(Re)introducing regular graph languages

Citation for published version:

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Publisher's PDF, also known as Version of record

Published In:
Proceedings of the 15th Meeting on the Mathematics of Language

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
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 Representation Bank (AMR; Bansarescu 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

Anna fehlt ihrem Kater → miss arg1 ev poss Anna’
Anna’s cat misses her

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’.

Rappoport, 2013). To make use of this data, we require probabilistic models of graphs.

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; Al-lauzen 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,
(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 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.\(^4\)

\(\text{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).}

\(^{1}\)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 a similar term in NLP.

\(^{3}\)The equality of SCFL and MSOL \(\cap\) HRL was recently proved by (Bojanczyk and Pilipczuk, 2016).

\(^{4}\)goo.gl/Z5L2gP
We assume that both the elements of \( \text{ext}_G \) and the elements of \( \text{att}_G(e) \) for each edge \( e \) are pair-wise 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 \( \text{ext}_G \) are labeled by their position in \( \text{ext}_G \). 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, \( \text{rank}(|\text{lab}(e)|) = k \)). 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 \( HG_{\Sigma, \Gamma} \) to be the set of all hypergraphs with node labels in \( \Sigma \) and edge labels in \( \Gamma \) (the hypergraphs as defined here have no node labels so are in \( HG_{\emptyset, \Gamma} \)).

**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 the 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:

\[
\begin{align*}
\text{att}_G(e_1) &= (v_1, v_2) & \text{lab}_G(e_1) &= a \\
\text{att}_G(e_2) &= (v_2, v_3) & \text{lab}_G(e_2) &= b \\
\text{att}_G(e_3) &= (v_1, v_4, v_3) & \text{lab}_G(e_3) &= X \\
\text{ext}_G &= (v_4, v_2).
\end{align*}
\]

MSO on graphs quantifies over nodes, sets of nodes, edges, and sets of edges. The atomic formulas are \( x \in X, x = y, \text{lab}_b(x), \) and \( \text{vert}(x, i) = y \). We construct MSO sentences using the atomic formulas, connectives \( \land, \lor, \neg, \Rightarrow, \land, \lor, \neg, \Rightarrow, \land, \lor, \neg, \Rightarrow, \) 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 \( \text{lab}_b(x) \) must be an edge. We define the formula \( \text{edg}(x, y_1, \ldots, y_k) : \text{vert}(x, 1) = y_1 \land \cdots \text{vert}(x, k) = y_k \land \forall k' > k \forall y \neg \text{vert}(x, k') = y \) 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.

\[
\begin{align*}
\forall x [x &\in X_1 \cup \cdots \cup X_n \land \neg x \in X_1 \cap X_2 \\
&\land \neg x \in X_1 \cap X_3 \land \cdots \neg x \in X_{n-1} \cap X_n]
\end{align*}
\]

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

\[
\exists ! x R(x) : \exists x R(x) \land \forall y R(y) \rightarrow x = y.
\]

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:

\[
\begin{align*}
\text{STRING} : & \forall y, \exists x_1 \exists x_2 \text{edg}(y, x_1, x_2) \\
& \exists x_1 \forall y - \text{vert}(y, 2) = x_1 \land \exists x_2 \forall y - \text{vert}(y, 1) = x_2 \\
& \land \forall x \neg x = x_1 \Rightarrow \exists y \exists x' \text{edg}(y, x', x) \\
& \land \forall x \neg x = x_2 \Rightarrow \exists y \exists x' \text{edg}(y, x, x')
\end{align*}
\]

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 \( x \in X_i \land \exists y \exists x' \text{edg}(y, x, x') \land \text{lab}_b(y) \Rightarrow x' \in X_j \).

From \( A \), we construct the formula \( \text{aut}_A \):

\[
\begin{align*}
\text{aut}_A : & \text{STRING} \land \exists X_0 \exists X_1 \text{PART}(X_0, X_1) \\
& \land \forall x \text{First}(x) \Rightarrow x \in X_0 \\
& \land \forall x \text{Last}(x) \Rightarrow x \in X_1 \\
& \land \forall x \in X_0 \exists y \exists x' \text{edg}(y, x, x') \land \text{lab}_b(y) \Rightarrow x' \in X_1 \\
& \land \forall x \in X_1 \exists y \exists x' \text{edg}(y, x, x') \land \text{lab}_b(y) \Rightarrow x' \in X_0
\end{align*}
\]

For a graph \( G \) and an MSO statement \( \phi \) we say that \( G \models \phi \) (or \( G \) satisfies \( \phi \)) when there is an assignment of variables of \( \phi \) to nodes and edges of
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 $X_1 = \{v_1, v_2, v_3\}$ then $G = \text{aut}_A$.

