they are expressed by the rules of the deduction schema. Finally, we note

(1) $\dfrac{}{(q_0, p_0)}$

(2) $\dfrac{(q, p)}{(\rho, \varepsilon, p)}$

(3) $\dfrac{(q, p) \qquad [\rho, \varepsilon, p, \theta]}{[q, p, p']} \;\; \{ \;\; p' = (\theta(y_1), \ldots, \theta(y_m))$

(4) $\dfrac{(\rho, w, p)}{[\rho, w, 0, p, p']} \;\; \Big\{ \;\; (p', \zeta(w), p) \in R_M^{(\mathrm{rk}_\zeta(w))}$

(5) $\dfrac{[\rho, w, j, p, (p_1, \ldots, p_k)]}{(\rho, w(j+1), p_{j+1})} \;\; \{ \;\; 0 \leq j < k$

(6) $\dfrac{[\rho, w, j, p, (p_1, \ldots, p_k)] \qquad [\rho, w(j+1), p_{j+1}]}{[\rho, w, j+1, p, (p_1, \ldots, p_k)]} \;\; \{ \;\; 0 \leq j < k$

(7) $\dfrac{[\rho, w, p, \theta]}{[\rho, w, p]}$

(8) $\dfrac{(\rho, w, p)}{(q_\iota, p)} \;\; \{ \;\; \zeta(w) = x_\iota$

(9) $\dfrac{(\rho, w, p) \qquad [q_\iota, p, p']}{[\rho, w, 0, p, p']} \;\; \{ \;\; \zeta(w) = x_\iota$

(10) $\dfrac{(\rho, w, p)}{[\rho, w, p, \{y_\iota \mapsto p\}]} \;\; \{ \;\; \zeta(w) = y_\iota$

(11) $\dfrac{[\rho, w, k, p, p'] \qquad \begin{matrix}[\rho, w1, p_1, \theta_1] \\ \cdots \; [\rho, wk, p_k, \theta_k]\end{matrix}}{[\rho, w, p, \theta \cup \theta_1 \cup \cdots \cup \theta_k]} \;\; \{ \;\; p' = (p_1, \ldots, p_k)$

where $\theta = \{\zeta(w) \mapsto (p, p')\}$ if $\zeta(w) \in X$, and
$\theta = \emptyset$ otherwise

Note: we assume that $\rho \in R$, $\rho = (q_1 \cdots q_l, \langle \zeta \zeta' \rangle, q)$, and $q \in Q^{(m)}$.

Figure 3.10: Deductive parsing schema for the input product.

Algorithm 3.1 Product construction algorithm.

Require: a WSCFTG G and a WRTG M with
$G = (Q, R, \mu, q_0)$ and
$M = (P, R_M, \mu_M, p_0)$,

Ensure:
R_u contains at least the useful transitions of $M \lhd G$,
μ_u coincides with the weight assignment of $M \lhd G$ on R_u

▷ step 1: compute $\mathcal{I}$

1: $\mathcal{I} \leftarrow \emptyset$
2: **repeat**
3: add items to $\mathcal{I}$ by applying the rules in Fig. 3.10
4: **until** convergence

▷ step 2: compute transitions

5: $R_u \leftarrow \emptyset$
6: **for** $[\rho, \varepsilon, p, \theta] \in \mathcal{I}$ **do**
7: $\theta' \leftarrow \theta \cup \{\diamond \mapsto p\}$
8: $R_u \leftarrow R_u \cup \{\rho\theta'\}$
9: $\mu_u(\rho\theta') \leftarrow \mu(\rho) \cdot [\![M\theta']\!](\zeta)$ where ζ is the input tree of ρ

that the deduction schema in Fig. 3.10 can be considered as an attribute grammar [107, 63, 48] that is based on a macro grammar rather than a context-free grammar. From this perspective, Fig. 3.11 shows the dependency graph on ζ, where the items are the attribute occurrences and the arrows are the attribute dependencies. □

The following theorem describes the behavior of Alg. 3.1.

Theorem 3.4.7 *Let G be a WSCFTG and M a WTA, both in root-state form. Moreover, let $M \lhd G = (Q', R', \mu', q'_0)$, R_U the set of useful transitions of $M \lhd G$, and (R_u, μ_u) the output of Alg. 3.1. Then*

- *$R_u \subseteq R'$ and $\mu_u = \mu'|_{R_u}$, i.e., the algorithm is* correct*;*
- *$R_U \subseteq R_u$, i.e., the algorithm is* complete*.*

PROOF. Follows from Lms. 3.4.11 and 3.4.12, which we show in Sec. 3.4.4. ■

Now we analyze the worst-case space and time complexity of the first step. To this end, we assume that each item occupies unit space. Following [136] we determine the space complexity by the number of items, and we determine the time complexity by the number of instantiations of the inference rules.

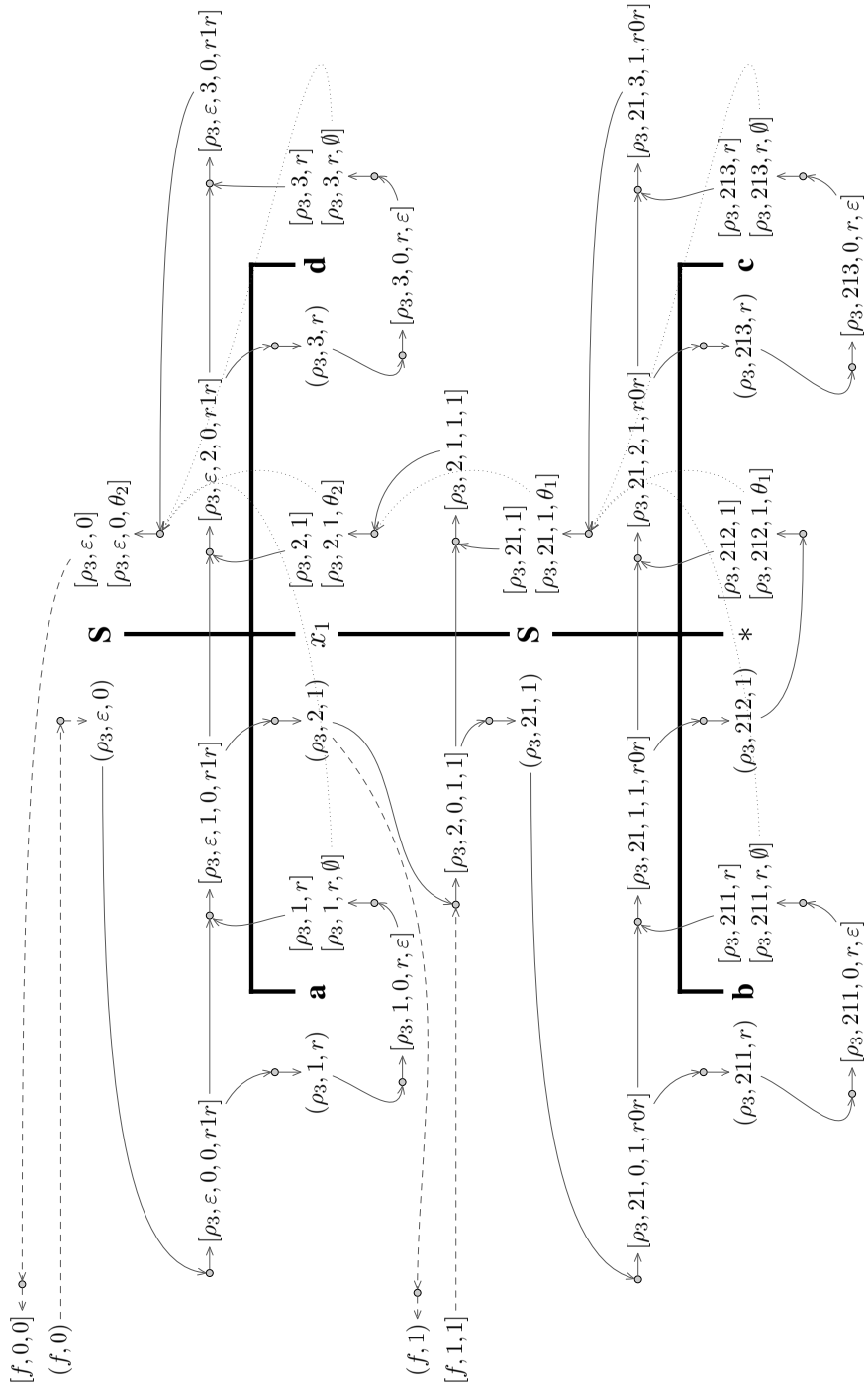


Figure 3.11: Item generation on the transition ρ_3 of Ex. 3.3.1, where $\theta_1 = \{y_1 \mapsto 0\}$ and $\theta_2 = \{y_1 \mapsto 0, x_1 \mapsto (1, 1)\}$.

In our case, the space complexity is either dominated by $\mathbb{I}_2$ or $\mathbb{I}_4$, and we have $|\mathbb{I}_2| \in O\big(|G|_{\mathrm{in}} \cdot |P|^C\big)$ and $|\mathbb{I}_4| \in O\big(|G|_{\mathrm{in}} \cdot |R_M|\big)$. Here, the factor $|G|_{\mathrm{in}}$ denotes the *input size of* G, defined by $\sum_{\rho \in R} |\mathrm{pos}(\zeta(\rho))|$, where $\zeta(\rho)$ is the input tree of ρ. It captures the components ρ and w (ρ, w, and j, respectively) in said items, which together identify exactly one node of an input tree of G. The factor $|R_M|$ captures the components p and $p_1 \cdots p_k$. Finally, the factor $|P|^C$ captures p and θ, where C is given at the end of Sec. 3.3.2.

In the worst case, which is what we consider here, the algorithm obviously uses more space than the resulting $M \lhd G$. The reason is that we need extra space to manage the depth-first left-to-right traversal. It is our hope, however, that this traversal pays off in the average case, and that a lot of useless rules can be avoided.

The time complexity is dominated by Rule 12; so it is in $O(|G|_{\mathrm{in}} \cdot |R_M| \cdot |P|^C)$, where $|P|^C$ captures the union $\theta \cup \theta_1 \cup \cdots \cup \theta_k$, which is disjoint, because the input tree of any rule is linear in $X \cup Y$.

3.4.4 Correctness and completeness

In this section, we show that Alg. 3.1 is correct and complete. We follow a top-down approach, that is, we prove higher-level statements first and auxiliary statements second. Formally, we view the deductive system as a relation

$$\mathcal{R} \subseteq \mathbb{I}^* \times \mathbb{I}\,.$$

Figure 3.10 specifies this relation in terse form, and it translates into

$$\mathcal{R} = \{(\varepsilon, (q_0, p_0))\} \cup \{((q,p), (\rho, \varepsilon, p)) \mid \rho \in R|_q, q \in Q, p \in P\} \cup \ldots$$

We define the mapping $\mathcal{F}\colon \mathcal{P}(\mathbb{I}) \to \mathcal{P}(\mathbb{I})$ such that, for every $\mathcal{I} \subseteq \mathbb{I}$, $\mathcal{F}(\mathcal{I})$ is the set of all items that can be generated by applying inference rules to items in $\mathcal{I}$, i.e.,

$$\mathcal{F}(\mathcal{I}) = \{I \mid \exists n \in \mathbb{N}, I_1, \ldots, I_n \in \mathcal{I}\colon (I_1 \cdots I_n, I) \in \mathcal{R}\}\,.$$

Then the set $\mathcal{I}$ after Line 4 of Alg. 3.1 is the set $\bigcup_i \mathcal{F}^i(\emptyset)$. We show that the deductive system is correct and complete; that is, we show the following theorem.

Theorem 3.4.8 *Let $\mathcal{I}_\star \subseteq \mathbb{I}$ be the set of all valid items, and $\mathcal{I} \subseteq \mathbb{I}$ be the set of items computed in Alg. 3.1, i.e., $\mathcal{I} = \bigcup_i \mathcal{F}^i(\emptyset)$. Then*

- *$\mathcal{I} \subseteq \mathcal{I}_\star$, i.e., the deductive system is correct; and*
- *$\mathcal{I}_\star \subseteq \mathcal{I}$, i.e., the deductive system is complete.*

PROOF. Correctness (by contradiction): assume that $\bigcup_i \mathcal{F}^i(\emptyset)$ contains an invalid item. Then there is a minimal i such that $\mathcal{F}^i(\emptyset)$ contains an invalid item. Let I be such an item. If $i = 0$, then $I \in \emptyset$, which is a contradiction. Thus $i > 0$. Then there are $n \in \mathbb{N}$ and $I_1, \ldots, I_n \in \mathcal{F}^{i-1}(\emptyset)$ such that $(I_1 \cdots I_n, I) \in \mathcal{R}$. Assume for the time being that $I_1, \ldots, I_n \in \mathcal{I}_\star$. By Lm. 3.4.9, also $I \in \mathcal{I}_\star$, which is a contradiction. Hence, there is a j such that I_j is invalid. But $I_j \in \mathcal{F}^{i-1}(\emptyset)$, and thus i is not minimal, which is our final contradiction.

Completeness (by contradiction): assume that $\mathcal{I}_\star \setminus \mathcal{I}$ is not empty. Then the set

$$C = \{(I, \delta, \omega) \mid (\delta, \omega) \models I, I \notin \mathcal{I}\}$$

is not empty. Let $\sqsubseteq$ be the well-founded order postulated in Lm. 3.4.10. Since $C \neq \emptyset$, this set has a minimal element with respect to $\sqsubseteq$, say (I, δ, ω). By said lemma, there are $(I_1, \delta_1, \omega_1), \ldots, (I_n, \delta_n, \omega_n)$ such that $(I_1 \cdots I_n, I) \in \mathcal{R}$, $(\delta_j, \omega_j) \models I_j$, and $(I_j, \delta_j, \omega_j) \sqsubseteq (I, \delta, \omega)$. Assume for the time being that $I_1, \ldots, I_n \in \mathcal{I}$. Then there is an i such that $I_1, \ldots, I_n \in \mathcal{F}^i(\emptyset)$. Since $(I_1 \cdots I_n, I) \in \mathcal{R}$, $I \in \mathcal{F}^{i+1}(\emptyset)$, and thus $I \in \mathcal{I}$, which contradicts our assumption that $(I, \delta, \omega) \in C$. Hence, there is a j such that $I_j \notin \mathcal{I}$. It is easy to see from the definition of $\mathcal{R}$ that $I_j \neq I$, and thus $(I_j, \delta_j, \omega_j) \neq (I, \delta, \omega)$. But then $(I_j, \delta_j, \omega_j)$ is strictly smaller than (I, δ, ω), which contradicts our assumption that (I, δ, ω) is minimal. ∎

Lemma 3.4.9 *Let $(I_1 \cdots I_n, I) \in \mathcal{R}$, and let $(\delta_1, \omega_1) \models I_1$, …, $(\delta_n, \omega_n) \models I_n$. Then there is a pair (δ, ω) such that $(\delta, \omega) \models I$.*

Instead of a full proof, we only consider how to construct (δ, ω). To this end, we use a terse notation, like in the deductive system itself. The construction is shown in Figs. 3.12 and 3.13. We employ the following auxiliary definitions.

Let $\rho \in R$, $\rho = (q_1 \cdots q_l, \langle \zeta\zeta' \rangle, q)$, $\delta_1 \in \bigcup_p \mathbb{D}^{(q,p)}$, $\delta_1(\varepsilon) = (\rho, d, \bullet, 0)$, $w \in \mathrm{pos}(d)$, $\bullet w$, $p \in P$, and $d(w) = p$.

– Let $r \in R_M^{(\mathrm{rk}_\zeta(w))}$, $r = (p', \zeta(w), p)$, and $p' = (p_1, \ldots, p_k)$. We write $(\delta_1, r) \xrightarrow{(4)} \delta$ to indicate that

$$\delta = (\rho, d[r(p_1, \ldots, p_k)]_w, \bullet\!\downarrow, 0)\big(\delta_1|_1, \ldots, \delta_1|_l\big) \ .$$

– Let $\delta_2 \in \bigcup_p \mathbb{D}^{(q,p)}$, $\delta_2(\varepsilon) = (\rho, d_2, \bullet\!\rightarrow, 0)$, and $d_2|_w \in D^p(M_\rho, \zeta|_w)$. Then $(\delta_1, \delta_2) \xrightarrow{(6)} \delta$ means

$$\delta = (\rho, d[d_2|_w]_w, \bullet\!\rightarrow, 0)\big(\delta'_1, \ldots, \delta'_l\big) \ ,$$

where δ'_ι is $\delta_1|_\iota$ if x_ι does not occur in $\zeta|_w$; otherwise δ'_ι is $\delta_2|_\iota$.

(1) $$\frac{}{((\Omega, p_0, (\varepsilon, 0), 1)\big((q_0, p_0)\big), \varepsilon) \models (q_0, p_0)}$$

(2) $$\frac{(\delta_1, \omega_1) \models (q, p)}{(\delta_1[(\rho, p, (\varepsilon, 0), 0)\big(q_1, \ldots, q_l\big)]_{\omega_1 \iota}, \omega_1 \iota) \models (\rho, \varepsilon, p)} \quad \{\ \delta_1(\omega_1) \searrow \iota$$

(4) $$\frac{(\delta_1, \omega_1) \models (\rho, w, p)}{(\delta_1[\delta]_{\omega_1}, \omega_1) \models [\rho, w, 0, p, p']} \quad \begin{cases} r = (p', \zeta(w), p) & r \in R_M^{(\mathrm{rk}_\zeta(w))} \\ (\delta_1|_{\omega_1}, r) \xrightarrow{(4)} \delta \end{cases}$$

(5) $$\frac{(\delta_1, \omega_1) \models [\rho, w, j, p, (p_1, \ldots, p_k)]}{(\delta_1, \omega_1) \models (\rho, w(j+1), p_{j+1})} \quad \{\ 0 \leq j < k$$

(7) $$\frac{(\delta_1, \omega_1) \models [\rho, w, p, \theta]}{(\delta_1, \omega_1) \models [\rho, w, p]}$$

(8) $$\frac{(\delta_1, \omega_1) \models (\rho, w, p)}{(\delta_1[\delta]_{\omega_1}, \omega_1) \models (q_\iota, p)} \quad \begin{cases} \zeta(w) = x_\iota & \delta_1|_{\omega_1} \xrightarrow{(8)} \delta \end{cases}$$

(10) $$\frac{(\delta_1, \omega_1) \models (\rho, w, p)}{(\delta_1[\delta]_{\omega_1}, \omega_1) \models [\rho, w, p, \{y_\iota \mapsto p\}]} \quad \begin{cases} \zeta(w) = y_\iota \\ \delta_1|_{\omega_1} \xrightarrow{(10)} \delta \end{cases}$$

Note: we assume that $\rho \in R$, $\rho = (q_1 \cdots q_l, \langle \zeta \zeta' \rangle, q)$, and $q \in Q^{(m)}$.

Figure 3.12: Construction for Lm. 3.4.9 (continued in Fig. 3.13).

(3) $$\frac{(\delta_1, \omega_1) \models (q, p) \qquad (\delta_2, \omega_2) \models [\rho, \varepsilon, p, \theta']}{(\delta_1[(\rho\theta)(\delta_2|_{\omega_2 1}, \ldots, \delta_2|_{\omega_2 l})]_{\omega_1 \iota}, \omega_1) \models [q, p, p']} \quad \begin{cases} p' = (\theta'(y_1), \ldots, \theta'(y_m)) \\ \delta_1(\omega_1) \searrow \iota \quad \theta = \theta' \cup \{\diamond \mapsto p\} \end{cases}$$

(6) $$\frac{(\delta_1, \omega_1) \models [\rho, w, j, p, (p_1, \ldots, p_k)] \qquad (\delta_2, \omega_2) \models [\rho, w(j+1), p_{j+1}]}{(\delta_1[\delta]_{\omega_1}, \omega_1) \models [\rho, w, j+1, p, (p_1, \ldots, p_k)]} \quad \begin{cases} 0 \leq j < k \\ (\delta_1|_{\omega_1}, \delta_2|_{\omega_2}) \xrightarrow{(6)} \delta \end{cases}$$

(9) $$\frac{(\delta_1, \omega_1) \models (\rho, w, p) \qquad (\delta_2, \omega_2) \models [q_\iota, p, p']}{(\delta_1[\delta]_{\omega_1}, \omega_1) \models [\rho, w, 0, p, p']} \quad \begin{cases} \zeta(w) = x_\iota \quad \delta_2(\omega_2) \searrow \iota' \\ (\delta_1|_{\omega_1}, \delta_2|_{\omega_2 \iota'}) \xrightarrow{(9)} \delta \end{cases}$$

(11) $$\frac{(\delta', \omega') \models [\rho, w, k, p, p_1 \cdots p_k] \qquad \begin{array}{c}(\delta_1, \omega_1) \models [\rho, w1, p_1, \theta_1] \\ \cdots \; (\delta_k, \omega_k) \models [\rho, wk, p_k, \theta_k]\end{array}}{(\delta'[\delta]_{\omega'}, \omega') \models [\rho, w, p, \theta \cup \theta_1 \cup \cdots \cup \theta_k]} \quad \begin{cases} p' = (p_1, \ldots, p_k) \\ (\delta_1|_{\omega_1}, \ldots, \delta_k|_{\omega_k}, \delta'|_{\omega'}) \xrightarrow{(11)} \delta \end{cases}$$

where θ is defined as in Fig. 3.10.

Note: we assume that $\rho \in R$, $\rho = (q_1 \cdots q_l, \langle \zeta \zeta' \rangle, q)$, and $q \in Q^{(m)}$.

Figure 3.13: Continuation of Fig. 3.12.

– Let $\zeta(w) = x_a$. Then $\delta_1 \xrightarrow{(8)} \delta$ means

$$\delta = (\rho, d, \bullet, a)\big(\delta'_1, \ldots, \delta'_l\big) \ ,$$

where δ'_ι is $\delta_1|_\iota$ if $\iota \neq a$, and δ'_a is (q_a, p).
– Let $\zeta(w) = x_\iota$, $q_\iota \in Q^{(m)}$, $p' \in P^m$, $\delta_2 \in D_{\mathrm{co}}^{(q_\iota, p, p')}(M \lhd G)$, and $p' = (p_1, \ldots, p_m)$. Then $(\delta_1, \delta_2) \xrightarrow{(9)} \delta$ shall mean

$$\delta = (\rho, d[(p', x_\iota, p)\big(p_1, \ldots, p_m\big)]_w, \bullet\downarrow, 0)\big(\delta'_1, \ldots, \delta'_l\big) \ ,$$

where $\delta'_{\iota'}$ is $\delta_1|_{\iota'}$ if $\iota' \neq \iota$, and $\delta'_{\iota'}$ is δ_2 otherwise.
– Let $\zeta(w) = y_\iota$. Then $\delta_1 \xrightarrow{(10)} \delta$ means

$$\delta = (\rho, d[(\varepsilon, y_\iota, d(w))]_w, \bullet\rightarrow, 0)\big(\delta_1|_1, \ldots, \delta_1|_l\big) \ .$$

– Let $r = (p', \zeta(w), p)$, $p' \in P^{(\mathrm{rk}_\zeta(w))}$, $p' = (p_1, \ldots, p_k)$, $\delta_1, \ldots, \delta_k, \delta' \in \bigcup_p \mathbb{D}^{(q,p)}$, $\delta_1(\varepsilon) = (\rho, d_1, \bullet_1, 0)$, …, $\delta_k(\varepsilon) = (\rho, d_k, \bullet_k, 0)$, $\delta'(\varepsilon) = (\rho, d, \bullet_k, 0)$, $w \in \mathrm{pos}(d)$, $d(w) = r$, $\bullet_j = (1w, j)$, and $d_j|_{wj} \in D^{p_j}(M_\rho, \zeta|_{wj})$. Then $(\delta_1, \ldots, \delta_k, \delta') \xrightarrow{(11)} \delta$ means

$$\delta = (\rho, d[r(d_1|_{w1}, \ldots, d_k|_{wk})]_w, \bullet_k\uparrow, 0)\big(\delta'_1, \ldots, \delta'_l\big) \ ,$$

where δ'_ι is $\delta|_\iota$ if x_ι does not occur in $\zeta|_w$; otherwise δ'_ι is $\delta_j|_\iota$ where j is the unique integer such that x_ι occurs in $\zeta|_{wj}$.

This finishes our proof sketch for Lm. 3.4.9. Now we turn to the lemma that we use to prove completeness of the deductive system. To this end, we define $\mathbb{I}' = \{(I, \delta, \omega) \mid I \in \mathbb{I}, \delta \in \mathbb{D}^{(\hat{q}_0, p_0)}, \omega \in \mathrm{pos}(\delta)\}$.

Lemma 3.4.10 *There is a well-founded order $\sqsubseteq$ on $\mathbb{I}'$ such that the following holds. Let $(\delta, \omega) \models I$. Then there are $(I_1, \delta_1, \omega_1), \ldots, (I_n, \delta_n, \omega_n) \in \mathbb{I}'$ such that*

- $(I_1 \cdots I_n, I) \in \mathcal{R}$,
- $(\delta_j, \omega_j) \models I_j$, *and*
- $(I_j, \delta_j, \omega_j) \sqsubseteq (I, \delta, \omega)$.

Again, we will not provide a full proof, but the definition for $\sqsubseteq$ and the construction of $(I_1, \delta_1, \omega_1), \ldots, (I_n, \delta_n, \omega_n)$.

First we define the order $\sqsubseteq$. As intermediate steps, we will define an order on $\bigcup_{(q,p)} \mathbb{D}^{(q,p)}$ and one on $\mathbb{B}$ as well. As a basis, we will refer back to the order $\sqsubseteq$ on

$D_{\text{pr}}(M_\rho)$ for some ρ (see Sec. 2.4.3) and the order $\sqsubseteq$ on $\mathrm{bpos}(d)$ for some d. As is becoming evident now, we keep reusing the symbol $\sqsubseteq$. This avoids clutter, and it is warranted, because the underlying sets are disjoint and the definitions are adjacent and only relevant for the purpose of the proof.

We define the binary relation $\sqsubseteq$ on $\mathbb{B}$ by letting $I \sqsubseteq I'$ if there are ρ, d, d', $\bullet$, $\bullet'$, a, and a' such that $I = (\rho, d, \bullet, a)$, $I' = (\rho, d', \bullet', a')$, $d \sqsubseteq d'$, $\bullet \sqsubseteq \bullet'$, and $a \leq a'$. We define the binary relation $\sqsubseteq$ on the set $\bigcup_{(q,p)} \mathbb{D}^{(q,p)}$ inductively by letting $\delta \sqsubseteq \delta'$ if one of the following statements holds:

- there is a $(q,p) \in Q \times P$ such that $\delta = q$ and $\delta' \in \mathbb{D}^{(q,p)}$,
- there is a $(q,p) \in Q \times P$ such that $\delta = (q,p)$ and $\delta' \in \mathbb{D}^{(q,p)} \setminus \{q\}$,
- there are $I \in \mathbb{B}$, $I' \in \mathbb{B}$, $\delta_1, \ldots, \delta_l$, and $\delta'_1, \ldots, \delta'_l$ such that $\delta = I(\delta_1, \ldots, \delta_l)$, $\delta' = I'(\delta'_1, \ldots, \delta'_l)$, $I \sqsubseteq I'$, and $\delta_j \sqsubseteq \delta'_j$,
- there are $I \in \mathbb{B}$, $\rho' \in R'$, $\delta_1, \ldots, \delta_l$, and $\delta'_1, \ldots, \delta'_l$ such that $\delta = I(\delta_1, \ldots, \delta_l)$, $\delta' = \rho'(\delta'_1, \ldots, \delta'_l)$, and $\delta_j \sqsubseteq \delta'_j$,
- there are $\rho \in R'$, $\delta_1, \ldots, \delta_l$, and $\delta'_1, \ldots, \delta'_l$ such that $\delta = \rho(\delta_1, \ldots, \delta_l)$, $\delta' = \rho(\delta'_1, \ldots, \delta'_l)$, and $\delta_j \sqsubseteq \delta'_j$.

Finally, we define the binary relation $\sqsubseteq$ on $\mathbb{I}'$ by letting $(I, \delta, \omega) \sqsubseteq (I', \delta', \omega')$ if one of the following statements holds:

- $|\mathrm{pos}(\delta)| < |\mathrm{pos}(\delta')|$,
- $|\mathrm{pos}(\delta)| = |\mathrm{pos}(\delta')|$, $\delta \neq \delta'$, and $\delta \sqsubseteq \delta'$,
- $|\mathrm{pos}(\delta)| = |\mathrm{pos}(\delta')|$, $\delta = \delta'$, and $\mathit{type}(I) < \mathit{type}(I')$,
- $|\mathrm{pos}(\delta)| = |\mathrm{pos}(\delta')|$, $\delta = \delta'$, $\mathit{type}(I) = \mathit{type}(I')$, and $\omega \leq \omega'$.

We omit the proof that $\sqsubseteq$ is indeed a well-founded order on $\mathbb{I}'$.

Now we show the construction of $(I_1, \delta_1, \omega_1), \ldots, (I_n, \delta_n, \omega_n)$. To this end, we use a terse notation again. The construction is shown in Figs. 3.14 and 3.15. Note that for every possible triple (I, δ, ω), there is a rule in these figures that can be applied to that triple. In one case, this is not immediately apparent, namely, if I has the form (q,p). Then we distinguish two cases. Either $\omega = \varepsilon$; then I and δ are uniquely determined, and we can apply Rule (1). Or there are $\omega' \in \mathbb{N}^*$ and $j \in \mathbb{N}$ such that $\omega = \omega' j$; then we can apply Rule (8). We employ the following auxiliary definition.

Let $\rho \in R$, $\rho = (q_1 \cdots q_l, \langle \zeta \zeta' \rangle, q)$, d a partial run of M_ζ on ζ, $w \in \mathrm{pos}(d)$, and $\delta \in \bigcup_p \mathbb{D}^{(q,p)}$.

(1) $$\frac{}{((\Omega, p_0, (\varepsilon, 0), 1)\big((q_0, p_0)\big), \varepsilon) \models (q_0, p_0)}$$

(2) $$\frac{(\delta, \omega) \models (\rho, \varepsilon, p)}{(\delta[(q, p)]_\omega, \omega) \models (q, p)}$$

(4) $$\frac{(\delta, \omega) \models [\rho, w, 0, p, p']}{(\delta[\delta_1]_\omega, \omega) \models (\rho, w, p)} \quad \begin{cases} (p', \zeta(w), p) \in R_M^{(\mathrm{rk}_\zeta(w))} \\ \delta_1 \xleftarrow{(4)} \delta|_\omega \end{cases}$$

(5) $$\frac{(\delta, \omega) \models (\rho, w(j+1), p_{j+1})}{(\delta, \omega) \models [\rho, w, j, p, (p_1, \ldots, p_k)]} \quad \{\ 0 \leq j < k$$

(7) $$\frac{(\delta, \omega) \models [\rho, w, p]}{(\delta, \omega) \models [\rho, w, p, \theta]} \quad \begin{cases} \delta(\omega) = (\rho, d, v, j) \\ \theta \text{ and } d \text{ agree} \end{cases}$$

(8) $$\frac{(\delta, \omega) \models (q_\iota, p)}{(\delta[\delta_1]_\omega, \omega) \models (\rho, w, p)} \quad \begin{cases} \omega \neq \varepsilon \\ \delta_1 \xleftarrow{(8)} \delta|_\omega \end{cases}$$

(9) $$\frac{(\delta, \omega) \models [\rho, w, 0, p, p']}{(\delta[\delta_1]_\omega, \omega) \models (\rho, w, p) \qquad (\delta[\delta_2]_\omega, \omega) \models [q_\iota, p, p']} \quad \begin{cases} \zeta(w) = x_\iota \\ (\delta_1, \delta_2) \xleftarrow{(9)} \delta|_\omega \end{cases}$$

(10) $$\frac{(\delta, \omega) \models [\rho, w, p, \theta]}{(\delta[\delta_1]_\omega, \omega) \models (\rho, w, p)} \quad \begin{cases} \zeta(w) = y_\iota \\ \delta_1 \xleftarrow{(10)} \delta|_\omega \end{cases}$$

Note: we assume that $\rho \in R$, $\rho = (q_1 \cdots q_l, \langle \zeta \zeta' \rangle, q)$, and $q \in Q^{(m)}$.

Figure 3.14: Construction for Lm. 3.4.10 (continued in Fig. 3.15).

(3) $$\frac{(\delta,\omega) \models [q,p,p']}{(\delta[(q,p)]_{\omega\iota},\omega) \models (q,p) \qquad (\delta[(\rho,d,\varepsilon,1)\big(\delta|_{\omega 1},\ldots,\delta|_{\omega l}\big)]_{\omega},\omega) \models [\rho,\varepsilon,p,\theta]} \quad \begin{cases} \delta(\omega) \searrow \iota \quad \rho\theta' = \delta(\omega\iota) \\ d \in D^{\theta'(\diamond)}(M\theta',\zeta) \\ \theta \text{ and } d \text{ agree} \end{cases}$$

(6) $$\frac{(\delta,\omega) \models [\rho,w,j,p,(p_1,\ldots,p_k)]}{(\delta[\delta_1]_{\omega},\omega) \models [\rho,w,j-1,p,(p_1,\ldots,p_k)] \qquad (\delta,\omega) \models [\rho,wj,p_j]} \quad \begin{cases} 0 < j \leq k \\ \delta_1 \xleftarrow{(6)} \delta|_{\omega} \end{cases}$$

(11) $$\frac{(\delta,\omega) \models [\rho,w,p,\theta]}{\begin{array}{l}(\delta[\delta_k]_{\omega},\omega) \models [\rho,w,k,p,p'] \\ \end{array} \quad \begin{array}{l}(\delta[\delta_1]_{\omega},\omega) \models [\rho,w1,p_1,\theta|_{w1}] \\ \cdots\ (\delta[\delta_k]_{\omega},\omega) \models [\rho,wk,p_k,\theta|_{wk}]\end{array}} \quad \begin{cases} (\delta_1,\ldots,\delta_k,p') \xleftarrow{(11)} \delta|_{\omega} \\ p' = (p_1,\ldots,p_k) \end{cases}$$

where $\theta|_{wj}$ is θ restricted to the variables occurring in $\zeta|_{wj}$

Note: we assume that $\rho \in R$, $\rho = (q_1 \cdots q_l, \langle\zeta\zeta'\rangle, q)$, and $q \in Q^{(m)}$.

Figure 3.15: Continuation of Fig. 3.14.

– Let $\delta(\varepsilon) = (\rho, d, \bullet\downarrow, 0)$ and $\bullet w$. When we write $\delta_1 \xleftarrow{(4)} \delta$, we mean that

$$\delta_1 = (\rho, d[\pi_P(d|_w)]_w, \bullet, 0)\big(\delta|_1, \ldots, \delta|_l\big) \ .$$

– Let $\delta(\varepsilon) = (\rho, d, \bullet\rightarrow, 0)$, $\bullet w$, $p \in P$, and $d|_w \in D^p(M_\rho, \zeta|_w)$. When we write $\delta_1 \xleftarrow{(6)} \delta$, we mean

$$\delta_1 = (\rho, d[p]_w, \bullet, 0)\big(\delta'_1, \ldots, \delta'_l\big) \ ,$$

where δ'_ι is $\delta_1|_\iota$ if x_ι does not occur in $\zeta|_w$, and δ'_ι is q_ι otherwise.
– Let $\delta(\varepsilon) = (\rho, d, \bullet, a)$ and $a > 0$. Then $\delta_1 \xleftarrow{(8)} \delta$ means

$$\delta_1 = (\rho, d, \bullet, 0)\big(\delta'_1, \ldots, \delta'_l\big) \ ,$$

where δ'_ι is $\delta_1|_\iota$ if $\iota \neq a$, and $\delta'_a = q_a$.
– Let $\delta(\varepsilon) = (\rho, d, \bullet\rightarrow, 0)$, $\bullet w$, and $d(w) = (p', x_a, p)$. By $(\delta_1, \delta_2) \xleftarrow{(9)} \delta$, we mean

$$\begin{aligned} \delta_1 &= (\rho, d[p]_w, \bullet, 0)\big(\delta'_1, \ldots, \delta'_l\big) \ , \\ \delta_2 &= (\rho, d[p]_w, \bullet, a)\big(\delta|_1, \ldots, \delta|_l\big) \ , \end{aligned}$$

where δ'_ι is $\delta_1|_\iota$ if $\iota \neq a$, and $\delta'_a = q_a$.
– Let $\delta(\varepsilon) = (\rho, d, \bullet\rightarrow, 0)$, $\bullet w$, and $d(w) = (\varepsilon, y_\iota, p)$. By $\delta_1 \xleftarrow{(10)} \delta$, we mean

$$\delta_1 = (\rho, d[p]_w, \bullet, 0)\big(\delta|_1, \ldots, \delta|_l\big) \ .$$

– Let $\delta(\varepsilon) = (\rho, d, \bullet\rightarrow, 0)$, $\bullet w$, and $d(w) = (p_1 \cdots p_k, \zeta(w), p)$. Let $w_1, \ldots, w_l$ be the positions in ζ labeled $x_1, \ldots, x_l$, respectively. By $(\delta_1, \ldots, \delta_k, p') \xleftarrow{(11)} \delta$, we mean

$$\begin{aligned} p' &= (p_1, \ldots, p_k) \ , \\ \delta_j &= (\rho, d_j, (1w, j), 0)\big(\delta'_{j1}, \ldots, \delta'_{jl}\big) \ , \end{aligned}$$

where

- $d_k = d$ and $d_{j-1} = d_j[p_j]_{wj}$,
- $\delta'_{j\iota}$ is $\delta|_\iota$ if $w_\iota \leq wj$, and $\delta'_{j\iota}$ is q_ι otherwise.

This finishes our proof sketch for Lm. 3.4.10.
Next we show that Alg. 3.1 is correct.

Lemma 3.4.11 *Let $M \lhd G = (Q', R', \mu', q'_0)$ and (R_u, μ_u) the output of Alg. 3.1. Then $R_u \subseteq R'$ and $\mu_u = \mu'|_{R_u}$.*

PROOF. Let $\rho' \in R_u$. By Line 8, there are $\rho \in R$, an ε-assignment θ for ρ, and a $p \in P$ such that $[\rho, \varepsilon, p, \theta] \in \mathcal{I}$, and $\rho' = \rho\theta'$ with $\theta' = \theta \cup \{\diamond \mapsto p\}$. It remains to show that $\zeta \in L(M\theta')$, where ζ is the input tree of ρ, and that $\mu_u(\rho') = \mu'(\rho')$. The latter is trivial because of Line 9 of the algorithm. We focus on the former.

By Thm. 3.4.8, we have that there is a pair (δ, ω) such that $(\delta, \omega) \models [\rho, \varepsilon, p, \theta]$. That is, there is a d such that $\delta(\omega) = (\rho, d, \varepsilon, 1)$, $d \in D^p(M_\rho, \zeta)$, and θ and d agree. It is easy to see that then $d \in D^p(M\theta', \zeta)$. ∎

Now we show that the algorithm is complete.

Lemma 3.4.12 *Let R_U be the set of useful transitions of $M \lhd G$ and (R_u, μ_u) the output of Alg. 3.1. Then $R_U \subseteq R_u$.*

PROOF. This lemma is a direct consequence of Lm. 3.4.5, Thm. 3.4.8, and Line 8 of the algorithm. ∎

3.5 Conclusion, discussion, and outlook

We have defined WSCFTGs, we have shown that the class of meanings of WSCFTGs is closed under input and output product with recognizable weighted tree languages, and we have considered an Earley-like algorithm for computing the corresponding product WSCFTG.

Originally, context-free tree grammars (CFTGs) have been defined in [160, 67, 68], and from that perspective, our WSCFTGs are *simple*; which refers to the requirement that each of the variables $y_1, \ldots, y_m$ shall occur exactly once in every tree in $C_\Sigma(m, r_1, \ldots, r_l)$. Synchronous CFTGs have already been defined in [146], with the same requirement. Our definition is inspired by [30, Def. 1].

In view of [102], we might call a WSCFTG whose states have at most rank 1 a weighted synchronous (non-strict) tree-adjoining grammar. Likewise, if the states have at most rank 0, we may speak of a weighted synchronous tree-substitution grammar. Since our product construction does not alter the maximal rank of the states, the closure result also holds for the restricted classes.

As mentioned at the very top, this chapter is a considerably expanded and revised version of [33, 32]. In particular, the proofs in these papers are very sketchy and arguably faulty. In the case of Thm. 3.3.3, a corresponding theorem is proved in [33, Sec. 6.2] in the spirit of Lm. 3.3.2, i.e., using a bijection π and an equivalence relation

$\equiv$. However, these objects are defined in a way that does not lend itself to a rigorous proof, and correspondingly, the proof is very heavy-handed. In this chapter, the key to the proof of Lm. 3.3.2 is the recursive definition of these objects on the one hand and Lm. 3.3.6 on the other, which in turn rests on the *seemingly* simple Obs. 3.2.2.

In the case of Thm. 3.4.7, the proof idea that we used in this chapter is already present in [33, Sec. 8], using a precursor of our base-item tree, dubbed a partial enriched derivation. This precursor lacks the pseudo-rule Ω, the bullet position, and the active index; and the concept of a partial run is only approximated. With these shortcomings, it is not possible to prove the deductive system sound nor complete; for instance, the item (q, p) should not mean that $R|_q$ is nonempty, but according to [33, Sec. 8] it does.

It should be noted that, in both cases, the proof ideas in [33] are in fact adequate and the proofs actually quite convincing; this just underscores the obstacles that lie between an adequate, convincing proof idea and its implementation.

We note that our closure result, when combined with the result of [132], yields closure under input product and output product with regular weighted string languages. This combined result is even effective, but most likely inefficient. Consequently, a possible future contribution might be an algorithm specifically tailored to the input product with a regular weighted string language. To the author's knowledge, such an algorithm has not yet been considered, for existing contributions only consider special cases [178, 141, 149, 52].

One might also explore the possibility of variable-deleting WSCFTGs, i.e., where a variable x_j can be omitted in the input or output tree of a transition. The STSGs of [62] permit this kind of variable deletion.

Furthermore, it might be interesting to consider alternative approaches to computing the product WSCFTG. For instance, particularly if M is bu-det, one might explore $M \lhd G$ bottom-up, as in a productivity analysis. In addition, one could incorporate pruning. Roughly speaking, pruning amounts to partitioning the set of items and imposing a bound on the size of each block. Such a technique has already been presented in [39] for the cube-pruning algorithm.

4 Generic binarization of weighted grammars

This chapter is a greatly expanded version of [29].

4.1 Introduction

In natural-language processing and statistical machine translation (SMT), the tasks of parsing and decoding play an important role. Both tasks can be described conveniently using intersection-like operations, e.g., the intersection of a context-free language with a regular language [9]. The complexity of the corresponding product construction is usually exponential in the *rank* of one of the grammars, i.e., the maximum number of nonterminal occurrences in the right-hand side of any rule. Consequently, we obtain substantially better parsing and decoding efficiency if we can transform the grammar into an equivalent grammar of lower rank. *Binarizing* a grammar, in particular, means transforming it into an equivalent grammar of rank at most 2.

It will be helpful to view binarization as the application of a binarization mapping; roughly speaking, a binarization mapping is a partial mapping from a grammar formalism into itself that preserves meaning and reduces the rank to 2. A common way to construct a binarization mapping might be dubbed "rule by rule", as known from the Chomsky-normal-form transformation for context-free grammars (CFGs). In this setting, we replace each rule of rank greater than 2 by an equivalent collection of rules of rank 2. For instance, given a rule of rank 4 such as

$$A \to BCDE$$

we might introduce new nonterminals $[[BC]D]$ and $[BC]$ and replace the rule by

$$A \to [[BC]D]E\ , \qquad [[BC]D] \to [BC]D\ , \qquad [BC] \to BC\ .$$

This way, the rule-by-rule technique replaces each rule of rank k, $k > 2$, by $k - 1$ rules of rank 2. This increase in the number of rules is reasonable because it still improves parsing complexity. In general, we expect binarization mappings to be reasonable in this sense, but we do not formalize this requirement for the sake of simplicity.

We can classify any binarization mapping with respect to, in ascending weakness,

publication	formalism	totality	completeness
[45]	CFG	yes	yes
[159]	LCFRS	yes*	yes*
[97]	SCFG	no	no
[147]	STAG	no	no

* if increased fanout is permitted

Table 4.1: Results concerning rule-by-rule complete binarization mappings.

- totality: the domain contains every grammar;
- completeness: the domain contains every grammar that has an equivalent representation of rank 2;
- rule-by-rule completeness: the domain contains every grammar such that for every rule of rank greater than 2, there is an equivalent collection of rules of rank at most 2.

The third property is obtained when we use the above rule-by-rule technique. It is the state of the art; and rule-by-rule complete binarization mappings (RCBMs) have been defined for several formalisms, such as CFGs, linear context-free rewriting systems (LCFRSs, [180]), synchronous CFGs (SCFGs, [119, 39]), and synchronous tree-adjoining grammars (STAGs, [171]). In some cases, the RCBM is even total; see Tab. 4.1 for the details. For SCFGs and STAGs, it is not surprising that the respective RCBM is *not* total, because such a binarization mapping does not exist [2].

In this chapter, we consider a generic approach for deriving an RCBM for some grammar formalism. At the core of this approach is an algorithm that can be adapted to a new formalism by changing a parameter at runtime. Thus the algorithm needs to be implemented only once and can then be reused for a variety of formalisms. As a proof of concept, we derive RCBMs for two formalisms, namely hedge-to-string transducers (which encompass tree-to-string transducers) and weighted synchronous context-free hedge grammars (which encompass the WSCFTGs of Ch. 3), and we review how the former RCBM performed on a large hedge-to-string transducer for English-German SMT. To the author's knowledge, these RCBMs are the first ones for these cases.

As a theoretical foundation we use interpreted regular tree grammars (IRTG, [112]). IRTGs subsume many grammar formalisms encountered in SMT models, among them all those mentioned so far. We proceed in the following five steps. First, we define a weighted version of IRTGs (Sec. 4.2). Second, we use IRTG terminology to for-

malize the concepts "binarization mapping", "complete", and "rule-by-rule complete" (Sec. 4.3). Third, we define a simple "template" that gives rise to a class of RCBMs for IRTGs (Sec. 4.4). However, these mappings are not computable per se. Fourth, we therefore "outsource the noncomputable part" to the user; i.e., we introduce the above-mentioned parameter, called *b-rule*. We thus arrive at a template for a class of efficiently computable binarization mappings for IRTGs, and we define a condition with respect to the b-rules that guarantees that these binarization mappings be rule-by-rule complete (Sec. 4.5). Fifth, and last, we consider how these RCBMs for IRTGs can be used to derive RCBMs for established formalisms (Sec. 4.6).

We end this chapter with a conclusion, discussion, and outlook (Sec. 4.7).

4.2 Interpreted regular tree grammars

Grammar formalisms employed in parsing and SMT, such as those mentioned in the introduction, differ in the derived objects – e.g., strings, trees, and graphs – and the operations involved in the derivation – e.g., concatenation, substitution, and adjoining. Interpreted regular tree grammars (IRTGs) permit a uniform treatment of many of these formalisms. To this end, IRTGs combine the following two concepts:

Algebras IRTGs represent the objects and operations symbolically using terms; the object in question is obtained by interpreting each symbol in the term as a function. In the parlance of universal-algebra theory, we are employing *initial-algebra semantics* [86].

Tree homomorphisms IRTGs separate the finite control (state behavior) of a derivation from its derived object (in its term representation; generational behavior); the former is captured by a recognizable tree language, while the latter is obtained by applying a tree homomorphism. This idea goes back to the *tree bimorphisms* of [6].

Now we define the concept of IRTG formally (cf. Fig. 4.1).

A *(linear, nondeleting) tree homomorphism* is a mapping $h\colon T_\Gamma(X) \to T_\Delta(X)$ that satisfies the following condition: there is a mapping $g\colon \Gamma \to T_\Delta(X)$ such that (i) $g(\sigma) \in C_\Delta(X_k)$ for every $\sigma \in \Gamma^{(k)}$, (ii) $h(\sigma(t_1, \ldots, t_k))$ is the tree obtained from $g(\sigma)$ by replacing the occurrence of x_j by $h(t_j)$, and (iii) $h(x_j) = x_j$. This extends the usual definition of linear and nondeleting homomorphisms [80] to trees with variables. Note that $h(\sigma(x_1, \ldots, x_k)) = g(\sigma)$ for every $\sigma \in \Gamma^{(k)}$. We abuse notation and write $h(\sigma)$ for $g(\sigma)$ for every $\sigma \in \Gamma$.

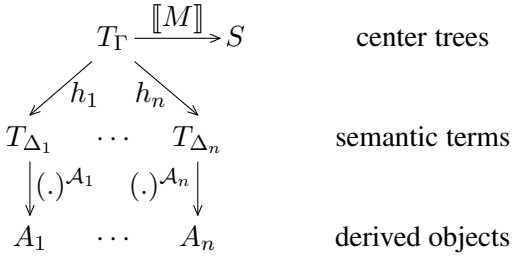


Figure 4.1: Overview of the concept IRTG.

Let $\mathcal{S}$ be a semiring and $\Delta = (\Delta_1, \ldots, \Delta_n)$ a sequence of ranked alphabets. An *interpreted regular tree grammar (IRTG) over* Δ *and* $\mathcal{S}$ is a triple $B = (\Gamma, M, h)$ where Γ is a ranked alphabet (*control alphabet*), M is a WTA over Γ and $\mathcal{S}$, and $h = (h_1, \ldots, h_n)$ is a sequence such that $h_i \colon T_\Gamma(X) \to T_{\Delta_i}(X)$ is a tree homomorphism.

Let $B = (\Gamma, M, h)$ be an IRTG over Δ and $\mathcal{S}$ and $M = (Q, R, \mu, \nu)$. We call the trees in $L(M)$ *center trees*. A *rule of* B is a transition of M, and the *rank* $\mathrm{rk}(B)$ *of* B is $\max\{\mathrm{rk}(\rho) \mid \rho \in R\}$. We define the meaning of B with respect to given algebras. For this, let $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_n)$ be a sequence such that $\mathcal{A}_i$ is a Δ_i-algebra. We say that B is $\mathcal{A}$*-admissible* if $\mathcal{S}$ is complete or $\{\xi \mid \xi \in L(M), \forall i \colon h_i(\xi)^{\mathcal{A}_i} = a_i\}$ is finite for every $(a_1, \ldots, a_n)$. Let B be $\mathcal{A}$-admissible. Then the $\mathcal{A}$*-meaning* $[\![B]\!]_\mathcal{A}$ *of* B is

$$[\![B]\!]_\mathcal{A} \colon A_1 \times \ldots \times A_n \to S\,, \qquad (a_1, \ldots, a_n) \mapsto \textstyle\sum_{\xi \colon \forall i \colon h_i(\xi)^{\mathcal{A}_i} = a_i} [\![M]\!](\xi)\,.$$

We call the terms in $T_{\Delta_i}(X)$ *semantic terms*. We say that two $\mathcal{A}$-admissible IRTGs B and B' are $\mathcal{A}$*-equivalent* if $[\![B]\!]_\mathcal{A} = [\![B']\!]_\mathcal{A}$. Usually we consider $\mathcal{A}$ fixed, and then we omit the subscript from $[\![B]\!]_\mathcal{A}$, and we simply say "admissible" and "equivalent".

