Universität Ulm
Fakultät für Informatik

The Translation Power of Top-Down Tree-To-Graph Transducers

Joost Engelfriet
Leiden University
Heiko Vogler
Universität Ulm

Nr. 92-14

Ulmer Informatik-Berichte

Dezember 1992

The Translation Power of Top-Down Tree-To-Graph Transducers

JOOST ENGELFRIET [1]
Department of Computer Science, Leiden University
P.O.Box 9512, 2300 RA Leiden, The Netherlands
e-mail: engelfriet@rulcri.leidenuniv.nl

HEIKO VOGLER
Department of Theoretical Computer Science, University of Ulm
Oberer Eselsberg, 7900 Ulm, Germany
vogler@informatik.uni-ulm.de

Abstract: We introduce a new syntax-directed translation device called top-down tree-to-graph transducer. Such transducers are very similar to the usual top-down tree transducers except that the right-hand sides of their rules are hypergraphs rather than trees. Since we are aiming at a device which also allows to translate trees into objects different from graphs, we focus our attention on so-called tree-generating top-down tree-to-graph transducers. Then the result of every computation is a hypergraph which represents a tree, and in its turn the tree can be interpreted in any algebra of appropriate signature. Although for both devices, top-down tree transducers and tree-generating top-down tree-to-graph transducers, the translation of a subtree of an input tree does not depend on its context, the latter transducers have much more transformational power than the former. In this paper we prove that tree-generating top-down tree-to-graph transducers are equivalent to macro tree transducers, which are transducers for which the translation of a subtree may depend on its context.

[1]Supported by COMPUGRAPH ("Computing by Graph Transformation"), ESPRIT Basic Research Working Group Number 3299

1 Introduction

The concept of syntax-directed translation as introduced in [Iro61], has become a well-known and elegant method to associate meaning to tree-structured objects (cf. [Eng81] for a general discussion). The main idea of this concept is to define the translation (or semantics or meaning) of a tree in terms of translations of its subtrees. Nowadays, there exist several formalizations of this concept which, from the schematic point of view, have different translation power, e.g., *generalized syntax-directed translation schemes* [AU71, AU73], *top-down tree transducers* [Tha70, Rou70, Bak78, Eng75, GS84], *attributed tree transducers* [Fül81b, Fül81a] which are based on attribute grammars [Knu68], *macro tree transducers* [CF82, Eng80, EV85], and *high-level tree transducers* [EV88]. Actually, viewed as schemes, every device in this list is strictly more powerful than its predecessor except for the first two devices which are equivalent. The macro tree transducer closely corresponds to the model of syntax-directed translation in [Iro61].

In this paper, we introduce a new, quite simple, formal model of syntax-directed translation: the *top-down tree-to-graph transducer* (for short: td-tg transducer). Its definition has the same simplicity as the definitions of top-down tree transducers or attribute grammars with synthesized attributes only: the translation of a subtree of the input tree does not depend on its context.

Intuitively, td-tg transducers can be viewed as context-free hypergraph grammars of which the derivations are controlled by an input tree; the result of the derivation is the output (hyper)graph. This is entirely analogous to the fact that top-down tree transducers can be viewed as ordinary context-free grammars with tree-controlled derivations (see [Eng86a]). Context-free hypergraph grammars (for short: cfhg grammars) are introduced in [BC87, HK87b] and are studied in [Hab89, Cou87b, Cou87a, Cou88] [Cou90, ER90, HK87a, Lau88a] [Lau88b, MR87, EH91, EH92]. A cfhg grammar generates a set of edge labeled, directed hypergraphs. These are graphs in which an edge may be incident with any number of nodes rather than two as for usual graphs. Every edge is labeled by a symbol of some ranked alphabet where the rank of the label is equal to the number of incident nodes. Figure 1 shows a hypergraph where edges and nodes are represented by boxes and circles, respectively. The line connecting an edge and a node is called a tentacle. To implement a direction of tentacles and a functionality of edges, we associate with an edge e the sequence $v_1 \dots v_k v_{k+1}$ of incident nodes. The nodes $v_1, \dots, v_k$ are called the arguments of e and the node v_{k+1} is called the result of e; the tentacle which connects v_i, $i \in \{1, \dots, k\}$, with e (or v_{k+1} with e), is called outgoing from v_i (or incoming to v_{k+1}, respectively). Some nodes of the hypergraph may be distinguished; they are called external nodes and their order is indicated by natural numbers.

The derivation mechanism of cfhg grammars is an easy edge replacement: for two hypergraphs ξ_1 and ξ_2 over some ranked alphabet, ξ_1 can be rewritten to ξ_2 by replacing some X-labeled edge e of ξ_1 (where X is a nonterminal) by the right-hand side h of some rule $X \to h$, and by pairwise identifying the nodes that were incident with e in ξ_1 with the external nodes of h (cf. Figure 2); the number of external nodes of h equals the rank of X. Now td-tg transducers can be understood as cfhg grammars of which the derivations are controlled by an input tree, in the sense that the nonterminals of the grammar correspond

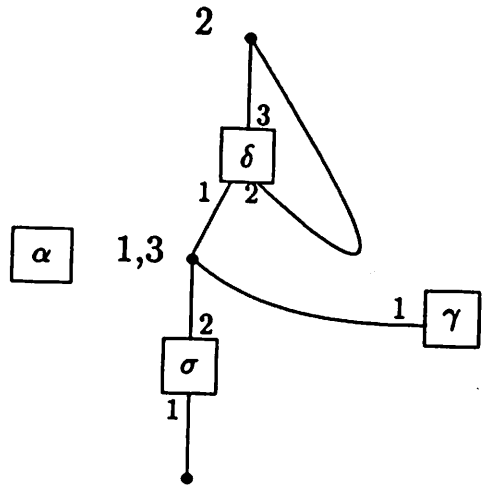


Figure 1: A hypergraph.

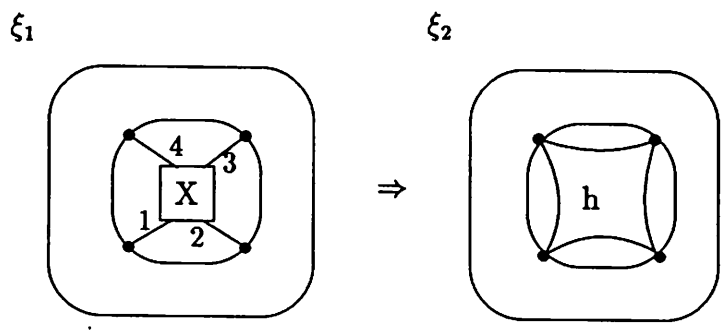


Figure 2: Application of the rule $X \to h$, where X has rank 4.

to pairs ⟨state, input subtree⟩ of the transducer. In the formal model of td-tg transducers we use input trees over some ranked alphabet. This can be understood as considering a one-sorted abstract syntax; the many-sorted case can be easily superposed on our model.

Formally, a td-tg transducer M consists of three ranked alphabets Q, Σ, and Δ of *states*, *input symbols*, and *output symbols*, respectively, an *initial state* q_{in} of rank 1, and a finite set R of *rules* of the form:

$$(*) \qquad \langle q, \sigma(x_1, \ldots, x_k)\rangle \to h$$

where q is a state, $\sigma \in \Sigma$ with rank $k \geq 0$, $x_1, \ldots, x_k$ are subtree variables, and h is a hypergraph in which the edges are labeled either by output symbols or by pairs $\langle q', x_j\rangle$ where q' is a state and $x_j \in \{x_1, \ldots, x_k\}$. The number of external nodes of h is equal to the rank of q. Moreover, we require that for every pair q and σ, there is exactly one rule with left-hand side $\langle q, \sigma(x_1, \ldots, x_k)\rangle$ in M. A rule like $(*)$ expresses that the q-translation of an input (sub)tree $\sigma(s_1, \ldots, s_k)$ is the graph h in which every edge with label $\langle q', x_j\rangle$ is

replaced by the q'-translation of the tree s_j. Thus, similar to top-down tree transducers, the q-translation of $\sigma(s_1, \ldots, s_k)$ does not depend on its context.

The set R of rules induces a derivation relation $\Rightarrow_M$ where the application of a rule $(*)$ to a $\langle q, \sigma(s_1, \ldots, s_k)\rangle$-labeled edge e of a sentential form ξ consists of three steps: e is replaced by h, e-incident nodes in ξ are identified with corresponding external nodes of h as in the derivation mechanism of cfhg grammars, and every x_j is replaced by s_j. The induced derivation relation $\Rightarrow_M$ is confluent and noetherian (cf., respectively, [Hue80, Der87] for these notions). Thus, for every tree s over Σ, M computes exactly one hypergraph g over output symbols only, such that $sing(\langle q_{in}, s\rangle) \Rightarrow_M^* g$, where $sing(\langle q_{in}, s\rangle)$ is the hypergraph which consists of one $\langle q_{in}, s\rangle$-labeled edge and one node v; v is incident with e and it is an external node. The hypergraph g is called the translation of the tree s. We associate with M the mapping $\tau(M)$, called *tree-to-(hyper)graph translation of* M, which maps an input tree to its translation. We note that, since $sing(\langle q_{in}, s\rangle)$ has one external node and since $\Rightarrow_M$ preserves the number of external nodes of sentential forms, the translation of s also has one external node (this restriction is not essential, but it is assumed here because we will investigate tree-generating td-tg transducers only; the external node represents the root of the tree).

The main advantage of having output graphs rather than trees, is the fact that graphs can represent trees with shared common subtrees. In fact, in this paper we focus our attention on so-called *tree-generating td-tg transducers* M. For such transducers, every computation results in a hypergraph which is acyclic and in which every node has exactly one incoming tentacle. Such a hypergraph (called "jungle") represents a tree (cf. [Cou88, HK87a, Hab89, HKP91]) and, in its turn, this tree can be interpreted into a semantic domain of appropriate signature. Intuitively, a tree is obtained from a jungle g by unfolding g, starting from the (unique) external node. Let $g = \tau(M)(s)$ be the translation of some input tree s and let $tree(g)$ denote the tree which is obtained by unfolding g. Then we can associate with M a tree-to-tree translation, denoted by $\tau_t(M)$ and defined by $\tau_t(M)(s) = tree(g)$. Let $tgtT$ denote the class of tree-to-tree translations computed by tree-generating td-tg transducers (where $tgtT$ stands for "tree-to-graph-to-tree Translations").

We remark that recently another formalization of the concept of syntax-directed translation was introduced [EH92] which is also based on cfhg grammars: the *cfhg-based syntax-directed translation schemes* (for short: cts). There the starting point is a usual context-free grammar G. For the description of the translation, to every production of G, a production of some appropriate tree-generating cfhg-grammar is associated. In [EH92] it is proved that cts have the same power as attribute grammars with respect to string-to-value translations.

In this paper we investigate the translation power of tree-generating td-tg transducers and, in particular, compare it to the power of another formalization of the concept of syntax-directed translation: the macro tree transducer. These are particular term rewriting systems in which the left-hand side and the right-hand side of every rule are trees over states, output symbols, and rewrite variables. Intuitively, macro tree transducers are top-down tree transducers in which the translation of a subtree of an input tree depends on its context. Let MT denote the class of tree-to-tree translations computed by macro tree transducers. We prove that tree-generating td-tg transducers have the same translation

power as macro tree transducers, i.e., $tgtT = MT$. (Thus, they are more powerful than cts.)

Before exposing the structure of this paper, we briefly discuss the two directions of the proof of our main theorem. The construction involved in the inclusion $MT \subseteq tgtT$ can be considered as the formalization of a graph-reduction for macro tree transducers. This is achieved by sharing common subtrees which occur at different places in the right-hand side of a rule of the macro tree transducer. The resulting graphs are particular hypergraphs called *parjungles*. Thus, the macro tree transducer is turned into a td-tg transducer in which the right-hand side of each rule is a parjungle. The sharing does not change the computed tree-to-tree translation, because we consider in this paper only macro tree transducers such that for every input tree s and state q, there is a unique q-translation of s. Hence, every occurrence of one subtree is eventually rewritten to the same tree over output symbols.

The main idea of the more involved proof of $tgtT \subseteq MT$ is the following: first, translate a tree-generating td-tg transducer M into a td-tg transducer M' which computes the same tree-to-tree translation as M and in which the right-hand sides of all rules are parjungles, and second, construct an equivalent macro tree transducer M'' by unfolding the right-hand sides of rules of M'. It turns out that, for the most natural construction of M' from M, M' has to be equipped with the possibility of inspecting the subtrees $s_1, \ldots, s_k$ when applying a rule $\langle q, \sigma(x_1, \ldots, x_k)\rangle \to h$ to an edge with label $\langle q, \sigma(s_1, \ldots, s_k)\rangle$. This capability is called *regular look-ahead*, because it suffices in fact to allow M' to test whether $s_1, \ldots, s_k$ belong to certain regular tree languages. This then leads to a macro tree transducer M'' which also uses regular look-ahead. For top-down tree transducers and macro tree transducers this feature of regular look-ahead is well-known to be useful [Eng77, EV85, FV89a, FV89b]. In fact, in [EV85] it is shown that regular look-ahead does not increase the translation power of macro tree transducers. Thus, M'' can be turned into an equivalent macro tree transducer without regular look-ahead.

This paper is organized in six sections. In Section 2 basic notations are collected, and trees and hypergraphs are defined (and, in particular, the representation of trees by (par)jungles). In Section 3 td-tg transducers and td-tg transducers with regular look-ahead are defined formally. We prove that, by adding regular look-ahead, td-tg transducers can be assumed to have properties which are useful in the proof of the inclusion $tgtT \subseteq MT$. In Section 4 macro tree transducers without and with regular look-ahead are recalled. In Section 5 the graph reduction of macro tree transducers is formalized by relating them to particular td-tg transducers; this proves the inclusion $MT \subseteq tgtT$. In Section 6 we prove the inclusion $tgtT \subseteq MT$.

2 Preliminaries

2.1 Notations

The empty set is denoted by $\emptyset$. For an arbitrary set A, the *powerset of A* is denoted by $\mathcal{P}(A)$. For $n \geq 0$, $[n] = \{1, \ldots, n\}$; in particular, $[0] = \emptyset$. $\mathbb{N}$ denotes the set $\{0, 1, 2, \ldots\}$ of natural numbers and $\mathbb{N}_+ = \mathbb{N} - \{0\}$ denotes the set of positive integers.

A word is a finite sequence. The empty word is denoted by λ. For a set A, A^* and A^+ denote the sets of words over A and of words over A with at least length 1, respectively. For a word $w = a_1 a_2 \ldots a_k \in A^+$ with $a_i \in A$ for $i \in [k]$, a_i is denoted by $w(i)$. The length of a word w is denoted by $lg(w)$.

The infinite set $Y = \{y_1, y_2, \ldots\}$ is called the set of *parameters* and the infinite set $X = \{x_1, x_2, \ldots\}$ is called the set of *subtree variables*. For $m \geq 0$, $Y_m = \{y_1, \ldots, y_m\}$ and $X_m = \{x_1, \ldots, x_m\}$.

Let v be a word and let $u_1, \ldots, u_n$ and $v_1, \ldots, v_n$ be two lists of words for some $n \geq 0$, such that no word occurs twice in the first list. If the words $u_1, \ldots, u_n$ occur in v without any overlapping, then $v[u_1/v_1, \ldots, u_n/v_n]$ is the word obtained from v by replacing every occurrence of u_i by v_i for every $i \in [n]$.

2.2 Ranked alphabets and trees

A *ranked set* is a tuple $(\Gamma, rank_\Gamma)$ where Γ is a (possibly infinite) set and $rank_\Gamma : \Gamma \to \mathbb{N}$ is a mapping; for every $k \geq 0$, $\Gamma^{(k)} = \{\gamma \in \Gamma | rank_\Gamma(\gamma) = k\}$. If Γ is known from the context, then it is dropped from $rank_\Gamma$. We also write $\gamma^{(k)}$ to denote the fact that γ has rank k. If A is a set, then $\langle \Gamma, A \rangle$ denotes the ranked set $\{\langle \gamma, a \rangle | \gamma \in \Gamma \text{ and } a \in A\}$ with $rank_{(\Gamma,A)}(\langle \gamma, a \rangle) = rank_\Gamma(\gamma)$. A *ranked alphabet* Γ is a finite ranked set.

Let Γ be a ranked alphabet. Then $dec(\Gamma)$ denotes the ranked alphabet $(\Gamma - \Gamma^{(0)}, rank')$ with $rank'(\gamma) = rank_\Gamma(\gamma) - 1$ for every $\gamma \in \Gamma - \Gamma^{(0)}$; $inc(\Gamma)$ denotes the ranked alphabet $(\Gamma, rank')$ with $rank'(\gamma) = rank_\Gamma(\gamma) + 1$.

Let Γ be a ranked set and let A be a set. The set of *(finite, labeled, and ordered) trees over Γ indexed by A*, denoted by $T\langle\Gamma\rangle(A)$, is the smallest set T such that (i) $A \subseteq T$ and (ii) if $\gamma \in \Gamma^{(k)}$ with $k \geq 0$ and $t_1, \ldots, t_k \in T$, then $\gamma(t_1, \ldots, t_k) \in T$. In case $k = 0$, we identify $\gamma()$ with γ. In particular, $T\langle\Gamma\rangle(\emptyset)$ is denoted by $T\langle\Gamma\rangle$. Thus, viewing the symbols of A as symbols of rank 0, $T\langle\Gamma\rangle(A) = T\langle\Gamma \cup A\rangle$. A set $L \subseteq T\langle\Gamma\rangle$ is called a *tree language*.

Let $t \in T\langle\Gamma\rangle$. The set of *subtrees* of t, denoted by $sub(t)$, is defined inductively as follows: for $t = \gamma(t_1, \ldots, t_k)$ with $\gamma \in \Gamma^{(k)}$, $k \geq 0$, and $t_1, \ldots, t_k \in T\langle\Gamma\rangle$, $sub(t) = \{t\} \cup \bigcup\{sub(t_i) | i \in [k]\}$. The *height* of t, denoted by $height(t)$, is defined by: $height(t) = 1 + max\{height(t_i) | i \in [k]\}$ if $t \notin \Gamma^{(0)}$ and $height(t) = 1$ if $t \in \Gamma^{(0)}$.

A *finite state (deterministic bottom-up) tree automaton (without final states)* is a tuple $B = (P, \Sigma, \delta)$ where P and Σ are finite sets of states and ranked input symbols, respectively, and $\delta = \{\delta_\sigma\}_{\sigma \in \Sigma}$ is the family of transition functions where $\delta_\sigma : P^k \to P$ for every $\sigma \in \Sigma^{(k)}$.

The transition function extends to a function $\widetilde{\delta} : T\langle\Sigma\rangle \to P$ by the following recursive definition [TW68, GS84]: for $\alpha \in \Sigma^{(0)}$, $\widetilde{\delta}(\alpha) = \delta_\alpha$, and for $\sigma \in \Sigma^{(k)}$ with $k \geq 1$ and $t_1, \ldots, t_k \in T\langle\Sigma\rangle$, $\widetilde{\delta}(\sigma(t_1, \ldots, t_k)) = \delta_\sigma(\widetilde{\delta}(t_1), \ldots, \widetilde{\delta}(t_k))$.

2.3 Hypergraphs

Let Γ be a ranked set. A *(directed, edge-labeled) hypergraph over* Γ is a tuple $g = (V, E, lab, nod, ext)$ where V is a finite set of *nodes* (or vertices), E is a finite set of *hyperedges* (or just edges), $lab : E \to \Gamma$ is the *edge labeling function*, $nod : E \to V^*$ is the *incidence function* such that, for every $e \in E$, $lg(nod(e)) = rank_\Gamma(lab(e))$, and $ext \in V^*$ is a word of *external nodes*. The nodes of V which do not occur in ext, are called *internal nodes*.

For a given hypergraph g, its components are denoted by V_g, E_g, lab_g, nod_g, and ext_g, respectively. Let $e \in E_g$ and $nod_g(e) = v_1 \ldots v_k$ with $v_1, \ldots, v_k \in V_g$. The *rank of* e, denoted by $rank_g(e)$, is k; if $k \geq 1$, then v_i with $i \in [k]$ is called *i-incident with e* or *e-incident*.

If $lg(ext_g) = k$, then g is called a *k-hypergraph* and is said to be of rank k, also denoted by $rank(g)$. For every ranked set Γ, the set of *(k-)hypergraphs over* Γ is denoted by $HGR(\Gamma)$ (k-$HGR(\Gamma)$, respectively). Two hypergraphs g, h over Γ are *disjoint* if $V_g \cap V_h = \emptyset$ and $E_g \cap E_h = \emptyset$.

