Transformation of Turing Machines into Context-Dependent Fusion Grammars

Aaron Lye
University of Bremen, Department of Computer Science and Mathematics
P.O.Box 33 04 40, 28334 Bremen, Germany
lye@math.uni-bremen.de

Context-dependent fusion grammars were recently introduced as devices for the generation of hypergraph languages. In this paper, we show that this new type of hypergraph grammars, where the application of fusion rules is restricted by positive and negative context conditions, is a universal computation model. Our main result is that Turing machines can be transformed into these grammars such that the recognized language of the Turing machine and the generated language of the corresponding context-dependent fusion grammar coincide up to representation of strings as graphs. As a corollary we get that context-dependent fusion grammars can generate all recursively enumerable string languages.

1 Introduction

In 2017 we introduced fusion grammars as generative devices on hypergraphs [7]. They are motivated by the observation, that one encounters various fusion processes in various scientific fields like DNA computing, chemistry, tiling, fractal geometry, visual modeling and others. The common principle is, that a few small entities may be copied and fused to produce more complicated entities. However, it seems that the generative power of fusion grammars (without context-conditions or regulations) is limited (cf. [7, 10]). Furthermore, there are numerous examples of fusion processes restricted to certain conditions, e.g. the presence of enzymes accelerating chemical reactions. In [9] we introduced context-dependent fusion grammars as a generalization of fusion grammars to simulate Petri nets. It turns out, that context-dependent fusion grammars are powerful enough to simulate Turing machines. We construct a transformation of Turing machines into context-dependent fusion grammars in such a way that the recognized language of the Turing machine and the language generated by the corresponding grammar coincide up to representation of strings as graphs.[1] As a corollary we get that context-dependent fusion grammars can generate all recursively enumerable string languages (up to representation) and that they are universal in this respect.

Relating computational models to Turing machines is an old and established approach which can be found in most foundations textbooks in theoretical computer science. Moreover, it is well known that graph transformation in general is Turing-complete. In 1978 Uesu presented a system of graph grammars that generates all recursively enumerable sets of labeled graphs (cf. [12]). In [1] Ehrig et. al. presented a transformation of Chomsky grammars in graph grammars (cf. also [5, 6] for similar results). Furthermore, asking "what programming constructs are needed on top of graph transformation rules to obtain a computationally complete language" [2] is not a new question. In [2] Habel and Plump

[1] Instead of Turing machines some other equivalent computational formalism could be chosen, e.g. Petri nets with inhibitor arcs which would be an extension of the transformation presented in [9]. In the considered approaches the transformations were technical and of similar complexity. Turing machines have the advantage of being an established and well known computational model.

Rachid Echahed and Detlef Plump (Eds.): Tenth International Workshop on Graph Computation Models (GCM 2019)
EPTCS 309, 2019, pp. 53–70, doi:10.4204/EPTCS.309.3

presented a graph program that simulates a Turing machine. Due to the novelty of (variants of) fusion grammars, there are many open questions. Enhancing fusion grammars by the inversion of fusions led to the introduction of splitting/fusion grammars in [8]. It is shown that splitting/fusion grammars can simulate Chomsky grammars and connective hypergraph grammars.

Our construction differs significantly from those cited above due to the semantics of context-dependent fusion grammars. A context-dependent fusion grammar provides a start hypergraph and a finite set of fusion labels (besides some markers and terminals). The fusion labels have complements and serve as rules. A context-dependent fusion is defined by choosing two complementarily labeled hyperedges provided that certain positive and negative context conditions are satisfied, removing them and merging the corresponding attachment vertices. Given a hypergraph, the set of all possible fusions is finite as fusions never create anything. To overcome this limitation, we allow arbitrary multiplications of connected components, i.e., connected subhypergraphs of maximal size, within derivations in addition to fusion. All modifications must be expressed in this way.

The paper is organized as follows. In Section 2, basic notions and notations of hypergraphs are recalled. Section 3 and 4 recall the notions of Turing machines and context-dependent fusion grammars, respectively. Section 5 presents the reduction of Turing machines to context-dependent fusion grammars. Section 6 concludes the paper pointing out some open problems.

2 Preliminaries

We consider hypergraphs the hyperedges of which have multiple sources and multiple targets. A *hypergraph* over a given label alphabet Σ is a system $H = (V,E,s,t,lab)$ where V is a finite set of *vertices*, E is a finite set of *hyperedges*, $s,t\colon E \to V^*$ are two functions assigning to each hyperedge a sequence of *sources* and *targets*, respectively, and $lab\colon E \to \Sigma$ is a function, called *labeling*. The components of $H = (V,E,s,t,lab)$ may also be denoted by V_H, E_H, s_H, t_H, and lab_H respectively. The class of all hypergraphs over Σ is denoted by $\mathscr{H}_\Sigma$.

Let $pr\colon V^* \times \mathbb{N} \to V$ be defined as $pr(v_1v_2\dots v_n,i) = v_i$ if $1 \leq i \leq n$, where n is the length of the sequence. It is undefined otherwise.

Let $H \in \mathscr{H}_\Sigma$, and let $\equiv$ be an equivalence relation on V_H. Then the *fusion of the vertices in H with respect to* $\equiv$ yields the hypergraph $H/\!\equiv = (V_H/\!\equiv, E_H, s_{H/\equiv}, t_{H/\equiv}, lab_H)$ with the set of equivalence classes $V_H/\!\equiv = \{[v] \mid v \in V_H\}$ and $s_{H/\equiv}(e) = [v_1]\cdots[v_{k_1}]$, $t_{H/\equiv}(e) = [w_1]\cdots[w_{k_2}]$ for each $e \in E_H$ with $s_H(e) = v_1\cdots v_{k_1}$, $t_H(e) = w_1\cdots w_{k_2}$. We often use the notation of specifying only the equivalent vertices.

Given $H,H' \in \mathscr{H}_\Sigma$, a *hypergraph morphism* $g\colon H \to H'$ consists of two mappings $g_V\colon V_H \to V_{H'}$ and $g_E\colon E_H \to E_{H'}$ such that $s_{H'}(g_E(e)) = g_V^*(s_H(e))$, $t_{H'}(g_E(e)) = g_V^*(t_H(e))$ and $lab_{H'}(g_E(e)) = lab_H(e)$ for all $e \in E_H$, where $g_V^*\colon V_H^* \to V_{H'}^*$ is the canonical extension of g_V, given by $g_V^*(v_1\cdots v_n) = g_V(v_1)\cdots g_V(v_n)$ for all $v_1\cdots v_n \in V_H^*$.

Given $H,H' \in \mathscr{H}_\Sigma$, H is a *subhypergraph* of H', denoted by $H \subseteq H'$, if $V_H \subseteq V_{H'}$, $E_H \subseteq E_{H'}$, $s_H(e) = s_{H'}(e)$, $t_H(e) = t_{H'}(e)$, and $lab_H(e) = lab_{H'}(e)$ for all $e \in E_H$. $H \subseteq H'$ implies that the two inclusions $V_H \subseteq V_{H'}$ and $E_H \subseteq E_{H'}$ form a hypergraph morphism from $H \to H'$.

Let $H' \in \mathscr{H}_\Sigma$ as well as $V \subseteq V_{H'}$ and $E \subseteq E_{H'}$. Then the *removal* of (V,E) from H' given by $H = H' - (V,E) = (V_{H'} - V, E_{H'} - E, s_H, t_H, lab_H)$ with $s_H(e) = s_{H'}(e)$, $t_H(e) = t_{H'}(e)$ and $lab_H(e) = lab_{H'}(e)$ for all $e \in E_{H'} - E$ defines a subgraph $H \subseteq H'$ if $s_{H'}(e), t_{H'}(e) \in (V_{H'} - V)^*$ for all $e \in E_{H'} - E$. Let $H \in \mathscr{H}_\Sigma$, $H' \subseteq H$. Then $H - H' = H - (V_{H'}, E_{H'})$.

Let $H \in \mathscr{H}_\Sigma$ and $H' = (V',E',s',t'\colon E' \to (V_H + V')^*, lab'\colon E' \to \Sigma)$ be some quintuple with two sets V', E' and three mappings s',t' and lab' where $+$ denotes the disjoint union of sets. Then the *extension* of

H by H' given by $H'' = (V_H + V', E_H + E', s, t, lab)$ with $s(e) = s_H(e)$, $t(e) = t_H(e)$ and $lab(e) = lab_H(e)$ for all $e \in E_H$ as well as $s(e) = s'(e)$, $t(e) = t'(e)$ and $lab(e) = lab'(e)$ for all $e \in E'$ is a hypergraph with $H \subseteq H''$.

Let $H \in \mathscr{H}_\Sigma$ and let $att(e)$ be the set of source and target vertices for $e \in E_H$. H is *connected* if for each $v, v' \in V_H$, there exists a sequence of triples $(v_1, e_1, w_1) \dots (v_n, e_n, w_n) \in (V_H \times E_H \times V_H)^*$ such that $v = v_1, v' = w_n$ and $v_i, w_i \in att(e_i)$ for $i = 1, \dots, n$ and $w_i = v_{i+1}$ for $i = 1, \dots, n-1$. A subgraph C of H, denoted by $C \subseteq H$, is a *connected component* of H if it is connected and there is no larger connected subgraph, i.e., $C \subseteq C' \subseteq H$ and C' connected implies $C = C'$. The set of connected components of H is denoted by $\mathscr{C}(H)$.

Given $H, H' \in \mathscr{H}_\Sigma$, the *disjoint union* of H and H' is denoted by $H + H'$. Further, $k \cdot H$ denotes the disjoint union of H with itself k times. We use the *multiplication* of H defined by means of $\mathscr{C}(H)$ as follows. Let $m \colon \mathscr{C}(H) \to \mathbb{N}$ be a mapping, called *multiplicity*, then $m \cdot H = \sum_{C \in \mathscr{C}(H)} m(C) \cdot C$.

A string can be represented by a simple path where the sequence of labels along the path equals the given string. Let $w = x_1 \dots x_n \in \Sigma^*$ for $n \geq 1$ and $x_i \in \Sigma$ for $i = 1, \dots, n$. Let $[n] = \{1, \dots, n\}$. Then the *string graph* of w is defined by $sg(w) = (\{0\} \cup [n], [n], s_w, t_w, lab_w)$ with $s_w(i) = (i-1), t_w(i) = i$ and $lab(i) = x_i$ for $i = 1, \dots, n$. The string graph of the empty string ε, denoted by $sg(\varepsilon)$, is the discrete graph with a single vertex 0. Obviously, there is a one-to-one correspondence between Σ^* and $sg(\Sigma^*) = \{sg(w) \mid w \in \Sigma^*\}$. We define a mapping *begin* assigning to every string graph to its vertex 0.

3 Turing Machines

In this section, we shortly recall the notion of Turing machines (see, e.g., [11, 4, 3]) and their recognized languages. We consider Turing machines with a designated start and accept state and one two-sided infinitely extendable (working) tape. We use two delimiters $\triangleright$ and $\triangleleft$ to indicate the end of the tape to the left and to the right, respectively. If the head moves beyond a delimiter a new cell labeled $\square$ (the blank symbol) is added.