Let $\text{aut}'_A(X_0, X_1)$ be the MSO formula identical to $\text{aut}_A$ with $\exists X_0 \sqcap X_1$ removed from the beginning of the formula. $X_0$ and $X_1$ are free variables of $\text{aut}'_A$, and we refer to the set of free variables of a formula as its parameters. Given a graph $G$ and a formula $\phi(W)$ with parameters $W$, let $\alpha$ be function from $W$ to subsets of nodes and edges in $G$. Then we say that $(G, \alpha) \models \phi(W)$ if $G$ and $\alpha$ satisfy $\phi(W)$. We call $\alpha$ 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 $\alpha$.

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

We can use an MSO statement $\phi$ 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 $L(\phi_1) \cap L(\phi_2) = \{G \mid G \models \phi_1 \land \phi_2\}$. 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 $\tau = (\rho(W), \delta(x, W), (\theta_r(x_1, \ldots, x_{N(r)}, W))_{r \in R})$. $W$ is a set of parameters; $\rho(W)$ is a precondition which input graphs must satisfy; $\delta(x, W)$ is a domain formula defining the output domain (i.e., nodes); and $\theta_r(x_1, \ldots, x_{N(r)}, W)$ 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 non-determinism. Given a graph $G$ and a parameter assignment $\alpha$ from $W$ to $V_G \cup E_G$ such that $(G, \alpha) \models \rho(W)$, we define the output of the transducer $\tau(G, \alpha) = (D, R)$ such that $D = \{x \in G \mid (G, x, \alpha) \models \delta(x, W)\}$ and $R = \{\theta_r(x_1, \ldots, x_{N(r)}, \alpha) \mid (G, x_1, \ldots, x_{N(r)}, \alpha) \models \theta_r(x_1, \ldots, x_{N(r)}, W), r \in R\}$. Define $\tau(G) = \{\tau(G, \alpha) \mid (G, \alpha) \models \rho(W)\}$ 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 $V_{G'} = (V_G \cup V_H) - \text{ext}_H, E_{G'} = (E_G \cup E_H) - \{e\}$. Let $\text{ext}_H = (v_1, \ldots, v_k)$, $\text{att}_G(e) = (u_1, \ldots, u_k)$ 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 $(V_G \cup V_H)^k$ is also denoted $f$. Let $E = E_G - \{e\}$, then $\text{att}_{G'} = \text{att}_G|_E \cup (f \circ \text{att}_H)$. $\text{lab}_{G'} = \text{lab}_{G}|_E \cup \text{lab}_H$.

**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.
It can also be viewed in the related framework of interpreted regular tree grammars (Koller and Kuhlmann, 2011).
2. \[ p = \text{lab}_T(v), p' = \text{lab}_T(v'). \]
3. \[ \bar{x} \text{ is the } j \text{th node of } \text{NT}(v), \bar{y} = \text{ext}_R(p')(j). \]

We define \( \sim \) as the reflexive, symmetric, transitive closure of \( \sim \).

The mapping \( \text{VAL} \) 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 \( \text{VAL} \) maps a derivation tree to a graph.

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

**Lemma 1.** Let \( G \) be a HRG, and let \( G \) be a graph in \( L\langle G \rangle \) with derivation tree \( T \). If \( \bar{x} \) and \( \bar{x}' \) are nodes such that \[ h_v(\bar{x}, v) = h_v(\bar{x}', v') \] with \( v \neq v' \), then \( \bar{x} \) is an external node of \( G \). If \( \bar{x} \) is internal in \( G \), then \( \bar{x} \) is an external node of \( G \).