Example 2.1 Consider the ranked alphabet $\Gamma = \{\delta^{(3)}, \sigma^{(2)}, \gamma^{(1)}, \alpha^{(0)}\}$. Figure 3 shows a 3-hypergraph g over Γ, where $V_g = \{v_1, v_2, v_3\}$, $E_g = \{e_1, e_2, e_3, e_4\}$, $lab_g(e_1) = \alpha$, $lab_g(e_2) = \delta$, $lab_g(e_3) = \sigma$, and $lab_g(e_4) = \gamma$, $nod_g(e_1) = \lambda$, $nod_g(e_2) = v_2 v_1 v_1$, $nod_g(e_3) = v_2 v_3$, and $nod_g(e_4) = v_2$, $ext_g = v_2 v_1 v_2$. Note that this is the same hypergraph as in Figure 1.

Nodes and edges are indicated by fat circles and boxes, respectively (which are sometimes labeled by the denotation of the corresponding nodes and edges). An edge e with $lg(nod_g(e)) = 2$ is also drawn as a directed line (as in usual graphs), e.g., $lg(nod_g(e_3)) = 2$. An external node v is indicated by labels $i_1, \ldots, i_r$, if $v = ext_g(i_1) = \ldots = ext_g(i_r)$, e.g., v_2 is labeled by 1 and 3, because $v_2 = ext_g(1) = ext_g(3)$. For every edge e, its label is shown inside the box which represents e. Connections between a box and a fat circle are called tentacles. A tentacle between edge e and node v is labeled by a small natural number i if v is i-incident with e; e.g., v_2 is 1-incident with e_2. □

Actually, we view a hypergraph h as an abstract graph which stands for the equivalence class of all concrete graphs that are isomorphic to h. However, in order to avoid technicalities, we deal with concrete graphs in all our definitions and constructions, taking isomorphic copies whenever necessary.

For hypergraphs with one edge only, we introduce the following notation. Let $\gamma \in \Gamma^{(m)}$ with $m \geq 0$. The *singular hypergraph labeled by* γ, denoted by $sing(\gamma)$, is the m-hypergraph $([m], \{e\}, lab, nod, ext)$ with $lab(e) = \gamma$ and $nod(e) = ext = 12 \ldots m$ (note that ext is a word of length m).

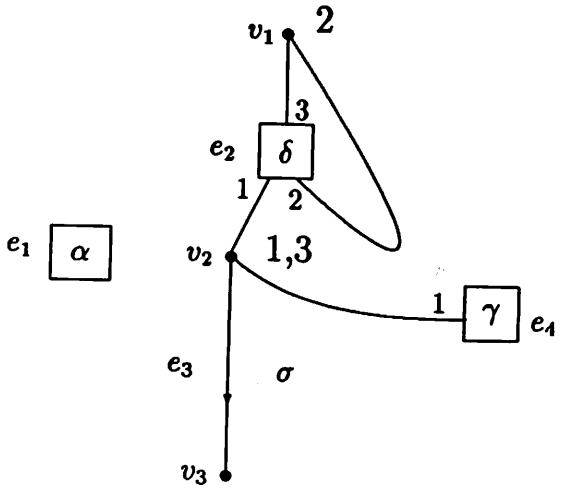


Figure 3: The hypergraph g over Γ.

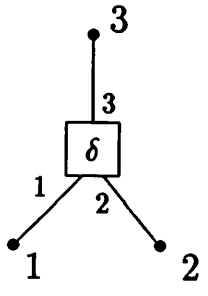


Figure 4: The 3-hypergraph $sing(\delta)$.

Example 2.2 Let $\delta \in \Gamma$ be a symbol of rank 3. The 3-hypergraph $sing(\delta)$ is shown in Figure 4. □

The identification of nodes in a hypergraph is formalized as follows. Let g be a hypergraph and let $R \subseteq V_g \times V_g$ be a binary relation over the set of nodes of g. Let $\equiv_R$ denote the smallest equivalence relation over V_g which contains R and let $\phi : V_g \to V_g/\equiv_R$ denote the canonical mapping to the set $V_g/\equiv_R$ of equivalence classes. Then g/R denotes the hypergraph $(V', E_g, lab_g, nod', ext')$ where $V' = V/\equiv_R$, for every $e \in E_g$ and $i \in [rank(e)]$, $nod'(e)(i) = \phi(nod_g(e)(i))$, and for every $i \in [rank(g)]$, $ext'(i) = \phi(ext_g(i))$.

Let g and h be hypergraphs. We say that h is a *subgraph* of g if $V_h \subseteq V_g$, $E_h \subseteq E_g$, and lab_h and nod_h are the restrictions of lab_g and nod_g, respectively, to E_h. Note that ext_h need not be equal to ext_g. We say that a subgraph h of g is a *full subgraph* of g if every internal node v of h is also an internal node of g, and, for every edge $e \in E_g$, if internal node v is incident with e, then $e \in E_h$.

2.4 Tree Representing Hypergraphs

There are particular hypergraphs that can be considered as a space efficient representation of trees. The tree can be recovered from such a hypergraph g by unfolding g, starting at a particular external node that represents the root of the tree. The determinacy and termination of this unfolding is guaranteed by certain requirements. Here we distinguish between two types of such particular hypergraphs, viz. jungles (that represent trees) and parjungles (that represent trees with parameters).

To represent trees over a ranked alphabet Γ, we consider hypergraphs over the ranked alphabet $inc(\Gamma)$. In fact, for $\gamma \in \Gamma^{(k)}$, a tree $t = \gamma(t_1, \ldots, t_k)$ will be represented by an edge e with $lab(e) = \gamma$ together with representations of $t_1, \ldots, t_k$; if $nod(e) = v_1 \ldots v_k v$, then v represents the root of t, and v_i represents the root of t_i, $i \in [k]$. This leads to the following formal definitions.

For a word $w = a_1 \ldots a_k \in A^+$, the set $ar(w)$ or *arguments* of w is the set $\{a_1, \ldots, a_{k-1}\}$, and the *result* of w, denoted by $res(w)$, is a_k. For a hypergraph g and an edge e of g with $nod_g(e) = v_1 \ldots v_k$, the set $ar_g(e)$ of *arguments* of e is the set $\{v_1, \ldots, v_{k-1}\}$, and the *result* of e, denoted by $res_g(e)$, is v_k. In other words, $ar_g(e) = ar(nod_g(e))$ and $res_g(e) = res(nod_g(e))$. For a node v of g, the cardinality of $res_g^{-1}(v)$ is called the *in-degree* of v. For a node v with in-degree 1, the unique edge in $res_g^{-1}(v)$ will also be denoted by $res_g^{-1}(v)$. For instance, in the hypergraph g shown in Figure 3, $ar_g(e_2) = \{v_1, v_2\}$ and $res_g(e_2) = v_1$. Moreover, every node of g has in-degree 1.

A *path of g from node v_0 to node v_k* is an element $v_0 e_1 v_1 \ldots e_k v_k$ of $V_g(E_g V_g)^*$, with $v_i \in V_g$ and $e_j \in E_g$, such that, for every $j \in [k]$, $v_{j-1} \in ar_g(e_j)$ and $res_g(e_j) = v_j$. Then g is *acyclic* if no path of g contains a node twice, more precisely, for every path $v_0 e_1 v_1 \ldots e_k v_k$ of g and for every $i, j \in \{0, \ldots, k\}$, if $i \neq j$, then $v_i \neq v_j$. Clearly, the hypergraph shown in Figure 3 is cyclic, because it contains the path $v_1 e_2 v_1$.

A *jungle* is an acyclic hypergraph of rank 1, of which every node has in-degree 1.

Example 2.3 Consider the ranked alphabet $\Gamma = \{\delta^{(3)}, \sigma^{(2)}, \gamma^{(1)}\}$. Figure 5 shows a jungle.

Jungles represent trees. But we also need hypergraphs which represent trees with parameters, called parjungles (standing for "jungles with parameters"). Recall that $Y = \{y_1, y_2, \ldots\}$ is the set of parameters, and for $m \geq 0$, $Y_m = \{y_1, \ldots, y_m\}$.

For $m \geq 0$, a *parjungle* with m parameters is an acyclic hypergraph g of rank $m + 1$, such that

(1) $ext_g(1), \ldots, ext_g(m)$ are all distinct,

(2) $ext_g(1), \ldots, ext_g(m)$ have in-degree 0, and

(3) every node $v \notin \{ext_g(1), \ldots, ext_g(m)\}$ has in-degree 1.

Note that $ext_g(m+1)$ has in-degree 0 if it is in $\{ext_g(1), \ldots, ext_g(m)\}$, but has in-degree 1 if it is not in $\{ext_g(1), \ldots, ext_g(m)\}$. Note also that a jungle is the same as a parjungle with 0 parameters.

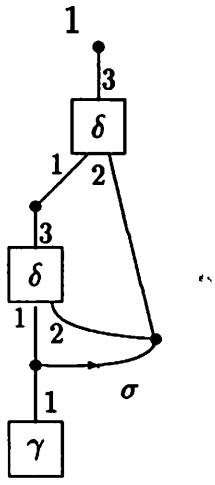


Figure 5: A jungle.

Let g be a parjungle with m parameters over Γ. The *tree represented by* g, denoted by $tree(g)$, is the tree $\tau(ext_g(m+1))$ where the mapping $\tau : V_g \to T\langle dec(\Gamma)\rangle(Y_m)$ is defined recursively by

(i) if v is an external node $ext_g(i)$ for some $i \in [m]$, then $\tau(v) = y_i$, and

(ii) otherwise $\tau(v) = \gamma(\tau(v_1), \ldots, \tau(v_p))$ where $\gamma = lab_g(res_g^{-1}(v))$ and $v_1 \ldots v_p v = nod_g(res_g^{-1}(v))$.

Note that, if g is a jungle, then $tree(g) \in T\langle dec(\Gamma)\rangle$.

The function τ is also refered to as the *unfolding function of* g. Note that τ prunes off the parts of g that are not connected with the external node $m+1$.

Example 2.4 Consider the ranked alphabet $\Gamma = \{\nu^{(4)}, \delta^{(3)}, \sigma^{(2)}, \gamma^{(1)}\}$. Figures 6 (a), (b), and (c) show parjungles g_1, g_2, and g_3 with 2 parameters, respectively. Obviously, the parjungles are only different with respect to the third external node. Let us compute, for g_1, g_2, and g_3, the represented tree.

$tree(g_1) = \tau_{g_1}(v_1) = \delta(\tau_{g_1}(v_2), \tau_{g_1}(v_4))$

$= \delta(\sigma(\tau_{g_1}(v_3)), \nu(\tau_{g_1}(v_3), \tau_{g_1}(v_5), \tau_{g_1}(v_6))$

$= \delta(\sigma(y_1), \nu(y_1, y_2, \gamma))$

Similarly, $\tau_{g_2}(v_2)$ and $\tau_{g_3}(v_3)$ are computed:

$tree(g_2) = \tau_{g_2}(v_2) = \sigma(y_1)$ and

$tree(g_3) = \tau_{g_3}(v_3) = y_1$.

Note finally that for the jungle g of Figure 5, $tree(g) = \delta(\delta(\gamma, \sigma(\gamma)), \sigma(\gamma))$.

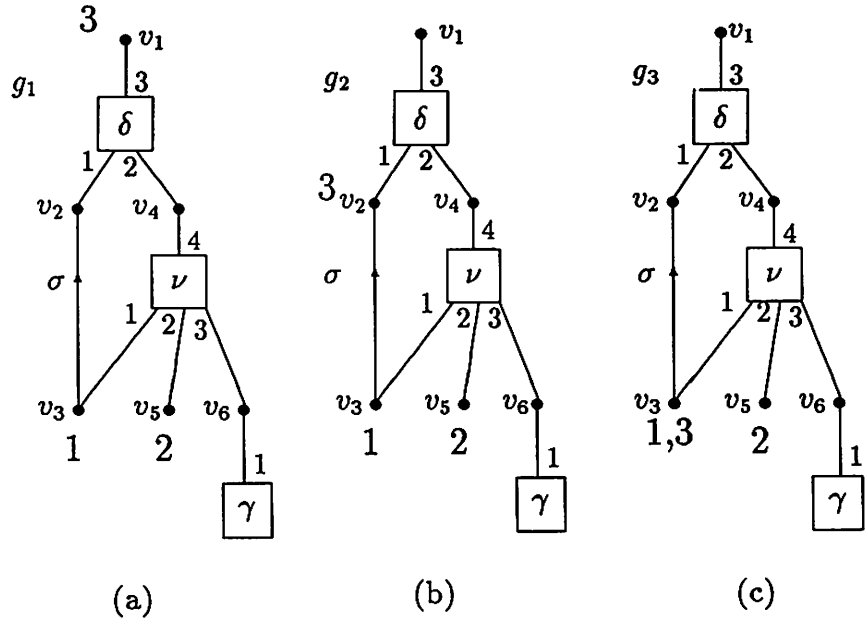


Figure 6: Parjungles g_1, g_2, and g_3.

3 Top-Down Tree-To-Graph Transducers

In this section we formally define the *top-down tree-to-graph transducer* and its tree-generating version. Moreover, we enrich this formalism by adding the possibility of checking the input tree by means of *regular look-ahead.* We prove a useful normal form for top-down tree-to-graph transducers, provided they are equipped with regular look-ahead.

Recall that $X = \{x_1, x_2, \ldots\}$ is the set of subtree variables, and that $X_k = \{x_1, \ldots, x_k\}$ for $k \geq 0$.

Definition 3.1 A *top-down tree-to-graph transducer* (for short: td-tg transducer) is a tuple $M = (Q, \Sigma, \Delta, q_{in}, R)$ where

- Q is a ranked alphabet of states
- Σ and Δ are ranked alphabets of input and output symbols, respectively
- $q_{in} \in Q$ is the initial state of rank 1
- R is a finite set of rules; for every $q \in Q^{(m)}$ with $m \geq 0$ and $\sigma \in \Sigma^{(k)}$ with $k \geq 0$ there is exactly one rule of the form

$$(*) \quad \langle q, \sigma(x_1, \ldots, x_k)\rangle \to h$$

where $h \in m\text{-}HGR(\langle Q, X_k\rangle \cup \Delta)$. □

The rule π of M of the form $\langle q, \sigma(x_1, \ldots, x_k)\rangle \to h$ is called the (q, σ)-rule of M and h is also denoted by $rhs(\pi)$ or $rhs(q, \sigma)$. Intuitively, the rule $\langle q, \sigma(x_1, \ldots, x_k)\rangle \to h$ expresses that the q-translation of an input tree $\sigma(s_1, \ldots, s_k)$ is the graph h in which every edge with label $\langle q', x_j\rangle$ is replaced by the q'-translation of s_j (cf. Lemma 3.13).

An *output labeled (state labeled) edge of* $rhs(\pi)$ is an edge that is labeled by some output symbol (pair $\langle q', x_j\rangle$ where q' is a state and x_j is a subtree variable, respectively).

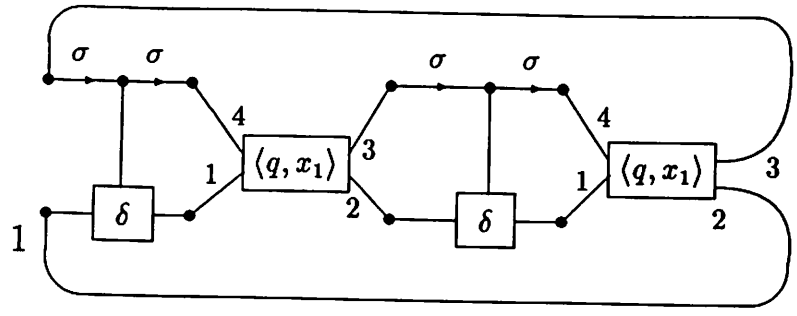
Example 3.2 (cf. [Lau88a, EH92]). Consider the td-tg transducer $M = (Q, \Sigma, \Delta, q_{in}, R)$ with $Q = \{q_{in}^{(1)}, q^{(4)}\}$, $\Sigma = \{\gamma^{(1)}, \alpha^{(0)}\}$, and $\Delta = \{\delta^{(3)}, \sigma^{(2)}\}$. The set $R = \{\pi_1, \pi_2, \pi_3, \pi_4\}$ of rules is shown in Figure 7.

For every input tree, the td-tg transducer M outputs two concentric circles which are connected by radial lines (cf. Figure 8). Actually, M translates $\gamma^n(\alpha)$ into such a hypergraph with 2^n connecting lines. □

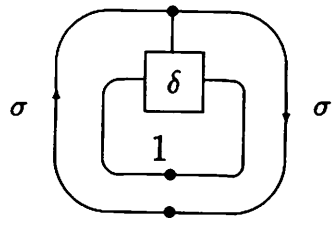
The restriction in Definition 3.1 that there is exactly one (q, σ)-rule for every q and σ, means that we consider "total deterministic" td-tg transducers. Without this restriction the nondeterministic td-tg transducer is obtained, which is not studied in this paper.

From the point of view of controlled grammars, td-tg transducers are context-free hypergraph grammars (cfhg grammars) of which the derivations are controlled by input trees.

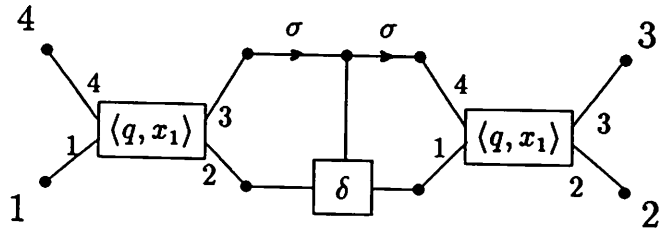
$\pi_1 = \langle q_{in}, \gamma(x_1) \rangle \rightarrow$



$\pi_2 = \langle q_{in}, \alpha \rangle \rightarrow$



$\pi_3 = \langle q, \gamma(x_1) \rangle \rightarrow$



$\pi_4 = \langle q, \alpha \rangle \rightarrow$

- 3,4
- 1,2

Figure 7: Rules of M.

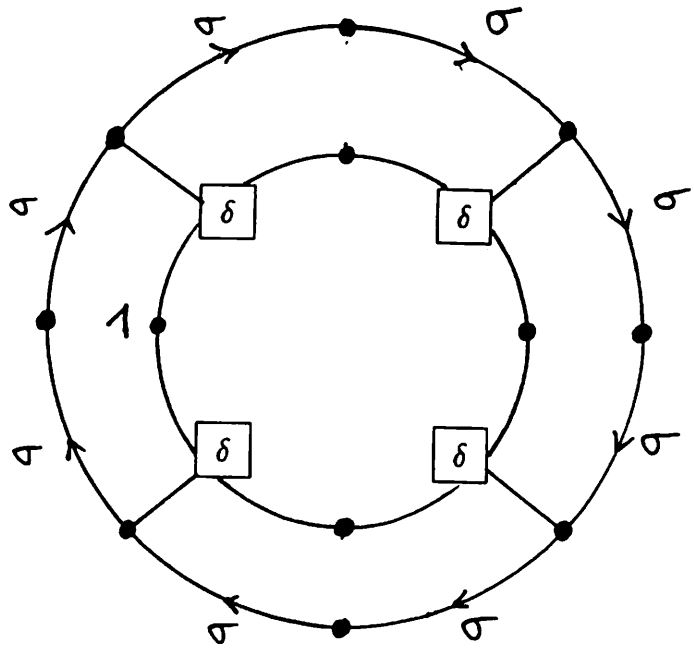


Figure 8: Translation of $\gamma(\gamma(\alpha))$ by M.

In this sense, (nondeterministic) td-tg transducers generalize $ET0L$-systems (investigated, e.g., in [GR75, Asv77, Lan83, ERS80, Eng76, Eng86a, Eng86b]) where a context-free grammar is controlled by strings of tables and tables are finite collections of productions. Thus, the generalization is twofold: we consider cfhg grammars and control trees rather than context-free grammars and control strings. The reader is refered to [Eng76, ERS80] for the fact that the parallelism inherent in $ET0L$ systems can be replaced by using a sequence of tables as distributed control (cf. [Eng86a] for a discussion of the concept of grammar with control or equivalently: with storage). By generalizing in just one direction, two other formalisms are obtained. Context-free grammars with control trees are essentially top-down tree transducers (see [Eng86a, ERS80]). Cfhg grammars with control strings are the appropriate generalization of $ET0L$ systems to graph grammars: they are cfhg grammars with parallel rewriting as introduced in [Kre92]. Generalizing the result of [Eng76], it is not difficult to prove that such parallel cfhg grammars generate the class of ranges of (nondeterministic) td-tg transducers that have a monadic input alphabet Σ (i.e., every symbol in Σ has rank 1 or 0).

The definition of derivation relation is prepared by the general notion of hypergraph substitution [BC87, HKS7a, HK87b] see Fig.2. Roughly speaking, in a hypergraph g, an edge e of rank m is replaced by a hypergraph h with m external nodes by pairwise identifying the nodes that are incident with e, with corresponding external nodes. (Put your fingertips together and think about it.)

Definition 3.3 For a ranked alphabet Γ, let $g \in n\text{-}HGR(\Gamma)$ with $n \geq 0$. Let $e \in E_g$ with rank $m \geq 0$ and let $h \in m\text{-}HGR(\Gamma)$ such that g and h are disjoint.