Definition 1. *1. A* Turing machine *is a system $TM = (Q, \Omega, \Gamma, \Delta)$, where Q is a finite set of states with two designated different states q_{start} and q_{accept}, Ω is the input alphabet, Γ is the tape alphabet with $\Omega \subseteq \Gamma$ and $\square \in \Gamma \setminus \Omega$, and $\Delta \subseteq (Q \setminus \{q_{accept}\}) \times \Gamma \times \Gamma \times \{l, n, r\} \times Q$ is the transition relation.*

2. $conf(TM) = Q \times \Gamma^* \times \Gamma^*$ is the set of configurations.

3. A step *of TM is defined by the relation $\vdash_{TM} \subseteq conf(TM) \times conf(TM)$:*

$$
\begin{aligned}
(p, \alpha u, x\beta) &\vdash_{TM} (q, \alpha, uy\beta) && \text{if } (p, x, y, l, q) \in \Delta \\
(p, \varepsilon, x\beta) &\vdash_{TM} (q, \varepsilon, \square y\beta) && \text{if } (p, x, y, l, q) \in \Delta \\
(p, \alpha, \varepsilon) &\vdash_{TM} (q, \alpha, y) && \text{if } (p, \square, y, l, q) \in \Delta \\
(p, \alpha, x\beta) &\vdash_{TM} (q, \alpha, y\beta) && \text{if } (p, x, y, n, q) \in \Delta \\
(p, \alpha, x\beta) &\vdash_{TM} (q, \alpha y, \beta) && \text{if } (p, x, y, r, q) \in \Delta \\
(p, \alpha, \varepsilon) &\vdash_{TM} (q, \alpha y, \varepsilon) && \text{if } (p, \square, y, r, q) \in \Delta
\end{aligned}
$$

where $\alpha, \beta \in \Gamma^*, u \in \Gamma$ and ε is the empty string.

4. A computation *of TM is a potentially infinite sequence of configurations $c_0, c_1, \dots$ where $c_0 = (q_{start} \times \varepsilon \times w)$ is the start configuration wrt the input $w \in \Omega^*$, and $c_i \vdash_{TM} c_{i+1}$ for all $i \in \mathbb{N}$.*

5. The recognized language of TM is defined as $L(TM) = \{w \in \Omega^* \mid (q_{start}, \varepsilon, w) \vdash_{TM}^* (q_{accept}, \alpha, \beta)\}$, where $\alpha, \beta \in \Gamma^*$ are arbitrary.

Remark 1. *1.* $(p,x,y,dir,q) \in \Delta$ *means if the Turing machine is in state* p *and reads the symbol* x*, it can replace* x *by* y *and move the (read/write) head to the left if* $dir = l$*, to the right if* $dir = r$ *or leave the head stationary if* $dir = n$*. Afterwards the machine is in state* q*.*

2. *A configuration is of the form* (q,α,β) *which means the machine is in state* q *and the contents of the tape to the left and right of the head are* α *and* β*, respectively. The machine reads the first symbol of* β *if* $\beta \neq \varepsilon$ *and* $\square$ *otherwise.*
3. *A computation is finite if a halting configuration is reached, i.e., if there is no possibility of continuing the computation. If the machine enters the state* q_{accept}*, then it accepts the input.*
4. *The recognized language consists of all strings for which a computation exists such that the machine enters the accepting state* q_{accept}*.*

4 Context-Dependent Fusion Grammars

In this section, we recall the notion of context-dependent fusion grammars (cf. [9]). Context-dependent fusion grammars generate hypergraph languages from start hypergraphs via successive applications of context-dependent fusion rules, multiplications of connected components, and a filtering mechanism. A fusion rule is defined by two complementary-labeled hyperedges and positive and negative context-conditions. Such a rule is applicable if both the positive and negative context-conditions of the rule are satisfied. Its application consumes the two hyperedges and fuses the sources of the one hyperedge with the sources of the other as well as the targets of the one with the targets of the other.

Definition 2. *1.* $F \subseteq \Sigma$ *is a* fusion alphabet *if it is accompanied by a* complementary fusion alphabet $\overline{F} = \{\overline{A} \mid A \in F\} \subseteq \Sigma$*, where* $F \cap \overline{F} = \emptyset$ *and* $\overline{A} \neq \overline{B}$ *for* $A,B \in F$ *with* $A \neq B$ *and a* type function $type\colon F \cup \overline{F} \to (\mathbb{N} \times \mathbb{N})$ *with* $type(A) = type(\overline{A})$ *for each* $A \in F$*.*

2. *For each* $A \in F$ *with* $type(A) = (k_1,k_2)$*, the* fusion rule $fr(A)$ *is the hypergraph, depicted in Figure 1, with* $V_{fr(A)} = \{v_i, v'_i \mid i = 1,\ldots,k_1\} \cup \{w_j, w'_j \mid j = 1,\ldots,k_2\}$*,* $E_{fr(A)} = \{e,\overline{e}\}$*,* $s_{fr(A)}(e) = v_1 \cdots v_{k_1}$*,* $s_{fr(A)}(\overline{e}) = v'_1 \cdots v'_{k_1}$*,* $t_{fr(A)}(e) = w_1 \cdots w_{k_2}$*,* $t_{fr(A)}(\overline{e}) = w'_1 \cdots w'_{k_2}$*, and* $lab_{fr(A)}(e) = A$ *and* $lab_{fr(A)}(\overline{e}) = \overline{A}$*.*
3. *The application of* $fr(A)$ *to a hypergraph* $H \in \mathscr{H}_\Sigma$ *proceeds according to the following steps: (1) Choose a* matching morphism $g\colon fr(A) \to H$*. (2) Remove the images of the two hyperedges of* $fr(A)$ *yielding* $X = H - (\emptyset, \{g(e), g(\overline{e})\})$*. (3) Fuse the corresponding source and target vertices of the removed hyperedges yielding the hypergraph* $H' = X/\equiv$ *where* $\equiv$ *is generated by the relation* $\{(g(v_i), g(v'_i)) \mid i = 1,\ldots,k_1\} \cup \{(g(w_j), g(w'_j)) \mid j = 1,\ldots,k_2\}$*. The application of* $fr(A)$ *to* H *is denoted by* $H \underset{fr(A)}{\Longrightarrow} H'$ *and called a* direct derivation*.*
4. *A* context-dependent fusion rule *is a triple* $cdfr = (fr(A), PC, NC)$ *for some* $A \in F$ *where* PC *and* NC *are two finite sets of hypergraph morphisms with domain* $fr(A)$ *mapping into finite contexts defining* positive *and* negative context conditions *respectively.*
5. *The rule* $cdfr$ *is applicable to some hypergraph* H *via a matching morphism* $g\colon fr(A) \to H$ *if for each* $(c\colon fr(A) \to C) \in PC$ *there exists a hypergraph morphism* $h\colon C \to H$ *such that* h *is injective on the set of hyperedges and* $h \circ c = g$*, and for all* $(c\colon fr(A) \to C) \in NC$ *there does not exist a hypergraph morphism* $h\colon C \to H$ *such that* $h \circ c = g$*.*
6. *If* $cdfr$ *is applicable to* H *via* g*, then the direct derivation* $H \underset{cdfr}{\Longrightarrow} H'$ *is the direct derivation* $H \underset{fr(A)}{\Longrightarrow} H'$*.*

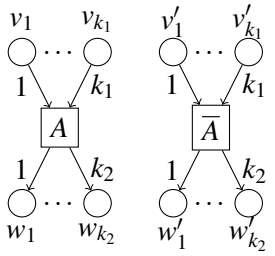


Figure 1: The fusion rule $fr(A)$ with $type(A) = (k_1, k_2)$

Remark 2. *$fr(A)$ and $(fr(A), \emptyset, \emptyset)$ are equivalent. We use the first as an abbreviation for the latter.*

Given a finite hypergraph, the set of all possible successive fusions is finite as fusion rules never create anything. To overcome this limitation, arbitrary multiplications of disjoint components within derivations are allowed. The generated language consists of the terminal part of all resulting connected components that contain no fusion symbols and at least one marker symbol, where marker symbols are removed in the end. These marker symbols allow us to distinguish between wanted and unwanted terminal components.

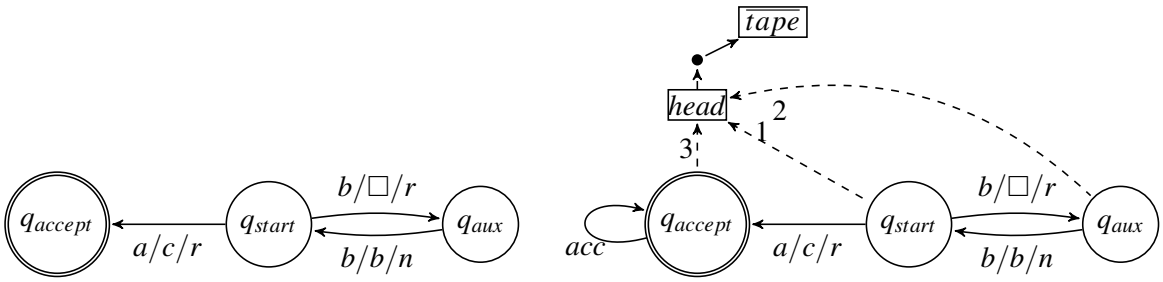
Definition 3. 1. *A* context-dependent fusion grammar *is a system $CDFG = (Z, F, M, T, P)$ where $Z \in \mathscr{H}_{F \cup \overline{F} \cup T \cup M}$ is a* start hypergraph *consisting of a finite number of connected components, $F \subseteq \Sigma$ is a finite fusion alphabet, $M \subseteq \Sigma$ with $M \cap (F \cup \overline{F}) = \emptyset$ is a finite set of* markers*, $T \subseteq \Sigma$ with $T \cap (F \cup \overline{F}) = \emptyset = T \cap M$ is a finite set of* terminal labels*, and P is a finite set of context-dependent fusion rules.*

2. *A* direct derivation $H \Longrightarrow H'$ *is either a context-dependent fusion rule application $H \underset{cdfr}{\Longrightarrow} H'$ for some $cdfr \in P$ or a multiplication $H \underset{m}{\Longrightarrow} m \cdot H$ for some multiplicity $m \colon \mathscr{C}(H) \to \mathbb{N}$. A* derivation *$H \overset{n}{\Longrightarrow} H'$ of length $n \geq 0$ is a sequence of direct derivations $H_0 \Longrightarrow H_1 \Longrightarrow \ldots \Longrightarrow H_n$ with $H = H_0$ and $H' = H_n$. If the length does not matter, we may write $H \overset{*}{\Longrightarrow} H'$.*

3. *$L(CDFG) = \{rem_M(Y) \mid Z \overset{*}{\Longrightarrow} H, Y \in \mathscr{C}(H) \cap (\mathscr{H}_{T \cup M} \setminus \mathscr{H}_T)\}$ is the* generated language *where $rem_M(Y)$ is the terminal hypergraph obtained by removing all hyperedges with labels in M from Y.*

5 Transformation of Turing Machines into Context-Dependent Fusion Grammars

In this section, we show that Turing machines can be simulated by context-dependent fusion grammars. The construction works roughly as follows: (1) A Turing machine is represented by the usual state graph, (2) the tape is represented by a sequence of successive edges each labeled with symbols of the working alphabet, (3) the state graph and the tape are connected by a hyperedge, called *head*, which also indicates the current state (specifically, the current state is the first source) and is attached to the current position on the tape, (4) in addition, the start hypergraph contains components that allow to generate the initial tape in a terminal and a fusion version, (5) components that allow to simulate a transition step of the Turing machine by a sequence of applications of context-dependent fusion rules, and (6) there is a terminating component that enables to disconnect the terminal tape with the input string from the rest of the working hypergraph whenever an accepting state is reached. In other words, the grammar generates a tape with a detachable input string if and only if the Turing machine accepts this string.