Consequently, if \( h_v(\bar{x}, v) = h_v(\bar{x}', v') \) 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 \( \text{vert}(e_r, i_r) = v_{r-1} \) and \( \text{vert}(e_r, j_r) = v_r \) 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 \( G \) is a **Regular Graph Grammar** if each nonterminal in \( N \) has rank at least one and for each \( p \in P \) 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 HRGs by definition; we will prove that they are also MSOLs by constructing the inverse of \( \text{VAL} \), 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.** \( \text{RGL} \subseteq \text{MSOL} \).

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
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 $T(p) = \{\bar{f}_p, 1, \ldots, \bar{f}_p, |T(p)|\}$ 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 $j \in [|T(p)|]$. Let $|NT(p)|$ be the number of nonterminal edges in $p$ and let $|NT(P)| = \max_{p \in P} |NT(p)|$. Given a node $v$ in a derivation tree $T$, we say that $v$ is an $i$-child if it is the $i$th child of some other node in $T$. By convention, the root node is the only 0-child.

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

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

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

$C_{i,p} = \{h(\tilde{e}_p, v) \mid p = \text{lab}_T(v) \text{, } v \text{ is an } i\text{-child}\}$

where $h = h_e \cup e_v$ since $\tilde{e}_p$ can either be an edge or a node. Let $C = \cup_i C_{i,p}$.

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).

**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, $h_e(\tilde{e}_s, v_8) = v_1$, and $h_e(\bar{f}_{u,1}, v_9) = e_5$.

**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 $v_1 = h(\tilde{e}_p, v_8)$ is an anchor.

Let $\mathcal{W} = \mathcal{E} \cup \mathcal{C}$.
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 \( T \) be a derivation tree of \( G \). In the following, we relate paths within individual productions in \( P \) (denoted \( \pi \)) to paths in \( G \) (denoted \( \lambda \)). For each \( e \) in \( G \), we define \( o(e) = (i, p, j) \) iff \( e \in E_{i,p,j} \).

For every path \( \lambda \) in \( G \) of the form
\[
(v, i_1, e_1, j_1, v_1)(v_1, i_2, e_2, j_2, v_2) \ldots (v_{k-1}, i_k, e_k, j_k, v_k)
\]
we define its trace as the sequence
\[
\text{tr}(\lambda) := (o(e_1), i_1, j_1)(o(e_2), i_2, j_2) \ldots (o(e_k), i_k, j_k).
\]

Now let \( \pi \) be a path
\[
(\bar{v}, i_1, \bar{e}_1, j_1, \bar{v}_1) \ldots (\bar{v}_{k-1}, i_k, \bar{e}_k, j_k, \bar{v})
\]
in \( R(p) \) for some \( p \in P \). Let \( v \in V_T \), \( p = \text{lab}_T(v) \). We denote by \( h(\pi, v) \) the following path in \( G \):
\[
(h(\bar{v}, v), i_1, h(\bar{e}_1, v), j_1, h(\bar{v}_1, v)) \ldots (h(\bar{v}_{k-1}, v), i_1, h(\bar{e}_k, v), j_1, h(\bar{v}', v))
\]

If \( v \) is an \( i \)-child of some node in \( V_T \) then
\[
\text{tr}(h(\pi, v)) \text{ is the sequence}
\]
\[
((i, p, m_1), i_1, j_1) \ldots ((i, p, m_k), i_k, j_k)
\]
where \( \bar{e}_j = f_{p,m_j} \) for each \( j \in [k] \). Note that
\[
\text{tr}(\pi) = \text{tr}(h(\pi, v)).
\]
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 \( v \in V_T \), we write \( \pi \in R(\text{lab}_T(v)) \) to denote that \( \pi \) is a path in the production which is the label of \( v \).

