Code Selection by Tree Series Transducers

Björn Borchardt*

Dresden University of Technology,
Department of Computer Science, 01062 Dresden, Germany
borchard@tcs.inf.tu-dresden.de

Abstract. In this paper we model code selection by tree series transducers. We are given an intermediate representation of some compiler as well as a machine grammar with weights, which reflect the number of machine cycles of the instructions. The derivations of the machine grammar are machine codes. In general, a machine grammar is ambiguous and hence there might exist more than one derivation of an intermediate code. We show how to filter out a cheapest such derivation and thereby perform tree parsing and tree pattern matching using tree series transducers.

1 Introduction

In this paper we model code selection (cf.[GG78]) by tree series transducers (for short: trstr's). In general, a machine grammar is ambiguous and hence, for some intermediate representation (for short: IR) there might exist several machine codes. We would like to find a cheapest machine code, i.e., a machine code with the least number of machine cycles. To visualize this, let us consider the following example (cf. [GL97]): for the *C*-expression (`f` + `i`), where `f` and `i` are of type `float` and `int`, respectively, a cheapest machine code for an Intel iapX86 instruction set should be generated. All floating point operations are performed in an internal format. There are several possibilities for encoding (`f` + `i`) in the Intel instruction set: first both `f` and `i` are loaded and converted into the internal format and then put into registers, from which the floating point addition finally takes its arguments. Alternatively, only `f` is loaded and converted into the internal format and put into a register; the floating point addition then would take `i` from the memory and implicitly perform the loading and converting, or vice versa. It turns out that the second of these alternatives is best.

In this paper we follow and extend the approach of [FSW94], in which techniques of tree automata (e.g., subset construction) are applied to code selection. We generate the cheapest machine code by using the more powerful model of trstr's (cf. [EFV02]). This gives us the chance to describe the cheapest machine code as output of a sequence of trstr's. Let us therefore briefly recall the concept of (polynomial, top-down) trstr's. Basically, trstr's generalize tree transducers (for short: trtr's) by associating to every transition a weight, which is taken from

* Financially supported by the German Research Foundation (DFG, grant GK 334/3).

M. Domaratzki et al. (Eds.): CIAA 2004, LNCS 3317, pp. 57–67, 2005.

a semiring. The tree transformation of a trstr is similar to that of classical trtr, but additionally weights are accumulated: the weights of the transitions of a run on an input tree s are multiplied and finally the weights of all accepting runs which translate s to the same output tree t are summed up. Hence s is transformed by M into the tree series $\tau_M(s)$. The support of $\tau_M(s)$ can be considered as the set of output trees and $\big(\tau_M(s)\big)(t)$ denotes the weight of the transformation from s to t. We note that trstr's also cover (top-down) finite state weighted tree automata (for short: w-fta, cf. [BR82]).

We select the cheapest machine code by modeling the given machine grammar G by a regular, weighted tree grammar. Then the trstr M_G^{TP} translates the given IR s into the tree series $\tau_{M_G^{\mathrm{TP}}}(s)$. Each tree of the support of this tree series uniquely corresponds to a machine code, i.e., a derivation of the associated regular, weighted tree grammar, and the coefficient of such an output tree is the number of required machine cycles of the corresponding machine code, i.e., the weight of the corresponding derivation (tree parsing; also cf. [GG78]). The cheapest machine code of s then can be found by searching in the tree series $\tau_{M_G^{\mathrm{TP}}}(s)$ for a tree with minimal weight. We also show how to compute this minimal weight by providing the w-fta M_G^{mincost}. Moreover, we would like to find all occurrences of the right hand side of a rule r of G in s (tree pattern matching; also cf. [HO82]). Therefore we present the w-fta $M_{\tilde{r}}^{\mathrm{PM}}$, which generates the set of all occurrences of the right hand side of r in the input tree.

Let us point out the two main improvements to [FSW94]. There tree parsing is done by representing machine code as a computation of tree automata, while we generate the machine code explicitly. Moreover, tree pattern matching is solved in the aforementioned paper by deciding, whether or not a pattern is contained in the input tree, whereas we compute all the references on the occurrences of the pattern in the input tree.

This paper is organized as follows: in Sect. 2 we recall basic concepts, while the code selection problem is attacked in Sect. 3. We conclude this paper in Sect. 4 by stating open problems.

2 Preliminaries

2.1 Notions on Trees

Throughout this paper $\mathbb{N}$ and $\mathbb{N}_+$ denote the sets of all non-negative integers and all positive integers, respectively. Moreover, for every $i, j \in \mathbb{N}$, $[i,j] = \{n \in \mathbb{N} \mid i \leq n \leq j\}$. We abbreviate $[1,i]$ by $[i]$. Also, let $x_1, x_2, \ldots$ be variables, $X = \{x_1, x_2, \ldots\}$, and $X_n = \{x_1, \ldots, x_n\}$ for every $n \in \mathbb{N}$. A *ranked alphabet* is a tuple (Σ, rk) consisting of a non-empty, finite set Σ being disjoint with X and a rank mapping $\mathrm{rk} : \Sigma \to \mathbb{N}$. We always assume the rank mapping to be implicitly given and write Σ rather than (Σ, rk). Moreover, let $\Sigma^{(k)} = \{\sigma \in \Sigma \mid \mathrm{rk}(\sigma) = k\}$ for every $k \in \mathbb{N}$ and let $\mathrm{maxrk}(\Sigma) \in \mathbb{N} = \max\{k \in \mathbb{N} \mid \Sigma^{(k)} \neq \emptyset\}$. Now let $X' \subseteq X$. The set of *trees over Σ (indexed by X')* is denoted by $T_\Sigma(X')$ and defined to be the smallest subset T of $(\Sigma \cup X' \cup \{\mathtt{(}, \mathtt{)}, \mathtt{,}\})^*$ satisfying (i) $X' \subseteq T$

