(Re)introducing Regular Graph Languages

Sorcha Gilroy
University of Edinburgh
s.gilroy@sms.ed.ac.uk

Adam Lopez
University of Edinburgh
alopez@inf.ed.ac.uk

Sebastian Maneth
Universität Bremen
smaneth@uni-bremen.de

Pijus Simonaitis
Ecole Normale Supérieure de Lyon
pijus.simonaitis@ens-lyon.fr

Abstract

Distributions over strings and trees can be represented by probabilistic regular languages, which characterise many models in natural language processing. Recently, several datasets have become available which represent natural language phenomena as graphs, so it is natural to ask whether there is an equivalent of probabilistic regular languages for graphs. This paper presents **regular graph languages**, a formalism due to Courcelle (1991) that has not previously been studied in natural language processing. RGL is crucially a subfamily of both Hyperedge Replacement Languages (HRL), which can be made probabilistic; and Monadic Second Order Languages (MSOL), which are closed under intersection. We give an accessible introduction to Courcelle's proof that RGLs are in MSOL, providing clues about how RGL may relate to other recently introduced graph grammar formalisms.

1 Introduction

NLP systems for machine translation, summarisation, paraphrasing, and other problems often fail to preserve the compositional semantics of sentences and documents because they model language as bags of words, or at best syntactic trees. To preserve semantics, they must model semantics. In pursuit of this goal, several datasets have been produced which pair natural language with compositional semantic representations in the form of directed acyclic graphs (DAGs), including the Abstract Meaning Represenation Bank (AMR; Banarescu et al. 2013), the Prague Czech-English Dependency Treebank (Hajič et al., 2012), Deepbank (Flickinger et al., 2012), and the Universal Conceptual Cognitive Annotation (Abend and Rappoport, 2013). To make use of this data, we require probabilistic models of graphs.

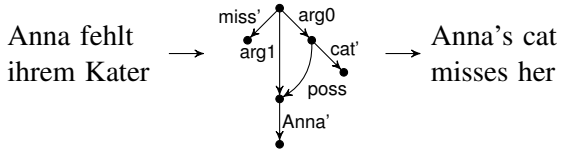

Figure 1: Semantic machine translation using AMR (Jones et al., 2012). The edge labels identify 'cat' as the subject of the verb 'miss', 'Anna' as the object of 'miss' and 'Anna' as the possessor of 'cat'.

Consider how we might use compositional semantic representations in machine translation (Figure 1). We first parse a source sentence to its semantic representation, and then generate a target sentence from this representation. To do this practically, we must be able to compose a string-to-graph model with a graph-to-string model, and we must be able to compute the probability of this composition. To compose the models, we need to be able to compute the intersection of the graph domains of each model. Hence, we must be able to define probability distributions over the graph domains and efficiently compute their intersection.

For NLP problems in which data is in the form of strings and trees, such distributions can be represented by finite automata (Mohri et al., 2008; Allauzen et al., 2014), which are closed under intersection and can be made probabilistic. It is therefore natural to ask whether there is a family of graph languages with similar properties to finite automata. Recent work in NLP has focused primarily on two families of graph languages: **hyperedge replacement languages** (HRL; Drewes et al. 1997), a context-free graph rewriting formalism that has been studied in an NLP context by several researchers (Chiang et al., 2013; Peng et al., 2015; Bauer and Rambow, 2016); and **DAG automata languages**, (DAGAL; Kamimura and Slutzki 1981), studied by (Quernheim and Knight,

Proceedings of the 15th Meeting on the Mathematics of Language, pages 100–113,
London, UK, July 13–14, 2017.

2012). (Thomas, 1991) showed that the latter are a subfamily of the **monadic second order languages** (MSOL), which are of special interest to us, since, when restricted to strings or trees, they exactly characterise the **recognisable**—or regular—languages of each (Büchi, 1960; Büchi and Elgot, 1958; Trakhtenbrot, 1961).

The HRL and MSOL families are incomparable: that is, the context-free graph languages do not contain the recognisable graph languages, as is the case in languages of strings and trees (Courcelle, 1990). So, while each formalism has appealing characteristics, neither appear adequate for the problem outlined above: HRLs can be made probabilistic, but they are not closed under intersection; and while DAGAL and MSOL are closed under intersection, it is unclear how to make them probabilistic (Quernheim and Knight, 2012).[1]

This situation suggests that we might want a family of languages that is a subfamily of both HRL and MSOL. Courcelle (1991) defines all such languages as the family of **strongly context-free languages** (SCFL).[2,3] Unfortunately, SCFLs are defined non-constructively, but Courcelle (1991) exhibits a constructively-defined subfamily: **Regular Graph Languages** (RGL), defined as a restricted form of HRL, which Courcelle demonstrates is also in MSOL.

Recently, two new graph grammar formalisms have been defined which are also restricted forms of HRL: Tree-like Grammars (TLG; Matheja et al. 2015) and Restricted DAG Grammars (RDG; Björklund et al. 2016). TLGs are claimed to be in SCFL, but the relationship of RDG to SCFL is unknown. The grammar restrictions of TLGs, RDGs and RGGs are incomparable, but they share important characteristics, which we discuss in §5.

This paper provides an accessible proof that RGL are a subfamily of MSOL, since only a sketch is provided in Courcelle (1991). Our aim in studying the proof is to gain insights into the relationship of RGL, TLG, and RDG, which might enable us to define more general classes of graph languages that are also within SCFL. Our discussion emphasises points at which Courcelle's proof relies on particular restrictions of RGL, and is intended to highlight the places where relaxations of these restrictions may be possible.

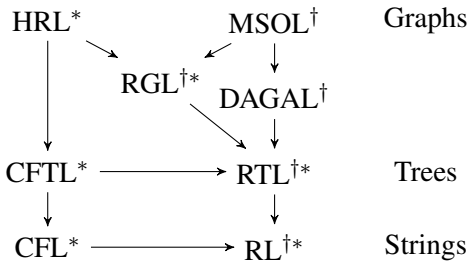

Figure 2: Containment relationships for families of regular and context-free string and tree languages, hyperedge replacement languages (HRL), monadic second order definable graph languages (MSOL), directed acyclic graph automata languages (DAGAL), and the regular graph languages (RGL). $*$ indicates that the family of languages is probabilistic and † indicates that the family of languages is intersectible.

Figure 2 summarises the relationship of RGL to other formalisms and their properties. The proof of each Lemma, Proposition and Theorem in this paper that does not appear here is provided in full in the supplementary materials.[4]

2 Monadic Second-Order Logic

The regular string and tree languages precisely coincide with the monadic second-order logic (MSO) definable sets of strings and trees, respectively (Büchi, 1960; Büchi and Elgot, 1958; Trakhtenbrot, 1961), so it is natural to look at MSO over graphs.

We use the following notation. If n is an integer, $[n]$ denotes the set $\{1, \ldots, n\}$. If A is a set, $s \in A^*$ denotes that s is a sequence of arbitrary length, each element of which is in A. We denote by $|s|$ the length of s. A **ranked alphabet** is an alphabet A paired with an arity function rank: $A \to \mathbb{N}$.

Definition 1. *A **hypergraph** over a ranked alphabet Γ is a tuple $G = (V_G, E_G, att_G, lab_G, ext_G)$ where V_G is a finite set of nodes; E_G is a finite set of edges (distinct from V_G); $att_G : E_G \to V_G^*$ maps each edge to a sequence of nodes; $lab_G : E_G \to \Gamma$ maps each edge to a label such that $|att_G(e)| = rank(lab_G(e))$; and ext_G is an ordered subset of V_G called the **external nodes** of G.*

[1] *Semiring-weighted* MSOLs have been defined, where weights may be in the tropical semiring (Droste and Gastin, 2005). However, for the weights to define a probability distribution, they must meet the stronger condition that the sum of multiplied weights over all definable objects is one. This does not appear to have been demonstrated for DAGAL, which violate the sufficient conditions that (Booth and Thompson, 1973) give for probabilistic languages. We suspect that there are DAGAL (hence MSOL) for which it is not possible.

[2] Courcelle's definition of strongly context-free is unrelated to use of this term in NLP.

[3] The equality of SCFL and MSOL ∩ HRL was recently proved by (Bojanczyk and Pilipczuk, 2016).

[4] goo.gl/Z5L2gP

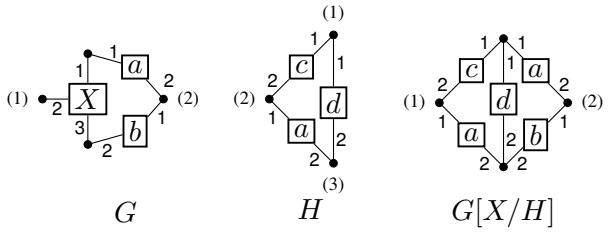

Figure 3: The replacement of the X-labeled edge in G by the graph H.

We assume that both the elements of ext_G and the elements of $\text{att}_G(e)$ for each edge e are pairwise distinct. An edge e is attached to its nodes by **tentacles**, each labeled by an integer indicating the node's position in $\text{att}_G(e) = (v_1, \ldots, v_k)$. The tentacle from e to v_i has label i, so the tentacle labels lie in the set $[k]$ where $k = \text{rank}(e)$. To express that a node v is attached to the i-th tentacle of an edge e we say $\text{vert}(e, i) = v$. The nodes in ext_G are labeled by their position in ext_G. In figures, the i-th external node is labeled (i). The **rank** of an edge e is k if $\text{att}(e) = (v_1, \ldots, v_k)$ (or equivalently, rank(lab(e)) = k). The **rank** of a hypergraph G is $|\text{ext}_G|$. An **induced subgraph** of a hypergraph G by edges $E' \subseteq E_G$ is the subgraph of G formed by including all edges in E' and their endpoints. Define $\text{HG}_{\Sigma,\Gamma}$ to be the set of all hypergraphs with node labels in Σ and edge labels in Γ (the hypergraphs as defined here have no node labels so are in $\text{HG}_{\emptyset,\Gamma}$).