(a) the usual state graph of a Turing machine (b) the hypergraph representation of the Turing machine

Figure 2: Example state graph and the corresponding hypergraph representation of a Turing machine

Because the transformation is quite complicated we introduce the ideas step by step. First, we give a hypergraph representation of Turing machines. Then we introduce the tape graph representing the working tape as well as the input to the Turing machine. This leads us directly to hypergraph representations of configurations. Afterwards, we demonstrate how a step can be simulated by a sequence of context-dependent fusion rules. Finally, the two constructions are combined and our main theorem is presented.

5.1 Representation of a Turing machine by a hypergraph

In the hypergraphical representation of a Turing machine, denoted by $hg(TM)$, vertices represent states and edges between these vertices represent the elements of the transition relation. Initially, there is one additional vertex and three special hyperedges: an *acc*-loop indicates the accepting state, a hyperedge with $|Q|$ sources and one target, called *head*, connects the state graph with the additional vertex to which a $\overline{tape}$-hyperedge is attached. The latter enables fusion of the Turing machine with the tape graph.

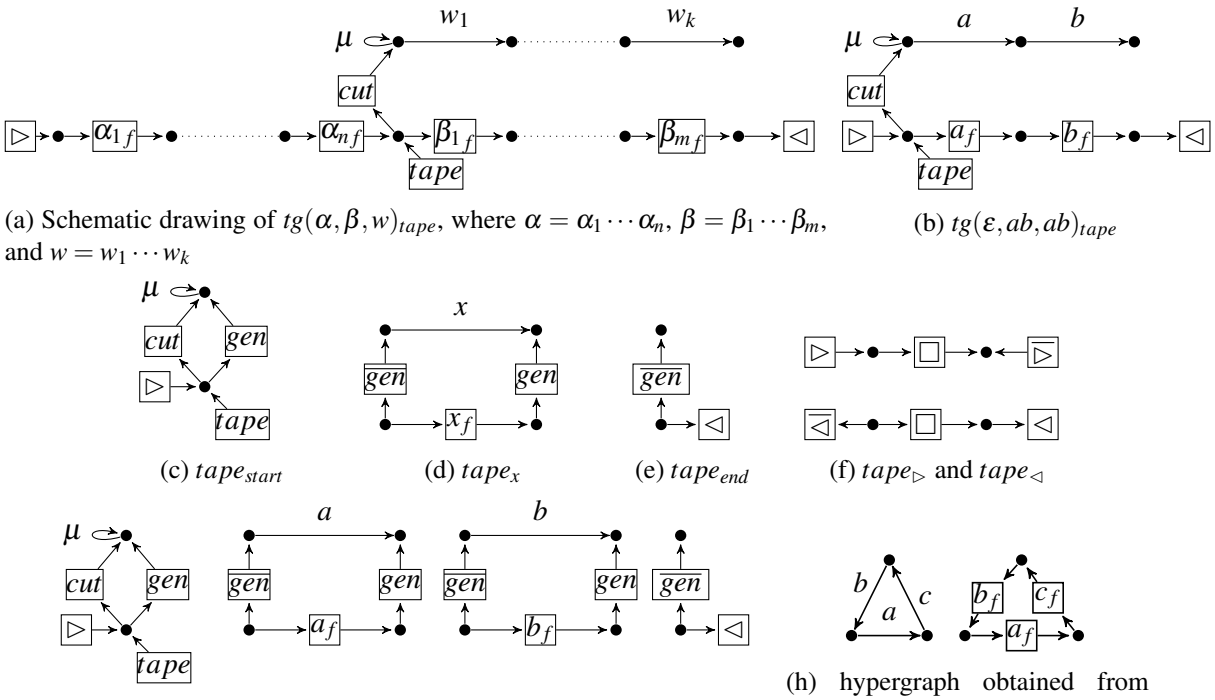
Definition 4. *Let $TM = (Q, \Omega, \Gamma, \Delta)$ be a Turing machine. Let $\sigma = q_1 \cdots q_{|Q|}$ be a sequence of states, where each state occurs exactly once. Define $hg(TM, \sigma) = (Q + \{v_{head}\}, \{acc, head, \overline{tape}\} + \Delta, s, t, lab)$, where $s(acc) = t(acc) = q_{accept}$, $lab(acc) = acc$, $s(head) = \sigma$, $t(head) = v_{head}$, $lab(head) = head$, $s(\overline{tape}) = v_{head}$, $t(\overline{tape}) = \varepsilon$, $lab(\overline{tape}) = \overline{tape}$, $s(\delta) = p$, $t(\delta) = q$, and $lab(\delta) = x/y/dir$, where $\delta = (p, x, y, dir, q) \in \Delta$.*

Example 1. *Consider the Turing machine in Figure 2a. The corresponding hypergraph is depicted in Figure 2b where the head-hyperedge is dashed.*

Remark 3. *The order in which the states of the Turing machine are connected to the sources of the head-hyperedge implements a permutation. σ is permuted when a transition step is simulated.*

5.2 Tape graph

In our construction the tape is represented by an infinitely extendable tape graph. Due to technical reasons, the tape graph contains two connected string graphs, where one is labeled over the terminal alphabet Ω and the other is labeled over the fusion alphabet Γ_f (defined later). The construction can be seen as having two tapes (input and working tape) initially with the same content, where the input tape is left invariant, but the working tape may be modified or extended by $\square$-cells. If the machine halts in the accepting state, then the content of the input tape is used as a contribution to the generated language.



(a) Schematic drawing of $tg(\alpha,\beta,w)_{tape}$, where $\alpha = \alpha_1 \cdots \alpha_n$, $\beta = \beta_1 \cdots \beta_m$, and $w = w_1 \cdots w_k$

(b) $tg(\varepsilon, ab, ab)_{tape}$

(c) $tape_{start}$

(d) $tape_x$

(e) $tape_{end}$

(f) $tape_{\triangleright}$ and $tape_{\triangleleft}$

(g) The connected components needed to generate $tg(\varepsilon, ab, ab)_{tape}$

(h) hypergraph obtained from $tape_a, tape_c, tape_b$

Figure 3: Tape graphs and connected components needed for their generation

In our construction five additional hyperedges are used. The terminal-labeled string graph carries a marker hyperedge. The two corresponding string graphs are connected via a hyperedge labeled *cut* which is used to disconnect the terminal- and marker-labeled string graph. A *tape*-hyperedge is connected to the first vertex of the fusion-labeled string graph and is later used for attaching the tape graph to a hypergraph representation of a Turing machine. Two hyperedges labeled $\triangleright$ and $\triangleleft$ are used to extend the fusion-symbol labeled string graph with $\square$-labeled hyperedges an unbounded number of times via the connected components $\boxed{\triangleright}\bullet\to\boxed{\square}\bullet\leftarrow\boxed{\triangleright}$ and $\boxed{\triangleleft}\bullet\leftarrow\boxed{\square}\bullet\to\boxed{\triangleleft}$ as well as the fusion rules $fr(\triangleright)$ and $fr(\triangleleft)$.

$\Gamma_f = (\Gamma \setminus \Omega) + \Omega_f$, where $\Omega_f = \{x_f \mid x \in \Omega\}$. This is because in fusion grammars fusion alphabets and terminal alphabets are disjoint but $\Omega \subsetneq \Gamma$ by definition of the Turing machine. For example, the string ab is represented by the graphs $sg(ab)$ and ${}_{\triangleright}sg(f(ab))_{\triangleleft} = \boxed{\triangleright}\to\bullet\to\boxed{a_f}\to\bullet\to\boxed{b_f}\to\bullet\to\boxed{\triangleleft}$, where $f\colon \Gamma^* \to \Gamma_f^*$ is defined by $f(x) = x_f$, if $x \in \Omega$, and $f(x) = x$, otherwise.

Definition 5. *Let $\alpha, \beta \in \Gamma^*, w \in \Omega^*$. Let cut, tape, $\triangleright$ and $\triangleleft$ be fusion symbols with $type(\triangleright) = (0,1)$, $type(\triangleleft) = type(tape) = (1,0)$, and $type(cut) = (1,1)$, and let $sg(cut \cdot w)_{\mu,tape}$ be the string graph $sg(cut \cdot w)$ with an additional μ-labeled loop attached to $begin(sg(w))$ and a tape-hyperedge attached to $begin(sg(cut))$. Then $tg(\alpha,\beta,w)_{tape} = ({}_{\triangleright}sg(f(\alpha\beta))_{\triangleleft} + sg(cut \cdot w)_{\mu,tape})/_{begin(sg(f(\beta)))\equiv s(cut)}$ is a* tape graph. *A schematic drawing is depicted in Figure 3a.*

We depict hyperedges with one source and one target with labels $x \in \Sigma \setminus F$ by $\bullet \xrightarrow[x]{} \bullet$.

Example 2. *The tape graph $tg(\varepsilon, ab, ab)_{tape}$ is depicted in Figure 3b.*

Because the input to the Turing machine may be any $w \in \Omega^*$ we need a construction to generate arbitrary tape graphs corresponding to inputs. This is realized by the following context-dependent fusion

grammar, where the fusion rule $fr(gen)$ is used to fuse gen- and $\overline{gen}$-hyperedges in order to generate two corresponding string graphs (one terminal, one fusion labeled), and the fusion rules $fr(\triangleright)$ and $fr(\triangleleft)$ are used to add $\square$-edges to the left and right of the fusion-labeled string graph as described above.

Definition 6. *Let $cut, \triangleright$ and $\triangleleft$ be as before. Let $F_{tg} = \{tape, gen, cut, \triangleright, \triangleleft\} + \Gamma_f$ be a fusion alphabet with $type(gen) = type(x) = (1,1)$ for each $x \in \Gamma_f$. Define the context-dependent fusion grammar $CDFG_{tg}(\Omega,\Gamma) = (Z_{tg}, F_{tg}, \{\mu\}, \Omega, \{fr(gen), fr(\triangleright), fr(\triangleleft)\})$, where the start hypergraph $Z_{tg} = tape_{start} + tape_{end} + \sum_{x\in\Omega} tape_x + tape_\triangleright + tape_\triangleleft$ consists of the connected components depicted in Figure 3c–3f.*

Example 3. *Let $\Omega = \{a,b,c\}$ and $\Gamma = \{a,b,c,\square\}$. Then $CDFG_{tg}(\Omega,\Gamma) = (Z_{example}, \{tape, gen, cut, \triangleright, \triangleleft, a_f, b_f, c_f, \square\}, \{\mu\}, \Omega, \{fr(gen), fr(\triangleright), fr(\triangleleft)\})$ with $Z_{example} = tape_{start} + tape_{end} + tape_a + tape_b + tape_c + tape_\triangleright + tape_\triangleleft$.*

The tape graph $tg(\varepsilon, ab, ab)_{tape}$, depicted in Figure 3b, can be generated by applying $fr(gen)$ three times to the connected components $tape_{start}, tape_a, tape_b$ and $tape_{end}$, depicted in Figure 3g. However, due to the context-freeness of $fr(gen)$ fusions within some connected component are also possible yielding e.g. the hypergraph in Figure 3h obtained from $tape_a, tape_c, tape_b$. Note that the left connected component is terminal labeled. However, it does not contribute to the generated language because it lacks a marker hyperedge.