The *substitution of h for e in g*, denoted by $g[e/h]$, is the n-hypergraph f/R over Γ defined as follows:

- $V_f = V_h \cup V_g$
- $E_f = (E_g - \{e\}) \cup E_h$
- lab_f is $lab_h \cup lab_g$ restricted to E_f
- nod_f is $nod_h \cup nod_g$ restricted to E_f
- $ext_f = ext_g$, and
- $R = \{(u,v) \in V_f \times V_f | u = nod_g(e)(i)$ and $v = ext_h(i)$ for some $i \in [m]\}$. □

For a hypergraph g, a set $E' \subseteq E_g$ of edges, and a family $\{h(e)\}_{e \in E'}$ of hypergraphs, with $rank(h(e)) = rank(e)$ for every $e \in E'$, we define $g[e/h(e); e \in E']$ to be $g[e_1/h(e_1)] \ldots [e_r/h(e_r)]$ where $E' = \{e_1, \ldots, e_r\}$. It is an easy observation that the result of this simultaneous substitution does not depend on the order of the single substitutions. For a set $\Gamma' \subseteq \Gamma$ of symbols, and a family $\{h(\gamma)\}_{\gamma \in \Gamma'}$ of hypergraphs, with $rank(h(\gamma)) = rank(\gamma)$ for every $\gamma \in \Gamma'$, we define $g[\gamma/h(\gamma); \gamma \in \Gamma']$ to be $g[e/h(lab_g(e)); e \in E']$ where $E' = \{e \in E_g | lab_g(e) \in \Gamma'\}$.

Definition 3.4 Let $M = (Q, \Sigma, \Delta, q_{in}, R)$ be a td-tg transducer.

The *derivation relation* $\Rightarrow_M$ of M is a binary relation on $HGR(\langle Q, T\langle\Sigma\rangle\rangle \cup \Delta)$ such that, for every $\xi_1, \xi_2 \in HGR(\langle Q, T\langle\Sigma\rangle\rangle \cup \Delta)$, $\xi_1 \Rightarrow_M \xi_2$ iff there is a rule $\langle q, \sigma(x_1, \ldots, x_k)\rangle \to h$ in R, there is an edge $e \in E_{\xi_1}$, and there are $s_1, \ldots, s_k \in T\langle\Sigma\rangle$, such that

(a) $lab_{\xi_1}(e) = \langle q, \sigma(s_1, \ldots, s_k)\rangle$, and

(b) ξ_2 is isomorphic to $\xi_1[e/h']$ where h' is obtained from h by replacing every edge label $\langle q', x_j\rangle$ by the label $\langle q', s_j\rangle$ (and then, if necessary, taking an isomorphic copy disjoint with ξ_1). □

We note that, if ξ_1 is an n-hypergraph and $\xi_1 \Rightarrow \xi_2$, then ξ_2 is an n-hypergraph too. Sometimes it is useful to add particular information to the denotation of the derivation step: If rule $\pi = \langle q, \sigma(x_1, \ldots, x_k)\rangle \to h$ of M has been applied to edge e of ξ_1 resulting in the hypergraph ξ_2, then this step is also denoted by

$$\xi_1 \Rightarrow_{M,e,\pi} \xi_2.$$

Just as for right-hand sides of rules, we define an output labeled (state labeled) edge of a hypergraph $g \in HGR(\langle Q, T\langle\Sigma\rangle\rangle \cup \Delta)$ to be an edge of g that is labeled by an element of Δ (of $\langle Q, T\langle\Sigma\rangle\rangle$, respectively).

Example 3.5 Consider the td-tg transducer M of Example 3.2. Figure 9 shows a derivation of M starting with $sing(\langle q_{in}, \gamma(\alpha)\rangle)$. □

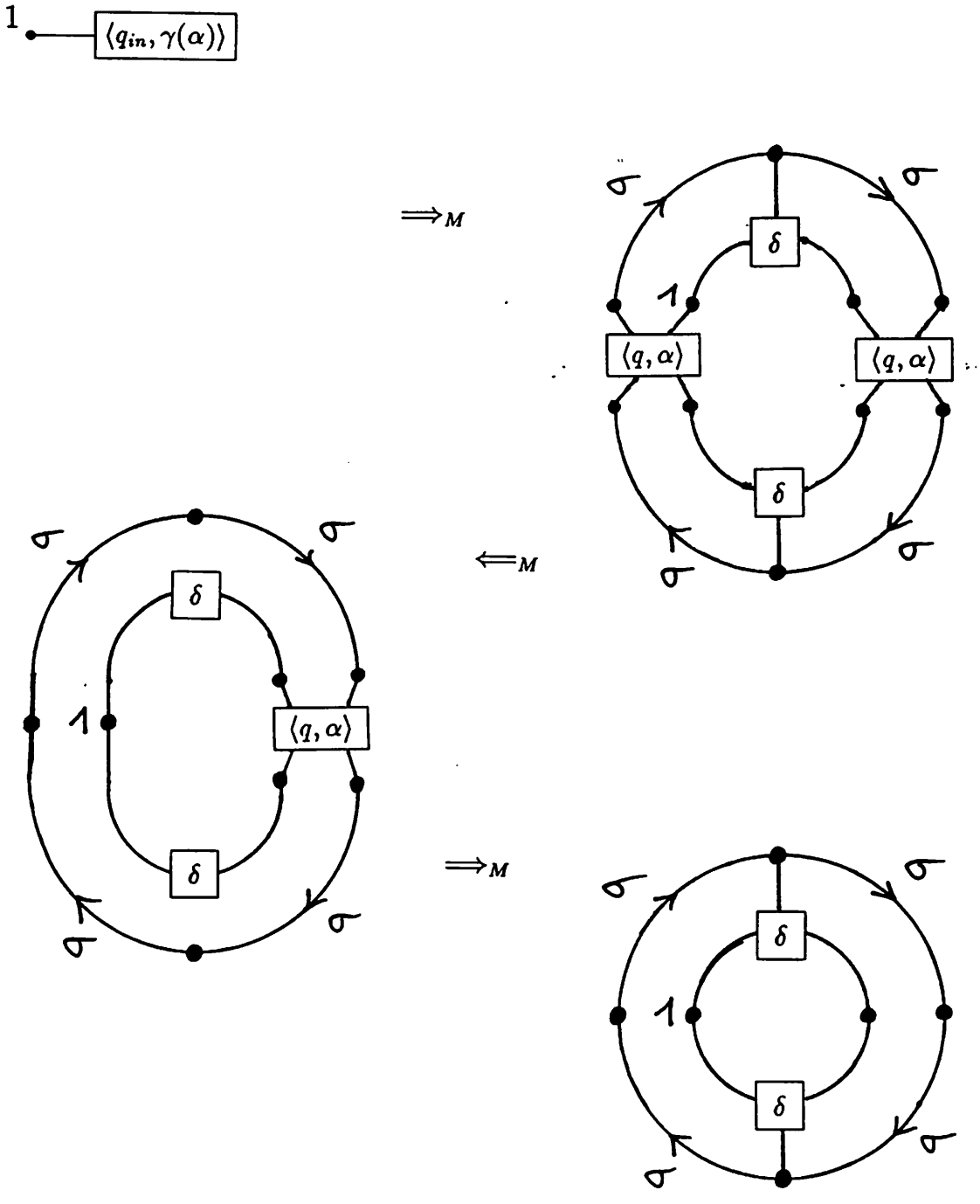


Figure 9: A derivation of M starting at $sing(\langle q_{in}, \gamma(\alpha)\rangle)$.

Before we define the tree-to-graph translation computed by a td-tg transducer, we first prove that the derivation relations of these devices are locally confluent (cf. [Hue80] for this notion) and noetherian, i.e., for every td-tg transducer M, there does not exist an infinite sequence $\xi_1 \Rightarrow_M \xi_2 \Rightarrow_M \xi_3 \Rightarrow_M \ldots$ for hypergraphs $\xi_1, \xi_2, \xi_3, \ldots$ (cf. [Der87] for a survey about termination of rewriting). Thus, $\Rightarrow_M$ is confluent.

Lemma 3.6 For every td-tg transducer M, $\Rightarrow_M$ is locally confluent.

Proof: Given a td-tg transducer M, it is an easy observation that, if $\xi \Rightarrow_M \xi_0$ and $\xi \Rightarrow_M \xi_1$ for hypergraphs ξ, ξ_0, ξ_1 with $\xi_0 \neq \xi_1$, then there is a hypergraph ξ' such that $\xi_i \Rightarrow_M \xi'$ for $i \in \{0,1\}$: Since ξ_0 and ξ_1 are obtained by applying two rules π_0 and π_1 of M to two distinct edges e_0 and e_1 of ξ, respectively, and these derivation steps do not interfere, ξ' can be obtained from ξ_i by applying rule π_{1-i} to e_{1-i}. In fact, $\xi_0[e_1/rhs(\pi_1)]$ $= \xi[e_0/rhs(\pi_0)][e_1/rhs(\pi_1)] = \xi[e_1/rhs(\pi_1)][e_0/rhs(\pi_0)] = \xi_1[e_0/rhs(\pi_0)] = \xi'$ (cf. the observation following Definition 3.3). Thus $\Rightarrow_M$ is locally confluent. □

Now we prove the property that derivation relations of td-tg transducers are noetherian. This property is based on the observation that in every derivation step an occurrence of an input subtree is replaced by finitely many (possibly zero) input subtrees with a smaller height; this induces a well founded ordering on multisets. Then it is easy to see that $\Rightarrow_M$ is also well founded, i.e., noetherian. For the formal proof we need the technical notions of partially ordered set, finite multiset, and multiset ordering.

A *partially ordered set* $(S, >)$ consists of a set S and a transitive and irreflexive binary relation $>$ on S. Let $>_{\mathbb{N}}$ be the usual partial order on natural numbers. A partially ordered set is *well founded* if there are no infinite descending sequences $s_1 > s_2 > \ldots$ of elements $s_1, s_2, \ldots$ of S. Clearly, $(\mathbb{N}, >_{\mathbb{N}})$ is well founded.

Intuitively, a multiset is a set in which elements may occur several times. Formally, for a set S, a *multiset over* S, is a mapping $M : S \to \mathbb{N}$. A multiset M over S is *finite* if the set $\{s \in S | M(s) \neq 0\}$ is finite. The *set of finite multisets over* S is denoted by $\mathcal{M}(S)$. Let $M_1, M_2 \in \mathcal{M}(S)$, then $M_1 + M_2$ is the multiset M such that $M(s) = M_1(s) + M_2(s)$ for every $s \in S$. If, for every $s \in S$, $M_1(s) \geq M_2(s)$, then M_2 is a *subset of* M_1, denoted by $M_2 \subseteq M_1$, and $M_1 - M_2$ is the multiset M, such that $M(s) = M_1(s) - M_2(s)$ for every $s \in S$. The *empty multiset over* S, denoted by $\emptyset$, satisfies $\emptyset(s) = 0$ for every $s \in S$.

If $(S, >)$ is a partially ordered set, then $\mathcal{M}(S)$ can be partially ordered. Intuitively, a multiset M' is smaller than the multiset M, if M' is obtained from M by replacing at least one element of M by any finite number of elements each of which is smaller than one of the elements that have been removed. Formally, the *multiset ordering* $\gg$ *induced by* $(S, >)$ *on* $\mathcal{M}(S)$ [DM79] is defined as follows: for $M, M' \in \mathcal{M}(S)$, $M \gg M'$ iff there are multisets $X, Y \in \mathcal{M}(S)$ such that $\emptyset \neq X \subseteq M$, $M' = (M - X) \cup Y$, and for every $y \in Y$ there is an $x \in X$ with $x > y$.

It is easy to prove that $(\mathcal{M}(S), \gg)$ is a partially ordered set iff $(S, >)$ is. Moreover, well-foundedness is preserved by constructing the multiset ordering.

Observation 3.7 ([DM79]). The multiset ordering $(\mathcal{M}(S), \gg)$ is well founded if and only if $(S, >)$ is well founded. □

Now we are prepared to prove that the derivation relations of td-tg transducers are noetherian. Let $\gg_{\mathbb{N}}$ be the multiset ordering induced by $(\mathbb{N}, >_{\mathbb{N}})$ on $\mathcal{M}(\mathbb{N})$.

Lemma 3.8 For every td-tg transducer M, $\Rightarrow_M$ is noetherian.

Proof: Let $M = (Q, \Sigma, \Delta, q_{in}, R)$ be a td-tg transducer. For sentential forms occurring in derivations of M, we define the following finite multisets over $\mathbb{N}$. For $\xi \in HGR(\langle Q, T\langle\Sigma\rangle\rangle \cup \Delta)$, let *multi-height*$(\xi)$ be the finite multiset over $\mathbb{N}$ such that, for every $k \in \mathbb{N}$, *multi-height*$(\xi)(k)$ is the number of edges e of ξ, such that $lab_\xi(e) = \langle q, s\rangle \in \langle Q, T\langle\Sigma\rangle\rangle$ and *height*$(s) = k$.

Let $\xi_1 \Rightarrow_{M,e,\pi} \xi_2$ for some edge $e \in E_{\xi_1}$. Hence, if e is labeled by $\langle q, \sigma(s_1, \ldots, s_k)\rangle$ for some state q and some $\sigma(s_1, \ldots, s_k) \in T\langle\Sigma\rangle$, then e is replaced by $rhs(\pi)$ in which labels of the form $\langle q', x_j\rangle$ are replaced by $\langle q', s_j\rangle$. Hence, *multi-height*$(\xi_1) \gg_{\mathbb{N}}$ *multi-height*(ξ_2).

Now assume that there is an infinite derivation $\xi_1 \Rightarrow_M \xi_2 \Rightarrow_M \ldots$. Then there is an infinite sequence of multisets such that *multi-height*$(\xi_1) \gg_{\mathbb{N}}$ *multi-height*$(\xi_2) \gg_{\mathbb{N}} \ldots$. But this contradicts the well-foundedness of $\gg_{\mathbb{N}}$ which is induced by the well-foundedness of $>_{\mathbb{N}}$ on $\mathbb{N}$ (by Observation 3.7). This proves that $\Rightarrow_M$ is noetherian. □

By Lemma 2.4 of [Hue80], it follows from Lemmas 3.6 and 3.8 that, for every td-tg transducer $M = (Q, \Sigma, \Delta, q_{in}, R)$, the relation $\Rightarrow_M$ is confluent. Since, in total, $\Rightarrow_M$ is confluent and noetherian, every sentential form ξ has a unique normal form, i.e., there is a unique hypergraph g with output labeled edges only such that $\xi \Rightarrow_M^* g$. In particular, this holds for $\xi = sing(\langle q, s\rangle)$ for some state q and input tree s.

Definition 3.9 Let $M = (Q, \Sigma, \Delta, q_{in}, R)$ be a td-tg transducer, $q \in Q$, and $s \in T\langle\Sigma\rangle$.

The *q-translation of* s, denoted by $M(q, s)$, is the unique hypergraph $g \in HGR(\Delta)$, such that $sing(\langle q, s\rangle) \Rightarrow_M^* g$. □

Note that, if q has rank m, then $sing(\langle q, s\rangle)$ is an m-hypergraph. Since the derivation relation preserves the rank of hypergraphs, also $M(q, s)$ has rank m.

Definition 3.10 Let $M = (Q, \Sigma, \Delta, q_{in}, R)$ be a td-tg transducer.

M is *tree-generating* if, for every $s \in T\langle\Sigma\rangle$, the 1-hypergraph $M(q_{in}, s)$ is a jungle. □

Definition 3.11 Let $M = (Q, \Sigma, \Delta, q_{in}, R)$ be a td-tg transducer.

(a) The *tree-to-graph translation computed by* M is the mapping $\tau(M) : T\langle\Sigma\rangle \to HGR(\Delta)$, such that $\tau(M)(s) = M(q_{in}, s)$ for every $s \in T\langle\Sigma\rangle$. □

(b) If M is tree-generating, then the *tree-(to-graph)-to-tree translation computed by* M is the mapping $\tau_t(M) : T\langle\Sigma\rangle \to T\langle dec(\Delta)\rangle$, such that $\tau_t(M)(s) = tree(M(q_{in}, s))$ for every $s \in T\langle\Sigma\rangle$. □

Example 3.12 Consider the td-tg transducer $M = (Q, \Sigma, \Delta, q_{in}, R)$ with $Q = \{q_{in}^{(1)}, q^{(2)}\}$, $\Sigma = \{\gamma^{(1)}, \alpha^{(0)}\}$, and $\Delta = \{\delta^{(3)}, \gamma^{(1)}\}$. The rules $\pi_1, \pi_2, \pi_3, \pi_4$ of M are shown in Figure 10.

For the input tree $\gamma^n(\alpha)$, M computes the jungle $mon(n) = (\{0, \ldots, 2^n\}, \{e_i | 0 \leq i \leq 2^n\}, lab, nod, ext)$ with $lab(e_0) = \gamma$ and $nod(e_0) = 0$, and for every $1 \leq i \leq 2^n$, $lab(e_i) = \delta$ and $nod(e_i) = (i-1)(i-1)i$, $ext = 2^n$ (cf. Figure 11). That means, $\tau(M)(\gamma^n(\alpha)) = mon(n)$. Clearly, the unfolding of $mon(n)$ yields a full binary tree $bin(n)$ over $dec(\Delta)$ of height $2^n + 1$. Hence $tree(mon(n)) = bin(n)$, and so $\tau_t(M)(\gamma^n(\alpha)) = bin(n)$. □

Two td-tg transducers M_1 and M_2 are *equivalent* if they compute the same tree-to-graph translation, i.e., $\tau(M_1) = \tau(M_2)$. The class of all tree-to-graph translations computed by td-tg transducers is denoted by tgT. For tree-generating td-tg transducers, the class of computed tree-to-tree translations is denoted by $tgtT$. It is straightforward to prove that the translation of an input tree by a td-tg transducer M can be characterized inductively as follows (the proof needs the associativity of hypergraph substitution, see [Cou87a]).

Lemma 3.13 Let $M = (Q, \Sigma, \Delta, q_{in}, R)$ be a td-tg transducer. For every $q \in Q$, $\sigma \in \Sigma^{(k)}$ with $k \geq 0$, and $s_1, \ldots, s_k \in T\langle\Sigma\rangle$,

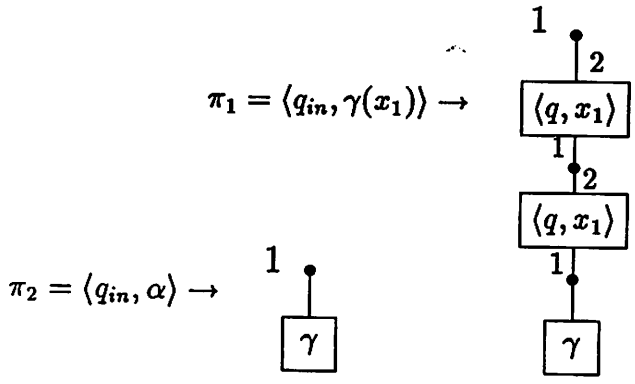
$$M(q, \sigma(s_1, \ldots, s_k)) = rhs(q, \sigma)[\langle q', x_j\rangle / M(q', s_j); \langle q', x_j\rangle \in \langle Q, X_k\rangle].$$

□

In the remainder of this section we prove a normal form result for td-tg transducers M (with regular look-ahead). This normal form is motivated by our wish that M has the "subgraph property", i.e., that, in the equation of Lemma 3.13, every $M(q', s_j)$ is (isomorphic to) a subgraph of $M(q, \sigma(s_1, \ldots, s_k))$. This may not be true if $rhs(q, \sigma)$ contains a "loop", i.e., an edge e with label $\langle q', x_j\rangle$ such that $nod_{rhs(q,\sigma)}(e)$ contains a repetition of nodes (because then some external nodes of $M(q', s_j)$ are possibly identified in $M(q, \sigma(s_1, \ldots, s_k))$). For this reason we will require M to be "loop-free".

Definition 3.14 Let $M = (Q, \Sigma, \Delta, q_{in}, R)$ be a td-tg transducer. An edge e of a hypergraph h is *loop-free* if for every $i, j \in [rank(e)]$, $i \neq j$ implies $nod_h(e)(i) \neq nod_h(e)(j)$. A hypergraph h in $HGR(\langle Q, T\langle\Sigma\rangle\rangle \cup \Delta)$ or in $HGR(\langle Q, X\rangle \cup \Delta)$ is *loop-free* if all the state labeled edges of h are loop-free. M is *loop-free* if $rhs(\pi)$ is loop-free for every $\pi \in R$. □

But even for a loop-free td-tg transducer it may not be true that every $M(q', s_j)$ is (isomorphic to) a subgraph of $M(q, \sigma(s_1, \ldots, s_k))$. This may go wrong if the sequence of external nodes of $M(q', s_j)$ contains a repetition of nodes (because then some of the nodes of $rhs(q, \sigma)$ that are incident with a $\langle q', x_j\rangle$-labeled edge are possibly identified in $M(q, \sigma(s_1, \ldots, s_k))$, possibly leading to a loop in another state labeled edge). For this reason we will require each right-hand side h of a rule of M to be "identification-free", which means that ext_h is a sequence of distinct nodes.