Example 1. Hypergraph G in Figure 3 has four nodes (shown as black dots) and three hyperedges labeled a, b, and X (shown boxed). The bracketed numbers (1) and (2) denote its external nodes and the numbers between edges and the nodes are tentacle labels. Call the top node v_1 and, proceeding clockwise, call the other nodes v_2, v_3, and v_4. Call its edges e_1, e_2 and e_3. Its definition would state:

$$\begin{aligned} att_G(e_1) &= (v_1, v_2) & lab_G(e_1) = a \\ att_G(e_2) &= (v_2, v_3) & lab_G(e_2) = b \\ att_G(e_3) &= (v_1, v_4, v_3) & lab_G(e_3) = X \\ ext_G &= (v_4, v_2). \end{aligned}$$

MSO on graphs quantifies over nodes, sets of nodes, edges, and sets of edges.[5] The atomic formulas are $x \in X$, $x = y$, $\text{lab}_\gamma(x)$, and $\text{vert}(x, i) = y$. We construct MSO sentences using the atomic formulas, connectives $\wedge, \vee, \neg, \Rightarrow$, and quantifiers $\exists, \forall$. We allow $\text{vert}(x, i) = y$ to hold only when x is an edge and y is a node. In the case of edge-labelled graphs, the x in $\text{lab}_\gamma(x)$ must be an edge. We define the formula $\text{edg}(x, y_1, \ldots, y_k) : \text{vert}(x, 1) = y_1 \wedge \ldots \text{vert}(x, k) = y_k \wedge \bigwedge_{k'>k} \forall y \neg \text{vert}(x, k') = y$ which expresses $\text{att}(x) = (y_1, \ldots, y_k)$.

We can write down an MSO formula to express that sets $X_1, \ldots, X_n$ partition the domain. $\text{PART}(X_1, \ldots, X_n)$:

$$\begin{aligned} &\forall x[x \in X_1 \cup \cdots \cup X_n \wedge \neg x \in X_1 \cap X_2 \\ &\wedge \neg x \in X_1 \cap X_3 \wedge \cdots \wedge \neg x \in X_{n-1} \cap X_n] \end{aligned} \quad (1)$$

We use ! to denote unique existential quantification. For any formula R:

$\exists! x R(x) : \exists x R(x) \wedge \forall y R(y) \rightarrow x = y.$

We can define an MSO statement expressing that the graph is a string by defining an edge labelled graph where the edges have rank 2, there is exactly one node with no incoming edge, there is exactly one node with no outgoing edge, and every node has at most one incoming edge and at most one outgoing edge:

$$\begin{aligned} &\text{STRING} : \forall y \exists! x_1 \exists! x_2 \text{edg}(y, x_1, x_2) \wedge \\ &\exists! x_1 \forall y \neg \text{vert}(y, 2) = x_1 \wedge \exists! x_2 \forall y \neg \text{vert}(y, 1) = x_2 \\ &\wedge \forall x \neg x = x_1 \Rightarrow \exists! y \exists! x' \text{edg}(y, x', x) \\ &\wedge \forall x \neg x = x_2 \Rightarrow \exists! y \exists! x' \text{edg}(y, x, x') \end{aligned}$$

Let $\text{First}(x)$ denote that x has no incoming edges and $\text{Last}(x)$ denote that x has no outgoing edges.

Example 2. Let A be the automaton in Figure 4. The corresponding MSO quantifies over a subset X_i for each state q_i in the automaton. The subsets partition the nodes of the string graph to simulate a run of the automaton.

Finally, we encode each transition of the form (q_i, a, q_j) as $x \in X_i \wedge \exists y \exists x' \text{edg}(y, x, x') \wedge \text{lab}_a(y) \Rightarrow x' \in X_j$.

From A, we construct the formula aut_A:

$$\begin{aligned} &\text{aut}_A : \text{STRING} \wedge \exists X_0 \exists X_1 \text{PART}(X_0, X_1) \\ &\wedge \forall x \text{First}(x) \Rightarrow x \in X_0 \\ &\wedge \forall x \text{Last}(x) \Rightarrow x \in X_1 \\ &\wedge \forall x \in X_0 \wedge \exists y \exists x' \text{edg}(y, x, x') \wedge \text{lab}_a(y) \Rightarrow x' \in X_1 \\ &\wedge \forall x \in X_1 \wedge \exists y \exists x' \text{edg}(y, x, x') \wedge \text{lab}_a(y) \Rightarrow x' \in X_1 \\ &\wedge \forall x \in X_1 \wedge \exists y \exists x' \text{edg}(y, x, x') \wedge \text{lab}_b(y) \Rightarrow x' \in X_0 \end{aligned}$$

For a graph G and an MSO statement ϕ we say that $G \models \phi$ (or G satisfies ϕ) when there is an assignment of variables of ϕ to nodes and edges of

[5] Formally, this is called MS_2 (Courcelle and Engelfriet, 2011); MS_1 only quantifies over nodes and sets of nodes.

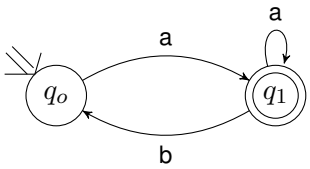

Figure 4: The finite automaton A.

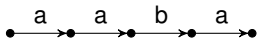

Figure 5: The graph representing the string $aaba$.

G that makes ϕ true.

Example 3. The string graph $G = aaba$ as shown in Figure 5 can be produced by automaton A. The letters are edge labels and call its nodes from left to right v_0, v_1, v_2, v_3, and v_4. If we let $X_0 = \{v_0, v_3\}$ and $X_1 = \{v_1, v_2, v_4\}$ then $G \models \text{aut}_A$.

Let $\text{aut}'_A(X_0, X_1)$ be the MSO formula identical to aut_A with $\exists X_0 \exists X_1$ removed from the beginning of the formula. X_0 and X_1 are free variables of aut'_A, and we refer to the set of free variables of a formula as its **parameters**. Given a graph G and a formula $\phi(\mathcal{W})$ with parameters $\mathcal{W}$, let α be function from $\mathcal{W}$ to subsets of nodes and edges in G. Then we say that $(G, \alpha) \models \phi(\mathcal{W})$ if G and α satisfy $\phi(\mathcal{W})$. We call α a **parameter assignment**. The MSO interpretation of an automaton is satisfied if we can find a parameter assignment that simulates a run of the automaton—more precisely, $G \models \text{aut}_A$ if $(G, \alpha) \models \text{aut}'_A(X_0, X_1)$. In general, there may be more than one such α.

Example 4. Let $\mathcal{W} = \{X_0, X_1\}$ be parameters. If $\alpha(X_0) = \{v_0, v_3\}$ and $\alpha(X_1) = \{v_1, v_2, v_4\}$ then $(G, \alpha) \models \text{aut}'_A(\mathcal{W})$.

We can use an MSO statement ϕ to define a language, $L(\phi) = \{G \mid G \models \phi\}$, and we call the family of languages definable this way as **MSOL**. We define the intersection of two languages $L(\phi_1) \cap L(\phi_2) = \{G \mid G \models \phi_1 \wedge \phi_2\}$. This clearly shows that MSO languages are closed under intersection.

2.1 MSO Transductions

One way to show that a language is MSO definable is to use the backwards translation theorem (Courcelle and Engelfriet, 2011), which depends on **MSO transductions (MSOT)**, a generalisation of finite-state string and tree transductions. The theorem is a generalisation to graphs of the fact that regular string and tree languages are closed under inverse finite-state transductions (Hopcroft and Ullman, 1979; **?**).

Theorem 1 (Backwards Translation Theorem)**.** *If L is an MSO definable graph language and f is an MSO graph transduction then $f^{-1}(L)$ is effectively MSO definable.*

Definition 2. *An MSO transducer $\tau : HG_{\Sigma,\Gamma} \to HG_{\Sigma',\Gamma'}$ is $\tau = \langle \rho(\mathcal{W}), \delta(x, \mathcal{W}), (\theta_r(x_1, \ldots, x_{N(r)}, \mathcal{W}))_{r \in \mathcal{R}} \rangle$. $\mathcal{W}$ is a set of parameters; $\rho(\mathcal{W})$ is a **precondition** which input graphs must satisfy; $\delta(x, \mathcal{W})$ is a **domain formula** defining the output domain (i.e. nodes); and $\theta_r(x_1, \ldots, x_{N(r)}, \mathcal{W})$ is a **relation formula** defining relationships between the elements in the output domain (i.e. edges).*[6]

The role of parameters here is to allow nondeterminism. Given a graph G and a parameter assignment α from $\mathcal{W}$ to $V_G \cup E_G$ such that $(G, \alpha) \models \rho(\mathcal{W})$, we define the output of the transducer $\tau(G, \alpha) = (D, R)$ such that $D = \{x \in G \mid (G, x, \alpha) \models \delta(x, \mathcal{W})\}$ and $R = \{\theta_r(x_1, \ldots, x_{N(r)}) \mid (G, x_1, \ldots, x_{N(r)}, \alpha) \models \theta_r(x_1, \ldots, x_{N(r)}, \mathcal{W}), r \in \mathcal{R}\}$. Define $\tau(G) = \{\tau(G, \alpha) \mid (G, \alpha) \models \rho(\mathcal{W})\}$ and for a language L, $\tau(L) = \{\tau(G) \mid G \in L\}$.

3 Hyperedge Replacement Grammars

If f is a function and S is a set, $f|_S$ is the restriction of f to domain elements in S. If f, g are functions, $f \circ g$ is their composition.