and (ii) given $k \in \mathbb{N}$, $\sigma \in \Sigma^{(k)}$, and $t_1, \ldots, t_k \in T$, then $\sigma(t_1, \ldots, t_k) \in T$. As usual, we set $T_\Sigma = T_\Sigma(\emptyset)$. We will be short in notation and write $s = \sigma(s_1, \ldots, s_k) \in T_\Sigma(X')$ as a shorthand for "there exist $k \in \mathbb{N}$, $\sigma \in \Sigma^{(k)}$, and $s_1, \ldots, s_k \in T_\Sigma(X')$". To define the tree substitution let $t \in T_\Sigma(X_n)$ for some $n \in \mathbb{N}$ and $s_1, \ldots, s_n \in T_\Sigma(X)$. Then $t[s_1, \ldots, s_n] \in T_\Sigma(X)$ is obtained from t by replacing simultaneously every occurrence of every variable $x_i \in X_n$ by s_i.

Let us now define some properties of a tree $s \in T_\Sigma(X')$. The *number* $\#_{X'}(s) \in \mathbb{N}$ *of occurrences of elements of* X' *in* s and the *set* $\mathrm{pos}(s) \in \mathfrak{P}(\mathbb{N}^*)$ *of positions of* s are given by $\#_{X'}(s) = 1$ and $\mathrm{pos}(s) = \{\varepsilon\}$ if $s \in X'$, and $\#_{X'}(s) = \sum_{i \in [k]} \#_{X'}(s_i)$ and $\mathrm{pos}(s) = \{\varepsilon\} \cup \{i.o \mid i \in [k], o \in \mathrm{pos}(s_i)\}$ provided that $s = \sigma(s_1, \ldots, s_k) \in T_\Sigma(X')$. Further, for every $i \in \mathbb{N}$ the *position* $o(s, i) \in \mathrm{pos}(s)$ *of the* i*th occurrence of an element of* X' *in* s is $o(s, 1) = \varepsilon$ provided that $s \in X'$, $o(s, i) = j.o(s_j, i')$ if $s = \sigma(s_1, \ldots, s_k) \in T_\Sigma(X')$ and there exist an $i' \in \mathbb{N}$ and a $j \in [k]$ such that $j = \sum_{l \in [j-1]} \#_{X'}(s_l) + i'$; otherwise $o(s, i)$ is undefined. Further, for every $o \in \mathrm{pos}(s)$ the *subtree* $s|_o \in T_\Sigma$ *of* s *at position* o is defined by $s|_o = s$ and provided that $o = \varepsilon$ and if $o = i.o'$ for some $i \in [k]$ and $o' \in \mathrm{pos}(s_i)$ then $s|_o = s_i|_{o'}$. We call s a *subtree of* t, denoted by $s \leq t$ if $s = t|_o$ for some $o \in \mathrm{pos}(t)$. Finally, for every $o \in \mathrm{pos}(s)$ the *label* $\mathrm{lab}_s(o) \in \Sigma$ *of* s *at position* o is given by $\mathrm{lab}_s(o) = s$ provided that $s \in X'$, $\mathrm{lab}_s(o) = \sigma$ if $s = \sigma(s_1, \ldots, s_k) \in T_\Sigma(X')$ and $o = \varepsilon$, and $\mathrm{lab}_s(o) = \mathrm{lab}_{s_i}(o')$, if $s = \sigma(s_1, \ldots, s_k) \in T_\Sigma(X')$ and $o = i.o'$ for some $i \in [k]$ and $o' \in \mathrm{pos}(s_i)$.

For a given $n \in \mathbb{N}$ and ranked alphabet Σ, a *pattern* (also: *(Σ-n-) context*) is a tree $C \in (X_n)$ such that every variable $x_i \in X_n$ occurs precisely once in C. The class of all Σ-n-contexts is denoted by $C_\Sigma(X_n)$. If $C \in C_\Sigma(X_n)$, $t \in T_\Sigma$, and $o \in \mathrm{pos}(t)$, then C *is a pattern of* t *at position* o, if $t|_o = C[t_1, \ldots, t_n]$ for some trees $t_1, \ldots, t_n \in T_\Sigma$. Finally, $\mathrm{occ}_C(t)$ is the set of all $o \in \mathrm{pos}(t)$ of t such that C is a pattern of t at o.

2.2 Semirings

A *semiring* is tuple $\mathcal{A} = (A, \oplus, \odot, \mathbf{0}, \mathbf{1})$ satisfying the following conditions: (i) $(A, \oplus, \mathbf{0})$ is a commutative monoid (i.e., $\oplus$ is a binary, associative, commutative operation on A with the neutral element $\mathbf{0}$), (ii) $(A, \odot, \mathbf{1})$ is a monoid, (iii) $\odot$ distributes over $\oplus$, (i.e., $a \odot (b \oplus c) = (a \odot b) \oplus (a \odot c)$ and $(a \oplus b) \odot c = (a \odot c) \oplus (b \odot c)$ for every $a, b, c \in A$), and (iv) $\mathbf{0}$ is absorptive (i.e., $\mathbf{0} \odot a = \mathbf{0} = a \odot \mathbf{0}$ for every $a \in A$). For a finite index set $I = \{i_1, \ldots, i_n\}$ and semiring elements $a_{i_j} \in A$ for every $j \in [n]$ we write $\bigoplus_{i \in I} a_i$ for $a_{i_1} \oplus \cdots \oplus a_{i_n}$ provided that $I \neq \emptyset$. For the sake of completeness we set $\bigoplus_{i \in I} a_i = \mathbf{0}$ if $I = \emptyset$. Throughout this paper let $\mathcal{A} = (A, \oplus, \odot, \mathbf{0}, \mathbf{1})$ be a semiring. In this paper we make us of the Boolean semiring $\mathrm{Bool} = (\{0, 1\}, \vee, \wedge, 0, 1)$, the Tropical semiring $\mathrm{Trop} = (\mathbb{N} \cup \{+\infty\}, \min, +, +\infty, 0)$, and for a (not necessarily finite) alphabet Σ the language semiring $\mathrm{Lang}_\Sigma = (\mathfrak{P}(\Sigma), \cup, ., \emptyset, \{\varepsilon\})$.