The following propositions show that this context-dependent fusion grammar generates certain tape graphs. But everything derivable does not contribute to the generated language because there is no possibility to obtain a connected component which is only terminal labeled. However, with a slight modification of the grammar the generated language is Ω^* up to representation of strings as graphs.

Proposition 1. *For each $i, j \in \mathbb{N}, w \in \Omega^*$ exists a derivation $Z \overset{*}{\Longrightarrow} tg(\square^i, w \cdot \square^j, w)_{tape}$ in $CDFG_{tg}(\Omega,\Gamma)$.*

Proof. by induction on i, j and the length of w.

We first prove for each $w = w_1 \dots w_n \in \Omega^*$ exists a derivation $tape_{start} + tape_{end} + \sum_{x\in\Omega} tape_x \overset{n}{\Longrightarrow} tg(\varepsilon, w, w)_{tape}$. Therefore, let $\#\colon \Omega^* \times \Omega \to \mathbb{N}$ be a mapping of a string and a symbol to the number of occurrences of this symbol in the string.

Induction base: $n = 0$: Let m be a multiplicity with $m(tape_{start}) = m(tape_{end}) = 1$, $m(tape_x) = 0$ for each $x \in \Omega$. Then $tape_{start} + tape_{end} + \sum_{x\in\Omega} tape_x \underset{m}{\Longrightarrow} tape_{start} + tape_{end} \underset{fr(gen)}{\Longrightarrow} tg(\varepsilon,\varepsilon,\varepsilon)_{tape}$.

Induction step: $w = w_1 \dots w_{n+1}, n \geq 0$. Let $m(tape_{start}) = m(tape_{end}) = 1$, $m(tape_x) = \#(w,x)$ for each $x \in \Omega$. Then by induction hypothesis there is a derivation $tape_{start} + tape_{end} + \sum_{x\in\Omega} tape_x \underset{m}{\Longrightarrow} tape_{start} + tape_{end} + \sum_{x\in\Omega} m(tape_x) \cdot tape_x \overset{n}{\underset{fr(gen)}{\Longrightarrow}} tg(\varepsilon, w_1 \dots w_n, w_1 \dots w_n)_{tape} + tape_{w_{n+1}}$. Due to the context-freeness of $fr(gen)$ one may assume w.l.o.g. that the connected components involved in the last derivation step are $X + tape_{end}$ where X is $tg(\varepsilon, w_1 \dots w_n, w_1 \dots w_n)_{tape}$ without the $\triangleleft$-hyperedge but with an additional $\overline{gen}$-hyperedge attached to the ends of the string graphs. Then $X + tape_{w_{n+1}} + tape_{end} \overset{2}{\Longrightarrow} tg(\varepsilon, w_1 \dots w_{n+1}, w_1 \dots w_{n+1})_{tape}$.

Now we prove $tg(\alpha,\beta,w)_{tape} + tape_\triangleright \overset{i+1}{\Longrightarrow} tg(\square^i\alpha,\beta,w)_{tape}$ for any $tg(\alpha,\beta,w)_{tape}$. The proof for $tg(\alpha,\beta,w)_{tape} + tape_\triangleleft \overset{j+1}{\Longrightarrow} tg(\alpha,\beta\square^j,w)_{tape}$ for any $tg(\alpha,\beta,w)_{tape}$ is analog.

Induction base: $i = 0$. Let m be a multiplicity with $m(tg(\alpha,\beta,w)_{tape}) = 0$ and $m(tape_\triangleright) = 1$. Then $tg(\alpha,\beta,w)_{tape} + tape_\triangleright \underset{m}{\Longrightarrow} tg(\square^0\alpha,\beta,w)_{tape}$.

Induction step: Let m be a multiplicity with $m(tg(\alpha,\beta,w)_{tape}) = 1$ and $m(tape_\triangleright) = i+1$. Then $tg(\alpha,\beta,w)_{tape} + tape_\triangleright \underset{m}{\Longrightarrow} tg(\alpha,\beta,w)_{tape} + (i+1) \cdot tape_\triangleright \overset{i}{\underset{fr(\triangleright)}{\Longrightarrow}} tg(\square^i\alpha,\beta,w)_{tape} + tape_\triangleright \underset{fr(\triangleright)}{\Longrightarrow} tg(\square^{i+1}\alpha,\beta,w)_{tape}$. $\square$

Proposition 2. $L(CDFG_{tg}(\Omega,\Gamma)) = \emptyset$.

Proof. It is sufficient to focus on the connected components with some marker μ. The statement holds because there is no possibility to fuse the *cut*-hyperedge which is attached to the μ-hyperedge. Therefore, no connected component is only terminal and marker labeled. □

Proposition 3. *Let* $CDFG_{tg+\overline{cut}}(\Omega,\Gamma) = (Z_{tg} + z_{\overline{cut}}, F_{tg}, \{\mu\}, \Omega, P'_{tg})$, *where* $z_{\overline{cut}} = \bullet\!\!\rightarrow\boxed{\overline{cut}}\!\!\rightarrow\!\bullet$ *and* $P'_{tg} = P_{tg} \cup \{fr(cut)\}$. *Then* $L(CDFG_{tg+\overline{cut}}(\Omega,\Gamma)) = \{sg(w) \mid w \in \Omega^*\}$.

Proof. $Z \overset{*}{\Longrightarrow} tg(\square^i, w\cdot\square^j, w)_{tape} + z_{\overline{cut}} \underset{fr(cut)}{\Longrightarrow} {}_{\triangleright}sg(\square^i f(w)\square^j)_{\triangleleft} + sg(w)_{\mu}$ for any $i,j \in \mathbb{N}, w \in \Omega^*$, where the first part uses the same argument as in Proposition 1 and $fr(cut)$ matches the *cut*-hyperedge in the tape graph and the $\overline{cut}$-hyperedge in $z_{\overline{cut}}$. Because only the latter connected component contains a marker and is marker and terminal labeled it contributes to the language.

The converse follows from the fact that the derivation constructed above is a normal form because the derivation steps are interchangeable[2] due to the context-freeness. □

5.3 Hypergraph representation of a configuration

A hypergraph representation of a configuration consists of the hypergraph representation of the Turing machine fused to some tape graph. In this way the configuration is interlinked with a specific input and with some permutation of states. An initial configuration is obtained by fusing a *tape*-hyperedge in a hypergraph representation of the Turing machine with some $\overline{tape}$-hyperedge in a tape graph.

Example 4. *The application of the fusion rule* $fr(tape)$ *to the hypergraph representation of the Turing machine in Example 1 and the tape graph generated in Example 3 yields the hypergraph representation of the initial configuration* $(q_{start}, \varepsilon, ab)$ *wrt the input adjunct ab and the permutation* $q_{start}q_{aux}q_{accept}$ *depicted on the right-hand site in Figure 4.*

Definition 7. *Let* $c = (q,\alpha,\beta) \in conf(TM)$, *let* $w \in \Omega^*$, *let* $\sigma = q_1 \cdots q_{|Q|}$ *be a permutation of* Q *where* $q_1 = q$. *Then* $H = hg(\sigma,\alpha,\beta,w)$ *is the* hypergraph representation of the configuration c with adjunct w and permutation σ, *where* w *is the terminal labeled string graph in the tape graph.* H *is defined by* $hg(TM,\sigma) + tg(\alpha,\beta,w)_{tape} \underset{fr(tape)}{\Longrightarrow} H$.

5.4 Simulating steps of a Turing machine by context-dependent fusion rules

In order to simulate a step further connected components are needed. These connected components encode the substitution of a symbol on the tape, the movement of the head and the transition to the next state. In order to move the head to the left or to the right our construction takes both the current symbol and the symbol to the left of the head into account. The relations of the Turing machine can be seen as replacing or deleting the current symbol and (maybe) inserting a new symbol left or right of the head. In the graph representation this corresponds to deleting and inserting edges. These deletions and insertions are done with respective fusions of complementary labeled hyperedges. The hypergraphs in Figure 5a–5c are schematic drawings of the connected components used for simulating a step of a Turing machine. The dots indicate that there are $|Q|$ vertices as sources. Two complementary *head* and $\overline{head}$-hyperedges attached to the same vertices are part of this connected component, where the ordering

[2] In [7] it is shown that every derivation can be rearranged such that first all multiplications and afterwards all (context-free) fusions in arbitrary order are performed.

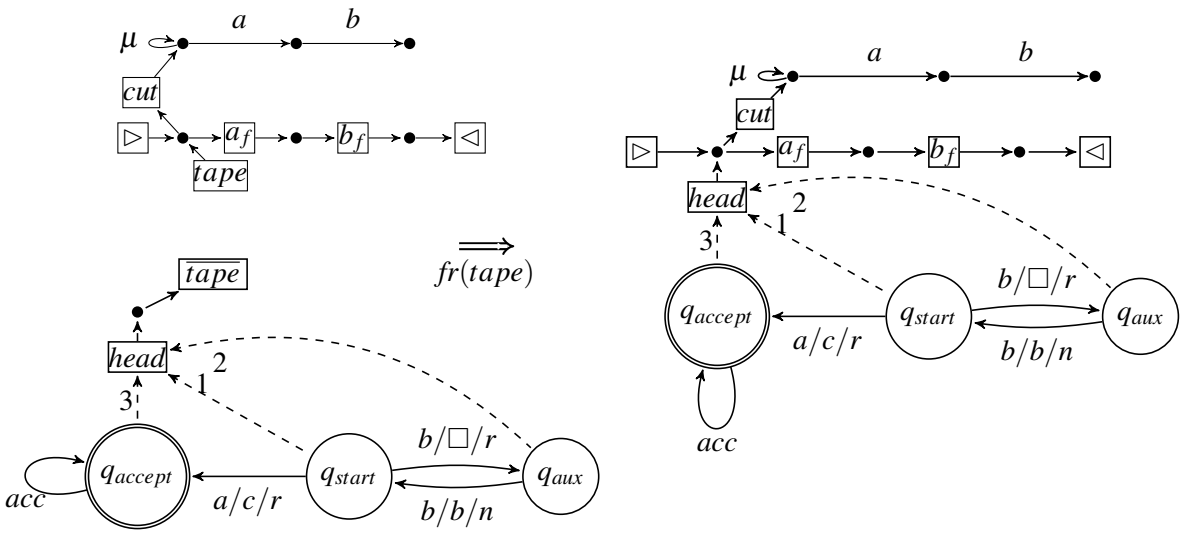
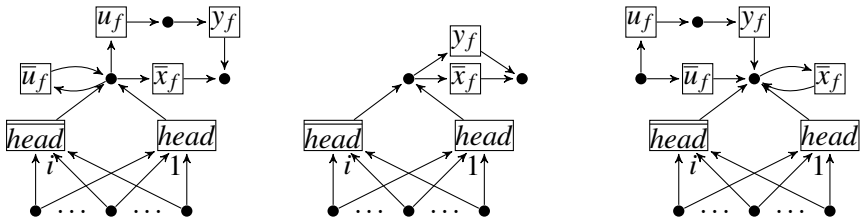


Figure 4: Fusion of the hypergraph representation of the Turing machine in Example 1 and the tape graph generated in Example 3 yields the configuration $(q_{start}, \varepsilon, ab)$ wrt the input adjunct ab and the permutation $q_{start}q_{aux}q_{accept}$.