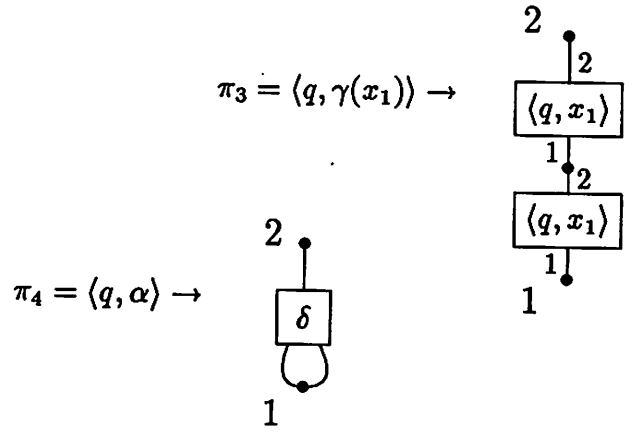


Figure 10: Rules of M.

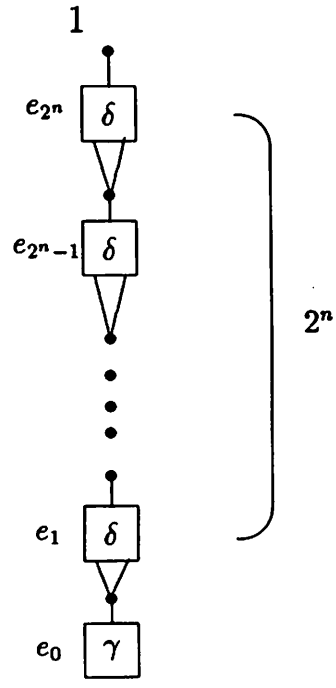


Figure 11: The jungle $mon(n)$.

Definition 3.15 Let $M = (Q, \Sigma, \Delta, q_{in}, R)$ be a td-tg transducer. A hypergraph h is *identification-free* if, for every $i, j \in [rank(h)]$, $i \neq j$ implies $ext_h(i) \neq ext_h(j)$. M is *identification-free* if $rhs(\pi)$ is identification-free for every $\pi \in R$. □

For instance, the hypergraph g shown in Figure 3 is not loop-free, because $nod_g(e_2)(2) = nod_g(e_2)(3) = v_1$, and not identification-free, because $ext_g(1) = ext_g(3) = v_2$.

Now we show that td-tg transducers which are loop-free and identification-free, have the "subgraph property". For the notion of full subgraph, see Section 2.3.

Lemma 3.16 Let $M = (Q, \Sigma, \Delta, q_{in}, R)$ be a loop-free and identification-free td-tg transducer. Then:

(1) for every $q \in Q$ and $s \in T\langle\Sigma\rangle$, $M(q, s)$ is identification-free, and

(2) for every $q \in Q$, $\sigma \in \Sigma^{(k)}$ with $k \geq 0$, and $s_1, \ldots, s_k \in T\langle\Sigma\rangle$, if $\langle q', x_j\rangle$ occurs in $rhs(q, \sigma)$, then $M(q', s_j)$ is a full subgraph of $M(q, \sigma(s_1, \ldots, s_k))$.

Proof: Statement (1) can be shown by induction on the structure of s, using Lemma 3.13. It is easy to see in general that if g and h are identification-free, then so is $g[e/h]$. Statement (2) then follows from Lemma 3.13 and the general fact that if g is loop-free, e is a state labeled edge of g, and h is identification-free, then h is a subgraph of $g[e/h]$ and $g[e/h]$ is loop-free. Note that, in general, if h is a subgraph of $g[e/h]$, then it is a full subgraph of $g[e/h]$. □

It is not clear to us whether every td-tg transducer can be transformed into an equivalent identification-free td-tg transducer, because the information about identification depends on the input tree. However, if we enrich td-tg transducers with some capability of inspecting its current input tree before applying a rule, then we can transform M into an equivalent, identification-free td-tg transducer. The capability which is sufficient to reach this normal form of td-tg transducers, is called regular look-ahead (cf. [Eng77, FV89a, FV89b] for top-down tree transducers with regular look-ahead, and cf. [EV85] for macro tree transducers with regular look-ahead).

Definition 3.17 A *top-down tree-to-graph transducer with regular look-ahead* (for short: td-tg^R transducer) is a tuple $M = (Q, P, \Sigma, \Delta, q_{in}, R, \delta)$ where

- (P, Σ, δ) is a finite state tree automaton, called the *look-ahead automaton of* M, and
- $(Q, \Sigma, \Delta, q_{in}, R)$ is a td-tg transducer in which the rules now have the form

$$(\langle q, \sigma(x_1, \ldots, x_k)\rangle, p_1, \ldots, p_k) \to h$$

with q, σ, and h as in Definition 3.1, and $p_1, \ldots, p_k \in P$. Moreover, for every $q \in Q^{(m)}$, $\sigma \in \Sigma^{(k)}$, $p_1, \ldots, p_k \in P$ there is exactly one rule in R with left-hand side $(\langle q, \sigma(x_1, \ldots, x_k)\rangle, p_1, \ldots, p_k)$. □

The definition of the derivation relation of a td-tg^R transducer is exactly the same as the definition of the derivation relation of a usual td-tg transducer (cf. Definition 3.4) with the following restriction: the rule which is applied, has to reflect in its look-ahead states the properties of the subtrees $s_1, \ldots, s_k$ of the current input tree s.

Definition 3.18 Let $M = (Q, P, \Sigma, \Delta, q_{in}, R, \delta)$ be a td-tg^R transducer.

The *derivation relation* $\Rightarrow_M$ of M is a binary relation on $HGR(\langle Q, T\langle\Sigma\rangle\rangle \cup \Delta)$ such that, for every $\xi_1, \xi_2 \in HGR(\langle Q, T\langle\Sigma\rangle\rangle \cup \Delta)$, $\xi_1 \Rightarrow_M \xi_2$ iff there is a rule $(\langle q, \sigma(x_1, \ldots, x_k)\rangle, p_1, \ldots, p_k) \to h$ in R, there is an edge $e \in E_{\xi_1}$, and there are $s_1, \ldots, s_k \in T\langle\Sigma\rangle$, such that

(a) $lab_{\xi_1}(e) = \langle q, \sigma(s_1, \ldots, s_k)\rangle$ and, for every $i \in [k]$, $\widetilde{\delta}(s_i) = p_i$, and

(b) as in Definition 3.4. □

Since Lemma's 3.6 and 3.8 also hold for td-tg^R transducers M, we can define $M(q, s)$ as for td-tg transducers in Definition 3.9, and also take over Definitions 3.10 and 3.11. The analogue of Lemma 3.13 is as follows, where we use $rhs(q, \sigma, p_1, \ldots, p_k)$ to denote the right-hand side of the unique rule with the left-hand side $(\langle q, \sigma(x_1, \ldots, x_k)\rangle, p_1, \ldots, p_k)$.

Lemma 3.19 Let $M = (Q, P, \Sigma, \Delta, q_{in}, R, \delta)$ be a td-tg^R transducer. For every $q \in Q$, $\sigma \in \Sigma^{(k)}$ with $k \geq 0$, and $s_1, \ldots, s_k \in T\langle\Sigma\rangle$,

$$M(q, \sigma(s_1, \ldots, s_k)) =$$
$$rhs(q, \sigma, \widetilde{\delta}(s_1), \ldots, \widetilde{\delta}(s_k))[\langle q', x_j\rangle / M(q', s_j); \langle q', x_j\rangle \in \langle Q, X_k\rangle].$$

□

Definitions 3.14 and 3.15 and Lemma 3.16 are also valid for td-tg^R transducers.

An identification of external nodes $ext(i)$ and $ext(j)$ can be represented by a pair $(i,j) \in \mathbb{N} \times \mathbb{N}$. For a hypergraph g of rank m, define $id(g) = \{(i,j) \in [m] \times [m] | ext_g(i) = ext_g(j)\}$: the *identification information* of g. The next elementary lemma relates identification and substitution.

Lemma 3.20 Let $h, h_1, \ldots, h_r$ be hypergraphs, and let $e_1, \ldots, e_r$ be distinct edges of h. Then

(1) $id(h[e_1/h_1] \ldots [e_r/h_r]) = id(h/S)$, and

(2) $h[e_1/h_1] \ldots [e_r/h_r] = (h/S)[e_1/h_1] \ldots [e_r/h_r]$,

where $S = \{(u,v) | u = nod_h(e_i)(c)$ and $v = nod_h(e_i)(d)$ for some $1 \leq i \leq r$ and some $c, d \in id(h_i)\}$. □

We now show, for every td-tg transducer, that there is a finite state tree automaton which computes, for every input tree s and every state q, the identification information of $M(q,s)$. Let $ID = \mathcal{P}(\mathbb{N} \times \mathbb{N})$ be the set of all identification sets. For a ranked set Q, let $[Q \to ID]$ denote the (finite) set of mappings ϕ from Q to ID, such that, for every $q \in Q$ with rank m, $\phi(q) \subseteq [m] \times [m]$.

Lemma 3.21 Let $M = (Q, \Sigma, \Delta, q_{in}, R)$ be a td-tg transducer. There is a finite state tree automaton $B_M = (P, \Sigma, \delta)$, such that $P = [Q \to ID]$ and, for every $s \in T\langle\Sigma\rangle$ and $q \in Q$, $\widetilde{\delta}(s)(q) = id(M(q,s))$.

Proof: The transition function δ is defined as follows. Let $\sigma \in \Sigma^{(k)}$ for some $k \geq 0$, $\phi_1, \ldots, \phi_k \in [Q \to ID]$, and let $q \in Q$. Then let $rhs(q,\sigma)/(\phi_1, \ldots, \phi_k)$ denote the hypergraph h/S where $h = rhs(q,\sigma)$ and $S = \{(u,v) \in V_h \times V_h |$ there is an edge $e \in E_h$ with $lab_h(e) = \langle q', x_j \rangle$ and there is a $(c,d) \in \phi_j(q')$, such that $u = nod_h(e)(c)$ and $v = nod_h(e)(d)\}$. We define $\delta_\sigma(\phi_1, \ldots, \phi_k)(q) = id(rhs(q,\sigma)/(\phi_1, \ldots, \phi_k))$.

The statement of the lemma follows by induction on the structure of s by using the inductive characterization of the translation of a td-tg transducer (Lemma 3.13) and the fact that $id(rhs(q,\sigma)/(\widetilde{\delta}(s_1), \ldots, \widetilde{\delta}(s_k))) = id(rhs(q,\sigma)[\langle q', x_j \rangle / M(q', s_j); \langle q', x_j \rangle \in \langle Q, X_k \rangle])$, i.e., with respect to identification of external nodes,

$$rhs(q,\sigma)/(\widetilde{\delta}(s_1), \ldots, \widetilde{\delta}(s_k))$$

and

$$rhs(q,\sigma)[\langle q', x_j \rangle / M(q', s_j); \langle q', x_j \rangle \in \langle Q, X_k \rangle]$$

are indistinguishable. This fact follows directly from Lemma 3.20(1), with $h = rhs(q,\sigma)$ and $h_i = M(q', s_j)$. □

Example 3.22 Consider the td-tg transducer $M = (Q, \Sigma, \Delta, q_{in}, R)$ where $Q = \{q_{in}^{(1)}, p^{(3)}, q^{(2)}\}$, and Σ and Δ are defined as in Example 3.12. Figure 12 shows some

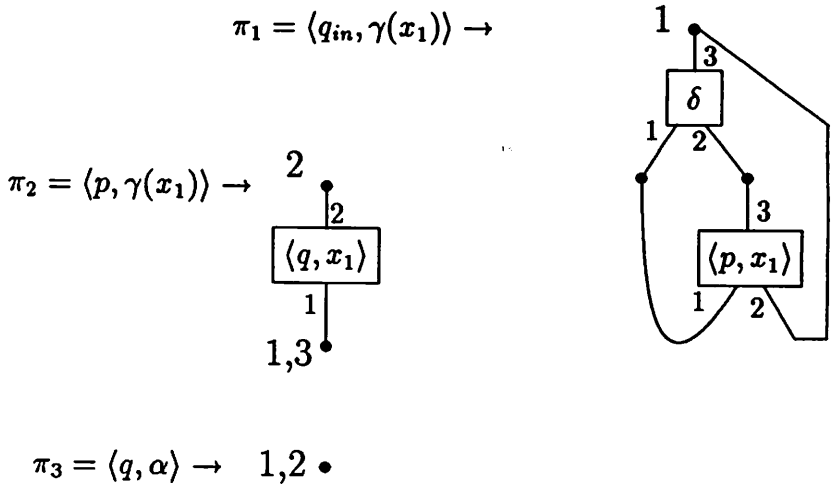


Figure 12: Some rules of M.

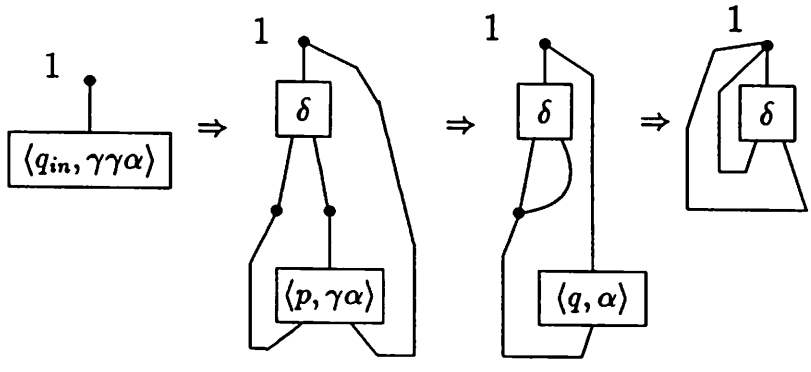
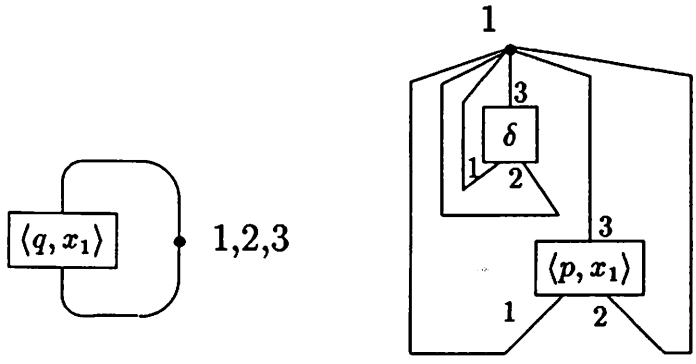


Figure 13: A derivation of M starting with $sing(\langle q_{in}, \gamma(\gamma(\alpha))\rangle)$.



(a) (b)

Figure 14: Right-hand sides with identified external nodes.

of the rules of M; the others are not important here. Figure 13 shows a derivation of M starting with $sing(\langle q_{in}, \gamma(\gamma(\alpha))\rangle)$.

Now let us indicate how B_M computes on $\gamma(\gamma(\alpha))$.

$\tilde{\delta}(\alpha)(q) = \delta_\alpha()(q) =$

$= id(rhs(q,\alpha)/()) = \{(1,2),(2,1),(1,1),(2,2)\} = [2] \times [2].$

$\tilde{\delta}(\gamma(\alpha))(p) = \delta_\gamma(\tilde{\delta}(\alpha))(p) =$

$= id(rhs(p,\gamma)/(\tilde{\delta}(\alpha))) = [3] \times [3].$

The graph $rhs(p,\gamma)/(\tilde{\delta}(\alpha))$ is shown in Figure 14(a).

$\tilde{\delta}(\gamma(\gamma(\alpha)))(q_{in}) = \delta_\gamma(\tilde{\delta}(\gamma(\alpha)))(q_{in}) =$

$= id(rhs(q_{in},\gamma)/(\tilde{\delta}(\gamma(\alpha)))) = [1] \times [1].$

The graph $rhs(q_{in},\gamma)/(\tilde{\delta}(\gamma(\alpha)))$ is shown in Figure 14(b). □

To prove that every td-tg transducer can be transformed into an equivalent identification-free td-tg^R transducer we use the same technique as it was used in the proof of Lemma 3.2 of [EH91] for cfhg grammars. However, here we do not guess identification information, attach it to the nonterminals of the grammar, and later on verify (or falsify) that it is correct, but rather we use look-ahead to determine the correct identification information directly.

Lemma 3.23 For every td-tg transducer there is an equivalent identification-free td-tg^R transducer.

Proof: Let $M = (Q, \Sigma, \Delta, q_{in}, R)$ be a td-tg transducer. First, we construct a so-called dynamically identification-free td-tg^R transducer M' which is equivalent to M, and second, from M' we construct an identification-free td-tg^R transducer M'' with $\tau(M) = \tau(M'')$. Note that, for every $\sigma \in \Sigma$, $rhs(q_{in},\sigma)$ is trivially identification-free, because it is a hypergraph with one external node.

A td-tg^R transducer M' is dynamically identification-free if the following holds: for all hypergraphs h and h' such that $sing(\langle q_{in}, s\rangle) \Rightarrow^*_{M'} h \Rightarrow_{M',e,\pi} h'$ for some $s \in T\langle\Sigma\rangle$, and for every $i, j \in [rank(rhs(\pi))]$, if $ext_{rhs(\pi)}(i) = ext_{rhs(\pi)}(j)$, then $nod_h(e)(i) = nod_h(e)(j)$. In other words, the application of π does not identify distinct nodes of h.

Construct the td-tg^R transducer $M' = (Q, P, \Sigma, \Delta, q_{in}, R', \delta)$ where (P, Σ, δ) is the finite state tree automaton B_M from Lemma 3.21, and R' is defined as follows. Let $\langle q, \sigma(x_1, \ldots, x_k)\rangle \to h$ be a rule in R. Then for every $\phi_1, \ldots, \phi_k \in [Q \to ID]$, the rule $(\langle q, \sigma(x_1, \ldots, x_k)\rangle, \phi_1, \ldots, \phi_k) \to h'$ is in R' where $h' = h/S$ and S is defined as in the proof of Lemma 3.21, i.e., $S = \{(u, v) |$ there is an edge $e \in E_h$ with $lab_h(e) = \langle q', x_j\rangle$ and there is a $(c, d) \in \phi_j(q')$, such that $u = nod_h(e)(c)$ and $v = nod_h(e)(d)\}$.

It is straightforward to show by induction on the structure of s that $M'(q, s) = M(q, s)$ for every $q \in Q$, using Lemma 3.13, Lemma 3.19, and Lemma 3.20(2). It remains to prove that M' is dynamically identification-free. First, it can easily be shown by induction on the length of the derivation $sing(\langle q_{in}, s\rangle) \Rightarrow^*_{M'} h$ that, for every edge e of h with label $\langle q, s'\rangle$, if $(i, j) \in id(M(q, s'))$, then $nod_h(e)(i) = nod_h(e)(j)$. In fact, this follows directly from the construction of M'. Now consider $h \Rightarrow_{M',e,\pi} h'$, and let edge e have the label $\langle q, s'\rangle$. If $ext_{rhs(\pi)}(i) = ext_{rhs(\pi)}(j)$, then, by Lemma 3.19, $(i, j) \in id(M'(q, s')) = id(M(q, s'))$; hence, by the previous fact, $nod_h(e)(i) = nod_h(e)(j)$.