2.3 Tree Series and Tree Series Substitution

Let Σ be a ranked alphabet and $X' \subseteq X$. A *(formal) tree series (over* Σ *and* $\mathcal{A}$*)* is a total mapping $S : T_\Sigma(X') \to A$. The image $S(t) \in A$ is called *coefficient of*

$t \in T_\Sigma(X')$, and as usual, we write (S,t) rather than $S(t)$. The tree series, which maps every $t \in T_\Sigma(X')$ to $\mathbf{0}$, is denoted by $\widetilde{\mathbf{0}}$. The *support of* S is defined to be the set $\mathrm{supp}(S) = \{t \in T_\Sigma(X') \mid (S,t) \neq \mathbf{0}\}$. We will be short in notation and write $\bigoplus_{t \in \mathrm{supp}(S)} (S,t)\, t$ to denote $\bigoplus_{t \in T_\Sigma} (S,t)\, t$. The tree series S is called *polynomial*, if its support is finite. Further, $A\langle\langle T_\Sigma(X')\rangle\rangle$ and $A\langle T_\Sigma(X')\rangle$ are the *classes of all tree series and of all polynomial tree series over* Σ *and* $\mathcal{A}$, respectively. The sum of two tree series $S, T \in A\langle\langle T_\Sigma(X')\rangle\rangle$ is denoted by $S+T$ and defined by pointwise addition, i.e. $(S+T,s) = (S,s) \oplus (T,s)$ for every $s \in T_\Sigma(X')$.

To define the tree series substitution (cf. [EFV02]) let $T \in A\langle T_\Sigma(X_k)\rangle$ and $\boldsymbol{S} = (S_1, \ldots, S_k) \in A\langle T_\Sigma\rangle^k$ for some $k \in \mathbb{N}$. Then for every $s \in T_\Sigma$

$$(T \leftarrow \boldsymbol{S}, s) = \bigoplus_{\substack{t \in \mathrm{supp}(T) \\ (\forall i \in [k]): t_i \in \mathrm{supp}(S_i) \\ s = t[t_1, \ldots, t_k]}} (T,t) \odot (S_1, t_1) \odot \cdots \odot (S_k, t_k) \ .$$

2.4 Regular, Weighted Tree Grammars

Definition 1 (cf.[AB87]). *A* regular, weighted tree grammar *is defined to be a 6-tuple* $G = (\mathcal{N}, \Sigma, I, \mathcal{R}, \mathcal{A}, \mathrm{wt})$ *satisfying* $\mathcal{N} \cap \Sigma = \emptyset$*, where* $\mathcal{N}$ *and* Σ *are ranked alphabets (of non-terminals and terminals, respectively) with* $\mathcal{N} = \mathcal{N}^{(0)}$*,* $I \in \mathcal{N}$ *(the initial non-terminal),* $\mathcal{R}$ *is a finite set (of rules)* $N \to s$*, where* $N \in \mathcal{N}$*,* $s \in T_\Sigma(\mathcal{N}) \backslash \mathcal{N}$*, and* $\mathrm{wt} : \mathcal{R} \to \mathcal{A}$ *is the weight mapping. Let* $r = (N \to s) \in \mathcal{R}$ *be a rule. The* type *of* r *is* $\mathrm{type}(r) = (N_1, \ldots, N_n) \to N$*, where* $(N_1, \ldots, N_n) \in \mathcal{N}^n$ *is the sequence of non-terminals, which is obtained by reading the leaves of* s *from left to right and omitting all terminals. Moreover, we denote the right hand side* s *of* r *by* $\mathrm{RHS}(r)$ *and define the set* $\mathrm{RHS}(G) = \{\mathrm{RHS}(r) \mid r \in \mathcal{R}\}$.

In order to define the semantics of a regular, weighted tree grammar G let us introduce the notation $\widetilde{r}$ for some rule $r \in \mathcal{R}$ of type $(N_1, \ldots, N_n) \to N$: we define $\widetilde{r} \in C_\Sigma(X_n)$ as the context, which is obtained from $\mathrm{RHS}(r)$ by replacing N_i by x_i for every $i \in [n]$. Moreover, let $\Delta(G) = \{r^{(n)} \mid r \in \mathcal{R}, \mathrm{type}(r) = (N_1, \ldots, N_n) \to N\}$ be a ranked alphabet. An *N-derivation (tree) of $s \in T_\Sigma$ (with respect to G)* is a tree $\psi = r(\psi_1, \ldots, \psi_n) \in T_{\Delta(G)}$ such that $r \in \mathcal{R}$ is of type $(N_1, \ldots, N_n) \to N$ and there exist trees $s_1, \ldots, s_n \in T_\Sigma$ with $s = \widetilde{r}[s_1, \ldots, s_n]$ and ψ_i is an N_i-derivation of s_i for every $i \in [n]$. An I-derivation of s is also called a *derivation (tree)* (also: *abstract syntax tree*) *of* s. The *set of all derivations of s with respect to G* is denoted by $\mathrm{der}_G(s)$. The weight of an N-derivation $r(\psi_1, \ldots, \psi_n)$ where $r \in \mathcal{R}$ is of type $(N_1, \ldots, N_n) \to N$ and ψ_i is a N_i-derivation for every $i \in [n]$ is defined by $\mathrm{wt}(r(\psi_1, \ldots, \psi_n)) = \mathrm{wt}(r) \odot \mathrm{wt}(\psi_1) \odot \cdots \odot \mathrm{wt}(\psi_n)$.