(a) $C(u,x/y/l,i)$ (b) $C(u,x/y/n,i)$ (c) $C(u,x/y/r,i)$

Figure 5: Schematic drawings of connected components used for simulating a step of a Turing machine

of the source attachments of the *head*-hyperedge implements a permutation such that the first and ith source are swapped. Formally, these components are defined as follows.

Definition 8. *Let $\{head\}+\Gamma_f$ be a fusion alphabet with $type(head) = (|Q|,1)$ and $type(x) = (1,1)$ for each $x \in \Gamma_f$. Let $\Lambda = \{x/y/dir \mid x,y \in \Gamma, dir \in \{l,n,r\}\}$.*

Define for each $u \in \Gamma_f, x/y/l \in \Lambda, i \in \{1,\ldots,|Q|\}$ the connected component $C(u,x/y/l,i) = (\{v_1,\ldots,v_{|Q|+3}\},\{head,\overline{head},\overline{u},u,\overline{x},y\},s,t,lab)$ where $s(head) = v_1v_2\cdots v_i\cdots v_{|Q|}$, $t(head) = v_{|Q|+1}$, $lab(head) = head$, $s(\overline{head}) = v_iv_2\cdots v_1\cdots v_{|Q|}$, $t(\overline{head}) = v_{|Q|+1}$, $lab(\overline{head}) = \overline{head}$, $s(\overline{u}) = t(\overline{u}) = v_{|Q|+1}$, $lab(\overline{u}) = \overline{u}_f$, $s(u) = v_{|Q|+1}$, $t(u) = v_{|Q|+3}$, $lab(u) = u_f$, $s(\overline{x}) = v_{|Q|+1}$, $t(\overline{x}) = v_{|Q|+2}$, $lab(\overline{x}) = \overline{x}_f$, $s(y) = v_{|Q|+2}$, $t(y) = v_{|Q|+3}$, $lab(y) = y_f$.

Define for each $x/y/n \in \Lambda, i \in \{1,\ldots,|Q|\}$ the connected component $C(u,x/y/n,i) = (\{v_1,\ldots,v_{|Q|+2}\},\{head,\overline{head},\overline{x},y\},s,t,lab)$ where $s(head) = v_1v_2\cdots v_i\cdots v_{|Q|}$, $t(head) = v_{|Q|+1}$, $lab(head) = head$, $s(\overline{head}) = v_iv_2\cdots v_1\cdots v_{|Q|}$, $t(\overline{head}) = v_{|Q|+1}$, $lab(\overline{head}) = \overline{head}$, $s(\overline{x}) = s(y)v_{|Q|+1}$, $t(\overline{x}) = t(y) = v_{|Q|+2}$, $lab(\overline{x}) = \overline{x}_f$, $lab(y) = y_f$.

Define for each $u \in \Gamma_f, x/y/r \in \Lambda, i \in \{1,\ldots,|Q|\}$ the connected component $C(u,x/y/r,i) = (\{v_1,\ldots,v_{|Q|+3}\},\{head,\overline{head},\overline{x},u,\overline{u},y\},s,t,lab)$ where $s(head) = v_1v_2\cdots v_i\cdots v_{|Q|}$, $t(head) = v_{|Q|+2}$,

$lab(head) = head$, $s(\overline{head}) = v_i v_2 \cdots v_1 \cdots v_{|Q|}$, $t(\overline{head}) = v_{|Q|+2}$, $lab(\overline{head}) = \overline{head}$, $s(\overline{x}) = t(\overline{x}) = v_{|Q|+2}$, $lab(\overline{x}) = \overline{x}_f$, $s(u) = v_{|Q|+1}$, $t(u) = v_{|Q|+3}$, $lab(u) = u_f$, $s(\overline{u}) = v_{|Q|+1}$, $t(\overline{u}) = v_{|Q|+2}$, $lab(\overline{u}) = \overline{u}_f$, $s(y) = v_{|Q|+2}$, $t(e_4) = v_{|Q|+3}$, $lab(y) = y_f$.

In order to simulate a step of a Turing machine context-dependent fusion rules are needed. Some of the context conditions derive directly from the semantics of a Turing machine. For example, the step $(p, \alpha u, x, \beta) \vdash_{TM} (q, \alpha, u, y\beta)$ can only be applied if $(p, x, y, l, q) \in \Delta$, the Turing machine is in state p and reads the symbol x. Other context conditions are needed because (context-dependent) fusion rules can only consume two complementary hyperedges in one derivation step but a step of a Turing machine is much more complicated (head movement, tape manipulation, state transition). Hence, several rules and rule applications are needed to simulate such a step. Furthermore, in our construction positive and negative context conditions are needed to restrict the application to obtain a correct and sound simulation. The set of context-dependent fusion rules P_Δ specified in Definition 9 contains rules with respect to the fusion symbol *head* and rules with respect to the fusion symbol x for each $x \in \Gamma$. The first are used to fuse the *head*-hyperedge in the graph representation of a configuration with the correct connected component used for simulating the step of the Turing machine and perform the state transition. The latter are used to modify the tape and move the head correctly.

Definition 9. *Define P_Δ as the following set of context-dependent fusion rules.*

$$P_\Delta = \{\Delta(u,\lambda) \mid u \in \Gamma, \lambda \in \Lambda\} \cup \{fuse\_2in(x), fuse\_2out(x), fuse\_loop\_in(x), fuse\_loop\_out(x) \mid x \in \Gamma\}$$

$\Delta(u,\lambda) = (fr(head), \{fr(head) \to PC(u,\lambda,j) + C(u,\lambda,j)\}, \{fr(head) \to twoin(u) + \overline{h}^\bullet \mid u \in \Gamma\}) \cup \{fr(head) \to twoout(u) + \overline{h}^\bullet \mid u \in \Gamma\})$, *where the morphism in the positive context maps the head-hyperedge to the one in $PC(u,\lambda,j)$, depicted in Figure 6a, and the $\overline{head}$-hyperedge to the one in $C(u,\lambda,j)$. Note that, the connected component $C(u,\lambda,j)$ is induced by the jth source of the head-hyperedge and the parameters u,λ. $twoin(u)$, $twoout(u)$ and $\overline{h}^\bullet$ are depicted in Figure 6b, 6c and 6d, respectively.*

$fuse\_2in(x) = (fr(x), \{fr(x) \to twoin(x)\}, \{fr(x) \to twoin2h(x), fr(x) \to loop/in(x), fr(x) \to tri(x)\} \cup \{fr(x) \to twoinextraloop(x,z) \mid z \in \Gamma\})$, *where $fr(x) \to tri(x)$ maps the x-hyperedge to $e \in E_{tri(x)}$ with $s(e) = v_2$ and $t(e) = v_3$; $fr(x) \to twoinextraloop(x,z)$ maps the $\overline{x}$-hyperedge to the one above the the x-hyperedge (which is relevant for the case $z = x$). The respective hypergraphs used in the context conditions are depicted in Figure 6.*

$fuse\_2out(x) = (fr(x), \{fr(x) \to twoout(x)\}, \{fr(x) \to twoout2h(x), fr(x) \to loop/out(x), fr(x) \to tri(x)\} \cup \{fr(x) \to twooutextraloop(x,z) \mid z \in \Gamma\})$ *is analog but $fr(x) \to tri(x)$ maps the x-hyperedge to $e \in E_{tri(x)}$ with $s(e) = v_1$ and $t(e) = v_2$.*

$fuse\_loop\_in(x) = (fr(x), \{fr(x) \to PCloopin(x)\}, \emptyset)$, *where $PCloopin(x)$ is depicted in Figure 6l and in the positive context condition the x-hyperedge matches the one with target v and the $\overline{x}$-hyperedge matches the one with source and target v.*

$fuse\_loop\_out(x) = (fr(x), \{fr(x) \to PCloopout(x)\}, \emptyset)$, *is analog but the x-hyperedge matches the one with source v. $PCloopout(x)$ is depicted in Figure 6m.*

Remark 4. *The rules in the latter subset of P_Δ are constructed in such a way that the two complementary hyperedges must be attached to the same vertex and that two rules are never applicable at the same time with respect to the same connected component. This is achieved by defining $fuse\_2in(x)$ and $fuse\_loop\_in(x)$ as well as $fuse\_2out(x)$ and $fuse\_loop\_out(x)$ in such a way that the positive context condition of the first is reflected in the negative context condition of the other. Further, in order to distinguish the applicability of $fuse\_loop\_in(x)$ and $fuse\_loop\_out(x)$ if in-going and out-going edges (wrt*

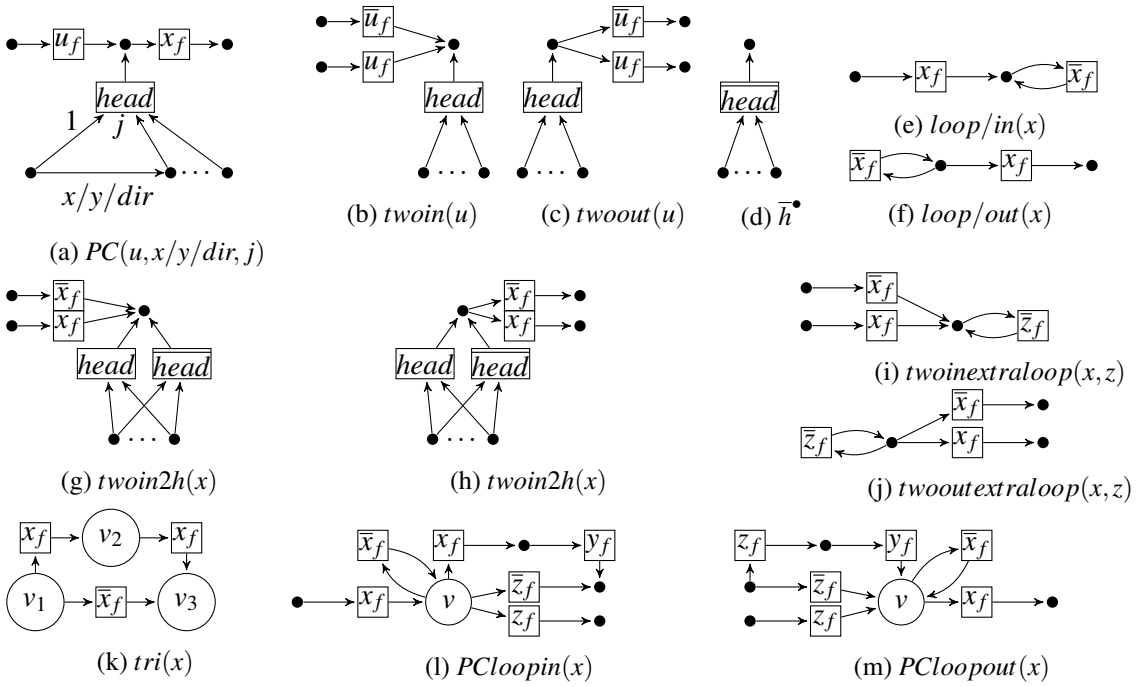


Figure 6: Some contexts for applying rules simulating a step

some vertex where the loop is attached) are labeled with the same symbol, the number of out-going (in-going) edges, respectively, is of relevance. If the number of out-going (in-going) edges is 4, then the loop must be fused with the in-going (out-going) edge, respectively. $fuse\_loop\_in(x)$ *and* $fuse\_loop\_out(x)$ *do not need negative context conditions because the positive context does not occur in the C-components of Definition 8.*

The following example illustrates the simulation of the transition step for $(q_{start},a,c,r,q_{accept})$. It works analogously for other transition steps.