Observation 4.2.1 *Let* $\mathcal{B} = (\mathcal{T}_{\Delta_1}(\emptyset), \ldots, \mathcal{T}_{\Delta_n}(\emptyset))$. *Then* $\mathcal{A}$*-admissible implies* $\mathcal{B}$*-admissible and*

$$[\![B]\!]_\mathcal{A}(a_1, \ldots, a_n) = \textstyle\sum_{(t_1, \ldots, t_n) \in T_{\Delta_1} \times \cdots \times T_{\Delta_n} \colon t_i^{\mathcal{A}_i} = a_i} [\![B]\!]_\mathcal{B}(t_1, \ldots, t_n)\,.$$

Consequently, $[\![B]\!]_\mathcal{B} = [\![B']\!]_\mathcal{B}$ *implies* $[\![B]\!]_\mathcal{A} = [\![B']\!]_\mathcal{A}$.

In the case that $\mathcal{S}$ is the Boolean semiring, our IRTGs correspond to original IRTGs in the literature [112]. If $\mathcal{S}$ is the Boolean semiring, $n = 2$, and $\mathcal{A}_i$ is the Δ_i-term algebra, then IRTGs are the tree bimorphisms known from the literature [6]; our use of the letter B for an IRTG can be attributed to this fact.

Example 4.2.2 We consider the following SCFG rule:

$$S \to \alpha_3(PPER, NP)\ , \qquad \text{where } \alpha_3 = \langle x_1 \text{ ließ } x_2 \text{ frei}, x_1 \text{ freed } x_2 \rangle\ .$$

Informally, this rule tells us to derive a pair (w_1, w_2) of strings for the state $PPER$ as well as a pair (w_1', w_2') for the state NP. Then we obtain a pair for S from α_3 by replacing x_1 and x_2 in the first component by w_1 and w_2, respectively, and replacing x_1 and x_2 in the second component by w_1' and w_2', respectively. Now we make this procedure explicit by defining an IRTG.

Let $\Sigma = \{\text{freed}, \text{ließ}, \text{frei}, \dots\}$ be the set of terminal symbols of our SCFG. We consider a ranked alphabet Δ and a Δ-algebra $\mathcal{A}$ with the domain Σ^* that allows us to encode string substitution symbolically. We let

$$\Delta = \{(\mathrm{con}^2)^{(2)}, (\mathrm{con}^3)^{(3)}, (\mathrm{con}^4)^{(4)}, (\mathrm{con}^5)^{(5)}\} \cup \{\sigma^{(0)} \mid \sigma \in \Sigma\}\ ,$$

$$(\mathrm{con}^k)^{\mathcal{A}}(w_1, \dots, w_k) = w_1 \cdots w_k\ , \qquad (k \in \{2,3,4,5\})$$

$$\sigma^{\mathcal{A}} = \sigma\ . \qquad (\sigma \in \Sigma)$$

Then the following terms t_1 and t_2 represent our substitution procedure for the first and second component, respectively:

$$t_1 = \mathrm{con}^4(x_1, \text{ließ}, x_2, \text{frei})\ , \qquad t_2 = \mathrm{con}^3(x_1, \text{freed}, x_2)\ .$$

That is, the pair for S, given w_1, w_2, w_1', w_2' as above, is $(t_1^{\mathcal{A}}(w_1, w_2), t_2^{\mathcal{A}}(w_1', w_2'))$.

Now we define the IRTG. We let $\Gamma = \{\alpha_1^{(1)}, \alpha_2^{(1)}, \alpha_3^{(2)}, \alpha_4^{(2)}, \alpha_5^{(0)}, \alpha_6^{(0)}\}$ and $M = (Q, R, S)$ be the FTA over Γ with $Q = \{S, PPER, NP\}$ and

$$R = \{(NP, \alpha_1, S), (PPER, \alpha_2, S), (PPER\,NP, \alpha_3, S),$$
$$(PPER\,NP, \alpha_4, S), (\varepsilon, \alpha_5, PPER), (\varepsilon, \alpha_6, NP)\}\ .$$

The tree homomorphisms $h_1, h_2 \colon T_\Gamma(X) \to T_\Delta(X)$ are shown in Fig. 4.2.

Finally, we let $B = (\Gamma, M, (h_1, h_2))$; then B is an IRTG over (Δ, Δ) and the Boolean semiring. Since this semiring is complete, B is trivially admissible. We indicate the $(\mathcal{A}, \mathcal{A})$-meaning. To this end, let $\xi = \alpha_4(\alpha_5, \alpha_6)$. Clearly, $\xi \in L(M)$ and

$$h_1(\xi)^{\mathcal{A}} = \text{die katze ließ er frei}\ , \qquad h_2(\xi)^{\mathcal{A}} = \text{he freed the cat}\ .$$

We note that ξ is not the only center tree for this sentence pair, but since we calculate in the Boolean semiring, we are content with one center tree. We conclude that

$$[\![B]\!](\text{er ließ die katze frei}, \text{he freed the cat}) = 1\ .$$

$$\begin{aligned}
\mathrm{con}^4(x_1, \text{ließ}, \text{er}, \text{frei}) &\xleftarrow{h_1} \alpha_1 \xmapsto{h_2} \mathrm{con}^3(\text{he}, \text{freed}, x_1)\\
\mathrm{con}^5(\text{die}, \text{katze}, \text{ließ}, x_1, \text{frei}) &\xleftarrow{h_1} \alpha_2 \xmapsto{h_2} \mathrm{con}^5(x_1, \text{let}, \text{the}, \text{cat}, \text{out})\\
\mathrm{con}^4(x_1, \text{ließ}, x_2, \text{frei}) &\xleftarrow{h_1} \alpha_3 \xmapsto{h_2} \mathrm{con}^3(x_1, \text{freed}, x_2)\\
\mathrm{con}^4(x_2, \text{ließ}, x_1, \text{frei}) &\xleftarrow{h_1} \alpha_4 \xmapsto{h_2} \mathrm{con}^3(x_1, \text{freed}, x_2)\\
\text{er} &\xleftarrow{h_1} \alpha_5 \xmapsto{h_2} \text{he}\\
\mathrm{con}^2(\text{die}, \text{katze}) &\xleftarrow{h_1} \alpha_6 \xmapsto{h_2} \mathrm{con}^2(\text{the}, \text{cat})
\end{aligned}$$

Figure 4.2: Tree homomorphisms h_1 and h_2.

Coincidentally, the IRTG B corresponds to the SCFG from Fig. 1.2.

Technically, we do not need the symbols con^3, con^4, and con^5, because

$$\begin{aligned}
(\mathrm{con}^3)^{\mathcal{A}} &= \mathrm{con}^2(\mathrm{con}^2(x_1, x_2), x_3)^{\mathcal{A}}\ ,\\
(\mathrm{con}^4)^{\mathcal{A}} &= \mathrm{con}^2(\mathrm{con}^3(x_1, x_2, x_3), x_4)^{\mathcal{A}}\ ,\\
(\mathrm{con}^5)^{\mathcal{A}} &= \mathrm{con}^2(\mathrm{con}^4(x_1, x_2, x_3, x_4), x_5)^{\mathcal{A}}\ .
\end{aligned}$$

However, since concatenation is associative, it is unusual to specify an explicit bracketing. After all, the bracketing is rather arbitrary, and we do not want to make our symbolic representation more specific than necessary. □

Example 4.2.2 shows that, even with fixed algebras, IRTGs can offer a high degree of freedom for expressing the generational behavior of a rule; for instance, we can express con^4 by nesting con^2.

Example 4.2.3 (Ex. 4.2.2 contd.) We modify B in the slightest way: we change h_1 so that the image of α_3 becomes $\mathrm{con}^4(x_2, \text{ließ}, x_1, \text{frei})$. Then $h_1(\alpha_3) = h_1(\alpha_4)$ and $h_2(\alpha_3) = h_2(\alpha_4)$. Clearly, this changes the meaning of B, but not only that: since the α_3-rule and the α_4-rule now describe the same SCFG rule, one might argue that B no longer corresponds to any SCFG, that it rather corresponds to a variant of SCFG whose rules are equipped with multiplicities. □

Example 4.2.3 shows that we have to be careful when we describe an established formalism as a class of IRTGs. The following normal form will be helpful in this respect. Let $B = (\Gamma, M, h)$ be an IRTG over Δ and $\mathcal{S}$, and let $M = (Q, R, \mu, \nu)$. Then we define the IRTG $\psi(B)$ over Δ and $\mathcal{S}$ by letting

- $\psi(B) = (\Gamma', M', h')$,
- $\Gamma' = \big\{(h_i(\alpha) \mid i \in \{1, \ldots, n\}) \;\big|\; \alpha \in \Gamma\big\}$,

- $M' = (Q, R', \mu', \nu)$,
- $R' = \big\{(q_1 \cdots q_l, (h_i(\alpha) \mid i \in \{1, \ldots, n\}), q) \;\big|\; (q_1 \cdots q_l, \alpha, q) \in R\big\}$,
- $\mu'(q_1 \cdots q_l, \gamma, q) = \sum_{\alpha \in \Gamma\colon \gamma_i = h_i(\alpha)} \mu(q_1 \cdots q_l, \alpha, q)$, and
- $h'_i(\gamma) = \gamma_i$ for every $\gamma \in \Gamma'$.

Lemma 4.2.4 *The mapping ψ preserves admissibility and meaning.*

PROOF. We overload the symbol ψ; we let $\psi\colon \Gamma \to \Gamma'$ with $\psi(\alpha) = (h_i(\alpha) \mid i \in \{1, \ldots, n\})$, and likewise for $\psi\colon T_\Gamma \to T_{\Gamma'}$, $\psi\colon R \to R'$, and $\psi\colon T_R \to T_{R'}$. For the preservation of admissibility, one easily proves by induction on the size of a tree that

$$\{\xi \mid \xi \in L(M'), \forall i\colon h'_i(\xi)^{\mathcal{A}_i} = a_i\} \subseteq \psi(\{\xi \mid \xi \in L(M), \forall i\colon h_i(\xi)^{\mathcal{A}_i} = a_i\}) \; .$$

We prove the preservation of meaning. By Obs. 4.2.1 it suffices to show the case that $\mathcal{A} = (\mathcal{T}_{\Delta_1}(\emptyset), \ldots, \mathcal{T}_{\Delta_n}(\emptyset))$. It can be shown using standard techniques (mutual inclusion, induction on the size of a tree) that ($\star$) $\psi(\bigcap_i h_i^{-1}(t_i)) = \bigcap_i h_i'^{-1}(t_i)$ for every sequence $(t_1, \ldots, t_n)$ with $t_i \in T_{\Delta_i}$.

We prove by induction on m that, for every $m \in \mathbb{N}$ and $\xi' \in T_{\Gamma'}$ with $|\mathrm{pos}(\xi')| \leq m$, we have $[\![\xi']\!]_{M'} = \sum_{\xi \in \psi^{-1}(\xi')} [\![\xi]\!]_M$. For the induction base ($m = 0$), there is nothing to show. For the induction step ($m \to m + 1$), we let $m \in \mathbb{N}$ and $\xi' \in T_{\Gamma'}$ with $|\mathrm{pos}(\xi')| \leq m + 1$. Then there are $\gamma \in \Gamma'$ and $\xi'_1, \ldots, \xi'_k$ such that $\xi' = \gamma(\xi'_1, \ldots, \xi'_k)$, and we derive

$$\begin{aligned}
&[\![\gamma(\xi'_1, \ldots, \xi'_k)]\!]_{M'} = [\![\gamma([\![\xi'_1]\!]_{M'}, \ldots, [\![\xi'_k]\!]_{M'})]\!]_{M'} && \text{(Obs. 2.4.4)}\\
&= \textstyle\sum_{\alpha\colon h_i(\alpha) = \gamma_i} [\![\alpha([\![\xi'_1]\!]_{M'}, \ldots, [\![\xi'_k]\!]_{M'})]\!]_M\\
&= \textstyle\sum_{\alpha\colon h_i(\alpha) = \gamma_i} [\![\alpha(\sum_{\xi \in \psi^{-1}(\xi'_1)} [\![\xi]\!]_M, \ldots, \sum_{\xi \in \psi^{-1}(\xi'_k)} [\![\xi]\!]_M)]\!]_M\\
& && \text{(induction hypothesis)}\\
&= \textstyle\sum_{\alpha, \xi_1, \ldots, \xi_k\colon h_i(\alpha) = \gamma_i, \xi_j \in \psi^{-1}(\xi'_j)} [\![\alpha(\xi_1, \ldots, \xi_k)]\!]_M = \sum_{\xi \in \psi^{-1}(\xi')} [\![\xi]\!]_M \; .
\end{aligned}$$

Finally, we derive using ($\star$)

$$\begin{aligned}
&[\![B]\!](t_1, \ldots, t_n) = \textstyle\sum_{\xi \in \bigcap_i h_i^{-1}(t_i)} \sum_q [\![\xi]\!]_M(q) \cdot \nu_q\\
&= \textstyle\sum_q \big(\sum_{\xi \in \bigcap_i h_i^{-1}(t_i)} [\![\xi]\!]_M(q)\big) \cdot \nu_q\\
&= \textstyle\sum_q \big(\sum_{\xi' \in \bigcap_i h_i'^{-1}(t_i)} \sum_{\xi \in \psi^{-1}(\xi')} [\![\xi]\!]_M(q)\big) \cdot \nu_q\\
&= \textstyle\sum_q \big(\sum_{\xi' \in \bigcap_i h_i'^{-1}(t_i)} [\![\xi']\!]_{M'}(q)\big) \cdot \nu_q\\
&= \textstyle\sum_{\xi' \in \bigcap_i h_i'^{-1}(t_i)} \sum_q [\![\xi']\!]_{M'}(q) \cdot \nu_q = [\![\psi(B)]\!](t_1, \ldots, t_n) \; .
\end{aligned}$$

■

Observation 4.2.5 *We have that* $\psi(B) = \psi(\psi(B))$.

Example 4.2.6 (Ex. 4.2.2 contd.) We consider an alternative IRTG B' for the same SCFG. For this, we let $\Delta' = \Gamma$, $B' = (\Gamma, M, (h'_1, h'_2))$, and $h'_i(\alpha) = \alpha(x_1, \ldots, x_k)$ for every $\alpha \in \Gamma^{(k)}$. Then h'_i is merely the identity on $T_\Gamma(X)$. Moreover, we define two Δ'-algebras $\mathcal{A}_1$ and $\mathcal{A}_2$ with domain Σ^* by letting $\alpha^{\mathcal{A}_i} = h_i(\alpha)^{\mathcal{A}}$. One can show that $[\![B]\!]_{(\mathcal{A},\mathcal{A})} = [\![B']\!]_{(\mathcal{A}_1,\mathcal{A}_2)}$. □

Example 4.2.6 shows that, as long as we are free to choose the algebras, we can dispense with the tree homomorphisms. However, the algebras constitute a kind of "black box", and statements about IRTGs will usually depend on external information about this black box. For instance, the upcoming binarization method requires (from the user) information about term equivalence in each algebra. Consequently, it will be beneficial to consider classes of IRTGs with a *fixed* sequence Δ of ranked alphabets, together with a *fixed* sequence $\mathcal{A}$ of algebras. In such a setting, the tree homomorphisms are, of course, essential.

The following corollary is a consequence of Cor. 2.3.2.

Corollary 4.2.7 *Let* $h\colon T_\Gamma(X) \to T_\Delta(X)$ *be a tree homomorphism,* $\mathcal{A}$ *a* Δ*-algebra, and* $\alpha(\xi_1, \ldots, \xi_l) \in T_\Gamma$. *Then*

$$h(\alpha(\xi_1, \ldots, \xi_l))^{\mathcal{A}} = h(\alpha)^{\mathcal{A}}(h(\xi_1)^{\mathcal{A}}, \ldots, h(\xi_l)^{\mathcal{A}}) .$$

For the following observation, we use Obs. 2.2.2.

Observation 4.2.8 *Let* $h\colon T_\Gamma(X) \to T_\Delta(X)$ *be a tree homomorphism and* $f\colon X_k \to T_\Gamma(X)$. *Then* $h(f^\flat(t)) = (h \circ f)^\flat(h(t))$ *for every* $m \in \mathbb{N}$ *and* $t \in T_\Gamma(X_k)$ *with* $|\mathrm{pos}(t)| \leq m$.

Corollary 4.2.9 *Let* $h\colon T_\Gamma(X) \to T_\Delta(X)$ *be a tree homomorphism,* $k, l \in \mathbb{N}$, $\zeta \in T_\Gamma(X_k)$, *and* $\xi_1, \ldots, \xi_k \in T_\Sigma(X_l)$. *Then*

$$h(\zeta[x_1/\xi_1] \cdots [x_k/\xi_k]) = h(\zeta)[x_1/h(\xi_1)] \cdots [x_k/h(\xi_k)] .$$

Combining Cor. 2.3.2 and Cor. 4.2.9, we obtain the following corollary.

Corollary 4.2.10 *Let* $h\colon T_\Gamma(X) \to T_\Delta(X)$ *be a tree homomorphism,* $\mathcal{A}$ *a* Δ*-algebra,* $l \in \mathbb{N}$, $\zeta \in C_\Gamma(X_l)$, *and* $\xi_1, \ldots, \xi_l \in T_\Gamma$. *Then*

$$h(\zeta[x_1/\xi_1] \cdots [x_l/\xi_l])^{\mathcal{A}} = h(\zeta)^{\mathcal{A}}(h(\xi_1)^{\mathcal{A}}, \ldots, h(\xi_l)^{\mathcal{A}}) .$$

$$(BCD, \alpha, A)\ , \quad (\varepsilon, \alpha_1, B)\ , \quad (\varepsilon, \alpha_2, C)\ , \quad (\varepsilon, \alpha_3, D)$$

$$\begin{aligned}
\mathrm{con}^3(x_1, x_2, x_3) &\xleftarrow{h_1} \alpha \xmapsto{h_2} \mathrm{con}^4(x_3, \mathrm{a}, x_1, x_2)\\
\mathrm{b} &\xleftarrow{h_1} \alpha_1 \xmapsto{h_2} \mathrm{b}\\
\mathrm{c} &\xleftarrow{h_1} \alpha_2 \xmapsto{h_2} \mathrm{c}\\
\mathrm{d} &\xleftarrow{h_1} \alpha_3 \xmapsto{h_2} \mathrm{d}
\end{aligned}$$

Figure 4.3: An IRTG of rank 3 encoding an SCFG.

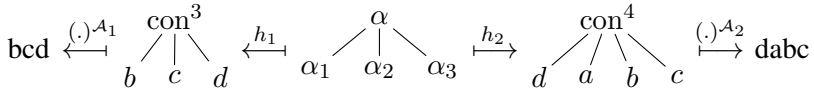


Figure 4.4: Center tree (innermost), semantic terms, derived objects (outermost).

4.3 Binarization mappings

Roughly speaking, our aim is to construct a partial mapping from IRTGs into IRTGs that preserves meaning and reduces the rank to 2, and its domain should contain all IRTGs that can be binarized rule by rule. In this section, we formalize this problem statement. We proceed as follows. First, we consider an example. Second, we define what a binarization of a single rule is, and how the rule is to be replaced. Third, we define the concept of a rule-by-rule complete binarization mapping.

Example 4.3.1 We consider the IRTG shown in Fig. 4.3, which can be viewed as an SCFG in the same way as in Ex. 4.2.2. In particular, we reuse the algebra $\mathcal{A}$. Figure 4.4 shows a center tree with its two homomorphic images, which evaluate to the strings bcd and $dabc$.

Consider the first transition in Fig. 4.3, which has rank three. It occurs in the run

$$(BCD, \alpha, A)\big(B, C, D\big)\ ,$$

which is a partial run on the fragment $\alpha(x_1, x_2, x_3)$ of the center tree in Fig. 4.4. This fragment is mapped to the semantic terms $h_1(\alpha)$ and $h_2(\alpha)$ shown in Fig. 4.3.

Now consider the transitions in Fig. 4.5. These transitions make up the run

$$(A'D, \alpha', A)\big((BC, \alpha'', A')\big(B, C\big), D\big)\ ,$$

which is a partial run on the fragment $\alpha'(\alpha''(x_1, x_2), x_3)$. Let us call this fragment ξ. Note that the terms $h'_1(\xi)$ and $h_1(\alpha)$ are *equivalent* in that they denote the same term

$$(A'D, \alpha', A)\ , \quad (BC, \alpha'', A')$$

$$\mathrm{con}^2(x_1, x_2) \xleftarrow{h'_1} \alpha' \xmapsto{h'_2} \mathrm{con}^2(\mathrm{con}^2(x_2, \mathrm{a}), x_1)$$
$$\mathrm{con}^2(x_1, x_2) \xleftarrow{h'_1} \alpha'' \xmapsto{h'_2} \mathrm{con}^2(x_1, x_2)$$

Figure 4.5: Binarization of the ternary rule in Fig. 4.3.

function, and so are the terms $h'_2(\xi)$ and $h_2(\alpha)$. Thus, replacing the α-transition by the transitions in Fig. 4.5 (and merging h_i and h'_i accordingly) does not change the meaning of the IRTG. However, since the new rules are binary, parsing and translation will be cheaper. □

Rule-by-rule binarization of IRTGs closely follows the intuition laid out in this example: it means processing each suprabinary rule, attempting to replace it with an equivalent collection of binary rules. For the remainder of this chapter (unless noted otherwise), let $\Delta = (\Delta_1, \ldots, \Delta_n)$ be a sequence of ranked alphabets, $\mathcal{S}$ a commutative semiring, and $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_n)$ a sequence such that $\mathcal{A}_i$ is a Δ_i-algebra. Moreover, let $B = (\Gamma, M, h)$ be an $\mathcal{A}$-admissible IRTG over Δ and $\mathcal{S}$, $M = (Q, R, \mu, \nu)$, $\rho \in R$, $\rho = (q_1 \cdots q_k, \alpha, q)$, and $k > 2$.

4.3.1 Binarization of a rule

Let Γ' be a binary ranked alphabet and $\xi \in C_{\Gamma'}(X_k)$. A *ξ-binarization B' of ρ* is an IRTG (Γ', M', h') over Δ and $\mathcal{S}$ such that there is a d with

- $d \in D^q(M', \xi[x_1/q_1] \cdots [x_k/q_k])$,
- $\langle d \rangle = \mu(\rho)$,
- $h'_i(\xi)^{\mathcal{A}_i} = h_i(\alpha)^{\mathcal{A}_i}$,
- $d' \in D^q_{\mathrm{pr}}(M')$ implies $d' \sqsubseteq d$, and
- $D^{q_j}_{\mathrm{pr}}(M') = \{q_j\}$.

We call B' *rank normal* if $\Gamma' = \Gamma'^{(2)}$.

Example 4.3.2 (Ex. 4.3.1 contd.) Let $\rho = (BCD, \alpha, A)$. A rank-normal ξ-binarization of ρ is given in Fig. 4.5, where

$$\xi = \alpha'(\alpha''(x_1, x_2), x_3)\ ,$$
$$d = (A'D, \alpha', A)\big((BC, \alpha'', A')\big(B, C\big), D\big)\ .$$ □

We note that a binarization of ρ need not exist, even if B as a whole admits an equivalent representation of rank at most 2.

Example 4.3.3 It is easy to specify an SCFG (and, thus, an IRTG) of rank 0 for the singleton language $\{(\mathrm{abcd}, \mathrm{cadb})\}$. Likewise, one can use the following SCFG of rank 4:

$$S \to \langle x_1x_2x_3x_4, x_3x_1x_4x_2\rangle(A, B, C, D)\ ,$$
$$A \to \langle \mathrm{a}, \mathrm{a}\rangle, \quad \ldots, \quad D \to \langle \mathrm{d}, \mathrm{d}\rangle\ .$$

Using an IRTG representation in the spirit of Exs. 4.2.2 and 4.3.1, one will find that there is no binarization of the first rule. We will elaborate on this in Ex. 4.4.12, when we will have the appropriate tools at our disposal. □

Let $B' = (\Gamma', M', h')$ be a ξ-binarization of ρ and $M' = (Q', R', \mu', \nu')$. We say that B and B' are *compatible* if the following conditions are satisfied:

- $R'|_q \cap R = \emptyset$,
- $q' \in Q \cap (Q' \setminus \{q, q_1, \ldots, q_k\})$ implies $R|_{q'} = R'|_{q'}$, and
- $h_i|_{\Gamma\cap\Gamma'} = h'_i|_{\Gamma\cap\Gamma'}$ and $\mu|_{R\cap R'} = \mu'|_{R\cap R'}$.

We note that the property of compatibility can be readily established by using suitable alphabets Q' and Γ'; e.g., one can use fresh symbols ($\Gamma \cap \Gamma' = \emptyset = Q \cap (Q' \setminus \{q, q_1, \ldots, q_k\})$), or one can reuse symbols from Q and Γ wherever possible.

Let B and B' be compatible. Then we define the IRTG $B[\rho/B']$ over Δ and $\mathcal{S}$ by

- $B[\rho/B'] = (\Gamma \cup \Gamma', M[\rho/M'], h'')$,
- $M[\rho/M'] = (Q \cup Q', (R \cup R') \setminus \{\rho\}, \mu'', \nu'')$ where

$$\mu''(\rho') = \begin{cases} \mu(\rho') & \text{if } \rho' \in R, \\ \mu'(\rho') & \text{if } \rho' \in R' \setminus R, \end{cases} \qquad \nu''_{q'} = \begin{cases} \nu_{q'} & \text{if } q' \in Q, \\ 0 & \text{otherwise,} \end{cases}$$

- $h''_i \colon T_{\Gamma\cup\Gamma'}(X) \to T_{\Delta_i}(X)$ is the tree homomorphism with

$$h''_i(\alpha) = \begin{cases} h_i(\alpha) & \text{if } \alpha \in \Gamma, \\ h'_i(\alpha) & \text{if } \alpha \in \Gamma' \setminus \Gamma. \end{cases}$$

In the following, we will omit the subscript from π_Γ and $\pi_{\Gamma\cup\Gamma'}$.

Lemma 4.3.4 *There is a* $(h_q \mid q \in Q)$ *such that* $h_q \colon D^q_{co}(M) \to D^q_{co}(M[\rho/M'])$ *is bijective and, for every* $d \in D^q_{co}(M)$*,*

$$h_i(\pi(d))^{\mathcal{A}_i} = h''_i(\pi(h_q(d)))^{\mathcal{A}_i} \qquad \textit{and} \qquad \langle d \rangle = \langle h_q(d) \rangle \,.$$

PROOF. Since B' is a ξ-binarization, there is a $d_\rho \in D^q(M', \xi[x_1/q_1] \cdots [x_k/q_k])$ with the properties mentioned in the definition of a ξ-binarization. We let $R'' = (R \cup R') \setminus \{\rho\}$, $w_1, \ldots, w_k$ be the positions of $x_1, \ldots, x_k$ in ξ, respectively, and $h' \colon T_R(X) \to T_{R''}(X)$ be the tree homomorphism with

$$\begin{aligned} h'(\rho) &= d_\rho[x_1]_{w_1} \cdots [x_k]_{w_k} \,, \\ h'(\rho') &= \rho'(x_1, \ldots, x_{\mathrm{rk}(\rho')}) \,. \qquad (\rho' \neq \rho) \end{aligned}$$

We will prove the statement $P(n)$ for every $n \in \mathbb{N}$, where

$P(n)$: Let $p \in Q$.

1. Let $d \in D^p_{\mathrm{co}}(M)$ with $|\mathrm{pos}(d)| \leq n$. Then $h'(d) \in D^p_{\mathrm{co}}(M[\rho/M'])$, $h_i(\pi(d))^{\mathcal{A}_i} = h''_i(\pi(h'(d)))^{\mathcal{A}_i}$, and $\langle d \rangle = \langle h'(d) \rangle$.
2. Let $d_1, d_2 \in D^p_{\mathrm{co}}(M)$ with $|\mathrm{pos}(d_1)| + |\mathrm{pos}(d_2)| \leq n$. Then $h'(d_1) = h'(d_2)$ implies $d_1 = d_2$.
3. Let $d' \in D^p_{\mathrm{co}}(M[\rho/M'])$ with $|\mathrm{pos}(d')| \leq n$. Then there is a $d \in D^p_{\mathrm{co}}(M)$ with $h'(d) = d'$.

With this statement, it is clear that we obtain the desired mapping h_q for every $q \in Q$ simply by restricting h' appropriately.

For the induction base ($n = 0$), there is nothing to show. We show the induction step ($n \to n + 1$). To this end, let $n \in \mathbb{N}$ such that $P(n)$ holds. We show that $P(n + 1)$ holds. To this end, let $p \in Q$.

Statement 1: Let $d \in D^p_{\mathrm{co}}(M)$ and $|\mathrm{pos}(d)| \leq n + 1$. Then there are $\rho' \in R$, $\rho' = (p_1 \cdots p_k, \alpha, p)$, and $d_1, \ldots, d_k$ such that $d = \rho'(d_1, \ldots, d_k)$, $d_j \in D^{p_j}_{\mathrm{co}}(M)$, and $|\mathrm{pos}(d_j)| \leq n$. By the induction hypothesis (i.e., $P(n)$ holds), we have that

- $h'(d_j) \in D^{p_j}_{\mathrm{co}}(M[\rho/M'])$,
- $h_i(\pi(d_j))^{\mathcal{A}_i} = h''_i(\pi(h'(d_j)))^{\mathcal{A}_i}$, and
- $\langle d_j \rangle = \langle h'(d_j) \rangle$.

We distinguish two cases. The case that $\rho' \neq \rho$ is easy. We turn to the case that $\rho' = \rho$. Then $p = q$. It is easy to see that $h'(d) \in D^q_{\text{co}}(M[\rho/M'])$. We derive

$$\begin{aligned}
&h_i(\pi(d))^{\mathcal{A}_i} = h_i(\pi(\rho(d_1,\dots,d_k)))^{\mathcal{A}_i} = h_i(\alpha(\pi(d_1),\dots,\pi(d_k)))^{\mathcal{A}_i} \\
&= \big[h_i(\alpha)\big(h_i(\pi(d_1)),\dots,h_i(\pi(d_k))\big)\big]^{\mathcal{A}_i} \\
&= h_i(\alpha)^{\mathcal{A}_i}\big(h_i(\pi(d_1))^{\mathcal{A}_i},\dots,h_i(\pi(d_k))^{\mathcal{A}_i}\big) && \text{(Cor. 4.2.7)} \\
&= h_i''(\pi(h'(\rho)))^{\mathcal{A}_i}\Big(h_i''(\pi(h'(d_1)))^{\mathcal{A}_i},\dots,h_i''(\pi(h'(d_k)))^{\mathcal{A}_i}\Big) && (\dagger) \\
&= h_i''(\pi(h'(d)))^{\mathcal{A}_i}\,, && \text{(Cor. 4.2.10)}
\end{aligned}$$

where, for (†), we derive $h_i(\alpha)^{\mathcal{A}_i} = h_i'(\xi)^{\mathcal{A}_i} = h_i'(\pi(h'(\rho)))^{\mathcal{A}_i}$. Finally, we derive

$$\begin{aligned}
\langle h'(d)\rangle &= \langle h'(d_1)\rangle \cdots \langle h'(d_k)\rangle \cdot \langle d_\rho\rangle && \text{(commutativity)} \\
&= \langle d_1\rangle \cdots \langle d_k\rangle \cdot \mu(\rho) = \langle d\rangle\,.
\end{aligned}$$

Statement 2: Let $d_1, d_2 \in D^p_{\text{co}}(M)$, $|\text{pos}(d_1)| + |\text{pos}(d_2)| \leq n+1$, and $h'(d_1) = h'(d_2)$. We distinguish three cases. Case 1: Let $d_1(\varepsilon), d_2(\varepsilon) \neq \rho$ or $d_1(\varepsilon) = d_2(\varepsilon) = \rho$. Then it is easy to apply the induction hypothesis. Case 2: Let $d_1(\varepsilon) = \rho$ and $d_2(\varepsilon) \neq \rho$. Then $h'(d_1)(\varepsilon) \in R'|_q$ and $h'(d_2)(\varepsilon) \in R$. Since M and M' are compatible, $R'|_q \cap R = \emptyset$. So this case does not occur. Case 3: Let $d_1(\varepsilon) \neq \rho$ and $d_2(\varepsilon) = \rho$. For reasons of symmetry, this case does not occur either.

Statement 3: Let $d' \in D^p_{\text{co}}(M[\rho/M'])$ with $|\text{pos}(d')| \leq n+1$. We distinguish two cases. Case 1: Let $d'(\varepsilon) \in R$. Moreover, let $d'(\varepsilon) = (p_1 \cdots p_k, \alpha, p)$. Then $p_1,\dots,p_k \in Q$ and, by the induction hypothesis, there are $d_1,\dots,d_k$ with $h'(d_j) = d'|_j$. We construct $d = d'(\varepsilon)\big(d_1,\dots,d_k\big)$. It is easy to see that $d \in D^p_{\text{co}}(M)$ and $h'(d) = d'$. Case 2: Let $d'(\varepsilon) \notin R$. Since $p \in Q$, and since M and M' are compatible, we have that $p = q$. Recall that, for every $d'' \in D^q_{\text{pr}}(M')$, we have that $d'' \sqsubseteq d_\rho$. Since the run d' is complete, we have that $d_\rho \sqsubseteq d'$, i.e., there are $d_1',\dots,d_k'$ such that $d' = d_\rho[d_1']_{w_1} \cdots [d_k']_{w_k}$. By the induction hypothesis, we obtain that there are $d_1,\dots,d_k$ with $h(d_j) = d_j'$. We construct $d = \rho(d_1,\dots,d_k)$. ∎

Corollary 4.3.5 *Let ρ be a suprabinary rule of B and B' a binarization of ρ. Then $B[\rho/B']$ is admissible and equivalent to B.*

PROOF. Let $(h_q \mid q \in Q)$ be a family of mappings as postulated in Lm. 4.3.4. Using said lemma, it is easy to see that, for every $q \in Q$ and $(a_1,\dots,a_n) \in A_1 \times \cdots \times A_n$,

$$\begin{aligned}
&h_q(\{d \mid d \in D^q_{\text{co}}(M[\rho/M']), \forall i\colon h_i''(\pi(d))^{\mathcal{A}_i} = a_i\}) \\
&= \{d \mid d \in D^q_{\text{co}}(M), \forall i\colon h_i(\pi(d))^{\mathcal{A}_i} = a_i\}\,.
\end{aligned}$$

This implies that $B[\rho/B']$ is admissible.

Moreover, let $(a_1, \ldots, a_n) \in A_1 \times \cdots \times A_n$. Then

$$
\begin{aligned}
&[\![B]\!](a_1, \ldots, a_n) = \textstyle\sum_{\xi \in T_\Gamma \colon h_i(\xi)^{\mathcal{A}_i} = a_i} \sum_{q \in Q, d \in D^q(M,\xi)} \langle d \rangle \cdot \nu_q \\
&= \textstyle\sum_{\xi \in T_\Gamma, q \in Q, d \in D^q(M,\xi) \colon h_i(\xi)^{\mathcal{A}_i} = a_i} \langle d \rangle \cdot \nu_q \\
&= \textstyle\sum_{q \in Q, d \in D^q_{\mathrm{co}}(M) \colon h_i(\pi(d))^{\mathcal{A}_i} = a_i} \langle d \rangle \cdot \nu_q \\
&= \textstyle\sum_{q \in Q, d \in D^q_{\mathrm{co}}(M) \colon h''_i(\pi(h_q(d)))^{\mathcal{A}_i} = a_i} \langle h_q(d) \rangle \cdot \nu_q \\
&= \textstyle\sum_{q \in Q, d' \in D^q_{\mathrm{co}}(M') \colon h''_i(\pi(d'))^{\mathcal{A}_i} = a_i} \langle d' \rangle \cdot \nu''_q \qquad (h_q \text{ is bijective}) \\
&= \textstyle\sum_{\xi \in T_{\Gamma \cup \Gamma'} \colon h''_i(\xi)^{\mathcal{A}_i} = a_i} \sum_{q \in Q, d' \in D^q(M', \xi)} \langle d' \rangle \cdot \nu''_q \\
&= \textstyle\sum_{\xi \in T_{\Gamma \cup \Gamma'} \colon h''_i(\xi)^{\mathcal{A}_i} = a_i} \sum_{q \in Q \cup Q', d' \in D^q(M', \xi)} \langle d' \rangle \cdot \nu''_q = [\![B[\rho/B']]\!](a_1, \ldots, a_n) \,. \blacksquare
\end{aligned}
$$

4.3.2 Binarization mappings

Now we are able to formally define binarization mappings for IRTGs. To this end, let $\mathcal{C}$ be a set of admissible IRTGs over Δ and $\mathcal{S}$. A *binarization mapping bin for $\mathcal{C}$* is a partial mapping $bin \colon \mathcal{C} \to \mathcal{C}$ that preserves meaning. The *binarization domain* $\mathrm{bdom}(bin)$ *of bin* is the set $\{B \mid B \in \mathrm{dom}(bin), \mathrm{rk}(bin(B)) \leq 2\}$. A binarization mapping bin is *complete* if $\mathrm{bdom}(bin) \supseteq \{B \mid B \in \mathcal{C}, \exists B' \in \mathcal{C} \colon [\![B]\!] = [\![B']\!], \mathrm{rk}(B') \leq 2\}$. It is *rule-by-rule complete* if $\mathrm{bdom}(bin)$ contains every $B \in \mathcal{C}$ such that for every suprabinary rule ρ of B there is a binarization of ρ.

We distinguish between the domain of bin and its binarization domain so as to enable "best-effort binarization"; i.e., even if, for some $B \in \mathcal{C}$, we do not find an equivalent B' of rank 2, we can at least attempt to reduce the number of suprabinary rules. In this case $B \in \mathrm{dom}(bin) \setminus \mathrm{bdom}(bin)$. This case can be useful in practice; our theoretical considerations, however, are limited to $\mathrm{bdom}(bin)$.

We abbreviate "rule-by-rule complete binarization mapping" by RCBM. If an RCBM bin exists, then every complete binarization mapping is also rule-by-rule complete, for then $bin(B)$ is a witness that B is in the binarization domain of every complete binarization mapping. Such an RCBM bin, however, need not exist; for instance, the class $\mathcal{C}$ may be so severely restricted that replacing a suprabinary rule by its binarization leads out of $\mathcal{C}$.

4.4 Constructing a binarization mapping

In this section, we construct an RCBM for IRTGs over Δ and $\mathcal{S}$. First, we consider an example of our construction, then we define the necessary concepts, and finally, we

arrive at the construction. The binarization mapping will not be computable in general, and we will tackle this problem in the next section.

Example 4.4.1 (Ex. 4.3.1 contd.) Now we construct the binarization of our rule systematically. We proceed as follows (cf. Fig. 4.6). For each of the terms $h_1(\alpha)$ and $h_2(\alpha)$ (Fig. 4.6a), we consider all terms that satisfy two properties (Fig. 4.6b): (i) they are equivalent to $h_1(\alpha)$ and $h_2(\alpha)$, respectively, and (ii) at each node at most two subtrees contain variables. As Fig. 4.6 suggests, there may be several different terms of this kind. For each of these terms, we analyze the bracketing of variables, obtaining what we call a *variable tree* (Fig. 4.6c). Now we pick terms t_1 and t_2 corresponding to $h_1(\alpha)$ and $h_2(\alpha)$, respectively, such that (iii) they have the same variable tree, say τ. We construct a tree ξ from τ by a simple relabeling, and we read off the tree homomorphisms h'_1 and h'_2 from a decomposition we perform on t_1 and t_2, respectively; see Fig. 4.6, dotted arrows, and compare the boxes in Fig. 4.6d with the homomorphisms in Fig. 4.5. Now the rules in Fig. 4.5 are easily extracted from ξ.

With the tree ξ, the rules, and the tree homomorphisms, we have all ingredients for a ξ-binarization; and, indeed, we obtain a binarization: our rules are equivalent to the original one because of (i); they are binary because ξ is binary, which in turn holds because of (ii); finally, the decompositions of t_1 and t_2 are compatible with ξ because of (iii). We call a sequence (t_1, t_2) a *binarization hedge* if (i)–(iii) are satisfied. We will see below that the existence of a binarization is tantamount to the existence of a binarization hedge. Our main task will be finding a binarization hedge. □

4.4.1 Variable trees and term decomposition

Let us define the concept of variable trees as well as the decomposition that corresponds to the two outer dotted arrows in Fig. 4.6. In order to keep notation uncluttered, we will disregard the IRTG B in this subsection and rather proceed in a general setting. To this end, let Δ be an arbitrary ranked alphabet and $t \in T^{lin}_{\Delta}(X)$.

We begin with a few auxiliary concepts. By $\mathrm{var}(t)$ we denote the set of all elements of X that occur in t, i.e., $\mathrm{var}(t) = \{t(w) \mid t(w) \in X\}$. Let $t_1, \ldots, t_l \in T^{lin}_{\Delta}(X)$. We call this sequence *eligible (for a canonical sort)* if (i) $\mathrm{var}(t_j) \neq \emptyset$ and (ii) $\mathrm{var}(t_j) \cap \mathrm{var}(t_{j'}) \neq \emptyset$ implies $j = j'$. Let $t_1, \ldots, t_l$ be eligible. Then the *canonical sort* $\mathrm{csort}(t_1, \ldots, t_l)$ *of* $t_1, \ldots, t_l$ is the sequence obtained from $t_1, \ldots, t_l$ by sorting the trees according to the least variable index. For instance, we have that

$$\mathrm{csort}(\delta'(x_3), \delta(x_2, x_4)) = \big(\delta(x_2, x_4), \delta'(x_3)\big) ,$$

because the least variable index is 3 in the first argument tree, it is 2 in the second, and $3 > 2$, so we have to swap the trees. We call the sequence $t_1, \ldots, t_l$ *canonically sorted* if it is equal to its canonical sort.

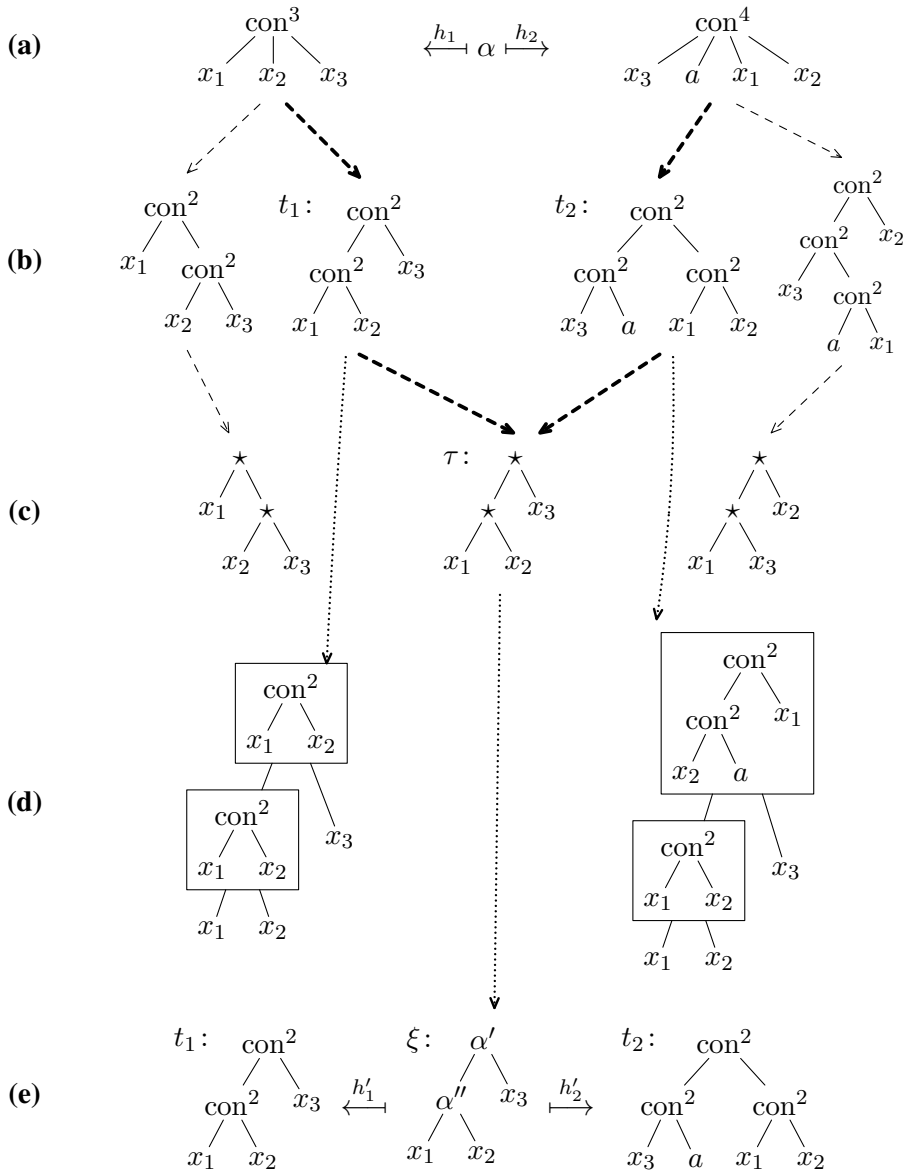


Figure 4.6: Outline of the binarization algorithm.

We define the *variable tree* $v(t)$ *of* t by induction. For this, we let $v\colon T_\Delta^{lin}(X) \to T_{\{\star,\emptyset\}}^{lin}(X)$ with

$$v(x_j) = x_j\ ,$$
$$v(\delta(t_1,\dots,t_k)) = \begin{cases} \emptyset & \text{if } l = 0, \\ v(t'_1) & \text{if } l = 1, \\ \star(v(t'_1),\dots,v(t'_l)) & \text{otherwise,} \end{cases} \qquad (\delta \in \Delta)$$

where $t'_1,\dots,t'_l$ is the canonical sort of the sequence that is obtained from $t_1,\dots,t_k$ by removing every occurrence of any tree that does not contain any variable; since t is linear, that sequence is eligible.

Example 4.4.2 Let t_1, t_2, and τ be given by Fig. 4.6. Then $v(t_1) = \tau = v(t_2)$. □

Observation 4.4.3 *(i)* $\mathrm{var}(t) = \emptyset$ *iff* $v(t) = \emptyset$ *and (ii)* $\mathrm{var}(t) = \mathrm{var}(v(t))$.

Next we show that applying a tree homomorphism to a binary tree preserves the variable tree.

Lemma 4.4.4 *Let* $h\colon T_\Gamma(X) \to T_\Delta(X)$ *a tree homomorphism. For every* $m \in \mathbb{N}$ *and binary* $\xi \in T_\Gamma^{lin}(X)$ *with* $|\mathrm{pos}(\xi)| \le m$*, we have* $v(\xi) = v(h(\xi))$.

PROOF. By induction on m. For the induction base ($m = 0$), there is nothing to show. For the induction step ($m \to m+1$), let $m \in \mathbb{N}$ and $\xi \in T_\Gamma^{lin}(X)$ be binary with $|\mathrm{pos}(\xi)| \le m+1$. We distinguish three cases. Case 1: $\xi \in X$. Trivial.

Case 2: $\xi(\varepsilon) \in \Gamma^{(2)}$. There are $\alpha \in \Gamma$ and $\xi_1, \xi_2 \in T_\Sigma^{lin}(X)$ with $\xi = \alpha(\xi_1,\xi_2)$. We let $v_1 = v(\xi_1)$ and $v_2 = v(\xi_2)$, and we define the following predicates:

$$P(j) \iff \mathrm{var}(\xi_j) \neq \emptyset\ , \qquad P(j,w) \iff P(j) \wedge x_j \text{ occurs in } h(\alpha)|_w\ .$$

Note that $P(j) \iff P(j,\varepsilon)$. We derive

$$\begin{aligned} v(\xi) &= v(\alpha(\xi_1,\xi_2)) \\ &= \begin{cases} \emptyset & \neg P(1) \wedge \neg P(2) \\ v_1 & P(1) \wedge \neg P(2) \\ v_2 & \neg P(1) \wedge P(2) \\ \star\big(\mathrm{csort}(v_1,v_2)\big) & P(1) \wedge P(2) \end{cases} \\ &= v(h(\alpha)|_\varepsilon[x_1/h(\xi_1)][x_2/h(\xi_2)]) \qquad (\dagger) \\ &= v(h(\alpha)[x_1/h(\xi_1)][x_2/h(\xi_2)]) \\ &= v(h(\alpha(\xi_1,\xi_2))) = v(h(\xi))\ , \end{aligned}$$

where (†) follows from the following statement, which is easily proved by induction: for every $m' \in \mathbb{N}$ and $w \in \mathrm{pos}(h(\alpha))$ with $|\mathrm{pos}(h(\alpha)|_w)| \leq m'$, we have that

$$v\big(h(\alpha)|_w[x_1/h(\xi_1)][x_2/h(\xi_2)]\big) = \begin{cases} \emptyset & \neg P(1,w) \wedge \neg P(2,w) \\ v_1 & P(1,w) \wedge \neg P(2,w) \\ v_2 & \neg P(1,w) \wedge P(2,w) \\ \star\big(\mathrm{csort}(v_1,v_2)\big) & P(1,w) \wedge P(2,w) \end{cases}$$

The outer induction hypothesis, $v(\xi_j) = v(h(\xi_j))$, is needed when $h(\alpha)|_w = x_j$.

Case 3: $\xi(\varepsilon) \in \Gamma^{(0)} \cup \Gamma^{(1)}$; similar to Case 2. ∎

Example 4.4.5 (Ex. 4.4.2 contd.) Lemma 4.4.4 requires that ξ be binary. We consider a tree homomorphism h and a suprabinary ξ where $v(\xi) \neq v(h(\xi))$. To this end, we assume a ternary α that is mapped by h to t_2. We let $\xi = \alpha(x_1, x_2, x_3)$. Then $h(\xi) = t_2$, $v(t_2) = \tau$, and $v(\xi) = \star(x_1, x_2, x_3)$.

Let Δ_1 and Δ_2 be ranked alphabets, $t_1 \in T_{\Delta_1}(X)$, and $t_2 \in T_{\Delta_2}(X)$. We say that t_1 and t_2 are *congruent* if $\mathrm{pos}(t_1) = \mathrm{pos}(t_2)$ and, for every $w \in \mathrm{pos}(t_1)$ and $j \in \mathbb{N}$, $t_1(w) = x_j$ iff $t_2(w) = x_j$.

Next we define the *term decomposition* $f(t)$ *of* t by defining the mapping f. For instance, the term decompositions of t_1 and t_2 are shown in Fig. 4.6(d). Our aim is to make $f(t)$ and $v(t)$ congruent, so that we can read off $h'_1, \ldots, h'_n$ from $f(t_1), \ldots, f(t_n)$.

For the range of f, we stretch the notion of a ranked alphabet and permit an infinite set of symbols. Ultimately we will apply f only to a finite number of trees, and for those instances a finite set of symbols suffices, but that set is cumbersome to describe in advance. Let $\Delta' = \{t^{(k)} \mid t \in C_\Delta(X_k)\}$. We call each element of Δ' a *fragment*.

Before we define $f\colon T^{lin}_\Delta(X) \to T^{lin}_{\Delta'}(X)$ formally, we consider some intuition. For this, let $t \in T^{lin}_\Delta(X)$, $t = \delta(t_1, \ldots, t_k)$, and $\delta \in \Delta$. Then we construct the root label of $f(t)$ from the fragment $\delta(x_1, \ldots, x_k)$ in two steps. First, for every j such that t_j does not contain any variable, we replace x_j by t_j. The resulting fragment contains the variables $x_{j_1}, \ldots, x_{j_l}$ where $t_{j_1}, \ldots, t_{j_l}$ are the trees that do contain variables. With the second step, we avoid nodes of rank 1, because those are not present in a variable tree, and we want to achieve congruency. So, if $l = 1$ and the root fragment of $f(t_{j_1})$ is not a variable itself, we replace x_{j_1} by that fragment. If $l \neq 1$, then for every ι such that the root fragment of $f(t_{j_\iota})$ is unary, we replace x_{j_ι} by that fragment. The successors of the root of $f(t)$ are computed accordingly.