Definition 3. *Let G be a hypergraph with an edge e of rank k and let H be a hypergraph also of rank k disjoint from G. The **replacement** of e by H is the graph $G' = G[e/H]$. Let $V_{G'} = (V_G \cup V_H) - ext_H$, $E_{G'} = (E_G \cup E_H) - \{e\}$. Let $ext_H = (v_1, \ldots, v_k)$, $att_G(e) = (u_1, \ldots, u_k)$ and let $f : (V_G \cup V_H) \to V_{G'}$ replace v_i by u_i for $i \in [k]$ and be the identity otherwise. The extension of f to $(V_G \cup V_H)^*$ is also denoted f. Let $E = E_G - \{e\}$, then $att_{G'} = att_G|_E \cup (f \circ att_H)$, $lab_{G'} = lab_G|_E \cup lab_H$.*

Example 5. Replacement is shown in Figure 3. We denote the replacement as $G[X/H]$ since the edge is unambiguous given its label.

Definition 4. *A **hyperedge replacement grammar** (HRG) $G = (N, T, P, S)$ consists of disjoint ranked alphabets N and T of nonterminals and*

[6] Technically, this is a non-copying MSO transducer. In general, MSO transductions can define multiple copies of each element of the input domain.

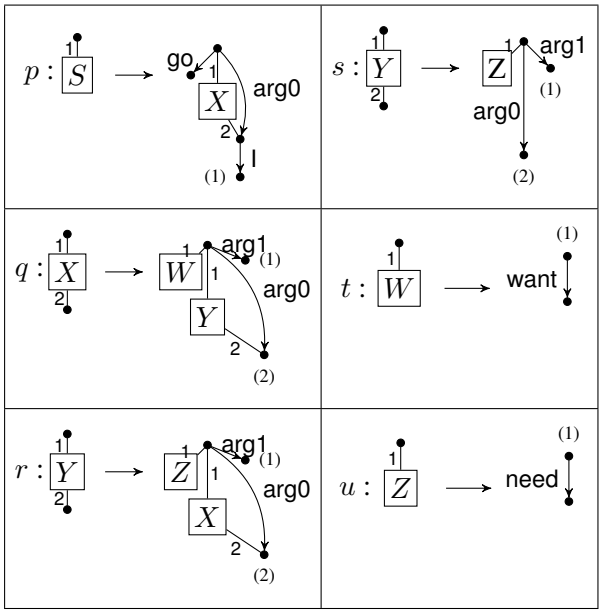

Table 1: Productions of a HRG. The labels p, q, r, s, t, and u label the productions so that we can refer to them in the text. Note that Y can be rewritten either via production r or s.

terminals, a finite set of productions P, and a start symbol $S \in N$. Every production in P is of the form $X \to H$ where $X \in N$ is of rank k and H is a hypergraph of rank k over N and T.

A HRG $\mathcal{G}$ produces graphs in $\text{HG}_{\emptyset,T_\mathcal{G}}$. In each example, we only show terminal edges of rank 2, and depict them as directed edges where the direction is determined by the tentacle labels: tentacle 1 attaches to the source and 2 attaches to the target (Table 1). For each production $p : X \to G$, we use $L(p)$ to refer to its left-hand side (X) and $R(p)$ to refer to its right-hand side (G). An edge is a **terminal edge** if its label is terminal and a **nonterminal edge** if its label is nonterminal. A graph is **terminal** if all of its edges are labeled with terminal symbols. The **terminal subgraph** of a graph is the subgraph induced by its terminal edges. Let $\text{NT}(p) = \{e_1, \ldots, e_n\}$ be an enumeration of the nonterminal edges in $R(p)$, let $|\text{NT}(p)|$ be the number of nonterminal edges in $R(p)$ and let $|\text{NT}(P)| = \max_{p \in P} |\text{NT}(p)|$.

Given a HRG $\mathcal{G}$, we say that graph G **derives** graph G', denoted $G \to G'$, iff there is an edge $e \in E_G$ and a nonterminal $X \in N$ such that $\text{lab}_G(e) = X$ and $G' = G[e/H]$, where $X \to H$ is in P. We extend the idea of derivation to its transitive closure $G \to^* G'$. For every $X \in N$ we also use X to denote the connected graph consisting of a single edge e with $\text{lab}(e) = X$ and nodes $(v_1, \ldots, v_{\text{rank}(X)})$ such that $\text{att}(e) = (v_1, \ldots, v_{\text{rank}(X)})$, and we define the language $L_X(\mathcal{G}) = \{G \mid X \to^* G, G \text{ is terminal}\}$. The **language of** $\mathcal{G}$ is then $L(\mathcal{G}) = L_S(\mathcal{G})$. We call the family of languages that can be produced by any HRG the **hyperedge replacement languages (HRL)**.

3.1 HRL and MSOT

Since HRGs are context-free, for each HRG $\mathcal{G}$, there is an underlying regular tree grammar $\mathcal{T}_\mathcal{G}$ defining the derivation trees of the graphs in $L(\mathcal{G})$. Each $\mathbf{T} \in \mathcal{T}_\mathcal{G}$ has node labels in $P_\mathcal{G}$ and edge labels in $|\text{NT}(P)|$. If a node has label p and $R(p)$ has n nonterminals $X_1, \ldots, X_n$ then for each $i \in [n]$, there is an i labelled edge from p to a node labelled q where $L(q) = X_i$. The label of the root of $\mathbf{T}$ must be p for some p with $L(p) = S$. Let $\text{VAL} : L(\mathcal{T}_\mathcal{G}) \to L(\mathcal{G})$ be a mapping from derivation trees to graphs so that $G = \text{VAL}(\mathbf{T})$ iff $\mathbf{T}$ is a derivation tree of G. Since HRGs can be ambiguous, this mapping is not injective. (Courcelle, 1991) shows that VAL is an MSO transduction.[7] This does not imply that HRLs are MSOL, since in general MSOL is not closed under MSOT. Hence an MSOT representing the inverse of VAL may not exist for an arbitrary HRG, but we later discuss a subfamily for which it does (§4), allowing us to apply Theorem 1.

To distinguish between elements of a graph and its derivation tree, we denote a grammar by $\mathcal{G}$, graph by G, derivation tree by $\mathbf{T}$, derivation tree node by $\mathbf{v}$, edges and nodes in productions are written with a bar ($\bar{v}$) and nodes and edges in G are unmarked (x).

The transduction VAL preserves the terminal subgraph of every production used in a derivation and fuses nodes from different productions together in the output graph. Node fusion is determined by an equivalence relation $\sim$ generated by a relation $\sim_0$. Let $\text{NT}(p) = (e_1, \ldots, e_n)$ the nonterminal edges of $R(p)$, let $\text{NT}_i(p) = e_i$, and let $\text{ext}_G(i)$ be the ith external node of G.

Definition 5. *Let $\mathcal{G}$ be a HRG and $\mathbf{T}$ be a derivation tree of $\mathcal{G}$, so that $G = \text{VAL}(\mathbf{T})$. Define a binary relation $\sim_0$ on pairs $(\bar{x}, \mathbf{v})$ where $\bar{x}$ is a node in $R(p)$ for some $p \in P$ and $\mathbf{v}$ is a node of $\mathbf{T}$ with label p. Then $(\bar{x}, \mathbf{v}) \sim_0 (\bar{y}, \mathbf{v}')$ iff:*

1. $\mathbf{v}, \mathbf{v}'$ are nodes in $\mathbf{T}$ and $\mathbf{v}'$ is the ith child of $\mathbf{v}$ in $\mathbf{T}$.

[7] It can also be viewed in the related framework of interpreted regular tree grammars (Koller and Kuhlmann, 2011).

2. $p = lab_{\mathbf{T}}(\mathbf{v})$, $p' = lab_{\mathbf{T}}(\mathbf{v}')$.
3. $\bar{x}$ *is the* j*th node of* $NT_i(p)$, $\bar{y} = ext_{R(p')}(j)$.

We define $\sim$ as the reflexive, symmetric, transitive closure of $\sim_0$.

The mapping VAL translates derivation trees to graphs in two steps. First, the terminal subgraph of every instance of every production used in the derivation tree is produced in the output. Then, all equivalent nodes under $\sim$ are fused. (Courcelle, 1991) shows that each step is a MSOT; their composition is also a MSOT.

Example 6. Figure 6 illustrates how VAL maps a derivation tree to a graph.

The mapping VAL can be defined in terms of two finer-grained mappings. Let $E_P = \cup_{p \in P} E_{R(p)}$ and $V_P = \cup_{p \in P} V_{R(p)}$. Then $h_e : E_P \times V_{\mathbf{T}} \to E_G$ maps a pair $(\bar{e}, \mathbf{v})$ to its image e in the graph, where $\bar{e}$ is a terminal edge in p and $\text{lab}(\mathbf{v}) = p$. This mapping is one-to-one since edges cannot be fused. $h_v : V_P \times V_{\mathbf{T}} \to V_G$ maps a pair $(\bar{x}, \mathbf{v})$ to its image v, where $\bar{x}$ is a node in p and $\text{lab}(\mathbf{v}) = p$. It is not one-to-one since nodes can be fused.

Lemma 1. *Let* $\mathcal{G}$ *be a HRG, and let* G *be a graph in* $L(\mathcal{G})$ *with derivation tree* $\mathbf{T}$. *If* $\bar{x}$ *and* $\bar{x}'$ *are nodes such that* $h_v(\bar{x}, \mathbf{v}) = h_v(\bar{x}', \mathbf{v}')$ *with* $\mathbf{v} \neq \mathbf{v}'$ *and, if* $\bar{x}$ *is internal in* $R(p)$ *for* $p = lab_{\mathbf{T}}(\mathbf{v})$, *then* $\bar{x}'$ *is an external node of* $R(p')$ *for* $p' = lab_{\mathbf{T}}(\mathbf{v}')$ *and* $\mathbf{v}$ *is an ancestor of* $\mathbf{v}'$ *in* $\mathbf{T}$.

Consequently, if $h_v(\bar{x}, \mathbf{v}) = h_v(\bar{x}', \mathbf{v}')$ then $\bar{x}$ and $\bar{x}'$ cannot both be internal.