**Example 9.** Let \( \pi \) be the path \((x_3, 2, f_q(2), 1, \bar{e}_q)(\bar{e}_q, 1, f_q, 1, 2, \bar{e}_q) \) in production \( q \) in Table 2. \( h(\pi, v_4) \) for \( v_4 \) is the path \((v_1, 2, e_{13}, 1, v_4)((v_4, 1, e_{4}, 2, v_5) \) 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 \( T \) be a derivation tree of \( G \). Let \( \lambda \) be a path in \( G \) of the form \( h(\pi, v) \) for some \( v \in V_T \) and some terminal path \( \pi \in R(\text{lab}_T(v)) \). The final node of \( \pi \) may be internal or external but every other node must be internal. If \( \lambda \) is another path in \( G \) with the same trace and the same initial node as \( \lambda \), 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 \( \pi = (\bar{e}, i, \bar{v}) \), 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, \( \alpha_T \), relates anchors to the nodes in the graph. Throughout this section, given a derivation tree \( T \), we will refer to \( \alpha_T \) which is the parameter assignment from \( W \) 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 \( T \) 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 \( \text{ANC}_{p,i,\bar{x}}(u, w, \{W\}) \) such that, for every \( u \in V_G \cup E_G \), \( w \in V_G \):
\[
(G, u, w, \alpha_T) = \text{ANC}_{p,i,\bar{x}}(u, w, \{W\})
\]
iff \( u = h(\bar{e}, v) \) and \( w = h_v(\bar{x}, v) \) for some \( v \in V_T \) which is an \( i \)-child and \( p = \text{lab}_T(v) \).

We say that node \( u \) anchors node \( v \) if for some \( p, i \) and \( \bar{x} \), \( \text{ANC}_{p,i,\bar{x}}(u, w, \{W\}) \) 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 $\text{ANC}_{p,0,\bar{x}_1}(v_5, v_{11}, \{W\})$ 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 \text{PAR} defines the edges of the output.

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

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

iff $u = h(c_p, v), w = h(c_{p'}, v')$ for some $v, v'$ in $V_T$ where $p = \text{lab}_{p}(v), p' = \text{lab}_{p'}(v')$, $v$ is an $i$-child, and $v'$ is the $i'$th child of $v$ in $T$.

If $\text{PAR}_{p,i,p',i'}(u, u', \{W\})$ 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, $\text{PAR}_{q,1,2}(v_2, v_1, \{W\})$ holds.

As introduced in §3, we have a binary equivalence relation $\sim$ over pairs of the form $(\bar{x}, v)$ where $\bar{x}$ is a node in a production $p$ and $v$ 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 allow them to be. We project $\sim$ into the graph to construct a relation over anchors such that $\text{FUSE}_{p,i,\bar{x},p',i'}(u, u', \{W\})$ holds if and only if $(\bar{x}, v) \sim (\bar{x}', v')$, $u = h(c_p, v), u' = h(c_{p'}, v')$, and $h(\bar{x}, v) = h(\bar{x}', v')$.

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

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

iff $u = h(c_p, v), u' = h(c_{p'}, v')$ for some $v, v'$ in $V_T$ where $p = \text{lab}_{p}(v), p' = \text{lab}_{p'}(v')$, $v$ is the $i$th child of some node, $v'$ is the $i'$th child of some node, and $h(\bar{x}, v) = h(\bar{x}', v')$.

Example 12. From Table 2 and Figure 7, we can see that $\text{FUSE}_{p,0,\bar{x}_1,s,\bar{x}_2}(v_5, v_{11}, \{W\})$ holds since $v_5 = h(\bar{x}_1, v_1), v_1 = h(\bar{x}_8, v_8), v_{11} = h(\bar{x}_1, v_1) = h(\bar{x}_8, v_8), \text{ANC}_{p,0,\bar{x}_1}(v_5, v_{11}, \{W\})$ and $\text{ANC}_{s,2,\bar{x}_2}(v_1, v_{11}, \{W\})$.

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: $\alpha(E) \models \text{PAR}(E_G)$.

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 $j \in \{T(p)\}$ 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:

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

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

$\text{EDGE}_X(W) : \text{resPART}(E_G, E) \land \bigwedge_{i,p,j} \forall e \in E_{i,p,j} \exists \gamma_{p,j}(e) \land \bigwedge_{p \in P_X} \exists j \not\in \{E_{0,p,j} = \emptyset\} \land \bigwedge_{p' \in P_X, p' \neq p} \exists j \not\in \{E_{0,p',j} = \emptyset\}$.