For the construction of M'' we define for every hypergraph g the hypergraph $split(g)$ which is an arbitrary, but fixed identification-free hypergraph of the same rank as g, such that $split(g)/S = g$ where $S = \{(ext_{split(g)}(i), ext_{split(g)}(j)) | (i, j) \in id(g)\}$. Intuitively, such a $split(g)$ is obtained by splitting identified external nodes of g. Now construct the identification-free td-tg^R transducer $M'' = (Q, P, \Sigma, \Delta, q_{in}, R'', \delta)$, such that, if $(\langle q, \sigma(x_1, \ldots, x_k)\rangle, \phi_1, \ldots, \phi_k) \to h$ is in R', then $(\langle q, \sigma(x_1, \ldots, x_k)\rangle, \phi_1, \ldots, \phi_k) \to split(h)$ is in R''. The correctness of M'' is obvious from the fact that M' is dynamically identification-free. □

Example 3.24 Consider the td-tg transducer M of Example 3.22. Clearly, M is not identification-free, because $rhs(\pi_2)$ and $rhs(\pi_3)$ are hypergraphs which are not identification-free. Figure 15 and Figure 16 show the dynamically identification-free td-tg^R transducer M' (with $\phi_{\gamma(\alpha)}(p) = [3] \times [3]$ and $\phi_\alpha(q) = [2] \times [2]$) and the identification-free td-tg^R transducer M'' (with $\phi_{\gamma(\alpha)}$ and ϕ_α as in Figure 15), respectively, as constructed in Lemma 3.23. Figure 17 shows a derivation of M'' starting with $sing(\langle q_{in}, \gamma(\gamma(\alpha))\rangle)$. □

Every identification-free td-tg^R transducer $M = (Q, \Sigma, \Delta, q_{in}, R)$ can be transformed into an equivalent identification-free and loop-free td-tg^R transducer. This statement also holds if the regular look-ahead is dropped from M and M'. Consider a hypergraph h which has been derived from $sing(\langle q_{in}, s\rangle)$ for some input tree s, and consider an edge e of h which is labeled by $\langle q, s'\rangle$ for some state q and subtree s' of s. Moreover, assume that e has a "loop", i.e., there are $i, j \in [rank(e)]$ with $i \neq j$ such that $nod_h(e)(i) = nod_h(e)(j)$. A general observation about the derivation relation of a td-tg transducer is the fact that nodes never split. Now consider any hypergraph $g \in HGR(\Delta)$ which has been derived from $sing(\langle q, s'\rangle)$. It follows from the general observation that external nodes i and j are identified, i.e., $ext_g(i)$ and $ext_g(j)$ are identified in $h[e/g]$. Thus, we could just as well join

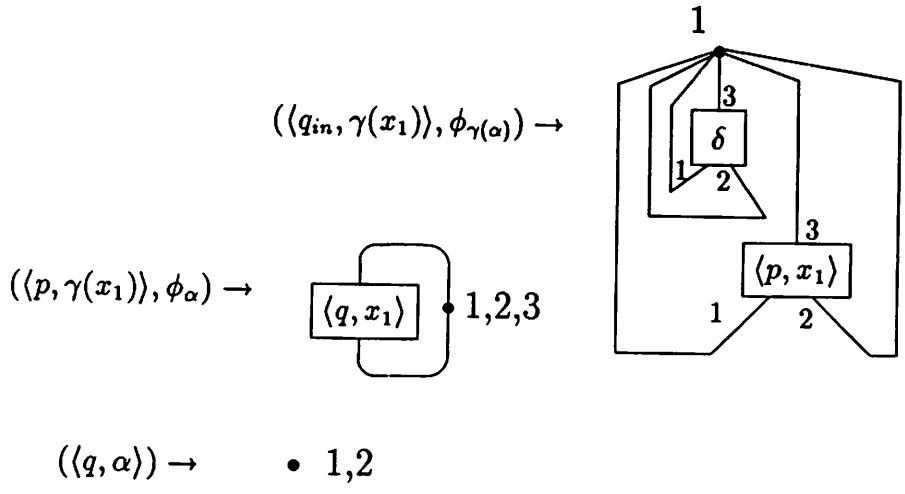


Figure 15: Dynamically identification-free td-tg^R transducer M'.

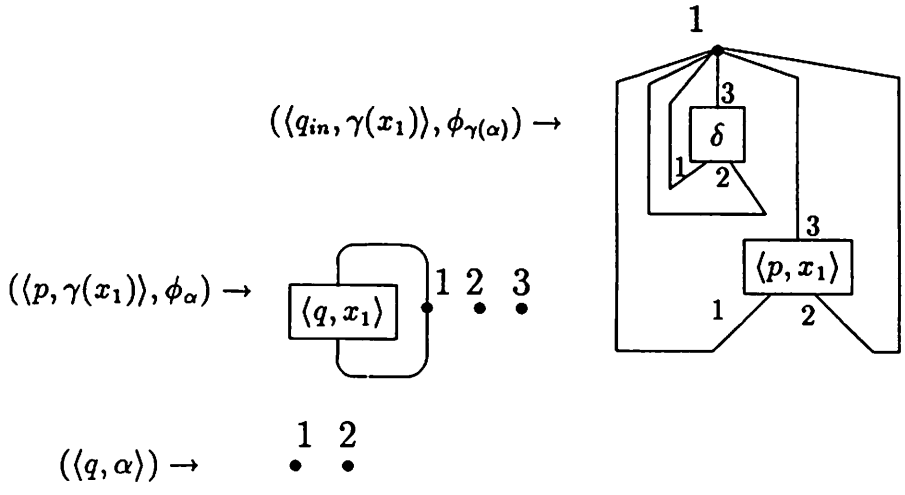


Figure 16: Identification-free td-tg^R transducer M''.

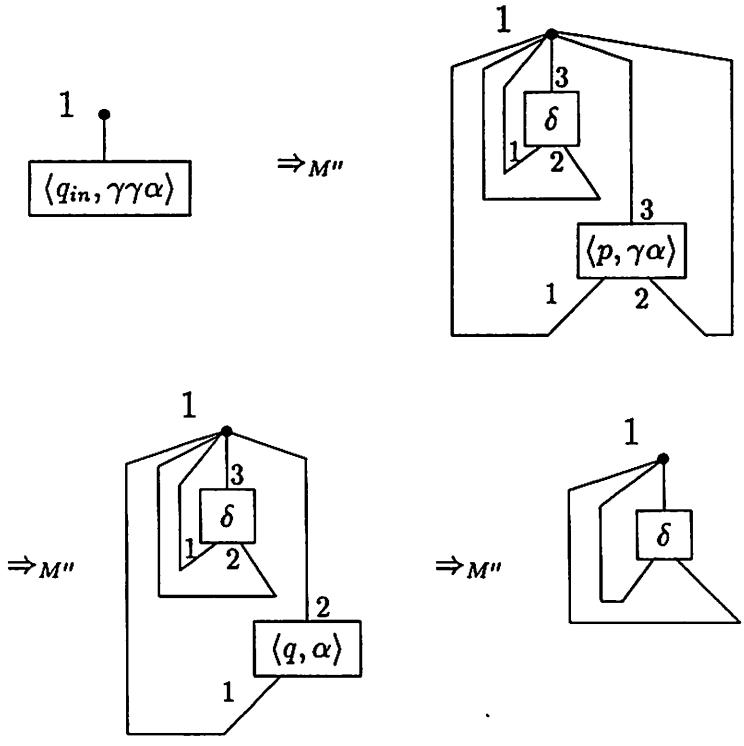


Figure 17: A derivation of M''.

the tentacles i and j of the edge e into one tentacle (cf. Theorem I.4.6 of [Hab89] and Proposition 2.4 of [EH92]).

Lemma 3.25 For every identification-free td-tg transducer M, there is an equivalent identification-free and loop-free td-tg transducer M'. The same statement holds if M and M' are td-tg^R transducers.

Proof: Let $M = (Q, \Sigma, \Delta, q_{in}, R)$ be an identification-free td-tg transducer. We construct an identification-free and loop-free td-tg transducer $M' = (Q', \Sigma, \Delta, q'_{in}, R')$ with $\tau(M') = \tau(M)$.

Every new state in Q' contains a partition of the set of the original tentacles with the intention that tentacles in one set lead to the same node. The elements of the partition are called *loop sets.*

$Q' = \{\langle q, n, L_1, \ldots, L_n\rangle | q \in Q, n \geq 0, \{L_1, \ldots, L_n\}$ is a partition of $[rank_Q(q)]\}$ with $rank_{Q'}(\langle q, n, L_1, \ldots, L_n\rangle) = n$. The initial state q'_{in} of M' is $\langle q_{in}, 1, \{1\}\rangle$.

For $A = \langle q, n, L_1, \ldots, L_n\rangle \in Q'$, we denote q as $LAB(A)$, n as $NUM(A)$, and for every $a \in [n]$, L_a as $L(A, a)$.

For the construction of R' consider the rule $\langle q, \sigma(x_1, \ldots, x_k)\rangle \to h$ in R and let $A \in Q'$ with $LAB(A) = q$.

First, from h and A, we derive for every state labeled edge of h its new label which contains the list of appropriate loop sets. For this purpose consider the hypergraph $H = h/P_A$ where $P_A = \{(ext_h(i), ext_h(j))|$ there is an $a \in [NUM(A)]$ such that $\{i, j\} \subseteq L(A, a)\}$. Let e be a state labeled edge of H, with $lab_H(e) = \langle q', x_j\rangle \in \langle Q, X\rangle$ and $r_e = rank(q')$. Let L_e be the partition of $[r_e]$ corresponding to the equivalence relation $\equiv_{A,e}$ on $[r_e]$ defined by $c \equiv_{A,e} d$ iff $nod_H(e)(c) = nod_H(e)(d)$. Let n_e be the number of sets in L_e and let $(L_{e1}, L_{e2}, \ldots, L_{en_e})$ be an enumeration of elements of L_e (in any order). Then the new label of edge e (under the assumptions recorded in A) is $new(A, e) = \langle\langle q', n_e, L_{e1}, L_{e2}, \ldots, L_{en_e}\rangle, x_j\rangle$.

Then R' contains the rule $\langle A, \sigma(x_1, \ldots, x_k)\rangle \to H'$ where $H' = (V, E, nod, lab, ext)$ with $V = V_H$, $E = E_H$, and for every $e \in V_H$ the following holds:

- if $lab_H(e) \in \Delta$, then $nod(e) = nod_H(e)$ and $lab(e) = lab_H(e) = lab_h(e)$,
- if $lab_H(e) \in \langle Q, X\rangle$, then for every $a \in [n_e]$, $nod(e)(a) = nod_H(e)(c)$ for some $c \in L_{ea}$ and $lab(e) = new(A, e)$, and
- for every $a \in [NUM(A)]$, $ext(a) = ext_H(c)$ for some $c \in L(A, a)$.

Note that, in both cases, there is no need to specify c further, because $nod_H(e)(c) = nod_H(e)(d)$ if $\{c, d\} \subseteq L_{ea}$, and $ext_H(c) = ext_H(d)$ if $\{c, d\} \subseteq L(A, a)$.

Every right-hand side of a rule $\langle A, \sigma(x_1, \ldots, x_k)\rangle \to H'$ of M' is identification-free, because the right-hand side of the corresponding rule $\langle LAB(A), \sigma(x_1, \ldots, x_k)\rangle \to h$ of M is identification-free and external nodes $ext_h(c)$ and $ext_h(d)$ that are identified in $H = h/P_A$ due to the fact that $\{c, d\} \subseteq L(A, a)$ for some a, are turned into one external node in

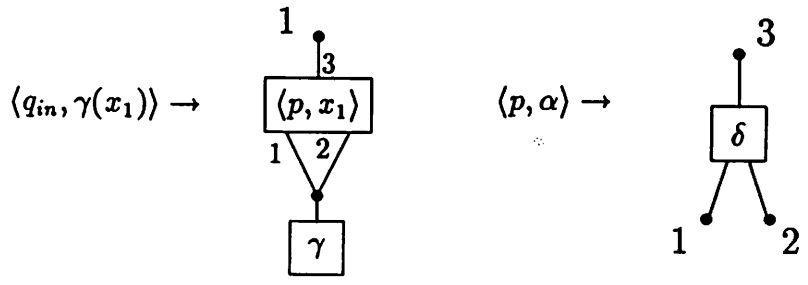


Figure 18: Some rules of M.

H', viz., $ext_{H'}(a)$. Thus, since for every a, b with $a \neq b$, $L(A,a) \cap L(A,b) = \emptyset$, M' is identification-free.

Also, M' is loop-free, because, for every rule $\langle A, \sigma(x_1, \ldots, x_k)\rangle \to H'$ in R', the following holds for every state labeled edge e of the right-hand side of the corresponding rule $\langle LAB(A), \sigma(x_1, \ldots, x_k)\rangle \to h$ of R: the elements of the partition L_e are pairwise disjoint and all tentacles that are in the same element L_{ea} of L_e (i.e., that are in the same equivalence class with respect to $\equiv_{A,e}$ and thus form loops) are turned into one tentacle a of e in H'.

Since M is identification-free, the following equivalence can easily be shown by induction on the length of the derivations. For every $s \in T\langle\Sigma\rangle$ and $g \in HGR(\langle Q, T\langle\Sigma\rangle\rangle \cup \Delta)$, $sing(\langle q_{in}, s\rangle) \Rightarrow_M^* g$ iff there exists a $G \in HGR(\langle Q', T\langle\Sigma\rangle\rangle \cup \Delta)$ such that $sing(\langle q'_{in}, s\rangle) \Rightarrow_{M'}^* G$ and g is (isomorphic to) $(V_G, E_G, nod, lab, ext_G)$ with, for every output labeled edge e, $lab(e) = lab_G(e)$ and $nod(e) = nod_G(e)$, and for every state labeled edge e, if $lab_G(e) = \langle A, s\rangle$ then $lab(e) = \langle LAB(A), s\rangle$ and, for every $i \in [rank_Q(LAB(A))]$, $nod(e)(i) = nod_G(e)(a)$ for the a such that $i \in L(A,a)$.

In particular, if $g \in HGR(\Delta)$, then g and G are equal. Thus, $\tau(M) = \tau(M')$.

Clearly, the addition of regular look-ahead does not interfere with the construction. □

Example 3.26 Consider the identification-free td-tg transducer $M = (Q, \Sigma, \Delta, q_{in}, R)$ where $Q = \{q_{in}^{(1)}, p^{(3)}\}$, $\Sigma = \{\gamma^{(1)}, \alpha^{(0)}\}$, $\Delta = \{\delta^{(3)}, \gamma^{(1)}\}$, and some of the rules are given in Figure 18. Obviously, M is not loop-free, because $rhs(q_{in}, \gamma)$ contains a loop. Figure 19 shows a derivation of M starting with $sing(\langle q_{in}, \gamma(\alpha)\rangle)$.

The identification-free, loop-free td-tg transducer M' which is constructed from M as defined in Lemma 3.25, is shown in Figure 20. Recall from Definition 3.14 that loop-freeness only refers to state labeled edges. Figure 21 shows a derivation of M' starting with $sing(\langle\langle q_{in}, 1, \{1\}\rangle, \gamma(\alpha)\rangle)$. □

Finally, we combine the previous lemmas.

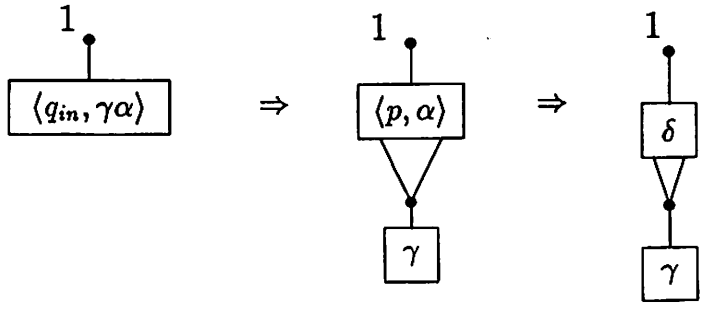


Figure 19: A derivation of M starting with $sing(\langle q_{in}, \gamma(\alpha)\rangle)$.

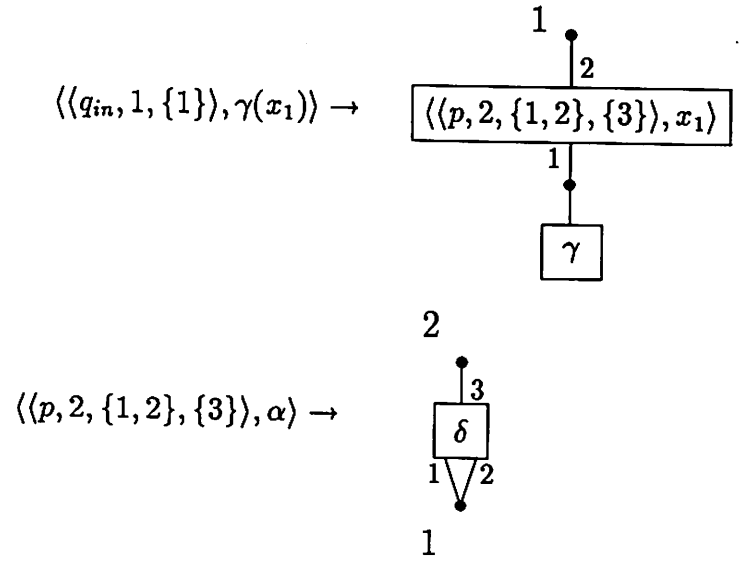


Figure 20: Identification-free, loop-free td-tg transducer M'.

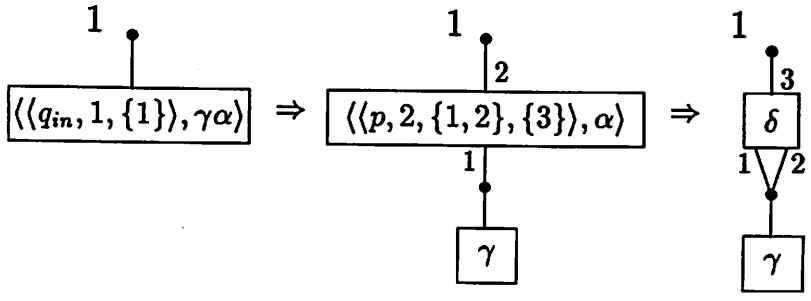


Figure 21: A derivation of M' starting with $sing(\langle\langle q_{in}, 1, \{1\}\rangle, \gamma(\alpha)\rangle)$.

Theorem 3.27 *For every td-tg transducer M there is an equivalent td-tg^R transducer M' which is identification-free and loop-free.* □

Note that M' satisfies Lemma 3.16. Note also that if M is tree-generating, then so is M'.

4 Macro Tree Transducers

In this section we briefly recall the concept of *macro tree transducer* [CF82, Eng80, EV85]. These are devices for syntax-directed translation in which the translation of an input tree may depend on its context.

Recall that $Y = \{y_1, y_2, \ldots\}$ is the set of parameters, and that $Y_m = \{y_1, \ldots, y_m\}$ for $m \geq 0$; $X = \{x_1, x_2, \ldots\}$ is the set of subtree variables, and $X_m = \{x_1, \ldots, x_m\}$.

Definition 4.1 A *macro tree transducer* (for short: mt transducer) is a tuple $M = (Q, \Sigma, \Delta, q_{in}, R)$ where Q is the ranked alphabet of *states*, Σ and Δ are the ranked alphabets of *input* and *output symbols*, respectively, $q_{in} \in Q^{(0)}$ is the *initial state*, and R is the finite set of *rules*; for every $q \in Q^{(m)}$ with $m \geq 0$ and $\sigma \in \Sigma^{(k)}$ with $k \geq 0$, there is exactly one rule in R of the form

$$(*) \qquad \langle q, \sigma(x_1, \ldots, x_k)\rangle(y_1, \ldots, y_m) \to \zeta$$

where $\zeta \in T\langle \Gamma \cup \Delta\rangle(Y_m)$ with $\Gamma = \langle Q, X_k\rangle$. □

We note that in [EV85] such transducers are called total deterministic macro tree transducers. A rule π of the form $(*)$ is called *the* (q, σ)*-rule of* M; its right-hand side is denoted by $rhs(\pi)$ or $rhs(q, \sigma)$. A *top-down tree transducer* is a macro tree transducer in which every state has rank 0.

Definition 4.2 Let $M = (Q, \Sigma, \Delta, q_{in}, R)$ be an mt transducer.

The *derivation relation of* M, denoted by $\Rightarrow_M$, is the binary relation on $T\langle \Gamma \cup \Delta\rangle$ with $\Gamma = \langle Q, T\langle\Sigma\rangle\rangle$ such that, for every $\xi_1, \xi_2 \in T\langle \Gamma \cup \Delta\rangle$, $\xi_1 \Rightarrow_M \xi_2$ iff

1. there is a $\xi \in T\langle \Gamma \cup \Delta\rangle(\{z\})$ in which z occurs exactly once
2. there are $\sigma \in \Sigma^{(k)}$, $q \in Q^{(m)}$, $s_1, \ldots, s_k \in T\langle\Sigma\rangle$, and $t_1, \ldots, t_m \in T\langle \Gamma \cup \Delta\rangle$ such that
   - $\xi_1 = \xi[z/\langle q, \sigma(s_1, \ldots, s_k)\rangle(t_1, \ldots, t_m)]$ and
   - $\xi_2 = \xi[z/\zeta']$ with $\zeta' = rhs(q, \sigma)[x_j/s_j; j \in [k]][y_i/t_i; i \in [m]]$. □

Example 4.3 Consider the mt transducer $M = (Q, \Sigma, \Delta, q_{in}, R)$ with $Q = \{q_{in}^{(0)}, q^{(1)}\}$, $\Sigma = \{\gamma^{(1)}, \alpha^{(0)}\}$, and $\Delta = \{\delta^{(2)}, \gamma^{(0)}\}$. R contains the following rules.

$$\begin{aligned}
\langle q_{in}, \gamma(x_1)\rangle &\to \langle q, x_1\rangle(\langle q, x_1\rangle(\gamma)) \\
\langle q_{in}, \alpha\rangle &\to \gamma \\
\langle q, \gamma(x_1)\rangle(y_1) &\to \langle q, x_1\rangle(\langle q, x_1\rangle(y_1)) \\
\langle q, \alpha\rangle(y_1) &\to \delta(y_1, y_1)
\end{aligned}$$

A derivation of M starting with $\langle q_{in}, \gamma(\alpha)\rangle$ is as follows:

$\langle q_{in}, \gamma(\alpha)\rangle$

$\Rightarrow \langle q, \alpha\rangle(\langle q, \alpha\rangle(\gamma))$

$\Rightarrow \delta(\langle q, \alpha\rangle(\gamma), \langle q, \alpha\rangle(\gamma))$

$\Rightarrow \delta(\delta(\gamma, \gamma), \langle q, \alpha\rangle(\gamma))$

$\Rightarrow \delta(\delta(\gamma, \gamma), \delta(\gamma, \gamma))$.