4 Regular Graph Grammars

A regular graph grammar (RGG; Courcelle 1991) is a restricted form of HRG. To explain the restrictions, we first require some definitions.

Definition 6. *Given a graph* G, *a* ***path*** *in* G *from a node* v *to a node* v' *is a sequence*

$$(v_0, i_1, e_1, j_1, v_1)(v_1, i_2, e_2, j_2, v_2) \ldots (v_{k-1}, i_k, e_k, j_k, v_k)$$

such that $vert(e_r, i_r) = v_{r-1}$ *and* $vert(e_r, j_r) = v_r$ *for each* $r \in [k]$, $v_0 = v$, *and* $v_k = v'$. *The length of this path is* k.

A path is **terminal** if every edge in the path is terminal. A path is **internal** if each v_i is internal for $1 \leq i \leq k-1$. The endpoints v_0 and v_k of an internal path can be external.

Definition 7. *A HRG* $\mathcal{G}$ *is a* ***Regular Graph Grammar*** *if each nonterminal in* N *has rank at least one and for each* $p \in P_{\mathcal{G}}$ *the following hold:*

(C1) $R(p)$ *has at least one edge. Either it is a single terminal edge, all nodes of which are external, or each of its edges has at least one internal node.*

(C2) Every pair of nodes in $R(p)$ *is connected by a terminal and internal path.*

RGLs are HRLs by definition; we will prove that they are also MSOLs by constructing the inverse of VAL, a transducer from RGL graphs to their derivation trees. Since the derivation trees are MSO definable, RGLs must also be MSO definable by Theorem 1. The construction requires a unique **anchor** element (a node or edge) for each production in the grammar. Given an input graph, the transducer first guesses—via parameter assignment—the preimage of each edge and the set of elements whose preimages are anchors. It then checks whether the guess satisfies constraints that must be true for every derived graph:

1. It must be possible to partition the graph into a set of edge-disjoint connected subgraphs, each isomorphic to the terminal subgraph of some production.
2. For each node that is in two such subgraphs, the node must be the image of two nodes in the productions that are allowed to be fused under the grammar.

If these constraints are satisfied, the transducer outputs each guessed anchor and an edge between anchors that it identifies to be in a parent-child relationship.

Every valid parameter assignment corresponds to a different output from the transducer, and we will show that all derivation trees for any input graph in the grammar lie in this output set.

Theorem 2. *RGL* $\subseteq$ *MSOL.*

The proof of each Lemma and Proposition in this section either appears here or in the supplementary materials. The proof of Theorem 2 is provided in §4.2.1.

4.1 Anchors and Parameters

There are two types of productions in RGGs: those with a single terminal edge, all nodes of which are external; and those where each edge has an internal node. We call the former **ext-productions** and the latter **int-productions**. For each int-production, we arbitrarily choose one of its internal nodes to be its anchor. For each ext-production, we choose its single terminal edge to be the anchor. By Lemma 1, this choice ensures

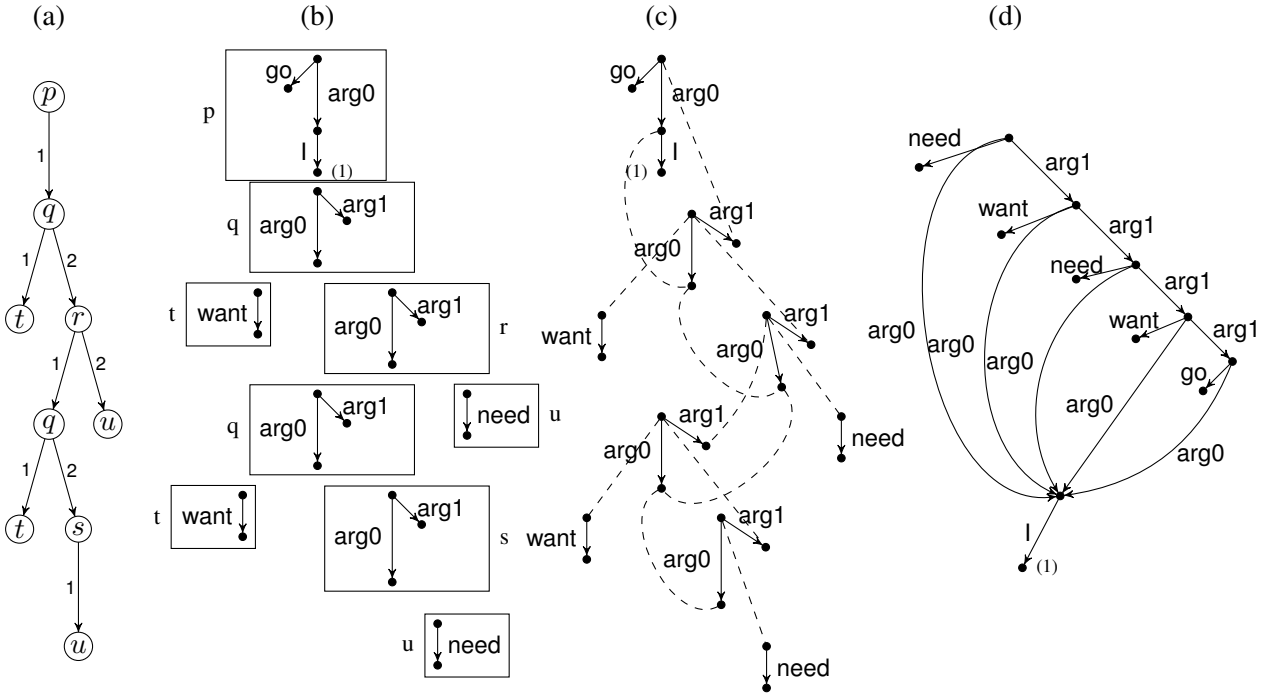

Figure 6: A derivation tree (a), the terminal subgraphs of every copy of every production in the derivation tree (b), the relation $\sim$ illustrated with dashed lines (c) and the resulting graph (d).

that a pair of anchors cannot be fused, so the set of anchors in any derived graph is guaranteed to be in one-to-one correspondence with the nodes of its derivation tree.

We define two sets of parameters: $\mathcal{E}$ and $\mathcal{C}$, where $\mathcal{E}$ guesses preimages of edges, and $\mathcal{C}$ guesses anchors (which may be either nodes or edges). To define $\mathcal{E}$ precisely, we require some notation. Let $\mathcal{G}$ be an RGG, and for each $p \in P$, let $\mathrm{T}(p) = \{\bar{f}_{p,1}, \ldots, \bar{f}_{p,|\mathrm{T}(p)|}\}$ enumerate the terminal edges of $R(p)$ and let $\gamma_{p,j}$ be the label of $\bar{f}_{p,j}$ for each $p \in P$ and $j \in [|\mathrm{T}(p)|]$. Let $|\mathrm{NT}(p)|$ be the number of nonterminal edges in p and let $|\mathrm{NT}(P)| = \max_{p \in P} |\mathrm{NT}(p)|$. Given a node $\mathbf{v}$ in a derivation tree $\mathbf{T}$, we say that $\mathbf{v}$ is an i-child if it is the ith child of some other node in $\mathbf{T}$. By convention, the root node is the only 0-child.

Let G be in $L(\mathcal{G})$ and let $\mathbf{T}$ be a derivation tree of G. For each $i \in [0, |\mathrm{NT}(P)|]$, $p \in P$ and $j \in [|T(p)|]$, we define a parameter $E_{i,p,j}$:

$$E_{i,p,j} = \{e \in E_G \mid h_e(\bar{f}_{p,j}, \mathbf{v}) \text{ and } \mathbf{v} \text{ is an } i\text{-child.}\}$$

Let $\mathcal{E} = \{E_{i,p,j}\}$ for $i \in [0, |\mathrm{NT}(P)|]$, $p \in P$ and $j \in [|\mathrm{T}(p)|]$.

For each $i \in [|\mathrm{NT}(P)|]$ and $p \in P$, define

$$C_{i,p} = \{h(\bar{c}_p, \mathbf{v}) \mid p = \mathrm{lab}_{\mathbf{T}}(\mathbf{v}), \mathbf{v} \text{ is an } i\text{-child.}\}$$

Where $h = h_e \cup e_v$ since $\bar{c}_p$ can either be an edge or a node. Let $\mathcal{C} = \cup_{i,p} C_{i,p}$.

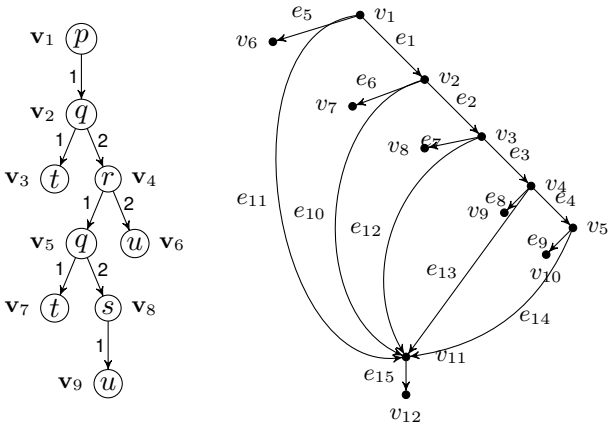

Figure 7: The derivation tree from Figure 6 (a) and the graph from Figure 6 (d) with variable names for the nodes and edges.

Let $\mathcal{W} = \mathcal{E} \cup \mathcal{C}$.

Example 7. Table 2 shows the productions of Table 1 with labels on each node and edge. Figure 7 shows the derivation tree and graph from Figure 6 with variable names added. We use these variable names to refer to specific nodes and edges in the text. For example, $h_v(\bar{c}_s, \mathbf{v}_8) = v_1$, and $h_e(\bar{f}_{u,1}, \mathbf{v}_9) = e_5$.

Example 8. Using the labels in Table 2 and Figure 7, we see that $E_{0,p,1} = \{e_9\}$, $E_{1,q,2} = \{e_{12}, e_{14}\}$, and $v_1 = h(\bar{c}_p, \mathbf{v}_8)$ is an anchor.