Example 5. *Consider the two connected components* $H = hg(q_{start}q_{aux}q_{accept},d,ab,ab)$ *and* $C = C(d, a/c/r,3)$. *Then* $H' = hg(q_{accept}q_{aux}q_{start},dc,b,ab)$ *can be derived as illustrated in Figure 7. The context-dependent fusion rule* $\Delta(d,a/c/r)$ *can be applied by matching the head-hyperedge in H and the* $\overline{head}$*-hyperedge in C. The two complementary hyperedges are fused and due to the head-hyperedge in C a new head-hyperedge reconstructed; its first source is* q_{accept}. *Furthermore, the application attaches additional vertices and hyperedges to the tape graph. The resulting connected component is depicted in the lower left in Figure 7 (the acc-loop and the adjunct, consisting of the cut-hyperedge and terminal and marker string graph, is omitted in order to clarify the drawing). The* $\overline{a}_f$*-hyperedge is then fused with the out-going* a_f*-hyperedge by an application of* $fuse\_loop\_out(a)$ *with the result that the source and the target vertex of the* a_f*-hyperedge are glued together yielding the connected component depicted in the lower right in Figure 7 (again omitting the acc-loop and the adjunct). Afterwards,* $fuse\_2in(d)$ *can be applied to the* $\overline{d}_f$*- and* d_f*-hyperedge. The resulting connected component is* H'.

We will later prove this one-to-one correspondence of a transition step in *TM* and a particular derivation sequence concerning hypergraph representations of respective configurations wrt the same adjunct in the corresponding context-dependent fusion grammar $CDFG(TM)$.

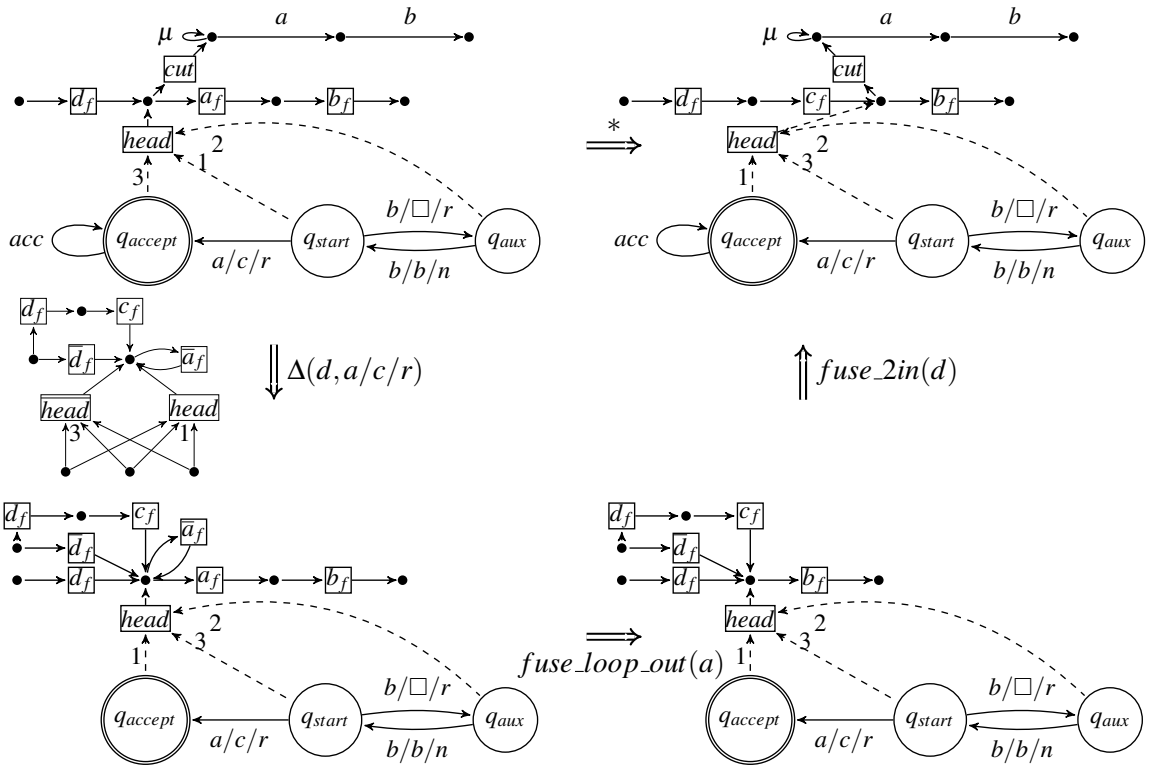


Figure 7: Simulating the transition step for $(q_{start}, a, c, r, q_{accept})$

5.5 Construction of a context-dependent fusion grammar corresponding to a Turing machine

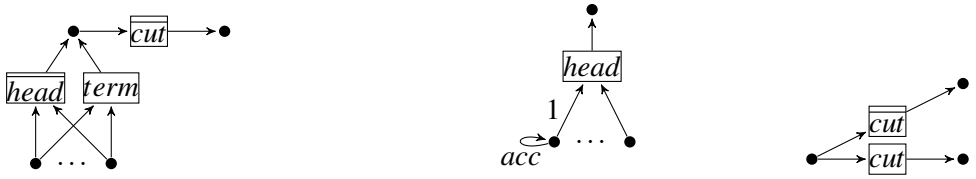
Now we combine the previous constructions in a context-dependent fusion grammar. In our construction two additional connected components Acc and ${}_{\triangleright}sg(\overline{\square})_{\triangleleft}$ and four additional (context-dependent) fusion rules are needed (1) for connecting the initial hypergraph representation of the Turing machine with the generated tape graph (as described in Definition 7), (2) for manipulating a hypergraph representation of an accepting configuration such that it indicates that an input string is accepted, (3) for disconnecting[3] the terminal and marker labeled string graph $sg(w)_\mu$ for some $w \in \Omega^*$, and (4) for removing a $\square$-symbol from the left on the tape (for technical reasons).

Definition 10. *Let $TM = (Q, \Omega, \Gamma, \Delta)$ be a Turing machine. Let $CDFG_{tg}(\Omega,\Gamma) = (Z_{tg}, F_{tg}, \{\mu\}, \Omega, P_{tg})$ be the corresponding context-dependent fusion grammar generating tape graphs (cf. Definition 6), let $C(u,\lambda,i)$ be the connected component defined in Definition 8 and P_Δ be the set of context-dependent fusion rules defined in Definition 9. Then $CDFG(TM) = (Z_{TM}, \{head\} + F_{tg}, \{\mu\}, \{term\} + \Omega + \Lambda, P_{TM})$ is the corresponding context-dependent fusion grammar with Z_{TM} and P_{TM} defined as follows.*

$$Z_{TM} = Z_{tg} + hg(TM)_{init} + Acc +_{\triangleright} sg(\overline{\square})_{\triangleleft} + \sum_{\substack{u\in\Gamma_f,\lambda\in\Lambda \\ 0<i\le|Q|}} C(u,\lambda,i) \text{ where } hg(TM)_{init} = hg(TM, q_{start}\sigma)$$

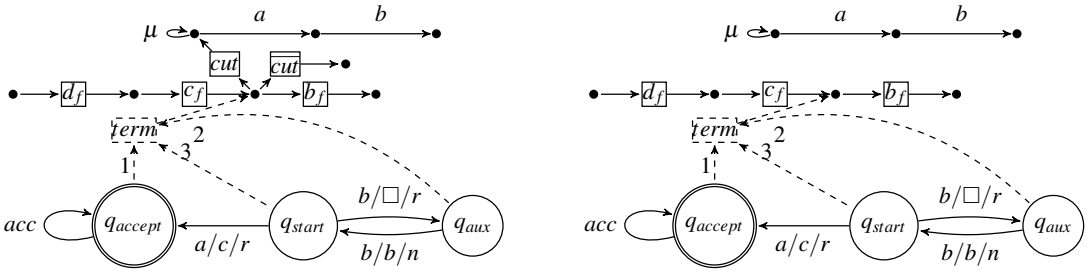
for some arbitrary permutation $\sigma \in Q \setminus \{q_{start}\}$; and $Acc = (\{v_1,\ldots,v_{|Q|+2}\}, \{term, \overline{head}, cut\}, s, t, lab)$ where $s(term) = v_1 \cdots v_{|Q|}$, $t(term) = v_{|Q|+1}$, $lab(term) = term$, $s(\overline{head}) = v_1 \cdots v_{|Q|}$, $t(\overline{head}) = v_{|Q|+1}$,

[3]Disconnecting $sg(w)_\mu$ uses similar ideas as discussed in Proposition 3 but uses context conditions.



(a) *Acc* (b) PC_{acc} (c) PC_{cut}

Figure 8: Schematic drawings of connected components used for acceptance and for disconnecting the terminal and marker labeled string graph.



(a) H'': hypergraph representation of a terminated Turing machine after applying *accept*

(b) H''': two connected components obtained by applying *cut* to H''

Figure 9: Connected components obtained after the application of *accept* and *cut*.