2.5 Tree Series Transducer

Let us now recall the definition of trstr's. Being more precise, we instantiate the concept of trstr's introduced in [EFV02]: for our purposes it suffices to consider top-down trstr, which is reflected in Condition (b) of the next paragraph; also,

we restrict the devices to polynomial trstr', i.e., the weight of every transition is a polynomial tree series.

The transitions of a trstr and their weights are coded in a *tree representation (over a non-empty ranked alphabet $Q = Q^{(1)}$ of states, ranked alphabets Σ and Δ (of input and output symbols, respectively), and $\mathcal{A}$)*, which is a family $\mu = (\mu_k : \Sigma^{(k)} \to A\langle T_\Delta(X)\rangle^{Q \times Q(X_k)^*} \mid k \in \mathbb{N})$ of mappings such that

(a) for every $\sigma \in \Sigma^{(k)}$ there exist only finitely many indices $(q, w) \in Q \times Q(X_k)^*$ satisfying $\mu_k(\sigma) \neq \widetilde{\mathbf{0}}$ and
(b) for every $\sigma \in \Sigma^{(k)}$ and $(q, w) \in Q \times Q(X_k)^*$ it holds that $\mathrm{supp}(\mu_k(\sigma)_{q,w}) \subseteq C_\Sigma(X_l)$, where l denotes the length of w.

The semantics of a trstr is defined in terms of the mapping $h_\mu : T_\Sigma \to A\langle T_\Delta\rangle^Q$, given for every $s = \sigma(s_1, \ldots, s_k) \in T_\Sigma$ and $q \in Q$ by

$$h_\mu(s)_q = \bigoplus_{w = q_1(x_{i_1}) \ldots q_l(x_{i_l}) \in Q(X_k)^*} \mu_k(\sigma)_{q,w} \leftarrow \left(h_\mu(s_{i_1})_{q_1}, \ldots, h_\mu(s_{i_l})_{q_l}\right) .$$

Definition 2 ([EFV02]). *A* (polynomial, top-down) tree series transducer (for short: trstr) *is a tuple $M = (Q, \Sigma, \Delta, Q_d, \mathcal{A}, \mu)$, where $Q = Q^{(1)}$, Σ, and Δ are ranked alphabets, $Q_d \subseteq Q$, and μ is a tree representation over Q, Σ, Δ, and $\mathcal{A}$. Moreover, M is called* tree transducer (for short: trtr)*, if $\mathcal{A} =$ Bool, and it is called* (finite state) weighted tree automaton (for short: w-fta) *if $\Sigma = \Delta$ and for every $k \in \mathbb{N}$, $\sigma \in \Sigma^{(k)}$, $q \in Q$, and $w \in Q(X_k)^*$ it holds that $\mu_k(\sigma)_{q,w} = a\,\sigma(x_1, \ldots, x_k)$ for some $a \in A$ provided that $w = q_1(x_1) \ldots q_k(x_k)$, and $\mu_k(\sigma)_{q,w} = \widetilde{\mathbf{0}}$ otherwise. The semantics of M is a mapping $\tau_M : T_\Sigma \to A\langle T_\Delta\rangle$, which is defined for every $s \in T_\Sigma$ by $\tau_M(s) = \bigoplus_{q \in Q_d} h_\mu(s)_q$.*

As usual, we simplify notations for a trtr by writing $M = (Q, \Sigma, \Delta, F, \mu)$ rather than $M = (Q, \Sigma, \Delta, F, \mathrm{Bool}, \mu)$ and identifying every tree series occurring in the syntax or semantics of M with its support. Moreover, the generation of the output tree by a w-fta is superfluous. Further, if $\mu_k(\sigma)_{q,w} \neq \widetilde{\mathbf{0}}$, then w is of type $q_1(x_1) \ldots q_k(x_k)$. In particular, $\mu_k(\sigma)_{q,w}$ with w not being of the aforementioned type do not contribute to any generated tree series. We therefore shorten notation by writing $M = (Q, \Sigma, Q_d, \mathcal{A}, \mu)$ and $\mu_k(\sigma)_{q,(q_1,\ldots,q_k)} = a$ rather than $M = (Q, \Sigma, \Sigma, Q_d, \mathcal{A}, \mu)$ and $\mu_k(\sigma)_{q,(q_1(x_1),\ldots,q_k(x_k))} = a\,\sigma(x_1, \ldots, x_k)$, respectively. Also, in the accepted tree series $\tau_M(s)$ we omit the output tree, i.e., a stands for $a\,s$. Hence, every input tree is accepted by a w-fta with a semiring element. Thus the semantics of M also can be considered as a tree series, which we denote by S_M.

3 Code Selection

For the rest of this section let $G = (\mathcal{N}, \Sigma, I, \mathcal{R}, \mathrm{Trop}, \mathrm{wt})$ be a regular, weighted tree grammar and $s \in T_\Sigma$.

3.1 Tree Parsing

In this section we generate a representation of all derivations of s together with their costs, i.e. we solve the (extended) tree parsing problem (cf. [GG78]):

> **(Extended) Tree Parsing Problem:** Compute explicitly $\mathrm{der}_G(s)$ and the weight of every $\psi \in \mathrm{der}_G(s)$.