M translates $\gamma^n\alpha$ into the full binary tree $bin(n)$ of height $2^n + 1$ (cf. Example 3.12).

□

For every mt transducer $M = (Q, \Sigma, \Delta, q_{in}, R)$ and sentential form $\xi \in T\langle\Gamma \cup \Delta\rangle$ with $\Gamma = \langle Q, T\langle\Sigma\rangle\rangle$, there is a unique tree t over Δ such that $\xi \Rightarrow_M^* t$ (by Corollary 3.13 and the remarks at the beginning of Section 4.1 of [EV85]). In particular, for $q \in Q^{(m)}$ with $m \geq 0$ and $s \in T\langle\Sigma\rangle$, we denote by $M(q, s)$ the *q-translation of s* which is the unique tree $t \in T\langle\Delta\rangle(Y_m)$ such that $\langle q, s\rangle(y_1, \ldots, y_m) \Rightarrow_M^* t$, where the definition of $\Rightarrow_M$ is extended in the obvious way to a binary relation on $T\langle\Gamma \cup \Delta\rangle(Y)$ with $\Gamma = \langle Q, T\langle\Sigma\rangle\rangle$.

Definition 4.4 Let $M = (Q, \Sigma, \Delta, q_{in}, R)$ be an mt transducer.

The *tree-to-tree translation computed by* M, denoted by $\tau(M)$, is the mapping of type $T\langle\Sigma\rangle \to T\langle\Delta\rangle$, such that $\tau(M)(s) = M(q_{in}, s)$ for every $s \in T\langle\Sigma\rangle$. □

The class of tree-to-tree translations computed by mt transducers is denoted by MT. Next we recall the inductive characterization of the translation $M(q, s)$. For this purpose we first formalize the concept of second-order term substitution [Cou83] which is just a reformulation of the notion of tree homomorphism [GS84].

Definition 4.5 Let Σ and Δ be two ranked alphabets. For a set $\Sigma' \subseteq \Sigma$ and a family $\{t(\sigma)\}_{\sigma\in\Sigma'}$ of trees with $t(\sigma) \in T\langle\Delta\rangle(Y_{rank(\sigma)})$, we define the tree $s[\![\sigma/t(\sigma); \sigma \in \Sigma']\!]$ inductively on the structure of $s \in T\langle\Sigma\rangle$ as follows (abbreviating $s[\![\sigma/t(\sigma); \sigma \in \Sigma']\!]$ by $s[\![...]\!]$): if $s = \sigma(s_1, \ldots, s_k)$ for some $\sigma \in \Sigma^{(k)}$ with $k \geq 0$, then

1. if $\sigma \notin \Sigma'$, then $s[\![...]\!] = \sigma(s_1[\![...]\!], \ldots, s_k[\![...]\!])$, and
2. if $\sigma \in \Sigma'$, then $s[\![...]\!] = t(\sigma)[y_i/s_i[\![...]\!]; i \in [k]]$. □

Lemma 4.6 (cf. Definition 3.18 of [EV85]). Let $M = (Q, \Sigma, \Delta, q_{in}, R)$ be an mt transducer. For every $q \in Q$, $\sigma \in \Sigma^{(k)}$ with $k \geq 0$, and $s_1, \ldots, s_k \in T\langle\Sigma\rangle$,

$$M(q, \sigma(s_1, \ldots, s_k)) = rhs(q, \sigma)[\![\langle q', x_j\rangle/M(q', s_j); \langle q', x_j\rangle \in \langle Q, X_k\rangle]\!].$$

□

Note that the inductive characterization of $M(q,s)$ looks exactly the same as the one for td-tg transducers (cf. Lemma 3.13) except that $[\![...]\!]$ is used rather than [...].

In the simulation of a tree-generating td-tg transducer by an mt transducer we shall use a particular property of mt transducers: they are closed under regular look-ahead.

Definition 4.7 A *macro tree transducer with regular look-ahead* (for short: mt^R transducer) is a tuple $M = (Q, P, \Sigma, \Delta, q_{in}, R, \delta)$ where

- (P, Σ, δ) is a finite state tree automaton, called the *look-ahead automaton of* M, and
- $(Q, \Sigma, \Delta, q_{in}, R)$ is an mt transducer where the rules now have the form

$$(**) \qquad (\langle q, \sigma(x_1, \ldots, x_k)\rangle(y_1, \ldots, y_m), p_1, \ldots, p_k) \to \zeta$$

with q, σ, ζ as in Definition 4.1, and $p_1, \ldots, p_k \in P$. Moreover, for every $q \in Q^{(m)}$, $\sigma \in \Sigma^{(k)}$, $p_1, \ldots, p_k \in P$ there is exactly one rule in R with left-hand side $(\langle q, \sigma(x_1, \ldots, x_k)\rangle(y_1, \ldots, y_m), p_1, \ldots, p_k)$. □

The unique rule with left-hand side $(\langle q, \sigma(x_1, \ldots, x_k)\rangle(y_1, \ldots, y_m), p_1, \ldots, p_k)$ is called the $(q, \sigma, p_1, \ldots, p_k)$-rule of M and its right-hand side is denoted by $rhs(q, \sigma, p_1, \ldots, p_k)$.

The derivation relation of M is defined as in Definition 4.2, with $rhs(q,\sigma)$ replaced by $rhs(q, \sigma, \widetilde{\delta}(s_1), \ldots, \widetilde{\delta}(s_k))$.

For every mt^R transducer $M = (Q, P, \Sigma, \Delta, q_{in}, R, \delta)$ and sentential form $\xi \in T\langle \Gamma \cup \Delta\rangle$ with $\Gamma = \langle Q, T\langle\Sigma\rangle\rangle$, there is a unique tree t over Δ such that $\xi \Rightarrow_M^* t$ (cf. Remark 4.19 of [EV85]). Thus, for every $q \in Q^{(m)}$ with $m \geq 0$ and $s \in T\langle\Sigma\rangle$, there is a unique tree $t \in T\langle\Delta\rangle(Y_m)$ such that $\langle q, s\rangle(y_1, \ldots, y_m) \Rightarrow_M^* t$; again we denote this tree by $M(q,s)$.

The tree-to-tree translation $\tau(M)$ computed by M is defined as in Definition 4.4. The class of translations computed by mt^R transducers is denoted by MT^R. Also for mt^R transducers we can provide an inductive characterization of the translations computed by them (cf. Remark 4.19 of [EV85]).

Lemma 4.8 Let $M = (Q, P, \Sigma, \Delta, q_{in}, R, \delta)$ be an mt^R transducer. For every $q \in Q$, $\sigma \in \Sigma^{(k)}$ with $k \geq 0$, and $s_1, \ldots, s_k \in T\langle\Sigma\rangle$,

$$M(q, \sigma(s_1, \ldots, s_k)) = rhs(q, \sigma, \widetilde{\delta}(s_1), \ldots, \widetilde{\delta}(s_k))[\![\langle q', x_j\rangle / M(q', s_j); \langle q', x_j\rangle \in \langle Q, X_k\rangle]\!].$$

□

In Section 6 we will use the fact that mt transducers are closed under regular look-ahead.

Lemma 4.9 (Theorem 4.21 of [EV85]). $MT = MT^R$.

5 Graph-Reduction for Macro Tree Transducers

The aim of this section is the proof of the inclusion $MT \subseteq tgtT$, i.e., for every mt transducer there is a tree-generating M' such that $\tau(M) = \tau_t(M')$. We approach this aim by first defining a relationship between mt transducers and td-tg transducers in which the right-hand sides of rules are parjungles. Recall the definition of a parjungle h and the tree $tree(h)$ it represents from Section 2.4.

Definition 5.1 Let $M = (Q, \Sigma, \Delta, q_{in}, R)$ be a td-tg transducer such that the right-hand side of every rule is a parjungle. Let $M' = (dec(Q), \Sigma, dec(\Delta), q_{in}, R')$ be an mt transducer.

M and M' are *related* if $R' = \{\langle q, \sigma(x_1, \ldots, x_k)\rangle(y_1, \ldots, y_m) \to tree(h) \mid \langle q, \sigma(x_1, \ldots, x_k)\rangle \to h \text{ is in } R\}$. □

Obviously, the td-tg transducer of Example 3.12 and the mt transducer of Example 4.3 are related.

Definition 5.2 Let $M = (Q, P, \Sigma, \Delta, q_{in}, R, \delta)$ be a td-tg^R transducer such that the right-hand side of every rule is a parjungle. Let $M' = (dec(Q), P, \Sigma, dec(\Delta), q_{in}, R', \delta)$ be an mt^R transducer.

M and M' are *related* if $R' = \{(\langle q, \sigma(x_1, \ldots, x_k)\rangle(y_1, \ldots, y_m), p_1, \ldots, p_k) \to tree(h) \mid (\langle q, \sigma(x_1, \ldots, x_k)\rangle, p_1, \ldots, p_k) \to h \text{ is in } R\}$. □

To show that related transducers compute the same tree-to-tree translation, we need the basic fact that the mapping *tree* distributes over substitution.

Lemma 5.3 Let g be a hypergraph over Γ, and let, for every $\gamma \in \Gamma$, $h(\gamma)$ be a hypergraph of the same rank as γ. If g and every $h(\gamma)$ are parjungles, then $g[\gamma/h(\gamma); \gamma \in \Gamma]$ is a parjungle, and

$$tree(g[\gamma/h(\gamma); \gamma \in \Gamma]) = tree(g)[\![\gamma/tree(h(\gamma)); \gamma \in \Gamma]\!].$$

Proof: We first show that $k = g[\gamma/h(\gamma); \gamma \in \Gamma]$ is a parjungle. In fact, this is obvious in the case that every $h(\gamma)$ is identification-free. To reduce the general case to this particular case, we introduce an extra symbol ε of rank 2. If $h(\gamma)$ is not identification-free, i.e., $ext_{h(\gamma)}(n+1) = ext_{h(\gamma)}(i)$ for some $i \in [n]$, where $n+1 = rank(\gamma)$, then we transform $h(\gamma)$ into an identification-free parjungle $h'(\gamma)$ by adding a new node v and a new edge e with $lab(e) = \varepsilon$ and $nod(e) = v_1 v_2$ with $v_1 = ext_{h(\gamma)}(i)$ and $v_2 = v$, and redefine $ext_{h'(\gamma)}(n+1) = v$. If $h(\gamma)$ is identification-free, then we define $h'(\gamma) = h(\gamma)$. As claimed above, $k' = g[\gamma/h'(\gamma); \gamma \in \Gamma]$ is a parjungle. Clearly, k is obtained from k' by contracting all ε-labeled edges, i.e., $k = k'[\varepsilon/c]$ where c is the hypergraph with $V_c = \{v\}$, $E_c = \emptyset$, and $ext_c = vv$. Since we may contract these edges one by one, it now suffices to observe that the contraction of one ε-labeled edge transforms a parjungle into a parjungle.

Next we show that $tree(g[\gamma/h(\gamma); \gamma \in \Gamma]) = tree(g)[\![\gamma/tree(h(\gamma)); \gamma \in \Gamma]\!]$. Although this is intuitively clear, we provide a detailed proof. Let τ be the unfolding function of

$k = g[\gamma/h(\gamma); \gamma \in \Gamma]$, and τ_g the one of g. For a node v of g or $h(\gamma)$, we denote by $[v]$ the corresponding node in k. Let $m+1 = rank(g) = rank(k)$. Since $ext_k(m+1) = [ext_g(m+1)]$, it suffices to show that, for every $v \in V_g$,

$$\tau([v]) = \tau_g(v)[\![\gamma/tree(h(\gamma)); \gamma \in \Gamma]\!].$$

We prove this by induction "on v", i.e., on the maximal length of a path leading to v (which is possible because g is acyclic).

Case (1): $v = ext_g(i)$ for some $i \in [m]$. Then $\tau_g(v) = y_i$, and $\tau([v]) = \tau([ext_g(i)]) = \tau(ext_k(i)) = y_i$.

Case (2): v has in-degree 1. Let $e = res_g^{-1}(v)$, and let $lab_g(e) = \gamma_0$ and $nod_g(e) = v_1 \dots v_u v$. Then

$\tau_g(v)[\![\gamma/tree(h(\gamma)); \gamma \in \Gamma]\!]$

$= (\gamma_0(\tau_g(v_1), \dots, \tau_g(v_u)))[\![\gamma/tree(h(\gamma)); \gamma \in \Gamma]\!]$ by definition of τ_g

$= tree(h(\gamma_0))[y_i/\tau_g(v_i)[\![\gamma/tree(h(\gamma)); \gamma \in \Gamma]\!]; i \in [u]]$ by Definition 4.5

$= tree(h(\gamma_0))[y_i/\tau([v_i]); i \in [u]]$ by the induction hypothesis for v_i.

Thus, it remains to show that

$$\tau([v]) = tree(h(\gamma_0))[y_i/\tau([v_i]); i \in [u]].$$

Let $h = h(\gamma_0)$, and let τ_h denote the unfolding function of h. Note that $rank(h) = u+1$. Since $[ext_h(u+1)] = [v]$, it suffices to show that, for every $w \in V_h$,

$$\tau([w]) = \tau_h(w)[y_i/\tau([v_i]); i \in [u]].$$

We prove this again by induction "on w" (because h is acyclic).

Case (1): $w = ext_h(i)$ for some $i \in [u]$. Note that $[w] = [v_i]$. Hence $\tau_h(w)[y_i/\tau([v_i]); i \in [u]] = y_i[y_i/\tau([v_i]); i \in [u]] = \tau([v_i]) = \tau([w])$.

Case (2): w has in-degree 1. Let $f = res_h^{-1}(w)$, and let $lab_h(f) = \beta$ and $nod_h(f) = w_1 \dots w_s w$. Then $lab_k(f) = \beta$ and $nod_k(f) = [w_1] \dots [w_s][w]$. Hence

$\tau([w]) = \beta(\tau([w_1]), \dots, \tau([w_s]))$ by definition of τ

$= \beta(\tau_h(w_1)[y_i/\tau([v_i]); i \in [u]], \dots, \tau_h(w_s)[y_i/\tau([v_i]); i \in [u]])$

by induction hypothesis for w_i

$= \beta(\tau_h(w_1), \dots, \tau_h(w_s))[y_i/\tau([v_i]); i \in [u]]$

$= \tau_h(w)[y_i/\tau([v_i]); i \in [u]]$ by definition of τ_h.

This proves the lemma. □

Lemma 5.4 Let M and M' be given either as in Definition 5.1 or as in Definition 5.2. If M and M' are related, then M is tree-generating and $\tau_t(M) = \tau(M')$.

Proof: The proof of the statement for M and M' as in Definition 5.1 can be transcribed easily to a proof of the statement for M and M' as in Definition 5.2. Thus, let M and M' be given as in Definition 5.1.

The statement of the lemma follows immediately from the next statement which implies, in particular, that $M(q_{in}, s)$ is a jungle.

> For every $q \in Q$ and $s \in T\langle\Sigma\rangle$, $M(q,s)$ is a parjungle and $tree(M(q,s)) = M'(q,s)$.

On its turn, this statement follows immediately (by induction on the structure of s) from the inductive characterization lemmas of td-tg transducers and mt transducers (Lemmas 3.13 and 4.6; if M and M' are given as in Definition 5.2, then apply Lemmas 3.19 and 4.8) and Lemma 5.3 (with $g = rhs(q,\sigma)$, $h(\langle q', x_j\rangle) = M(q', s_j)$, and $h(\gamma) = sing(\gamma)$ for $\gamma \in \Delta$). □

It now suffices to construct, for every tree t over an alphabet Γ, a parjungle which can be unfolded into t. We construct the parjungle that realizes the maximal sharing of different occurrences of equal subtrees of t.

Lemma 5.5 Let Γ be a ranked alphabet and let $m \geq 0$. For every $t \in T\langle\Gamma\rangle(Y_m)$, there is a parjungle $graph(t)$ of rank $m+1$ over $inc(\Gamma)$ such that $tree(graph(t)) = t$.

Proof: Construct the hypergraph $graph(t) = (V, E, lab, nod, ext)$ over $inc(\Gamma)$ as follows.

- $V = \{v_s | s \in sub(t)\} \cup \{v_{y_r} | y_r \in Y_m \text{ and } y_r \text{ does not occur in } t\}$
- $E = \{e_s | s \in sub(t) \text{ and } s \notin Y_m\}$

for every $e_s \in E$ with $s = \gamma(s_1, \ldots, s_k)$ for $\gamma \in \Gamma^{(k)}$ with $k \geq 0$, and $s_1, \ldots, s_k \in T\langle\Gamma\rangle$,

- $lab(e_s) = \gamma$
- $nod(e_s) = v_{s_1} \ldots v_{s_k} v_s$
- $ext = v_{y_1} \ldots v_{y_m} v_t$.

Clearly, $graph(t)$ is a parjungle with m parameters. Note that for $s \notin Y_m$, $res^{-1}(v_s) = \{e_s\}$. It is easy to show by induction on the structure of s that $\tau(v_s) = s$ (for every subtree s of t), where τ is the unfolding function of $graph(t)$. Hence $tree(graph(t)) = \tau(v_t) = t$. □

Example 5.6 Let $\Gamma = \{\delta^{(3)}, \sigma^{(2)}, \gamma^{(1)}, \alpha^{(0)}\}$ be a ranked alphabet. Figure 22(a) shows a tree $t \in T\langle\Gamma\rangle(Y_3)$ (in which y_3 does not occur), and Figure 22(b) shows the corresponding parjungle $graph(t)$ of rank 4. □

By transforming the right-hand side of every rule of an mt transducer into a parjungle as described in the previous lemma, the main result of this section follows directly from Lemma 5.4.

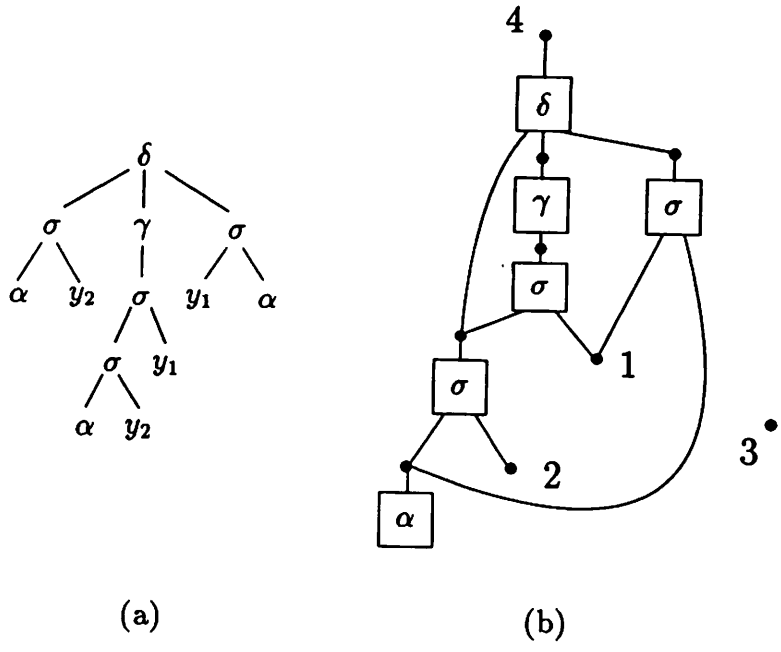


Figure 22: Tree t and corresponding parjungle $graph(t)$.

Lemma 5.7 $MT \subseteq tgtT$.

Proof: Let $M = (Q, \Sigma, \Delta, q_{in}, R)$ be an mt transducer and let $graph(M) = (inc(Q), \Sigma, inc(\Delta), q_{in}, R')$ be the td-tg transducer such that, if $\langle q, \sigma(x_1, \ldots, x_k)\rangle(y_1, \ldots, y_m) \rightarrow \zeta$ is in R, then $\langle q, \sigma(x_1, \ldots, x_k)\rangle \rightarrow graph(\zeta)$ is in R'. Since, by Lemma 5.5, $graph(M)$ and M are related, it follows from Lemma 5.4 that $graph(M)$ is tree-generating and $\tau_t(graph(M)) = \tau(M)$. □

6 Tree-Generating Top-Down Tree-To-Graph Transducers

Here we prove the more difficult inclusion $tgtT \subseteq MT$, i.e., the fact that for every tree-generating td-tg transducer M there is an mt transducer K such that $\tau_t(M) = \tau(K)$. As mentioned in the introduction, the proof splits into three steps. First, a td-tg^R transducer M' is constructed which computes the same tree-to-tree translation as M and in which the right-hand sides of all rules are parjungles. Second, since the right-hand sides are parjungles, the unfolding function can be applied to them, thereby relating M' with an mt^R transducer M'' (cf. Definition 5.2); thus, by Lemma 5.4, $\tau_t(M') = \tau(M'')$. Third, by Lemma 4.9, there is an mt transducer K such that $\tau(M'') = \tau(K)$. Thus, the only missing step is the first one.