$lab(\overline{head}) = \overline{head}$, $s(cut) = v_{|Q|+1}$, $t(cut) = v_{|Q|+2}$, $lab(cut) = cut$. *A schematic drawings of Acc is depicted in Figure 8a.*

$P_{TM} = P_{tg} \cup P_{\Delta} \cup \{fr(tape), accept, cut, shrink\}$ *where* $accept, cut$ *and* $shrink$ *are defined as follows:*

$accept = (fr(head), \{fr(head) \to PC_{acc} + Acc\}, \emptyset)$, *where* $PC_{acc} = (Q + \{v_{tape}\}, \{head, acc\}, s, t, lab)$ *with* $s(head) = q_{accept}\sigma'$, $t(head) = v_{tape}$, $lab(head) = head$, $s(acc) = t(acc) = q_{accept}$, *and* $lab(acc) = acc$, *where* σ' *are the states of* $Q \setminus \{q_{accept}\}$ *in arbitrary order. A schematic drawings of* PC_{acc} *is depicted in Figure 8b.*

$cut = (fr(cut), \{fr(cut) \to (\{v_1, v_2, v_3\}, \{cut, \overline{cut}\}, s, t, lab)\}, \emptyset)$, *where* $s(cut) = s(\overline{cut}) = v_1, t(cut) = v_2, t(\overline{cut}) = v_3$ *and* $lab(cut) = cut, lab(\overline{cut}) = \overline{cut}$, *i.e., the source of the two complementary hyperedges must be the same and the targets different. The connected component is depicted in Figure 8c.*

$shrink = (fr(\Box), \{fr(\Box) \to \triangleright \to \bullet \to \Box \to \bullet \quad \bullet \to \overline{\Box} \to \bullet \leftarrow \triangleright\}, \emptyset)$.

Example 6. *Reconsidering the hypergraph* $H' = hg(q_{accept}q_{aux}q_{start}, dc, b, ab)$ *depicted in the upper right in Figure 7. Then it is easy to see that* $H' + Acc \underset{accept}{\Longrightarrow} H'' \underset{cut}{\Longrightarrow} H'''$ *where* H'' *and* H''' *are depicted in Figure 9. Note that, the upper part of* H''' *is terminal and marker labeled only.*

Our main theorem is that the language recognized by some Turing machine and the generated language of the corresponding context-dependent fusion grammar coincide up to representation of strings as graphs.

Theorem 1. $L(CDFG(TM)) = \{sg(w) \mid w \in L(TM)\}$.

The proof is based on the following lemmata.

Lemma 1. *Let* $c_i = (p, \alpha, \beta), c_{i+1} = (q, \alpha', \beta') \in conf(TM), \delta = (p, x, y, dir, q) \in \Delta$, $w \in \Omega^*$, *and let* $pr(s(head), k) = q$ *for the head-labeled hyperedge in* $H = hg(pq_2 \cdots q \cdots q_{|Q|}, \alpha, \beta, w)$. *Then* $c_i \vdash_{TM}$

c_{i+1} wrt δ if and only if $H+C(u,x/y/dir,k)+tape_{\triangleright}+tape_{\triangleleft}+_{\triangleright} sg(\overline{\square})_{\triangleleft} \overset{*}{\Longrightarrow} H'$, where u is the last symbol of α if $\alpha \neq \varepsilon$ and $\square$ otherwise, and $H' = hg(qq_2 \cdots p \cdots q_{|Q|}, \alpha', \beta', w)$.

Proof. If $c_i = c_{i+1}$, then the one-to-one correspondence trivially holds. Assume that $c_i \vdash_{TM} c_{i+1} \neq c_i$ wrt δ holds. We distinguish the following six cases.

Case 1) $(p, \alpha u, x\beta) \vdash_{TM} (q, \alpha, uy\beta)$ wrt $\delta = (p,x,y,l,q)$. Then we can restrict the hypergraph to $H + C(u,x/y/l,k)$ (by multiplying unneeded connected components by 0) and apply the derivation $H + C(u,x/y/l,k) \underset{\Delta(u,x/y/l)}{\Longrightarrow} X \underset{fuse\_loop\_in(u)}{\Longrightarrow} X' \underset{fuse\_2out(x)}{\Longrightarrow} H'$ due to the following reasoning. Because u is the symbol left of x and $pr(s(head),k) = q$ the rule $\Delta(u,x/y/dir)$ can be applied matching the *head*-hyperedge in H and the $\overline{head}$-hyperedge in $C(u,x/y/dir,k)$ yielding the connected component X which is H extended by the respective hypergraphs $hg(u,x/y/l)$, where $v \in V_{hg(u,x/y/l)}$ and $v_{tape} \in V_H$ are identified. Because of the additional edges $fuse\_loop\_in(u)$ becomes applicable and the derivation $X \underset{fuse\_loop\_in(u)}{\Longrightarrow} X'$ fuses the previously added $\overline{u}$-loop and the in-going u-edge in X. Afterwards, $fuse\_2out(x)$ is applicable and the derivation $X' \underset{fuse\_2out(x)}{\Longrightarrow} H'$ yields the requested hypergraph due to the fact that the two out-going x_f- and $\overline{x}_f$-hyperedges are fused.

The other cases use similar arguments and are therefore stated less explicit.

Case 2) $(p, \varepsilon, x\beta) \vdash_{TM} (q, \varepsilon, \square y\beta)$ wrt $\delta = (p,x,y,l,q)$. Then we can restrict the hypergraph to the three connected components $H + C(\square,x/y/l,k) + tape_{\triangleright}$ and apply the derivation $H + C(\square,x/y/l,k) + tape_{\triangleright} \underset{fr(\triangleright)}{\Longrightarrow} \tilde{H} + C(\square,x/y/l,k) \underset{\Delta(\square,x/y/l)}{\Longrightarrow} X \underset{fuse\_loop\_in(\square)}{\Longrightarrow} X' \underset{fuse\_2out(x)}{\Longrightarrow} H'$, where $\tilde{H}$ is H but $\alpha = \square$ instead of ε; afterwards the same reasoning as the previous case is applied.

Case 3) $(p, \alpha, \varepsilon) \vdash_{TM} (q, \alpha, y)$ wrt $\delta = (p, \square, y, l, q)$. In the subcase $\alpha \neq \varepsilon$ we have the derivation $H + C(\square,x/y/l,k) + tape_{\triangleleft} \underset{fr(\triangleleft)}{\Longrightarrow} \tilde{H} + C(\square,\square/y/l,k) \underset{\Delta(\square,x/y/l)}{\Longrightarrow} X \underset{fuse\_loop\_in(\square)}{\Longrightarrow} X' \underset{fuse\_2out(x)}{\Longrightarrow} H'$ similar to the previous case and in the subcase $\alpha = \varepsilon$ all five connected components are needed and the derivation of the previous subcase is prepended by $\underset{fr(\triangleright)}{\Longrightarrow}$ and appended by $\underset{shrink}{\Longrightarrow}$.

Case 4) $(p, \alpha, x\beta) \vdash_{TM} (q, \alpha, y\beta)$ wrt $\delta = (p,x,y,n,q)$. Then we have the derivation $H + C(u,x/y/n,k) \underset{\Delta(u,x/y/n)}{\Longrightarrow} X \underset{fuse\_2out(x)}{\Longrightarrow} H'$.

Case 5) $(p, \alpha, x\beta) \vdash_{TM} (q, \alpha y, \beta)$ wrt $\delta = (p,x,y,r,q)$. Subcase $\alpha \neq \varepsilon$ is analog to Case 1 yielding the derivation $H + C(u,x/y/r,k) \underset{\Delta(u,x/y/r)}{\Longrightarrow} X \underset{fuse\_loop\_out(u)}{\Longrightarrow} X' \underset{fuse\_2in(x)}{\Longrightarrow} H'$ and subcase $\alpha = \varepsilon$ is analog to the respective subcase in Case 3.

Case 6) $(p, \alpha, \varepsilon) \vdash_{TM} (q, \alpha y, \varepsilon)$ wrt $\delta = (p, \square, y, r, q)$ is analog to Case 3 wrt both cases.

Conversely, given $H + C(u,x/y/dir,k) + tape_{\triangleright} + tape_{\triangleleft} +_{\triangleright} sg(\overline{\square})_{\triangleleft} \overset{*}{\Longrightarrow} H' \neq H$. This derivation can be reduced to the six cases above. The applicability of $\Delta(u,x/y/dir)$ to $H + C(u,x/y/dir,k)$ implies that there exists an $x/y/dir$-edge between some vertices $p,q \in V_H$ and that the current symbol read is x, where $pr(s(head),1) = p$ and $pr(s(head),k) = q$ for $head \in E_H, lab(head) = head$. Furthermore, the construction of $\Delta(u,x/y/dir)$ and $C(u,x/y/dir,k)$ gives that $pr(s(head),0) = q$ for the reconstructed hyperedge $head \in E_{H'}$. This implies $c_i \vdash_{TM} c_{i+1}$ wrt $\delta = (p,x,y,dir,q)$. □

Lemma 2. *$c_0 \vdash_{TM}^{k} c_k = (q_k, \alpha, \beta)$ wrt input w implies $Z \overset{*}{\Longrightarrow} Z + hg(q_k\sigma, \alpha, \beta, w)$ for some $\sigma \in Q \setminus \{q_k\}$.*

Proof. Induction base: $k = 0$. For each $w = w_1 \cdots w_n \in \Omega^*$ exists a derivation in $CDFG(TM)$ such that $hg(TM)_{init} + w_{start} + w_{end} + \sum_{i=1}^{n} w_{w_i} \overset{n+2}{\Longrightarrow} H_0 = hg(q_{start}\sigma, \varepsilon, w, w)$ using $fr(tape)$ once and $fr(gen)$ $n+1$-

times due to Proposition 1. Consequently, $Z \underset{m}{\Longrightarrow} Z + hg(TM)_{init} + w_{start} + w_{end} + \sum_{i=1}^{n} w_{w_i} \overset{n+2}{\Longrightarrow} Z + H_0$.

Induction step: $c_0 \vdash_{TM}^{k+1} c_{k+1}$ implies $c_0 \vdash_{TM}^{k} c_k \vdash_{TM} c_{k+1}$ for some $c_k = (q_k, \alpha', \beta')$. By induction hypothesis $c_0 \vdash_{TM}^{k} c_k$ implies $Z \overset{*}{\Longrightarrow} Z + H_k$, where $H_k = hg(q_k\sigma', \alpha', \beta', w)$ for some σ'. Let $c_k \vdash_{TM} c_{k+1}$ be wrt $\delta = (q_k, x, y, dir, q_{k+1})$. By construction of $CDFG(TM)$ there exists $C(u, x/y/dir, j) \in \mathscr{C}(Z)$ for each $(p, x, y, dir, q) \in \Delta, u \in \Gamma_f, 1 \leq j \leq |Q|$. Consequently, such a connected component also exists for δ such that $pr(s(head), 1) = q_k$ and $pr(s(head), j) = q_{k+1}$. Let C_δ be this suitable connected component. Then there is a derivation $Z + H_k \underset{m}{\Longrightarrow} Z + H_k + C_\delta + tape_{\triangleright} +_{\triangleright} sg(\overline{\square})_{\triangleleft} \overset{*}{\Longrightarrow} Z + H_{k+1}$, where m is a multiplication, $H' = hg(q_{k+1}\sigma, \alpha, \beta, w)$ and σ is the same as σ' except that q_{k+1} is replaced by q_k, due to Lemma 1. □

Definition 11. *$hg(TM, \sigma, \alpha, \beta, w)_{acc}$ is a connected component isomorphic to $hg(TM, \sigma, \alpha, \beta, w)$ but the label of the head-hyperedge is term.*

Proof. of Theorem 1. We show first $w \in L(TM)$ implies $sg(w) \in L(CDFG(TM))$. Let $w \in L(TM)$. Then $c_0 = (q_{start}, \varepsilon, w) \vdash_{TM}^{*} c_a = (q_{accept}, \alpha, \beta)$ for some $w \in \Omega^*, \alpha, \beta \in \Gamma^*$. Then

$$
\begin{aligned}
& Z \overset{*}{\Longrightarrow} Z + hg(q_{accept}\sigma, \alpha, \beta, w) \quad \text{due to Lemma 2} \\
& \underset{m}{\Longrightarrow} Acc + hg(q_{accept}\sigma, \alpha, \beta, w) \quad \text{where } m(x) = \begin{cases} 1 & x \in \{Acc, hg(q_{accept}\sigma, \alpha, \beta, w)\} \\ 0 & \text{otherwise} \end{cases} \\
& \underset{accept}{\Longrightarrow} hg(q_{accept}\sigma, \alpha, \beta, w)_{acc} \\
& \underset{cut}{\Longrightarrow} (hg(TM, q_{accept}\sigma) +_{\triangleright} sg(\alpha, \beta)_{\triangleleft})/_{begin(sg(\beta)) \equiv t(head)} + sg(w)_{\mu}.
\end{aligned}
$$

Hence, $sg(w) \in L(CDFG(TM))$.