Therefore we define the trstr M_G^{TP}. This trstr generates for every input tree t a tree series the support of which uniquely corresponds to $\mathrm{der}_G(t)$. Moreover, the coefficient of a tree contained in $\mathrm{supp}(\tau_{M_G^{\mathrm{TP}}})$ is the weight of the corresponding derivation.

Let us first show, how we represent a derivation $\psi \in \mathrm{der}_G(s)$. We introduce for every $k \in [0, \mathrm{maxrk}(\Sigma)]$ a symbol e_k. The pseudo-code tree also contains nodes the label of which represents a rule $r \in \mathcal{R}$ and the rank of which equals the rank of the topmost element of $\mathrm{RHS}(r)$: $\Delta_{\mathrm{pseu}} = \{e_k^{(k)} \mid k \in [0, \mathrm{maxrk}(\Sigma)]\} \cup \{r_{\mathrm{pseu}}^{(k)} \mid r \in \mathcal{R}, k = \mathrm{rk}(\mathrm{lab}_{\mathrm{RHS}(r)}(\varepsilon))\}$. By definition it holds that $\psi = r(\psi_1, \ldots, \psi_n)$ for some $r \in \mathcal{R}$, $n \in \mathbb{N}$, and $\psi_1, \ldots, \psi_n \in \mathrm{der}_G$. The *pseudo-code tree of* ψ is inductively defined to be the tree $\mathrm{pseu}(\psi) = C_r[\mathrm{pseu}(\psi_1), \ldots, \mathrm{pseu}(\psi_n)] \in T_{\Delta_{\mathrm{pseu}}}$, where $C_r \in C_{\Delta_{\mathrm{pseu}}}(X_n)$ is a context satisfying $\mathrm{pos}(C_r) = \mathrm{pos}(\widetilde{r})$ $(= \mathrm{pos}(\mathrm{RHS}(r)))$ and for every $o \in \mathrm{pos}(C)$,

$$\mathrm{lab}_{C_r}(o) = \begin{cases} r_{\mathrm{pseu}} & \text{, if } o = \varepsilon \\ e_k & \text{, if } o \neq \varepsilon \text{ and } \mathrm{lab}_{\widetilde{r}}(o) \in \Sigma^{(k)} \text{ for some } k \in \mathbb{N} \\ \mathrm{lab}_{\widetilde{r}}(o) & \text{, otherwise .} \end{cases}$$

Clearly, from the pseudo-code tree the original computation ψ can be inductively reobtained by replacing the context C_r by the n-ary label r.

Let us now present the trstr M_G^{TP}, which solves the tree parsing problem. It traverses the input tree s and successively replaces patterns of s corresponding to a right hand side of some $r \in \mathcal{R}$ by C_r also checking, whether the "connecting" non-terminals are of appropriate type. Hence the states of M_G^{TP} are all the proper subtrees of trees contained in $\mathrm{RHS}(G)$ as well as the initial non-terminal I. The transitions are defined in the obvious way, where the weight of the rule r is assigned to the transition, which consumes the topmost element of $\mathrm{RHS}(r)$.

Definition 3. *The trstr* $M_G^{\mathrm{TP}} = (Q, \Sigma, \Delta, F, \mathcal{A}, \mu)$ *is defined by* $Q = \{I\} \cup \{t' \in T_\Sigma(\mathcal{N}) \mid (\exists t \in \mathrm{RHS}(G)) : t' < t\}$, $\Delta = \Delta_{\mathrm{pseu}}$, $Q_d = \{I\}$, $\mathcal{A} = \mathrm{Trop}$, *and for every* $k \in \mathbb{N}$, $\sigma \in \Sigma^{(k)}$, $w = q_1(x_{i_1}) \ldots q_l(x_{i_l}) \in Q(X_k)^*$, *and* $q \in Q$ *it holds that*

$$\mu_k(\sigma)_{q,w} = \begin{cases} \mathrm{wt}(r)\, r_{\mathrm{pseu}}(x_1, \ldots, x_k) & \text{, if } w = q_1(x_1) \ldots q_k(x_k), \\ & \quad r = (q \to \sigma(q_1, \ldots, q_k)) \in \mathcal{R} \\ 0\, e_k(x_1, \ldots, x_k) & \text{, if } w = q_1(x_1) \ldots q_k(x_k), \\ & \quad q = \sigma(q_1, \ldots, q_k) \\ \widetilde{+\infty} & \text{, otherwise .} \end{cases}$$

Lemma 1. *It holds that* $\tau_{M_G^{\mathrm{TP}}}(s) = \bigoplus_{\psi \in \mathrm{der}_G(s)} \mathrm{wt}(\psi)\ \mathrm{pseu}(\psi)$.

Example 1. Let us consider the regular, weighted tree grammar G given by $\mathcal{N} = \{I, A, B\}$, $\Sigma = \{\sigma^{(2)}, \alpha^{(0)}\}$, and $\mathcal{R} = \{r_1, r_2, r_3, r_4\}$, where

$$\begin{array}{llll} r_1 : I \to \sigma(\sigma(A,A),A) & \mathrm{wt}(r_1) = 3, & r_3 : I \to \sigma(B,\alpha) & \mathrm{wt}(r_3) = 2, \\ r_2 : A \to \alpha & \mathrm{wt}(r_2) = 1, & r_4 : B \to \sigma(A,A) & \mathrm{wt}(r_4) = 3. \end{array}$$