Formally, we define $f\colon T^{lin}_\Delta(X) \to T^{lin}_{\Delta'}(X)$ inductively as follows. Let $t = x_j$. Then $f(t) = x_j$. Let $t = \delta(t_1, \ldots, t_k)$, $\delta \in \Delta$, and $t'_1, \ldots, t'_l$ be the canonical sort of

the sequence that is obtained from $t_1, \ldots, t_k$ by removing every occurrence of any tree that does not contain any variable. There are uniquely determined $j_1, \ldots, j_l$ such that $t'_\iota = t_{j_\iota}$ for every $\iota \in \{1, \ldots, l\}$. Moreover, let $u_1, \ldots, u_l$ with $u_\iota = f(t'_\iota)$. We proceed by case distinction.

1. If $l = 1$ and $u_1(\varepsilon) \notin X$, then

$$f(t) = \delta(t''_1, \ldots, t''_k)\big(u_1|_1, \ldots, u_1|_{\mathrm{rk}_{u_1}(\varepsilon)}\big)$$

   where t''_{j_1} is $u_1(\varepsilon)$, and $t''_j = t_j$ for $j \neq j_1$.

2. Otherwise, we let

$$f(t) = \delta(t''_1, \ldots, t''_k)\big(u'_1, \ldots, u'_l\big)$$

   where, for every $\iota \in \{1, \ldots, l\}$,

   - if $\mathrm{rk}_{u_\iota}(\varepsilon) = 1$, then $t''_{j_\iota} = u_\iota(\varepsilon)[x_1/x_\iota]$ and $u'_\iota = u_\iota|_1$,
   - if $\mathrm{rk}_{u_\iota}(\varepsilon) \neq 1$, then $t''_{j_\iota} = x_\iota$ and $u'_\iota = u_\iota$,

   and $t''_j = t_j$ for $j \notin \{j_1, \ldots, j_l\}$.

Example 4.4.6 (Ex. 4.4.2 contd.) We show how to compute $f(t_2)$. First, we perform the canonical sort of the sequence $t_2|_1, t_2|_2$, which yields $t_2|_2, t_2|_1$. Second, we compute $u_1 = f(t_2|_2)$ and $u_2 = f(t_2|_1)$.

For u_1, we first observe that the sequence $t_2|_{21}, t_2|_{22}$ is already canonically sorted. We compute $f(t_2|_{21})$ and $f(t_2|_{22})$, which is x_1 and x_2, respectively. Now we construct u_1. Since $\mathrm{rk}_{x_1}(\varepsilon), \mathrm{rk}_{x_2}(\varepsilon) \neq 1$, we do not merge, and we obtain that $u_1 = [\mathrm{con}^2(x_1, x_2)](x_1, x_2)$. For u_2, the sequence $t_2|_{11}$ is also already sorted. We compute $f(t_2|_{11})$, which is x_3. Now we construct u_2. Since $\mathrm{rk}_{x_3}(\varepsilon) \neq 1$, we also do not merge, and we obtain that $u_2 = [\mathrm{con}^2(x_1, a)](x_3)$.

Finally, we construct $f(t_2)$. We have that $\mathrm{rk}_{u_1}(\varepsilon) \neq 1$ and $\mathrm{rk}_{u_2}(\varepsilon) = 1$, so we have to merge once. We derive

$$\begin{aligned} f(t_2) &= [\mathrm{con}^2(u_2(\varepsilon)[x_1/x_2], x_1)](u_1, u_2|_1) \\ &= [\mathrm{con}^2(\mathrm{con}^2(x_2, a), x_1)]\big([\mathrm{con}^2(x_1, x_2)](x_1, x_2), x_3\big)\ . \end{aligned}$$ □

We show that $v(t)$ and $f(t)$ are congruent if $|\mathrm{var}(t)| > 1$, and we show how to construct a tree homomorphism that maps $f(t)$ back to t. To this end, let $\Delta'' \subseteq \Delta'$ be finite. Then we define the tree homomorphism $h_{\Delta''} \colon T_{\Delta''}(X) \to T_\Delta(X)$ by $h_{\Delta''}(\delta) = \delta$ for every $\delta \in \Delta''$. Moreover, we define $\Delta'|_t = \{\delta^{(k)} \mid \delta \in \Delta'^{(k)}, \exists w \in \mathrm{pos}(t) \colon t(w) = \delta\}$.

Lemma 4.4.7 *For every $m \in \mathbb{N}$ and $t \in T_\Delta^{lin}(X)$ with $|\mathrm{pos}(t)| \leq m$, we have the following. If $|\mathrm{var}(t)| > 1$, then $v(t)$ and $f(t)$ are congruent; otherwise, $\mathrm{var}(f(t)) = \mathrm{var}(t)$ and $\mathrm{pos}(f(t)) \subseteq \{\varepsilon, 1\}$. Moreover, for every finite Δ'' with $\Delta'|_{f(t)} \subseteq \Delta'' \subseteq \Delta'$, we have that $h_{\Delta''}(f(t)) = t$.*

PROOF. By induction on m. For the induction base ($m = 0$), there is nothing to show. We show the induction step ($m \to m+1$). To this end, let $m \in \mathbb{N}$ and $t \in T_\Delta^{lin}(X)$ with $|\mathrm{pos}(t)| \leq m + 1$. If $t \in X$, then the statements are easy to see. Let $t = \delta(t_1, \ldots, t_k)$ with $\delta \in \Delta$, and let $t'_1, \ldots, t'_l$, $j_1, \ldots, j_l$, and $u_1, \ldots, u_l$ be as in the definition of f.

Let $|\mathrm{var}(t)| \leq 1$. Then $l \leq 1$. If $l = 0$, then it is easy to see that $f(t) = t$ and, hence, $\mathrm{var}(f(t)) = \emptyset = \mathrm{var}(t)$ and $\mathrm{pos}(f(t)) = \{\varepsilon\}$ (recall that t is the label of the root of $f(t)$). If $l = 1$, then $|\mathrm{var}(t'_1)| = 1$. We derive $\mathrm{var}(f(t)) = \mathrm{var}(f(t'_1)) = \mathrm{var}(t'_1) = \mathrm{var}(t)$. By the induction hypothesis, $\mathrm{pos}(u_1) \subseteq \{\varepsilon, 1\}$. By the definition of $f(t)$, $\mathrm{pos}(f(t)) \subseteq \{\varepsilon, 1\}$ holds as well.

Let $|\mathrm{var}(t)| > 1$. If $l = 1$ and $u_1(\varepsilon) \notin X$, then $|\mathrm{var}(t'_1)| > 1$. By the induction hypothesis, $v(t'_1)$ and u_1 are congruent. We derive $\mathrm{pos}(f(t)) = \mathrm{pos}(f(t'_1)) = \mathrm{pos}(v(t'_1)) = \mathrm{pos}(v(t))$. Moreover, $f(t)(w) = x_j$ iff $f(t'_1)(w) = x_j$ iff $v(t'_1)(w) = x_j$ iff $v(t)(w) = x_j$. If $l \neq 1$ or $u_1(\varepsilon) \in X$, we proceed as follows. By the induction hypothesis, $v(t'_\iota)$ and $f(t'_\iota)$ are congruent for every ι with $|\mathrm{var}(t'_\iota)| > 1$. We derive

$$\begin{aligned}
\mathrm{pos}(f(t)) &= \{\varepsilon\} \cup \{\iota w \mid \iota \in \{1, \ldots, l\}, \mathrm{rk}_{u_\iota}(\varepsilon) = 1, w \in \mathrm{pos}(u_\iota|_1)\} \\
&\qquad \cup \{\iota w \mid \iota \in \{1, \ldots, l\}, \mathrm{rk}_{u_\iota}(\varepsilon) \neq 1, w \in \mathrm{pos}(u_\iota)\} \\
&= \{\varepsilon\} \cup \{\iota \quad \mid \iota \in \{1, \ldots, l\}, \mathrm{rk}_{u_\iota}(\varepsilon) = 1\} \\
&\qquad \cup \{\iota w \mid \iota \in \{1, \ldots, l\}, \mathrm{rk}_{u_\iota}(\varepsilon) \neq 1, w \in \mathrm{pos}(v(t'_\iota))\} \\
&= \mathrm{pos}(v(t)) \,,
\end{aligned}$$

where we use the following reasoning. If $\mathrm{rk}_{u_\iota}(\varepsilon) = 1$, then u_ι is not congruent with any variable tree, and the induction hypothesis yields $|\mathrm{var}(t'_\iota)| \leq 1$, which in turn implies that $\mathrm{pos}(t'_\iota) \subseteq \{\varepsilon, 1\}$. By assumption, $|\mathrm{var}(t'_\iota)| \geq 0$, so $|\mathrm{var}(t'_\iota)| = 1$. Hence, and since $\mathrm{rk}_{u_\iota}(\varepsilon) = 1$, we obtain that $u_\iota|_1 \in X$. By similar reasoning, one can prove that $f(t)(w) = x_j$ iff $v(t)(w) = x_j$.

Now we show that $h_{\Delta''}(f(t)) = t$. We distinguish two cases. First, let $l = 1$ and $u_1(\varepsilon) \notin X$. Let $t''_1, \ldots, t''_k$ be as in the definition of f. We let $l' = \mathrm{rk}_{u_1}(\varepsilon)$ and $g\colon X_{l'} \to T_\Delta(X)$ with $g(x_j) = h_{\Delta''}(u_1|_j)$. First, we prove that $g^\sharp(t''_j) = t_j$. If $j \neq j_1$, then this is trivial. If $j = j_1$, then we derive

$$\begin{aligned}
g^\sharp(t''_j) &= g^\sharp(u_1(\varepsilon)) = h_{\Delta''}(u_1) \\
&= t_j \,. && \text{(induction hypothesis)}
\end{aligned}$$

Second, we derive

$$h_{\Delta''}(f(t)) = g^\sharp(\delta(t''_1, \dots, t''_k)) = \delta(g^\sharp(t''_1), \dots, g^\sharp(t''_k)) = \delta(t_1, \dots, t_k) = t \; .$$

Now let $l \neq 1$ or $u_1(\varepsilon) \in X$. Let $t''_1, \dots, t''_k$ and $u'_1, \dots, u'_l$ be as in the definition of f, and let $g\colon X_{l'} \to T_\Delta(X)$ with $g(x_j) = h_{\Delta''}(u'_j)$. As a first step, we prove that $g^\sharp(t''_j) = t_j$. If $j \notin \{j_1, \dots, j_l\}$, then this is trivial. If there is a ι with $j = j_\iota$ and $\mathrm{rk}_{u_\iota}(\varepsilon) = 1$, then we derive

$$\begin{aligned} g^\sharp(t''_j) &= g^\sharp(u_\iota(\varepsilon)[x_1/x_\iota]) = u_\iota(\varepsilon)[x_1/h_{\Delta''}(u_\iota|_1)] = h_{\Delta''}(u_\iota) \\ &= t_j \; . && \text{(induction hypothesis)} \end{aligned}$$

If there is a ι with $j = j_\iota$ and $\mathrm{rk}_{u_\iota}(\varepsilon) \neq 1$, then we derive

$$\begin{aligned} g^\sharp(t''_j) &= g^\sharp(x_\iota) = h_{\Delta''}(u_\iota) \\ &= t_j \; . && \text{(induction hypothesis)} \end{aligned}$$

The second step, $h_{\Delta''}(f(t)) = t$, is as in the former case. ∎

4.4.2 Constructing a binarization

Now we apply the concepts of the previous subsection in our context of IRTGs. Let $t = (t_1, \dots, t_n)$ be a sequence such that $t_i \in T_{\Delta_i}(X_k)$. We call t a *binarization hedge of* ρ if the following two properties hold:

(i) $h_i(\alpha)^{\mathcal{A}_i} = t_i^{\mathcal{A}_i}$,

(ii) the terms $t_1, \dots, t_n$ have the same variable tree, which is binary.

Let t be a binarization hedge of ρ, and let

$$\Gamma' = \{(\delta_1, \dots, \delta_n)^{(k)} \mid \exists w \in \mathrm{pos}(v(t_1))\colon k = \mathrm{rk}_{v(t_1)}(w), \forall i\colon \delta_i = f(t_i)(w)\} \; .$$

This definition is sound because $\mathrm{pos}(f(t_i)) = \mathrm{pos}(v(t_i))$, by Lm. 4.4.7, and $v(t_i) = v(t_1)$, by assumption. We construct a tree $\xi \in C_{\Gamma'}(X_k)$ and a ξ-binarization of ρ. We let ξ be obtained from $v(t_1)$ by replacing, at each $\star$-labeled position w, the label $\star$ by $(f(t_1)(w), \dots, f(t_n)(w))$. Since $v(t_1)$ is binary, so is ξ. Moreover, we let $\xi' = \xi[x_1/q_1] \cdots [x_k/q_k]$, and we define the IRTG $B(\rho, t)$ over Δ and $\mathcal{S}$ with

- $B(\rho, t) = (\Gamma', M', h')$,
- $M' = (Q' \cup \{q\}, q, R', \mu')$,

- $Q' = \{\xi'|_w \mid w \in \mathrm{pos}(\xi'), w \neq \varepsilon\}$, and
- $R' = \{\rho_w \mid w \in \mathrm{pos}(\xi'), \xi'(w) \notin Q\}$ where

$$\begin{aligned} \rho_\varepsilon\colon &\quad (\xi'|_1 \cdots \xi'|_{\mathrm{rk}_{\xi'}(\varepsilon)}, \xi'(\varepsilon), q) \;, \\ \rho_w\colon &\quad (\xi'|_{w1} \cdots \xi'|_{w\,\mathrm{rk}_{\xi'}(w)}, \xi'(w), \xi'|_w) \;, \qquad (w \neq \varepsilon) \end{aligned}$$

- $\mu'(\rho_\varepsilon) = \mu(\rho)$ and $\mu'(\rho_w) = 1$ for $w \neq \varepsilon$,
- $h'_i((\delta_1, \ldots, \delta_n)) = \delta_i$.

By Lm. 4.4.7, we have that $h_i(\alpha)^{\mathcal{A}_i} = t_i^{\mathcal{A}_i} = h'_i(\xi)^{\mathcal{A}_i}$. With this, it is easy to see that $B(\rho, t)$ is indeed a ξ-binarization of ρ. It is even rank normal, because a variable tree such as $v(t_1)$ does not contain any unary nodes.

Example 4.4.8 We can view the binarization of Fig. 4.5 as an instance of our construction, where we have

$$\begin{aligned} \alpha' &= \big(\mathrm{con}^2(x_1, x_2), \mathrm{con}^2(\mathrm{con}^2(x_2, a), x_1)\big) \;, \\ \alpha'' &= \big(\mathrm{con}^2(x_1, x_2), \mathrm{con}^2(x_1, x_2)\big) \;, \\ \xi &= \alpha'(\alpha''(x_1, x_2), x_3) \;, \\ A' &= \alpha''(B, C) \;. \end{aligned}$$

□

Lemma 4.4.9 *The following statements are equivalent:*

1. *There is a binarization of ρ.*
2. *There is a binarization hedge of ρ.*
3. *There is a rank-normal binarization of ρ.*

PROOF. "1 ⇒ 2". Let (Γ', M', h') be a ξ-binarization of ρ. Then ξ is binary, and so is $v(\xi)$. By Lm. 4.4.4, the sequence $(h'_1(\xi), \ldots, h'_n(\xi))$ is a binarization hedge. "2 ⇒ 3". Let t be a binarization hedge of ρ. Then $B(\rho, t)$ is a rank-normal binarization of ρ, as we have seen. "3 ⇒ 1". Trivial. ■

It remains to show how we can find a binarization hedge of ρ, if there is any. We begin our investigation with the following observation.

Observation 4.4.10 *Let* $(b_i \mid i \in \{1, \dots, n\})$ *with*

$$b_i \colon C_{\Delta_i}(X_k) \to \mathcal{P}(C_{\Delta_i}(X_k)) \,,$$
$$t \mapsto \{t' \mid t' \in C_{\Delta_i}(X_k), t^{\mathcal{A}_i} = t'^{\mathcal{A}_i}, v(t') \textit{ is binary}\} \,.$$

Then there is a binarization hedge of ρ *precisely when* $\bigcap_i v(b_i(h_i(\alpha))) \neq \emptyset$.

Example 4.4.11 (Ex. 4.3.1 contd.) Figure 4.6(b) shows some elements of $b_1(h_1(\alpha))$ and $b_2(h_2(\alpha))$. □

Example 4.4.12 (Ex. 4.3.3 contd.) Now we can argue that the first rule does not admit a binarization, by looking at $v(b_1(h_1(\alpha))) \cap v(b_2(h_2(\alpha)))$, where

$$\mathrm{con}^4(x_1, x_2, x_3, x_4) \xleftarrow{h_1} \alpha \xmapsto{h_2} \mathrm{con}^4(x_3, x_1, x_4, x_2) \,.$$

It is straightforward to enumerate $v(b_i(h_i(\alpha)))$. For $i = 1$, each element has yield $x_1x_2x_3x_4$; for $i = 2$, no element does. Hence, the corresponding sets are disjoint. □

Observation 4.4.10 constitutes an RCBM "template", which is depicted in Fig. 4.7. This template gives rise to a class of RCBMs, where we obtain a concrete RCBM by specifying the precise order in which the for-loop in Line 2 iterates over the rules and by making the selections in Lines 5 and 8 deterministic. We note that there is a technical problem with this template: in Line 9, it is not guaranteed that B' and $B(\rho, (t_1, \dots, t_n))$ are compatible. However, since our construction for $B(\rho, t)$ follows a rather strict regime with respect to Γ' and Q', we can assume that the assignment in Line 1 also prepares B' so that conflicts with this regime are ruled out.

The binarization mappings specified by the template are total (technically, left-total on the set of all IRTGs over Δ and $\mathcal{S}$): if a rule of the given IRTG does not have a binarization, then it is simply carried over to the new grammar, which then has a rank higher than 2.

4.5 Constructing a computable binarization mapping

In Fig. 4.7, we have seen a template for RCBMs. This template is based on the mappings $b_1, \dots, b_n$ of Obs. 4.4.10, which map a term to the set of all equivalent terms whose variable tree is binary. However, without any restrictions on the algebras, term equivalence is undecidable, and b_i is thus not computable. Consequently, said RCBMs are not computable either. In this section, we revise our template so that it describes computable binarization mappings. To this end, we "outsource" b_i to the user. Put more precisely, we require the user to specify an explicit approximation $\mathfrak{b}_i$ of the mapping b_i. We call this approximation a binarization rule (b-rule).

Input: IRTG $B = (\Gamma, M, h)$ over Δ and $\mathcal{S}$

Output: IRTG B' over Δ and $\mathcal{S}$

1: $B' \leftarrow B$
2: **for** each rule $\rho\colon (q_1 \cdots q_k, \alpha, q)$ of B with $k > 2$ **do**
3: $\quad L \leftarrow \bigcap_i v(b_i(h_i(\alpha)))$
4: $\quad$ **if** $L \neq \emptyset$ **then**
5: $\quad\quad$ select $\tau \in L$
6: $\quad\quad$ **for** $i = 1, \ldots, n$ **do**
7: $\quad\quad\quad L_i \leftarrow b_i(h_i(\alpha)) \cap v^{-1}(\tau)$
8: $\quad\quad\quad$ select $t_i \in L_i$
9: $\quad\quad B' \leftarrow B'[\rho / B(\rho, (t_1, \ldots, t_n))]$

Figure 4.7: RCBM template.

4.5.1 Binarization rule and algorithmization

Let us define the concept of a b-rule. In order to keep notation uncluttered, we will disregard the IRTG B in this subsection and rather proceed in a general setting. To this end, let Δ be an arbitrary ranked alphabet. For this section, we extend the notion of a (recognizable) tree language to subsets of $T_\Delta(X')$ for finite $X' \subseteq X$. We achieve this by identifying $T_\Delta(X')$ with $T_{\Delta \cup \{x^{(0)} | x \in X'\}}$.

A *binarization rule (b-rule)* $\mathfrak{b}$ *over* Δ is a mapping $\mathfrak{b}\colon \Delta \to \mathcal{P}(T_\Delta(X))$ such that for every $\delta \in \Delta^{(k)}$

- $\mathfrak{b}(\delta) \subseteq C_\Delta(X_k)$,
- $\mathfrak{b}(\delta)$ is a recognizable tree language, and
- $v(t)$ is binary for every $t \in \mathfrak{b}(\delta)$.

We extend $\mathfrak{b}$ to $T_\Delta(X)$ by letting

$$\begin{aligned} \mathfrak{b}(x_j) &= \{x_j\} \\ \mathfrak{b}(\delta(t_1, \ldots, t_k)) &= \left\{ t[x_1/t'_1] \cdots [x_k/t'_k] \;\middle|\; t \in \mathfrak{b}(\delta), t'_j \in \mathfrak{b}(t_j) \right\} . \end{aligned}$$

Given an algebra $\mathcal{A}$ over Δ, $\mathfrak{b}$ is called *a b-rule over* $\mathcal{A}$ if

$$t' \in \mathfrak{b}(t) \implies t^{\mathcal{A}} = t'^{\mathcal{A}} .$$

Such a b-rule encodes equivalence in $\mathcal{A}$, and it does so in an explicit and compact way: since $\mathfrak{b}(\delta)$ is a recognizable tree language, a b-rule can be specified by a finite collection of FTAs, one for each symbol $\delta \in \Delta$.

Example 4.5.1 We consider a b-rule $\mathfrak{b}$ for the algebra $\mathcal{A}$ from Ex. 4.3.1. Each symbol $a \in \Delta^{(0)}$ is mapped to the language $\{a\}$. Each symbol con^k, $k \geq 2$, is mapped to the language recognized by the following FTA with states of the form $[j, j']$ (where $0 \leq j < j' \leq k$) and root state $[0, k]$:

$$(\varepsilon, x_j, [j-1, j]) \,, \qquad (1 \leq j \leq k)$$
$$([j, j''] \; [j'', j'], \mathrm{con}^2, [j, j']) \,. \qquad (0 \leq j < j'' < j' \leq k)$$

This language expresses all possible ways in which con^k can be equivalently written in terms of con^2. □

Lemma 4.5.2 *For every $m \in \mathbb{N}$ and $t \in T_\Delta(X)$ with $|\mathrm{pos}(t)| \leq m$, the set $\mathfrak{b}(t)$ is a recognizable tree language, and effectively so.*

PROOF. By induction on m. For the base case ($m = 0$), there is nothing to show. We show the induction step ($m \to m + 1$). To this end, let $m \in \mathbb{N}$ and $t \in T_\Delta(X)$ with $|\mathrm{pos}(t)| \leq m + 1$. We distinguish two cases.

Case 1: Let $t = x_j$. Then $\mathfrak{b}(t) = \{x_j\}$, which is trivially recognizable.

Case 2: Let $t = \delta(t_1, \ldots, t_k)$ with $\delta \in \Delta$. By definition, $\mathfrak{b}(\delta)$ is recognizable. By the induction hypothesis, $\mathfrak{b}(t_1), \ldots, \mathfrak{b}(t_k)$ are recognizable. The class of recognizable tree languages is closed under substitution, and effectively so [80, Prop. 7.3]. ■

Now we show that, for every finite $X' \subseteq X$ and recognizable tree language $L \subseteq C_\Delta(X')$, also $v(L)$ is recognizable. To this end, we introduce an auxiliary result. Let $X' \subseteq X$ be finite and $G = (P, R, p_0)$ an FTA over $\Delta \cup X'$ in root-state form. A *(variable) inspection η of G* is a mapping $\eta \colon P \to \mathcal{P}(X')$ such that for every $p \in P$, $t \in T_\Delta(X')$, $d \in D^{p_0}(G, t)$, and $w \in \mathrm{pos}(d)$, we have $\mathrm{var}(t|_w) = \eta(\pi_P(d|_w))$.

Lemma 4.5.3 *Let $L(G) \subseteq C_\Delta(X')$. Then there is effectively an inspection η of G.*

PROOF. Algorithm 4.1 constructs η, along with the set P' of productive states. It terminates when P' is saturated, which is bound to happen because P' never shrinks, and it is bounded by P. Now we show that η is a variable inspection of G. We note that this holds regardless of the order in which the rules are iterated in the for loops. Let us assume that the order is arbitrary, but fixed.

Algorithm 4.1 Algorithm for computing a variable inspection.

Input: FTA $G = (P, R, p_0)$ with $L(G) \subseteq C_\Delta(X')$

Output: variable inspection η of G

1: $\eta \leftarrow \eta_\emptyset$ ▷ $\eta_\emptyset$ maps every state to $\emptyset$
2: $P' \leftarrow \emptyset$
3: **for** rule (ε, x_j, p) in R **do**
4: **if** $p \notin P'$ **then**
5: $\eta(p) \leftarrow \{x_j\}$
6: $P' \leftarrow P' \cup \{p\}$
7: **while** P' not saturated **do**
8: **for** rule $(p_1 \cdots p_k, \delta, p)$ in R with $\delta \in \Delta$ **do**
9: **if** $\{p_1, \ldots, p_k\} \subseteq P'$ and $p \notin P'$ **then**
10: $\eta(p) \leftarrow \bigcup_j \eta(p_j)$
11: $P' \leftarrow P' \cup \{p\}$

First of all, it is easy to see that the following invariant holds during the run of the algorithm: for every $p \in P'$, there is a $t \in T_\Delta(X')$ with $D^p(G,t) \neq \emptyset$ and $\mathrm{var}(t) = \eta(p)$. Now we prove our main statement by contradiction. For this, let

$$C = \{(p,t,d,w) \mid t \in T_\Delta(X'), d \in D^{p_0}(G,t), w \in \mathrm{pos}(t), p = \pi_P(d|_w), \\ p \notin P' \vee \mathrm{var}(t|_w) \neq \eta(p)\} .$$

We assume that C is nonempty. Then there is a $(p,t,d,w) \in C$ such that $|\mathrm{pos}(t|_w)|$ is minimal (minimality assumption). We distinguish three cases.

Case 1: Let $t(w) = x_j$ and $p \notin P'$. Then $(\varepsilon, x_j, p) \in R$. By Lines 3 to 6, $p \in P'$.

Case 2: Let $t(w) = \delta$, $\delta \in \Delta$, and $p \notin P'$. Then there are $p_1, \ldots, p_k$ with $d(\varepsilon) = (p_1 \cdots p_k, \delta, p)$ and $d|_{wj} \in D^{p_j}(G, t|_{wj})$. By our minimality assumption, $(p_j, d, t, wj) \notin C$. Hence, $p_j \in P'$. By Lines 8 to 11, $p \in P'$.

Case 3: Let $p \in P'$. Then $\mathrm{var}(t|_w) \neq \eta(p)$. Since $p \in P'$, there is a $t' \in T_\Delta(X')$ with $D^p(G,t') \neq \emptyset$ and $\mathrm{var}(t') = \eta(p)$. Thus, there is a $d' \in D^p(G,t')$. We construct $t'' = t[t']_w$ and $d'' = d[d']_w$. It is easy to see that $t, t'' \in L(G)$ and that at least one of them is not in $C_\Delta(X')$, which contradicts our assumption that $L(G) \subseteq C_\Delta(X')$.

We obtained a contradiction in each case, so we conclude that C is empty. ■

Example 4.5.4 (Ex. 4.5.1 contd.) We apply Alg. 4.1 to the FTA for $\mathfrak{b}(\mathrm{con}^3)$, and we protocol the values of P' and η at the end of certain lines. Then we obtain Tab. 4.2. □

line	P'	$\eta([0,1])$	$\eta([1,2])$	$\eta([2,3])$	$\eta([0,2])$	$\eta([1,3])$	$\eta([0,3])$
3	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
7	$\{[0,1],[1,2],[2,3]\}$	$\{x_1\}$	$\{x_2\}$	$\{x_3\}$	$\emptyset$	$\emptyset$	$\emptyset$
11	$\{[0,1],[1,2],[2,3],$ $[0,2],[1,3]\}$	$\{x_1\}$	$\{x_2\}$	$\{x_3\}$	$\{x_1,x_2\}$	$\{x_2,x_3\}$	$\emptyset$
11	$\{[0,1],[1,2],[2,3],$ $[0,2],[1,3],[0,3]\}$	$\{x_1\}$	$\{x_2\}$	$\{x_3\}$	$\{x_1,x_2\}$	$\{x_2,x_3\}$	$\{x_1,x_2,x_3\}$

Table 4.2: A run of Alg. 4.1.

For every finite $X' \subseteq X$, we transfer the definition of a canonical sort to finite sequences over $\mathcal{P}(X')$ as follows. A sequence $u_1, \ldots, u_l \subseteq X'$ is *eligible* if $u_j \neq \emptyset$ and $u_j \cap u_{j'} \neq \emptyset$ implies $j = j'$. Let $u_1, \ldots, u_l$ be eligible. Then the *canonical sort* $\mathrm{csort}(u_1, \ldots, u_l)$ *of* $u_1, \ldots, u_l$ is the sequence obtained from $u_1, \ldots, u_l$ by sorting according to the least variable index. For instance,

$$\mathrm{csort}(\{x_3\}, \{x_2, x_4\}) = \big(\{x_2, x_4\}, \{x_3\}\big) \,.$$

Observation 4.5.5 *Let G be trim and η an inspection of G. Then $\mathrm{var}(t) = \eta(p)$ for every $p \in P$ and t with $D^p(G,t) \neq \emptyset$. Consequently, $\eta(p) = \bigcup_j \eta(p_j)$ for every $(p_1 \cdots p_k, \delta, p) \in R$, and the sequence obtained from $\eta(p_1), \ldots, \eta(p_k)$ by removing every occurrence of the empty set is eligible for a canonical sort. Let η' be an inspection of G. Then $\eta = \eta'$.*

Next, we show that $v(L)$ is recognizable for every recognizable tree language L. We begin with the corresponding construction. Let G be trim and η an inspection for G. We define the FTA $\eta(G)$ over $\{\star, \emptyset\} \cup X'$ by

$$\eta(G) = \big(\eta(P), \eta(R), \eta(p_0)\big)$$

where $\eta(R)$ is the smallest set R' of transitions such that the following holds.

- Let $(\varepsilon, x_j, p) \in R$. Then $(\varepsilon, x_j, \{x_j\}) \in R'$.
- Let $(p_1 \cdots p_k, \delta, p) \in R$, $\delta \in \Delta$, and $u_1, \ldots, u_l$ be the canonical sort of the sequence obtained from $\eta(p_1), \ldots, \eta(p_k)$ by removing every occurrence of $\emptyset$. If $l = 0$, then $(\varepsilon, \emptyset, p) \in R'$. If $l \geq 2$, then $(u_1 \cdots u_l, \star, \eta(p)) \in R'$.

Example 4.5.6 (Ex. 4.5.1 contd.) Let G be the FTA for $\mathfrak{b}(\mathrm{con}^3)$. In this case, each transition of $\eta(G)$ is constructed from a transition of G:

$$\begin{aligned}
(\varepsilon, x_1, [0,1]) &\rightsquigarrow (\varepsilon, x_1, \{x_1\}) \,, \\
(\varepsilon, x_2, [1,2]) &\rightsquigarrow (\varepsilon, x_2, \{x_2\}) \,, \\
(\varepsilon, x_3, [2,3]) &\rightsquigarrow (\varepsilon, x_3, \{x_3\}) \,, \\
([0,1][1,2], \mathrm{con}^2, [0,2]) &\rightsquigarrow (\{x_1\}\{x_2\}, \star, \{x_1, x_2\}) \,, \\
([1,2][2,3], \mathrm{con}^2, [1,3]) &\rightsquigarrow (\{x_2\}\{x_3\}, \star, \{x_2, x_3\}) \,, \\
([0,1][1,3], \mathrm{con}^2, [0,3]) &\rightsquigarrow (\{x_1\}\{x_2, x_3\}, \star, \{x_1, x_2, x_3\}) \,, \\
([0,2][2,3], \mathrm{con}^2, [0,3]) &\rightsquigarrow (\{x_1, x_2\}\{x_3\}, \star, \{x_1, x_2, x_3\}) \,.
\end{aligned}$$

□

Lemma 4.5.7 *Let $L \subseteq C_\Delta(X')$ be recognizable. Then $v(L)$ is effectively recognizable.*

PROOF. By Lm. 2.4.10, there is a trim FTA G in root-state form with $L(G) = L$. Let $G = (P, R, p_0)$. By Lm. 4.5.3, there is an inspection η of G. For the proof of $v(L(G)) = L(\eta(G))$, one shows by induction on m that the following two statements hold. This is straightforward. *Statement 1*: For every $m \in \mathbb{N}$, $p \in P$, and $d \in D^p_{\mathrm{co}}(G)$ with $|\mathrm{pos}(d)| \leq m$, there is a $d' \in D^{\eta(p)}_{\mathrm{co}}(\eta(G))$ with $\pi_{\{\star,\emptyset\}\cup X'}(d') = v(\pi_{\Delta\cup X'}(d))$. *Statement 2*: For every $m \in \mathbb{N}$, $p \in P$, and $d' \in D^{\eta(p)}_{\mathrm{co}}(\eta(G))$, there is a $d \in D^p_{\mathrm{co}}(G)$ with $\pi_{\{\star,\emptyset\}\cup X'}(d') = v(\pi_{\Delta\cup X'}(d))$. ∎

Now we show that $L \cap v^{-1}(\tau)$ is recognizable for every recognizable tree language L and every variable tree τ. Again, we begin with the corresponding construction. Let G be trim, η an inspection for G, and $\tau = v(t)$ for some $t \in C_\Delta(X')$. We define the FTA $\eta(G, \tau)$ over $\Delta \cup X'$ by

$$\eta(G, \tau) = \big(P, \eta(R, \tau), p_0\big)$$

where $\eta(R, \tau)$ is the smallest set R' of transitions such that the following holds.

- Let $(\varepsilon, x_j, p) \in R$. If $\tau(w) = x_j$ for some $w \in \mathrm{pos}(\tau)$, then $(\varepsilon, x_j, p) \in R'$.
- Let $(p_1 \cdots p_k, \delta, p) \in R$, $\delta \in \Delta$, and $u_1, \ldots, u_l$ be the canonical sort of the sequence obtained from $\eta(p_1), \ldots, \eta(p_k)$ by removing every occurrence of $\emptyset$. If $l < 2$ or if there is a $w \in \mathrm{pos}(\tau)$ with $\mathrm{rk}_\tau(w) = l$, $\mathrm{var}(\tau|_w) = \eta(p)$, and $\mathrm{var}(\tau|_{wj}) = u_j$, then $(p_1 \cdots p_k, \delta, p) \in R'$.

Example 4.5.8 (Ex. 4.5.1 contd.) Let G be the FTA for $\mathfrak{b}(\mathrm{con}^3)$ and τ as in Fig. 4.6, i.e., $\tau = \star(\star(x_1, x_2), x_3)$. Then $\eta(G, \tau)$ has the following transitions: $(\varepsilon, x_1, [0, 1])$, $(\varepsilon, x_2, [1, 2])$, $(\varepsilon, x_3, [2, 3])$, $([0, 1][1, 2], \mathrm{con}^2, [0, 2])$, $([0, 2][2, 3], \mathrm{con}^2, [0, 3])$. □

Lemma 4.5.9 *Let $L \subseteq C_\Delta(X')$ be recognizable and $\tau = v(t)$ for some $t \in L$. Then $L \cap v^{-1}(\tau)$ is effectively recognizable.*

PROOF. By Lm. 2.4.10, there is a trim FTA G in root-state form with $L(G) = L$. Let $G = (P, R, p_0)$. By Lm. 4.5.3, there is an inspection η of G. We prove that $L(G) \cap v^{-1}(\tau) = L(\eta(G, \tau))$.

We begin with "$\subseteq$". To this end, let $t \in L(G)$ such that $v(t) = \tau$. Then there is a p_0-run d of G on t. We show by induction on m that $d|_w \in D^p(G)$ implies $d|_w \in D^p(\eta(G, \tau))$ for every $m \in \mathbb{N}$, $p \in P$, and $w \in \mathrm{pos}(d)$ with $|\mathrm{pos}(d|_w)| \leq m$. For the induction base ($m = 0$), there is nothing to show. We show the induction step ($m \to m + 1$). For this, let $m \in \mathbb{N}$, $p \in P$, and $w \in \mathrm{pos}(d)$ such that $|\mathrm{pos}(d|_w)| \leq m + 1$ and

$d|_w \in D^p(G)$. By the induction hypothesis, it suffices to show that $d(w) \in \eta(R, \tau)$. We distinguish two cases.

Case 1: Let $d(w) = (\varepsilon, x_j, p)$. Then $d(w) \in \eta(R, \tau)$.

Case 2: Let $d(w) = (p_1 \cdots p_k, \delta, p)$ and $\delta \in \Delta$. Let $u_1, \ldots, u_l$ be the canonical sort of the sequence obtained from $\eta(p_1), \ldots, \eta(p_k)$ by removing every occurrence of $\emptyset$. If $l < 2$, then $d(w) \in \eta(R, \tau)$ holds trivially. Let $l \geq 2$; then by definition $v(t|_w) = \star(v(t'_1), \ldots, v(t'_l))$ where $t'_1, \ldots, t'_l$ is obtained from $t|_{w1}, \ldots, t|_{wk}$ by removing every occurrence of any tree that does not contain any variables. It is easy to see from the recursive definition of v that $v(t|_w)$ occurs in $v(t)$. Since η is an inspection, and since $\mathrm{var}(v(t)) = \mathrm{var}(t)$ for every $t \in T_\Delta$, we obtain that $\eta(p) = \mathrm{var}(t|_w) = \mathrm{var}(v(t|_w))$ and $\eta(p_j) = \mathrm{var}(t|_{wj}) = \mathrm{var}(v(t|_{wj}))$. Hence, $d(w) \in \eta(R, \tau)$.

Now we show "$\supseteq$". Since $\eta(R, \tau) \subseteq R$, we obtain that $L(\eta(G, \tau)) \subseteq L(G)$ and that η is also an inspection for $\eta(G, \tau)$. We show that $L(\eta(G, \tau)) \subseteq v^{-1}(\tau)$. The case that $\eta(p_0) = \emptyset$ is easy. Let $\eta(p_0) \neq \emptyset$. We make the following crucial observation: for every $u \subseteq X_k$, there is at most one $w' \in \mathrm{pos}(\tau)$ with $\mathrm{var}(\tau|_{w'}) = u$. Let $t \in T_\Delta$ and $d \in D^{p_0}(G, t)$. We show by induction on m that, for every $m \in \mathbb{N}$ and $w \in \mathrm{pos}(d)$, if $|\mathrm{pos}(d|_w)| \leq m$ and $\eta(\pi_P(d|_w)) \neq \emptyset$, then there is a $w' \in \mathrm{pos}(\tau)$ such that $v(t|_w) = \tau|_{w'}$. Then, since $\mathrm{var}(t) = \eta(p_0) = \mathrm{var}(\tau|_\varepsilon)$ and by our observation, we have that $v(t) = \tau$.

For the induction base ($m = 0$), there is nothing to show. We show the induction step ($m \to m + 1$). For this, let $m \in \mathbb{N}$ and $w \in \mathrm{pos}(d)$ such that $|\mathrm{pos}(d|_w)| \leq m + 1$ and $\eta(\pi_P(d|_w)) \neq \emptyset$. We distinguish two cases.

Case 1: Let $d(w) = (\varepsilon, x_j, p)$. Clearly, there is a $w' \in \mathrm{pos}(\tau)$ with $\tau(w') = x_j$.

Case 2: Let $d(w) = (p_1 \cdots p_k, \delta, p)$ and $\delta \in \Delta$. Let $u_1, \ldots, u_l$ be obtained from $\eta(p_1), \ldots, \eta(p_k)$ by removing every occurrence of $\emptyset$. There are uniquely determined $j_1, \ldots, j_l$ such that $u_\iota = \eta(p_{j_\iota})$. If $l \leq 2$, then $l = 1$, and $\nu(p) = u_1$. By the induction hypothesis, there is a w' with $v(t|_{wj_1}) = \tau|_{w'}$. Hence, we can derive that $v(t|_w) = v(t|_{wj_1}) = \tau|_{w'}$. If $l \geq 2$, then $v(t|_w) = \star(v(t|_{wj_1}), \ldots, v(t|_{wj_l}))$. Since $d(w) \in \eta(R, \tau)$, we have that there is a $w' \in \tau$ with $\mathrm{var}(\tau|_{w'}) = \eta(p)$ and $\mathrm{var}(\tau|_{w'\iota}) = u_\iota$. By the induction hypothesis, we have that there are $w'_1, \ldots, w'_l$ with $v(t|_{wj_\iota}) = \tau|_{w'_\iota}$. By our initial observation, we obtain that $w'_\iota = w'\iota$. Hence, $v(t|_w) = \tau|_{w'}$. ∎

4.5.2 Binarization under binarization rules

Let $\mathfrak{b} = (\mathfrak{b}_1, \ldots, \mathfrak{b}_n)$ such that $\mathfrak{b}_i$ is a b-rule over $\mathcal{A}_i$. A ξ-binarization (Γ', M', h') of ρ is called "*under* $\mathfrak{b}$" if $h'_i(\xi) \in \mathfrak{b}_i(h_i(\alpha))$. Likewise, a binarization hedge t of ρ is called "*under* $\mathfrak{b}$" if $t_i \in \mathfrak{b}_i(h_i(\alpha))$. Lemma 4.4.9 and Obs. 4.4.10 carry over to these restricted notions. A binarization mapping $bin\colon \mathcal{C} \to \mathcal{C}$ is called $\mathfrak{b}$-*complete* if $\mathrm{bdom}(bin)$ contains every $B \in \mathcal{C}$ such that for every suprabinary rule ρ of B there is a

Algorithm 4.2 Binarization algorithm.

Input: IRTG $B = (\Gamma, M, h)$ over Δ and $\mathcal{S}$,
b-rules $\mathfrak{b}_1, \ldots, \mathfrak{b}_n$ over $\Delta_1, \ldots, \Delta_n$, respectively

Output: IRTG B' over Δ and $\mathcal{S}$

1: $B' \leftarrow B$
2: **for** each rule $\rho\colon (q_1 \cdots q_k, \alpha, q)$ of B with $k > 2$ **do**
3: compute FTA G' for $\bigcap_i v(\mathfrak{b}_i(h_i(\alpha)))$
4: **if** $L(G') \neq \emptyset$ **then**
5: select $\tau \in L(G')$
6: **for** $i = 1, \ldots, n$ **do**
7: compute FTA G'_i for $\mathfrak{b}_i(h_i(\alpha)) \cap v^{-1}(\tau)$
8: select $t_i \in L(G'_i)$
9: $B' \leftarrow B'[\rho / B(\rho, t_1, \ldots, t_n)]$

binarization of ρ under $\mathfrak{b}$.

By definition, rule-by-rule complete implies $\mathfrak{b}$-complete. The converse need not be true. However, if the b-rules $\mathfrak{b}$ have a certain property, we can guarantee that $\mathfrak{b}$-complete also implies rule-by-rule complete. More specifically, we say that $\mathfrak{b}$ is *complete on* B if $v(\mathfrak{b}_i(h_i(\alpha)) = v(b_i(h_i(\alpha))$ for every $\alpha \in \Gamma$ and $i \in \{1, \ldots, n\}$. Then the intersection in Obs. 4.4.10 is empty in the restricted case iff it is empty in the general case, i.e.,

$$\bigcap_i v(\mathfrak{b}_i(h_i(\alpha))) \neq \emptyset \qquad \iff \qquad \bigcap_i v(b_i(h_i(\alpha))) \neq \emptyset \,.$$

Consequently, if $\mathfrak{b}$ is complete on every element of $\mathcal{C}$, then $\mathfrak{b}$-complete implies rule-by-rule complete.

Now we have the ingredients that we need for a template that gives rise to a class of computable binarization mappings. It is shown as Alg. 4.2. As before, we obtain a concrete binarization mapping by making the for-loop and the selections deterministic. In Line 3, we use Lms. 4.5.2 and 4.5.7 and that the class of recognizable tree languages is effectively closed under intersection [80, Prop. 7.1]. In Line 7, we use Lm. 4.5.9. The following theorem documents the behavior of the template. In short, when we fix $\mathfrak{b}$, the template describes a class of $\mathfrak{b}$-complete binarization mappings.

Theorem 4.5.10 *Let*

- $\Delta = (\Delta_1, \ldots, \Delta_n)$ *be a sequence of ranked alphabets,*
- $\mathcal{S}$ *a commutative semiring,*
- $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_n)$ *a sequence such that* $\mathcal{A}_i$ *is a* Δ_i*-algebra,*

- $\mathfrak{b} = (\mathfrak{b}_1, \ldots, \mathfrak{b}_n)$ *a sequence such that* $\mathfrak{b}_i$ *is a b-rule over* $\mathcal{A}_i$.

Moreover, let B *be an IRTG over* Δ *and* $\mathcal{S}$*. If we execute Alg. 4.2 with input* B *and* $\mathfrak{b}$*, then it terminates, and it outputs an IRTG* B' *over* Δ *and* $\mathcal{S}$ *such that (i)* B' *is of rank* 2 *iff every suprabinary rule of* B *has a binarization under* $\mathfrak{b}$*, (ii) if* B *is* $\mathcal{A}$*-admissible, then so is* B'*, and (iii) in that case,* B *and* B' *are* $\mathcal{A}$*-equivalent.*

The runtime of Alg. 4.2 is dominated by the intersection in Line 3, whose runtime is in $O(m_1 \cdots m_n)$, where m_i is the size of the FTA for $\mathfrak{b}_i(h_i(\alpha))$. The quantity m_i is linear in the size of the terms in $\{h_i(\alpha) \mid \alpha \in \Gamma\}$ and in the number of transitions in the FTAs for the b-rule $\mathfrak{b}_i$. It is convenient to consider this quantity to be constant; then the overall runtime of our algorithm is in $O(|R| \cdot c^n)$ for some $c \in \mathbb{N}$.

4.6 Application to established formalisms

In this section we consider how the RCBMs for IRTGs can (or cannot) be used to obtain RCBMs for established formalisms such as SCFGs, tree-to-string transducers (yXTTs), or WSCFTGs.

4.6.1 General approach

First, we briefly consider two possible approaches to this question: the solution-transfer approach and the problem-transfer approach. To this end, let $\mathcal{F}$ be a class of devices (such as SCFGs).

In the solution-transfer approach, we are interested in a partial mapping $bin\colon \mathcal{F} \to \mathcal{F}$ that preserves meaning and reduces the rank to 2. We assume that there is a suitable subclass $\mathcal{C}$ of IRTGs for which we already have a binarization mapping bin'. Then we define bin as follows: given an element of $\mathcal{F}$, we convert it into an IRTG in $\mathcal{C}$, we apply (if possible) bin', and then we convert the resulting IRTG back; naturally, the conversion must preserve rank and meaning. The problem is that we cannot make any meaningful statement about bin (e.g., whether it is rule-by-rule complete) without the whole formal apparatus for $\mathcal{F}$.

In the problem-transfer approach, we also assume that there is a suitable subclass $\mathcal{C}$ of IRTGs as well as means of converting back and forth. However, we regard this conversion as rather hypothetical, for when it comes to treating binarization and other problems formally, we stipulate that $\mathcal{C}$ rather than $\mathcal{F}$ is the formalism in question, and that $\mathcal{F}$ is effectively obsolete. That is, our aim then is to find a binarization mapping $bin\colon \mathcal{C} \to \mathcal{C}$. This change of perspective enables us to use IRTG terminology throughout. We note that it remains possible to use existing infrastructure for $\mathcal{F}$, namely via conversion; we just do not make any formal statements pertaining to $\mathcal{F}$.

In the following, we will pursue the problem-transfer approach. To this end, we define what a grammar formalism is (from the point of view of IRTGs), we motivate that definition, and then we define the binarization mapping for a formalism, given appropriate b-rules. We will consider examples of formalisms, such as SCFGs, and the respective binarization mappings in the subsequent subsections.

Let $\Delta = (\Delta_1, \ldots, \Delta_n)$ be a sequence of ranked alphabets and $\mathcal{S}$ a semiring. A *(grammar) formalism* is a triple $(\mathcal{C}, \mathcal{A}, \varphi)$ such that

- $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_n)$ is a sequence such that $\mathcal{A}_i$ is a Δ_i-algebra, and
- $\mathcal{C}$ is a set of $\mathcal{A}$-admissible IRTGs over Δ and $\mathcal{S}$,
- $\varphi\colon \mathcal{C} \to \mathcal{C}$ is an idempotent mapping (called *normal-form mapping*) that preserves meaning, rank, and rule-by-rule (non)binarizability, i.e., every suprabinary rule of B has a binarization iff the same is true for $\varphi(B)$.

Let $(\mathcal{C}, \mathcal{A}, \varphi)$ be a formalism. We usually identify $(\mathcal{C}, \mathcal{A}, \varphi)$ with $\mathcal{C}$. For every $B \in \mathcal{C}$, we define the *$\mathcal{C}$-meaning $[\![B]\!]_{\mathcal{C}}$ of G* by letting $[\![B]\!]_{\mathcal{C}} = [\![B]\!]_{\mathcal{A}}$.

Let us motivate our definition of a formalism. First, we observe that the definition of a grammar class such as SCFG implicitly uses a fixed selection of operations, and these operations can be captured by a fixed sequence $\mathcal{A}$ of algebras. Second, it is important to note that $\mathcal{C}$ is often a *strict* subset of all IRTGs over Δ and $\mathcal{S}$; e.g., in a WSCFTG the variables in Y may not occur arbitrarily often in a rule. Third, we recall that IRTGs can offer a high degree of freedom for expressing the generational behavior of a rule, as illustrated in Ex. 4.2.2. We accommodate this fact by the normal-form mapping. We require that rank and rule-by-rule (non)binarizability be preserved so that the normal form does not interfere with binarization.

Let $\mathfrak{b} = (\mathfrak{b}_1, \ldots, \mathfrak{b}_n)$ be a sequence such that $\mathfrak{b}_i$ is a b-rule over $\mathcal{A}_i$. Moreover, let $bin_{\mathfrak{b}}$ be a computable $\mathfrak{b}$-complete binarization mapping for $\mathcal{A}$-admissible IRTGs over Δ and $\mathcal{S}$; we know that such a mapping exists because of Thm. 4.5.10. We define the partial mapping $bin_{\mathcal{C}}\colon \mathcal{C} \to \mathcal{C}$ by letting

$$bin_{\mathcal{C}} = (bin_{\mathfrak{b}} \circ \varphi) \cap (\mathcal{C} \times \mathcal{C}) \ .$$

There are three possible causes when $B \in \mathcal{C} \setminus \mathrm{bdom}(bin_{\mathcal{C}})$:

1. some suprabinary rule of B does not have a binarization,
2. the b-rules are not complete on $\varphi(B)$, or
3. $bin_{\mathfrak{b}}(\varphi(B)) \notin \mathcal{C}$.

In a practical application, each of these causes may be acceptable. We, however, take a theoretical stance and accept only the first cause. In other words, we want $bin_{\mathcal{C}}$ to be rule-by-rule complete. Correspondingly, we say that $\mathcal{C}$ and $\mathfrak{b}$ are *admissible* if $bin_{\mathfrak{b}}(\varphi(\mathcal{C})) \subseteq \mathcal{C}$, and they are *complete* if $\mathfrak{b}$ is complete on every element of $\varphi(\mathcal{C})$. If we have admissibility and completeness, then $bin_{\mathcal{C}}$ is rule-by-rule complete.