The converse is more complicated to show. $sg(w) \in L(CDFG(TM))$ means there is a derivation $Z \overset{*}{\Longrightarrow} H$ with $Y \in \mathscr{C}(H), H \in \mathscr{H}_{\{\mu, term\}+\Lambda+\Omega} - \mathscr{H}_{\{term\}+\Lambda+\Omega}$ and $rem_{\mu}(Y) = sg(w)$. Without loss of generality, one can assume:

- $H = Y$ because the other connected components can be multiplied by 0.
- There is exactly one marker component in each derived hypergraph because two marked derived hypergraphs can never be fused with each other.
- The set of sources of the hypergraph representation of a Turing machine (and extended connected components) is Q because source vertices of these connected components cannot be fused with each other.
- All necessary multiplications are done as first derivation step and all applications of context-free fusion rules ($fr(gen), fr(\triangleright), fr(\triangleleft)$ and $fr(tape)$), are done before any application of some context-dependent fusion rule with context conditions.

Moreover, some of the rules are sequentially dependent with respect to the same connected component.

1. *cut* and *accept* are sequentially dependent, because the *cut*-hyperedge required in positive context condition of the *cut*-rule is added to v_{tape} by the application of *accept* to $hg(q_{accept}\sigma, \alpha, \beta, w) + Acc$.
2. $\Delta(u, \lambda)$ and *accept* are sequentially dependent, because $q_{start} \neq q_{accept}$.

3. $\Delta(u,\lambda)$ and and some rule r in the latter set in P_Δ are sequentially dependent, because the complementary edges attached to v_{tape} required by r are attached by the application of $\Delta(u,\lambda)$.
4. $fr(gen)$,$fr(tape)$ and $\Delta(u,\lambda)$ are sequentially dependent, because only if the tape graph is attached to the *head*-hyperedge of some hypergraph derived from $hg(TM)_{init}$, then the positive context conditions of $\Delta(u,\lambda)$ are satisfied.

Furthermore, the positive and negative context conditions restrict the fusion process dramatically.

1. *cut* is only applicable to $hg(q_{accept}\sigma,\alpha,\beta,w)_{acc}$ for arbitrary σ,α,β,w.
2. No two $\Delta(u_1,\lambda_1)$ and $\Delta(u_2,\lambda_2)$ are applicable to some hypergraph representation of some configuration directly one after the other, because the application of $\Delta(u_1,x_1/y_1/dir_1)$ attaches a $\overline{u_1}$- (if $dir_1 \in \{l,r\}$) and a $\overline{x_1}$-hyperedge to v_{tape}, hence, the negative context conditions of $\Delta(u_2,\lambda_2)$ are not satisfied.
3. $\Delta(u,\lambda)$ may only be applicable to some $hg(q_1\cdots q_{|Q|},\alpha u,x\beta,w)$ and $C(u,\lambda,j)$, where $\lambda = x/y/dir$ and $q_1\cdots q_{|Q|},\alpha,u,x,\beta',w$ are arbitrary. No fusion is possible wrt $C(u_1,\lambda_1,j)+C(u_2,\lambda_2,k)+Acc$.
4. A rule of the latter set in P_Δ is only applicable to the connected component obtained by the fusion wrt $\Delta(u,\lambda)$. No fusion is possible inside $C(u,\lambda,k)$.

The last argument is that the context-dependent fusion rule *accept* can only be applied if there exists a match into some connected component derived from $hg(TM)_{init}$ and Acc. The restriction to $hg(TM)_{init}$ comes from the fact, that the *head*-hyperedge must not be part of some C connected component. Hence, *accept* and *cut* are only applicable to a hypergraph representation of a configuration wrt input w if and only if $w \in L(TM)$. Moreover, *accept* and *cut* can be delayed to the end of the derivation.

Let $C_1,\ldots,C_n$ be the C-components in the order in which they are used in the derivation $Z \overset{*}{\Longrightarrow} sg(w)_\mu$. Then, using the remarks above, one can rearrange the derivation such that is is of the form

$$Z \underset{m}{\Longrightarrow} X_0 \underset{fr(gen)}{\overset{*}{\Longrightarrow}} X_1 \underset{fr(\triangleright)}{\overset{*}{\Longrightarrow}} X_2 \underset{fr(\triangleleft)}{\overset{*}{\Longrightarrow}} X_3 \underset{fr(tape)}{\overset{*}{\Longrightarrow}} hg(\sigma_0,\alpha_0,\beta_0,w)+Acc+\sum_{i=1}^{n} C_i \overset{*}{\Longrightarrow} hg(\sigma_1,\alpha_1,\beta_1,w)+Acc+\sum_{i=2}^{n} C_i$$
$$\overset{*}{\Longrightarrow} \ldots \overset{*}{\Longrightarrow} hg(\sigma_n,\alpha_n,\beta_n,w)+Acc \underset{accept}{\Longrightarrow} Y_n \underset{cut}{\Longrightarrow} Y_n' + sg(w)_\mu \underset{m_0}{\Longrightarrow} sg(w)_\mu.$$

Consequently, due to Lemma 1[4], this implies $c_0 \vdash^*_{TM} (q_{accept},\alpha_n,\beta_n)$ wrt input w. Hence, $w \in L(TM)$. □

6 Conclusion

In this paper, we have continued the research on context-dependent fusion grammars by transforming Turing machines into this type of hypergraph grammars. This reduction gives us interesting insights into these grammars because the transformation proves that context-dependent fusion grammars are another universal computing model and can generate all recursive enumerable string languages (up to representation). Note that a similar construction also works for computation of partial functions. In this case the connected components $tape_{start}$,$tape_{end}$ and $tape_x$ are replaced by a tape graph representing the Turing machines input $x_1\ldots x_n \in \Sigma^*$, where the start is attached to some $tape$-hyperedge. However, further research is needed including the following open question. In the literature, one encounters model transformations from several modeling approaches into Turing machines. Now they can be extended to

[4]The cases where $fr(\triangleright)$,$fr(\triangleleft)$,*shrink* are applied do not occur due to the assumption that all context-free fusion rules are applied first.

context-dependent fusion grammars. Does this provide interesting insights? Are only positive or only negative context conditions powerful enough to cover Turing machines? How does a natural transformation of context-dependent fusion grammars into splicing/fusion grammars or the other way round look like?

References

[1] Hartmut Ehrig, Annegret Habel & Hans-Jörg Kreowski (1992): *Introduction to graph grammars with applications to semantic networks*. *Computers and Mathematics with Applications* 23(6–9), pp. 557–572, doi:10.1016/0898-1221(92)90124-Z.

[2] Annegret Habel & Detlef Plump (2001): *Computational Completeness of Programming Languages Based on Graph Transformation*. In Furio Honsell & Marino Miculan, editors: *Proc. Foundations of Software Science and Computation Structures (FOSSACS 2001)*, *Lecture Notes in Computer Science* 2030, Springer, pp. 230–245, doi:10.1007/3-540-45315-6_15.

[3] John E. Hopcroft, Rajeev Motwani & Jeffrey D. Ullman (2003): *Introduction to automata theory, languages, and computation - international edition (2. ed)*. Addison-Wesley.

[4] Juraj Hromkovic (2004): *Theoretical computer science*. Texts in theoretical computer science, EATCS series, Springer, Berlin, doi:10.1007/978-3-662-05269-3.

[5] Hans-Jörg Kreowski (1993): *Translations into the Graph Grammar Machine*. In Ronan Sleep, Rinus Plasmeijer & Marko van Eekelen, editors: *Term Graph Rewriting: Theory and Practice*, chapter 13, John Wiley, New York, pp. 171–183.

[6] Hans-Jörg Kreowski, Renate Klempien-Hinrichs & Sabine Kuske (2006): *Some Essentials of Graph Transformation*. In Zoltán Ésik, Carlos Martín-Vide & Victor Mitrana, editors: *Recent Advances in Formal Languages and Applications*, *Studies in Computational Intelligence* 25, Springer, pp. 229–254, doi:10.1007/978-3-540-33461-3_9.

[7] Hans-Jörg Kreowski, Sabine Kuske & Aaron Lye (2017): *Fusion Grammars: A Novel Approach to the Generation of Graph Languages*. In Detlef Plump & Juan de Lara, editors: *Proc. 10th International Conference on Graph Transformation (ICGT 2017)*, *Lecture Notes in Computer Science* 10373, Springer, pp. 90–105, doi:10.1007/978-3-319-61470-0.

[8] Hans-Jörg Kreowski, Sabine Kuske & Aaron Lye (2018): *Splicing/Fusion Grammars and Their Relation to Hypergraph Grammars*. In Leen Lambers & Jens H. Weber, editors: *Proc. 11th International Conference on Graph Transformation (ICGT 2018)*, *LNCS* 10887, Springer, pp. 3–19, doi:10.1007/978-3-319-92991-0_1.

[9] Hans-Jörg Kreowski, Sabine Kuske & Aaron Lye (2019): *Transformation of Petri Nets into Context-Dependent Fusion Grammars*. In Carlos Martín-Vide, Alexander Okhotin & Dana Shapira, editors: *Proc. 13th International Conference on Language and Automata Theory and Applications (LATA 2019)*, *LNCS* 11417, Springer, pp. 246–258, doi:10.1007/978-3-030-13435-8_18.

[10] Aaron Lye (2018): *Decidability and Complexity of the Membership and Emptiness Problem of Fusion Grammars*. In: *Pre-Proc. 9th International Workshop on Graph Computation Models, (GCM 2018).*

[11] Alan M. Turing (1936): *On Computable Numbers, with an Application to the Entscheidungsproblem*. *Proceedings of the London Mathematical Society* 2(42), pp. 230–265, doi:10.1112/plms/s2-42.1.230. A correction was published in Proceedings of the London Mathematical Society. 2 (1937). 43 (6): 5446.

[12] Tadahiro Uesu (1978): *A system of graph grammars which generates all recursively enumerable sets of labelled graphs*. *Tsukuba journal of mathematics* 2, pp. 11–26, doi:10.21099/tkbjm/1496158502.