According to Definition 3 the set of states of M_G^{TP} is $Q = \{I, A, B, \alpha, \sigma(A,A)\}$, where I is the unique designated state. Moreover,

$$\begin{array}{lll} r_1 : & \mu_2(\sigma)_{I,\,\sigma(A,A)(x_1)\,A(x_2)} & = 3\,(r_1)_{\mathrm{pseu}}(x_1,x_2), \\ & \mu_2(\sigma)_{\sigma(A,A),\,A(x_1)\,A(x_2)} & = 0\,e_2(x_1,x_2), \\ r_2 : & \mu_0(\alpha)_{A,\,()} & = 1\,(r_2)_{\mathrm{pseu}}, \\ r_3 : & \mu_2(\sigma)_{I,\,B(x_1)\,\alpha(x_2)} & = 2\,(r_3)_{\mathrm{pseu}}(x_1,x_2), \\ & \mu_0(\alpha)_{\alpha,\,()} & = 0\,e_0, \\ r_4 : & \mu_2(\sigma)_{B,\,A(x_1)\,A(x_2)} & = 3\,(r_4)_{\mathrm{pseu}}(x_1,x_2), \end{array}$$

and the not yet defined entries of the tree representation μ are set $\widetilde{+\infty}$. Let us consider the input tree $s = \sigma(\sigma(\alpha,\alpha),\alpha)$. Clearly, $\mathrm{der}_G(s) = \{\psi_1, \psi_2\}$, where $\psi_1 = r_1(r_2, r_2, r_2)$, $\mathrm{wt}(\psi_1) = 6$, $\psi_2 = r_3(r_4(r_2, r_2))$, and $\mathrm{wt}(\psi_2) = 7$. Let us now compute $\tau_{M_G^{\mathrm{TP}}}(s)$. For this purpose we calculate the characteristic vector $h_\mu(s)$ of s, which is shown in the following table, where t_1 and t_2 denote the trees $\mathrm{pseu}(\psi_1) = (r_1)_{\mathrm{pseu}}(e_2((r_2)_{\mathrm{pseu}}, (r_2)_{\mathrm{pseu}}), (r_2)_{\mathrm{pseu}})$ and $\mathrm{pseu}(\psi_2) = (r_3)_{\mathrm{pseu}}((r_4)_{\mathrm{pseu}}((r_2)_{\mathrm{pseu}}, (r_2)_{\mathrm{pseu}}), e_0)$, respectively.

$h_\mu(t)_q$	α	$\sigma(\alpha,\alpha)$	s
I	$\widetilde{+\infty}$	$\widetilde{+\infty}$	$\min\{6\,t_1, 7\,t_2\}$
A	$1\,(r_2)_{\mathrm{pseu}}$	$\widetilde{+\infty}$	$\widetilde{+\infty}$
B	$\widetilde{+\infty}$	$5\,(r_4)_{\mathrm{pseu}}((r_2)_{\mathrm{pseu}}, (r_2)_{\mathrm{pseu}})$	$\widetilde{+\infty}$
α	$0\,e_0$	$\widetilde{+\infty}$	$\widetilde{+\infty}$
$\sigma(A,A)$	$\widetilde{+\infty}$	$2\,e_2((r_2)_{\mathrm{pseu}}, (r_2)_{\mathrm{pseu}})$	$\widetilde{+\infty}$

Consequently, $\tau_{M_G^{\mathrm{TP}}}(s) = h_\mu(s)_I = \min\{6\ \mathrm{pseu}(\psi_1), 7\ \mathrm{pseu}(\psi_2)\}$.

3.2 Cost of a Cheapest Derivation

Now we compute the weight of a cheapest derivation of s with respect to G by the w-fta M_G^{mincost}, which works very similar to M_G^{TP}. Rather than replacing input symbols it just copies them. Moreover, it accumulates in every run the weight of a derivation of the input tree. Since a w-fta finally sums up (in Trop: takes the minimum) over the weights of all successful runs, M_G^{mincost} indeed computes the minimum of the weights of all derivations of the input tree.

Definition 4. *The w-fta* $M_G^{\mathrm{mincost}} = (Q, \Sigma, Q_d, \mathcal{A}, \mu)$ *is defined by* $Q = \{I\} \cup \{t' \in T_\Sigma(\mathcal{N}) \mid (\exists t \in \mathrm{RHS}(G)) : t' < t\}$, $Q_d = \{I\}$, $\mathcal{A} = \mathrm{Trop}$, *and for every*

$k \in \mathbb{N}$, $\sigma \in \Sigma^{(k)}$, $\boldsymbol{q} = (q_1, \ldots, q_k) \in Q^k$, and $q \in Q$ it holds that

$$\mu_k(\sigma)_{q,\boldsymbol{q}} = \begin{cases} \mathrm{wt}(r) & \text{, if } r = (q \to \sigma(q_1, \ldots, q_k)) \in \mathcal{R} \\ 0 & \text{, if } q = \sigma(q_1, \ldots, q_k) \\ +\infty & \text{, otherwise .} \end{cases}$$

Lemma 2. *It holds that* $(S_{M_G^{\mathrm{mincost}}}, s) = \min\{\mathrm{wt}(\psi) \in \mathbb{N} \mid \psi \in \mathrm{der}_G(s)\}$.

The pseudo-code tree of a derivation having minimal weight is obtained by searching in $\tau_{M_G^{\mathrm{TP}}}(s)$ for an output tree with the coefficient $(S_{M_G^{\mathrm{mincost}}}, s)$.