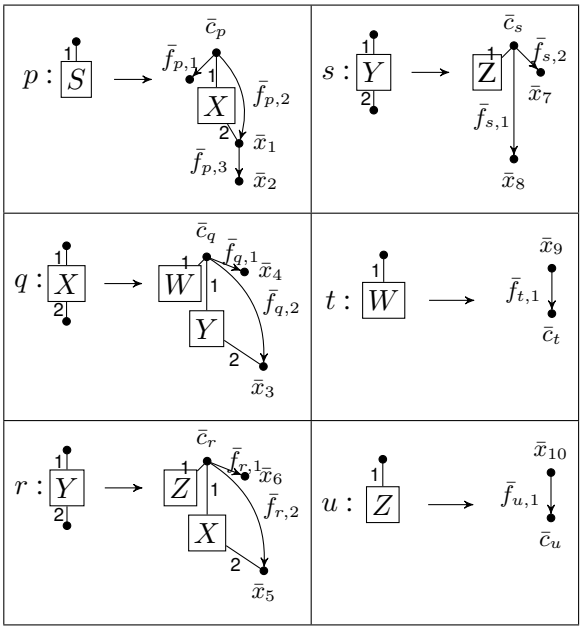

Table 2: The productions from Table 1 with variable names added to each of the nodes and terminal edges. Node variables of the form $\bar{c}_x$ for $x \in \{p, q, r, s, t, u\}$ indicate anchors.

4.1.1 Path Properties of RGLs

The precondition will exploit the properties of RGGs, particularly the properties of paths between nodes. Let $\mathcal{G}$ be an RGG, $G \in L(\mathcal{G})$, and let $\mathbf{T}$ be a derivation tree of G. In the following, we relate paths within individual productions in P (denoted π) to paths in G (denoted λ). For each e in G, we define $\mathrm{o}(e) = (i, p, j)$ iff $e \in E_{i,p,j}$.

For every path λ in G of the form

$$(v, i_1, e_1, j_1, v_1)(v_1, i_2, e_2, j_2, v_2) \dots (v_{k-1}, i_k, e_k, j_k, v')$$

we define its **trace** as the sequence

$$\mathrm{tr}(\lambda) := (\mathrm{o}(e_1), i_1, j_1)(\mathrm{o}(e_2), i_2, j_2) \dots (\mathrm{o}(e_k), i_k, j_k).$$

Now let π be a path

$$(\bar{v}, i_1, \bar{e}_1, j_1, \bar{v}_1) \dots (\bar{v}_{k-1}, i_k, \bar{e}_k, j_k, \bar{v}')$$

in $R(p)$ for some $p \in P$. Let $\mathbf{v} \in V_{\mathbf{T}}$, $p = lab_{\mathbf{T}}(\mathbf{v})$. We denote by $h(\pi, \mathbf{v})$ the following path in G:

$$(h(\bar{v}, \mathbf{v}), i_1, h(\bar{e}_1, \mathbf{v}), j_1, h(\bar{v}_1, \mathbf{v})) \dots \\ (h(\bar{v}_{k-1}, \mathbf{v}), i_1, h(\bar{e}_k, \mathbf{v}), j_1, h(\bar{v}', \mathbf{v}))$$

If $\mathbf{v}$ is an i-child of some node in $V_{\mathbf{T}}$ then $\mathrm{tr}(h(\pi, \mathbf{v}))$ is the sequence

$$((i, p, m_1), i_1, j_1) \dots ((i, p, m_k), i_k, j_k)$$

where $\bar{e}_j = \bar{f}_{p,m_j}$ for each $j \in [k]$. Note that $\mathrm{tr}(\pi) = \mathrm{tr}(h(\pi, \mathbf{v}))$. The trace is a property that remains constant when a path is projected from a production into a graph. This projection is not one-to-one since a production can be applied several times; a trace appears in the graph once for each application of the corresponding production in a derivation. For $\mathbf{v} \in V_{\mathbf{T}}$, we write $\pi \in R(\mathrm{lab}_{\mathbf{T}}(\mathbf{v}))$ to denote that π is a path in the production which is the label of $\mathbf{v}$.

Example 9. Let π be the path $(\bar{x}_3, 2, \bar{f}_{q,2}, 1, \bar{c}_q)(\bar{c}_q, 1, \bar{f}_{q,1}, 2, \bar{x}_4)$ in production q in Table 2. $h(\pi, \mathbf{v}_4)$ for $\mathbf{v}_4$ is the path $(v_1 1, 2, e_{13}, 1, v_4)(v_4, 1, e4, 2, v_5)$ in Figure 7, and its trace is $((1, q, 2), 2, 1)((1, q, 1), 1, 2)$.

Lemma 2 (Lemma 5.5 from (Courcelle, 1991))**.** *Let $\mathcal{G}$ be an RGG, G be a graph in $L(\mathcal{G})$, and $\mathbf{T}$ be a derivation tree of G. Let λ be a path in G of the form $h(\pi, \mathbf{v})$ for some $\mathbf{v} \in V_{\mathbf{T}}$ and some terminal path $\pi \in R(lab_{\mathbf{T}}(\mathbf{v}))$. The final node of π may be internal or external but every other node must be internal. If λ' is another path in G with the same trace and the same initial node as λ, then $\lambda' = \lambda$.*

Lemma 2 guarantees a unique trace for every path in a graph that is the projection of a path in a single production. By property C2 of RGGs, this guarantee must hold for at least one path from the anchor node of an int-production to every other node in the production. For ext-productions, all paths are of the form $\pi = (\bar{e}, i, \bar{v}_i)$, where e is the single nonterminal edge; these paths are also guaranteed unique traces.

4.1.2 MSO Formulas for the Precondition

Given an assignment to our parameters, we can use the path property in Lemma 2 to define some useful MSO statements. The first, ANC, relates anchors to the nodes in the graph. Throughout this section, given a derivation tree $\mathbf{T}$, we will refer to $\alpha_{\mathbf{T}}$ which is the parameter assignment from $\mathcal{W}$ to $V_G \cup E_G$ as defined above.

Lemma 3 (Lemma 5.6 from (Courcelle, 1991))**.** *Let $\mathcal{G}$ be an RGG, G be a graph in $L(\mathcal{G})$, and $\mathbf{T}$ be a derivation tree of G. For every $p \in P$, every $i \in [0, |NT(P)|]$, and every node $\bar{x} \in R(p)$, one can construct a formula $\mathrm{ANC}_{p,i,\bar{x}}(u, w, \{\mathcal{W}\})$ such that, for every $u \in V_G \cup E_G$, $w \in V_G$:*

$$(G, u, w, \alpha_{\mathbf{T}}) \models \mathrm{ANC}_{p,i,\bar{x}}(u, w, \{\mathcal{W}\})$$

iff $u = h(\bar{c}_p, \mathbf{v})$ and $w = h_v(\bar{x}, \mathbf{v})$ for some $\mathbf{v} \in V_{\mathbf{T}}$ which is an i-child and $p = lab_{\mathbf{T}}(\mathbf{v})$.

We say that node u **anchors** node v if for some p, i and $\bar{x}$, $\mathrm{ANC}_{p,i,\bar{x}}(u, v, \{\mathcal{W}\})$ holds. We use the fact that a node or edge anchors itself to establish its corresponding production.

Example 10. Looking at Table 2 and Figure 7, we can establish that $\text{ANC}_{p,0,\bar{x}_1}(v_5, v_{11}, \{\mathcal{W}\})$ holds and that $v_5 \in C_{0,p}$.

The next MSO formula we construct relates pairs of anchors to each other. Since the anchors define the output domain of the transducer, the formula PAR defines the edges of the output.

Lemma 4 (Lemma 5.7 of (Courcelle, 1991))**.** *Let $\mathcal{G}$ be an RGG, G be in $L(\mathcal{G})$, $\mathbf{T}$ be a derivation tree of G, and α be the parameter assignment defined with respect to $\mathbf{T}$. One can construct a formula $\text{PAR}_{p,i,p',i'}(u, w, \{\mathcal{W}\})$ such that, for $u, w \in V_G \cup E_G$:*

$(G, u, w, \alpha) \models \text{PAR}_{p,i,p',i'}(u, w, \{\mathcal{W}\})$

iff $u = h(\bar{c}_p, \mathbf{v}), w = h(\bar{c}_{p'}, \mathbf{v}')$ for some $\mathbf{v}, \mathbf{v}'$ in $V_\mathbf{T}$ where $p = lab_\mathbf{T}(\mathbf{v})$, $p' = lab_\mathbf{T}(\mathbf{v}')$, $\mathbf{v}$ is an i-child, and $\mathbf{v}'$ is the i'th child of $\mathbf{v}$ in $\mathbf{T}$.

If $\text{PAR}_{p,i,p',i'}(u, u', \{\mathcal{W}\})$ holds, then u will become the parent of u' in the output tree. The proof of this lemma relies on C1 of RGG.

Example 11. From Table 2 and Figure 7, $\text{PAR}_{q,1,s,2}(v_2, v_1, \{\mathcal{W}\})$ holds.

As introduced in §3, we have a binary equivalence relation $\sim$ over pairs of the form $(\bar{x}, \mathbf{v})$ where $\bar{x}$ is a node in a production p and $\mathbf{v}$ is a node in the derivation tree with label p. We use this relation for the precondition of the transducer so that a pair of nodes are only fused if the grammar and derivation tree allows them to be. We project $\sim$ into the graph to construct a relation over anchors such that $\text{FUSE}_{p,i,\bar{x},p',i',\bar{x}'}(u, u', \{\mathcal{W}\})$ holds if and only if $(\bar{x}, \mathbf{v}) \sim (\bar{x}', \mathbf{v}')$, $u = h(\bar{c}_p, \mathbf{v}), u' = h(\bar{c}_{p'}, \mathbf{v}')$, and $h(\bar{x}, \mathbf{v}) = h(\bar{x}', \mathbf{v}')$.