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

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

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

\[
\text{SUBGRAPH}_{i,p,j}(W) : \\
(\forall v \exists e \forall j \text{ vert}(e, j) = v) \land \\
(\forall c \in C_{i,p} \exists e \in E_{i,p,j} \\
\exists v_1 \anc_{p,i,x_1} (c, v_1, \{W\}) \land \ldots \land \\
\exists v_j \anc_{p,i,x_j} (c, v_j, \{W\}) \land \\
edg(e, v_1, \ldots, v_j))
\]

Define \( \text{SUBGRAPH}(W) : \bigwedge_{i,p,j} \text{SUBGRAPH}_{i,p,j} \).

**Lemma 7.** Let \( G \) be an RGG and let \( G \in L(G) \) then for each derivation tree \( T \) of \( G \), \((G, \alpha_T) \models \text{SUBGRAPH}(W) \).

**Example 14.** For the graph in Figure 7, we look at \( \text{SUBGRAPH}_{1,q,1} \). We need to look at \( C_{1,q} = \{v_4, v_2\} \), and \( E_{1,q,1} = \{e_2, e_4\} \). Looking at the graph, we can see that each of \( \anc_{q,1,c_2} (v_4, v_2, \{W\}), \anc_{q,1,c_4} (v_4, v_5, \{W\}), \anc_{q,1,c_6} (v_2, v_3, \{W\}) \) holds. The label ‘arg1’ is on \( e_2, e_4 \) and \( f_4,q \), so we can quickly verify that this shows that the graph satisfies \( \text{SUBGRAPH}_{1,q,1} \).

**Subgraph Composition**

We require that for a graph \( G \) with derivation tree \( T \), \((G, \alpha_T) \models \anc_{p,i,x} (u, v, \{W\}) \) for \( u \in C_{i,p} \) and \((G, \alpha_T) \models \anc_{p',i',x'} (u', v', \{W\}) \) for \( u' \in C_{i',p'} \) if and only if \((G, \alpha_T) \models \fuse_{p,i,x,p',i',x'} (u, u', \{W\}) \) 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 \( \text{FUSE} \) which is the equivalence relation generated by a relation over the neighbouring nodes in the derivation tree.

\[
\text{SHARE}(W) : \\
\bigwedge_{i,p,j} \forall c_1 \in C_{i,p} \forall c_2 \in C_{i',p'} \\
(\anc_{p,i,x} (c_1, v, \{W\}) \land \\
\anc_{p',i',x'} (c_2, v', \{W\}) \land \\
\fuse_{p,i,x,p',i',x'} (u, u', \{W\})
\]

**Lemma 8.** Let \( G \) be an RGG and let \( G \in L(G) \) then for each derivation tree \( T \) of \( G \), there exists \( \alpha_T \) such that \((G, \alpha_T) \models \text{SHARE}(W) \).

**Example 15.** Looking at Table 2 and Figure 7, \( \fuse_{p,0,x_1,s_{2,2}} (v_5, v_1, \{W\}) \) holds and \( \anc_{p,0,x_1} (v_5, v_6, \{W\}) \) and \( \anc_{s,2,s_8} (v_1, v_6, \{W\}) \) 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 \( T \) that \((G, \alpha_T) \models R(W) \) for \( R \in \{\text{EDGE}, \text{SUBGRAPH}, \text{SHARE}\} \). In each induction, we use the equations (defined below) which express \( \alpha_T \) in terms of the parameter assignments of sub-trees of \( T \).

Let \( G \in L_X (G) \) 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 \( H_\eta \in L_{Y_\eta} (G) \) for each \( \eta \in [n] \). Let \( T_\eta \) be a derivation tree for \( H_\eta \) and let \( \alpha_{T_\eta} \) be the assignment of \( W \) to the nodes and edges in \( H_\eta \). Then we can define \( \alpha_T (\mathcal{C}) \) in terms of the set of \( \alpha_{T_\eta} (\mathcal{C}) \):

\[
\alpha_T (\mathcal{C}) = c \cup_{\eta \in [n]} \alpha_{T_\eta} (\mathcal{C}) \tag{3}
\]

Where \( e = h_c (\vec{f}_{q,j}, v_0) \) means that \( e \) can be uniquely identified as corresponding to \( \vec{f}_{q,j} \) since \( H \) and \( R(q) \) are isomorphic and \( v_0 \) is the root of \( T \). For the anchor set,

\[
\alpha_T (\mathcal{C}) = c \cup_{\eta \in [n]} \alpha_{T_\eta} (\mathcal{C}) \tag{3}
\]

where \( c = h_c (\vec{c}_{q,j}, v_0) \).