We note that the properties "admissible" and "complete" refer to the combination of formalism and b-rules. However, for the sake of simplicity, we will also say that some formalism is admissible or complete, assuming that the b-rules are fixed.

In the following subsections we consider concrete algebras and formalisms.

4.6.2 Useful algebras and b-rules

We consider three algebras together with suitable b-rules: the string algebra, the hedge algebra, and the hedge algebra with substitution. A *hedge* is a sequence of trees; this notion is central to XML-related theory [140, 177], and in work related to natural-language processing, hedges are also called s-terms [169]. In this section, we deviate from Sec. 2.2 and define trees and hedges anew.

To this end, let Σ and V be sets. Then the *set $H_\Sigma(V)$ of hedges over Σ indexed by V* and the *set $T_\Sigma(V)$ of trees over Σ indexed by V* are defined by

$$(H_\Sigma(V), T_\Sigma(V)) = (H, T) \,,$$

where (H, T) is the smallest pair, according to the pointwise subset relation, such that

- $T^* \subseteq H$,
- $V \subseteq T$, and
- $\sigma(u) \in T$ if $\sigma \in \Sigma$ and $u \in H$.

Let $V' \subseteq \Sigma \cup V \cup X \cup Y$ and $f\colon V' \to H_\Sigma(V)$. Then we define the mappings $f^\flat, f^{\flat\flat}\colon H_\Sigma(V) \to H_\Sigma(V)$, called *first-order substitution* and *second-order substitution*, respectively, as follows. We let

$$f^\flat(\varepsilon) = f^{\flat\flat}(\varepsilon) = \varepsilon \,.$$

For every $v \in V$ and $u \in H_\Sigma(V)$, we let

$$f^\flat(vu) = \begin{cases} f(v)f^\flat(u) & \text{if } v \in V', \\ vf^\flat(u) & \text{if } v \notin V'. \end{cases} \qquad f^{\flat\flat}(vu) = \begin{cases} f(v)f^{\flat\flat}(u) & \text{if } v \in V', \\ vf^{\flat\flat}(u) & \text{if } v \notin V'. \end{cases}$$

Ranked alphabets Σ alphabet, $K \in \mathbb{N}$

$$\begin{aligned}
\mathrm{SYM}(\Sigma) &= \{\sigma^{(0)} \mid \sigma \in \Sigma\} \\
\mathrm{TOP}(\Sigma) &= \{\sigma^{(1)} \mid \sigma \in \Sigma\} \\
\mathrm{CON}_K &= \{(\mathrm{con}^k)^{(k)} \mid 0 \leq k \leq K, k \neq 1\} \\
\Pi_K &= \{\pi_k \mid 1 \leq k \leq K\} \\
\mathrm{SUB}_K &= \{(\mathrm{sub}^k)^{(k+1)} \mid 1 \leq k \leq K\}
\end{aligned}$$

Operations $\sigma \in \Sigma$, $k \in \mathbb{N}$, D set

$$\begin{aligned}
sym_\sigma &\colon ((\Sigma \cup X)^*)^0 \to (\Sigma \cup X)^*, && () \mapsto \sigma \\
top_\sigma &\colon (H_{\Sigma\cup X}(Y))^1 \to H_{\Sigma\cup X}(Y), && (u_1) \mapsto \sigma(u_1) \\
con_{k,D} &\colon (D^*)^k \to D^*, && (w_1, \ldots, w_k) = w_1 \cdots w_k \\
pi_k &\colon (H_{\Sigma\cup X}(Y))^0 \to H_{\Sigma\cup X}(Y), && () \mapsto y_k \\
sub_k &\colon (H_{\Sigma\cup X}(Y))^{k+1} \to H_{\Sigma\cup X}(Y), \\
& && (u, u_1, \ldots, u_k) \mapsto u[y_1/u_1] \cdots [y_k/u_k]
\end{aligned}$$

Table 4.3: Ranked alphabets and operations for our algebras.

For every $\sigma(u_1)u_2 \in H_\Sigma(V)$, we let

$$f^\flat(\sigma(u_1)u_2) = \begin{cases} f(\sigma)f^\flat(u_2) & \text{if } \sigma \in V', \\ \sigma(f^\flat(u_1))f^\flat(u_2) & \text{if } \sigma \notin V'. \end{cases}$$

If $V' = \{v_1, \ldots, v_l\}$, then we also denote $f^\flat(u)$ by $u[v_1/f(v_1)] \cdots [v_l/f(v_l)]$. We let

$$f^{\flat\flat}(\sigma(t_1, \ldots, t_k)u) = \begin{cases} f(\sigma)[y_1/f^{\flat\flat}(t_1)] \cdots [y_k/f^{\flat\flat}(t_k)]f^{\flat\flat}(u) & \text{if } \sigma \in V', \\ \sigma(f^{\flat\flat}(t_1), \ldots, f^{\flat\flat}(t_k))f^{\flat\flat}(u) & \text{if } \sigma \notin V'. \end{cases}$$

If $V' = \{v_1, \ldots, v_l\}$, then we also denote $f^{\flat\flat}(t)$ by $t(\!|v_1/f(v_1)|\!) \cdots (\!|v_l/f(v_l)|\!)$.

Now we proceed to discuss the three algebras promised above. We define these algebras based on the alphabets and the operations shown in Tab. 4.3; the algebras themselves are given in Tab. 4.4.

String algebra Roughly speaking, the string algebra is defined like the algebra $\mathcal{A}$ in Ex. 4.2.2. In addition to the alphabet Σ, we have another parameter $K \in \mathbb{N}$ that restricts the number of con-symbols.

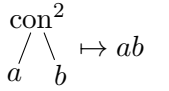
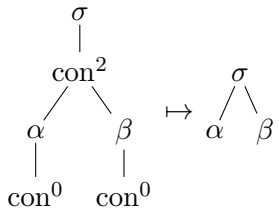
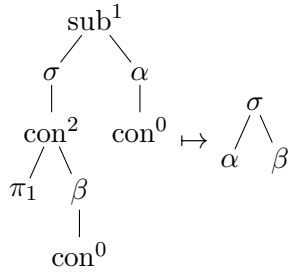
	string algebra over Σ and K	**hedge algebra over Σ and K**	**hedge algebra with substitution over Σ and K**
example term and denoted object			
signature	$\mathrm{SYM}(\Sigma) \cup \mathrm{CON}_K$	$\mathrm{TOP}(\Sigma) \cup \mathrm{CON}_K$	$\mathrm{TOP}(\Sigma) \cup \mathrm{CON}_K \cup \Pi_K \cup \mathrm{SUB}_K$
domain	$(\Sigma \cup X)^*$	$H_{\Sigma \cup X}(Y)$	$H_{\Sigma \cup X}(Y)$
realization mapping	$\sigma \mapsto sym_\sigma$ $\mathrm{con}^k \mapsto con_{k,\Sigma \cup X}$	$\sigma \mapsto top_\sigma$ $\mathrm{con}^k \mapsto con_{k,T_{\Sigma \cup X}(Y)}$	$\sigma \mapsto top_\sigma$ $\mathrm{con}^k \mapsto con_{k,T_{\Sigma \cup X}(Y)}$ $\pi_k \mapsto pi_k$ $\mathrm{sub}^k \mapsto sub_k$

Table 4.4: Algebras for strings and hedges, given an alphabet Σ and a maximum arity $K \in \mathbb{N}$.

We use the following b-rule $\mathfrak{b}$ (cf. Ex. 4.5.1): it maps each $\sigma \in \Sigma$ to $\{\sigma\}$, and it maps con^0 to $\{\mathrm{con}^0\}$. Each symbol con^k, $k \geq 2$, is mapped to the language recognized by the following FTA with states of the form $[j, j']$ (where $0 \leq j < j' \leq k$) and root state $[0, k]$:

$$(\varepsilon, x_j, [j-1, j]) \,, \qquad (1 \leq j \leq k)$$
$$([j, j''] \; [j'', j'], \mathrm{con}^2, [j, j']) \,. \qquad (0 \leq j < j'' < j' \leq k)$$

Hedge algebra This algebra incorporates two main ideas:

1. We can construct a tree $\sigma(t_1, \ldots, t_k)$ in two steps: first, we construct the hedge $t_1 \cdots t_k$ of children, and second, we "put" a node labeled σ "on top".
2. We can identify any tree t with the hedge (t) of length 1.

Correspondingly, the domain is the set $H_{\Sigma \cup X}(Y)$ of hedges, and we have two kinds of operations: (i) for every $\sigma \in \Sigma$, we can put σ on top of a hedge, yielding a hedge of length 1; and (ii) we can concatenate k hedges, as in the string algebra.

We use the following b-rule $\mathfrak{b}$: it maps con^0 to $\{\mathrm{con}^0\}$ and each unary symbol σ to $\{\sigma(x_1)\}$. Each symbol con^k, $k \geq 2$, is treated as in the string case.

Hedge algebra with substitution We can supplement the hedge algebra with a substitution operation; this way we can describe formalisms like WSCFTGs. As an example for substitution, we consider the term function f of $\mathrm{sub}^1(x_2, S(x_1))$. For every $t_1 \in T_\Sigma$ and $t_2 \in C_\Sigma(Y_1)$, we have that $f(t_1, t_2) = t(\!|x_2/t_2|\!)(\!|x_1/t_1|\!)$, where $t = x_2(S(x_1))$. A similar algebra with substitution has been described in [113, 127]; the basic idea goes back to the derived alphabets of [67].

We use the following b-rule $\mathfrak{b}$: it maps con^0 to $\{\mathrm{con}^0\}$ and each unary symbol σ to $\{\sigma(x_1)\}$. Each symbol con^k, $k \geq 2$, is treated as in the string case. Furthermore, it maps π_k, $k \geq 1$, to $\{\pi_k\}$, sub^1 to $\{\mathrm{sub}^1(x_1, x_2)\}$, and sub^k, $k > 1$, to $\emptyset$.

In the following, when we use a string algebra (or hedge algebra, or hedge algebra with substitution), say, $\mathcal{A}_1$, we will silently assume that Δ_1 is its signature and $\mathfrak{b}_1$ is the corresponding b-rule.

4.6.3 Synchronous context-free grammars

Let Σ be an alphabet and $K \in \mathbb{N}$. We will define the formalism $\mathrm{SCFG}(\Sigma, K)$, which, informally speaking, represents SCFGs over Σ whose rules contain strings of length at most K. To this end, let $\mathcal{A}$ be the string algebra over Σ and K.

We let $\mathrm{SCFG}(\Sigma, K) = (\mathcal{C}, (\mathcal{A}, \mathcal{A}), \varphi)$, where we define $\mathcal{C}$ and φ as follows. We let $\mathcal{C}$ be the set of all IRTGs B over (Δ, Δ) and Real such that, if $B = (\Gamma, M, h)$, then $|h_i(\alpha)^{\mathcal{A}}(x_1, \ldots, x_l)| \leq K$ for every $i \in \{1, 2\}$, $l \in \mathbb{N}$, and $\alpha \in \Gamma^{(l)}$. Now we define $\varphi\colon \mathcal{C} \to \mathcal{C}$. To this end, let $B \in \mathcal{C}$ and $B = (\Gamma, M, h)$. We construct $\varphi(B) = (\Gamma, M, h')$ where, for every $l \in \mathbb{N}$ and $\alpha \in \Gamma^{(l)}$, $h'_i(\alpha) = f(h_i(\alpha)^{\mathcal{A}}(x_1, \ldots, x_l))$, and $f\colon (\Sigma \cup X)^* \to T_\Delta(X)$ is defined by

$$f(\sigma_1, \ldots, \sigma_k) = \begin{cases} \mathrm{con}^k(\sigma_1, \ldots, \sigma_k) & \text{if } k \neq 1, \\ \sigma_1 & \text{otherwise.} \end{cases}$$

It is easy to see that φ is idempotent, and that it preserves rank and variable trees (and, a fortiori, rule-by-rule (non)binarizability).

It remains to show that φ preserves meaning. To this end, let $i \in \{1, 2\}$, $l \in \mathbb{N}$, and $g\colon X_l \to \Sigma^*$. Recall that $g^\sharp$ is the homomorphic extension of g to $\mathcal{A}$. For this proof, we extend $g^\sharp$ to $T_{\Delta_i}(X_l)^*$ by letting $g^\sharp((t_1, \ldots, t_k)) = g^\sharp(t_1) \cdots g^\sharp(t_k)$. A simple proof by induction on m yields that $g^\sharp(t) = g^\sharp(t^{\mathcal{A}}(x_1, \ldots, x_l))$ for every $m \in \mathbb{N}$ and $t \in T_{\Delta_i}(X_l)$ with $|\mathrm{pos}(t)| \leq m$. From this, we conclude that $t^{\mathcal{A}} = f_i(t)^{\mathcal{A}}$, which implies $h_i(\alpha)^{\mathcal{A}} = h'_i(\alpha)^{\mathcal{A}}$ for every $\alpha \in \Gamma$. Now one can employ Cor. 4.2.7 in another simple proof by induction to show that $h_i(\xi)^{\mathcal{A}} = h'_i(\xi)^{\mathcal{A}}$ for every $\xi \in T_\Gamma$. From this, we conclude that $[\![G]\!] = [\![\varphi(G)]\!]$.

The formalism $\mathrm{SCFG}(\Sigma, K)$ is trivially admissible, because the b-rule for the string algebra does not introduce con^k with $k > K$. It is also complete. We note that our formalism would not be complete if φ did not collapse occurrences of con^k. For instance, the term $\mathrm{con}^2(\mathrm{con}^2(x_1, x_2), x_3)$ is equivalent to itself and to $\mathrm{con}^2(x_1, \mathrm{con}^2(x_2, x_3))$, but the b-rules only cover the former. Thus they miss one variable tree. For the collapsed version $\mathrm{con}^3(x_1, x_2, x_3)$, however, the b-rules cover both variable trees.

The binarization mapping $bin_{\mathcal{C}}$ coincides with that of [97]: any rule can be binarized in both frameworks or neither. For instance, for the SCFG rule

$$A \to \alpha(B, C, D, E), \qquad \alpha = \langle x_1\, x_2\, x_3\, x_4,\ x_2\, x_4\, x_1\, x_3 \rangle,$$

the sets $v(\mathfrak{b}(h_1(\alpha)))$ and $v(\mathfrak{b}(h_2(\alpha)))$ are disjoint; thus, no binarization exists.

4.6.4 Tree-to-string and hedge-to-string transducers

Some approaches to SMT go beyond string-to-string translation models such as SCFG by exploiting known syntactic structures in the source or target language. This perspective on translation naturally leads to the use of yXTTs [181, 79, 95, 90].

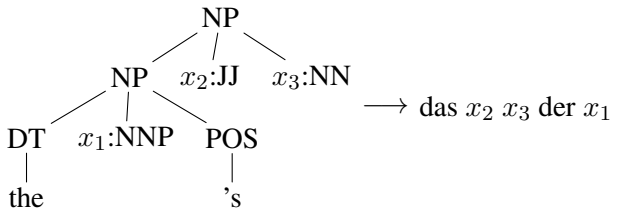


Figure 4.8: A yXTT rule in the notation of [79].

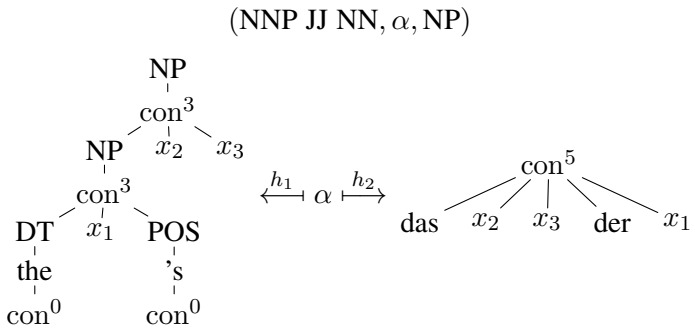


Figure 4.9: An IRTG rule encoding the rule in Fig. 4.8.

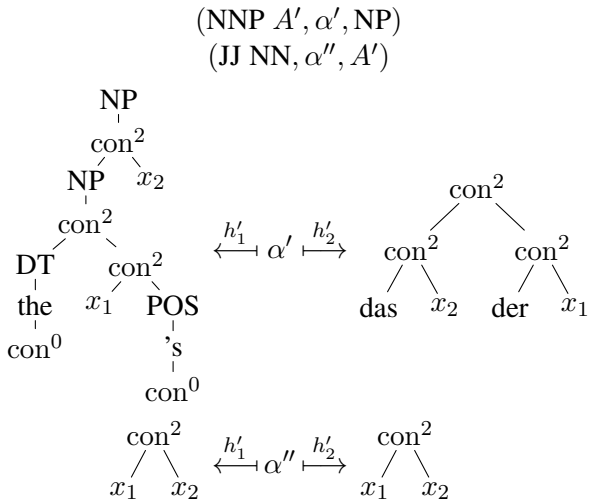


Figure 4.10: Binarization of the rule in Fig. 4.9.

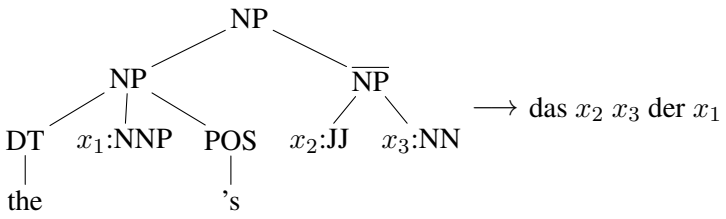


Figure 4.11: yXTT rule, slightly adapted to enable binarization.

Example 4.6.1 Figure 4.8 shows an example of a yXTT rule in the notation of [79]. This rule might be used to translate "the Commission's strategic plan" into "das langfristige Programm der Kommission". By employing both the hedge and the string algebra, we can represent this rule in an IRTG B, as indicated in Fig. 4.9. We may replace this suprabinary rule by the two binary rules shown in Fig. 4.10 without affecting the meaning, obtaining the IRTG B'.

However, the binary rules lack a counterpart in the original notation of yXTT, because $h'_1(\alpha')$ does not denote a sequence of length 1. Assume that we have a formalism $(\mathcal{C}, \mathcal{A}, \varphi)$ and b-rules $\mathfrak{b}_1, \mathfrak{b}_2$ such that $B' = bin_{(\mathfrak{b}_1,\mathfrak{b}_2)}(\varphi(B))$. Now either $B' \in \mathcal{C}$; then it is unclear what yXTT it corresponds to. Or $B' \notin \mathcal{C}$; then our formalism is not admissible.

It is not just a coincidence that finding an admissible formalism for yXTT seems hard. After all, since each child of the root node in Fig. 4.8 contains a variable that may be replaced with unbounded material, we just cannot factor the rule and at the same time stay within yXTT. If we are willing to accept a change of meaning, then we could instead factor the rule in Fig. 4.11. □

Example 4.6.1 illustrates that finding an admissible formalism for yXTTs is not straightforward. In fact, the author is not aware of an admissible combination of a formalism and b-rules for yXTTs; and, contrary to its appearance, the article [29] certainly does not provide such an admissible combination.

Instead of yXTTs, we therefore consider *hedge-to-string transducers (yXHTs)*, an ad-hoc straightforward generalization of yXTTs (not to be confused with the hedge-to-string transducers of [34]). On the one hand, this course of action seems logical; for if we identify a hedge of length 1 with its only tree, then every yXTT is also a yXHT, and its "yXTT meaning" coincides with its "yXHT meaning". On the other hand, we should be aware that existing infrastructure needs to be adapted in order to accommodate yXHTs, e.g., when it comes to computing the input product.

Let Σ be an alphabet and $K \in \mathbb{N}$. We will define the formalism $\mathrm{yXHT}(\Sigma, K)$, which, informally speaking, represents hedge-to-string transducers over Σ. To this

end, let $\mathcal{A}_1$ be the hedge algebra over Σ and K, and let $\mathcal{A}_2$ be the string algebra over Σ and K. We let $\mathrm{yXHT}(\Sigma, K) = (\mathcal{C}, (\mathcal{A}_1, \mathcal{A}_2), \varphi)$ as follows. We let $\mathcal{C}$ be the set of all IRTGs B over (Δ_1, Δ_2) and Real such that, if $B = (\Gamma, M, h)$, then, for every $l \in \mathbb{N}$ and $\alpha \in \Gamma^{(l)}$,

- $\max\{\mathrm{rk}_t(w) \mid w \in \mathrm{pos}(t)\} \leq K$, where $t = h_1(\alpha)^{\mathcal{A}_1}(x_1, \ldots, x_l)$, and
- $|h_2(\alpha)^{\mathcal{A}_2}(x_1, \ldots, x_l)| \leq K$.

Now we define $\varphi\colon \mathcal{C} \to \mathcal{C}$. To this end, let $B \in \mathcal{C}$ and $B = (\Gamma, M, h)$. We construct $\varphi(B) = (\Gamma, M, h')$, where h'_2 is defined as in the SCFG case and, for every $l \in \mathbb{N}$ and $\alpha \in \Gamma^{(l)}$, $h'_1(\alpha) = f(h_1(\alpha)^{\mathcal{A}_1}(x_1, \ldots, x_l))$, where $f\colon H_{\Sigma_1 \cup X}(Y) \to T_{\Delta_1}(X)$ inserts con^k appropriately, i.e., we let

$$\begin{aligned} f(t_1, \ldots, t_k) &= \mathrm{con}^k(f(t_1), \ldots, f(t_k)) \,, && (k \neq 1) \\ f(x_j) &= x_j \,, \\ f(\sigma(t)) &= \sigma(f(t)) \,, \\ f(\sigma(t_1, \ldots, t_k)) &= \sigma(\mathrm{con}^k(f(t_1), \ldots, f(t_k))) \,. && (k \neq 1) \end{aligned}$$

We omit the proof that φ is idempotent and preserves rank, meaning, and rule-by-rule (non)binarizability.

The formalism $\mathrm{yXHT}(\Sigma, K)$ is trivially admissible, for the same reason as in the SCFG case. It is also complete. The binarization mapping $bin_{\mathcal{C}}$ acts as in Ex. 4.6.1.

The author has implemented Alg. 4.2 and the b-rules $\mathfrak{b}_1$ and $\mathfrak{b}_2$. In order to test the implementation, he extracted a yXHT from about a million parallel sentences of English-German Europarl data [109], using the GHKM rule extractor [78]. Then he applied the binarization algorithm to the yXHT. The results are shown in Fig. 4.12: of the 2.15 million rules in the extracted transducer, 460,000 were suprabinary, and 67 % of these had a binarization. Binarization took 4.4 minutes on a single core of an Intel Core i5 2520M processor.

4.6.5 Weighted synchronous context-free hedge grammars

Ideally we would like to embed the class of all WSCFTGs into IRTGs and use the resulting formalism for binarization. However, as in the case of yXTT, and for the same reasons, the author is not aware of an admissible combination of a formalism and b-rules. We follow the same course of action as before: instead of WSCFTGs, we consider an ad-hoc generalization that we call *weighted synchronous context-free hedge grammars (WSCFHGs)*. We note that "WSCFHG" is not to be confused with the context-free hypergraph grammar (CFHG) of [64].

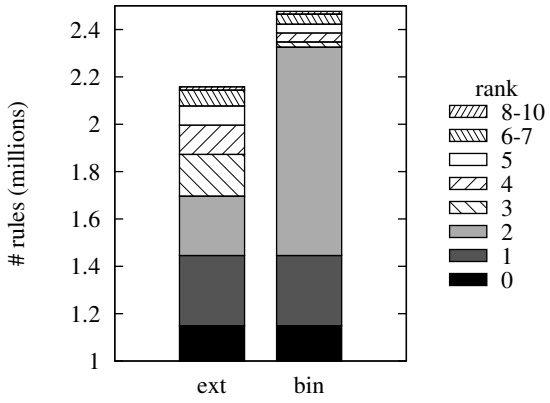


Figure 4.12: Rules of a yXTT extracted from Europarl (ext) vs. its binarization (bin).

Let Σ be an alphabet, $K \in \mathbb{N}$, and $\mathcal{S}$ a complete commutative semiring. We define the formalism $\mathrm{CFHG}(\Sigma, K, \mathcal{S})$, which, informally speaking, represents WSCFHGs over Σ and $\mathcal{S}$. This definition is the spirit of [30, Def. 5, Def. 7], which is in turn inspired by [49, Prop. 4.10] and [69, Lm. 5.8]. Let $\mathcal{A}$ be the hedge algebra with substitution over Σ and K. Moreover, let $m, r_1, \ldots, r_l \in \mathbb{N}$. Then $\mathcal{T}_\Delta(m, r_1, \ldots, r_l)$ denotes the set of all $t \in T_\Delta(X_l)$ such that

- t is linear and nondeleting in Π_m,
- for every $w \in \mathrm{pos}(t)$ and $j \in \mathbb{N}$, if $t(w) = \pi_j$, then $j \leq m$,
- for every $w \in \mathrm{pos}(t)$ and $k \in \mathbb{N}$, if $t(w) = sub^k$, then there is a $\iota \in \{1, \ldots, l\}$ with $t(w1) = x_\iota$ and $k = r_\iota$,
- for every $j \in \{1, \ldots, l\}$, if $r_j \neq 0$, then there is a $w \in \mathrm{pos}(t)$ with $t(w) = sub^{r_j}$ and $t(w1) = x_j$.

In addition, we define the mapping $f' \colon \mathcal{T}_\Delta(m, r_1, \ldots, r_l) \to H_{\Sigma \cup X}(Y)$ by letting

$$f'(t) = t^{\mathcal{A}}(x_1(y_1, \ldots, y_{r_1}), \ldots, x_l(y_1, \ldots, y_{r_l})) \,.$$

Example 4.6.2 Figure 4.13 illustrates f'. □

We let $\mathrm{CFHG}(\Sigma, K, \mathcal{S}) = (\mathcal{C}, (\mathcal{A}, \mathcal{A}), \varphi)$ as follows. We let $\mathcal{C}$ be the set of all IRTGs B over (Δ, Δ) and $\mathcal{S}$ such that if $B = (\Gamma, M, h)$ and $M = (Q, R, \mu, \nu)$, then there is a mapping $\mathrm{rk} \colon Q \to \mathbb{N}$ such that $\nu_q \neq 0$ implies $\mathrm{rk}(q) = 0$, and, for every $(q_1 \cdots q_l, \alpha, q) \in R$ and $i \in \{1, 2\}$,

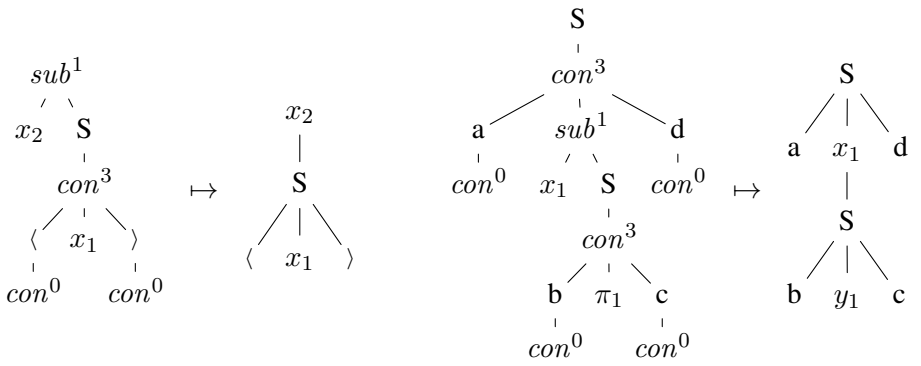


Figure 4.13: Illustration of f'.

- $h_i(\alpha) \in \mathcal{T}_\Delta(\mathrm{rk}(q), \mathrm{rk}(q_1), \ldots, \mathrm{rk}(q_l))$ and
- $\max\{\mathrm{rk}_t(w) \mid w \in \mathrm{pos}(t), t(w) \in \Sigma\} \leq K$ where $t = f'(h_i(\alpha))$.

Now we define $\varphi\colon \mathcal{C} \to \mathcal{C}$. To this end, let $B \in \mathcal{C}$ and $B = (\Gamma, M, h)$. We construct $\varphi(B) = (\Gamma, M, h')$, where, for every $l \in \mathbb{N}$, $\alpha \in \Gamma^{(l)}$, and $i \in \{1, 2\}$,

$$h'_i(\alpha) = f(f'(h_i(\alpha)))$$

and $f\colon H_{\Sigma \cup X}(Y) \to T_\Delta(X)$ inserts con^k and sub^k appropriately, i.e., we let

$$\begin{aligned}
f(t_1, \ldots, t_k) &= \mathrm{con}^k(f(t_1), \ldots, f(t_k))\,, && (k \neq 1)\\
f(x_j) &= x_j\,,\\
f(x_j(t_1, \ldots, t_k)) &= \mathrm{sub}^k(x_j, f(t_1), \ldots, f(t_k))\,, && (k \neq 0)\\
f(\sigma(t_1, \ldots, t_k)) &= \sigma(f(t_1))\,, && (k = 1)\\
f(\sigma(t_1, \ldots, t_k)) &= \sigma(\mathrm{con}^k(f(t_1), \ldots, f(t_k)))\,. && (k \neq 1)
\end{aligned}$$

We omit the proof that φ is idempotent and preserves rank, meaning, and rule-by-rule (non)binarizability.

Example 4.6.3 By reversing the arrows in Fig. 4.13, we obtain an illustration of f. In fact, $f'(f(u)) = u$, and one might say that $f(u)$ is a particularly designated element of $f'^{-1}(u)$. □

The formalism $\mathrm{CFHG}(\Sigma, K, \mathcal{S})$ is trivially admissible, for the same reason as in the SCFG and yXHT cases. It is not complete; for instance, the term

$$\mathrm{sub}^2(x_1, \sigma(\mathrm{con}^0), \sigma(\mathrm{con}^0))$$

is mapped to the empty set by our b-rule, although the term itself has a binary variable tree, namely x_1. However, we can define a restricted variant of $\mathrm{CFHG}(\Sigma, K, \mathcal{S})$ where the use of sub^k and π_k with $k > 1$ is banned; this variant might be called "synchronous hedge-adjoining grammar". With the restricted variant, we have both admissibility and completeness.

We indicate that we can embed the WSCFTGs of Ch. 3 into $\mathrm{CFHG}(\Sigma, K, \mathcal{S})$. To this end, let $M = (Q, R, \mu, \nu)$ be a WSCFTG over Σ and $\mathcal{S}$ such that for every $(q_1 \cdots q_l, \langle \zeta_1 \zeta_2 \rangle, q) \in R$ and $i \in \{1, 2\}$, we have that

- $\max\{\mathrm{rk}_{\zeta_i}(w) \mid w \in \mathrm{pos}(\zeta_i), \zeta_i(w) \in \Sigma\} \leq K$.

We construct the IRTG $B = (\Gamma, M, h)$ over Δ and $\mathcal{S}$ where

- $\Gamma = \{\gamma^{(k)} \mid \exists q_1, \ldots, q_k, q \in Q \colon (q_1 \cdots q_k, \gamma, q) \in R\}$,
- $h_i(\langle \zeta_1 \zeta_2 \rangle) = f(\zeta_i)$.

Then $B \in \mathrm{CFHG}(\Sigma, K, \mathcal{S})$. We omit the proof that $[\![B]\!] = [\![M]\!]$.

Conversely, under suitable conditions, we can convert back from $\mathrm{CFHG}(\Sigma, K, \mathcal{S})$ into WSCFTG. To this end, we employ the normal form via φ as well as the normal form via ψ established below Ex. 4.2.3. Let $B \in \psi(\varphi(\mathcal{C}))$, $B = (\Gamma, M, h)$, and $M = (Q, R, \mu, \nu)$. Then we construct the quadruple $G = (Q, R', \mu', \nu)$ with

- $R' = \{(q_1 \cdots q_l, \langle f'(\gamma_1) f'(\gamma_2) \rangle, q) \mid (q_1 \cdots q_l, \gamma, q) \in R\}$ and
- $\mu'(q_1 \cdots q_l, \langle f'(\gamma_1) f'(\gamma_2) \rangle, q) = \mu(q_1 \cdots q_l, \gamma, q)$.

Note that μ' is well defined because the mapping with $\gamma \mapsto \langle f'(\gamma_1) f'(\gamma_2) \rangle$ is injective. Let $f'(\gamma_i)$ be a hedge of length 1 for every $\gamma \in \Gamma$ and $i \in \{1, 2\}$. Then G is a WSCFTG. We omit the proof that $[\![B]\!] = [\![G]\!]$.

Example 4.6.4 Recall the WSCFTG of Ex. 3.3.1. The application of f' in Fig. 4.13 yields the input trees of the rules ρ_1 and ρ_3. □

4.7 Conclusion, discussion, and outlook

We have developed a template that gives rise to a class of efficiently computable binarization mappings for IRTGs, given b-rules. If the b-rules are complete in a sense, then these mappings are rule-by-rule complete, which is on par with the state of the art. We have shown how to apply this technology for deriving rule-by-rule complete binarization mappings for established formalisms, such as SCFGs. In the process, we

discovered that yXHT and WSCFHGs are better suited for binarization than the conventional formalisms yXTT and WSCFTG, respectively.

As mentioned in the introduction, binarization is used to speed up operations that occur in a typical decoder. Unfortunately, the binarization domain of a rule-by-rule complete binarization mapping need not contain every grammar, in particular for $n \geq 2$. According to [54], there is an alternative way of improving the runtime of such a decoder that only requires binarization for the case $n = 1$, however at the price that the result is approximate.

As mentioned at the very top, this chapter is a greatly expanded version of [29]. The author would like to point out a mistake in said publication: it claims that it provides a binarization mapping for yXTTs, which is untrue; just like this chapter, it provides a binarization mapping for yXHTs.

In [113], the authors provide an algebra for representing STAGs as IRTGs. In contrast to our hedge algebra with substitution, that algebra is more akin to the Σ-term algebra, i.e., its domain only covers ranked trees, and it does not provide a string concatenation operation. Consequently, the IRTGs using this algebra are indeed close to STAGs, as opposed to the IRTGs using our algebra, which we therefore call weighted synchronous context-free *hedge* grammars. We stress that this deviation is on purpose, for WSCFHGs are better suited for binarization than WSCFTGs, as illustrated already for yXTTs in Ex. 4.6.1.

There are at least six items for further research, which we discuss in the following.

First, one might investigate input and output products for the formalisms yXHT and WSCFHG. To this end, one could start off with existing work for unweighted IRTGs [112, 113] and "add weights", or one could start off with existing work for yXTTs or WSCFTGs (e.g., from Ch. 3) and "add hedges".

Second, it would be interesting how to select a binarization mapping for "space-optimal" binarization. To elucidate this problem, we recall the CFG example from Sec. 4.1, where we replaced

$$A \to BCDE \quad \text{by} \quad A \to [[BC]D]E\,, \quad [[BC]D] \to [BC]D\,, \quad [BC] \to BC\,.$$

If the next replacement was to be

$$D \to EBCD \quad \text{by} \quad D \to E[[BC]D]\,,$$

then we could reuse the nonterminal $[[BC]D]$ as well as the corresponding rule. If, however, we encounter the rule

$$E \to ECDB\,,$$

then we cannot reuse any nonterminal. Then our first replacement had better been

$$A \to BCDE \quad \text{by} \quad A \to [B[CD]]E\,, \quad [B[CD]] \to B[CD]\,, \quad [CD] \to CD\,,$$

so that $[CD]$ could be reused. It has been suggested that keeping the number of nonterminals of the binarized grammar low also reduces parsing time [174].

In our template, reuse of nonterminals, or rather states, happens automatically due to the way we construct the binarization $B(\rho, t)$ of a rule ρ from a binarization hedge t. The interesting question is whether we replace $A \to BCDE$ in the former or the latter way, and this is controlled by the selection of the variable tree in Line 5 of Alg. 4.2. It has been stated that finding a space-optimal grammar is impractical, because it cannot be done rule by rule [97, p. 568]. Therefore, it would be interesting to investigate (and evaluate empirically) criteria that can be satisfied more easily, e.g., using a heuristic.

The third item is concerned with a formalism that is close to WSCFHGs, but better suited for binarization. Instead of the operation sub^k that replaces the variables $y_1, \ldots, y_k$ all at once, one could use the operation $sub_{Y'}$, $Y' \subseteq Y_K$, that replaces exactly the variables in Y' (it is a small technicality to specify which argument corresponds to which variable). Then the operations sub^k and sub_{Y_k} coincide, but, contrary to the former, the latter kind of operation can be easily decomposed, e.g.,

$$sub_{Y_2}(t, t_1, t_2) = sub_{\{y_2\}}(sub_{\{y_1\}}(t, t_1), t_2)$$

for every $t, t_1, t_2 \in H_{\Sigma \cup X}(Y)$. This kind of decomposition can be captured on the syntactic level (where $sub_{Y'}$ is represented by the symbol $\mathrm{sub}_{Y'}$) by a b-rule. Based on this new kind of substitution operation, we can define a formalism close to WSCFHG. Figure 4.14 shows what a rule of this formalism might look like, as well as a binarization of this rule. This example shows that a ranked alphabet is no longer sufficient to type the states; roughly speaking, a tree pair (t_1, t_2) described by q' now contains exactly the variable y_2 in t_1 and the variable y_1 in t_2. It is this heterogeneity that makes the new formalism binarization-friendly.

Fourth, one might investigate whether it is worthwhile to introduce a type system for IRTGs. For instance, we might consider a sequence $T = (T_1, \ldots, T_n)$ where T_i is a (bottom-up deterministically) recognizable tree language over Δ_i, whose elements we might call correctly-typed terms. Then an IRTG (Γ, M, h) over T and $\mathcal{S}$ is an IRTG over Δ and S such that $h_i(L(M)) \subseteq T_i$. This property is decidable, because linear tree homomorphisms preserve recognizability [80, Prop. 7.8] and inclusion of recognizable tree languages is decidable [80, Prop. 5.3, Prop. 7.1].

For instance, let us consider the hedge algebra with substitution over Σ and K. Each element u of $H_{\Sigma \cup X}(Y_K)$ can be categorized as follows:

$$(q_1 q_2 q_3, \alpha, q)$$

$$\mathrm{sub}_{Y_2}(x_1, x_2, x_3) \xleftarrow{h_1} \alpha \xmapsto{h_2} \mathrm{sub}_{Y_2}(x_1, x_3, x_2)$$

$$(q' q_3, \alpha', q)\ , \quad (q_1 q_2, \alpha'', q')$$

$$\mathrm{sub}_{\{y_2\}}(x_1, x_2) \xleftarrow{h_1} \alpha' \xmapsto{h_2} \mathrm{sub}_{\{y_1\}}(x_1, x_2)$$
$$\mathrm{sub}_{\{y_1\}}(x_1, x_2) \xleftarrow{h_1} \alpha'' \xmapsto{h_2} \mathrm{sub}_{\{y_2\}}(x_1, x_2)$$

Figure 4.14: A rule and its binarization in a binarization-friendly WSCFHG variant.

$$\begin{aligned}
(q, \sigma, q) \qquad & \text{if } \sigma \in \Sigma\ , \\
(\varepsilon, \pi_k, \{y_k\}) & \\
(q_1 \cdots q_k, \mathrm{con}^k, q) \qquad & \text{if (i) } P(q_1, \ldots, q_k, q), q = \textstyle\bigcup_j q_j \\
& \quad \text{or (ii) } \neg P(q_1, \ldots, q_k, q), q = \bot \\
(Y' q_1 \cdots q_k, \mathrm{sub}_{Y'}, q) \qquad & \text{if (i) or (ii) as above}\ ,
\end{aligned}$$

where $P(q_1, \ldots, q_k, q)$ iff $q_1, \ldots, q_k, q \subseteq Y_K$ and $q_j \cap q_{j'} \neq \emptyset$ implies $j = j'$.

Figure 4.15: Transitions of the FTA for correctly-typed terms.

- either there is a $Y' \subseteq Y_K$ such that $u \in H_{\Sigma \cup X}(Y')$ and y occurs exactly once in u for every $y \in Y'$;
- or we have no use for u.

Correspondingly, we define the bu-det FTA $G = (Q, R, \emptyset)$ with $Q = \{\bot\} \cup \mathcal{P}(Y_K)$ and R in Fig. 4.15. Now we may define the formalism $\mathrm{CFHG}(\Sigma, K, \mathcal{S})$, or the variant mentioned in the third item, in terms of IRTGs over $(L(G), L(G))$ and $\mathcal{S}$, and then we can dispense with the mapping $\mathrm{rk}\colon Q \to \mathbb{N}$ that we currently use.

We note that a more restricted alternative to this kind of type system would be many-sorted algebras. In that setting, the FTA for the set T_j has exactly one transition for each operation symbol; such an FTA is called a (many-sorted) signature, and its states are called sorts. We can adapt above FTA G to this setting by enriching the operation symbols; e.g., we replace the transition

$$(Y' q_1 \cdots q_k, \mathrm{sub}_{Y'}, q) \qquad \text{by} \qquad (Y' q_1 \cdots q_k, \mathrm{sub}_{Y', q_1 \cdots q_k}, q)\ .$$

Fifth, it is an interesting question how the binarization framework established in this chapter can be generalized. For instance, one might "k-arize", i.e., reduce the rank of

a grammar to a fixed k or even some "best possible" value that can be achieved using a rule-by-rule technique. In our terminology, the latter problem consists in constructing a mapping from IRTGs into IRTGs that replaces each rule by an equivalent collection of rules of minimal rank. Naturally, there is no reason for such a mapping to be partial, as opposed to a binarization mapping. In the case of STAGs, this problem has been dealt with [147].

Another interesting generalization would be to reduce parsing complexity instead of rank. Recall from the introduction that the rank of a grammar occurs as an exponent in the parsing complexity. We note that other factors can be crucial as well. For instance, for LCFRSs and WSCFTGs, the maximum rank of any state (called "fanout" for LCFRS) also plays a key role in said complexity, and reducing the rank alone need not be optimal. The topics of reducing (a) fanout and (b) parsing complexity of LCFRSs have been addressed in [88] and [83], respectively.

Finally, one might investigate IRTGs with deleting homomorphisms.

5 Determinizing weighted tree automata using factorizations

This chapter is an extensively revised version of [31] and [27].

5.1 Introduction

The determinization problem for WTA over Σ and $\mathcal{S}$ consists in, given a WTA M, finding a WTA M' that is bu-det and equivalent to M. Before we elaborate on what is known about this problem, let us consider the advantages and disadvantages of WTA that are bu-det, as opposed to WTA that are not bu-det. In other words, let us consider what we can expect from determinization.

We begin with the advantages. First, bu-det WTA are unambiguous (Obs. 2.4.3). Consequently, the n best complete runs of a bu-det WTA correspond to the n best trees. As argued in Sec. 1.5.3, this paves the way for syntax-based decoders that aim at the best translation rather than some translation with the best run. Second, efficient minimization of WTA either requires that the WTA be bu-det or that the semiring be a field [126]. Third, bu-det WTA and their meaning can be implemented using space-efficient data structures, because the weight vector $[\![t]\!]$ of a tree t has at most one nonzero component (Obs. 2.4.3).

As for the disadvantages, we observe that WTA generalize weighted finite-state (string) automata (WSA), which in turn generalize finite-state automata (FSA). It is well known that deterministic FSA are as powerful as general FSA, but nondeterministic FSA are exponentially more succinct than deterministic ones. More formally, for each n there is an FSA with n states whose minimal equivalent deterministic FSA has 2^n states [182, p. 102]. In the weighted case, it is known that there are WSA for which an equivalent deterministic WSA does not even exist [22, Sec. 1]. Naturally, we cannot expect the situation to be any better for WTA than for FSA or WSA.

The determinization problem can, therefore, only be solved partially, i.e., for subclasses of WTA. Table 5.1 shows known results in this respect. Given a WSA/WTA M, each of the underlying constructions defines an object M' that differs from a WSA/WTA in one respect only: the set of states and the set of transitions can be infinite. If, however, these sets are finite, then M' is indeed a solution to the problem, i.e.,

publication	device	restriction	semiring restriction	remarks
[158]	FSA	–	Boolean	
[80, Sec. 5]	FTA	–	Boolean	
[138, 22, 139]	WSA	twins property	tropical	
[18]	WTA	–	locally finite semifield	
[15]	WTA	–	locally finite	
[105]	WSA	twins property	commutative, extremal	(1)
[134]	WTA	acyclic	nonnegative reals	(2)
(this chapter)	WTA	acyclic	commutative	
(this chapter)	WTA	–	locally finite	
(this chapter)	WTA	twins property	commutative, extremal	(1)

legend: (1) requires a maximal factorization from the user
(2) lacks formal proof

Table 5.1: Results concerning determinization of WTA subclasses.

it is a deterministic WSA/bu-det WTA equivalent to M. The requirements mentioned in the table ensure that this is the case, i.e., that M' is a solution.

As is the case for FSA, determinization of FTA is accomplished using the powerset construction. Determinization of WTA was first described by Borchardt and Vogler [18]. They used a Myhill-Nerode approach, which is restricted to semifields, and they showed that their construction yields a WTA if the semifield is locally finite. Borchardt [15] extended this result to locally finite semirings by generalizing the powerset construction. In his method, the states of M' simulate the Σ-algebra $\mathcal{M}$ associated with M. If the semiring is not locally finite, this may yield an infinite set of states. Let us exemplify this method using the WTA M of Ex. 2.4.2. For this, let $T = \{t_n \mid n \in \mathbb{N}\}$. The new set Q' of states is obtained as follows:

$$\begin{aligned} Q' &= [\![T_\Sigma]\!] = [\![T_\Sigma \setminus T]\!] \cup [\![T]\!] \\ &= \{\begin{pmatrix} 0 \\ 0 \end{pmatrix}\} \cup \{\begin{pmatrix} 1 \\ 0.2 \end{pmatrix}, \begin{pmatrix} 0 \\ 0.1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0.05 \end{pmatrix}, \begin{pmatrix} 0 \\ 0.025 \end{pmatrix}, \ldots\} \,. \end{aligned}$$

Part of the "infinite WTA" resulting from the construction is shown in Fig. 5.1. Notice how this "WTA" mimics the calculation in $\mathcal{M}$ using its states.

Borchardt's method has one obvious drawback: it does not use the full capacity of WTA, because the transition weights are "crisp", i.e., either 0 or 1. Another generalization of the powerset construction to the weighted case goes further by using the

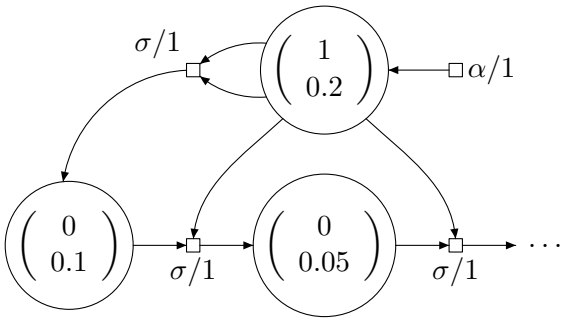


Figure 5.1: "Infinite WTA" obtained via Borchardt's method.

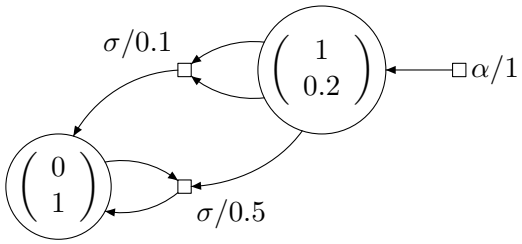


Figure 5.2: Bu-det WTA obtained via factorization.

concept of a factorization. In order to elucidate this approach, let us consider an example in the realm of WTA (anticipating this chapter's results). Roughly speaking, instead of moving the complete computation of weights from $\mathcal{M}$ into the new states, we factor the elements of S^Q so that the transition mapping in the new automaton is equipped with the factor common to all components. When we apply this method to the WTA of Ex. 2.4.2, we obtain the bu-det WTA of Fig. 5.2.

The first method to use the factorization approach, albeit implicitly, was the one by Mohri for WSA over the tropical semiring [138]. Later, Kirsten and Mäurer [105] made the notion of a factorization explicit. This way, they were able to generalize Mohri's method to commutative semirings where $a + b \in \{a, b\}$ holds; this property is called extremal [124]. The factorization is a user-supplied parameter that depends on the semiring. For zero-sum free semifields, a suitable factorization is readily available.

Both Mohri's and Kirsten and Mäurer's method yields a WSA if M has a certain property that is called twins property [44]. The question whether the twins property is decidable has remained open for a long time. Decision procedures existed for subclasses of WSA, namely for trim, unambiguous WSA over the tropical semiring [138, Thm. 13] and for trim, cycle-unambiguous WSA over commutative, cancellative semi-

rings [5]. Only recently, Kirsten [104] provided a decision procedure for general WSA over the tropical semiring. He also showed that the decision problem is PSPACE-complete.

May and Knight [134] transferred Mohri's method to acyclic WTA over the semiring of nonnegative reals, and they provided empirical evidence that their algorithm was effective in machine translation and parsing systems, but they did not provide a formal proof of correctness.

In this chapter, we use the factorization approach of [138, 105] to develop a determinization construction for WTA that subsumes the above results; cf. Tab. 5.1. Moreover, we transfer the aforementioned decision results regarding the twins property from WSA to WTA; in particular, we show that the twins property is decidable (i) for cycle-unambiguous WTA over commutative, zero-sum-free, zero-divisor-free semirings (generalizing [5]) and (ii) for WTA over extremal semifields (generalizing [104]).

We proceed in the following four steps. First, we formalize and investigate the necessary notions, such as factorizations, extremal semirings, and the twins property (Sec. 5.2). Second, we develop our determinization construction for the case of classical WTA and prove its correctness (Sec. 5.3). Third, we develop our decision results, again for classical WTA (Sec. 5.4). Finally, we transfer the results from classical WTA to arbitrary (i.e., not necessarily classical) WTA (Sec. 5.5).

We end this chapter with a conclusion, discussion, and outlook (Sec. 5.6).

5.2 Preliminary notions and results

Let Σ be an alphabet, $\mathcal{S} = (S, +, \cdot, 0, 1)$ a semiring, and $M = (Q, R, \mu, \nu)$ a WTA over Σ and $\mathcal{S}$. Recall from Sec. 2.3.2 that $\mathcal{S}^Q$ is a semiring and that $\tilde{0}$ is the vector that consists of 0's only.

5.2.1 Factorizations

We adopt the notion of a factorization from [105]. Let Q be a nonempty finite set. A pair (f, g) is a *factorization (of dimension Q)* if

- $f\colon S^Q \setminus \{\tilde{0}\} \to S^Q$,
- $g\colon S^Q \setminus \{\tilde{0}\} \to S$, and
- $u = g(u) \cdot f(u)$ for every $u \in S^Q \setminus \{\tilde{0}\}$.

A factorization (f, g) is called *maximal* if for every $u \in S^Q$ and $s \in S$, we have that $s \cdot u \neq \tilde{0}$ implies $f(u) = f(s \cdot u)$. Note that even if $f(\tilde{0})$ were defined, the case

$s \cdot u = \tilde{0}$ would still have to be excluded here, because otherwise we would obtain that $f(u) = f(0 \cdot u) = f(\tilde{0})$ for every $u \in S^Q$. The *trivial factorization* is the factorization (f, g) with $f(u) = u$ and $g(u) = 1$. We will abbreviate $f([\![t]\!])$ and $g([\![t]\!])$ by $f[\![t]\!]$ and $g[\![t]\!]$, respectively.