3.3 Obtaining the Cheapest Machine Code

In this section we translate each pseudo-code tree $\mathrm{pseu}(\psi)$ into its corresponding machine code ψ by the trtr M_G^{trans}. Clearly, the input ranked alphabet of this trtr is Δ_{pseu} and its output ranked alphabet is $\Delta(G) = \{r^{(n)} \mid r \in \mathcal{R}, \mathrm{type}(r) = (N_1, \ldots, N_n) \to N\}$. Now let us show the states and transitions of M_G^{trans}. For this purpose let us consider the pseudo-code tree $\mathrm{pseu}(\psi) = C_r[\mathrm{pseu}(\psi_1), \ldots, \mathrm{pseu}(\psi_n)]$ for some rule $r \in \mathcal{R}$ of type $N \to (N_1, \ldots, N_n)$ and N_i-derivation ψ_i for every $i \in [n]$. The trtr M_G^{trans} should generate for this particular input tree the set $\{r(\psi_1, \ldots, \psi_n)\}$. By traversing $\mathrm{pseu}(\psi)$ the device consumes the topmost symbol r_{pseu}, generates $\{r(x_1, \ldots, x_n)\}$, and changes to $w = q_1(x_1) \ldots q_l(x_l)$. Since M_G^{trans} should substitute each of the variables x_i of $\{r(x_1, \ldots, x_n)\}$ by $\{\psi_i\}$ (which is generated by the "subrun" on $\mathrm{pseu}(\psi_i)$), the automaton has to traverse C_r to the node at position $o(C_r, i)$ when fulfilling the computation of $q_i(x_i)$. Therefore all the tuples (t, i) are states of M_G^{trans} where t is a subtree C_r for some rule r of G and i is a positive integer such that t contains at least i variables. Further, $(I, 1)$ is a state. In particular, it is the unique designated state of M_G^{trans}. The transitions are defined according to the aforementioned procedure.

Definition 5. *Let $M_G^{\mathrm{trans}} = (Q, \Sigma, \Delta, Q_d, \mu)$ be the trtr which is given by $Q = \{(I, 1)\} \cup \{(t, i) \mid (\exists i \in \mathbb{N}_+)(\exists r \in \mathcal{R}) : t < C_r, o(t, i)$ defined$\}$, $\Delta = \Delta(G)$, and $Q_d = \{(I, 1)\}$. Moreover, for every $k \in \mathbb{N}$, $q \in Q$, and $\sigma \in \Sigma^{(k)}$, $w = (q_1(x_{i_1}), \ldots, q_l(x_{i_l}))$, it holds that*

$$\mu_k(\sigma)_{q,w} = \begin{cases} \{r(x_1, \ldots, x_l)\} & \text{, if } (\exists r = N \to t \in \mathcal{R}), (\forall j \in [l]), (\exists j' \in [l]) : \\ & \quad C_r = \sigma(C_r|_1, \ldots, C_r|_k) \in C_\Sigma(X_l), q = (N, 1), \\ & \quad \big(q_j = (C_r|_{i_j}, j') \iff o(C_r, j) = i_j.o(C_r|_{i_j}, j')\big) \\ \{x_1\} & \text{, if } (\exists t \in T_\Sigma(X_l), (\exists i \in [k]), (\exists j, j' \in \mathbb{N}) : \\ & \quad \sigma = e_k, t = \sigma(t|_1, \ldots, t|_k), q = (t, j), l = 1, \\ & \quad \big(q_1 = (t|_{i_1}, j') \iff o(t, j) = i_1.o(t|_{i_1}, j')\big) \\ \emptyset & \text{, otherwise .} \end{cases}$$

Lemma 3. *For every* $\psi \in \mathrm{der}_G(s)$ *it holds that* $\tau_{M_G^{\mathrm{trans}}}(\mathrm{pseu}(\psi)) = \{\psi\}$.

Example 2. Let us reconsider the regular, weighted tree grammar G of Example 1. According to Definition 5 the set of states of M_G^{trans} is given by $Q = \{(I,1),(A,1),(B,1),(e_2(A,A),1),(e_2(A,A),2)\}$, where $(I,1)$ is the unique designated state. The transitions having a weight different from $\emptyset$ are

$$
\begin{aligned}
C_{r_1}: \quad & \mu_2((r_1)_{\mathrm{pseu}})_{(I,1),\,(e_2(A,A),1)(x_1).(e_2(A,A),2)(x_1).(A,1)(x_2)} && = \{r_1(x_1,x_2,x_3)\},\\
& \mu_2(e_2)_{(e_2(A,A),1),\,(A,1)(x_1)} && = \{x_1\},\\
& \mu_2(e_2)_{(e_2(A,A),2),\,(A,1)(x_2)} && = \{x_1\},\\
C_{r_2}: \quad & \mu_0((r_2)_{\mathrm{pseu}})_{(A,1),\,()} && = \{r_2\},\\
C_{r_3}: \quad & \mu_2((r_3)_{\mathrm{pseu}})_{(I,1),\,(B,1)(x_1)} && = \{r_3(x_1)\},\\
C_{r_4}: \quad & \mu_2((r_4)_{\mathrm{pseu}})_{(B,1),\,(A,1)(x_1).(A,1)(x_1)} && = \{r_4(x_1,x_2)\}.
\end{aligned}
$$

Let us now consider $\mathrm{pseu}(\psi_1) = (r_1)_{\mathrm{pseu}}(e_2((r_2)_{\mathrm{pseu}},(r_2)_{\mathrm{pseu}}),(r_2)_{\mathrm{pseu}})$, which we generated in Example 1. The following table shows all the intermediates steps of the translation of M_G^{trans} from $\mathrm{pseu}(\psi_1)$ to $\{\psi_1\}$.