Consider a tree-generating td-tg transducer M, i.e., $M(q_{in}, s)$ is a jungle for every input tree s. If every right-hand side of a rule of M is a parjungle, then $M(q, s')$ is a parjungle for every state q and every input tree s', cf. the proof of Lemma 5.4. If, however, nothing is known about the rules of M, then $M(q, s')$ need not be a parjungle. Nevertheless, in general, $M(q, s')$ has to have special properties that we now define (for loop-free and identification-free transducers only). The right-hand sides of the rules of M do not necessarily have these properties.

Definition 6.1 A hypergraph is a *semi-jungle* if it is acyclic, identification-free, every internal node has in-degree 1, and every external node has in-degree ≤ 1. □

Figure 23 shows a semi-jungle which is not a parjungle (because $ext(1)$ has in-degree 1).

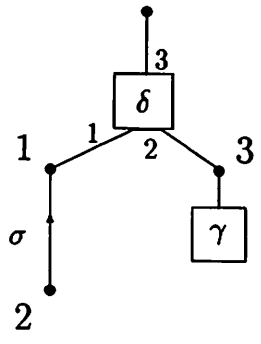


Figure 23: A semi-jungle.

Note that every parjungle is a semi-jungle, and hence every jungle is a semi-jungle.

Lemma 6.2 Let $M = (Q, P, \Sigma, \Delta, q_{in}, R, \delta)$ be a tree-generating, loop-free, and identification-free td-tg^R transducer. Then, for every $q \in Q$, $\sigma \in \Sigma^{(k)}$ with $k \geq 0$,

and $s_1, \ldots, s_k \in T\langle\Sigma\rangle$, if $M(q, \sigma(s_1, \ldots, s_k))$ is a semi-jungle and $\langle q', x_j\rangle$ occurs in $rhs(q, \sigma, \widetilde{\delta}(s_1), \ldots, \widetilde{\delta}(s_k))$, then $M(q', s_j)$ is a semi-jungle.

Proof: Immediate from Lemma 3.16 and the obvious fact that a full subgraph of a semi-jungle is a semi-jungle. □

Since $M(q_{in}, s)$ is a jungle, this lemma implies that if $sing(\langle q_{in}, s\rangle) \Rightarrow_M^* h$ and h contains an edge with label $\langle q, s'\rangle$, then $M(q, s')$ is a semi-jungle and a full subgraph of $M(q_{in}, s)$. The unfolding function of the jungle $M(q_{in}, s)$ will "visit" the semi-jungle $M(q, s')$ several times, in general. Such a "visit" starts at an external node of $M(q, s')$ of in-degree 1 and follows paths through $M(q, s')$ (in the opposite direction) that halt at edges of rank 1 or at other external nodes of $M(q, s')$. We will construct a "tree-equivalent" td-tg^R transducer M' of which all right-hand sides of rules are parjungles, in such a way that if $ext(i)$ is an external node of $M(q, s')$ of in-degree 1, then M' has a state (q, i, r) of rank $r + 1$ such that $tree(M'((q, i, r), s'))$ is the tree determined by the "visit" of the unfolding function of $M(q_{in}, s)$ to $M(q, s')$, starting at $ext(i)$, as described above; the rank $r + 1$ equals the number of visited external nodes. The information which external nodes of $M(q, s')$ have in-degree 1, and from which external nodes paths lead to other external nodes of in-degree 1, has to be computed by M', using regular look-ahead. This "visit" information is conveniently formalized in terms of particular hypergraphs which will be called i/o-graphs (these are similar to the i/o-graphs of attribute grammars, cf. [Knu68, KW76]).

Definition 6.3 Let $m \geq 0$. An *i/o-graph of rank m* is a hypergraph (V, E, lab, nod, ext) over $\{d_1, \ldots, d_m\}$ where d_r is a symbol of rank r, $V = [m]$, $E \subseteq \{e_i | i \in [m]\}$ with $res(e_i) = i$ for every $e_i \in E$, and $ext = 1...m$. □

The set of i/o-graphs is denoted by $IO\text{-}G$. Note that the symbols d_r and e_i are fixed, i.e., we use them for every i/o-graph.

Next we define the i/o-graph $io(h)$ of a hypergraph h where h is labeled over some arbitrary ranked alphabet Γ. Intuitively, $io(h)$ shows which external nodes of h have in-degree $\neq 0$, and it shows whether paths lead to these from other external nodes.

Definition 6.4 Let $m \geq 0$, and let h be an identification-free hypergraph of rank m over the ranked alphabet Γ.

The *i/o-graph of h*, denoted by $io(h)$, is the i/o-graph (V, E, lab, nod, ext) of rank m determined as follows. For $i \in [m]$, $e_i \in E$ if and only if $ext_h(i)$ has in-degree $\neq 0$ in h. For every $e_i \in E$, define $ar(e_i) = \{j \in [m] | i \neq j$ and there is path in h from $ext_h(j)$ to $ext_h(i)\}$. Let $ar(e_i) = \{j_1, \ldots, j_r\}$ with $j_1 < j_2 < \ldots < j_r$ with $r \geq 0$. Then $lab(e_i) = d_{r+1}$ and $nod(e_i) = j_1 j_2 \ldots j_r i$. □

Example 6.5 Consider the hypergraph h of rank 4 shown in Figure 24(a). The i/o-graph $io(h)$ of h is shown in Figure 24(b). □

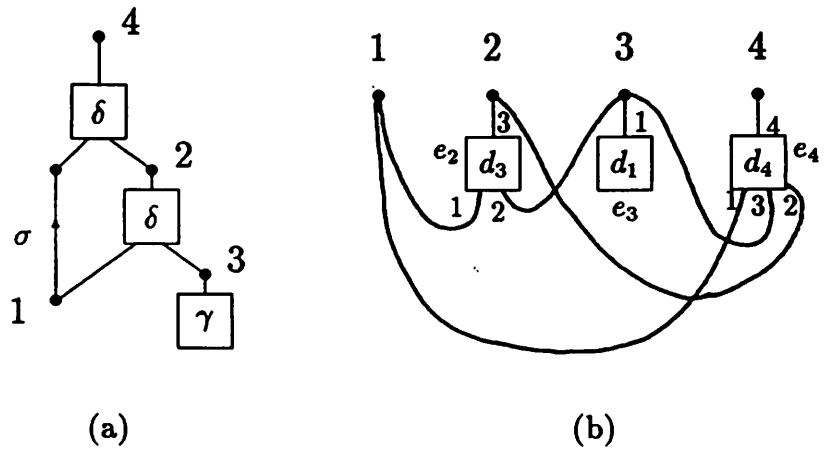


Figure 24: (a) Hypergraph h and (b) i/o-graph $io(h)$.

For every ranked alphabet Q, we denote by $[Q \to IO\text{-}G]$ the set of all rank preserving mappings from Q to $IO\text{-}G$. Now consider an arbitrary loop-free and identification-free td-tg^R transducer M. We will construct a finite state tree automaton B_M which computes, for every state q and input tree s, the i/o-graph $io(M(q,s))$.

Lemma 6.6 Let $M = (Q, P, \Sigma, \Delta, q_{in}, R, \delta)$ be a loop-free and identification-free td-tg^R transducer. There is a finite state tree automaton $B_M = (P', \Sigma, \delta')$ such that $P' = P \times [Q \to IO\text{-}G]$ and for every $q \in Q$ and $s \in T\langle\Sigma\rangle$, $\widetilde{\delta'}(s) = (\widetilde{\delta}(s), \phi)$ with $\phi(q) = io(M(q,s))$ for every $q \in Q$.

Proof: The family $\{\delta'_\sigma\}_{\sigma\in\Sigma}$ of transition functions of B_M is defined as follows. Let $\sigma \in \Sigma^{(k)}$ with $k \geq 0$, let $p_1, \ldots, p_k \in P$, and let $\phi_1, \ldots, \phi_k \in [Q \to IO\text{-}G]$. Define

$$\delta'_\sigma((p_1,\phi_1),\ldots,(p_k,\phi_k)) = (\delta_\sigma(p_1,\ldots,p_k),\phi)$$

with $\phi(q) = io(rhs(q,\sigma,p_1,\ldots,p_k)[\langle q', x_j\rangle/\phi_j(q'); \langle q', x_j\rangle \in \langle Q, X_k\rangle])$ for every $q \in Q$.

The proof that B_M satisfies the requirements is by induction on s, using the inductive characterization lemma of td-tg^R transducers (Lemma 3.19) and the following obvious property of the io-function:

> For every loop-free and identification-free hypergraph g over Γ, $\Gamma' \subseteq \Gamma$, and family $\{h(\gamma)\}_{\gamma\in\Gamma'}$ of identification-free hypergraphs,
>
> $$io(g[\gamma/h(\gamma); \gamma \in \Gamma']) = io(g[\gamma/io(h(\gamma)); \gamma \in \Gamma']).$$

Note that this property can be used, with $h(\langle q', x_j\rangle) = M(q', s_j)$, because it follows from Lemma 3.16 that $M(q', s_j)$ is identification-free. □

The information present in the i/o-graph allows us to give a precise definition of the unfolding of a semi-jungle h, corresponding to a "visit" to h of the unfolding function of a jungle of which h is a subgraph, as discussed above.

Definition 6.7 Let h be a semi-jungle over Γ. Let $ext_h(i)$ be an external node of h with in-degree 1, i.e., with $e_i \in E_{io(h)}$, and let $nod_{io(h)}(e_i) = j_1 \dots j_r i$. Then the *i-th tree represented by h*, denoted by $tree(i,h)$, is the tree $\tau_i(ext_h(i))$, where the partial function $\tau_i : V_h \to T\langle dec(\Gamma)\rangle(Y_r)$ is defined recursively by

(i) if v is an external node $ext_h(j_s)$ for some $s \in [r]$, then $\tau_i(v) = y_s$, and

(ii) if either $v = ext_h(i)$ or v is an internal node of h, then $\tau_i(v) = \gamma(\tau_i(v_1), \dots, \tau_i(v_p))$ where $\gamma = lab_h(res_h^{-1}(v))$ and $v_1 \dots v_p v = nod_h(res_h^{-1}(v))$.

τ_i is called the *i-th unfolding function of h*. □

Note that this definition really defines $tree(i,h)$, because h is acyclic, every internal node of h has in-degree 1, and there is no path from any $ext_h(j)$ to $ext_h(i)$ with $j \notin \{j_1, \dots, j_r\}$. Note also that the i-th unfolding function τ_i stops at the external nodes $ext_h(j_s)$, $s \in [r]$, even if they have in-degree 1 (this is a decision of technical nature).

Now we are ready to prove the first step in the inclusion $tgtT \subseteq MT$: the construction of M'. We can now be more precise about the way M' will be constructed: if $ext(i)$ is an external node of $M(q,s')$ of in-degree 1 and $nod_{io(M(q,s'))}(e_i) = j_1 \dots j_r i$, then $tree(M'((q,i,r),s')) = tree(i, M(q,s'))$.

Lemma 6.8 For every tree-generating td-tg transducer $\widetilde{M}$ there is a td-tg^R transducer M' such that the right-hand side of every rule of M' is a parjungle and $\tau_t(M') = \tau_t(\widetilde{M})$.

Proof: Let $\widetilde{M}$ be a tree-generating td-tg transducer. By Theorem 3.27 there is a loop-free and identification-free td-tg^R transducer $M = (Q, P, \Sigma, \Delta, q_{in}, R, \delta)$ which is equivalent to $\widetilde{M}$. Thus, in particular, M is also tree-generating and it satisfies Lemma 6.2. Construct the td-tg^R transducer $M' = (Q', P', \Sigma, \Delta, q'_{in}, R', \delta')$ as follows.

First, let $Q' = \{(q,i,r) | q \in Q \text{ and } i, r+1 \in [rank_Q(q)]\}$ with $rank_{Q'}((q,i,r)) = r+1$, and let $q'_{in} = (q_{in}, 1, 0)$. Second, the look-ahead automaton (P', Σ, δ') of M' is B_M, as defined in Lemma 6.6.

It remains to define R'. Let $(\langle q, \sigma(x_1, \dots, x_k)\rangle, p_1, \dots, p_k) \to h$ be in R. Let $\phi_1, \dots, \phi_k \in [Q \to IO\text{-}G]$. For every state labeled edge e of h, with $lab_h(e) = \langle q', x_j\rangle$, let $h(e)$ be the hypergraph obtained from the i/o-graph $\phi_j(q')$ by changing the label d_{u+1} of every edge e_m into $\langle (q', m, u), x_j\rangle$, where $u = rank(e_m) - 1$. Let $g = h[e/h(e); e \text{ is a state labeled edge of } h]$. Thus, intuitively, g is the right-hand side h in which the "visit" information given by $\phi_1, \dots, \phi_k$ has been integrated.

Assume now that g is a semi-jungle. Let $ext_g(i)$ have in-degree 1, and let $nod_{io(g)}(e_i) = j_1 \dots j_r i$. Then R' contains the rule

$$(\langle (q,i,r), \sigma(x_1, \dots, x_k)\rangle, (p_1, \phi_1), \dots, (p_k, \phi_k)) \to g'$$

where g' is the parjungle obtained from g by

- dropping all nodes (and incident edges) from which there is no path leading to $ext_g(i)$,

- dropping all nodes (and incident edges) from which there is a nonempty path to some $ext_g(j_s)$, $s \in [r]$, and
- defining $ext_{g'} = ext_g(j_1) \ldots ext_g(j_r) ext_g(i)$.

These are the rules of R' that are of importance. All remaining $((q,i,r),\sigma,(p_1,\phi_1), \ldots,(p_k,\phi_k))$-rules can be defined in an arbitrary fashion (with any parjungle of the correct rank as right-hand side). This ends the construction of M'.

To prove the correctness of M', we will show the following statement.

> For every $q \in Q$, $s \in T\langle\Sigma\rangle$, and $i \in [rank(q)]$,
> if $M(q,s)$ is a semi-jungle, $ext_{M(q,s)}(i)$ has in-degree 1, and $rank(e_i) = r+1$ in $io(M(q,s))$,
> then $tree(M'((q,i,r),s)) = tree(i, M(q,s))$.

Note that since all right-hand sides of M' are parjungles, $M'((q,i,r),s)$ is also a parjungle (cf. the proof of Lemma 5.4). Note also that taking $q = q_{in}$ in the above statement, we get $i = 1$, $r = 0$, and $tree(M'(q'_{in},s)) = tree(1, M(q_{in},s))$. Since obviously $tree(1, M(q_{in},s)) = tree(M(q_{in},s))$, this implies that $\tau_t(M') = \tau_t(M) = \tau_t(\widetilde{M})$.

Thus it remains to show the above statement. This is done by induction on the structure of s. Let $s = \sigma(s_1,\ldots,s_k)$ with $k \geq 0$. Then, by Lemma 3.19,

$tree(i, M(q,s)) =$

(1) $\quad tree(i, h[\langle q', x_j\rangle / M(q', s_j); \langle q', x_j\rangle \in \langle Q, X_k\rangle])$,

where h is the right-hand side of the rule $(\langle q, \sigma(x_1,\ldots,x_k)\rangle, p_1, \ldots, p_k) \to h$ of R, with $p_j = \widetilde{\delta}(s_j)$ for $j \in [k]$. Define $\phi_1,\ldots,\phi_k \in [Q \to IO\text{-}G]$ to contain the i/o-graphs of M for $s_1,\ldots,s_k$, i.e., $\phi_j(q') = io(M(q',s_j))$. Consider the hypergraph $g = h[e/h(e); e$ is a state labeled edge of $h]$, as described in the definition of R'. To prove that g is a semi-jungle, we use the following easy general fact (of which the last part was already shown in the proof of Lemma 6.6).

> Let h and f be hypergraphs, and e a loop-free edge of h. If f and $h[e/f]$ are semi-jungles, then $h[e/io(f)]$ is a semi-jungle and $io(h[e/io(f)]) = io(h[e/f])$.

This fact should be applied to all edges e of h with label $\langle q', x_j\rangle$, with $f = M(q', s_j)$. Then $h[e/f]$ is a semi-jungle because $M(q,s)$ is one by assumption, and f is a semi-jungle by Lemma 6.2. Since $io(f) = io(M(q',s_j)) = \phi_j(q')$, $h[e/io(f)]$ is the hypergraph g (apart from the labels of the edges).

This shows that g is a semi-jungle and that $io(g) = io(M(q,s))$. In particular, $ext_g(i)$ has in-degree 1. Consequently, R' contains the rule $(\langle(q,i,r),\sigma(x_1,\ldots,x_k)\rangle, (p_1,\phi_1),\ldots, (p_k,\phi_k)) \to g'$, where g' is obtained from g as described in the construction of R'. By Lemma 3.19,

$tree(M'((q,i,r),s)) =$

(2) $\quad tree(g'[\langle(q',m,u),x_j\rangle / M'((q',m,u),s_j); \langle(q',m,u),x_j\rangle \in \langle Q', X_k\rangle])$.

Since g' and all $M'((q',m,u),s_j)$ are parjungles, Lemma 5.3 shows that (2) =

$$tree(g')[\![\langle(q',m,u),x_j\rangle/tree(M'((q',m,u),s_j));\langle(q',m,u),x_j\rangle \in \langle Q',X_k\rangle]\!],$$

which by induction hypothesis is equal to

(3) $\quad tree(g')[\![\langle(q',m,u),x_j\rangle/tree(m,M(q',s_j));\langle(q',m,u),x_j\rangle \in \langle Q',X_k\rangle]\!].$

It now remains to show that (1) = (3). Since it should be obvious that $tree(g') = tree(i,g)$, we have to show that

(*) $\quad tree(i,h[\langle q',x_j\rangle/M(q',s_j);\langle q',x_j\rangle \in \langle Q,X_k\rangle]) =$

$$tree(i,g)[\![\langle(q',m,u),x_j\rangle/tree(m,M(q',s_j));\langle(q',m,u),x_j\rangle \in \langle Q',X_k\rangle]\!].$$

To abstract from this particular case, we will show the following claim, in general.

CLAIM. Let h be a loop-free hypergraph over Γ, and let, for every $\gamma \in \Gamma$, $h(\gamma)$ be a hypergraph of the same rank as γ. Let $\Gamma' = \{(\gamma,m,u)|\gamma \in \Gamma, m,u+1 \in [rank_\Gamma(\gamma)]\}$ with $rank_{\Gamma'}((\gamma,m,u)) = u+1$. Let $k = h[\gamma/h(\gamma);\gamma \in \Gamma]$ and $g = h[\gamma/io(h(\gamma))[e_m/sing((\gamma,m,u_m));e_m \in E_{io(h(\gamma))}];\gamma \in \Gamma]$, where $u_m = rank(e_m) - 1$. Finally, let $i_0 \in [rank(h)]$.

If k and all $h(\gamma)$ are semi-jungles, and $ext_k(i_0)$ has in-degree 1, then $tree(i_0,h[\gamma/h(\gamma);\gamma \in \Gamma]) = tree(i_0,g)[\![(\gamma,m,u)/tree(m,h(\gamma));(\gamma,m,u) \in \Gamma']\!]$.

Proof of the CLAIM: Note that, as observed before, $io(g) = io(k)$, by the proof of Lemma 6.6. Hence $ext_g(i_0)$ has in-degree 1 too, and $nod(e_{i_0}) = j_1 \ldots j_r i_0$ in both $io(g)$ and $io(k)$.

The proof of the claim is very similar to the proof of Lemma 5.3 (and the claim in fact generalizes that lemma).

Let τ be the i-th unfolding function of k, and τ_g the one of g. For a node v of h or $h(\gamma)$ we denote by $[v]$ the corresponding node in k. Note that $V_g = V_h$. Since $tree(i_0,k) = \tau([ext_h(i_0)])$ and $tree(i_0,g) = \tau_g(ext_h(i_0))$, it suffices to show that

$$\tau([v]) = \tau_g(v)[\![(\gamma,m,u)/tree(m,h(\gamma));(\gamma,m,u) \in \Gamma']\!]$$

for all nodes v of h such that there is a path in g from v to $ext_h(i_0)$, and there is no nonempty path in g from v to $ext_h(j_i)$, $i \in [r]$.

Case (1): $v = ext_h(j_i)$ for some $i \in [r]$. Then $\tau_g(v) = \tau_g(ext_g(j_i)) = y_i$, and $\tau([v]) = \tau(ext_k(j_i)) = y_i$.