Lemma 5.2.1 *For every $u \in S^Q \setminus \{\tilde{0}\}$, we have that $f(u) \neq \tilde{0}$ and $g(u) \neq 0$.*

PROOF. By contradiction. Let $f(u) = \tilde{0}$ or $g(u) = 0$. Then $u = f(u) \cdot g(u) = \tilde{0}$, which contradicts the assumption that $u \neq \tilde{0}$. ∎

The following lemma shows a maximal factorization in the case that $\mathcal{S}$ is a semifield and that there is a certain binary operation $+'$ on S. In particular, the lemma applies when $\mathcal{S}$ is zero-sum free and $+' = +$.

Lemma 5.2.2 *Let $\mathcal{S}$ be a semifield, $c \in S \setminus \{0\}$, $+'$ an associative, commutative binary operation on S such that (i) $s \cdot (s_1 +' s_2) = s \cdot s_1 +' s \cdot s_2$ and (ii) $s_1 +' s_2 = 0$ implies $s_1 = 0$ and $s_2 = 0$; and let (f, g) be the factorization with $g(u) = c \cdot \sum'_{q \in Q} u_q$ and $f(u) = g(u)^{-1} \cdot u$, where $\sum'$ is computed with respect to $+'$. Then (f, g) is maximal.*

PROOF. First, we show that (f, g) is a factorization. Let $u \in S^Q \setminus \{\tilde{0}\}$. Since $\mathcal{S}$ is a semifield and $+'$ is "zero-sum free", we obtain that $g(u) \neq 0$ and, hence, $g(u) \cdot f(u) = g(u) \cdot g(u)^{-1} \cdot u = u$. Second, we show that (f, g) is maximal. Let $s \in S$ with $s \cdot u \neq \tilde{0}$, and let $q \in Q$. Then

$$\begin{aligned}
[f(s \cdot u)]_q &= \left[g(s \cdot u)^{-1} \cdot s \cdot u\right]_q = \left(c \cdot \textstyle\sum'_{q' \in Q} s \cdot u_{q'}\right)^{-1} \cdot s \cdot u_q \\
&= \left(c \cdot s \cdot \textstyle\sum'_{q' \in Q} u_{q'}\right)^{-1} \cdot s \cdot u_q = \left(s \cdot c \cdot \textstyle\sum'_{q' \in Q} u_{q'}\right)^{-1} \cdot s \cdot u_q \\
&= \left(c \cdot \textstyle\sum'_{q' \in Q} u_{q'}\right)^{-1} \cdot s^{-1} \cdot s \cdot u_q = g(u)^{-1} \cdot u_q = [f(u)]_q \ .
\end{aligned}$$ ∎

Example 5.2.3 First, we consider four instances of Lm. 5.2.2 where $+' = +$. The factorization (f, g) is maximal if

1. $\mathcal{S}$ is the semiring $(\mathbb{R}^{\geq 0}, +, \cdot, 0, 1)$ of nonnegative reals, $g(u) = \sum_{q \in Q} u_q$, and $f(u) = \frac{1}{g(u)} \cdot u$;
2. $\mathcal{S}$ is the semiring $(\mathbb{R}^{\geq 0}, \max, \cdot, 0, 1)$, $g(u) = \sum_{q \in Q} u_q$, and $f(u) = \frac{1}{g(u)} \cdot u$;
3. $\mathcal{S}$ is the Viterbi semiring $([0, 1], \max, \cdot, 0, 1)$, $g(u) = \max\{u_q \mid q \in Q\}$, and $f(u) = \frac{1}{g(u)} \cdot u$;

4. $\mathcal{S}$ is the tropical semiring $(\mathbb{R}_{\infty}^{\geq 0}, \min, +, \infty, 0)$, $g(u) = \min\{u_q \mid q \in Q\}$, and $f(u) = -g(u) + u$.

The settings in [134] and [138] correspond to Cases 1 and 4, respectively. Their constructions implicitly employ the corresponding maximal factorization given here.

Second, we consider an instance of the lemma where $+' \neq +$, namely when $\mathcal{S}$ is the field $(\mathbb{R}, +, \cdot, 0, 1)$ of real numbers and $+' = \max$. We note that $\mathcal{S}$ itself, being a field, is not zero-sum free. In addition, we note that the neutral element of $\max$, which is not a real number, does not play a role in this scenario because, as Q is nonempty, we never compute the maximum of the empty set. □

The following lemma shows that (apart from the case that $|Q| \leq 1$) maximal factorizations only exist for zero-divisor-free semirings.

Lemma 5.2.4 *Let $\mathcal{S}$ be commutative and (f, g) a maximal factorization. Then $|Q| \leq 1$ or $\mathcal{S}$ is zero-divisor free.*

PROOF. By contradiction. Assume that $|Q| > 1$ and $\mathcal{S}$ has zero divisors, i.e., $s_1, s_2 \in S \setminus \{0\}$ such that $s_1 \cdot s_2 = 0$. We choose a pair $q_1, q_2 \in Q$ with $q_1 \neq q_2$. This is possible because $|Q| > 1$. We define the vectors $u_1, u_2 \in S^Q$ such that the q_i-component of u_i is 1 while the other components are 0. Since (f, g) is maximal and $s_1 \cdot s_2 = 0$, we have that

$$\begin{aligned} f(u_1) &= f(s_1 \cdot u_1) = f(s_1 \cdot u_1 + (s_1 \cdot s_2) \cdot u_2) = f(s_1 \cdot (u_1 + s_2 \cdot u_2)) \\ &= f(u_1 + s_2 \cdot u_2) \ . \end{aligned}$$

Thus, and since (f, g) is a factorization, we obtain the following equations (where $u = u_1 + s_2 \cdot u_2$)

$$g(u_1) \cdot f(u_1)_{q_1} = [u_1]_{q_1} = 1 \tag{I}$$
$$g(u_1) \cdot f(u_1)_{q_2} = [u_1]_{q_2} = 0 \tag{II}$$
$$g(u) \cdot f(u_1)_{q_1} = [u]_{q_1} = 1 \tag{III}$$
$$g(u) \cdot f(u_1)_{q_2} = [u]_{q_2} = s_2 \tag{IV}$$

By (II), (IV), and $s_2 \neq 0$, we derive that $g(u_1) \neq g(u)$. By (I) and (III), and using commutativity, we derive

$$g(u_1) = g(u_1) \cdot \big(f(u_1)_{q_1} \cdot g(u)\big) = \big(g(u_1) \cdot f(u_1)_{q_1}\big) \cdot g(u) = g(u) \ .$$

Thus, we have a contradiction, proving that $|Q| > 1$ or $\mathcal{S}$ zero-divisor free. ■

Let $|Q| > 1$. Then Lm. 5.2.4 yields that commutative semirings with zero divisors do not admit maximal factorizations, e.g., Semiring 8 of Ex. 2.3.5. The following example shows that even zero-divisor-free semirings can defy a maximal factorization.

Example 5.2.5 ([105, Sec. 3.5]) Let S' be the set of all natural numbers that can be factored into an even number of primes, e.g., $4 = 2 \cdot 2$ and $126 = 2 \cdot 3 \cdot 3 \cdot 7$ belong to S', but 2 and $18 = 2 \cdot 3 \cdot 3$ do not. Let $S = S' \cup \{1, \infty\}$. Then $(S, \min, \cdot, \infty, 1)$ is a semiring, where $\min$ is defined by the usual ordering of natural numbers, $\cdot$ is the usual multiplication of natural numbers, and ∞ denotes a new maximal element.

Consider the following chain of equations with vectors u and u_1, u_2, u_3:

$$\underbrace{\begin{pmatrix} 2 \cdot 3 \cdot 5 \cdot 7 \\ 3 \cdot 5 \cdot 7 \cdot 11 \end{pmatrix}}_{u} = (3 \cdot 5) \cdot \underbrace{\begin{pmatrix} 2 \cdot 7 \\ 7 \cdot 11 \end{pmatrix}}_{u_1} = (5 \cdot 7) \cdot \underbrace{\begin{pmatrix} 2 \cdot 3 \\ 3 \cdot 11 \end{pmatrix}}_{u_2} = (3 \cdot 7) \cdot \underbrace{\begin{pmatrix} 2 \cdot 5 \\ 5 \cdot 11 \end{pmatrix}}_{u_3} .$$

Clearly, the vectors u_1 up to u_3 can not be factored any further in S. Hence $g(u_i) = 1$ and $f(u_i) = u_i$ for every factorization (f, g). Now let (f, g) be a maximal factorization. We apply f to the above equation, obtaining

$$f(u) = f(u_1) = f(u_2) = f(u_3) .$$

Since $f(u_i) = u_i$, we obtain $u_1 = u_2 = u_3$, which is obviously a contradiction. Hence, there is no maximal factorization. □

We will frequently use the following observation, which can be shown by elementary calculations.

Observation 5.2.6 *Let $k \in \mathbb{N}$, $\sigma \in \Sigma$, $u_1, \ldots, u_k \in S^Q$, and $s_1, \ldots, s_k \in S$. If $s_1, \ldots, s_k \in \{0, 1\}$ or $\mathcal{S}$ is commutative, we have that $[\![\sigma(s_1 \cdot u_1, \ldots, s_k \cdot u_k)]\!] = s_1 \cdots s_k \cdot [\![\sigma(u_1, \ldots, u_k)]\!]$.*

We will use the following two lemmas.

Lemma 5.2.7 *Let (f, g) and $(\tilde{f}, \tilde{g})$ be factorizations, (f, g) maximal, and let $u \in S^Q \setminus \{\tilde{0}\}$. Then $f(\tilde{f}(u)) = f(u)$. In particular, $f(f(u)) = f(u)$.*

PROOF. We apply that (f, g) is maximal and that $(\tilde{f}, \tilde{g})$ is a factorization:

$$f(\tilde{f}(u)) = f(\tilde{g}(u) \cdot \tilde{f}(u)) = f(u) .$$ ■

Lemma 5.2.8 *Let $\mathcal{S}$ be commutative and (f, g) maximal. Furthermore, let $k \in \mathbb{N}$, $\sigma \in \Sigma$, $u_1, \ldots, u_k \in S^Q$, and $u'_1, \ldots, u'_k \in S^Q$ such that $u'_i \in \{u_i, f(u_i)\}$. Then*

$$[\![\sigma(u_1, \ldots, u_k)]\!] \neq \tilde{0} \implies [\![\sigma(u'_1, \ldots, u'_k)]\!] \neq \tilde{0}\,,$$

and the converse holds if $\mathcal{S}$ zero-divisor free. Furthermore,

$$[\![\sigma(u_1, \ldots, u_k)]\!] \neq \tilde{0} \implies f[\![\sigma(u_1, \ldots, u_k)]\!] = f[\![\sigma(u'_1, \ldots, u'_k)]\!]\,.$$

PROOF. We construct the sequence $s_1, \ldots, s_k \in S$ by letting

$$s_i = \begin{cases} g(u_i) & \text{if } u'_i = f(u_i)\,, \\ 1 & \text{otherwise.} \end{cases}$$

Using that (f, g) is a factorization and employing Obs. 5.2.6, we derive ($\star$):

$$[\![\sigma(u_1, \ldots, u_k)]\!] = [\![\sigma(s_1 \cdot u'_1, \ldots, s_k \cdot u'_k)]\!] = s_1 \cdots s_k \cdot [\![\sigma(u'_1, \ldots, u'_k)]\!]\,.$$

First, let $[\![\sigma(u_1, \ldots, u_k)]\!] \neq \tilde{0}$. By ($\star$) also $[\![\sigma(u'_1, \ldots, u'_k)]\!] \neq \tilde{0}$. Then

$$\begin{aligned} f[\![\sigma(u_1, \ldots, u_k)]\!] &= f(s_1 \cdots s_k \cdot [\![\sigma(u'_1, \ldots, u'_k)]\!]) && (\star) \\ &= f[\![\sigma(u'_1, \ldots, u'_k)]\!]\,. && ((f, g) \text{ maximal}) \end{aligned}$$

Second, let $\mathcal{S}$ be zero-divisor free and $[\![\sigma(u'_1, \ldots, u'_k)]\!] \neq \tilde{0}$. Since $s_i \neq 0$, ($\star$) yields that $[\![\sigma(u_1, \ldots, u_k)]\!] \neq \tilde{0}$. ∎

5.2.2 Extremal semirings

Observation 5.2.9 *If $\mathcal{S}$ is extremal, then it is zero-sum free and idempotent.*

For every $D \subseteq D(M)$ and $d \in D$ we call d *victorious on* D if $\langle d \rangle = \sum_{d \in D} \langle d \rangle$. Let $t \in T_\Sigma$, $d \in D(M, t)$, and $d = (q_1 \cdots q_k, \sigma, q)\big(d_1, \ldots, d_k\big)$. We call d *recursively victorious* if d is victorious on $D^q(M, t)$ and d_i is recursively victorious.

Observation 5.2.10 *If $\mathcal{S}$ is extremal, then, for every $t \in T_\Sigma(S^Q)$ and $q \in Q$, there is a victorious run d on $D^q(M, t)$. Consequently, $\langle d \rangle = [\![t]\!]_q$.*

Lemma 5.2.11 *Let $\mathcal{S}$ be extremal. For every $m \in \mathbb{N}$, $q \in Q$ and $t \in T_\Sigma$ with $|\mathrm{pos}(t)| \leq m$, there is a recursively victorious run $d \in D^q(M, t)$.*

PROOF. By induction (on m). For the induction base ($m = 0$), there is nothing to show. We show the induction step ($m \to m+1$). To this end, let $m \in \mathbb{N}$, $q \in Q$, $t \in T_\Sigma$, $t = \sigma(t_1, \ldots, t_k)$, and $|\mathrm{pos}(t)| \leq m+1$. By Obs. 5.2.10, there is a victorious run d' on $D^q(M,t)$. Let $d'(\varepsilon) = (q_1 \cdots q_k, \sigma, q)$. By the induction hypothesis, there are recursively victorious runs $d_1 \in D^{q_1}(M,t_1)$ up to $d_k \in D^{q_k}(M,t_k)$. We construct $d = d'(\varepsilon)\big(d_1, \ldots, d_k\big)$. It remains to show that d is victorious. We derive

$$
\begin{aligned}
\langle d' \rangle &= \textstyle\sum_{d'' \in D^q(M,t)} \langle d'' \rangle && (d' \text{ victorious}) \\
&= \langle d' \rangle + \big(\textstyle\sum_{d'' \in D^q(M,t)\colon \langle d'' \rangle \neq \langle d' \rangle} \langle d'' \rangle\big) && (\mathcal{S} \text{ idempotent}) \\
&= \Big(\langle d' \rangle + \big(\textstyle\sum_{d'' \in D^q(M,t)\colon \langle d'' \rangle \neq \langle d' \rangle, d''(\varepsilon) = d'(\varepsilon)} \langle d'' \rangle\big)\Big) \\
&\qquad + \big(\textstyle\sum_{d'' \in D^q(M,t)\colon \langle d'' \rangle \neq \langle d' \rangle, d''(\varepsilon) \neq d'(\varepsilon)} \langle d'' \rangle\big) \\
&= \langle d' \rangle + \big(\textstyle\sum_{d'' \in D^q(M,t)\colon \langle d'' \rangle \neq \langle d' \rangle, d''(\varepsilon) = d'(\varepsilon)} \langle d'' \rangle\big) && (\dagger) \\
&= \textstyle\sum_{d'_1 \in D^{q_1}(M,t_1), \ldots, d'_k \in D^{q_k}(M,t_k)} \langle d'(\varepsilon)\big(d'_1, \ldots, d'_k\big) \rangle && (\mathcal{S} \text{ idempotent}) \\
&= \big(\textstyle\sum_{d'_1 \in D^{q_1}(M,t_1)} \langle d'_1 \rangle\big) \cdots \big(\textstyle\sum_{d'_k \in D^{q_k}(M,t_k)} \langle d'_k \rangle\big) \cdot \mu(d'(\varepsilon)) && (\text{distributivity}) \\
&= \langle d_1 \rangle \cdots \langle d_k \rangle \cdot \mu(d'(\varepsilon)) = \langle d \rangle\ . && (d_i \text{ victorious})
\end{aligned}
$$

where (†) holds because the outer sum on the left-hand side is known to be $\langle d' \rangle$, thus it discards the second argument, which is known not to be $\langle d' \rangle$. ∎

Observation 5.2.12 *Let $\mathcal{S}$ be extremal. Let $\zeta \in C_\Sigma$, $t \in T_\Sigma$, $p, q \in P$, $d \in D^p(M,t)$, and $d' \in D^q(M, p \cdot \zeta)$ such that $d \cdot_p d'$ is victorious on $D^q(M, t \cdot \zeta)$. Then $\langle d \cdot_p d' \rangle = [\![(\langle d \rangle \cdot e_p) \cdot \zeta]\!]_q$.*

PROOF. We derive

$$
\begin{aligned}
\langle d \cdot_p d' \rangle &= \textstyle\sum_{d'' \in D^q(M, t \cdot \zeta)} \langle d'' \rangle && (\text{victorious run}) \\
&= \langle d \cdot_p d' \rangle + \textstyle\sum_{d'' \in D^q(M, t \cdot \zeta)\colon \langle d'' \rangle \neq \langle d \cdot_p d' \rangle} \langle d'' \rangle && (\mathcal{S} \text{ idempotent}) \\
&= \langle d \cdot_p d' \rangle + \textstyle\sum_{d'' \in D^q(M, p \cdot \zeta)\colon \langle \langle d \rangle \cdot_p d'' \rangle \neq \langle d \cdot_p d' \rangle} \langle \langle d \rangle \cdot_p d'' \rangle && (\dagger) \\
&= \textstyle\sum_{d'' \in D^q(M, p \cdot \zeta)} \langle \langle d \rangle \cdot_p d'' \rangle && (\mathcal{S} \text{ idempotent}) \\
&= \textstyle\sum_{d'' \in D^q(M, (\langle d \rangle \cdot e_p) \cdot \zeta)} \langle d'' \rangle = [\![(\langle d \rangle \cdot e_p) \cdot \zeta]\!]_q\ .
\end{aligned}
$$

For (†), we use the same kind of reasoning as for (†) in the proof of Lm. 5.2.11. ∎

5.2.3 Twins property

We define two binary relations $\mathrm{SIB}(M)$ (for *siblings*) and $\mathrm{TWINS}(M)$ on Q as follows. Let $p, q \in Q$. Then

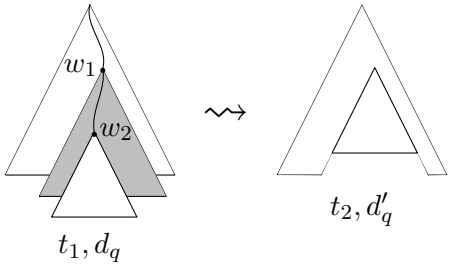


Figure 5.3: Cutting out the slice starting at w_1 and ending at w_2.

- $(p, q) \in \text{SIB}(M)$ iff there is a tree $t \in T_\Sigma$ such that $[\![t]\!]_p \neq 0$ and $[\![t]\!]_q \neq 0$.
- $(p, q) \in \text{TWINS}(M)$ iff for every context $\zeta \in C_\Sigma$ we have that $[\![e_p \cdot \zeta]\!]_p \neq 0$ and $[\![e_q \cdot \zeta]\!]_q \neq 0$ implies $[\![e_p \cdot \zeta]\!]_p = [\![e_q \cdot \zeta]\!]_q$.

The WTA M has the *twins property* if $\text{SIB}(M) \subseteq \text{TWINS}(M)$.

Example 5.2.13 (Ex. 2.4.1 contd.) We show that M has the twins property. For this, let $(p, q) \in \text{SIB}(M)$. Then there is a $t \in T_\Gamma$ such that $[\![t]\!]_p \neq 0$ and $[\![t]\!]_q \neq 0$. Moreover, let $\zeta \in C_\Gamma$ such that $[\![e_p \cdot \zeta]\!]_p \neq 0$ and $[\![e_q \cdot \zeta]\!]_q \neq 0$. We show that $[\![e_p \cdot \zeta]\!]_p = [\![e_q \cdot \zeta]\!]_q$. If $p = q$, this is trivial. For reasons of symmetry, it suffices to consider the case that $p = q_1$ and $q = q_0$. Since $[\![e_{q_1} \cdot \zeta]\!]_{q_1} \neq 0$ and $[\![t]\!]_{q_1} \neq 0$, we conclude that $\zeta = z$ and $t = \alpha$. Thus we obtain $[\![e_{q_1} \cdot z]\!]_{q_1} = 1 = [\![e_{q_0} \cdot z]\!]_{q_0}$. □

The matter of deciding the twins property is the subject of Sec. 5.4. In short: it is known that the twins property is decidable for cycle-unambiguous WTA over commutative, zero-sum-free, zero-divisor-free semirings and for WTA over extremal semifields.

Next, we show a fundamental property that follows from the twins property when $\mathcal{S}$ is commutative and extremal. Before we show the result in detail, we begin with a simple corollary that summarizes the result. Here and in the following, we use $S \cdot U$ to denote $\{s \cdot u \mid s \in S, u \in U\}$ for every set $U \subseteq S^Q$.

Corollary 5.2.14 *Let $\mathcal{S}$ be commutative and extremal, and let M have the twins property. Then there is a finite set $U \subseteq S^Q$ with $[\![T_\Sigma]\!] \subseteq S \cdot U$.*

PROOF. Direct consequence of Lm. 5.2.15. ∎

First, we sketch the proof idea; we develop the formal infrastructure and the corresponding lemma afterwards. Let $t_1 \in T_\Sigma$. Since $\mathcal{S}$ is extremal, for every $q \in Q$ there is a victorious run d_q on $D^q(M, t_1)$. If t_1 is sufficiently "large", then we find

positions w_1 and w_2 such that w_1 is strictly above w_2 and $\pi_Q(d_q|_{w_1}) = \pi_Q(d_q|_{w_2})$ for every $q \in Q$. Provided that we have chosen the family $(d_q \mid q \in Q)$ of runs in a suitable manner, the twins property guarantees that each run in this family assigns the same weight, say s_1, to the "slice" of t_1 starting at position w_1 and ending at position w_2 (depicted as the shaded area in Fig. 5.3). We can remove this slice from t_1, obtaining the smaller tree t_2 and family of runs $(d'_q \mid q \in Q)$ on t_2 with $\langle d_q \rangle = s_1 \cdot \langle d'_q \rangle$. This procedure can be iterated a finite number of times, yielding the trees $t_1, \ldots, t_n$ and weights $s_1, \ldots, s_{n-1}$, where t_n is in a finite set of "small" trees (giving rise to a finite set U of vectors).

Now we formalize this idea. For this, let $t \in T_\Sigma$ and $Q' \subseteq Q$. A *Q'-run family $\mathfrak{d}$ for t* is a family $(d_q \mid q \in Q')$ with $d_q \in D(M,t)$ and $\langle d_q \rangle \neq 0$ for every $q \in Q'$. Let $\mathfrak{d} = (d_q \mid q \in Q')$ be a Q'-run family for t. We define $[\![\mathfrak{d}]\!] \in S^Q$ by

$$[\![\mathfrak{d}]\!]_q = \begin{cases} \langle d_q \rangle & \text{if } q \in Q', \\ 0 & \text{otherwise.} \end{cases}$$

For every $w \in \mathrm{pos}(t)$, we define the Q'-run family $\mathfrak{d}|_w$ for $t|_w$ to be $(d_q|_w \mid q \in Q')$. We define $\pi'_Q(\mathfrak{d}) \in Q^{Q'}$ by letting $\pi'_Q(\mathfrak{d})_q = \pi_Q(d_q)$ for every $q \in Q'$; and we will omit the prime from π'_Q. We define three properties of $\mathfrak{d}$:

- It is *victorious* if d_q is victorious on $D^{\pi_Q(d_q)}(M,t)$ for every $q \in Q'$.
- It is *root* if $\pi_Q(\mathfrak{d})_q = q$ for every $q \in Q'$.
- It is *admissible* if it is root and for every $w_1, w_2 \in \mathrm{pos}(t)$ with w_1 strictly above w_2 and $\pi_Q(\mathfrak{d}|_{w_1}) = \pi_Q(\mathfrak{d}|_{w_2})$, we have that $\mathfrak{d}|_{w_1}$ is victorious.

For every $T \subseteq T_\Sigma$, we let

$$\mathcal{D}(T) = \{\mathfrak{d} \mid \exists Q' \subseteq Q, t \in T\colon \mathfrak{d} \text{ is an admissible } Q'\text{-run family for } t\} .$$

We define the *state number of* $\mathfrak{d}$ by $c(\mathfrak{d}) = |\{\pi_Q(\mathfrak{d}|_w) \mid w \in \mathrm{pos}(t)\}|$ and the *state number of* M by $c(M) = \max\{c(\mathfrak{d}) \mid \mathfrak{d} \in \mathcal{D}(T_\Sigma)\}$. We note that $c(M) \leq |Q|^{|Q|}$. The following lemma corresponds to a part of the proof of [105, Thm. 5].

Lemma 5.2.15 *Let $\mathcal{S}$ be commutative and extremal, and let M have the twins property. Then $[\![T_\Sigma]\!] \subseteq [\![\mathcal{D}(T_\Sigma)]\!] \subseteq S \cdot [\![\mathcal{D}(\{t \mid t \in T_\Sigma, \mathrm{ht}(t) \leq c(M)\})]\!]$.*

PROOF. We begin with the first inclusion. To this end, let $t \in T_\Sigma$. By Lm. 5.2.11, there is a recursively victorious run $d \in D^q(M,t)$ for every $q \in Q$. We let $Q' = \{q \mid [\![t]\!]_q \neq 0\}$, and we construct the family $(d_q \mid q \in Q')$ by letting $d_q \in D^q(M,t)$ be some

recursively victorious run. Then $(d_q \mid q \in Q')$ is an admissible Q'-run family for t, and $[\![t]\!] = [\![\mathfrak{d}]\!]$.

Now we show the second inclusion by contradiction. To this end, we let $T = \{t \mid t \in T_\Sigma, \mathrm{ht}(t) \leq c(M)\}$ and

$$C = \{(Q', t, \mathfrak{d}, |\mathrm{pos}(t)|) \mid t \in T_\Sigma, Q' \subseteq Q, \mathfrak{d} \text{ is an admissible } Q'\text{-run family for } t, \\ [\![\mathfrak{d}]\!] \in [\![\mathcal{D}(T_\Sigma)]\!], [\![\mathfrak{d}]\!] \notin S \cdot [\![\mathcal{D}(T)]\!]\} \ .$$

Let $(Q', t, \mathfrak{d}, m) \in C$ such that m is minimal. Then $\mathrm{ht}(t) > c(M)$ and $Q' \neq \emptyset$, because otherwise $\mathfrak{d} \in \mathcal{D}(T)$. We let $\mathfrak{d} = (d_q \mid q \in Q')$, and we fix some $q_0 \in Q'$ for later use.

We let

$$B = \{(w_1, w_2) \mid w_1, w_2 \in \mathrm{pos}(t), w_1 \text{ strictly above } w_2, \pi_Q(\mathfrak{d}|_{w_1}) = \pi_Q(\mathfrak{d}|_{w_2})\} \ .$$

Since $\mathrm{ht}(t) > c(\mathfrak{d})$, and by the pigeonhole principle, the set B is not empty – in other words, there is a $w_2 \in \mathrm{pos}(t)$ of sufficient length such that a prefix w_1 of w_2 exists such that w_1 and w_2 have the same image under $w \mapsto \pi_Q(\mathfrak{d}|_w)$.

Let $(w_1, w_2) \in B$ such that w_1 has minimal length. We construct the tree $t' = t[t|_{w_2}]_{w_1}$, the run family $\mathfrak{d}' = (d_q[d_q|_{w_2}]_{w_1} \mid q \in Q')$, the state $q_0' = \pi_Q(\mathfrak{d}|_{w_2})_{q_0}$, and the semiring element $s = [\![t[e_{q_0'}]_{w_2}|_{w_1}]\!]_{q_0'}$. We claim that (i) $\mathfrak{d}'$ is admissible and (ii) $[\![\mathfrak{d}]\!] = s \cdot [\![\mathfrak{d}']\!]$. Then either $[\![\mathfrak{d}']\!] \notin S \cdot [\![\mathcal{D}(T)]\!]$. But then, by (i), $(Q', t', \mathfrak{d}', |\mathrm{pos}(t')|) \in C$. Since $|\mathrm{pos}(t')| < |\mathrm{pos}(t)|$, this contradicts our assumption that m be minimal. Or $[\![\mathfrak{d}']\!] \in S \cdot [\![\mathcal{D}(T)]\!]$, but then so is $[\![\mathfrak{d}]\!]$, due to (ii), which contradicts our assumption that $(Q', t, \mathfrak{d}, m) \in C$.

It remains to show Statements (i) and (ii). For (i), let $w_1', w_2' \in \mathrm{pos}(t')$ such that w_1 is above w_2 and $\pi_Q(\mathfrak{d}'|_{w_1'}) = \pi_Q(\mathfrak{d}'|_{w_2'})$. We distinguish two cases, illustrated in Fig. 5.4. Either (a) there are $v_1, v_2 \in \mathbb{N}^*$ such that $w_1' = w_1 \cdot v_1$ and $w_2' = w_1 \cdot v_2$. Then $\mathfrak{d}'|_{w_1'} = \mathfrak{d}|_{w_2 v_1}$ and $\mathfrak{d}'|_{w_2'} = \mathfrak{d}|_{w_2 v_2}$. Or (b) $\mathfrak{d}'|_{w_1'} = \mathfrak{d}|_{w_1}$ and $\mathfrak{d}'|_{w_2'} = \mathfrak{d}|_{w_2}$. Since $\mathfrak{d}$ is admissible, we can derive that $\mathfrak{d}'|_{w_1'}$ is victorious in both cases.

For (ii), we let $q \in Q'$, $q' = \pi_Q(\mathfrak{d}|_{w_2})_q$, and $\zeta = t[z]_{w_2}|_{w_1}$. Then

$$\begin{aligned}
[\![\mathfrak{d}]\!]_q &= \langle d_q \rangle = \langle \langle d_q|_{w_1} \rangle \cdot d_q[z]_{w_1} \rangle \\
&= \langle [\![(\langle d_q|_{w_2} \rangle \cdot e_{q'}) \cdot \zeta]\!]_{q'} \cdot d_q[z]_{w_1} \rangle && \text{(Obs. 5.2.12)} \\
&= \langle d_q|_{w_2} \rangle \cdot [\![e_{q'} \cdot \zeta]\!]_{q'} \cdot \langle 1 \cdot d_q[z]_{w_1} \rangle && \text{(commutativity)} \\
&= \langle d_q|_{w_2} \rangle \cdot [\![e_{q_0'} \cdot \zeta]\!]_{q_0'} \cdot \langle 1 \cdot d_q[z]_{w_1} \rangle && (\dagger) \\
&= s \cdot \langle \langle d_q|_{w_2} \rangle \cdot d_q[z]_{w_1} \rangle = s \cdot [\![\mathfrak{d}']\!]_q \ . && \text{(commutativity)}
\end{aligned}$$

We show (†). First, we show that $(q', q_0') \in \textsc{Sib}(M)$. By definition we have $\langle d_{q_0} \rangle \neq 0$ and $\langle d_q \rangle \neq 0$. Hence, also $\langle d_{q_0}|_{w_2} \rangle \neq 0$ and $\langle d_q|_{w_2} \rangle \neq 0$. Since $\mathcal{S}$ is extremal, it is

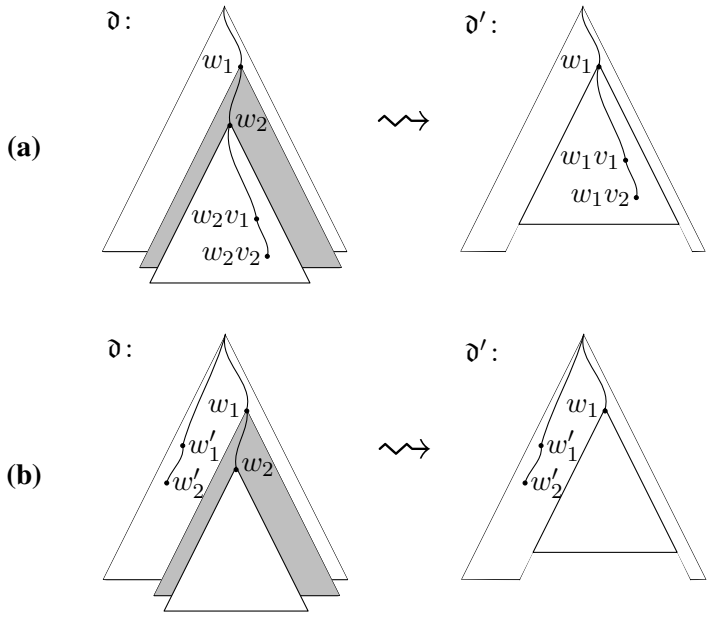


Figure 5.4: Two cases in the proof of Lm. 5.2.15.

also zero-sum free, and we obtain that $[\![t|_{w_2}]\!]_{q_0'} \neq 0$ and $[\![t|_{w_2}]\!]_{q'} \neq 0$. Hence, $(q', q_0') \in \textsc{Sib}(M)$. By a similar reasoning, we have $\langle d_{q_0}[z]_{w_2}|_{w_1}\rangle \neq 0$ and $\langle d_q[z]_{w_2}|_{w_1}\rangle \neq 0$, and thus $[\![e_{q_0'} \cdot \zeta]\!]_{q_0'} \neq 0$ and $[\![e_{q'} \cdot \zeta]\!]_{q'} \neq 0$. By the twins property, $(q', q_0') \in \textsc{Twins}(M)$, and $[\![e_{q_0'} \cdot \zeta]\!]_{q_0'} = [\![e_{q'} \cdot \zeta]\!]_{q'}$. ∎

We note that the cutting process in general destroys the recursively victorious property that we established for the first inclusion of the lemma. In other words, $[\![\mathfrak{d}]\!] = [\![t]\!]$ does not imply $[\![\mathfrak{d}']\!] = [\![t']\!]$. This is the reason why we cannot state our result in terms of $S \cdot [\![T]\!]$, and why we need the concept of admissible run families.

5.3 Determinization of classical WTA

We now apply the factorization approach [105, Sec. 3.3] to the tree case. We keep the notation concise by restricting our attention to classical WTA; we will consider arbitrary (i.e., not necessarily classical) WTA in Sec. 5.5. For the remainder of this section, let $M = (Q, \mu, \nu)$ be a classical WTA over Γ and $\mathcal{S}$.

Let (f, g) be a factorization of dimension Q. The *determinization* $det((f, g), M)$ *of* M *by* (f, g) is the triple (Q', μ', ν') where

- Q' is the smallest set $P \subseteq S^Q$ such that $\tilde{0} \in P$ and, for every $k \in \mathbb{N}$, $\sigma \in \Gamma^{(k)}$, and $u_1, \ldots, u_k \in P$, if $[\![\sigma(u_1, \ldots, u_k)]\!] \neq \tilde{0}$, then $f[\![\sigma(u_1, \ldots, u_k)]\!] \in P$.
- $\mu' \colon \bigcup_k (Q')^k \times \Gamma^{(k)} \times Q' \to S$ with

$$\mu'(u_1 \cdots u_k, \sigma, u) = \begin{cases} g[\![\sigma(u_1, \ldots, u_k)]\!] & \text{if } [\![\sigma(u_1, \ldots, u_k)]\!] \neq \tilde{0} \text{ and} \\ & \quad u = f[\![\sigma(u_1, \ldots, u_k)]\!] \,, \\ 0 & \text{otherwise} \,, \end{cases}$$

- $\nu' \colon Q' \to S$ with $\nu'_u = \sum_{q \in Q} u_q \cdot \nu_q$.

We note that Q' is uniquely determined because it is chosen from a set which is closed under intersection.

In the following, let $M' = det((f, g), M)$ and $M' = (Q', \mu', \nu')$.

Observation 5.3.1 *The triple M' is a classical WTA over Γ and $\mathcal{S}$ iff Q' is finite. If M' is a WTA, then it is bu-det.*

The following observation, which can be proved using Thm. 2.1.1, shows a stratification of Q'; this basically gives an algorithm for computing Q' (in case it is finite).

Observation 5.3.2 *Let $(Q'_i \mid i \in \mathbb{N})$ be the family with*

$$\begin{aligned} Q'_0 &= \emptyset \,, \\ Q'_{i+1} &= \{\tilde{0}\} \cup \{f[\![\sigma(u_1, \ldots, u_k)]\!] \mid k \in \mathbb{N}, \sigma \in \Gamma^{(k)}, u_1, \ldots, u_k \in Q'_i, \\ &\qquad\qquad [\![\sigma(u_1, \ldots, u_k)]\!] \neq \tilde{0}\} \,. \end{aligned}$$

Then $Q' = \bigcup_{i \in \mathbb{N}} Q'_i$; moreover, Q' is finite iff there is an $n \in \mathbb{N}$ with $Q' = Q'_n$.

Example 5.3.3 (Ex. 2.4.1 contd.) We compute $det((f, g), M) = (Q', \mu', \nu')$ using the maximal factorization (f, g) given for the Viterbi semiring in Ex. 5.2.3. First, we compute Q' according to Obs. 5.3.2. We write the elements of S^Q as column vectors, where the first row is the q_1-component; and we use the following abbreviations:

$$u_1 = \begin{pmatrix} 1 \\ 0.2 \end{pmatrix} \qquad \text{and} \qquad u_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \,.$$

Then $Q'_0 = \emptyset$ and

$$\begin{aligned} Q'_1 &= \{\tilde{0}\} \cup \{f[\![\alpha]\!]\} = \{\tilde{0}, f(u_1)\} = \{\tilde{0}, u_1\} \,, \\ Q'_2 &= Q'_1 \cup \{f[\![\sigma(u_1, u_1)]\!]\} = Q'_1 \cup \{f(0.1 \cdot u_2)\} = \{\tilde{0}, u_1, u_2\} \,, \\ Q'_3 &= Q'_2 \cup \{f[\![\sigma(u_1, u_2)]\!]\} = Q'_2 \cup \{f(0.5 \cdot u_2)\} = \{\tilde{0}, u_1, u_2\} \,, \qquad (\star) \end{aligned}$$

where we note for ($\star$) that $[\![\sigma(u_2, u_1)]\!] = \tilde{0} = [\![\sigma(u_2, u_2)]\!]$. Clearly, we have $Q'_i = Q'_2$ for $i \geq 2$. Hence, $Q' = Q_2$. Figure 5.2 shows μ'. In particular, we can read off from the above calculation that

$$\mu'(u_1u_1, \sigma, u_2) = g[\![\sigma(u_1, u_1)]\!] = g(0.1 \cdot u_2) = 0.1 \,,$$
$$\mu'(u_1u_2, \sigma, u_2) = g[\![\sigma(u_1, u_2)]\!] = g(0.5 \cdot u_2) = 0.5 \,,$$

and that μ' maps every remaining transition to 0. Finally, $\nu'_{\tilde{0}} = 0$, $\nu'_{u_1} = 0.2$, and $\nu'_{u_2} = 1$. □

The following theorem summarizes the behavior of $det((f, g), M)$. We will prove the individual statements of the theorem below.

Theorem 5.3.4 *Let $M = (Q, \mu, \nu)$ be a classical WTA over Γ and $\mathcal{S}$, and let (f, g) be the trivial or a maximal factorization. If (f, g) is not the trivial factorization, let $\mathcal{S}$ be commutative. Moreover, let one of the following conditions hold:*

- *M is acyclic,*
- *$\mathcal{S}$ is locally finite,*
- *(f, g) is maximal and M is bu-det, or*
- *(f, g) is maximal, M has the twins property, and $\mathcal{S}$ is extremal.*

Then $det((f, g), M)$ is a bu-det classical WTA over Γ and $\mathcal{S}$, and it is equivalent to M. Moreover, if (f, g) is maximal, then, regarding the number of states, $det((f, g), M)$ is minimal among all WTA which are obtained by factorization.

PROOF. Let $M' = det((f, g), M)$ and $M' = (Q', \mu', \nu')$. By Obs. 5.3.1, M' is a WTA iff Q' is finite. If M' is a WTA, then, by the same observation, it is bu-det, and M and M' are equivalent, as shown in Thm. 5.3.7. The statement about the number of states is shown in Thm. 5.3.8.

Finally, the set Q' is finite if

- M is acyclic – by Lm. 5.3.10 –,
- (f, g) is the trivial factorization and the semiring $\mathcal{S}$ is locally finite – by Lm. 4.7 of [15],
- (f, g) is maximal and $\mathcal{S}$ is commutative and locally finite – which follows from the previous item and Thm. 5.3.8 –,

- (f, g) is maximal, $\mathcal{S}$ is commutative, and M is bu-det – by Lm. 5.3.11 –,
- (f, g) is maximal, M has the twins property, and $\mathcal{S}$ is commutative and extremal – by Cor. 5.3.12. ∎

The reader is invited to compare Thm. 5.3.4 to the overview given in Tab. 5.1.

The theorem lists four conditions that guarantee that $det((f, g), M)$ be a WTA. With the first condition we provide a formal verification of [134]. The second condition is adapted from [15]. The third condition ensures that we can determinize a WTA that is already bu-det. The fourth condition is adapted from [105, Thm. 5]. We note that the third condition is mainly of theoretical interest. In fact, since testing bottom-up determinism can be done in linear time, we might precede the determinization procedure with such a test and, if the WTA is already bu-det, refrain from determinization altogether.

One might be led to believe that the trivial factorization on the one hand and maximal factorizations on the other hand represent two ends of a spectrum. In view of this, it may seem curious that we require the factorization to be trivial or maximal. However, as the following example shows, if this requirement is not satisfied, then $det((f, g), M)$ can be infinite, even if the semiring is locally finite and M is bu-det.

Example 5.3.5 Let $\Gamma = \{\gamma^{(1)}, \alpha^{(0)}\}$, $\mathcal{S} = (\mathbb{R}^{\geq 0} \cup \{\infty, -\infty\}, \min, \max, \infty, -\infty)$, and M the WTA over Γ and $\mathcal{S}$ given by (final weights do not matter)

$$(\varepsilon, \alpha, q) \mapsto 1 \qquad \text{and} \qquad (q, \gamma, q) \mapsto 0 \,.$$

Then $[\![\gamma^n(\alpha)]\!] = 1$ for every $n \in \mathbb{N}$. If (f, g) is the trivial factorization, then the determinization $det((f, g), M)$ is given by (again disregarding final weights)

$$(\varepsilon, \alpha, 1) \mapsto 1 \qquad \text{and} \qquad (1, \gamma, 1) \mapsto 1 \,.$$

Now we let (f, g) be the factorization with

$$g(u) = \min\{u_q \mid q \in Q\} \,, \qquad f(u)_q = \begin{cases} 0.9 \cdot u_q & \text{if } u_q = g(u), \\ u_q & \text{otherwise,} \end{cases}$$

where $0.9 \cdot u_q$ is the usual product in the reals. For instance, we may calculate

$$g\begin{pmatrix} 2 \\ 2.5 \end{pmatrix} \cdot f\begin{pmatrix} 2 \\ 2.5 \end{pmatrix} = 2 \cdot \begin{pmatrix} 1.8 \\ 2.5 \end{pmatrix} = \begin{pmatrix} \max(2, 1.8) \\ \max(2, 2.5) \end{pmatrix} = \begin{pmatrix} 2 \\ 2.5 \end{pmatrix} \,,$$

where $\cdot$ is the scalar product in our semiring. This example already shows that (f, g) is not trivial. It is not maximal either:

$$f(2 \cdot \begin{pmatrix} 2 \\ 2.5 \end{pmatrix}) = f \begin{pmatrix} 2 \\ 2.5 \end{pmatrix} = \begin{pmatrix} 1.8 \\ 2.5 \end{pmatrix} \neq \begin{pmatrix} 2.7 \\ 2.7 \end{pmatrix} = f \begin{pmatrix} 3 \\ 3 \end{pmatrix} = f(3 \cdot \begin{pmatrix} 2 \\ 2.5 \end{pmatrix}) .$$

We compute $det((f, g), M)$ again, this time with the new factorization:

$$(\varepsilon, \alpha, 0.9) \mapsto 1 , \quad (0.9, \gamma, 0.81) \mapsto 0.9 , \quad (0.81, \gamma, 0.729) \mapsto 0.81 , \quad \ldots \qquad \square$$

As the following example shows, however, there are cases where a maximal factorization does not exist, the trivial factorization leads to an infinite result, and another factorization does the trick. These cases are obviously not covered by our theorem.

Example 5.3.6 Let $\mathcal{S}$ be the semiring of Ex. 5.2.5. We define the factorization (f, g) as follows: $g(u)$ is the greatest common divisor (gcd) of the components of u if this number is in S, and otherwise $g(u)$ is the gcd divided by the largest prime factor it contains, e.g.,

$$g(\begin{pmatrix} 2^3 \\ 2^4 \end{pmatrix}) = \frac{2^3}{2} = 2^2 .$$

Finally, we let $f(u) = \frac{u}{g(u)}$.

Let $\Gamma = \{\gamma^{(1)}, \alpha^{(0)}\}$ and $M = (Q, \mu, \nu)$ be the WTA over Γ and $\mathcal{S}$ where $Q = \{q, p\}$, ν is immaterial to our concerns, and μ is given by

$$\begin{array}{lll} (\varepsilon, \alpha, q) \mapsto 2 \cdot 3 , & (p, \gamma, q) \mapsto 2 \cdot 3 , & (q, \gamma, q) \mapsto \infty , \\ (\varepsilon, \alpha, p) \mapsto 5 \cdot 7 , & (q, \gamma, p) \mapsto 5 \cdot 7 , & (p, \gamma, p) \mapsto \infty . \end{array}$$

We denote the elements of S^Q as vectors, with the q-component shown first. Let

$$u_1 = \begin{pmatrix} 2 \cdot 3 \\ 5 \cdot 7 \end{pmatrix} , \qquad u_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} .$$

It can be shown that

$$[\![\gamma^{2n}(\alpha)]\!] = (2 \cdot 3 \cdot 5 \cdot 7)^n \cdot u_1 , \qquad [\![\gamma^{2n+1}(\alpha)]\!] = (2 \cdot 3 \cdot 5 \cdot 7)^{n+1} \cdot u_2 .$$

Hence, determinization using the trivial factorization yields an infinite result. On the other hand, $det((f, g), M) = (Q', \mu', \nu')$ where $Q' = \{u_1, u_2\}$, ν' is again immaterial, and μ':

$$(\varepsilon, \alpha, u_1) \mapsto 1 , \qquad (u_1, \gamma, u_2) \mapsto 2 \cdot 3 \cdot 5 \cdot 7 , \qquad (u_2, \gamma, u_1) \mapsto 1 ,$$

and every other transition is mapped to ∞. $\square$

The following theorem is our correctness result; it corresponds to Thm. 1 of [105].

Theorem 5.3.7 *Let (f, g) be a factorization. If (f, g) is not the trivial factorization, then let $\mathcal{S}$ be commutative. If M' is a WTA, then $[\![M]\!] = [\![M']\!]$.*

PROOF. Let M' be a WTA, i.e., let Q' be finite. We abbreviate $[\![.]\!]_M$ by $[\![.]\!]$ and $\langle . \rangle_{\mu'}$ by $\langle . \rangle$. By Obs. 5.3.1, M' is bu-det.

We show the following statement by induction on t: for every $t \in T_\Gamma$ there are $u \in Q'$ and $d \in D^u(M', t)$ such that (i) $\langle d \rangle \cdot u = [\![t]\!]$ and (ii) $\langle d' \rangle \neq 0$ implies $d' = d$ for every $d' \in D(M', t)$. Note that, if (f, g) is the trivial factorization, then $\langle d \rangle \in \{0, 1\}$ for every $t \in T_\Gamma$, $u \in Q'$, and $d \in D^u(M', t)$.

Let $t = \sigma(t_1, \ldots, t_k)$. By the induction hypothesis, there are $u_1, \ldots, u_k \in Q'$ and $d_1, \ldots, d_k$ with $d_j \in D^{u_j}(M, t_j)$ such that (i) $\langle d_j \rangle \cdot u_j = [\![t_j]\!]$, and (ii) $\langle d'_j \rangle \neq 0$ implies $d'_j = d_j$. We set $u' = [\![\sigma(u_1, \ldots, u_k)]\!]$. We derive ($\star$):

$$\begin{aligned}
[\![t]\!] &= [\![\sigma([\![t_1]\!], \ldots, [\![t_k]\!])]\!] && \text{(Obs. 2.4.4)} \\
&= [\![\sigma(\langle d_1 \rangle \cdot u_1, \ldots, \langle d_k \rangle \cdot u_k)]\!] && \text{(induction hypothesis)} \\
&= \langle d_1 \rangle \cdots \langle d_k \rangle \cdot u' \,. && \text{(Obs. 5.2.6)}
\end{aligned}$$

Now we distinguish two cases.

Case 1: Let $u' = \tilde{0}$. Then, by ($\star$), $[\![t]\!] = \tilde{0}$. We construct $\rho = (u_1 \cdots u_k, \sigma, u')$ and $d = \rho\big(d_1, \ldots, d_k\big)$. Then $\mu'(\rho) = 0$, $\langle d \rangle = 0$, and $\langle d \rangle \cdot u' = \tilde{0}$. Now let $d' \in D(M', t)$ and $\langle d' \rangle \neq 0$. Then each of the factors $\langle d'|_1 \rangle$, …, $\langle d'|_k \rangle$, and $\mu'(d'(\varepsilon))$ is non-zero. By the induction hypothesis, $d'|_j = d_j$. By definition, $\mu'(u_1 \cdots u_k, \sigma, u) = 0$ for every $u \in Q'$. Hence, $\mu'(d'(\varepsilon)) = 0$, and our assumption that $\langle d' \rangle \neq 0$ was wrong.

Case 2: Let $u' \neq \tilde{0}$. Then we construct $u = f(u')$, $\rho = (u_1 \cdots u_k, \sigma, u)$, and $d = \rho\big(d_1, \ldots, d_k\big)$, and we derive

$$\begin{aligned}
[\![t]\!] &= \langle d_1 \rangle \cdots \langle d_k \rangle \cdot u' && (\star) \\
&= \big(\langle d_1 \rangle \cdots \langle d_k \rangle \cdot g(u')\big) \cdot f(u') && ((f, g) \text{ fact.}) \\
&= \langle d \rangle \cdot u \,.
\end{aligned}$$

Let $d' \in D(M', t)$ and $\langle d' \rangle \neq 0$. Then each of the factors $\langle d'|_1 \rangle$, …, $\langle d'|_k \rangle$, and $\mu'(d'(\varepsilon))$ is non-zero. By the induction hypothesis, $d'|_j = d_j$, and by the definition of μ', $d'(\varepsilon) = d(\varepsilon)$.

Now we show that $[\![M]\!] = [\![M']\!]$. Let $t \in T_\Gamma$. Then there are $u \in Q'$ and $d \in D^u(M', t)$ such that $\langle d \rangle \cdot u = [\![t]\!]$ and $\langle d' \rangle \neq 0$ implies $d' = d$. We derive

$$[\![M']\!](t) = \langle d \rangle \cdot \nu'_u = \textstyle\sum_{q \in Q} \langle d \rangle \cdot u_q \cdot \nu_q = \sum_{q \in Q} [\![t]\!]_q \cdot \nu_q = [\![M]\!](t) \,. \qquad \blacksquare$$

The next theorem is our minimality statement; it corresponds to Thm. 3 of [105].