$h_\mu(t)_q$	$(r_2)_{\mathrm{pseu}}$	$e_2((r_2)_{\mathrm{pseu}},(r_2)_{\mathrm{pseu}})$	$\mathrm{pseu}(\psi_1)$
$(I,1)$	$\emptyset$	$\emptyset$	$\{r_1(r_2,r_2,r_2)\}$
$(A,1)$	$\{r_2\}$	$\emptyset$	$\emptyset$
$(B,1)$	$\emptyset$	$\emptyset$	$\emptyset$
$(e_2(A,A),1)$	$\emptyset$	$\{r_2\}$	$\emptyset$
$(e_2(A,A),1)$	$\emptyset$	$\{r_2\}$	$\emptyset$

In particular, $\tau_{M_G^{\mathrm{trans}}}(\psi_1) = h_\mu(\mathrm{pseu}(\psi_1))_{(I,1)} = \{r_1(r_2,r_2,r_2)\} = \{\psi_1\}$.

3.4 Tree Pattern Matching

In this section we attack the tree pattern matching problem, i.e., for a given pattern $C \notin X$ we generate the set $\mathrm{occ}_C(s)$ of occurrences of the pattern C in an input tree $s \in T_\Sigma$ (cf. [HO82]).

Tree Pattern Matching Problem: Let $s \in T_\Sigma$ and $C \in C_\Sigma(X_n) \setminus X_n$ for some $n \in \mathbb{N}$. Compute the set $\mathrm{occ}_C(s)$.

The tree pattern matching problem is solved by the w-fta M_C^{PM} over $\mathrm{Lang}_{\mathbb{N}}$. By letting M_C^{PM} run on the input tree $s \in T_\Sigma$ we obtain a set containing an element $o \in \mathrm{occ}_C(s)$. How is this o computed? The automaton traverses s starting at its root as far as it assumes an occurrence of C. It also outputs as a weight the set containing the position of s, at which it assumes the copy of C. By consuming the top-most symbol of this assumed occurrence of C it changes the state keeping the information that it just has consumed the root of C. Now M_C^{PM} either meets the whole pattern C and consumes it without changing the

up to now generated output or it stops. Clearly, if the w-fta has consumed a copy of C, then it has to consume the subtrees of s at the open position of the copy of C and keep the information that a pattern C was found. The traversing of s up to the occurrence of C is done in a state C, while the consumption of the pattern C is done in the states $t < C$, $t \notin X$. The traversing of the subtrees at the open positions of C is done in the state $\bot$.

Definition 6. *Let $n \in \mathbb{N}$. Moreover, let $C \in C_\Sigma(X_n) \setminus X_n$. The w-fta $M_C^{\mathrm{PM}} = (Q, \Sigma, Q_d, \mathcal{A}, \mu)$ is defined by $Q = \{\bot\} \cup \{t \in T_\Sigma(X_n) \setminus X_n \mid t \leq C\}$, $Q_d = \{C\}$, $\mathcal{A} = \mathrm{Lang}_{\mathbb{N}}$, and for every $k \in \mathbb{N}$, $\sigma \in \Sigma^{(k)}$, $\boldsymbol{q} = (q_1, \ldots, q_k) \in Q^k$, $q \in Q$, and $l \in [k]$ by*

$$\mu_k(\sigma)_{q,\boldsymbol{q}} = \begin{cases} \{\varepsilon\} & , \text{ if } \big((\forall i \in [k]) : q = q_i = \bot\big) \text{ or} \\ & \quad \big((\exists I \subseteq [k]), (\forall i \in I), (\forall j \in [k] \setminus I), (\exists T_i, T_j \in T_\Delta(X_n)) : \\ & \qquad T_i = \mathrm{lab}_i(q) \in X_n, T_j = q_j, q = \sigma(T_1, \ldots, T_k), q_i = \bot, \\ & \qquad q_j \in T_\Sigma(X_n)\big) \\ \{l\} & , \text{ if } (\forall i \in [k] \setminus \{l\}) : q_l = C = q, q_i = \bot, \\ \emptyset & , \text{ otherwise } . \end{cases}$$

Lemma 4. *For every $n \in \mathbb{N}$ and $C \in C_\Sigma(X_n) \setminus X_n$, $(S_{M_C^{\mathrm{PM}}}, s) = \mathrm{occ}_C(s)$.*

4 Conclusion and Open Problems

We extended the techniques of [FSW94] for code selection by using trstr's rather than tree automata and represented the cheapest machine code as output of a sequence of trstr's. Thereby we solved the tree parsing and the tree pattern matching problems by trstr's. It remains to "optimize", i.e., determinize and minimize these devices. In particular, it is an interesting question under which conditions trstr's over Trop can be determinized and minimized.

Acknowledgment

I would like to thank the unknown referees as well as Heiko Vogler, Andreas Maletti, and Janis Voigtländer for their helpful comments on previous versions of this paper.

References

[AB87] A. Alexandrakis and S. Bozapalidis. Weighted grammars and Kleenes theorem. *Information Processing Letters*, 24(1):1–4, January 1987.

[BR82] J. Berstel and C. Reutenauer. Recognizable formal power series on trees. *Theoretical Computer Science*, 18(2):115–148, 1982.

[EFV02] J. Engelfriet, Z. Fülöp, and H. Vogler. Bottom-up and top-down tree series transformations. *J. Automata, Languages and Combinatorics*, 7:11–70, 2002.

[FSW94] C. Ferdinand, H. Seidl, and R. Wilhelm. Tree automata for code selection. *Acta Informatica*, 31(8):741–760, 1994.

[GG78] R.S. Glanville and S.L. Graham. A new method for compiler code generation. In *Proceedings of the 5th ACM Symposium on Principles of Programming Languages*, pages 231–240, 1978.

[GL97] K.J. Gough and J. Ledermann. Optimal code-selection using MBURG. Presented to the 20th Australian Computer Science Conference, Sydney, 1997.

[HO82] C. Hoffmann and M.J. O'Donnell. Pattern matching in trees. *J. ACM*, 29:68–95, 1982.