**4.1.4 RGLs Satisfy the Precondition**

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

\[
\rho_X (W) : \text{EDGE}_X (W) \land \text{SHARE}(W) \land \text{SUBGRAPH}(W)
\]

Define \( \rho(W) = \rho_S (W) \).

**Proposition 1.** Let \( G \) be an RGG and let \( G \in L(G) \), then for each derivation tree \( T \) of \( G \),
there exists a parameter assignment $\alpha_T$ such that $(G, \alpha_T) \models \rho(W)$.

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

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 $\rho(W)$ and next we define the domain and relation formulas. The domain formulas define the nodes of the derivation tree and so we write node$(x, \{W\})$. The relation formulas define which output node is the $i$th child of another output node, written child$(x, y, \{W\})$, and the labels of the output nodes, written lab$(x, \{W\})$.

The domain of the output for a parameter assignment $\alpha$ is $D_T$ where:

\[
D_T(\alpha) : \{ x \mid (G, x, \alpha) \models \text{node}(x, \{W\}) \}
\]

and node$(x, \{W\}) : x \in C$.

The relation formula child$(x, y, \{W\})$ defines the edges of the output of the transducer. We use the formula $\text{PAR}_{p,i,p,i'}(u, u', \{W\})$ 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 $i$th child of some other node).

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

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

\[
\text{lab}(x, \{W\}) : \bigvee_i x \in C_{i,p}
\]

Example 16. Figure 8 shows the output of the transducer when it takes $T$ as input with $\alpha$ 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 $\text{PAR}$ formulas, and the labels of the nodes, according to the $C_{i,p}$ sets.

We have now defined each part of the transducer $\tau$ from graphs to their derivation trees. Let $G$ be an RGG, and let $X \in N$. Then the corresponding transducer $\tau_X$ is

\[
(\rho_X(\{W\}), \text{node}(x, \{W\}), \text{lab}(x, \{W\}), \text{edge}(x, y, \{W\})) : \tau = \tau_S.
\]

For start symbol $S$ of $G$, let $\tau = \tau_S$. Let $G$ be a graph in $L(G)$, and let $\alpha$ be a parameter assignment such that $(G, \alpha) \models \rho(\{W\})$. Then the output of the transducer with respect to $\alpha$ is $\tau(G, \alpha) = (V_H, \text{lab}_H, \text{child}_H) : V_H \to P$ such that $\text{lab}_H(x) = p$ if $x \in \alpha(C_{i,p})$ for some $i$, and $\text{child}_H(x, y) : V_H \to V_H$ such that $\text{child}_H(x, y)$ if $(G, x, y, \alpha) \models \text{PAR}_{p,i,p,i'}(x, y, \{W\})$.

4.2.1 Transducer Output and Derivation Trees

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

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

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

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

Theorem 2. $RGL \subseteq MSOL$. 

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

110
Proof. Let $G$ be an RGG and $\tau$ be the corresponding transducer. By Propositions 2 and 3, for each $G \in L(G)$, $\tau(G)$ is a set which contains all of the derivation trees of $G$ and possibly other elements none of which are derivation trees of any $G' \in L(G)$ where $G' \neq G$. Therefore, for each $G \in L(G)$,

$$\tau(G) \cap T_G = \{T \in T_G \mid \text{VAL}(T) = G\}.$$ 

Therefore,

$$\tau(L(G)) \cap T_G = \{T \in T_G \mid \text{VAL}(T) = G, G \in L(G)\}.$$ 

And since $\{T \in T_G \mid \text{VAL}(T) = G, G \in L(G)\} = T_G$, 

$$\tau^{-1}(\tau(L(G)) \cap T_G) = \tau^{-1}(T_G).$$ 

$$\tau^{-1}(\tau(L(G)) \cap T_G) = \{G \in L(G) \mid \tau(G) \cap T_G \neq \emptyset\} = L(G).$$ 

Therefore,

$$L(G) = \tau^{-1}(T_G)$$

and so by Theorem 1 and the fact that $T_G$ is MSO definable, $L(G)$ is MSO definable.

\[\square\]

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](http://www.aclweb.org/anthology/W13-2322)

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