Theorem 5.3.8 *Let $\mathcal{S}$ be commutative, (f, g) and $(\tilde{f}, \tilde{g})$ factorizations, (f, g) maximal, $det((f, g), M) = (Q', \mu', \nu')$, and $det((\tilde{f}, \tilde{g}), M) = (\tilde{Q}, \tilde{\mu}, \tilde{\nu})$. Then $Q' \setminus \{\tilde{0}\} = f(\tilde{Q} \setminus \{\tilde{0}\})$. Consequently, $|Q'| \leq |\tilde{Q}|$ and, if $det((\tilde{f}, \tilde{g}), M)$ is a WTA, then so is $det((f, g), M)$.*

PROOF. We first consider the case that $|Q| = 1$. Then we can identify S^Q with S. Since (f, g) is maximal, we have that $f(S \setminus \{0\}) = \{f(1)\}$. By Lm. 5.2.1, $f(1) \neq 0$. Now either $Q' = \{\tilde{0}\} = \tilde{Q}$ or $Q' = \{\tilde{0}, f(1)\}$ and $\tilde{Q} \supset \{\tilde{0}\}$. In both cases, we have that $Q' \setminus \{\tilde{0}\} = f(\tilde{Q} \setminus \{\tilde{0}\})$.

Now let $|Q| > 1$. By Lm. 5.2.4, $\mathcal{S}$ is zero-divisor free. We prove $f(\tilde{Q} \setminus \{\tilde{0}\}) = Q' \setminus \{\tilde{0}\}$. We begin with "$\subseteq$". By Obs. 5.3.2, it suffices to prove the following statement by induction on i: for every $i \in \mathbb{N}$, $f(\tilde{Q}_i \setminus \{\tilde{0}\}) \subseteq Q' \setminus \{\tilde{0}\}$. To this end, let $i \in \mathbb{N}$ and $\tilde{u} \in \tilde{Q}_{i+1} \setminus \{\tilde{0}\}$. Then there are $k \in \mathbb{N}$, $\sigma \in \Gamma^{(k)}$, and $\tilde{u}_1, \ldots, \tilde{u}_k \in \tilde{Q}_i$ such that $\tilde{u} = \tilde{f}[\![\sigma(\tilde{u}_1, \ldots, \tilde{u}_k)]\!]$. By Lm. 5.2.1, $\tilde{u} \neq \tilde{0}$ and $f(\tilde{u}) \neq \tilde{0}$. We show that $f(\tilde{u}) \in Q'$ by deriving

$$\begin{aligned} f(\tilde{u}) &= f(\tilde{f}[\![\sigma(\tilde{u}_1, \ldots, \tilde{u}_k)]\!]) = f([\![\sigma(\tilde{u}_1, \ldots, \tilde{u}_k)]\!]) && \text{(Lm. 5.2.7)} \\ &= f([\![\sigma(f(\tilde{u}_1), \ldots, f(\tilde{u}_k))]\!]) && \text{(Lm. 5.2.8)} \\ &\in Q' \,. && \text{(induction hypothesis, def. of } Q') \end{aligned}$$

Now we prove "$\supseteq$". Using Obs. 5.3.2 again, it suffices to prove the following statement by induction on i: for every $i \in \mathbb{N}$, $Q'_i \setminus \{\tilde{0}\} \subseteq f(\tilde{Q} \setminus \{\tilde{0}\})$. To this end, let $i \in \mathbb{N}$ and $u' \in Q'_{i+1} \setminus \{\tilde{0}\}$. Then there are $k \in \mathbb{N}$, $\sigma \in \Gamma^{(k)}$, and $u'_1, \ldots, u'_k \in Q'_i$ such that $[\![\sigma(u'_1, \ldots, u'_k)]\!] \neq \tilde{0}$ and $u' = f[\![\sigma(u'_1, \ldots, u'_k)]\!]$. By the induction hypothesis, there are $\tilde{u}_1, \ldots, \tilde{u}_k \in \tilde{Q} \setminus \{\tilde{0}\}$ such that $u'_i = f(\tilde{u}_i)$. By Lm. 5.2.1, $u' \neq \tilde{0}$. We show that $u' \in f(\tilde{Q})$ by deriving

$$\begin{aligned} u' &= f[\![\sigma(f(\tilde{u}_1), \ldots, f(\tilde{u}_k))]\!] = f[\![\sigma(\tilde{u}_1, \ldots, \tilde{u}_k)]\!] && \text{(Lm. 5.2.8)} \\ &= f(\tilde{f}[\![\sigma(\tilde{u}_1, \ldots, \tilde{u}_k)]\!]) && \text{(Lm. 5.2.7)} \\ &\in f(\tilde{Q}) \,. \end{aligned}$$

■

The following corollary corresponds to [105, Lm. 2].

Corollary 5.3.9 *Let $\mathcal{S}$ be commutative and (f, g) maximal. Then $Q' \setminus \{\tilde{0}\} = f([\![T_\Gamma]\!] \setminus \{\tilde{0}\})$.*

PROOF. Follows from Thm. 5.3.8 when $(\tilde{f}, \tilde{g})$ is the trivial factorization. ■

In the remainder of this section, we deal with the sufficient conditions for Q' to be finite. We begin with a lemma that is useful when M is acyclic.

Lemma 5.3.10 *There is an injective mapping* $\varphi\colon Q' \setminus \{\tilde{0}\} \to \{t \mid t \in T_\Gamma, [\![t]\!] \neq \tilde{0}\}$.

PROOF. The main idea is to define φ as the supremum of an ω-chain of injective mappings. To this end, let $\tilde{T} = \{t \mid t \in T_\Gamma, [\![t]\!] \neq \tilde{0}\}$. We use Obs. 5.3.2. For every $i \in \mathbb{N}$, we let

$$T'_i = \{\sigma(u_1, \ldots, u_k) \mid \sigma \in \Gamma^{(k)}, u_\iota \in Q'_i, [\![\sigma(u_1, \ldots, u_k)]\!] \neq \tilde{0}\} \ .$$

Note that T'_i is finite. We assume that T'_i is well ordered in some way. Then we define, for every $i \in \mathbb{N}$, the mapping $\varphi_i\colon Q'_i \setminus \{\tilde{0}\} \to \tilde{T}$ as follows. If $i = 0$, there is nothing to define. If $u \in Q'_i$, then we let $\varphi_{i+1}(u) = \varphi_i(u)$. Otherwise, we proceed as follows. There is a least $\sigma(u_1, \ldots, u_k) \in T'_i$ with $u = f[\![\sigma(u_1, \ldots, u_k)]\!]$. This implies $u_\iota \neq \tilde{0}$. We define $\varphi_{i+1}(u) = \sigma(\varphi_i(u_1), \ldots, \varphi_i(u_k))$.

First, we show by induction on i that, for every $i \in \mathbb{N}$, we have

(i) $\varphi_i|_{Q'_j \setminus \{\tilde{0}\}} = \varphi_j$ for every $j \leq i$,

(ii) $i > 0$, $u \in Q'_i$, $u \notin Q'_{i-1}$, and $u \neq \tilde{0}$ implies $\mathrm{ht}(\varphi_i(u)) = i$,

(iii) φ_i is injective.

The induction base ($i = 0$) is trivial. We show the induction step ($i \to i + 1$). To this end, let $i \in \mathbb{N}$ such that the hypothesis holds.

We show (i). For this, let $j \leq i + 1$. The case $j = i + 1$ is trivial. If $j \leq i$, we derive $\varphi_{i+1}|_{Q'_j \setminus \{\tilde{0}\}} = \varphi_i|_{Q'_j \setminus \{\tilde{0}\}} = \varphi_j$.

We show (ii). Let $i + 1 > 0$, $u \in Q'_{i+1}$, $u \notin Q'_i$, and $u \neq \tilde{0}$. There is a least $\sigma(u_1, \ldots, u_k) \in T'_i$ with $u = f[\![\sigma(u_1, \ldots, u_k)]\!]$. If $i = 0$, then $k = 0$, and we obtain $\mathrm{ht}(\varphi_{i+1}(u)) = 1$. Let $i > 0$. Assume for the time being that $u_\iota \in Q'_{i-1}$ for every ι. Then $u \in Q_i$. Hence, there is a ι such that $u_\iota \notin Q'_{i-1}$. By the induction hypothesis, $\mathrm{ht}(\varphi_i(u_\iota)) = i$. Then $\mathrm{ht}(\varphi_{i+1}(u)) = i + 1$.

We show (iii). To this end, let $u, u' \in Q'_{i+1} \setminus \{\tilde{0}\}$ and $\varphi_{i+1}(u) = \varphi_{i+1}(u')$. Then $\mathrm{ht}(\varphi_{i+1}(u)) = \mathrm{ht}(\varphi_{i+1}(u'))$. By (ii), either $u, u' \in Q'_i$ or $u, u' \notin Q'_i$. In the former case, we invoke the induction hypothesis. Otherwise we proceed as follows. There are least $\sigma(u_1, \ldots, u_k), \sigma'(u'_1, \ldots, u'_{k'}) \in T'_i$ with $u = f[\![\sigma(u_1, \ldots, u_k)]\!]$ and $u' = f[\![\sigma'(u'_1, \ldots, u'_{k'})]\!]$. Since $\varphi_{i+1}(u) = \varphi_{i+1}(u')$, we have $\sigma = \sigma'$, $k = k'$, and $\varphi_i(u_\iota) = \varphi_i(u'_\iota)$. By the induction hypothesis, $u_\iota = u'_\iota$. Hence, $u = u'$.

This completes the inductive proof. Now we construct $\varphi\colon Q' \setminus \{\tilde{0}\} \to \tilde{T}$ as follows. Let $u \in Q'$. Then there is a least $i \in \mathbb{N}$ with $u \in Q'_i$. We let $\varphi(u) = \varphi_i(u)$. It remains

to show that φ is injective. For this, let $u, u' \in Q'$ and $\varphi(u) = \varphi(u')$. There are least i, i' with $u \in Q'_i$ and $u' \in Q'_{i'}$. Then $\varphi(u) = \varphi_i(u)$ and $\varphi(u') = \varphi_{i'}(u')$. Without loss of generality, we assume that $i \geq i'$. Using (i), we derive

$$\varphi_i(u) = \varphi(u) = \varphi(u') = \varphi_{i'}(u') = \varphi_i(u') \; .$$

Then (iii) yields that $u = u'$. ∎

Now we turn to the cases where (f, g) is a maximal factorization, $\mathcal{S}$ is commutative, and either M is already bu-det or M has the twins property and $\mathcal{S}$ is extremal.

Lemma 5.3.11 *Let $\mathcal{S}$ be commutative, (f, g) maximal, and M bu-det. Then Q' is finite.*

PROOF. By Cor. 5.3.9, $Q' \setminus \{\tilde{0}\} = f([\![T_\Gamma]\!] \setminus \{\tilde{0}\})$. Observation 2.4.3 yields that each vector in $[\![T_\Gamma]\!]$ has at most one nonzero component. By this fact and since (f, g) is maximal, we can derive that $|f([\![T_\Gamma]\!] \setminus \{\tilde{0}\})| \leq |Q|$. ∎

The following corollary generalizes Thm. 5 of [105] from strings to trees.

Corollary 5.3.12 *Let $\mathcal{S}$ be commutative and extremal, (f, g) maximal, and M have the twins property. Then Q' is finite.*

PROOF. By Cor. 5.2.14 there is a finite set $U \subseteq S^Q$ such that $[\![T_\Gamma]\!] \subseteq S \cdot U$. We derive

$$\begin{aligned} Q' \setminus \{\tilde{0}\} &= f([\![T_\Gamma]\!] \setminus \{\tilde{0}\}) && \text{(Cor. 5.3.9)} \\ &\subseteq f((S \cdot U) \setminus \{\tilde{0}\}) && \text{(Cor. 5.2.14)} \\ &\subseteq f(U \setminus \{\tilde{0}\}) \; . && ((f, g) \text{ maximal}) \end{aligned}$$

Since U is finite, so is Q'. ∎

5.4 Deciding the twins property

In this section, we consider two approaches to deciding the twins property. In both approaches we require that the semiring be commutative, zero-sum free, and zero-divisor free. For the first approach, we put an additional restriction on the semiring – namely, that it be an extremal semifield. For the second one, we put a restriction on the WTA – namely, that it be cycle-unambiguous. As a preparation for both approaches, we show that we can enumerate $\text{SIB}(M)$ in finite time. For the remainder of this section, let $M = (Q, \mu, \nu)$ be a classical WTA over Γ and $\mathcal{S}$.

Lemma 5.4.1 *If $\mathcal{S}$ is zero-sum free, then* $\textsc{Sib}(M) \subseteq \textsc{Sib}'(M)$*, where* $\textsc{Sib}'(M)$ *is defined like* $\textsc{Sib}(M)$*, with the additional condition that* $\mathrm{ht}(t) \leq |Q|^2$.

PROOF. By contradiction. Let

$$C = \{(p, q, t, |\mathrm{pos}(t)|) \mid p, q \in Q, t \in T_\Gamma, [\![t]\!]_p \neq 0, [\![t]\!]_q \neq 0, (p, q) \notin \textsc{Sib}'(M)\} ,$$

and let $(p, q, t, m) \in C$ such that m is minimal.

Since $[\![t]\!]_p \neq 0$ and $[\![t]\!]_q \neq 0$, there are $d_p \in D^p(M, t)$ and $d_q \in D^q(M, t)$ such that $\langle d_p \rangle \neq 0$ and $\langle d_q \rangle \neq 0$. Since $(p, q) \notin \textsc{Sib}'(M)$, we have that $\mathrm{ht}(t) > |Q|^2$. By the pigeonhole principle, there are $w_1, w_2 \in \mathrm{pos}(t)$ and $p', q' \in Q$ such that w_1 is strictly above w_2, $d_p|_{w_1}, d_p|_{w_2} \in D^{p'}(M)$, and $d_q|_{w_1}, d_q|_{w_2} \in D^{q'}(M)$. Cutting out the slice between w_1 and w_2, we construct the tree $t' = t[t|_{w_2}]_{w_1}$. Moreover, we construct the runs d'_p and d'_q accordingly, i.e., $d'_x = d_x[d_x|_{w_2}]_{w_1}$ for $x \in \{p, q\}$.

We have that $\langle d'_p \rangle \neq 0$ and $\langle d'_q \rangle \neq 0$, because otherwise $\langle d_p \rangle = 0$ or $\langle d_q \rangle = 0$. Since S is zero-sum free, we obtain that $[\![t']\!]_p \neq 0$ and $[\![t']\!]_q \neq 0$. Clearly, $(p, q, t', |\mathrm{pos}(t')|) \in C$ and $|\mathrm{pos}(t')| < |\mathrm{pos}(t)|$, which is our contradiction. ∎

5.4.1 Extremal semifields

In this section we prove the following theorem.

Theorem 5.4.2 *Let $\mathcal{S}$ be an extremal semifield. There is a procedure that takes any classical WTA M over Γ and $\mathcal{S}$ and outputs whether M has the twins property.*

PROOF. Follows from Lm. 5.4.10. ∎

We proceed as follows. First, we rephrase the problem of deciding the twins property as the problem of searching a set of vectors for "critical elements". Moreover, we indicate that applying a factorization to that set allows us to solve the search problem in finite time. Finally, we consider two algorithms that solve our problem.

Henceforth, let $\mathcal{S}$ be an extremal semifield. Then it is also zero-sum free. By Lm. 5.2.2, there is a maximal factorization (f, g).

In the definition of $\textsc{Twins}(M)$, we deal with two vectors $[\![e_p \cdot \zeta]\!]$ and $[\![e_q \cdot \zeta]\!]$ for each $\zeta \in C_\Gamma$. In the following we concatenate these vectors, which enables us to use a factorization. To this end, we construct a WTA $M \cup \bar{M}$ that runs two instances of M in parallel, as shown in Fig. 5.5. We let $\bar{M} = (\bar{Q}, \bar{\mu}, \bar{\nu})$ be the WTA obtained from M by renaming states via $q \mapsto \bar{q}$. We construct the WTA $M \cup \bar{M} = (Q \cup \bar{Q}, \mu', \nu')$ where μ' coincides with μ and $\bar{\mu}$ on the transitions of M and $\bar{M}$, respectively; it maps all other transitions to 0; and ν' coincides with ν and $\bar{\nu}$ on Q and $\bar{Q}$, respectively.

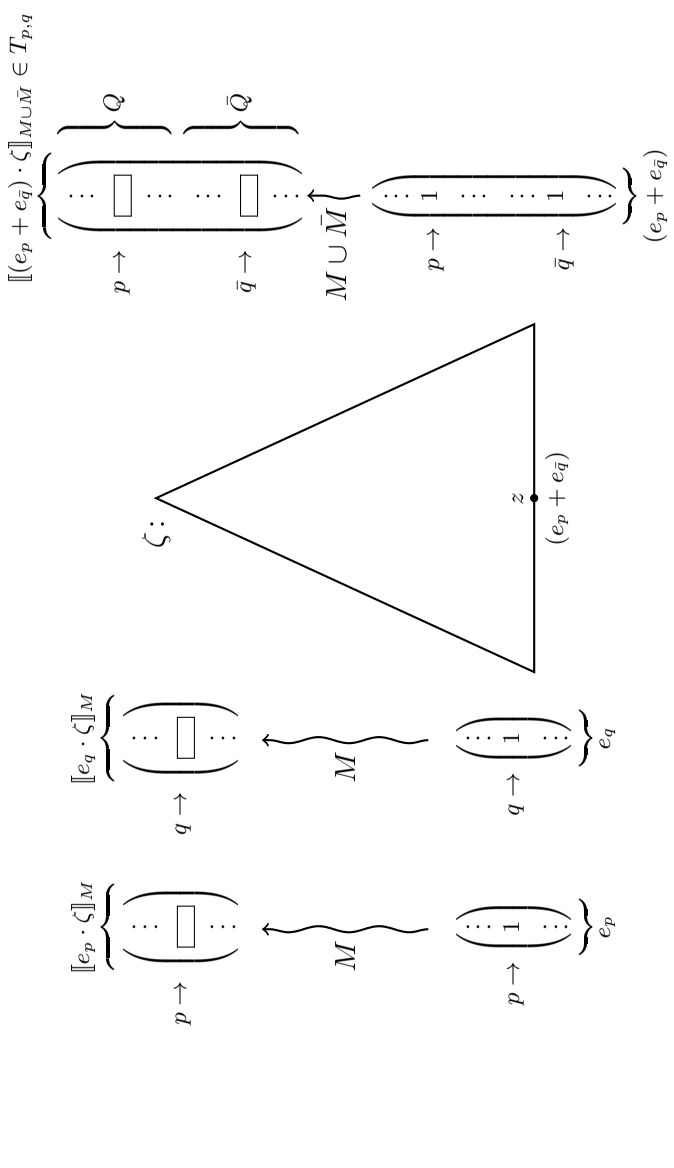


Figure 5.5: Moving from parallel execution of M (left-hand side) to the union WTA $M \cup \bar{M}$ (right-hand side).

Observation 5.4.3 *If M has the twins property, then so does $M \cup \bar{M}$.*

Using $M \cup \bar{M}$, we are now able to describe the search space of our problem. For every $p, q \in Q$ we define the set $T_{p,q} \subseteq S^{Q \cup \bar{Q}}$ by

$$T_{p,q} = [\![\{(e_p + e_{\bar{q}}) \cdot \zeta \mid \zeta \in C_\Gamma\}]\!]_{M \cup \bar{M}} ,$$

where we note that $e_p, e_{\bar{q}} \in S^{Q \cup \bar{Q}}$. Moreover, we call any vector $u \in S^{Q \cup \bar{Q}}$ *critical (for (p, q))* if $u_p \neq 0$, $u_{\bar{q}} \neq 0$, and $u_p \neq u_{\bar{q}}$. We note that $\tilde{0}$ is not a critical vector, and that a vector $u \in S^{Q \cup \bar{Q}} \setminus \{\tilde{0}\}$ is critical iff $f(u)$ is critical. With these prerequisites, we can make two easy observations.

Observation 5.4.4 *Let $p, q \in Q$. Then $(p, q) \in \text{TWINS}(M)$ iff $T_{p,q}$ does not contain any critical vector.*

Observation 5.4.5 *The following three statements are equivalent.*

(a) The WTA M has the twins property.

(b) The set $\bigcup_{(p,q) \in \text{SIB}(M)} T_{p,q}$ is devoid of critical vectors.

(c) The set $\bigcup_{(p,q) \in \text{SIB}(M)} f(T_{p,q} \setminus \{\tilde{0}\})$ is devoid of critical vectors.

We call the sets in Obs. 5.4.5(b) and (c) the *search space* and the *compressed search space*, respectively.

In the following, we show that the compressed search space is finite if M has the twins property. To this end, we will construct, for every $(p, q) \in \text{SIB}(M)$, a WTA $M_{(p,q)}$ over $\Gamma \cup \{*\}$ and $\mathcal{S}$ such that (i) there is an injective mapping from $T_{p,q}$ into $[\![T_{\Gamma \cup \{*\}}]\!]_{M_{(p,q)}}$ and (ii) if M has the twins property, then so does $M_{(p,q)}$. This will enable us later to apply Cor. 5.2.14.

Let $M \cup \bar{M} = (Q', \mu', \nu')$. We let $M_{(p,q)} = (Q'', \mu'', \nu'')$ where

- $Q'' = \{0, 1\} \times Q'$,
- $\mu''((b_1, q'_1) \cdots (b_k, q'_k), \sigma, (b, q')) = \mu'(q'_1 \cdots q'_k, \sigma, q')$ if $b = \sum_j b_j$,
- $\mu''(\varepsilon, *, (1, p)) = \mu''(\varepsilon, *, (1, \bar{q})) = 1$,
- μ'' maps all other transitions to 0,
- $\nu''(1, q') = \nu'(q')$, and ν'' maps all other states to 0.

For every $b \in \{0,1\}$, we define the mappings

$$\varphi_b \colon S^{Q \cup \bar{Q}} \to S^{\{0,1\} \times (Q \cup \bar{Q})} \qquad \text{and} \qquad \varphi'_b \colon S^{\{0,1\} \times (Q \cup \bar{Q})} \to S^{Q \cup \bar{Q}}$$

by letting $\varphi_b(u)_{(b,q')} = u_{q'}$, $\varphi_b(u)_{(b',q')} = 0$ for $b' \neq b$, and $\varphi'_b(u)_{q'} = u_{(b,q')}$. The following observation is easy to show by induction on m.

Observation 5.4.6 *For every $m \in \mathbb{N}$ and $t \in T_{\Gamma \cup \{*\}}$, if $|\mathrm{pos}(t)| \leq m$, then*

$$[\![t]\!]_{M_{(p,q)}} = \begin{cases} \varphi_0([\![t]\!]_{M \cup \bar{M}}) & \text{if } t \in T_\Gamma, \\ \varphi_1([\![t[*/(e_p + e_{\bar{q}})]]\!]_{M \cup \bar{M}}) & \text{if } * \text{ occurs exactly once in } t, \\ \tilde{0} & \text{otherwise.} \end{cases}$$

Corollary 5.4.7 *We have that $\varphi_1(T_{p,q}) \setminus \{\tilde{0}\} = (\varphi_1(S^{Q \cap \bar{Q}}) \cap [\![T_{\Gamma \cup \{*\}}]\!]_{M_{(p,q)}}) \setminus \{\tilde{0}\}$.*

PROOF. Let $u \in \varphi_1(T_{p,q})$ and $u \neq \tilde{0}$. Then there is a $\zeta \in C_\Gamma$ such that $u = \varphi_1([\![(e_p + e_{\bar{q}}) \cdot \zeta]\!]_{M \cup \bar{M}})$. By Obs. 5.4.6, we have that $[\![* \cdot \zeta]\!]_{M_{(p,q)}} = u$. Hence, $u \in (\varphi_1(S^{Q \cap \bar{Q}}) \cap [\![T_{\Gamma \cup \{*\}}]\!]_{M_{(p,q)}}) \setminus \{\tilde{0}\}$. Now let $u \in \varphi_1(S^{Q \cup \bar{Q}}) \cap [\![T_{\Gamma \cup \{*\}}]\!]_{M_{(p,q)}}$ and $u \neq \tilde{0}$. Then there is a $t \in T_{\Gamma \cup \{*\}}$ with $u = [\![t]\!]_{M_{(p,q)}}$. Since $u \in \varphi_1(S^{Q \cup \bar{Q}}) \setminus \{\tilde{0}\}$, Obs. 5.4.6 yields that $*$ occurs exactly once in t and that $u = \varphi_1([\![t[*/(e_p + e_{\bar{q}})]]\!]_{M \cup \bar{M}})$. Hence, $u \in \varphi_1(T_{p,q})$. ∎

Lemma 5.4.8 *If M has the twins property, then so does $M_{(p,q)}$.*

PROOF. Let $((b,p'),(c,q')) \in \mathrm{SIB}(M_{(p,q)})$. Then there is a $t \in T_{\Gamma \cup \{*\}}$ such that $[\![t]\!]_{(b,p')} \neq 0$ and $[\![t]\!]_{(c,q')} \neq 0$. Using Obs. 5.4.6, we obtain that either (i) $t \in T_\Gamma$ and $b = c = 0$ or (ii) $*$ occurs exactly once in t and $b = c = 1$.

We show that $(p',q') \in \mathrm{SIB}(M \cup \bar{M})$. In Case (i), this is trivial. In Case (ii), we use that $(p,q) \in \mathrm{SIB}(M)$, thus $(p,\bar{q}) \in \mathrm{SIB}(M \cup \bar{M})$, and thus there is a $t' \in T_\Gamma$ with $([\![t']\!]_{M \cup \bar{M}})_p \neq 0$ and $([\![t']\!]_{M \cup \bar{M}})_{\bar{q}} \neq 0$. Since $\mathcal{S}$ is an extremal semifield, it is zero-sum free and zero-divisor free, and we obtain that $([\![t[*/t']]\!]_{M \cup \bar{M}})_{p'} \neq 0$ and $([\![t[*/t']]\!]_{M \cup \bar{M}})_{q'} \neq 0$.

We show that $((b,p'),(c,q')) \in \mathrm{TWINS}(M_{(p,q)})$. Let $\zeta \in C_{\Gamma \cup \{*\}}$ such that $[\![e_{(b,p')} \cdot \zeta]\!]_{(b,p')} \neq 0$ and $[\![e_{(c,q')} \cdot \zeta]\!]_{(c,q')} \neq 0$. It is easy to see that then $*$ does not occur in ζ, and that $[\![e_{(b,p')} \cdot \zeta]\!]_{(b,p')} = [\![e_{p'} \cdot \zeta]\!]_{p'}$ and $[\![e_{(c,q')} \cdot \zeta]\!]_{(c,q')} = [\![e_{q'} \cdot \zeta]\!]_{q'}$. Then the statement follows because $M \cup \bar{M}$ has the twins property, by Obs. 5.4.3. ∎

Lemma 5.4.9 *Let M have the twins property. For every $(p,q) \in \mathrm{SIB}(M)$ the set $f(T_{p,q} \setminus \{\tilde{0}\})$ is finite.*

Algorithm 5.1 Decision algorithm.

Require:
$M = (Q, \mu, \nu)$ a classical WTA over Γ and $\mathcal{S}$,
$\mathcal{S}$ commutative and extremal,
(f, g) a maximal factorization of dimension $Q \cup \bar{Q}$

Ensure:
print "yes" iff M has the twins property

```
1: compute SIB(M)
2: for (p, q) ∈ SIB(M) in parallel do
3:   for u ∈ f(T_{p,q} \ {0̃}) do
4:     if u is a critical vector then
5:       print "no" and terminate
6: print "yes"
```

PROOF. Since $\mathcal{S}$ is an extremal semifield, it is also zero-sum free, and Lm. 5.2.2 yields that there is a maximal factorization (f', g') of dimension $\{0, 1\} \times (Q \cup \bar{Q})$. We construct a new factorization (f'', g'') where

$$(f''(u), g''(u)) = (\varphi_1(f(\varphi_1'(u))), g(\varphi_1'(u))) \,, \qquad \text{(if } u = \varphi_1(\varphi_1'(u)))$$
$$(f''(u), g''(u)) = (f'(u), g'(u)) \,. \qquad \text{(otherwise)}$$

It is easy to see that (f'', g'') is indeed a factorization; even a maximal one.

By Lm. 5.4.8 and Cor. 5.2.14, there is a finite set U such that

$$\begin{aligned}
\varphi_1(f(T_{p,q} \setminus \{\tilde{0}\})) &= f''(\varphi_1(T_{p,q} \setminus \{\tilde{0}\})) \\
&= f''((\varphi_1(S^{Q \cup \bar{Q}}) \cap [\![T_{\Gamma \cup \{*\}}]\!]) \setminus \{\tilde{0}\}) && \text{(Cor. 5.4.7)} \\
&\subseteq f''([\![T_{\Gamma \cup \{*\}}]\!] \setminus \{\tilde{0}\}) \subseteq f''(S \cdot U \setminus \{\tilde{0}\}) && \text{(Cor. 5.2.14)} \\
&= f''(U \setminus \{\tilde{0}\}) \,. && \text{(maximal factorization)}
\end{aligned}$$

Since U is finite and φ_1 is injective, $f(T_{p,q} \setminus \{\tilde{0}\})$ is finite as well. ∎

Now we consider two decision algorithms that are based on Lm. 5.4.9. The first one, Alg. 5.1, searches the compressed search space for critical vectors.

Lemma 5.4.10 *Algorithm 5.1 terminates, and it is correct.*

PROOF. First, the algorithm enumerates $\text{SIB}(M)$. This is possible due to Lm. 5.4.1. Second, for each $(p, q) \in \text{SIB}(M)$ in parallel, it enumerates $f(T_{p,q} \setminus \{\tilde{0}\})$, checking for critical vectors. For this step, we distinguish two cases.

If M has the twins property, then, by Lm. 5.4.9 and Obs. 5.4.5, $f(T_{p,q} \setminus \{\tilde{0}\})$ is finite and devoid of critical vectors, and the algorithm terminates with output "yes".

Otherwise, by Obs. 5.4.5, the algorithm finds a critical vector at some point and outputs "no". For this, the parallel processing (Line 2) is critical because there may be $(p, q) \in \text{SIB}(M)$ such that $f(T_{p,q} \setminus \{\tilde{0}\})$ is infinite, yet devoid of critical vectors. ∎

Algorithm 5.1 is rather straightforward, but that comes at a price. In order to enumerate the compressed search space, we can in principle enumerate C_Γ and compute $f[\![(e_p + e_{\bar{q}}) \cdot \zeta]\!]$ for each $\zeta \in C_\Gamma$. However, the weights already computed for subtrees and subcontexts of ζ are not reused in this approach. For this reason, we consider an alternative procedure – Alg. 5.2.

Algorithm 5.2 does not enumerate C_Γ explicitly; instead, it works on weight vectors, thereby avoiding redundant calculation. Roughly speaking, it employs a maximal factorization (f, g) of appropriate dimension and computes $det((f, g), M_{(p,q)})$ for every $(p, q) \in \text{SIB}(M)$. The following lemma shows that the state sets of these WTA comprise a legitimate alternative to our compressed search space. Let $u \in S^{\{0,1\}\times(Q\cup\bar{Q})}$; then u is a *critical vector* if $\varphi_1'(u)$ is a critical vector.

Lemma 5.4.11 *Let (f, g) be a maximal factorization of dimension $\{0, 1\} \times (Q \cup \bar{Q})$, and let $(Q'_{(p,q)} \mid (p, q) \in \text{SIB}(M))$ be the family such that $Q'_{(p,q)}$ is the set of states of $det((f, g), M_{(p,q)})$. Then the following are equivalent:*

(a) $\bigcup_{(p,q)\in\text{SIB}(M)} T_{p,q}$ *contains a critical vector,*

(b) $f(\varphi_1(\bigcup_{(p,q)\in\text{SIB}(M)} T_{p,q} \setminus \{\tilde{0}\}))$ *contains a critical vector,*

(c) $\varphi_1(S^{Q\cap\bar{Q}}) \cap \bigcup_{(p,q)\in\text{SIB}(M)} Q'_{(p,q)}$ *contains a critical vector.*

PROOF. The first equivalence is easy to see. For the second equivalence, we derive

$$\begin{aligned}
&f(\varphi_1(\textstyle\bigcup_{(p,q)\in\text{SIB}(M)} T_{p,q} \setminus \{\tilde{0}\})) \\
&= \textstyle\bigcup_{(p,q)\in\text{SIB}(M)} f(\varphi_1(T_{p,q} \setminus \{\tilde{0}\})) \\
&= \textstyle\bigcup_{(p,q)\in\text{SIB}(M)} f((\varphi_1^{-1}(S^{Q\cap\bar{Q}}) \cap [\![T_{\Gamma\cup\{*\}}]\!]_{M_{(p,q)}}) \setminus \{\tilde{0}\}) && \text{(Cor. 5.4.7)} \\
&= \varphi_1(S^{Q\cap\bar{Q}}) \cap \textstyle\bigcup_{(p,q)\in\text{SIB}(M)} f([\![T_{\Gamma\cup\{*\}}]\!]_{M_{(p,q)}} \setminus \{\tilde{0}\}) \\
&= \varphi_1(S^{Q\cap\bar{Q}}) \cap \textstyle\bigcup_{(p,q)\in\text{SIB}(M)} Q'_{(p,q)} \setminus \{\tilde{0}\} \,. && \text{(Cor. 5.3.9)}
\end{aligned}$$

Recall that $\tilde{0}$ is not a critical vector. ∎

Algorithm 5.2 Improved decision algorithm.

Require:
$M = (Q, \mu, \nu)$ a classical WTA over Γ and $\mathcal{S}$,
$\mathcal{S}$ commutative and extremal,
(f, g) a maximal factorization of dimension $\{0, 1\} \times (Q \cup \bar{Q})$

Ensure:
print "yes" iff M has the twins property

1: compute $\textsc{Sib}(M)$
2: **if** $\textsc{Sib}(M) = \emptyset$ **then**
3: print "yes" and terminate
4: select some $(p_0, q_0) \in \textsc{Sib}(M)$
5: compute Q'_0 (see below)
6: **for** $i = 1, 2, \ldots$ **do**
7: compute Q'_i (see below) ▷ uses (p_0, q_0) and $\textsc{Sib}(M)$
8: **if** $Q'_i = Q'_{i-1}$ **then**
9: print "yes" and terminate
10: **if** $\varphi_1(S^{Q \cup \bar{Q}}) \cap Q'_i$ contains a critical vector **then**
11: print "no" and terminate

where

$$
\begin{aligned}
Q'_0 &= \emptyset \\
Q'_{i+1} &= \{\tilde{0}\} \cup \{f[\![*]\!]_{M_{(p,q)}} \mid (p, q) \in \textsc{Sib}(M)\} \\
&\qquad \cup \{f[\![\sigma(u_1, \ldots, u_k)]\!]_{M_{(p_0,q_0)}} \mid k \in \mathbb{N}, \sigma \in \Gamma^{(k)}, u_1, \ldots, u_k \in Q'_i, \\
&\qquad\qquad [\![\sigma(u_1, \ldots, u_k)]\!]_{M_{(p_0,q_0)}} \neq \tilde{0}\} \; .
\end{aligned}
$$

Lemma 5.4.12 *Algorithm 5.2 terminates, and it is correct.*

PROOF. The case that $\mathrm{SIB}(M) = \emptyset$ is trivial. We turn to the converse case. We let $(Q'_{(p,q)} \mid (p,q) \in \mathrm{SIB}(M))$ be the family such that $Q'_{(p,q)}$ is the set of states of $det((f,g), M_{(p,q)})$. For every $(p,q) \in \mathrm{SIB}(M)$, we apply Obs. 5.3.2 to $M_{(p,q)}$ and denote the corresponding family by $(Q'_{(p,q),i} \mid i \in \mathbb{N})$. Let $(Q'_i \mid i \in \mathbb{N})$ be the family defined in Alg. 5.2. A straightforward proof by induction on i yields that

$$Q'_i = \bigcup_{(p,q)\in \mathrm{SIB}(M)} Q'_{(p,q),i} \,. \qquad (*)$$

Now we distinguish two cases. Either M has the twins property. Then, by Lm. 5.4.8, so does $M_{(p,q)}$ for every $(p,q) \in \mathrm{SIB}(M)$, and by Lm. 5.3.12, the set $Q'_{(p,q)}$ is finite, and $Q'_{(p,q),i} = Q'_{(p,q),i+1}$ for some i. By $(*)$, then, also $Q'_i = Q'_{i+1}$ for some i, and the algorithm terminates. By Lm. 5.4.11, it outputs "yes".

Or M does not have the twins property. By Obs. 5.4.5 and Lm. 5.4.11, the algorithm finds a critical vector at some point and outputs "no". ∎

We note that, as is evident from Obs. 5.4.6, at least half of the components of every vector in Q'_i is zero, so there is room for optimizing the algorithm. For instance, one can partition Q'_i into the following three blocks:

$$\{\tilde{0}\} \,, \qquad \big(\varphi_0(S^{Q\cup\bar{Q}}) \cap Q'_i\big) \setminus \{\tilde{0}\} \,, \qquad \big(\varphi_1(S^{Q\cup\bar{Q}}) \cap Q'_i\big) \setminus \{\tilde{0}\} \,.$$

Then the first block is irrelevant for the algorithm and may be omitted, and the remaining blocks can be represented by their images under φ'_0 and φ'_1, respectively.

5.4.2 Cycle-unambiguous weighted tree automata

In this section, we show that the twins property is decidable for a decidable subclass of WTA called *cycle unambiguous*. This result is inspired by a similar one for the string case found in [5, Thm. 5]. The following definition is also adapted from [5, Sec. 2.1]. A WTA $N = (Q, R, \mu, \nu)$ over Σ and $\mathcal{S}$ is called *cycle unambiguous* if for every $q \in Q$ and $\zeta \in C_\Sigma$ there is at most one $d \in D^q(N, q \cdot \zeta)$ such that $\langle d \rangle \neq 0$. For instance, the WTA of Ex. 2.4.1 is cycle unambiguous.

Lemma 5.4.13 *Let $\mathcal{S}$ be commutative, zero-sum-free, and zero-divisor-free, and let M be cycle unambiguous. Then* $\mathrm{TWINS}'(M) \subseteq \mathrm{TWINS}(M)$*, where* $\mathrm{TWINS}'(M)$ *is defined like* $\mathrm{TWINS}(M)$*, with the additional condition that* $\mathrm{ht}(\zeta) \leq 2 \cdot |Q|^2$.

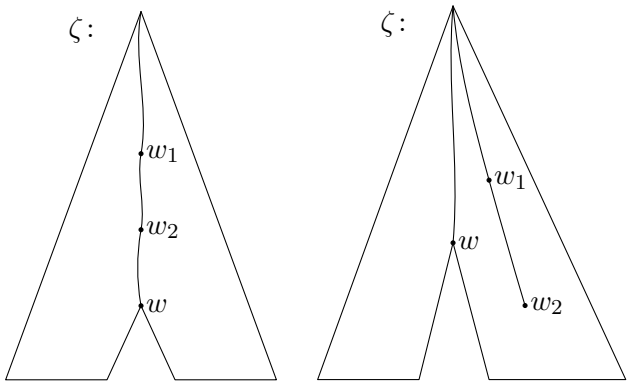


Figure 5.6: Finding w_1 and w_2; note that $\mathrm{ht}(\zeta) > 2 \cdot |Q|^2$.

PROOF. By contradiction. We let

$$C = \{(p, q, \zeta, |\mathrm{pos}(\zeta)|) \mid (p, q) \in \textsc{Twins}'(M), \zeta \in C_\Gamma, \\ [\![e_p \cdot \zeta]\!]_p \neq 0, [\![e_q \cdot \zeta]\!]_q \neq 0, [\![e_p \cdot \zeta]\!]_p \neq [\![e_q \cdot \zeta]\!]_q\} \,.$$

Let $(p, q, \zeta, m) \in C$ such that m is minimal. Since $[\![e_p \cdot \zeta]\!]_p \neq 0$ and $[\![e_p \cdot \zeta]\!]_p \neq 0$, and since M is cycle-unambiguous, there are $d_p \in D^p(M, p \cdot \zeta)$ and $d_q \in D^p(M, q \cdot \zeta)$ such that $[\![e_p \cdot \zeta]\!]_p = \langle d_p \rangle$ and $[\![e_p \cdot \zeta]\!]_q = \langle d_q \rangle$.

Let $w \in \mathrm{pos}(z)$ such that $\zeta(w) = z$. Since $(p, q) \in \textsc{Twins}'(M)$, we obtain that $\mathrm{ht}(\zeta) > 2 \cdot |Q|^2$. By the pigeonhole principle, there are $w_1, w_2 \in \mathrm{pos}(\zeta)$ (illustrated in Fig. 5.6) such that (i) w_1 is strictly above w_2, (ii) if w_1 is above w, then so is w_2, and (iii) $\pi_Q(d_q|_{w_1}) = \pi_Q(d_q|_{w_2})$ and $\pi_Q(d_p|_{w_1}) = \pi_Q(d_p|_{w_2})$. Let w_1, w_2 be such a pair of positions, and let $p' = \pi_Q(d_p|_{w_1})$, $q' = \pi_Q(d_q|_{w_2})$, $\zeta' = \zeta[z]_{w_2}|_{w_1}$, $d'_p = d_p[p']_{w_2}|_{w_1}$, $d_q[q']_{w_2}|_{w_1}$, $\zeta'' = \zeta[\zeta|_{w_2}]_{w_1}$, $d''_p = d_p[d_p|_{w_2}]_{w_1}$, and $d''_q = d_q[d_q|_{w_2}]_{w_1}$.

We note that $\langle d'_p \rangle, \langle d''_p \rangle, \langle d'_q \rangle, \langle d''_q \rangle \neq 0$, because otherwise $\langle d_p \rangle = 0$ or $\langle d_q \rangle = 0$. Since M is cycle-unambiguous, we obtain that $[\![p' \cdot \zeta']\!]_{p'} = \langle d'_p \rangle$, $[\![q' \cdot \zeta']\!]_{q'} = \langle d'_q \rangle$, $[\![p \cdot \zeta'']\!]_p = \langle d''_p \rangle$, $[\![q \cdot \zeta'']\!]_q = \langle d''_q \rangle$. We distinguish two cases.

Either $\langle d'_p \rangle = \langle d'_q \rangle$. Then $\langle d''_p \rangle \neq \langle d''_q \rangle$, because otherwise $\langle d_p \rangle = \langle d_q \rangle$. Hence, $(p, q, \zeta'', |\mathrm{pos}(\zeta'')|) \in C$. Since $|\mathrm{pos}(\zeta'')| < |\mathrm{pos}(\zeta)|$, we have a contradiction. Or $\langle d'_p \rangle \neq \langle d'_q \rangle$. Then $(p', q', \zeta', |\mathrm{pos}(\zeta')|) \in C$. Since $|\mathrm{pos}(\zeta')| < |\mathrm{pos}(\zeta)|$, we also have a contradiction in this case. ∎

Theorem 5.4.14 *Let $\mathcal{S}$ be a commutative, zero-sum-free, zero-divisor-free semiring. There is a procedure that takes any cycle-unambiguous classical WTA M over Γ and $\mathcal{S}$ and outputs whether M has the twins property.*

PROOF. By Lm. 5.4.1, we have that $\text{SIB}(M) \subseteq \text{SIB}'(M)$. By Lm. 5.4.13, we have that $\text{TWINS}'(M) \subseteq \text{TWINS}(M)$. In both cases, the converse is trivial, so we obtain that $\text{SIB}'(M) = \text{SIB}(M)$ and $\text{TWINS}'(M) = \text{TWINS}(M)$. Hence, the sets $\text{SIB}(M)$ and $\text{TWINS}(M)$ can be computed in finite time. Since these sets are finite, checking $\text{SIB}(M) \subseteq \text{TWINS}(M)$ is a trivial matter. ∎

We conclude this section by proving that the property "cycle unambiguous" is decidable. We call M *finitely cycle unambiguous (fcu)* if for every $q \in Q$ and $\zeta \in C_\Gamma$ with $\text{ht}(\zeta) \leq 3 \cdot |Q|^2$, there is at most one $d \in D^q(M, q \cdot \zeta)$ such that $\langle d \rangle \neq 0$.

Lemma 5.4.15 *If M is fcu, then it is cycle unambiguous.*

PROOF. By contradiction. To this end, let M be fcu, and let

$$C = \{(q, \zeta, d, d', |\text{pos}(\zeta)|) \mid q \in Q, \zeta \in C_\Gamma, d, d' \in D^q(M, q \cdot \zeta), \\ d \neq d', \langle d \rangle \neq 0, \langle d' \rangle \neq 0\} .$$

Let $(q, \zeta, d, d', m) \in C$ such that m is minimal. Since $d \neq d'$, there is a $w_1 \in \text{pos}(\zeta)$ with $d(w_1) \neq d'(w_1)$. Let $w_2 \in \text{pos}(\zeta)$ with $\zeta(z) = w_2$, and let $w \in \text{pos}(\zeta)$ be the longest common prefix of w_1 and w_2.

Since M is fcu, $|\text{pos}(\zeta)| > 3 \cdot |Q|^2$. By the pigeonhole principle, there are $w_3, w_4 \in \text{pos}(\zeta)$ (illustrated in Fig. 5.7) such that (i) w_3 is strictly above w_4, (ii) if w_3 is above w, w_1, or w_2, then so is w_4, respectively, and (iii) $\pi_Q(d|_{w_3}) = \pi_Q(d|_{w_4})$ and $\pi_Q(d'|_{w_3}) = \pi_Q(d'|_{w_4})$. We let $\zeta' = \zeta[\zeta|_{w_4}]_{w_3}$, $e = d[d|_{w_4}]_{w_3}$, and $e' = d'[d'|_{w_4}]_{w_3}$. We note that $\zeta' \in C_\Gamma$ and $e \neq e'$, both due to Condition (ii). Moreover, we have that $\langle e \rangle, \langle e' \rangle \neq 0$, because otherwise $\langle d \rangle = 0$ or $\langle d \rangle' = 0$. Hence, $(q, \zeta', e, e', |\text{pos}(\zeta')|) \in C$. Since $|\text{pos}(\zeta')| < m$, we have the contradiction. ∎

Corollary 5.4.16 *There is a procedure that takes any classical WTA M and outputs whether M is cycle-unambiguous.*

PROOF. Direct consequence of Lm. 5.4.15. ∎

5.5 The case of non-classical WTA

In this section, we transfer the results of the preceding two sections to arbitrary, i.e., not necessarily classical, WTA.

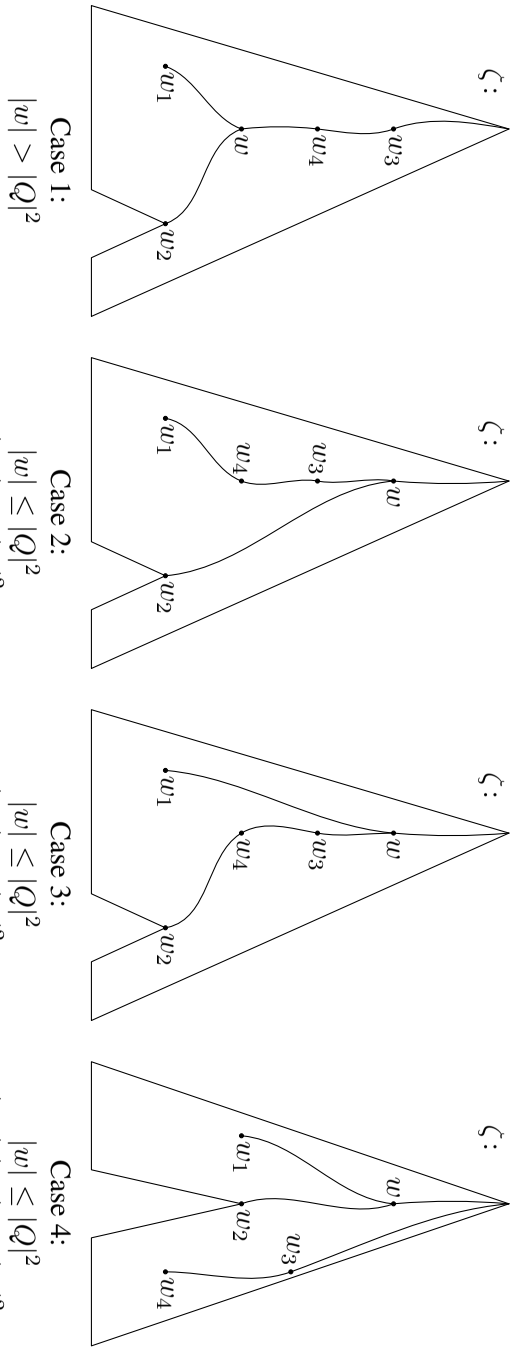


Figure 5.7: Finding w_3 and w_4; note that $\mathrm{ht}(\zeta) > 3 \cdot |Q|^2$.

To this end, let Σ be an alphabet, $\Gamma \subseteq \Sigma \times \mathbb{N}$ a ranked alphabet with $\mathrm{rk}(\sigma, k) = k$, and $f\colon T_\Gamma \to T_\Sigma$ the mapping that replaces each label by its first component. Two weighted tree languages $\varphi\colon T_\Sigma \to \mathcal{S}$ and $\varphi'\colon T_\Gamma \to \mathcal{S}$ are *related* if $\varphi(f(t)) = \varphi'(t)$ for every $t \in T_\Gamma$ and $\varphi(t) = 0$ for every $t \in T_\Sigma \setminus f(T_\Gamma)$.

Observation 5.5.1 *If φ_1 and φ' are related, and so are φ_2 and φ', then $\varphi_1 = \varphi_2$.*

Moreover, let $M = (P, R, \mu, \nu)$ be a WTA over Σ and $\mathcal{S}$ and $M' = (P', \mu', \nu')$ a classical WTA over Γ and $\mathcal{S}$. We say that M and M' are *related* if $(p_1 \cdots p_k, \sigma, p) \in R$ implies $(\sigma, k) \in \Gamma$, $P = P'$, $\nu = \nu'$, and

$$\mu'(p_1 \cdots p_k, (\sigma, k), p) = \begin{cases} \mu(p_1 \cdots p_k, \sigma, p) & \text{if } (p_1 \cdots p_k, \sigma, p) \in R\,, \\ 0 & \text{otherwise.} \end{cases}$$

Observation 5.5.2 *Let M and M' be related. Then*

(i) M is bu-det iff M' is bu-det,

(ii) M has the twins property iff M' has the twins property,

(iii) M is acyclic iff M' is acyclic,

(iv) M is cycle-unambiguous iff M' is cycle-unambiguous, and

(v) $[\![M]\!]$ and $[\![M']\!]$ are related.

We note that the definition of "related" gives rise to a natural construction turning any WTA M into a related classical WTA M', as well as the converse construction. From now on, we assume that these constructions are understood, and when we speak of "the (classical) WTA related to ...", we refer to these constructions.

Let $M = (Q, R, \mu, \nu)$ be a WTA over Σ and $\mathcal{S}$, and let (f, g) be a factorization of dimension Q. The *unranked determinization* $udet((f, g), M)$ *of* M *by* (f, g) is either a WTA over Σ and $\mathcal{S}$ or it is undefined, as follows. Let M' be the classical WTA related to M. If $det((f, g), M')$ is a WTA, then we let $det((f, g), M)$ be the WTA related to $det((f, g), M')$. Otherwise, $udet((f, g), M)$ is undefined.