Lemma 5. *Let $\mathcal{G}$ be an RGG, G be in $L(\mathcal{G})$, and $\mathbf{T}$ be a derivation tree of G. One can construct a formula $\text{FUSE}_{p,i,\bar{x},p',i',\bar{x}'}(u, u', \{\mathcal{W}\})$ such that, for $u, u' \in V_G \cup E_G$:*

$(G, u, u', \{\mathcal{W}\}) \models \text{FUSE}_{p,i,\bar{x},p',i',\bar{x}'}(u, u', \{\mathcal{W}\})$

iff $u = h(\bar{c}_p, \mathbf{v}), u' = h(\bar{c}_{p'}, \mathbf{v}')$ for some $\mathbf{v}, \mathbf{v}'$ in $V_\mathbf{T}$ where $p = lab_\mathbf{T}(\mathbf{v}), p' = lab_\mathbf{T}(\mathbf{v}')$,$\mathbf{v}$ is the ith child of some node, $\mathbf{v}'$ is the i'th child of some node, and $h_v(\bar{x}, \mathbf{v}) = h_v(\bar{x}', \mathbf{v}')$.

Example 12. From Table 2 and Figure 7, we can see that $\text{FUSE}_{p,0,\bar{x}_1,s,2,\bar{x}_8}(v_5, v_1, \{\mathcal{W}\})$ holds since $v_5 = h_v(\bar{c}_p, \mathbf{v}_1)$, $v_1 = h_v(\bar{c}_s, \mathbf{v}_8)$, $v_{11} = h_v(\bar{x}_1, \mathbf{v}_1) = h_v(\bar{x}_8, \mathbf{v}_8)$, $\text{ANC}_{p,0,\bar{x}_1}(v_5, v_6, \{\mathcal{W}\})$ and $\text{ANC}_{s,2,\bar{x}_8}(v_1, v_{11}, \{\mathcal{W}\})$

4.1.3 The Precondition of the Transducer

Let X be in N, then $P_X = \{p \in P | L(p) = X\}$, and an X-derivation tree is a derivation tree with respect to X as the start symbol (in this case, the root will have label in P_X). An S-derivation tree is referred to simply as a derivation tree.

Edge Requirements

(E1) $\alpha(\mathcal{E})$ partitions E_G,

(E2) for all $e \in \alpha(E_{i,p,j})$ e has label $\gamma_{p,j}$

(E3) there is a unique $p \in P_X$ such that $\alpha(E_{0,p,j})$ has exactly one element for each $j \in [|\text{T}(p)|]$ and for every $p' \neq p$, $\alpha(E_{0,p',j})$ is empty for all j.

Recall the MSO statement $\text{PART}(X_1, \ldots, X_n)$ from Equation 1 which expresses that $X_1, \ldots, X_n$ form a partition over the domain. We can also define a partition over a restricted domain Y to be:

$$\text{resPART}(Y, X_1, \ldots, X_n) :$$
$$\forall x \in Y [x \in X_1 \cup \cdots \cup X_n \wedge \neg x \in X_1 \cap X_2$$
$$\wedge \neg x \in X_1 \cap X_3 \wedge \cdots \wedge \neg x \in X_{n-1} \cap X_n]$$

Using this formula, the requirements E1,E2 and E3 are all expressible in MSO as follows:

$$\text{EDGE}_X(\mathcal{W}) : \text{resPART}(E_G, \mathcal{E}) \wedge$$
$$\bigwedge_{i,p,j} \forall e \in E_{i,p,j} \text{lab}_{\gamma_{p,j}}(e) \wedge$$
$$\bigvee_{p \in P_X} [\bigwedge_j \exists! e \in E_{0,p,j} \wedge \bigwedge_{p' \in P, p' \neq p} \bigwedge_j E_{0,p',j} = \emptyset]$$

Let $\text{EDGE}(\mathcal{W}) = \text{EDGE}_S(\mathcal{W})$. In using the symbol $\wedge_{i,p,j}$ we are quantifying over $i \in [0, |\text{NT}(P)|]$, $p \in P$, and $j \in [|\text{T}(p)|]$.

Lemma 6. *Let $\mathcal{G}$ be an RGG and let $G \in L(\mathcal{G})$ then for each derivation tree $\mathbf{T}$ of G, $(G, \alpha_\mathbf{T}) \models \text{EDGE}(\mathcal{W})$.*

Example 13. For the grammar in Table 2, and derivation tree and graph in Figure 7, we obtain $\mathcal{E} = \{E_{0,p,1} = \{e_9\}, E_{0,p,2} = \{e_{14}\}, E_{0,p,3} = \{e_{15}\}, E_{1,q,1} = \{e_4, e_2\}, E_{1,q,2} = \{e_{13}, e_{11}\}, E_{2,r,1} = \{e_3\}, E_{2,r,2} = \{e_{12}\}, E_{2,s,1} = \{e_1\}, E_{2,s,2} = \{e_{10}\}, E_{1,t,1} = \{e_6, e_8\}, E_{2,u,1} = \{e_7\}, E_{1,u,1} = \{e_5\}\}$. This clearly forms a partition of the edges, and we can easily check that the rest of the requirements of EDGE also hold.

Decomposition into Subgraphs This constraint partitions the graph into a set of connected subgraphs, each of which is isomorphic to the terminal subgraph of the right-hand side of some production. The requirements are:

(S1) Every node in G is attached to some edge,

(S2) for each anchor $u \in C_{i,p}$ we can identify a unique edge $e \in E_{i,p,j}$ for each $j \in |\mathrm{T}(p)|$ such that u anchors all of the endpoints of e,

(S3) for each edge $e \in E_{i,p,j}$ we can identify a unique anchor $u \in C_{i,p}$ such that u anchors all of the endpoints of e.

$\textsc{subgraph}_{i,p,j}(\mathcal{W})$:
$(\forall v \exists e \vee_j \mathrm{vert}(e,j) = v) \wedge$
$\Big(\forall c \in C_{i,p} \exists! e \in E_{i,p,j}$
$\exists v_1 \textsc{anc}_{p,i,\bar{x}_1}(c, v_1, \{\mathcal{W}\}) \wedge \cdots \wedge$
$\exists v_{j_k} \textsc{anc}_{p,i,\bar{x}_{j_k}}(c, v_{j_k}, \{\mathcal{W}\}) \wedge$
$edg(e, v_1, \ldots, v_{j_k})\Big) \wedge$
$\Big(\forall e \in E_{i,p,j} \exists! c \in C_{i,p}$
$\exists v_1 \textsc{anc}_{p,i,\bar{x}_1}(c, v_1, \{\mathcal{W}\}) \wedge \cdots \wedge$
$\exists v_{j_k} \textsc{anc}_{p,i,\bar{x}_{j_k}}(c, v_{j_k}, \{\mathcal{W}\}) \wedge$
$\mathrm{edg}(e, v_1, \ldots, v_{j_k})\Big)$
Define $\textsc{subgraph}(\mathcal{W}) : \bigwedge_{i,p,j} \textsc{subgraph}_{i,p,j}$.

Lemma 7. *Let $\mathcal{G}$ be an RGG and let $G \in L(\mathcal{G})$ then for each derivation tree $\mathbf{T}$ of G, $(G, \alpha_{\mathbf{T}}) \models \textsc{subgraph}(\mathcal{W})$.*

Example 14. For the graph in Figure 7, we look at $\textsc{subgraph}_{1,q,1}$. We need to look at $C_{1,q} = \{v_4, v_2\}$, and $E_{1,q,1} = \{e_4, e_2\}$. Looking at the graph, we can see that each of $\textsc{anc}_{q,1,\bar{c}_q}(v_4, v_4, \{\mathcal{W}\})$, $\textsc{anc}_{q,1,\bar{x}_4}(v_4, v_5, \{\mathcal{W}\})$, $\textsc{anc}_{q,1,\bar{c}_q}(v_2, v_2, \{\mathcal{W}\})$, $\textsc{anc}_{q,1,\bar{x}_4}(v_2, v_3, \{\mathcal{W}\})$ hold. The label 'arg1' is on e_2, e_4 and $f_{q,1}$ so we can quickly verify that this shows that the graph satisfies $\textsc{subgraph}_{1,q,1}$.

Subgraph Composition

We require that for a graph G with derivation tree $\mathbf{T}$, $(G, \alpha_{\mathbf{T}}) \models \textsc{anc}_{p,i,\bar{x}}(u, v, \{\mathcal{W}\})$ for $u \in C_{i,p}$ and $(G, \alpha_{\mathbf{T}}) \models \textsc{anc}_{p',i',\bar{x}'}(u', v, \{\mathcal{W}\})$ for $u' \in C_{i',p'}$ if and only if $(G, \alpha_{\mathbf{T}}) \models \textsc{fuse}_{p,i,\bar{x},p',i',\bar{x}'}(u, u', \{\mathcal{W}\})$ holds. This part of the precondition ensures that two different ways of looking at how nodes can be fused agree with one another. The first is if a node can be anchored by two different anchors then this node must be the image of two nodes from different production applications. The second is that we have FUSE which is the equivalence relation generated by a relation over the neighbouring nodes in the derivation tree.

$\textsc{share}(\mathcal{W})$:
$$\bigwedge_{i,p,\bar{x},i',p',\bar{x}'} \forall c_1 \in C_{i,p} \forall c_2 \in C_{i',p'}$$
$\Big(\textsc{anc}_{p,i,\bar{x}}(c_1, v, \{\mathcal{W}\}) \wedge$
$\textsc{anc}_{p',i',\bar{x}'}(c_2, v, \{\mathcal{W}\})$
$\leftrightarrow \textsc{fuse}_{i,p,\bar{x},i',p',\bar{x}'}(u, u', \{\mathcal{W}\})\Big)$

Lemma 8. *Let $\mathcal{G}$ be an RGG and let $G \in L(\mathcal{G})$ then for each derivation tree $\mathbf{T}$ of G, there exists $\alpha_{\mathbf{T}}$ such that $(G, \alpha_{\mathbf{T}}) \models \textsc{share}(\mathcal{W})$.*

Example 15. Looking at Table 2 and Figure 7, $\textsc{fuse}_{p,0,\bar{x}_1,s,2,\bar{x}_8}(v_5, v_1, \{\mathcal{W}\})$ holds and $\textsc{anc}_{p,0,\bar{x}_1}(v_5, v_6, \{\mathcal{W}\})$ and $\textsc{anc}_{s,2,\bar{x}_8}(v_1, v_6, \{\mathcal{W}\})$ both also hold.