Case (2): v has in-degree 1 in g. Then there is an edge e of h with $lab_h(e) = \gamma_0$, $nod_h(e) = v_1 \ldots v_n$, and $v = v_m$ for some $m \in [n]$, such that $ext_{h(\gamma_0)}(m)$ has in-degree 1 in $h(\gamma_0)$, and $nod_{io(h(\gamma_0))}(e_m) = m_1 \ldots m_u m$. Then

$\tau_g(v)[\![\ldots]\!]$

$= (\gamma_0,m,u)(\tau_g(v_{m_1}),\ldots,\tau_g(v_{m_u}))[\![\ldots]\!]$

$= tree(m,h(\gamma_0))[y_i/\tau_g(v_{m_i})[\![\ldots]\!];i \in [u]]$

$= tree(m,h(\gamma_0))[y_i/\tau([v_{m_i}]);i \in [u]].$

Thus, it remains to show that

$$\tau([v]) = tree(m, h(\gamma_0))[y_i/\tau([v_{m_i}]); i \in [u]].$$

Let $h' = h(\gamma_0)$ and let $\tau_{h'}$ be the m-th unfolding function of h'. Since $[ext_{h'}(m)] = [v]$, it suffices to show that

$$\tau([w]) = \tau_{h'}(w)[y_i/\tau([v_{m_i}]); i \in [u]]$$

for all nodes w of h' such that there is a path from w to $ext_{h'}(m)$, and there is no nonempty path from w to some $ext_{h'}(m_i)$, $i \in [u]$.

Case (1): $w = ext_{h'}(m_i)$ for some $i \in [u]$. Note that $[w] = [v_{m_i}]$. Hence $\tau_{h'}(w)[y_i/\tau([v_{m_i}]); i \in [u]] = y_i[y_i/\tau([v_{m_i}]); i \in [u]] = \tau([v_{m_i}]) = \tau([w])$.

Case (2): w has in-degree 1 in h'. This case is entirely the same as the corresponding case in the proof of Lemma 5.3, with h' instead of h, and v_{m_i} instead of v_i.

This ends the proof of the claim.

Just as in the proof of Lemma 5.4, this Claim can now be used with $h(\langle q', x_j \rangle) = M(q', s_j)$ and $h(\gamma) = sing(\gamma)$ for $\gamma \in \Gamma$.

This shows $(*)$ and ends the proof. □

Example 6.9 Consider the tree-generating, identification-free, loop-free td-tg transducer M which is defined as indicated in Figure 25. Figure 26 shows the rules of the td-tg transducer M' which is constructed as in the previous lemma. Note that the right-hand sides of rules of M' are parjungles. □

Recalling the remarks from the beginning of this section, this completes the proof of the second inclusion.

Lemma 6.10 $tgtT \subseteq MT$.

Proof: Let M be a tree-generating td-tg transducer. By Lemma 6.8 there is a td-tg^R transducer M' such that the right-hand side of every rule of M' is a parjungle and $\tau_t(M) = \tau_t(M')$. Construct an mt^R transducer M'' such that M' and M'' are related. Note that such an M'' exists, because the mapping *tree* is defined on parjungles. By Lemma 5.4, $\tau_t(M') = \tau(M'')$. Finally, by Lemma 4.9, there is an mt transducer M''' with $\tau(M'') = \tau(M''')$. □

And in total we obtain the main result of this paper: tree-generating td-tg transducers have the same power with respect to tree-to-tree translations as mt transducers.

Theorem 6.11 $tgtT = MT$.

Proof: Lemmas 5.7 and 6.10. □

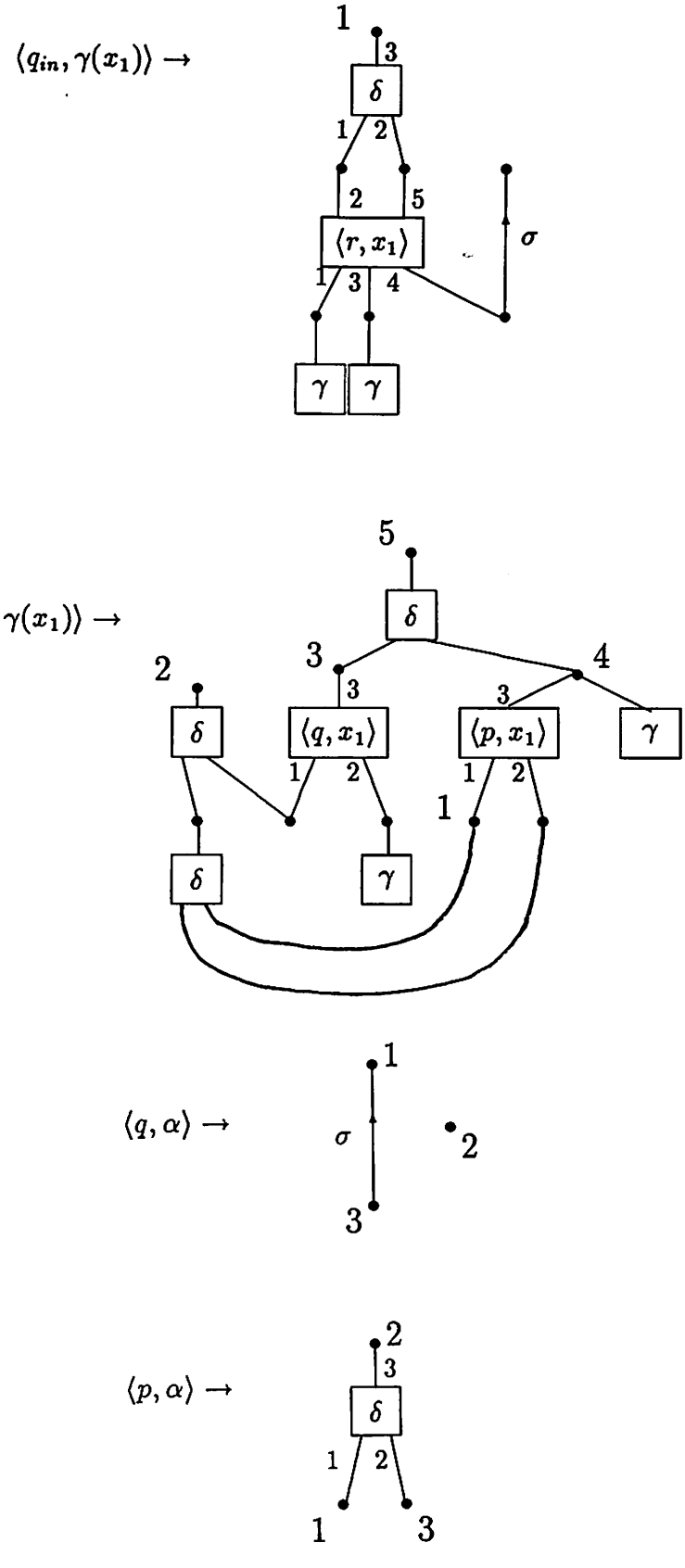
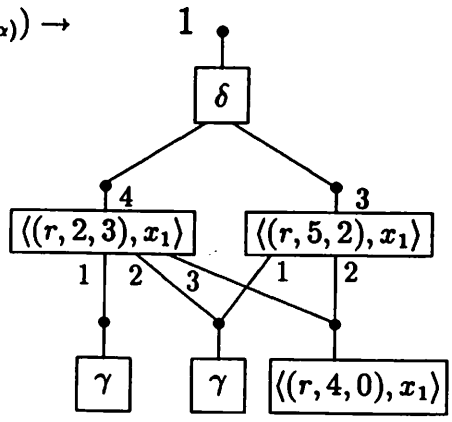
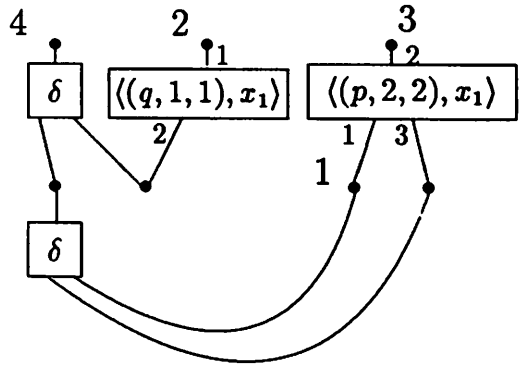


Figure 25: Rules of M.

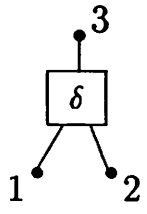
$(\langle(q_{in}, 1, 0), \gamma(x_1)\rangle, \phi_{\gamma(\alpha)}) \rightarrow$



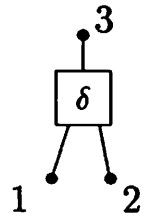
$(\langle(r, 2, 3), \gamma(x_1)\rangle, \phi_\alpha) \rightarrow$



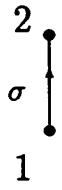
$(\langle(r, 5, 2), \gamma(x_1)\rangle, \phi_\alpha) \rightarrow$



$(\langle(p, 2, 2), \alpha\rangle) \rightarrow$



$(\langle(q, 1, 1), \alpha\rangle) \rightarrow$



$(\langle(r, 4, 0), \gamma(x_1)\rangle, \phi_\alpha) \rightarrow$

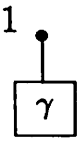


Figure 26: Rules of M'.

References

[Asv77] P.R.J. Asveld. Controlled iteration grammars and full hyper AFL's. *Inf. Control*, 34:248–269, 1977.

[AU71] A.V. Aho and J.D. Ullman. Translations on a context free grammar. *Inform. and Control*, 19:439–475, 1971.

[AU73] A.V. Aho and J.D. Ullman. *The Theory of Parsing, Translation and Compiling, Vol. I and Vol. II.* Prentice-Hall, 1973.

[Bak78] B. S. Baker. Generalized syntax directed translation, tree transducers, and linear space. *SIAM J. Comput.*, 7:376–391, 1978.

[BC87] M. Bauderon and B. Courcelle. Graph expressions and graph rewritings. *Math. Systems Theory*, 20:83–127, 1987.

[CF82] B. Courcelle and P. Franchi-Zannettacci. Attribute grammars and recursive program schemes. *Theoret. Comput. Sci.*, 17:163–191 and 235–257, 1982.

[Cou83] B. Courcelle. Fundamental properties of infinite trees. *Theoret. Comput. Sci.*, 25:95–169, 1983.

[Cou87a] B. Courcelle. An axiomatic definition of context-free rewriting and its application to NLC graph grammars. *Theoret. Comput. Sci.*, 55:141–181, 1987.

[Cou87b] B. Courcelle. On context-free sets of graphs and their monadic second-order theory. In H. Ehrig, M. Nagl, A. Rosenfeld, and G. Rozenberg, editors, *Graph-grammars and their application to computer science, LNCS 291*, pages 133–146. Springer-Verlag, 1987.

[Cou88] B. Courcelle. On the use of context-free graph grammars for analysing recursive definitions. In K. Fuchi and L. Kott, editors, *Programming of Future Generation Computers II*, pages 83–122. Elsevier/North-Holland, Amsterdam, 1988.

[Cou90] B. Courcelle. Graph rewriting: an algebraic and logic approach. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science Vol B*, pages 193–242. Elsevier Publishing Company, 1990.

[Der87] N. Dershowitz. Termination of rewriting. *J. Symb. Computation*, 3:69–116, 1987.

[DM79] N. Dershowitz and Z. Manna. Proving termination with multiset orderings. *Comm. Assoc. Comput. Mach.*, 22:465–476, 1979.

[EH91] J. Engelfriet and L. Heyker. The string-generating power of context-free hypergraph grammars. *J. Comput. System Sci.*, 43:328–360, 1991.

[EH92] J. Engelfriet and L. Heyker. Context-free hypergraph grammars have the same term-generating power as attribute grammars. *Acta Informatica*, 29:161–210, 1992.

[Eng75] J. Engelfriet. Bottom-up and top-down tree transformations - a comparison. *Math. Systems Theory*, 9:198–231, 1975.

[Eng76] J. Engelfriet. Surface tree languages and parallel derivation trees. *Theoret. Comput. Sci.*, 2:9–27, 1976.

[Eng77] J. Engelfriet. Top-down tree transducers with regular look-ahead. *Math. Systems Theory*, 10:289–303, 1977.

[Eng80] J. Engelfriet. Some open questions and recent results on tree transducers and tree languages. In R.V. Book, editor, *Formal language theory; perspectives and open problems*. New York, Academic Press, 1980.

[Eng81] J. Engelfriet. Tree transducers and syntax-directed semantics. Technical Report Memorandum 363, Technische Hogeschool Twente, 1981.

[Eng86a] J. Engelfriet. Context-free grammars with storage. Technical Report 86-11, University of Leiden, 1986.

[Eng86b] J. Engelfriet. The ET0L-hierarchy is inside the OI-hierarchy. In G. Rozenberg and A. Salomaa, editors, *The Book of L*, pages 101–109. Springer-Verlag, 1986.

[ER90] J. Engelfriet and G. Rozenberg. A comparison of boundary graph grammars and context-free hypergraph grammars. *Inform. and Computation*, 84:163–206, 1990.

[ERS80] J. Engelfriet, G. Rozenberg, and G. Slutzki. Tree transducers, L systems, and two-way machines. *J. Comput. System Sci.*, 20:150–202, 1980.

[EV85] J. Engelfriet and H. Vogler. Macro tree transducers. *J. Comput. System Sci.*, 31:71–146, 1985.

[EV88] J. Engelfriet and H. Vogler. High level tree transducers and iterated pushdown tree transducers. *Acta Informatica*, 26:131–192, 1988.

[Fül81a] Z. Fülöp. *Attributumos fatranszformatorok*. PhD thesis, Szeged, Hungary, 1981.

[Fül81b] Z. Fülöp. On attributed tree transducers. *Acta Cybernetica*, 5:261–279, 1981.

[FV89a] Z. Fülöp and S. Vagvölgyi. Top-down tree transducers with deterministic top-down look-ahead. *Inf. Proc. Letters*, 33:3–5, 1989.

[FV89b] Z. Fülöp and S. Vagvölgyi. Variants of top-down tree transducers with look-ahead. *Math. Systems Theory*, 21:125–145, 1989.

[GR75] S. Ginsburg and G. Rozenberg. T0L schemes and control sets. *Inf. Control*, 27:109–125, 1975.

[GS84] F. Gecseg and M. Steinby. *Tree Automata*. Akademiai Kiado, Budapest, 1984.

[Hab89] A. Habel. *Hyperedge replacement: grammars and languages*. PhD thesis, University of Bremen, 1989.

[HK87a] A. Habel and H.-J. Kreowski. May we introduce to you: hyperedge replacement. In H. Ehrig, M. Nagl, G. Rozenberg, and A. Rosenfeld, editors, *Graph grammars and their application to computer science, LNCS 291*, pages 15–26. Springer-Verlag, 1987.

[HK87b] A. Habel and H.-J. Kreowski. Some structural aspects of hypergraph languages generated by hyperedge replacement. In F.J. Brandenburg, G. Vidal-Naquet, and M. Wirsing, editors, *STACS 87 in LNCS 247*, pages 207–219. Springer-Verlag, 1987.

[HKP91] A. Habel, H.-J. Kreowski, and D. Plump. Jungle evaluation. *Fundamenta Informaticae*, 15:37–60, 1991.

[Hue80] G. Huet. Confluent reductions: abstract properties and applications to term rewriting systems. *J. Assoc. Comput. Mach.*, 27:797–821, 1980.

[Iro61] E.T. Irons. A syntax directed compiler for ALGOL 60. *Comm. Assoc. Comput. Mach.*, 4:51–55, 1961.

[Knu68] D.E. Knuth. Semantics of context-free languages. *Math. Systems Theory*, 2:127–145, 1968.

[Kre92] H.-J. Kreowski. Parallel hyperedge replacement. In G. Rozenberg and A. Salomaa, editors, *Lindenmayer Systems; Impacts on Theoretical Computer Science, Computer Graphics, and Developmental Biology*, pages 271–282. Springer-Verlag, 1992.

[KW76] K. Kennedy and S. K. Warren. Automatic generation of efficient evaluators for attribute grammars. In *Conf. Rec. of 3rd Symp. on Principles of Programming Languages*, pages 32–49, 1976.

[Lan83] K. Lange. Context-free controlled ET0L systems. In J. Diaz, editor, *Proc. of the 10th ICALP in LNCS 154*, pages 723–733. Springer-Verlag, 1983.

[Lau88a] C. Lautemann. Decomposition trees: structured graph representation and efficient algorithms. In M. Dauchet and M. Nivat, editors, *Proceedings of the CAAP 88 in LNCS 299*, pages 28–39. Springer-Verlag, 1988.

[Lau88b] C. Lautemann. Efficient algorithms on context-free graph languages. In T. Lepisto and A. Salomaa, editors, *Proceedings of the 15th ICALP in LNCS 317*, pages 362–378, 1988.

[MR87] U. Montanari and F. Rossi. An efficient algorithm for the solution of hierarchical networks of constraints. In H. Ehrig, M. Nagl, G. Rozenberg, and A. Rosenfeld, editors, *Graph grammars and their application to computer science, LNCS 291*, pages 440–457. Springer-Verlag, 1987.

[Rou70] W.C. Rounds. Mappings and grammars on trees. *Math. Systems Theory*, 4:257–287, 1970.

[Tha70] J.W. Thatcher. Generalized^2 sequential machine maps. *J. Comput. Syst. Sci.*, 4:339–367, 1970.

[TW68] J.W. Thatcher and J.B. Wright. Generalized finite automata theory with application to a decision problem of second-order logic. *Math. Systems Theory*, 2:57–81, 1968.

Liste der bisher erschienenen Ulmer Informatik-Berichte:
List of technical reports currently available from the University of Ulm:

91-01 KER-I KO, P. ORPONEN, U. SCHÖNING, O. WATANABE:
Instance Complexity.

91-02 K. GLADITZ, H. FASSBENDER, H. VOGLER:
Compiler-Based Implementation of Syntax-Directed Functional Programming.

91-03 ALFONS GESER:
Relative Termination.

91-04 JOHANNES KÖBLER, UWE SCHÖNING, JACOBO TORAN:
Graph Isomorphism is low for PP.

91-05 JOHANNES KÖBLER, THOMAS THIERAUF:
Complexity Restricted Advice Functions.

91-06 UWE SCHÖNING:
Recent Highlights in Structural Complexity Theory.

91-07 FREDERIC GREEN, JOHANNES KÖBLER, JACOBO TORAN:
The Power of the Middle Bit.

91-08 V. ARVIND, Y. HAN, L. HEMACHANDRA, J. KÖBLER, A. LOZANO,
M. MUNDHENK, M. OGIWARA, U. SCHÖNING, R. SILVESTRI, T. THIERAUF:
Reductions to Sets of Low Information Content.

92-01 VIKRAMAN ARVIND, JOHANNES KÖBLER, MARTIN MUNDHENK:
Bounded Truth-Table and Conjunctive Reductions to Sparse and Tally Sets.

92-02 THOMAS NOLL, HEIKO VOGLER:
Top-down Parsing with Simultaneous Evaluation of Noncircular Attribute Grammars

92-03 PROGRAM AND ABSTRACTS:
17. Workshop über Komplexitätstheorie, effiziente Algorithmen und Datenstrukturen
am 26. Mai 1992 in Ulm

92-04 V. ARVIND, J. KÖBLER, M. MUNDHENK
Lowness and the Complexity of Sparse and Tally Descriptions

92-05 JOHANNES KÖBLER
Locating P/poly Optimally in the Extended Low Hierarchy

92-06 ARMIN KÜHNEMANN, HEIKO VOGLER
Synthesized and inherited functions - a new computational model
for syntax-directed semantics

92-07 HEINZ FASSBENDER, HEIKO VOGLER
A Universal Unification Algorithm Based on Unification-Driven
Leftmost Outermost Narrowing.

92-08 UWE SCHÖNING
On Random Reductions from Sparse Sets to Tally Sets

92-09 HERMANN VON HASSELN, LAURA MARTIGNON
Consistency in Stochastic Networks

92-10 MICHAEL SCHMITT
A Slightly Improved Upper Bound on the Size of Weights Sufficient to Represent Any Linearly Separable Boolean Function

92-11 JOHANNES KÖBLER, SEINOSUKE TODA
On the Power of Generalized MOD-Classes

92-12 V. ARVIND, J. KÖBLER, M. MUNDHENK
Reliable Reductions, High Sets and Low Sets

92-13 ALFONS GESER
On a monotonic semantic path ordering

92-14 JOOST ENGELFRIET, HEIKO VOGLER
The Translation Power of Top-Down Tree-To-Graph Transducers

Ulmer Informatik-Berichte

ISSN 0939-5091

Herausgeber: Fakultät für Informatik

Universität Ulm, Oberer Eselsberg, W-7900 Ulm