Theorem 5.5.3 *Let $M = (Q, R, \mu, \nu)$ be a WTA over Σ and $\mathcal{S}$, and let (f, g) be the trivial or a maximal factorization. If (f, g) is not the trivial factorization, let $\mathcal{S}$ be commutative. Moreover, let one of the following conditions hold:*

- *M is acyclic,*
- *$\mathcal{S}$ is locally finite,*

- (f, g) *is maximal and* M *is bu-det, or*
- (f, g) *is maximal,* M *has the twins property, and* $\mathcal{S}$ *is extremal.*

Then $\mathit{udet}((f, g), M)$ *is a bu-det WTA over* Σ *and* $\mathcal{S}$*, and it is equivalent to* M*.*

PROOF. This follows from Thm. 5.3.4, Obs. 5.5.1, and Obs. 5.5.2. ∎

Theorem 5.5.4 *Let* $\mathcal{S}$ *be an extremal semifield. Then there is a procedure that takes any WTA* M *over* Σ *and* $\mathcal{S}$ *and outputs whether* M *has the twins property.*

PROOF. This follows from Thm. 5.4.2 and Obs. 5.5.2. ∎

Theorem 5.5.5 *Let* $\mathcal{S}$ *be a commutative, zero-sum-free, and zero-divisor-free semiring. Then there is a procedure that takes any cycle-unambiguous WTA* M *over* Σ *and* $\mathcal{S}$ *and outputs whether* M *has the twins property.*

PROOF. This follows from Thm. 5.4.14 and Obs. 5.5.2. ∎

Theorem 5.5.6 *There is a procedure that takes any WTA* M *and outputs whether* M *is cycle-unambiguous.*

PROOF. This follows from Cor. 5.4.16 and Obs. 5.5.2. ∎

5.6 Conclusion, discussion, and outlook

We have used the factorization approach of [138, 105] to develop a determinization construction for WTA. Theorem 5.5.3 and Tab. 5.1 summarize the requirements under which our construction solves the determinization problem. We have also shown that, roughly speaking, maximal factorizations only exist for zero-divisor-free semirings. Furthermore, we have shown that the twins property is decidable (i) for cycle-unambiguous WTA over commutative, zero-sum-free, zero-divisor-free semirings and (ii) for WTA over extremal semifields.

The present determinization result was largely obtained by generalizing [105] from strings to trees, and many of our proofs follow theirs. Likewise, the decidability result in Case (i) was obtained by generalizing [5] from strings to trees, but our proofs do not follow theirs. In particular, they provide a polynomial-time decision algorithm. In contrast, while our proofs are effective, they do not suggest efficient decision procedures. It is open whether efficient algorithms exist for WTA. We note that the transition from strings to trees made the proofs more intricate and at some points necessitated commutativity of the semiring.

As for Case (ii), the notion that the twins property can be decided by searching for critical vectors in a compressed search space is due to Kirsten [104]. We have generalized his work in the following two ways. First, we allow arbitrary extremal semifields instead of the tropical semiring. To this end, we use the notion of a maximal factorization, which is implicit in his work. Second, we consider WTA instead of WSA. This makes the proof more complex, as we have to distinguish between contexts and trees.

Kirsten's result that deciding the twins property is PSPACE-hard directly transfers to our setting, giving a lower bound on the complexity of our algorithms. In addition, he shows that the problem is PSPACE-complete by giving a PSPACE algorithm. It is open whether this result can be transferred to our setting as well. It is also an open question which algorithm, Alg. 5.1 or Alg. 5.2, performs better in practice.

As mentioned at the very top, this chapter is an extensively revised version of the papers [31] and [27]. The former work covers the determinization construction and Case (i) of the decidability problem, while the latter work is concerned with Case (ii). In this chapter, we added the case of non-classical WTA. Moreover, we reduced the question whether the compressed search space is finite to the question whether the set of all weight vectors of some WTA is finite (cf. the proof of Lm. 5.4.9). The original contribution, on the other hand, contains a direct proof [27, Lm. 3.3]. Likewise, we reused the determinization construction for the decision procedure in Alg. 5.2, in contrast to the original contribution [27, Alg. 2]. In both instances, the original contribution duplicates proof work considerably.

We already mentioned some items for further research in passing, namely efficient decision procedures for Case (i), PSPACE membership of the problem in Case (ii), and which decision algorithm performs better in practice. We name two more items, which are related to sufficient conditions for the determinization to be finite. First, one could consider general requirements that also cover cases like Ex. 5.3.6, where neither the trivial nor any maximal factorization is viable. Second, it might be desirable to determinize a WTA that is not acyclic over a semiring that is not extremal, say, the nonnegative reals.

6 Conclusion

Chapter 1 proposed a preliminary version of an algebraic framework for specifying decoders, promising the achievements in Tab. 1.1. Let us scrutinize this promise:

(a) Sections 1.4 and 1.5.1 showed how to specify syntax-based decoders in the preliminary framework. Our decoders were mostly inspired by Hiero, as are SMT systems to this day [14]. That being said, current research in SMT is largely concerned with discovering and selecting useful feature functions. The three features that we covered are not representative of the state of the art; the framework may need amendments for supporting contemporary features.

(b) We convinced ourselves that our specifications were readily effective from the outset. To this end, we treated each operation in isolation and gathered a suitable closure result. As a whole, the closure results imply a 1:1 translation of a specification into a (composite) algorithm. The preliminary version still forced us to provide small constructions ourselves; the full version is supposed to fix that. For the next refinement steps – the efficient specification and the computer program – it is yet to be shown that the same per-operation approach works.

(c) The closure results imply that said composite algorithm is correct.

(d) We exploited the theory of weighted tree automata and related devices in Sec. 1.4. We saw potential for developing the theory in Sec. 1.5. The three main contributions of this thesis (Chs. 3–5) underscore the viability of the framework, both as a specification mechanism and as an interface between theory and application.

In the following sections, we consider the full version of the framework, and we discuss further ways of developing both said theory and the framework itself. In particular, we gather preliminary evidence that the per-operation approach may be successful for said refinement steps.

6.1 The algebraic framework, full version

The full version of the algebraic framework capitalizes on semiring homomorphisms and the multiset semiring. Let us define the multiset semiring, as well as several useful

homomorphisms. For this, let $\mathcal{S} = (S, \boxdot, \bar{1})$ be a monoid. A *(finite) multiset (over $\mathcal{S}$)* is a mapping $u\colon S \to \mathbb{N}$ such that $\{s \mid u_s \neq 0\}$ is finite. We denote the set of all finite multisets over $\mathcal{S}$ by $\mathbb{N}\langle S\rangle$. We define $(1.)\colon S \to \mathbb{N}\langle S\rangle$ by letting $(1.s)_s = 1$ and $(1.s)_{s'} = 0$ for $s \neq s'$. The *(multiset) Cauchy product* $\cdot$ is the binary operation on $\mathbb{N}\langle S\rangle$ such that $(u_1 \cdot u_2)_s = \sum_{s_1, s_2 \in S\colon\, s = s_1 \boxdot s_2} (u_1)_{s_1} \cdot (u_2)_{s_2}$. The *semiring $\mathbb{N}\langle\mathcal{S}\rangle$ of (finite) multisets over $\mathcal{S}$* is $(\mathbb{N}\langle S\rangle, +, \cdot, \tilde{0}, 1.\bar{1})$, where $+$ is the conventional addition applied pointwise. The mapping $(1.)$ is a monoid homomorphism from $\mathcal{S}$ into the multiplicative monoid of $\mathbb{N}\langle\mathcal{S}\rangle$; and if $\mathcal{S}$ is commutative, then so is $\mathbb{N}\langle\mathcal{S}\rangle$.

Let $\mathcal{S}'$ be a monoid and $h\colon \mathcal{S} \to \mathcal{S}'$. We define $h^\sharp\colon \mathbb{N}\langle S\rangle \to \mathbb{N}\langle S'\rangle$ by letting $h^\sharp(u)_{s'} = \sum_{s\colon\, h(s) = s'} u_s$. Then $h^\sharp\colon \mathbb{N}\langle\mathcal{S}\rangle \to \mathbb{N}\langle\mathcal{S}'\rangle$ and $(h_2 \circ h_1)^\sharp = h_2^\sharp \circ h_1^\sharp$ (see Lm. 6.1.1 below). Now let $\mathcal{S}'$ be a semiring whose multiplicative monoid is $\mathcal{S}$. Then the *semiring $\mathbb{N}\langle\mathcal{S}'\rangle$ of multisets over $\mathcal{S}'$* is $\mathbb{N}\langle\mathcal{S}\rangle$. We define $h_{\mathcal{S}'}\colon \mathbb{N}\langle S\rangle \to S$ with $h_{\mathcal{S}'}(u) = \sum_{s \in S} \sum_{j\colon\, 0 \leq j < u_s} s$. Then $h_{\mathcal{S}'}\colon \mathbb{N}\langle\mathcal{S}'\rangle \to \mathcal{S}'$ (see Lm. 6.1.2 below), and we call it the *$\mathcal{S}'$-aggregation homomorphism*.

Let Γ be an alphabet, $\mathcal{S}$ a semiring, and $M = (Q, R, \mu, \nu)$ a WTA over Γ and $\mathcal{S}$. We define the WTA $1.M$ over Γ and $\mathbb{N}\langle\mathcal{S}\rangle$ by letting $1.M = (Q, R, (1.) \circ \mu, (1.) \circ \nu)$. The *m-meaning of M* is $[\![1.M]\!]$. The m-meaning of a WSA/a productive WSTSG is defined analogously. Let us develop some intuition for the multiset semiring and the m-meaning. Since $(1.)$ is a monoid homomorphism, we obtain that $\langle d\rangle_{(1.)\circ\mu} \cdot 1.\nu_q = 1.(\langle d\rangle_\mu \cdot \nu_q)$ for every $d \in D^q(M)$. Hence, thanks to the multiset semiring, $[\![1.M]\!](t)_s$ tells us how many runs with weight s contribute to $[\![M]\!](t)$. Of course, this information is sufficient to compute $[\![M]\!](t)$ itself; formally, $h_{\mathcal{S}}([\![1.M]\!](t)) = [\![M]\!](t)$. Intuitively speaking, the multiset semiring allows us to expose the runs to the meaning, and this is exactly what we were looking for in the closing remarks of Sec. 1.4.

Let I be a set, $\mathcal{S}, \mathcal{S}'$ semirings, and $h\colon S \to S'$. We define $h^\flat\colon S^I \to (S')^I$ by letting $h^\flat(u)_i = h(u_i)$. If $h\colon \mathcal{S} \to \mathcal{S}'$, then $h^\flat\colon \mathcal{S}^I \to (\mathcal{S}')^I$ (proof omitted). Note that $(h_2 \circ h_1)^\flat = h_2^\flat \circ h_1^\flat$. For every $d \in \mathbb{N}$, $d \geq 1$, and $u \in \mathbb{R}^d$, we define the unary operation $(\cdot u)$ on $\bar{\mathbb{R}}^d$ by letting $(\cdot u)(u') = u' \cdot u$, where $\cdot$ is the inner product. This operation is a homomorphism from the multiplicative monoid of Arct^d into itself. Here we do not distinguish between Arct^1 and Arct. For every $j \in \{1, \ldots, d\}$, we define $in_j^{(d)}\colon \bar{\mathbb{R}} \to \bar{\mathbb{R}}^d$ by letting

$$in_j^{(d)}(r)_{j'} = \begin{cases} -\infty & \text{if } r = -\infty, \\ r & \text{if } r \neq -\infty,\ j = j', \\ 0 & \text{otherwise.} \end{cases}$$

Then $in_j^{(d)}\colon \mathrm{Arct} \to \mathrm{Arct}^d$. This concludes the general definitions.

The full version of the algebraic framework is constituted by the operations of the preliminary version, as well as all operations of the form $h^\flat$, provided that the underly-

ing semirings are commutative.

In order to illustrate the framework, we rephrase (1.3). For this, let G be an STSG, μ a probability assignment for G, $\theta \in \mathbb{R}^3$, and $\theta = (\theta_1, \theta_2, \theta_3)$. Moreover, let G' be the WSTSG over Σ and Real that is obtained from G by using the weight assignment μ, let M be a deterministic WSA with $[\![M]\!](e) = P_{\mathrm{LM}}(e)$, and let M' be a bu-det WTA with $[\![M']\!](t) = P(t \mid \mathrm{yd}(t))$. Since $\mathbb{N}\langle\mathrm{Real}\rangle$ is not complete, we additionally require that G' be productive. We claim (without proof) that

$$\tau = (h_{\mathrm{Arct}} \circ ((\cdot\theta_1) \circ \log)^\sharp)^\flat([\![1.G']\!]) \;, \tag{6.1}$$

$$\varphi_{\mathrm{LM}} = ((\cdot\theta_2) \circ \log)^\flat([\![M]\!]) \;, \tag{6.2}$$

$$\varphi_{\mathrm{P}} = ((\cdot\theta_3) \circ \log)^\flat([\![M']\!]) \;, \tag{6.3}$$

where, via composition, $h_{\mathrm{Arct}} \circ ((\cdot\theta_j) \circ \log)^\sharp \colon \mathbb{N}\langle\mathrm{Real}\rangle \to \mathrm{Arct}$.

We show that the specification is again effective. To this end, we enhance the notation of our classes $\mathcal{K}$, $\mathcal{L}$, $\mathcal{T}$, and their subclasses by adding the underlying semiring as a superscript. We introduce the five classes

$$\begin{aligned}
\mathcal{K}^{\mathcal{S}}_{\mathrm{dRec}} &= \{\varphi \mid \varphi \in \mathcal{K}^{\mathcal{S}}, \varphi \text{ is the meaning of some deterministic WSA}\} \;, \\
\mathcal{L}^{\mathcal{S}}_{\mathrm{dRec}} &= \{\varphi \mid \varphi \in \mathcal{L}^{\mathcal{S}}, \varphi \text{ is the meaning of some bu-det WTA}\} \;, \\
\mathcal{K}^{\mathbb{N}\langle\mathcal{S}\rangle}_{\mathrm{mdRec}} &= \{\varphi \mid \varphi \in \mathcal{K}^{\mathbb{N}\langle\mathcal{S}\rangle}, \varphi \text{ is the m-meaning of some deterministic WSA}\} \;, \\
\mathcal{L}^{\mathbb{N}\langle\mathcal{S}\rangle}_{\mathrm{mdRec}} &= \{\varphi \mid \varphi \in \mathcal{L}^{\mathbb{N}\langle\mathcal{S}\rangle}, \varphi \text{ is the m-meaning of some bu-det WTA}\} \;, \\
\mathcal{T}^{\mathbb{N}\langle\mathcal{S}\rangle}_{\mathrm{mSTSG}} &= \{\tau \mid \tau \in \mathcal{T}^{\mathbb{N}\langle\mathcal{S}\rangle}, \tau \text{ is the m-meaning of some productive WSTSG}\} \;.
\end{aligned}$$

Table 6.1 lists results about the computability of the additional operations. For the second section of the table, we merely switch from "meaning" to "m-meaning" without touching the underlying device. For the third section of the table, we perform the corresponding constructions from Tab. 1.2 on the underlying device (over $\mathcal{S}$); note that the determinism requirement in $\mathcal{L}^{\mathbb{N}\langle\mathcal{S}\rangle}_{\mathrm{mdRec}}$ is crucial for $\odot$, $\lhd$, and $\rhd$. For the fourth section, we perform $g^\flat$ on the underlying device.

The entries referring to $g^\flat$ in the first section of the table deserve some discussion. Although it seems plausible that the semiring addition is immaterial for deterministic devices, caution is in fact advised: when there is no run, then the empty sum (the semiring zero) comes into play, and g need not map the zero of $\mathcal{S}$ to the zero of $\mathcal{S}'$. We can recover the closure result in two ways:

- Either we simply require that $g(0) = 0$. This property holds for $g = \log$, but not for $g = (1.)$, because multisets distinguish between 1.0 ("exactly one run; its weight is 0") and $\tilde{0}$ ("no run"). For our application above (φ'_{LM} and φ'_{P}), we can as well use a variation of $(1.)$ that maps 0 to $\tilde{0}$.

operation	closure/restrictions	publication	complexity
$h^\flat$	$\mathcal{K}^{\mathcal{S}}_{\text{Rec}} \to \mathcal{K}^{\mathcal{S}'}_{\text{Rec}}$	[17, Lm. 3]	$O(r)$
$h^\flat$	$\mathcal{K}^{\mathcal{S}}_{\text{dRec}} \to \mathcal{K}^{\mathcal{S}'}_{\text{dRec}}$	[17, Lm. 3]	$O(r)$
$h^\flat$	$\mathcal{L}^{\mathcal{S}}_{\text{Rec}} \to \mathcal{L}^{\mathcal{S}'}_{\text{Rec}}$	[17, Lm. 3]	$O(r)$
$h^\flat$	$\mathcal{L}^{\mathcal{S}}_{\text{dRec}} \to \mathcal{L}^{\mathcal{S}'}_{\text{dRec}}$	[17, Lm. 3]	$O(r)$
$h^\flat$	$\mathcal{T}^{\mathcal{S}}_{\text{STSG}} \to \mathcal{T}^{\mathcal{S}'}_{\text{STSG}}$	[17, Lm. 3]	$O(r)$
$g^\flat$	$\mathcal{K}^{\mathcal{S}}_{\text{dRec}} \to \mathcal{K}^{\mathcal{S}'}_{\text{dRec}}$	(conjecture)	$O(r)$
$g^\flat$	$\mathcal{L}^{\mathcal{S}}_{\text{dRec}} \to \mathcal{L}^{\mathcal{S}'}_{\text{dRec}}$	(conjecture)	$O(r)$
$(1.)^\flat$	$\mathcal{K}^{\mathcal{S}}_{\text{dRec}} \to \mathcal{K}^{\mathbb{N}\langle\mathcal{S}\rangle}_{\text{mdRec}}$	(conjecture)	$O(1)$
$(1.)^\flat$	$\mathcal{L}^{\mathcal{S}}_{\text{dRec}} \to \mathcal{L}^{\mathbb{N}\langle\mathcal{S}\rangle}_{\text{mdRec}}$	(conjecture)	$O(1)$
$(h_{\mathcal{S}})^\flat$	$\mathcal{T}^{\mathbb{N}\langle\mathcal{S}\rangle}_{\text{mSTSG}} \to \mathcal{T}^{\mathcal{S}}_{\text{STSG}}$	(conjecture)	$O(1)$
$1.$	$\Sigma^* \to \mathcal{K}^{\mathbb{N}\langle\mathcal{S}\rangle}_{\text{mdRec}}$	(conjecture)	$O(n)$
Yd^{-1}	$\mathcal{K}^{\mathbb{N}\langle\mathcal{S}\rangle}_{\text{mdRec}} \to \mathcal{L}^{\mathbb{N}\langle\mathcal{S}\rangle}_{\text{mdRec}}$	(conjecture)	$O(p^k)$
$\odot$	$\mathcal{L}^{\mathbb{N}\langle\mathcal{S}\rangle}_{\text{mdRec}} \times \mathcal{L}^{\mathbb{N}\langle\mathcal{S}\rangle}_{\text{mdRec}} \to \mathcal{L}^{\mathbb{N}\langle\mathcal{S}\rangle}_{\text{mdRec}}$	(conjecture)	$O(r_1 \cdot r_2)$
$\triangleleft$	$\mathcal{L}^{\mathbb{N}\langle\mathcal{S}\rangle}_{\text{mdRec}} \times \mathcal{T}^{\mathbb{N}\langle\mathcal{S}\rangle}_{\text{mSTSG}} \to \mathcal{T}^{\mathbb{N}\langle\mathcal{S}\rangle}_{\text{mSTSG}}$	(Lm. 3.3.2)	$O(r_2 \cdot p_1^{k_2})$
$\triangleright$	$\mathcal{T}^{\mathbb{N}\langle\mathcal{S}\rangle}_{\text{mSTSG}} \times \mathcal{L}^{\mathbb{N}\langle\mathcal{S}\rangle}_{\text{mdRec}} \to \mathcal{T}^{\mathbb{N}\langle\mathcal{S}\rangle}_{\text{mSTSG}}$	(Lm. 3.3.2)	$O(r_1 \cdot p_2^{k_1})$
$(g^\sharp)^\flat$	$\mathcal{K}^{\mathbb{N}\langle\mathcal{S}\rangle}_{\text{mdRec}} \to \mathcal{K}^{\mathbb{N}\langle\mathcal{S}'\rangle}_{\text{mdRec}}$	(conjecture)	$O(r)$
$(g^\sharp)^\flat$	$\mathcal{L}^{\mathbb{N}\langle\mathcal{S}\rangle}_{\text{mdRec}} \to \mathcal{L}^{\mathbb{N}\langle\mathcal{S}'\rangle}_{\text{mdRec}}$	(conjecture)	$O(r)$
$(g^\sharp)^\flat$	$\mathcal{T}^{\mathbb{N}\langle\mathcal{S}\rangle}_{\text{mSTSG}} \to \mathcal{T}^{\mathbb{N}\langle\mathcal{S}'\rangle}_{\text{mSTSG}}$	(conjecture)	$O(r)$

where $\mathcal{S}$ and $\mathcal{S}'$ are semirings, $h\colon \mathcal{S} \to \mathcal{S}'$, and $g\colon S \to S'$ is a monoid homomorphism from the multiplicative monoid of $\mathcal{S}$ into the multiplicative monoid of $\mathcal{S}'$

Table 6.1: Computability of operations, continued from Tab. 1.2.

- Or we preprocess the given WSA or WTA: roughly speaking, we add a sink state and, where necessary, transitions of weight 0. This intrusion does not affect the meaning, but it guarantees that there is exactly one run for every string or tree, so that empty sums are ruled out.

As in Sec. 1.4, let us now use the framework according to its purpose and specify a new decoder. We let $\chi_j = ((in_j^{(3)} \circ \log)^\sharp)^\flat$ and define

$$\begin{aligned}
&\mathbb{D}_2 \colon \mathcal{T}_{\text{mSTSG}}^{\mathbb{N}\langle\text{Real}\rangle} \times \mathcal{K}_{\text{mdRec}}^{\mathbb{N}\langle\text{Real}\rangle} \times \mathcal{L}_{\text{mdRec}}^{\mathbb{N}\langle\text{Real}\rangle} \times \mathbb{R}^3 \to E^F,\ \mathbb{D}_2(\tau, \varphi, \varphi', \theta) \colon \\
&\qquad f \mapsto \text{best}(\text{Yd}(\pi_2\big([h_{\text{Arct}} \circ (\cdot\theta)^\sharp]^\flat(\varphi_0)\big))) \text{ , where} \qquad (6.4)\\
&\varphi_0 = \big(\text{Yd}^{-1}(1.f) \odot \chi_3(\varphi')\big) \lhd \chi_1(\tau) \rhd \text{Yd}^{-1}\big(\chi_2(\varphi)\big) \ .
\end{aligned}$$

This specification enjoys the nice property that, by a minor change, we obtain a decoder in the spirit of $\mathbb{D}_0'$ of Sec. 1.2, which selects the translation with the highest aggregate score. The change consists in replacing h_{Arct} by $h_{\text{Real}} \circ \exp^\sharp$. Be advised that, like $\mathbb{D}_0'$, the resulting decoder is NP hard.

We conclude this section by providing the two promised lemmas and by discussing the prospects of "infinite multisets".

Lemma 6.1.1 *Let $\mathcal{S}, \mathcal{S}', \mathcal{S}''$ be monoids and $h \colon \mathcal{S} \to \mathcal{S}'$. Then $h^\sharp \colon \mathbb{N}\langle\mathcal{S}\rangle \to \mathbb{N}\langle\mathcal{S}'\rangle$. Moreover, let $h_1 \colon \mathcal{S} \to \mathcal{S}'$, $h_2 \colon \mathcal{S}' \to \mathcal{S}''$. Then $(h_2 \circ h_1)^\sharp = h_2^\sharp \circ h_1^\sharp$.*

PROOF. For the first statement, only the multiplication is somewhat tricky. We derive

$$\begin{aligned}
&[h^\sharp(u_1) \cdot h^\sharp(u_2)]_{s'} \\
&= \textstyle\sum_{s_1', s_2' \colon s' = s_1' \boxdot' s_2'} h^\sharp(u_1)_{s_1'} \cdot h^\sharp(u_2)_{s_2'} \\
&= \textstyle\sum_{s_1', s_2' \colon s' = s_1' \boxdot' s_2'} \big(\sum_{s_1 \colon h(s_1) = s_1'} (u_1)_{s_1}\big) \cdot \big(\sum_{s_2 \colon h(s_2) = s_2'} (u_2)_{s_2}\big) \\
&= \textstyle\sum_{s_1', s_2' \colon s' = s_1' \boxdot' s_2'} \sum_{s_1 \colon h(s_1) = s_1'} \sum_{s_2 \colon h(s_2) = s_2'} (u_1)_{s_1} \cdot (u_2)_{s_2} \qquad \text{(distributivity)} \\
&= \textstyle\sum_{s_1', s_2', s_1, s_2 \colon s' = h(s_1) \boxdot' h(s_2), h(s_1) = s_1', h(s_2) = s_2'} (u_1)_{s_1} \cdot (u_2)_{s_2} \\
&= \textstyle\sum_{s_1', s_2', s_1, s_2 \colon s' = h(s_1 \boxdot s_2), h(s_1) = s_1', h(s_2) = s_2'} (u_1)_{s_1} \cdot (u_2)_{s_2} \\
&= \textstyle\sum_{s_1, s_2 \colon s' = h(s_1 \boxdot s_2)} (u_1)_{s_1} \cdot (u_2)_{s_2} = \sum_{s, s_1, s_2 \colon h(s) = s', s = s_1 \boxdot s_2} (u_1)_{s_1} \cdot (u_2)_{s_2} \\
&= \textstyle\sum_{s \colon h(s) = s'} \sum_{s_1, s_2 \colon s = s_1 \boxdot s_2} (u_1)_{s_1} \cdot (u_2)_{s_2} = h^\sharp(u_1 \cdot u_2)_{s'} \ .
\end{aligned}$$

For the second statement, we derive

$$\begin{aligned}
&(h_2 \circ h_1)^\sharp(u)_{s''} = \textstyle\sum_{s \colon h_2(h_1(s)) = s''} u_s = \sum_{s', s \colon h_2(s') = s'', h_1(s) = s'} u_s \\
&= \textstyle\sum_{s' \colon h_2(s') = s''} \sum_{s \colon h_1(s) = s'} u_s = \sum_{s' \colon h_2(s') = s''} h_1^\sharp(u)_s = h_2^\sharp(h_1^\sharp(u))_{s''} \ . \qquad \blacksquare
\end{aligned}$$

Lemma 6.1.2 *Let $\mathcal{S}$ be a semiring. Then $h_{\mathcal{S}} \colon \mathbb{N}\langle\mathcal{S}\rangle \to \mathcal{S}$.*

PROOF. Let $h = h_{\mathcal{S}}$. We have that $h(\tilde{0}) = 0$ and $h(1.1) = 1$. For every $j \in \mathbb{N}$, let $[j] = \{j' \mid 0 \leq j' < j\}$. We derive

$$\begin{aligned}
&h(u_1 + u_2) = \textstyle\sum_{s\in S} \sum_{j\in[(u_1)_s+(u_2)_s]} s \\
&= \textstyle\sum_{s\in S} \sum_{j\in\{s\}\times\{1\}\times[(u_1)_s]\cup\{s\}\times\{2\}\times[(u_2)_s]} s \qquad (\star) \\
&= \textstyle\sum_{j\in\bigcup_{s\in S}\left(\{s\}\times\{1\}\times[(u_1)_s]\cup\{s\}\times\{2\}\times[(u_2)_s]\right)} s \\
&= \textstyle\sum_{j\in\left(\bigcup_{s\in S}\{s\}\times\{1\}\times[(u_1)_s]\right)\cup\left(\bigcup_{s\in S}\{s\}\times\{2\}\times[(u_2)_s]\right)} s \\
&= \left(\textstyle\sum_{j\in\bigcup_{s\in S}\{s\}\times\{1\}\times[(u_1)_s]} s\right) + \left(\textstyle\sum_{j\in\bigcup_{s\in S}\{s\}\times\{2\}\times[(u_2)_s]} s\right) \\
&= \left(\textstyle\sum_{s\in S}\sum_{j\in\{s\}\times\{1\}\times[(u_1)_s]} s\right) + \left(\textstyle\sum_{s\in S}\sum_{j\in\{s\}\times\{2\}\times[(u_2)_s]} s\right) \\
&= \left(\textstyle\sum_{s\in S}\sum_{j\in[(u_1)_s]} s\right) + \left(\textstyle\sum_{s\in S}\sum_{j\in[(u_2)_s]} s\right) = h(u_1) + h(u_2)\ .
\end{aligned}$$

Next, we derive

$$\begin{aligned}
&h(u_1 \cdot u_2) = \textstyle\sum_{s\in S} \sum_{j\in[(u_1\cdot u_2)_s]} s \\
&= \textstyle\sum_{s\in S} \sum_{j\in[\sum_{s_1,s_2\in S\colon\, s=s_1\cdot s_2}(u_1)_{s_1}\cdot(u_2)_{s_2}]} s \\
&= \textstyle\sum_{s\in S} \sum_{j\in\bigcup_{s_1,s_2\in S\colon\, s=s_1\cdot s_2}\{s_1\}\times\{s_2\}\times[(u_1)_{s_1}]\times[(u_2)_{s_2}]} s \\
&= \textstyle\sum_{s_1\in S}\sum_{s\in S}\sum_{j\in\bigcup_{s_2\in S\colon\, s=s_1\cdot s_2}\{s_1\}\times\{s_2\}\times[(u_1)_{s_1}]\times[(u_2)_{s_2}]} s \\
&= \textstyle\sum_{s_1\in S}\sum_{s_2\in S}\sum_{s\in S\colon\, s_1\cdot s_2=s}\sum_{j\in\{s_1\}\times\{s_2\}\times[(u_1)_{s_1}]\times[(u_2)_{s_2}]} s \\
&= \textstyle\sum_{s_1\in S}\sum_{s_2\in S}\sum_{j'\in[(u_1)_{s_1}]}\sum_{j\in\{s_1\}\times\{s_2\}\times\{j'\}\times[(u_2)_{s_2}]} s_1\cdot s_2 \\
&= \textstyle\sum_{s_1\in S}\sum_{j\in[(u_1)_{s_1}]}\sum_{s_2\in S}\sum_{j\in[(u_2)_{s_2}]} s_1\cdot s_2 \\
&= \textstyle\sum_{s_1\in S}\sum_{j\in[(u_1)_{s_1}]}\left(s_1\cdot\sum_{s_2\in S}\sum_{j\in[(u_2)_{s_2}]} s_2\right) \\
&= \left(\textstyle\sum_{s_1\in S}\sum_{j\in[(u_1)_{s_1}]} s_1\right)\cdot\left(\textstyle\sum_{s_2\in S}\sum_{j\in[(u_2)_{s_2}]} s_2\right) = h(u_1)\cdot h(u_2)\ .
\end{aligned}$$

■

Let $\mathcal{S}$ be a complete semiring. We let $\mathbb{N}_\infty = \mathbb{N} \cup \{\infty\}$. A *multiset over* $\mathcal{S}$ is a mapping $u \colon S \to \mathbb{N}_\infty$; we denote the set of all multisets over $\mathcal{S}$ by $\mathbb{N}_\infty\langle\langle S\rangle\rangle$. Note that this is a generalization of the finite multisets over $\mathcal{S}$. Recall that $(\mathbb{N}_\infty, +, \cdot, 0, 1)$ with conventional addition and multiplication is a complete semiring. The semiring $\mathbb{N}_\infty\langle\langle\mathcal{S}\rangle\rangle$ is defined in the same way as $\mathbb{N}\langle\mathcal{S}\rangle$, and it is complete. The definition of an $\mathcal{S}$-aggregation mapping transfers to the new setting easily – unlike Lm. 6.1.2.

In fact, we can *not* readily prove that $h_{\mathcal{S}}$ is a complete semiring homomorphism, even if $\mathcal{S}$ is complete. To see this, we inspect $(\star)$ of the proof. There we switch the

index set of the innermost sum:

$$[(u_1)_s + (u_2)_s] \quad \rightsquigarrow \quad \big(\{s\} \times \{1\} \times [(u_1)_s]\big) \cup \big(\{s\} \times \{2\} \times [(u_2)_s]\big) \,.$$

Here either both sets have the same finite cardinality, so the corresponding sums coincide, or both sets are countably infinite, and the sums coincide as well, because the infinitary sum is invariant with respect to renaming of index elements. When we try out the same technique to prove $h(\sum_{i \in I} u_i) = \sum_{i \in I} h(u)_i$, we are faced with

$$[\textstyle\sum_{i \in I}(u_i)_s] \quad \rightsquigarrow \quad \bigcup_{i \in I} \{s\} \times \{i\} \times [(u_i)_s] \,.$$

Here the following case can occur: the left set is countably infinite, while the right set is uncountable. Then the sums need not coincide. However, it should be possible to recover the result by considering countable multisets and "countably complete" semirings throughout, for countable unions of countable sets are again countable (assuming the axiom of choice). Or, should one be so inclined, one might try and define multisets where the multiplicities are cardinal numbers.

6.2 Outlook

Throughout this thesis, we encountered open problems. For instance, some entries in Tabs. 1.2 and 6.1 are marked as conjectures; they should be looked into. It was suggested to investigate variable-deleting WSCFTGs (Sec. 3.5) and IRTGs with deleting homomorphisms, to study the input product for yXHGs and WSCFHGs, to examine binarization-friendly WSCFHGs, and to work on the problem of rank-optimization (all Sec. 4.7). One might also consider determinization of non-acyclic WTA over non-extremal semirings (Sec. 5.6). Finally, it would be interesting to replace the concept of complete semirings by the concept of countably complete semirings, and to show whether Lm. 6.1.2 can be generalized to countable multisets.

In the remainder, we consider how to extend the algebraic framework.

6.2.1 An n-best operation

Our definition of a decoder as a mapping $\mathbb{D} \colon \Omega \to E^F$ is closely tailored to its application for translation: we assume that some "good" ω is known, and we translate any sentence f by computing $\mathbb{D}(\omega)(f)$. From this idealized perspective, the inner workings of $\mathbb{D}$ – e.g., the flow of information through the defining expression of $\mathbb{D}$ – are irrelevant, and therefore these inner workings are not exposed in the type of $\mathbb{D}$.

However, during development it is likely that a decoder does not meet our expectations. That is, it performs poorly on a test sentence, as measured by our professional

intuition. Simply put, we feel that the translation selected by best is bad. It is natural to ask whether the argument WSA of best offers a better, yet lower-scored translation. Maybe our defining expression of $\mathbb{D}$ needs a revision. Or maybe ω was not so good after all, and we can spot a problem with our training procedure.

An established diagnostic tool for this situation is the list of n best runs of said WSA [70, 94]. This list is also used to finitely approximate the meaning of the WSA: for instance, many training procedures work on this list, in particular, to select the feature weight vector (cf. [150, Sec. 4], [43, Sec. 2], [92, Sec. 2], [84, Sec. 2]).

In our algebraic ideology, we express the computation of the n-best list as an operation on the meaning level, and we already have the right notions at our disposal: the multiset semiring and the m-meaning. In order to keep the presentation simple, we let $\mathcal{S}$ be Arct or Real.

Let $n \in \mathbb{N}$ and I a set. We define $n\mathrm{best}\colon \mathbb{N}_\infty\langle\langle\mathcal{S}\rangle\rangle^I \to \mathbb{N}\langle\mathcal{S}\rangle^I$, also denoted $[.]_n$, as follows. Let $\varphi \in \mathbb{N}\langle\mathcal{S}\rangle^I$. Using the pointwise extension of $\leq$, we define

$$\begin{aligned}\mathcal{C}(n,\varphi) &= \{\varphi' \mid \varphi' \in \mathbb{N}\langle\mathcal{S}\rangle^I, \textstyle\sum_{i\in I, s\in S} \varphi'(i)_s \leq n, \varphi' \leq \varphi\}\ ,\\ n\mathrm{best}(\varphi) &= \mathrm{argmax}_{\varphi'\in\mathcal{C}(n,\varphi)} \textstyle\sum_{i\in I} h_{\mathcal{S}}(\varphi'(i))\ .\end{aligned}$$

With this operation, we can express the n-best list for $\mathbb{D}_2(\tau,\varphi,\varphi',\theta)(f)$ as

$$\big[\mathrm{Yd}(\pi_2([(\cdot\theta)^\sharp]^\flat(\varphi_0)))\big]_n\ . \tag{6.5}$$

Transferring the m-meaning from $\mathbb{N}\langle\mathcal{S}\rangle$ to $\mathbb{N}_\infty\langle\langle\mathcal{S}\rangle\rangle$, we define the six classes

$$\begin{aligned}\mathcal{K}_{\text{mRec}}^{\mathbb{N}_\infty\langle\langle\mathcal{S}\rangle\rangle} &= \{\varphi \mid \varphi \in \mathcal{K}^{\mathbb{N}_\infty\langle\langle\mathcal{S}\rangle\rangle}, \varphi \text{ is the m-meaning of some WSA}\}\ ,\\ \mathcal{L}_{\text{mRec}}^{\mathbb{N}_\infty\langle\langle\mathcal{S}\rangle\rangle} &= \{\varphi \mid \varphi \in \mathcal{L}^{\mathbb{N}_\infty\langle\langle\mathcal{S}\rangle\rangle}, \varphi \text{ is the m-meaning of some WTA}\}\ ,\\ \mathcal{T}_{\text{mSTSG}}^{\mathbb{N}_\infty\langle\langle\mathcal{S}\rangle\rangle} &= \{\tau \mid \tau \in \mathcal{T}^{\mathbb{N}_\infty\langle\langle\mathcal{S}\rangle\rangle}, \tau \text{ is the m-meaning of some WSTSG}\}\ ,\\ \mathcal{K}_{\text{fin}}^{\mathcal{S}} &= \{\varphi \mid \varphi \in \mathcal{K}^{\mathcal{S}}, \{i \mid \varphi(i) \neq 0\} \text{ finite}\}\ ,\\ \mathcal{L}_{\text{fin}}^{\mathcal{S}} &= \{\varphi \mid \varphi \in \mathcal{L}^{\mathcal{S}}, \{i \mid \varphi(i) \neq 0\} \text{ finite}\}\ ,\\ \mathcal{T}_{\text{fin}}^{\mathcal{S}} &= \{\tau \mid \tau \in \mathcal{T}^{\mathcal{S}}, \{i \mid \tau(i) \neq 0\} \text{ finite}\}\ .\end{aligned}$$

Table 6.2 shows computability results for $n\mathrm{best}$. Let us briefly sketch the idea behind the second section in the table on the basis of the string case. We convert the underlying WSA M into a WSA M' with weights in $1.S$ such that $[\![M]\!]_n = [\![M']\!]_n$ as follows. For every transition whose weight u is not in $1.S$, we introduce at most n "dummy" states, corresponding to the n best entries in u, we remove said transition, and for each dummy state, we introduce a transition from the original source to the dummy, reading the original symbol with the appropriate weight in $1.S$, and we a introduce

operation	closure/restrictions	publications	complexity
$n\mathrm{best}$	$\mathcal{K}_{\mathrm{mRec}}^{\mathbb{N}_\infty\langle\langle\mathcal{S}\rangle\rangle} \to \mathcal{K}_{\mathrm{fin}}^{\mathbb{N}\langle\mathcal{S}\rangle}$, (†)	[94, 28]	(ϕ)
$n\mathrm{best}$	$\mathcal{L}_{\mathrm{mRec}}^{\mathbb{N}_\infty\langle\langle\mathcal{S}\rangle\rangle} \to \mathcal{L}_{\mathrm{fin}}^{\mathbb{N}\langle\mathcal{S}\rangle}$, (†)	[94, 28]	(ϕ)
$n\mathrm{best}$	$\mathcal{T}_{\mathrm{mSTSG}}^{\mathbb{N}_\infty\langle\langle S\rangle\rangle} \to \mathcal{T}_{\mathrm{fin}}^{\mathbb{N}\langle S\rangle}$, (†)	[94, 28]	(ϕ)
$n\mathrm{best}$	$\mathcal{K}_{\mathrm{Rec}}^{\mathbb{N}_\infty\langle\langle\mathcal{S}\rangle\rangle} \to \mathcal{K}_{\mathrm{fin}}^{\mathbb{N}\langle\mathcal{S}\rangle}$, (†)	(conjecture)	(?)
$n\mathrm{best}$	$\mathcal{L}_{\mathrm{Rec}}^{\mathbb{N}_\infty\langle\langle\mathcal{S}\rangle\rangle} \to \mathcal{L}_{\mathrm{fin}}^{\mathbb{N}\langle\mathcal{S}\rangle}$, (†)	(conjecture)	(?)
$n\mathrm{best}$	$\mathcal{T}_{\mathrm{STSG}}^{\mathbb{N}_\infty\langle\langle S\rangle\rangle} \to \mathcal{T}_{\mathrm{fin}}^{\mathbb{N}\langle S\rangle}$, (†)	(conjecture)	(?)

(ϕ) $O(r + p \cdot n \cdot \log(r + n))$

Table 6.2: Computability of operations, continued from Tab. 1.2.

an ε-transition from the dummy to the original target with weight 1.1. Instead of ε, the latter transition can also read some special symbol; then we have to postprocess $[\![M']\!]_n$ accordingly in order to arrive at $[\![M]\!]_n$.

As for the effectiveness of (6.5), we observe that Tab. 6.1 lacks entries for Yd and for π_2. For these, we can keep using Tab. 1.2 via the trivial inclusions

$$\mathcal{L}_{\mathrm{mRec}}^{\mathbb{N}\langle\mathcal{S}\rangle} \subseteq \mathcal{L}_{\mathrm{Rec}}^{\mathbb{N}\langle\mathcal{S}\rangle} \subseteq \mathcal{L}_{\mathrm{Rec}}^{\mathbb{N}_\infty\langle\langle\mathcal{S}\rangle\rangle}\,, \qquad \mathcal{T}_{\mathrm{mSTSG}}^{\mathbb{N}\langle\mathcal{S}\rangle} \subseteq \mathcal{T}_{\mathrm{STSG}}^{\mathbb{N}\langle\mathcal{S}\rangle} \subseteq \mathcal{T}_{\mathrm{STSG}}^{\mathbb{N}_\infty\langle\langle\mathcal{S}\rangle\rangle}\,.$$

6.2.2 Reranking and crunching

The refinement step from an effective specification towards an efficient one usually involves introducing approximations. To that effect, n-best lists can be used.

For instance, if the computation of $\mathbb{D}_2$ is being held up by the output product, then we could use the following alternative of φ_0:

$$\varphi_0' = \Big[\big(\mathrm{Yd}^{-1}(1.f) \odot \chi_3(\varphi')\big) \triangleleft \chi_1(\tau)\Big]_n \triangleright \mathrm{Yd}^{-1}\big(\chi_2(\varphi)\big)\,,$$

Now the output product acts on a finite weighted tree transformation, which makes it cheap for reasonable n; in fact, we merely rerank a finite list. Therefore, this approximation technique is called *reranking*.

Another technique is *crunching* (cf. [134, Sec. 5.1], [121, Sec. 2.4]). As mentioned below our definition of $\mathbb{D}_2$, we obtain a decoder that sums over ASTs by replacing h_{Arct} by $h_{\mathrm{Real}} \circ \exp^\sharp$, and that decoder is NP hard. We define the decoder $\mathbb{D}_2'$ with the same

parameter space and with

$$\mathbb{D}'_2(\tau,\varphi,\varphi',\theta)(f) = \mathrm{best}(h^{\flat}_{\mathrm{Real}}(\Big[\mathrm{Yd}(\pi_2([[\exp\circ(\cdot\theta)]^{\sharp}]^{\flat}(\varphi_0)))\Big]_n)) \,.$$

Here the aggregation homomorphism can be imagined as "crunching" a finite list. Evidently, computing best of a finite weighted string language is trivial.

6.2.3 Relatively-useless pruning

Reducing a WSA to a finite list of runs, $n\mathrm{best}$ is a quite drastic and crude measure. In contrast, *pruning* refers to a class of more refined techniques, which merely reduce the number of states or the number of transitions in one way or another.

For instance, *relatively-useless pruning (RUP)* [93, Sec. 4.3] is based on evolving the useful/useless dichotomy for transitions that we established in Sec. 2.4.5. Let $M = (Q, R, \mu, q_0)$ be a WTA over Σ and Arct in root-state form. We define the *utility* $\eta(\rho)$ of a transition $\rho \in R$ as the highest weight of any complete run that contains said transition, i.e.,

$$\eta(\rho) = \max\{\langle d\rangle \mid d \in D^{q_0}_{\mathrm{co}}(M), \exists w \in \mathrm{pos}(d)\colon d(w) = \rho\} \,.$$

In [93], the utility is called *merit*, and it is computed efficiently using the inside and outside weights of M (where the former correspond to the mapping F of Lm. 2.4.7).

Clearly, a useless transition has utility 0, but not vice versa. We also consider the highest weight of any run

$$\eta_0 = \max\{\langle d\rangle \mid d \in D^{q_0}_{\mathrm{co}}(M)\} = \max\{\eta(\rho) \mid \rho \in R\} \,.$$

For RUP, we let $\delta \in \mathbb{R}^{\geq 0}$. We construct the WTA $\delta\mathrm{rup}(M)$ from M by dropping every transition ρ with $\eta_0 - \eta(\rho) > \delta$. We observe that

$$\{d \mid d \in D^{q_0}_{\mathrm{co}}(M), \langle d\rangle \geq \eta_0 - \delta\} \;\subseteq\; D^{q_0}_{\mathrm{co}}(\delta\mathrm{rup}(M)) \;\subseteq\; D^{q_0}_{\mathrm{co}}(M) \,, \tag{6.6}$$

where the second set may still contain runs with weight below $\eta_0 - \delta$. It is hard to estimate, without inspecting R, the number of dropped transitions, and thus the gain in efficiency of subsequent operations. We may need to try different values of δ until we find a practical trade-off between accuracy and efficiency.

Let us attempt to express RUP on the m-meaning level. For this, let I a set. A *δ-rup mapping* is an operation $\delta\mathrm{rup}\colon \mathbb{N}_\infty\langle\langle\mathrm{Arct}\rangle\rangle^I \to \mathbb{N}_\infty\langle\langle\mathrm{Arct}\rangle\rangle^I$ such that

$$\varphi_\delta \leq \delta\mathrm{rup}(\varphi) \leq \varphi \,, \tag{6.7}$$

where $\varphi_\delta \in \mathbb{N}_\infty\langle\langle \mathrm{Arct} \rangle\rangle^I$ with

$$\varphi_\delta(i)_s = \begin{cases} \varphi(i)_s & \text{if } s \geq s_0(\varphi) - \delta, \\ 0 & \text{otherwise,} \end{cases}$$
$$s_0(\varphi) = \max\{s \mid \exists i \colon \varphi(i)_s \neq 0\} \ .$$

The inequations (6.7) model (6.6). This definition of a δ-rup mapping is permissive: for instance, it includes the identity on $\mathbb{N}_\infty\langle\langle \mathrm{Arct} \rangle\rangle^I$. It is unclear how to narrow down the definition, because the identities of the runs and their interdependencies via the transitions are masked on the m-meaning level. For use in decoder specifications, we let $\delta\mathrm{rup}$ be a δ-rup mapping; and we stipulate that it be the one that the (fictitious, potential) implementation of choice provides.

6.2.4 Cube pruning

In the strict sense [39, 81, 82], *cube pruning* is an algorithm that explores a certain weighted hypergraph under a limited-memory restriction concerning the number of nodes. Roughly speaking, this weighted hypergraph encodes the output-product construction of an acyclic WSCFG and an n-gram model. In the broad sense [25], which we adopt, cube pruning is a general technique for approximating product constructions such as the output product, the input product, or the Hadamard product, when at least one operand is given by an acyclic device.

Although a detailed formal account of this technique is long overdue, it would go beyond the scope of this text. Suffice it to say the following. Since cube pruning is very operational in nature, it cannot be easily expressed as an operation on the m-meaning level. As was the case for RUP, one can probably only give a very permissive (or even vacuous) definition; for instance, one that does not reflect the limited-memory restriction. It may turn out inevitable that we supplement the formal definition with informal requirements that refer to the automata level.

6.2.5 Support for more feature functions

We have considered four features: one that is induced by a probability assignment, one that incorporates a language model for English, one that counts the number of words in the English string, and one that incorporates parsing probabilities for French. Each feature can be regarded as the composition of an "adapter mapping" and a "substance mapping"; for instance, in the setting of Sec. 1.3, we observe that

$$\phi_{\mathrm{LM}} = (\log \circ P_{\mathrm{LM}}) \circ (\mathrm{yd} \circ h_2 \circ \pi_\Gamma) \ ,$$

where $\mathrm{yd} \circ h_2 \circ \pi_\Gamma$ is the adapter and $\log \circ P_{\mathrm{LM}}$ caters for the substance.

For these features, the substance is either a probability assignment or the meaning of some grammar; and the adapter is either the identity, or it refers to the French side (via h_1), or it refers to the English side (via h_2). We could transfer these features into the algebraic framework easily, by using a WSTSG, by using the input product, or by using the output product, respectively.

Current SMT systems use features that depend on both the French and the English side, e.g., the feature $\tau_0 \circ ad$ with

$$\begin{aligned}
&ad\colon T_R \to T_\Sigma \times T_\Sigma \;,\\
&ad(d) = (h_1(\pi_\Gamma(d)), h_2(\pi_\Gamma(d)))\;,\\
&\tau_0\colon T_\Sigma \times T_\Sigma \to \bar{\mathbb{R}}\;,\\
&\tau_0(t_1,t_2) = \begin{cases} 1 & \text{if } t_1 \text{ contains } \mathrm{NN}(\text{katze}) \text{ and } t_2 \text{ contains } \mathrm{NN}(\text{cat}),\\ 0 & \text{otherwise.}\end{cases}
\end{aligned}$$

In order to incorporate such a feature into the framework, we need a subclass $\mathcal{T}' \subseteq \mathcal{T}$ such that (i) $\mathcal{T}'$ contains τ_0 (and the like) and (ii) $\tau \in \mathcal{T}_{\mathrm{STSG}}, \tau' \in \mathcal{T}'$ implies $\tau \odot \tau' \in \mathcal{T}_{\mathrm{STSG}}$, where we use the Hadamard product in the general sense of Sec. 2.3.2.

For instance, one might define the notion of a "duplex WTA" M that basically consists of two WTA M_1 and M_2, but it has a central root-weight mapping ν that acts on the product state space. Then the meaning $[\![M]\!]$ of M would be the weighted tree transformation defined by

$$[\![M]\!](t_1,t_2) = \textstyle\sum_{(q_1,q_2)} [\![t_1]\!]_{M_1}(q_1) \cdot [\![t_2]\!]_{M_2}(q_2) \cdot \nu(q_1,q_2)\;.$$

The author conjectures that the class of meanings of WSTSGs is closed under Hadamard product with meanings of duplex WTA, which includes τ_0.

6.2.6 Improving the learning curve

Evidently the framework incorporates a great many concepts, and its current form is closer to a proposal than to a product. Furthermore, the presentation in this thesis is geared towards researchers who may work on the framework itself, rather than towards practitioners. As a result, this thesis fails to achieve Objective (e). However, that does not mean that the framework inherently defies an instructive presentation.

Already, the "native" specifications in (1.4) and (6.4) are rather concise and instructive. We can make them more palatable by introducing a few conventions, e.g.,

- we omit the Yd^{-1}, the $\sharp$, and the $\flat$,

- we abbreviate $(\cdot\theta)(\tau)$ by $\tau \cdot \theta$, and
- if $\mathcal{S} = \mathbb{N}_\infty\langle\langle\mathcal{S}'\rangle\rangle$, then we define $\mathrm{best} = \mathrm{best} \circ h_{\mathcal{S}'}$ (this definition is not circular because the semiring is being switched).