The proof of each of the above lemmas is available in the supplementary materials. In each of these proofs, we prove by induction on the size of $\mathbf{T}$ that $(G, \alpha_{\mathbf{T}}) \models R(\mathcal{W})$ for $R \in \{\textsc{edge}, \textsc{subgraph}, \textsc{share}\}$. In each induction, we use the equations (defined below) which express $\alpha_{\mathbf{T}}$ in terms of the parameter assignments of sub-trees of $\mathbf{T}$.

Let $G \in L_X(\mathcal{G})$ and $q : X \to H$ such that H has nonterminals $Y_1, \ldots, Y_n$ and $G = H[Y_1/H_1] \ldots [Y_n/H_n]$. Then $H_\eta \in L_{Y_\eta}(\mathcal{G})$ for each $\eta \in [n]$. Let $\mathbf{T}_\eta$ be a derivation tree for H_η and let $\alpha_{\mathbf{T}_\eta}$ be the assignment of $\mathcal{W}$ to the nodes and edges in H_η. Then we can define $\alpha_{\mathbf{T}}(\mathcal{E})$ in terms of the set of $\alpha_{\mathbf{T}_\eta}(\mathcal{E})$s:

$$\alpha_{\mathbf{T}} : \begin{cases} e \in E_{i,p,j} & \text{if } e \in E_{H_\eta}, \alpha_{\mathbf{T}_\eta} : e \in E_{i,p,j}, i \neq 0 \\ e \in E_{\eta,p,j} & \text{if } e \in E_{H_\eta}, \alpha_{\mathbf{T}_\eta} : e \in E_{0,p,j} \\ e \in E_{0,q,j} & \text{if } e \in E_H, e = h_e(\bar{f}_{q,j}, \mathbf{v}_0). \end{cases} \tag{2}$$

Where $e = h_e(\bar{f}_{q,j}, \mathbf{v}_0)$ means that e can be uniquely identified as corresponding to $\bar{f}_{q,j}$ since H and $R(q)$ are isomorphic and $\mathbf{v}_0$ is the root of $\mathbf{T}$. For the anchor set,

$$\alpha_{\mathbf{T}}(\mathcal{C}) = c \cup_{\eta \in [n]} \alpha_{\mathbf{T}_\eta}(\mathcal{C}) \tag{3}$$

where $c = h(\bar{c}_q, \mathbf{v}_0)$.

4.1.4 RGLs Satisfy the Precondition

The precondition of the transducer is the conjunction of each of these formulas,

$\rho_X(\mathcal{W}) : \textsc{edge}_X(\mathcal{W}) \wedge \textsc{share}(\mathcal{W}) \wedge \textsc{subgraph}(\mathcal{W})$

Define $\rho(\mathcal{W}) = \rho_S(\mathcal{W})$.

Proposition 1. *Let $\mathcal{G}$ be an RGG and let $G \in L(\mathcal{G})$, then for each derivation tree $\mathbf{T}$ of G,*

there exists a parameter assignment $\alpha_{\mathbf{T}}$ such that $(G, \alpha_{\mathbf{T}}) \models \rho(\mathcal{W})$.

Proof. We use the parameter assignment $\alpha_{\mathbf{T}}$ which is defined from $\mathbf{T}$ in §4.1. Lemma 6 proves that $(G, \alpha_{\mathbf{T}}) \models \text{EDGE}(\mathcal{W})$. Lemma 7 proves that $(G, \alpha_{\mathbf{T}}) \models \text{SUBGRAPH}(\mathcal{W})$. Lemma 8 proves that $(G, \alpha_{\mathbf{T}}) \models \text{SHARE}(\mathcal{W})$. Therefore, $(G, \alpha_{\mathbf{T}}) \models \rho(\mathcal{W})$. □

4.2 Parsing as Transduction

The transducer is made up of three types of formulas: the precondition, the domain formulas, and the relation formulas. We have established the precondition $\rho(\mathcal{W})$ and next we define the domain and relation formulas. The domain formulas define the nodes of the derivation tree and so we write $\text{node}(x, \{\mathcal{W}\})$. The relation formulas define which output node is the ith child of another output node, written $\text{child}_i(x, y, \{\mathcal{W}\})$, and the labels of the output nodes, written $\text{lab}_p(x, \{\mathcal{W}\})$.

The domain of the output for a parameter assignment α is $D_{\mathbf{T}}$ where:
$D_{\mathbf{T}}(\alpha) : \{x \mid (G, x, \alpha) \models \text{node}(x, \{\mathcal{W}\})\}$
and $\text{node}(x, \{\mathcal{W}\}) : x \in \mathcal{C}$.

The relation formula $\text{child}_r(x, y, \{\mathcal{W}\})$ defines the edges of the output of the transducer. We use the formula $\text{PAR}_{p,i,p',i'}(u, u', \{\mathcal{W}\})$ from Lemma 4, this encodes that the derivation tree node corresponding to u' is the i'th child of the node corresponding to u (which itself is the ith child of some other node).

$$\text{child}_{i'}(x, y, \{\mathcal{W}\}) : \bigvee_{i,p,p'} (\text{PAR}_{p,i,p',i'}(x, y, \{\mathcal{W}\})$$

We also need to assign labels to the tree nodes which can be done via the unary relation:

$$\text{lab}_p(x, \{\mathcal{W}\}) : \bigvee_i x \in C_{i,p}$$

Example 16. Figure 8 shows the output of the transducer when it takes Figure 7 as input with α defined as in the previous examples. The domain formulas specify the existence of the 9 nodes and the relation formulas specify the edges between the nodes, labelled by PAR formulas, and the labels of the nodes, according to the $C_{i,p}$ sets.

We have now defined each part of the transducer τ from graphs to their derivation trees. Let $\mathcal{G}$ be an RGG, and let $X \in N$. Then the corresponding transducer τ_X is
$\langle \rho_X(\{\mathcal{W}\}), \text{node}(x, \{\mathcal{W}\}),$
$(\text{lab}_p(x, \{\mathcal{W}\}))_{p \in P},$
$(\text{edge}_r(x, y, \{\mathcal{W}\}))_{r \in [|\text{NT}(P)|]} \rangle$.

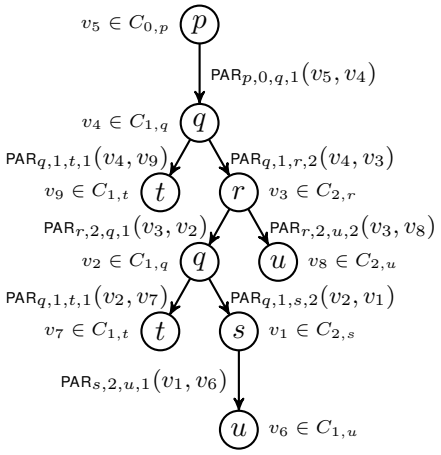

Figure 8: The output of the transducer, variable names are based on those of Figure 7. The PAR formulas are there to explain why the edge exists and the $v \in C_{i,p}$ formulas are there to show where the node labels come from.

For start symbol S of $\mathcal{G}$, let $\tau = \tau_S$. Let G be a graph in $L(\mathcal{G})$, and let α be a parameter assignment such that $(G, \alpha) \models \rho(\{\mathcal{W}\})$. Then the output of the transducer with respect to α is
$\tau(G, \alpha) = (V_H, \text{lab}_H, (\text{child}^i_H)_{i \in [0, |\text{NT}(P)|]})$
where $V_H = D_{\mathbf{T}}(\alpha) = \{x \mid (G, x, \alpha) \models \text{node}(x, \{\mathcal{W}\})\}$, $\text{lab}_H : V_H \to P$ such that $\text{lab}_H(x) = p$ if $x \in \alpha(C_{i,p})$ for some i, and $\text{child}^i_H : V_H \to V_H$ such that $\text{child}^H_i(x, y)$ if $(G, x, y, \alpha) \models \text{PAR}_{p,i,p',r}(x, y, \{\mathcal{W}\})$.

4.2.1 Transducer Output and Derivation Trees

We will show that for each $G \in L(\mathcal{G})$ if $\mathbf{T}$ is a derivation tree of G then $\mathbf{T} \in \tau(G)$. We will also show that for each $\mathbf{T} \in \tau(G)$, if it is a derivation tree in $\mathcal{T}_{\mathcal{G}}$ then it is a derivation tree of G.

Proposition 2. *Let $\mathcal{G}$ be an RGG and τ be the corresponding transducer. Let $G \in L(\mathcal{G})$ and $\mathbf{T}$ be a derivation tree of G. Then $\mathbf{T} \in \tau(G)$.*

By Proposition 2, we know that for each G, $\{\mathbf{T} | \text{val}(\mathbf{T}) = G\} \subseteq \tau(G)$.

Proposition 3. *Let $\mathcal{G}$ be an RGG and $G \in L(\mathcal{G})$. Let α be a parameter assignment such that $(G, \alpha) \models \rho(\mathcal{W})$. Then if $\mathbf{T} = \tau(G, \alpha)$ is in $\mathcal{T}_{\mathcal{G}}$ then $\text{VAL}(\mathbf{T}) = G$.*

Theorem 2. *RGL ⊆ MSOL.*

Proof. Let $\mathcal{G}$ be an RGG and τ be the corresponding transducer. By Propositions 2 and 3, for each $G \in L(\mathcal{G})$, $\tau(G)$ is a set which contains all of the derivation trees of G and possibly other elements none of which are derivation trees of any $G' \in L(\mathcal{G})$ where $G' \neq G$. Therefore, for each $G \in L(\mathcal{G})$,
$\tau(G) \cap \mathcal{T}_\mathcal{G} = \{\mathbf{T} \in \mathcal{T}_\mathcal{G} \mid \text{VAL}(\mathbf{T}) = G\}$.