Thereby we reduce redundancy, which is warranted because decoder specifications will consist of a number of established stereotypes. Metaphorically speaking, decoder specifications will usually be variations on a well-known theme.

Now $\mathbb{D}_2$ can be expressed by

$$\mathbb{D}_2 \colon \mathcal{T}_{\text{mSTSG}}^{\mathbb{N}\langle\text{Real}\rangle} \times \mathcal{K}_{\text{mdRec}}^{\mathbb{N}\langle\text{Real}\rangle} \times \mathcal{L}_{\text{mdRec}}^{\mathbb{N}\langle\text{Real}\rangle} \times \mathbb{R}^3 \to E^F,\ \mathbb{D}_2(\tau, \varphi, \varphi', \theta) \colon \\ f \mapsto \mathrm{best}(\mathrm{Yd}(\pi_2\big([\chi_3(1.f \odot \varphi') \lhd \chi_1(\tau) \rhd \chi_2(\varphi)] \cdot \theta\big))) \,,$$

and we can attach a casual narrative to this specification, e.g.,

> We take the tree transformation (τ), the target language model (φ), and the parsing model (φ'); we intersect the parsing model with our sentence (f); we convert all weights into vector form; we combine the three pieces; and we incorporate the feature weight vector. It remains to project the resulting transformation to the output string language and find the translation with the highest score.

It is an interesting open question whether the algebraic notation can be taught to a practitioner by means of a collection of pairs (example, narrative), without delving into concepts such as a multiset semiring, a complete semiring, or a closure result.

6.2.7 Formal syntax and term rewriting

So far, we were content with treating specifications informally. For instance, when we sloppily refer to "the defining expression of $\mathbb{D}_2(\tau, \varphi, \varphi', \theta)(f)$", we mean the mathematical expression shown in (6.4). In contrast to a tree over Δ, this expression is itself not a mathematical object, i.e., we cannot use mathematics to describe its properties or potential transformations.

The informal treatment got us very far, for we could in many cases prove that a specification is effective, using tables such as Tab. 1.2. However, these proofs still amounted to manual labor. If we are to automate these proofs, to implement a specification in a programming language, or to build an integrated development environment (IDE) for our specifications, then there is no way around a formal treatment, either in said programming language itself or in a more abstract framework.

One possible framework is the initial-algebra approach of [86, 85]. As mentioned in Sec. 1.6, we may view our algebraic framework as a many-sorted algebra over a

suitable signature and a specification as a term (read: a tree) over this signature. By considering additional algebras over the same signature, we obtain other meanings of the same term.

This way, we can, e.g., transform the term into a term over a more elaborate signature that captures a lower-level view. In this view, the meaning $[\![G]\!]'$ of a WSCFTG G (cf. Sec. 3.2) could be represented directly by the weighted tree language $[\![G]\!]$ of center trees. For example, a term that represents $\mathrm{best}(\mathrm{Yd}(\pi_2([\![G]\!]')))$ might be transformed into a term that represents $\mathrm{best}(\mathrm{Yd}(h_2([\![G]\!])))$.

Simple relationships, such as (for $\mathcal{S} = \mathrm{Arct}$)

$$\mathrm{best} \circ \mathrm{Yd} = \mathrm{yd} \circ \mathrm{best} \;, \qquad \mathrm{best} \circ h_2 = h_2 \circ \mathrm{best} \;,$$

can be encoded as formal equations and interpreted as rewrite rules, so that we may automatically rewrite

$$\mathrm{best}(\mathrm{Yd}(h_2([\![G]\!]))) \rightsquigarrow \mathrm{yd}(\mathrm{best}(h_2([\![G]\!]))) \rightsquigarrow \mathrm{yd}(h_2(\mathrm{best}([\![G]\!]))) \;,$$

which improves the runtime because yd is cheaper than Yd and h_2 is cheaper on a single tree than on a weighted tree language. It is an interesting question whether this kind of automatic rewriting is important to achieve state-of-the-art performance.

6.2.8 Implementation and tooling

Next to the initial-algebra approach, the programming language Haskell can be used to formalize the algebraic framework. In fact, a rudimentary precursor of the framework has been implemented as a Haskell library called Vanda [26, 23]. A key feature of Haskell is its evaluation strategy, which is called *lazy evaluation*. It is conceivable that during the evaluation of an operation such as best, only a part of the argument WSA is being explored, so it might be prudent to construct said WSA lazily, i.e., on demand. Note that a conventional decoder implementation typically uses a monolithic algorithm that is highly "hand-optimized", and that interweaves the operations and thereby saves redundant calculation. It is an open question whether lazy evaluation indeed helps attaining the performance of conventional decoders.

Having a formal language for specification also paves the way for tooling. For instance, we can build an IDE for decoder specification that represents the defining expression of a decoder graphically as a workflow diagram (Fig. 6.1). Indeed, the development of such an IDE had been begun under the name of Vanda Studio [26, 23]. This program turns the graphical specification into a Haskell program that uses the Vanda library. However, the focus of Vanda Studio has recently shifted [24] from algebraic specification to the integration of conventional tools and systems such as the Berkeley Parser [155], GIZA [151], or Moses [111].

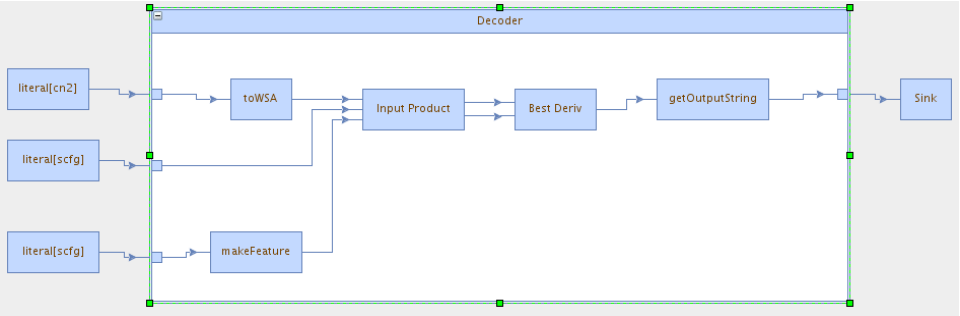


Figure 6.1: Graphical rendering of a decoder in Vanda Studio (taken from [23]).

It should be an interesting task to extend Vanda and Vanda Studio to cover the algebraic framework.

6.2.9 Context of decoder development

A decoder is only one component of an SMT system, and the best decoder is worth nothing without a training procedure and neat parallel corpora. Rule extraction is particularly crucial in this respect because it has to be tailored to the grammar formalism in question. It would be interesting to investigate the merit of the algebraic framework for the SMT system as a whole.

At any rate, if the algebraic framework is to gain the respect of practitioners, it has to be applied in the development of a competitive SMT system.

Bibliography

[1] Anne Abeillé, Yves Schabes, and Aravind K. Joshi. “Using lexicalized tags for machine translation”. In: *Proceedings of the Thirteenth International Conference on Computational Linguistics (COLING ’90)*. Vol. 3. 1990, pp. 1–6 (cit. on p. 9).

[2] Alfred V. Aho and Jeffrey D. Ullman. “Syntax directed translations and the pushdown assembler”. In: *Journal of Computer and System Sciences* 3 (1969), pp. 37–56 (cit. on pp. 19, 84).

[3] Athanasios Alexandrakis and Symeon Bozapalidis. “Weighted grammars and Kleene’s theorem”. In: *Information Processing Letters* 24.1 (1987), pp. 1–4 (cit. on pp. 37, 42, 43).

[4] Cyril Allauzen, Mehryar Mohri, and Brian Roark. “Generalized Algorithms for Constructing Statistical Language Models”. In: *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*. ACL ’03. 2003, pp. 40–47. DOI: 10.3115/1075096.1075102 (cit. on pp. 7, 16).

[5] Cyrill Allauzen and Mehryar Mohri. “Efficient Algorithms for Testing the Twins Property”. In: *J. of Automata, Languages and Combinatorics* 8.2 (2003), pp. 117–144 (cit. on pp. 21, 134, 159, 164).

[6] André Arnold and Max Dauchet. “Bi-transduction de forêts”. In: *Proc. 3rd Int. Coll. Automata, Languages and Programming*. 1976, pp. 74–86 (cit. on pp. 21, 48, 85, 86).

[7] Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998 (cit. on p. 29).

[8] James K. Baker. “Trainable grammars for speech recognition”. In: *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*. Ed. by D. H. Klatt and J. J. Wolf. 1979, pp. 547–550 (cit. on p. 9).

[9] Yehoshua Bar-Hillel, Micha Perles, and Eli Shamir. “On formal properties of simple phrase structure grammars”. In: *Z. Phonetik Sprach. Komm.* 14 (1961), pp. 143–172 (cit. on p. 83).

[10] Jean Berstel and Christophe Reutenauer. “Recognizable formal power series on trees”. In: *Theoret. Comput. Sci.* 18.2 (1982), pp. 115–148 (cit. on p. 43).

[11] Jean Berstel and Christophe Reutenauer. *Rational Series and Their Languages*. Vol. 12. EATCS Monographs on Theoretical Computer Science. Springer, 1988 (cit. on pp. 14–16).

[12] Dan Bikel. "On The Parameter Space of Generative Lexicalized Statistical Parsing Models". PhD thesis. University of Pennsylvania, 2004 (cit. on p. 10).

[13] Phil Blunsom, Trevor Cohn, and Miles Osborne. "A discriminative latent variable model for statistical machine translation". In: *Proceedings ACL-HLT 2008*. 2008, pp. 200–208. URL: http://aclweb.org/anthology-new/P/P08/P08-1024.pdf (cit. on p. 8).

[14] Ondřej Bojar, Christian Buck, Chris Callison-Burch, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Herve Saint-Amand, Radu Soricut, and Lucia Specia, eds. *Proceedings of the Eighth Workshop on Statistical Machine Translation*. ACL, 2013. URL: http://www.aclweb.org/anthology/W13-22 (cit. on p. 167).

[15] Björn Borchardt. "A pumping lemma and decidability problems for recognizable tree series". In: *Acta Cybernet.* 16.4 (2004), pp. 509–544 (cit. on pp. 15, 16, 132, 145, 146).

[16] Björn Borchardt. "The Theory of Recognizable Tree Series". PhD thesis. Technische Universität Dresden, 2005 (cit. on p. 42).

[17] Björn Borchardt, Andreas Maletti, Branimir Šešelja, Andreja Tepavčević, and Heiko Vogler. "Cut sets as recognizable tree languages". In: *Fuzzy Sets and Systems* 157.11 (2006), pp. 1560–1571. DOI: 10.1016/j.fss.2005.11.004 (cit. on p. 170).

[18] Björn Borchardt and Heiko Vogler. "Determinization of finite state weighted tree automata". In: *J. Autom. Lang. Combin.* 8.3 (2003), pp. 417–463 (cit. on p. 132).

[19] Symeon Bozapalidis. "Context-free series on trees". In: *Inform. and Comput.* 169.2 (2001), pp. 186–229 (cit. on p. 18).

[20] Fabienne Braune, Andreas Maletti, Daniel Quernheim, and Nina Seemann. "Shallow local multi bottom-up tree transducers in statistical machine translation". In: *Proc. 51st Annual Meeting of the Association for Computational Linguistics*. Ed. by Pascale Fung and Massimo Poesio. 2013, pp. 811–821 (cit. on p. 10).

[21] Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. "The mathematics of statistical machine translation: parameter estimation". In: *Comp. Ling.* 19 (2 1993), pp. 263–311 (cit. on p. 3).

[22] Adam L. Buchsbaum, Raffaele Giancarlo, and Jeffery R. Westbrook. "On the Determinization of Weighted Finite Automata". In: *SIAM J. Comput.* 30.5 (2000), pp. 1502–1531. DOI: 10.1137/S0097539798346676 (cit. on pp. 131, 132).

[23] Matthias Büchse. "As Easy As Vanda, Two, Three: Components for Machine Translation Based on Formal Grammars". In: *Proceedings 22nd Theorietag "Automata and Formal Languages", Prague, Czech Republic, October 3–5, 2012*. Ed. by František Mráz. Charles University in Prague, Faculty of Mathematics and Physics. 2012, pp. 41–44 (cit. on pp. 180, 181).

[24] Matthias Büchse. "Vanda Studio – Instructive, Rapid Experiment Development". Unpublished manuscript. 2013. URL: http://www.inf.tu-dresden.de/content/institutes/thi/gdp/research/vanda-demo.pdf (cit. on p. 180).

[25] Matthias Büchse, David Chiang, Liang Huang, and Michael Pust. "Some remarks on cube pruning". Unpublished manuscript. 2011 (cit. on p. 177).

[26] Matthias Büchse, Toni Dietze, Johannes Osterholzer, Anja Fischer, and Linda Leuschner. "Vanda – A Statistical Machine Translation Toolkit". In: *Proceedings 6th International Workshop "Weighted Automata: Theory and Applications" (WATA 2012)*. Ed. by Manfred Droste and Heiko Vogler. Online proceedings. 2012, pp. 36–38. URL: http://wwwtcs.inf.tu-dresden.de/wata2012/Proceedings.pdf (cit. on p. 180).

[27] Matthias Büchse and Anja Fischer. "Deciding the Twins Property for Weighted Tree Automata over Extremal Semifields". In: *Proceedings of the Workshop on Applications of Tree Automata Techniques in Natural Language Processing*. 2012, pp. 11–20. URL: http://www.aclweb.org/anthology/W/W12/W12-0802 (cit. on pp. 131, 165).

[28] Matthias Büchse, Daniel Geisler, Torsten Stüber, and Heiko Vogler. "n-Best Parsing Revisited". In: *Proceedings of the 2010 Workshop on Applications of Tree Automata in Natural Language Processing, ACL 2010*. Uppsala, Sweden, 16 July 2010. 2010, pp. 46–54 (cit. on pp. 8, 15, 20, 175).

[29] Matthias Büchse, Alexander Koller, and Heiko Vogler. "Generic binarization for parsing and translation". In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2013, pp. 145–154. URL: http://www.aclweb.org/anthology/P13-1015 (cit. on pp. 83, 122, 127).

[30] Matthias Büchse, Andreas Maletti, and Heiko Vogler. "Unidirectional derivation semantics for synchronous tree-adjoining grammars". In: *Proc. 16th Int. Conf. Developments in Language Theory*. Ed. by Hsu-Chun Yen and Oscar H. Ibarra. Vol. 7410. LNCS. 2012, pp. 368–379. DOI: 10.1007/978-3-642-31653-1_33 (cit. on pp. 81, 124).

[31] Matthias Büchse, Jonathan May, and Heiko Vogler. "Determinization of weighted tree automata using factorizations". In: *Journal of Automata, Languages and Combinatorics* 15.3/4 (2010) (cit. on pp. 131, 165).

[32] Matthias Büchse, Mark-Jan Nederhof, and Heiko Vogler. "Tree Parsing with Synchronous Tree-Adjoining Grammars". In: *Proceedings of the 12th International Conference on Parsing Technologies*. 2011, pp. 14–25. URL: http://www.aclweb.org/anthology/W11-2903 (cit. on pp. 45, 81).

[33] Matthias Büchse, Heiko Vogler, and Mark-Jan Nederhof. "Tree parsing for tree-adjoining machine translation". In: *Journal of Logic and Computation* 24.2 (2014). first published online December 6, 2012, pp. 351–373 (cit. on pp. 45, 81, 82).

[34] Mathieu Caralp, Emmanuel Filiot, Pierre-Alain Reynier, Frédéric Servais, and Jean-Marc Talbot. "Expressiveness of Visibly Pushdown Transducers". In: *Proceedings TTATT 2013*. 2013, pp. 17–26. URL: http://arxiv.org/abs/1311.5571 (cit. on p. 122).

[35] Francisco Casacuberta and Colin de la Higuera. "Computational Complexity of Problems on Probabilistic Grammars and Transducers". In: *ICGI*. Ed. by Arlindo L. Oliveira. Vol. 1891. Lecture Notes in Computer Science. 2000, pp. 15–24 (cit. on p. 8).

[36] Yin-Wen Chang and Michael Collins. "Exact Decoding of Phrase-Based Translation Models through Lagrangian Relaxation". In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. 2011, pp. 26–37. URL: http://www.aclweb.org/anthology/D11-1003 (cit. on p. 4).

[37] Eugene Charniak and Mark Johnson. "Coarse-to-Fine n-Best Parsing and MaxEnt Discriminative Reranking". In: *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*. 2005, pp. 173–180. URL: http://www.aclweb.org/anthology/P05-1022 (cit. on p. 10).

[38] David Chiang. "A hierarchical phrase-based model for statistical machine translation". In: *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. 2005, pp. 263–270 (cit. on p. 4).

[39] David Chiang. "Hierarchical Phrase-Based Translation". In: *Comp. Ling.* 33.2 (2007), pp. 201–228 (cit. on pp. 4, 8, 14, 19, 22, 82, 84, 177).

[40] David Chiang. "Learning to Translate with Source and Target Syntax". In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. 2010, pp. 1443–1452. URL: http://www.aclweb.org/anthology/P10-1146 (cit. on p. 10).

[41] David Chiang, Kevin Knight, and Wei Wang. "11,001 New Features for Statistical Machine Translation". In: *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. 2009, pp. 218–226. URL: http://www.aclweb.org/anthology/N/N09/N09-1025 (cit. on p. 10).

[42] David Chiang, Adam Lopez, Nitin Madnani, Christof Monz, Philip Resnik, and Michael Subotin. "The Hiero machine translation system: extensions, evaluation, and analysis". In: *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. 2005, pp. 779–786 (cit. on p. 4).

[43] David Chiang, Yuval Marton, and Philip Resnik. "Online large-margin training of syntactic and structural translation features". In: *Proceedings EMNLP 2008*. 2008. URL: http://www.isi.edu/~chiang/papers/mira.pdf (cit. on p. 174).

[44] Christian Choffrut. "Une Caracterisation des Fonctions Sequentielles et des Fonctions Sous-Sequentielles en tant que Relations Rationnelles." In: *Theoret. Comput. Sci.* 5.3 (1977), pp. 325–337 (cit. on pp. 21, 133).

[45] Noam Chomsky. "On certain formal properties of grammars". In: *Inform. Control* 2 (1959), pp. 137–167 (cit. on pp. 19, 84).

[46] Noam Chomsky and Marcel-Paul Schützenberger. "The algebraic theory of context-free languages". In: *Computer Programming and Formal Systems*. Ed. by Paul Braffort and David Hirschberg. 1963, pp. 118–161 (cit. on p. 14).

[47] Michael Collins. "Head-driven Statistical Models for Natural Language Parsing". PhD thesis. University of Pennsylvania, 1999 (cit. on p. 10).

[48] Bruno Courcelle. "Attribute grammars: definitions, analysis of dependencies, proof methods". In: *Methods and tools for compiler construction*. Ed. by Bernard Lorho. 1984, pp. 81–102 (cit. on p. 70).

[49] Bruno Courcelle and Paul Franchi-Zannettacci. “Attribute grammars and recursive program schemes I”. In: *Theoret. Comput. Sci.* 17.2 (1982), pp. 163–191 (cit. on p. 124).

[50] Steve DeNeefe. “Tree-Adjoining Machine Translation”. PhD thesis. University of Southern California, 2011 (cit. on pp. 10, 18).

[51] Steve DeNeefe and Kevin Knight. “Synchronous Tree-Adjoining Machine Translation”. In: *EMNLP ’09: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. 2009, pp. 727–736 (cit. on p. 10).

[52] John DeNero, Mohit Bansal, Adam Pauls, and Dan Klein. “Efficient Parsing for Transducer Grammars”. In: *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. 2009, pp. 227–235. URL: http://www.aclweb.org/anthology/N/N09/N09-1026 (cit. on p. 82).

[53] John DeNero, David Chiang, and Kevin Knight. “Fast Consensus Decoding over Translation Forests”. In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. 2009, pp. 567–575. URL: http://www.aclweb.org/anthology/P/P09/P09-1064 (cit. on p. 22).

[54] John DeNero, Adam Pauls, and Dan Klein. “Asynchronous Binarization for Synchronous Grammars”. In: *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*. 2009, pp. 141–144. URL: http://www.aclweb.org/anthology/P/P09/P09-2036 (cit. on p. 127).

[55] Edsger W. Dijkstra. “A note on two problems in connection with graphs”. In: *Numerische Mathematik* 1 (1959), pp. 269–271 (cit. on p. 9).

[56] Manfred Droste, Doreen Götze, Steffen Märcker, and Ingmar Meinecke. “Weighted Tree Automata over Valuation Monoids and Their Characterization by Weighted Logics”. In: *Algebraic Foundations in Computer Science*. Ed. by Werner Kuich and George Rahonis. Vol. 7020. Lecture Notes in Computer Science. 2011, pp. 30–55. DOI: 10.1007/978-3-642-24897-9_2 (cit. on p. 43).

[57] Manfred Droste, Werner Kuich, and Heiko Vogler, eds. *Handbook of Weighted Automata*. EATCS Monographs in Theoretical Computer Science. Springer, 2009 (cit. on p. 12).

[58] Manfred Droste and Heiko Vogler. "Weighted logics for unranked tree automata". In: *Theory of Computing Systems* 48.1 (2011). first published online June 29, 2009, pp. 23–47. DOI: 10.1007/s00224-009-9224-4 (cit. on p. 44).

[59] Chris Dyer, Adam Lopez, Juri Ganitkevitch, Jonathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. "cdec: A Decoder, Alignment, and Learning Framework for Finite-State and Context-Free Translation Models". In: *Proceedings of the ACL 2010 System Demonstrations*. 2010, pp. 7–12. URL: http://www.aclweb.org/anthology/P10-4002 (cit. on p. 22).

[60] Jay Earley. "An Efficient Context-Free Parsing Algorithm". In: *Communications of the ACM* 13.2 (1970), pp. 94–102 (cit. on pp. 47, 61).

[61] Samuel Eilenberg. *Automata, Languages, and Machines – Volume A*. Vol. 59. Pure and Applied Mathematics. Academic Press, 1974 (cit. on p. 12).

[62] Jason Eisner. "Learning non-isomorphic tree mappings for machine translation". In: *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 2*. ACL '03. 2003, pp. 205–208 (cit. on pp. 9, 82).

[63] Joost Engelfriet. "Attribute grammars: attribute evaluation methods". In: *Methods and tools for compiler construction*. Ed. by Bernard Lorho. 1984, pp. 103–138 (cit. on p. 70).

[64] Joost Engelfriet and Linda Heyker. "The string generating power of context-free hypergraph grammars". In: *Journal of Computer and System Sciences* 43.2 (1991), pp. 328–360. URL: http://www.sciencedirect.com/science/article/pii/002200009190018Z (cit. on p. 123).

[65] Joost Engelfriet, Eric Lilin, and Andreas Maletti. "Extended Multi Bottom-up Tree Transducers". In: *Proc. 12th Int. Conf. Developments in Language Theory*. Ed. by Masami Ito and F. M. Toyama. Vol. 5257. LNCS. 2008, pp. 289–300 (cit. on p. 10).

[66] Joost Engelfriet, Eric Lilin, and Andreas Maletti. "Extended Multi Bottom-up Tree Transducers – Composition and Decomposition". In: *Acta Inf.* 46.8 (2009), pp. 561–590 (cit. on p. 46).

[67] Joost Engelfriet and Erik M. Schmidt. "IO and OI I". In: *J. Comput. System Sci.* 15.3 (1977), pp. 328–353 (cit. on pp. 81, 119).

[68] Joost Engelfriet and Erik M. Schmidt. "IO and OI II". In: *J. Comput. System Sci.* 16.1 (1978), pp. 67–99 (cit. on p. 81).

[69] Joost Engelfriet and Heiko Vogler. "Macro Tree Transducers". In: *Journal of Computer and System Sciences* 31 (1985), pp. 71–146 (cit. on p. 124).

[70] David Eppstein. "Finding the k Shortest Paths". In: *SIAM Journal on Computing* 28.2 (1998), pp. 652–673. DOI: 10.1137/S0097539795290477 (cit. on pp. 9, 174).

[71] Zoltán Ésik and Werner Kuich. "Formal Tree Series". In: *J. Autom. Lang. Combin.* 8.2 (2003), pp. 219–285 (cit. on pp. 15, 18, 36, 43).

[72] Javier Esparza, Stefan Kiefer, and Michael Luttenberger. "Derivation Tree Analysis for Accelerated Fixed-Point Computation". In: *Theoretical Computer Science* 412.28 (2011), pp. 3226–3241 (cit. on p. 42).

[73] Zoltán Fülöp, Andreas Maletti, and Heiko Vogler. "A Kleene theorem for weighted tree automata over distributive multioperator monoids". In: *Theory Comput. Syst.* 44 (2009), pp. 455–499 (cit. on p. 43).

[74] Zoltán Fülöp, Andreas Maletti, and Heiko Vogler. "Preservation of Recognizability for Synchronous Tree Substitution Grammars". In: *Proceedings of the 2010 Workshop on Applications of Tree Automata in Natural Language Processing, ACL 2010*. Uppsala, Sweden, 16 July 2010. 2010, pp. 1–9. URL: http://www.aclweb.org/anthology/W/W10/W10-2501.pdf (cit. on pp. 14, 51).

[75] Zoltán Fülöp, Andreas Maletti, and Heiko Vogler. "Weighted Extended Tree Transducers". In: *Fundam. Inform.* 111.2 (2011), pp. 163–202 (cit. on pp. 15, 45).

[76] Zoltán Fülöp, Torsten Stüber, and Heiko Vogler. "A Büchi-like theorem for weighted tree automata over multioperator monoids". In: *Theory of Computing Systems* 50.2 (2012). first published online October 28, 2010, pp. 241–278. DOI: 10.1007/s00224-010-9296-1 (cit. on p. 43).

[77] Zoltán Fülöp and Heiko Vogler. "Weighted tree automata and tree transducers". In: *Handbook of Weighted Automata*. Ed. by Manfred Droste, Werner Kuich, and Heiko Vogler. EATCS Monographs in Theoretical Computer Science. 2009. Chap. 9 (cit. on pp. 40, 43).

[78] Michael Galley. *GHKM Rule Extractor*. http://www-nlp.stanford.edu/~mgalley/software/stanford-ghkm-latest.tar.gz, retrieved on March 28, 2012. 2010 (cit. on p. 123).

[79] Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. “What’s in a translation rule?” In: *HLT-NAACL 2004: Main Proceedings*. Ed. by Susan Dumais, Daniel Marcu, and Salim Roukos. 2004, pp. 273–280 (cit. on pp. 9, 10, 120–122).

[80] Ferenc Gécseg and Magnus Steinby. “Tree Languages”. In: *Handbook of Formal Languages*. Ed. by Grzegorz Rozenberg and Arto Salomaa. Vol. 3. 1997. Chap. 1, pp. 1–68 (cit. on pp. 85, 107, 113, 128, 132).

[81] Andrea Gesmundo and James Henderson. “Faster Cube Pruning”. In: *Proceedings of the seventh International Workshop on Spoken Language Translation (IWSLT)*. Ed. by Marcello Federico, Ian Lane, Michael Paul, and François Yvon. 2010, pp. 267–274. URL: http://www.mt-archive.info/IWSLT-2010-Gesmundo.pdf (cit. on p. 177).

[82] Andrea Gesmundo, Giorgio Satta, and James Henderson. “Heuristic Cube Pruning in Linear Time”. In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 2012, pp. 296–300. URL: http://www.aclweb.org/anthology/P12-2058 (cit. on p. 177).

[83] Daniel Gildea. “Optimal Parsing Strategies for Linear Context-Free Rewriting Systems”. In: *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. 2010, pp. 769–776. URL: http://www.aclweb.org/anthology/N10-1118 (cit. on pp. 17, 130).

[84] Kevin Gimpel and Noah A. Smith. “Structured Ramp Loss Minimization for Machine Translation”. In: *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2012, pp. 221–231. URL: http://www.aclweb.org/anthology/N12-1023 (cit. on pp. 8, 174).

[85] Joseph A. Goguen, Jim W. Thatcher, and Eric G. Wagner. “An Initial Algebra Approach to the Specification, Correctness and Implementation of Abstract Data Types”. In: *Current Trends in Programming Methodology*. Ed. by Raymond T. Yeh. Vol. IV: Data Structuring. also IBM Research Report RC-6487 (1976). 1978 (cit. on pp. 21, 179).

[86] Joseph A. Goguen, Jim W. Thatcher, Eric G. Wagner, and Jesse B. Wright. “Initial algebra semantics and continuous algebras”. In: *J. ACM* 24 (1977), pp. 68–95 (cit. on pp. 22, 40, 85, 179).

[87] Jonathan S. Golan. *Semirings and their Applications*. Kluwer Academic, 1999 (cit. on pp. 12, 33).

[88] Carlos Gómez-Rodríguez, Marco Kuhlmann, Giorgio Satta, and David Weir. "Optimal Reduction of Rule Length in Linear Context-Free Rewriting Systems". In: *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. 2009, pp. 539–547. URL: http://www.aclweb.org/anthology/N/N09/N09-1061 (cit. on p. 130).

[89] Joshua Goodman. "Semiring Parsing". In: *Comp. Ling.* 25.4 (1999), pp. 573–605 (cit. on pp. 8, 14, 24, 47, 68).

[90] Jonathan Graehl, Kevin Knight, and Jonathan May. "Training Tree Transducers". In: *Comp. Ling.* 34.3 (2008), pp. 391–427 (cit. on pp. 9, 120).

[91] Udo Hebisch and Hanns Joachim Weinert. *Semirings: Algebraic Theory and Applications in Computer Science*. Vol. 5. Series in Algebra. World Scientific, 1998 (cit. on pp. 12, 33).

[92] Mark Hopkins and Jonathan May. "Tuning as Ranking". In: *Proceedings EMNLP 2011*. 2011 (cit. on p. 174).

[93] Liang Huang. "Forest-based Algorithms in Natural-Language Processing". PhD thesis. University of Pennsylvania, 2008. URL: http://www.cis.upenn.edu/~lhuang3/Dissertation.pdf (cit. on pp. 12, 16, 176).

[94] Liang Huang and David Chiang. "Better k-best parsing". In: *Parsing '05: Proceedings of the Ninth International Workshop on Parsing Technology*. 2005, pp. 53–64. URL: http://www.cis.upenn.edu/~lhuang3/huang-iwpt-correct.pdf (cit. on pp. 8, 15, 20, 174, 175).

[95] Liang Huang, Kevin Knight, and Aravind Joshi. "A syntax-directed translator with extended domain of locality". In: *Proceedings of the Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*. CHSLP '06. 2006, pp. 1–8 (cit. on p. 120).

[96] Liang Huang, Kevin Knight, and Aravind Joshi. "Statistical syntax-directed translation with extended domain of locality". In: *Proceedings AMTA 2006*. 2006, pp. 66–73. URL: http://www.cis.upenn.edu/~lhuang3/amta06-sdtedl.pdf (cit. on pp. 9, 12).

[97] Liang Huang, Hao Zhang, Daniel Gildea, and Kevin Knight. "Binarization of Synchronous Context-Free Grammars". In: *Comp. Ling.* 35.4 (2009), pp. 559–595. URL: http://www.aclweb.org/anthology/J/J09/J09-4009.pdf (cit. on pp. 19, 84, 120, 128).

[98] Aravind K. Joshi and Yves Schabes. "Tree-Adjoining Grammars". In: *Handbook of Formal Languages*. Ed. by Grzegorz Rozenberg and Arto Salomaa. Vol. 3. 1997 (cit. on pp. 49, 52).

[99] Daniel Jurafsky and James H. Martin. *Speech and Language Processing – An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Second edition. Prentice-Hall, 2009 (cit. on pp. 7, 10, 47).

[100] Miriam Kaeshammer. "Synchronous Linear Context-Free Rewriting Systems for Machine Translation". In: *Proceedings of the Seventh Workshop on Syntax, Semantics and Structure in Statistical Translation*. 2013, pp. 68–77. URL: http://www.aclweb.org/anthology/W13-0808 (cit. on pp. 17, 23).

[101] Laura Kallmeyer, Wolfgang Maier, and Giorgio Satta. "Synchronous Rewriting in Treebanks". In: *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*. 2009, pp. 69–72. URL: http://www.aclweb.org/anthology/W09-3810 (cit. on p. 17).

[102] Stephan Kepser and James Rogers. "The Equivalence of Tree Adjoining Grammars and Monadic Linear Context-Free Tree Grammars". In: *The Mathematics of Language*. Ed. by Christian Ebert, Gerhard Jäger, and Jens Michaelis. Vol. 6149. Lecture Notes in Computer Science. 2010, pp. 129–144. DOI: 10.1007/978-3-642-14322-9_11 (cit. on pp. 47, 81).

[103] Stephan Kepser and Jim Rogers. "The Equivalence of Tree Adjoining Grammars and Monadic Linear Context-free Tree Grammars". In: *Journal of Logic, Language and Information* 20.3 (2011), pp. 361–384 (cit. on p. 18).

[104] Daniel Kirsten. "Decidability, Undecidability, and PSPACE-Completeness of the Twins Property in the Tropical Semiring". In: *Theoretical Computer Science* 420 (2012), pp. 56–63 (cit. on pp. 21, 134, 165).

[105] Daniel Kirsten and Ina Mäurer. "On the determinization of weighted automata". In: *J. Autom. Lang. Comb.* 10 (2005), pp. 287–312 (cit. on pp. 21, 132–134, 137, 141, 143, 146, 148, 149, 151, 164).

[106] Kevin Knight. "Automating knowledge acquisition for machine translation". In: *AI Magazine* 18.4 (1997), pp. 81–96. URL: http://www.aaai.org/ojs/index.php/aimagazine/article/viewArticle/1323 (cit. on p. 3).

[107] Donald E. Knuth. "Semantics of context–free languages". In: *Math. Systems Theory* 2 (1968), pp. 127–145 (cit. on p. 70).

[108] Donald E. Knuth. “A Generalization of Dijkstra’s Algorithm”. In: *Inform. Process. Lett.* 6.1 (1977), pp. 1–5 (cit. on pp. 8, 15, 20).

[109] Philipp Koehn. “Europarl: A Parallel Corpus for Statistical Machine Translation”. In: *Proceedings of MT Summit X*. 2005, pp. 79–86 (cit. on pp. 3, 123).

[110] Philipp Koehn. *Statistical Machine Translation*. Cambridge University Press, 2010 (cit. on p. 3).

[111] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. “Moses: open source toolkit for statistical machine translation”. In: *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*. ACL ’07. 2007, pp. 177–180. URL: http://www.aclweb.org/anthology/P07-2045 (cit. on pp. 22, 180).

[112] Alexander Koller and Marco Kuhlmann. “A Generalized View on Parsing and Translation”. In: *Proceedings of the 12th International Conference on Parsing Technologies*. 2011, pp. 2–13. URL: http://www.aclweb.org/anthology/W11-2902 (cit. on pp. 22, 46, 84, 86, 127).

[113] Alexander Koller and Marco Kuhlmann. “Decomposing TAG Algorithms Using Simple Algebraizations”. In: *Proceedings of the 11th Workshop on TAG and related formalisms (TAG+)*. 2012, pp. 135–143 (cit. on pp. 23, 46, 47, 119, 127).

[114] Marco Kuhlmann. “Mildly Non-Projective Dependency Grammar”. In: *Computational Linguistics* 39.2 (2013), pp. 355–387 (cit. on p. 23).

[115] Werner Kuich. “Semirings and Formal Power Series: Their Relevance to Formal Languages and Automata”. In: *Handbook of Formal Languages*. Ed. by Grzegorz Rozenberg and Arto Salomaa. Vol. 1. 1997. Chap. 9, pp. 609–677 (cit. on pp. 14, 28, 29, 35, 36).

[116] Werner Kuich. “Formal power series over trees”. In: *3rd International Conference on Developments in Language Theory, DLT 1997, Thessaloniki, Greece, Proceedings*. Ed. by Symeon Bozapalidis. 1998, pp. 61–101 (cit. on p. 43).

[117] Werner Kuich. “Linear systems of equations and automata on distributive multioperator monoids”. In: *Contributions to General Algebra 12 - Proceedings of the 58th Workshop on General Algebra “58. Arbeitstagung Allgemeine Algebra”, Vienna University of Technology. June 3-6, 1999*. 1999, pp. 1–10 (cit. on p. 43).

[118] Karim Lari and Steve J. Young. "The estimation of stochastic context-free grammars using the Inside-Outside algorithm". In: *Computer Speech and Language* 4 (1990), pp. 35–56 (cit. on p. 9).

[119] Philip M. Lewis and Richard E. Stearns. "Syntax directed transduction". In: *Foundations of Computer Science, IEEE Annual Symposium on* (1966), pp. 21–35 (cit. on pp. 5, 84).

[120] Zhifei Li, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren N. G. Thornton, Jonathan Weese, and Omar F. Zaidan. "Joshua: an open source toolkit for parsing-based machine translation". In: *Proceedings of the Fourth Workshop on Statistical Machine Translation*. StatMT '09. 2009, pp. 135–139. URL: http://dl.acm.org/citation.cfm?id=1626431.1626459 (cit. on p. 22).

[121] Zhifei Li, Jason Eisner, and Sanjeev Khudanpur. "Variational decoding for statistical machine translation". In: *ACL-IJCNLP '09: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2*. 2009, pp. 593–601 (cit. on pp. 8, 9, 175).

[122] Adam Lopez. "Statistical machine translation". In: *ACM Comput. Surv.* 40.3 (2008), pp. 1–49 (cit. on p. 3).

[123] Bernard Lorho, ed. *Methods and tools for compiler construction*. Cambridge University Press, 1984.

[124] Bernd Mahr. "Iteration and summability in semirings". In: *Annals of Discrete Mathematics* 19 (1984), pp. 229–256 (cit. on p. 133).

[125] Andreas Maletti. "Relating Tree Series Transducers and Weighted Tree Automata". In: *Int. J. Found. Comput. Sci.* 16.4 (2005), pp. 723–741 (cit. on p. 43).

[126] Andreas Maletti. "Minimizing Deterministic Weighted Tree Automata". In: *Inform. and Comput.* 207.11 (2009), pp. 1284–1299 (cit. on p. 131).

[127] Andreas Maletti. "A Tree Transducer Model for Synchronous Tree-Adjoining Grammars". In: *Proc. 48th Annual Meeting Association for Computational Linguistics*. Ed. by Jan Hajič, Sandra Carberry, Stephen Clark, and Joakim Nivre. 2010, pp. 1067–1076 (cit. on pp. 46, 47, 119).

[128] Andreas Maletti. "Input and Output Products for Weighted Extended Top-down Tree Transducers". In: *Proc. 14th Int. Conf. Developments in Language Theory*. Ed. by Yuan Gao, Hanlin Lu, Shinnosuke Seki, and Sheng Yu. Vol. 6224. LNCS. 2010, pp. 316–327 (cit. on pp. 15, 45).

[129] Andreas Maletti. "Why Synchronous Tree Substitution Grammars?" In: *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. 2010, pp. 876–884. URL: http://www.aclweb.org/anthology/N10-1130 (cit. on p. 14).

[130] Andreas Maletti. "Synchronous Forest Substitution Grammars". In: *Proc. 5th Int. Conf. Algebraic Informatics*. Ed. by Traian Muntean, Dimitris Poulakis, and Robert Rolland. Vol. 8080. LNCS. 2013, pp. 235–246 (cit. on p. 9).

[131] Andreas Maletti and Joost Engelfriet. "Strong Lexicalization of Tree Adjoining Grammars". In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2012, pp. 506–515. URL: http://www.aclweb.org/anthology/P12-1053 (cit. on p. 47).

[132] Andreas Maletti and Giorgio Satta. "Parsing Algorithms based on Tree Automata". In: *Proc. 11th Int. Conf. Parsing Technologies*. 2009, pp. 1–12 (cit. on pp. 15, 82).

[133] Jonathan May. "Weighted Tree Automata and Transducers for Syntactic Natural Language Processing". PhD thesis. University of Southern California, 2010 (cit. on p. 22).

[134] Jonathan May and Kevin Knight. "A better N-best list: practical determinization of weighted finite tree automata". In: *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*. 2006, pp. 351–358 (cit. on pp. 15, 19–21, 132, 134, 136, 146, 175).

[135] Jonathan May and Kevin Knight. "Tiburon: A Weighted Tree Automata Toolkit". In: *Proceedings of the 11th International Conference of Implementation and Application of Automata, CIAA 2006*. Ed. by Oscar H. Ibarra and Hsu-Chun Yen. Vol. 4094. Lecture Notes in Computer Science. 2006, pp. 102–113 (cit. on p. 22).

[136] David McAllester. "On the complexity analysis of static analyses". In: *J. ACM* 49 (4 2002), pp. 512–537 (cit. on p. 70).

[137] Haitao Mi, Liang Huang, and Qun Liu. "Forest-Based Translation". In: *Proceedings of ACL-08: HLT*. 2008, pp. 192–199. URL: http://www.aclweb.org/anthology/P/P08/P08-1023 (cit. on pp. 12, 16).

[138] Mehryar Mohri. "Finite-State Transducers in Language and Speech Processing". In: *Comp. Ling.* 23.2 (1997), pp. 1–42 (cit. on pp. 132–134, 136, 164).

[139] Mehryar Mohri. "Weighted automata algorithms". In: *Handbook of Weighted Automata*. Ed. by Manfred Droste, Werner Kuich, and Heiko Vogler. EATCS Monographs in Theoretical Computer Science. 2009. Chap. 6, pp. 213–254 (cit. on p. 132).

[140] Makoto Murata. "Extended Path Expressions of XML". In: *Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. PODS '01. 2001, pp. 126–137. DOI: 10.1145/375551.375569 (cit. on p. 116).

[141] Mark-Jan Nederhof. "The computational complexity of the correct-prefix property for TAGs". In: *Computational Linguistics* 25.3 (1999), pp. 345–360 (cit. on p. 82).

[142] Mark-Jan Nederhof. "Weighted deductive parsing and Knuth's algorithm". In: *Comp. Ling.* 29(1) (2003), pp. 135–143 (cit. on pp. 8, 68).

[143] Mark-Jan Nederhof. "A General Technique to Train Language Models on Language Models". In: *Computational Linguistics* 31.2 (2005), pp. 173–185 (cit. on p. 9).

[144] Mark-Jan Nederhof. "Weighted parsing of trees". In: *Proceedings of the 11th International Conference on Parsing Technologies*. 2009, pp. 13–24 (cit. on p. 46).

[145] Mark-Jan Nederhof and Giorgio Satta. "Probabilistic Parsing". In: *New Developments in Formal Languages and Applications*. Ed. by G. Bel-Enguix, M. Dolores Jiménez-López, and C. Martín-Vide. Vol. 113. Studies in Computational Intelligence. 2008, pp. 229–258 (cit. on pp. 24, 44, 45, 47).

[146] Mark-Jan Nederhof and Heiko Vogler. "Synchronous Context-Free Tree Grammars". In: *Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+11)*. 2012, pp. 55–63. URL: http://www.aclweb.org/anthology-new/W/W12/W12-4607 (cit. on pp. 46, 47, 81).

[147] Rebecca Nesson. "Synchronous and Multicomponent Tree-Adjoining Grammars: Complexity, Algorithms and Linguistic Applications". PhD thesis. Harvard University, 2009 (cit. on pp. 10, 18, 19, 84, 130).

[148] Rebecca Nesson, Stuart M. Shieber, and Alexander Rush. *Induction of Probabilistic Synchronous Tree-Insertion Grammars*. Tech. rep. Computer Science Group, Harvard University, Cambridge, Massachusetts, 2005 (cit. on p. 10).

[149] Rebecca Nesson, Stuart M. Shieber, and Alexander Rush. "Induction of Probabilistic Synchronous Tree-Insertion Grammars for Machine Translation". In: *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas (AMTA 2006)*. 2006, pp. 128–137 (cit. on pp. 10, 82).

[150] Franz Josef Och. "Minimum Error Rate Training in Statistical Machine Translation". In: *Proceedings ACL 2003*. 2003, pp. 160–167 (cit. on pp. 8, 174).

[151] Franz Josef Och and Hermann Ney. "Improved statistical alignment models". In: *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*. ACL '00. 2000, pp. 440–447. DOI: 10.3115/1075218.1075274 (cit. on pp. 8, 180).

[152] Franz Josef Och and Hermann Ney. "Discriminative Training and Maximum Entropy Models for Statistical Machine Translation". In: *Proceedings ACL 2002*. 2002, pp. 295–302 (cit. on p. 6).

[153] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. "BLEU: a method for automatic evaluation of machine translation". In: *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. 2002, pp. 311–318 (cit. on p. 8).

[154] Ion Petre and Arto Salomaa. "Algebraic systems and pushdown automata". In: *Handbook of Weighted Automata*. Ed. by Manfred Droste, Werner Kuich, and Heiko Vogler. EATCS Monographs in Theoretical Computer Science. 2009. Chap. 7, pp. 257–289 (cit. on p. 14).

[155] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. "Learning Accurate, Compact, and Interpretable Tree Annotation". In: *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*. 2006, pp. 433–440. URL: http://www.aclweb.org/anthology/P/P06/P06-1055 (cit. on pp. 10, 21, 44, 180).

[156] Slav Petrov and Dan Klein. "Improved Inference for Unlexicalized Parsing". In: *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*. 2007, pp. 404–411. URL: http://www.aclweb.org/anthology/N/N07/N07-1051 (cit. on pp. 10, 21).

[157] Detlef Prescher. "Inside-Outside Estimation Meets Dynamic EM". In: *Proceedings of the 7th International Workshop on Parsing Technologies (IWPT-01), October 17-19*. 2001, pp. 241–244 (cit. on p. 9).

[158] Michael O. Rabin and Dana Scott. “Finite automata and their decision problems”. In: *IBM J. Res.* 3 (1959), pp. 115–125 (cit. on p. 132).

[159] Owen Rambow and Giorgio Satta. “Independent parallelism in finite copying parallel rewriting systems”. In: *Theoretical Computer Science* 223.1–2 (1999), pp. 87–120 (cit. on p. 84).

[160] William C. Rounds. “Mappings and Grammars on Trees”. In: *Math. Systems Theory* 4.3 (1970), pp. 257–287 (cit. on p. 81).

[161] William C. Rounds. “Tree-oriented Proofs of Some Theorems on Context-free and Indexed Languages”. In: *Proceedings of the Second Annual ACM Symposium on Theory of Computing*. STOC ’70. 1970, pp. 109–116. DOI: 10.1145/800161.805156 (cit. on p. 46).

[162] Grzegorz Rozenberg and Arto Salomaa, eds. *Handbook of Formal Languages*. Vol. 1. Springer, 1997.

[163] Grzegorz Rozenberg and Arto Salomaa, eds. *Handbook of Formal Languages*. Vol. 3. Springer, 1997.

[164] Alexander M. Rush and Michael Collins. “Exact Decoding of Syntactic Translation Models through Lagrangian Relaxation”. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Techologies*. 2011, pp. 72–82. URL: http://www.aclweb.org/anthology/P11-1008 (cit. on pp. 4, 8).

[165] Jacques Sakarovitch. “Rational and recognisable power series”. In: *Handbook of Weighted Automata*. Ed. by Manfred Droste, Werner Kuich, and Heiko Vogler. EATCS Monographs in Theoretical Computer Science. 2009. Chap. 4 (cit. on pp. 14, 16).

[166] Arto Salomaa and Matti Soittola. *Automata-Theoretic Aspects of Formal Power Series*. Texts and Monographs in Computer Science. Springer, 1978 (cit. on p. 14).

[167] Yves Schabes and Richard C. Waters. “Tree insertion grammars: a cubic-time, parsable formalism that lexicalizes context-free grammar without changing the trees produced”. In: *Comput. Linguist.* 21 (1994), pp. 479–513 (cit. on p. 10).

[168] Marcel-Paul Schützenberger. “On the definition of a family of automata”. In: *Information and Control* 4 (1961), pp. 245–270 (cit. on pp. 15, 16).

[169] Hiroyuki Seki and Yuki Kato. “On the Generative Power of Multiple Context-Free Grammars and Macro Grammars”. In: *IEICE - Trans. Inf. Syst.* E91-D.2 (2008), pp. 209–221. DOI: 10.1093/ietisy/e91-d.2.209 (cit. on p. 116).

[170] Stuart M. Shieber. “Probabilistic Synchronous Tree-Adjoining Grammars for Machine Translation: The Argument from Bilingual Dictionaries”. In: *Proceedings of the Workshop on Syntax and Structure in Statistical Translation*. Ed. by Dekai Wu and David Chiang. 2007 (cit. on p. 17).

[171] Stuart M. Shieber and Yves Schabes. “Synchronous Tree-Adjoining Grammars”. In: *Proceedings of the 13th International Conference on Computational Linguistics (COLING '90)*. Vol. 3. 1990, pp. 253–258 (cit. on pp. 9, 84).

[172] Khalil Sima'an. “Computational complexity of probabilistic disambiguation by means of tree-grammars”. In: *Proceedings of the 16th Conference on Computational Linguistics - Volume 2*. 1996, pp. 1175–1180 (cit. on p. 8).

[173] Noah A. Smith. *Linguistic Structure Prediction*. Synthesis Lectures on Human Language Technologies. Morgan and Claypool, 2011 (cit. on pp. 3, 6).

[174] Xinying Song, Shilin Ding, and Chin-Yew Lin. “Better Binarization for the CKY Parsing”. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. EMNLP '08. 2008, pp. 167–176. URL: http://dl.acm.org/citation.cfm?id=1613715.1613739 (cit. on p. 128).

[175] Torsten Stüber, Heiko Vogler, and Zoltán Fülöp. “Decomposition of Weighted Multioperator Tree Automata”. In: *Int. J. Found. Comput. Sci.* 20.2 (2009), pp. 221–245 (cit. on p. 43).

[176] James W. Thatcher. “Characterizing derivation trees of context-free grammars through a generalization of finite automata theory.” In: *J. Comput. System Sci.* 1.4 (1967), pp. 317–322 (cit. on p. 15).

[177] Akihiko Tozawa. “Towards Static Type Checking for XSLT”. In: *Proceedings of the 2001 ACM Symposium on Document Engineering*. DocEng '01. 2001, pp. 18–27. DOI: 10.1145/502187.502191 (cit. on p. 116).

[178] K. Vijay-Shanker and Aravind K. Joshi. “Some computational properties of tree adjoining grammars”. In: *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*. 1985, pp. 82–93. URL: http://www.aclweb.org/anthology/P85-1011 (cit. on p. 82).

[179] Wolfgang Wechler. *Universal Algebra for Computer Scientists*. First edition. Vol. 25. Monogr. Theoret. Comput. Sci. EATCS Ser. Springer, 1992 (cit. on pp. 25, 28, 29, 32).

[180] David J. Weir. “Characterizing Mildly Context-Sensitive Grammar Formalisms”. PhD thesis. University of Pennsylvania, 1988 (cit. on p. 84).

[181] Kenji Yamada and Kevin Knight. "A syntax-based statistical translation model". In: *ACL '01: Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*. 2001, pp. 523–530 (cit. on p. 120).

[182] Sheng Yu. "Regular Languages". In: *Handbook of Formal Languages*. Ed. by Grzegorz Rozenberg and Arto Salomaa. Vol. 1. 1997. Chap. 2, pp. 41–110 (cit. on p. 131).

[183] Min Zhang, Hongfei Jiang, Aiti Aw, Haizhou Li, Chew Lim Tan, and Sheng Li. "A Tree Sequence Alignment-based Tree-to-Tree Translation Model". In: *Proceedings of ACL-08: HLT*. 2008, pp. 559–567. URL: http://www.aclweb.org/anthology/P/P08/P08-1064 (cit. on p. 9).

Index