Therefore,
$\tau(L(\mathcal{G})) \cap \mathcal{T}_\mathcal{G} = \{\mathbf{T} \in \mathcal{T}_\mathcal{G} \mid \text{VAL}(\mathbf{T}) = G, G \in L(\mathcal{G})\}$.

And since $\{\mathbf{T} \in \mathcal{T}_\mathcal{G} \mid \text{VAL}(\mathbf{T}) = G, G \in L(\mathcal{G})\} = \mathcal{T}_\mathcal{G}$,
$\tau^{-1}(\tau(L(\mathcal{G})) \cap \mathcal{T}_\mathcal{G}) = \tau^{-1}(\mathcal{T}_\mathcal{G})$.

$\tau^{-1}(\tau(L(\mathcal{G})) \cap \mathcal{T}_\mathcal{G}) = \{G \in L(\mathcal{G}) \mid \tau(G) \cap \mathcal{T}_\mathcal{G} \neq \emptyset)\} = L(\mathcal{G})$. Therefore,
$L(\mathcal{G}) = \tau^{-1}(\mathcal{T}_\mathcal{G})$

and so by Theorem 1 and the fact that $\mathcal{T}_\mathcal{G}$ is MSO definable, $L(\mathcal{G})$ is MSO definable.

□

5 Conclusions and Discussion

Property C1 of RGGs is used repeatedly in the proof that RGL is in MSOL. This property implies connectedness of the terminal subgraph, a property that both Tree-like Grammars (Matheja et al., 2015) and Restricted DAG Grammars (Björklund et al., 2016) share, although both of these formalisms allow nodes that are connected only to nonterminals, which is forbidden in RGG. We suspect that all three families of languages are incomparable. That these restricted forms of HRG all share the property of connectedness suggests that it may be an important property. In particular, we plan to investigate whether connectedness of terminal subgraphs implies that an HRL is in MSOL.

Languages which contain graphs of the form shown in Figure 9 are MSOL but not in RGL or TLG; hence both RGL and TLG are proper subfamilies of SCFL. Languages of this form can be produced by RDG, whose relationship to SCFL is unknown. To produce graphs like this, we must allow productions containing nonterminals that are not incident to any internal node. We would need to allow this only in certain circumstances however, as we could easily produce a language of graphs that look like the graph in Figure 9 with equal numbers of a-labelled and b-labelled edges; such languages are not MSO-definable. On a technical level, allowing such extensions would mean that PAR no longer holds. (Courcelle, 1991) discusses this problem and introduces an alternative representation of derivation trees called **reduced trees** which enable some cases of this type to be defined in MSOL. This point requires further investigation.

Another possible extension would be to consider alternative forms of Lemma 2. Every MSO formula in the transducer depends on this lemma. We could potentially extend RGG if we can define other cases in which a path could be defined in terms of its trace and initial vertex. We intend to investigate such cases in future work.

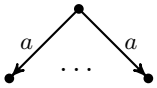

Figure 9: A graph where every edge is labelled a and has the same tail but each edge has a unique head.

Acknowledgments

This work was supported in part by the EPSRC Centre for Doctoral Training in Data Science, funded by the UK Engineering and Physical Sciences Research Council (grant EP/L016427/1) and the University of Edinburgh; and in part by a Google faculty research award (to AL). We thank Clara Vania, Sameer Bansal, Ida Szubert, Federico Fancellu, Antonis Anastasopoulos, Marco Damonte, and the anonymous reviews for helpful discussion of this work and comments on previous drafts of the paper.

References

Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (ucca). In *ACL (1)*. The Association for Computational Linguistics, pages 228–238. http://dblp.uni-trier.de/db/conf/acl/acl2013-1.html#AbendR13.

Cyril Allauzen, William Byrne, Adria de Gispert, Gonzalo Iglesias, and Michael Riley. 2014. Pushdown automata in statistical machine translation. *Computational Linguistics* .

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*. Association for Computational Linguistics, Sofia, Bulgaria, pages 178–186. http://www.aclweb.org/anthology/W13-2322.

Daniel Bauer and Owen Rambow. 2016. Hyperedge replacement and nonprojective dependency structures. In *Proceedings of the 12th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+12), June 29 - July 1, 2016, Heinrich Heine University, Düsseldorf, Germany*. pages 103–111. http://aclweb.org/anthology/W/W16/W16-3311.pdf.

Henrik Björklund, Frank Drewes, and Petter Ericson. 2016. *Between a Rock and a Hard Place – Uniform Parsing for Hyperedge Replacement DAG Grammars*, Springer International Publishing, Cham, pages 521–532. https://doi.org/10.1007/978-3-319-30000-9_40.

Mikolaj Bojanczyk and Michal Pilipczuk. 2016. Definability equals recognizability for graphs of bounded treewidth. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*. ACM, New York, NY, USA, LICS '16, pages 407–416. https://doi.org/10.1145/2933575.2934508.

Taylor L. Booth and Richard A. Thompson. 1973. Applying probability measures to abstract languages. *IEEE Transactions on Computers* 22(5):442–450. https://doi.org/http://doi.ieeecomputersociety.org/10.1109/TC.1973.223746.

Julius Richard Büchi. 1960. On a decision method in restricted second-order arithmetic. *Proceedings Logic, Methodology and Philosophy of Sciences* .

Julius Richard Büchi and Calvin Elgot. 1958. Decision problems of weak second order arithmetic and finite automata, part i. *Notices of the American Mathematical Society* page 5;834.

David Chiang, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, Bevan Jones, and Kevin Knight. 2013. Parsing graphs with hyperedge replacement grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 924–932. http://www.aclweb.org/anthology/P13-1091.

Bruno Courcelle. 1990. The monadic second-order logic of graphs i. recognizable sets of finite graphs. *Information and Computation* pages 12–75.

Bruno Courcelle. 1991. The monadic second-order logic of graphs V: on closing the gap between definability and recognizability. *Theoretical Computer Science* 80(2):153–202. https://doi.org/10.1016/0304-3975(91)90387-H.

Bruno Courcelle and Joost Engelfriet. 2011. *Graph Structure and Monadic Second-Order Logic, a Language Theoretic Approach*. Cambridge University Press.

Frank Drewes, Hans-Jörg Kreowski, and Annegret Habel. 1997. Hyperedge replacement graph grammars. In Grzegorz Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation*, World Scientific, pages 95–162.

Manfred Droste and Paul Gastin. 2005. *Weighted Automata and Weighted Logics*, Springer Berlin Heidelberg, Berlin, Heidelberg, pages 513–525. https://doi.org/10.1007/11523468_42.

Dan Flickinger, Yi Zhang, and Valia Kordoni. 2012. Deepbank : a dynamically annotated treebank of the wall street journal. In *Proceedings of the Eleventh International Workshop on Treebanks and Linguistic Theories (TLT11)*. Lisbon, pages 85–96. HU.

Jan Hajič, Eva Hajičová, Jarmila Panevov, Petr Sgall, Ondřej Bojar, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. 2012. Announcing prague czech-english dependency treebank 2.0. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uur Doan, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*. European Language Resources Association (ELRA), Istanbul, Turkey.

John E. Hopcroft and Jeffrey D. Ullman. 1979. *Introduction to automata theory, languages and computation*. Addison-Wesley.

Bevan Jones, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, and Kevin Knight. 2012. Semantics-based machine translation with hyperedge replacement grammars. In *Proceedings of COLING*.

Tsutomu Kamimura and Giora Slutzki. 1981. Parallel and two-way automata on directed ordered acyclic graphs. *Information and Control* 49(1):10–51.

Alexander Koller and Marco Kuhlmann. 2011. A generalized view on parsing and translation. In *Proceedings of the 12th International Conference on Parsing Technologies*. Association for Computational Linguistics, Dublin, Ireland, pages 2–13. http://www.aclweb.org/anthology/W11-2902.

Christoph Matheja, Christina Jansen, and Thomas Noll. 2015. *Tree-Like Grammars and Separation Logic*, Springer International Publishing, Cham, pages 90–108. https://doi.org/10.1007/978-3-319-26529-2_6.

Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. 2008. Speech recognition with weighted finite-state transducers. In Larry Rabiner and Fred Juang, editors, *Handbook on Speech Processing and Speech Communication, Part E: Speech recognition*, Springer.

Xiaochang Peng, Linfeng Song, and Daniel Gildea. 2015. A synchronous hyperedge replacement grammar based approach for AMR parsing. In *Proceedings of the 19th Conference on Computational Natural Language Learning, CoNLL 2015, Beijing, China, July 30-31, 2015*. pages 32–41. http://aclweb.org/anthology/K/K15/K15-1004.pdf.

Daniel Quernheim and Kevin Knight. 2012. Towards probabilistic acceptors and transducers for feature structures. In *Proceedings of the Sixth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Association for Computational Linguistics, Stroudsburg, PA, USA, SSST-6 '12, pages 76–85. http://dl.acm.org/citation.cfm?id=2392936.2392948.

Wolfgang Thomas. 1991. *Automata, Languages and Programming: 18th International Colloquium Madrid, Spain, July 8–12, 1991 Proceedings*, Springer Berlin Heidelberg, Berlin, Heidelberg, chapter On logics, tilings, and automata, pages 441–454. https://doi.org/10.1007/3-540-54233-7-154.

Boris Trakhtenbrot. 1961. Finite automata and logic of monadic predicates. *Doklady Akademii Nauk SSSR* pages 140:326–329.