Contextual Hyperedge Replacement Grammars for Abstract Meaning Representations

Frank Drewes
Umeå University
drewes@cs.umu.se

Anna Jonsson
Umeå University
aj@cs.umu.se

Abstract

We show how contextual hyperedge replacement grammars can be used to generate abstract meaning representations (AMRs), and argue that they are more suitable for this purpose than hyperedge replacement grammars. Contextual hyperedge replacement turns out to have two advantages over plain hyperedge replacement: it can completely cover the language of all AMRs over a given domain of concepts, and at the same time its grammars become both smaller and simpler.

1 Introduction

Natural language processing applications that receive sentences as input mainly make use of lexical and syntactic properties of the input sentences. Even though these properties are an important basis for the analysis of a sentence, one is usually more interested in the meaning of a sentence, i.e., its semantics. This is particularly true in the case of machine translation where a semantic error can cause far more bewilderment than a syntactic one.

Thus, a general-purpose formalism for modelling the semantics of sentences in a way that allows for efficient analysis would be widely useful in natural language processing. This study focuses on the generation of a semantic representation that was proposed some years ago, the abstract meaning representation (AMR) (Langkilde and Knight, 1998; Banarescu et al., 2013). An AMR[1] is a directed, rooted, acyclic, node- and edge-labelled graph that represents the semantics of an English sentence[2]; the nodes and edges represent concepts and their relations, respectively. A corpus of AMRs over a limited domain can be found in (Braune et al., 2014). As in the case of syntax trees, where tree grammars and tree automata (Knight and Graehl, 2005) provide a model for distinguishing structurally correct trees from incorrect ones, the algorithmic processing of AMRs would benefit from the existence of appropriate formal models for their generation or recognition. Here, we focus on the generation of AMRs by graph grammars, which have previously been proposed as formal models for this very task (Chiang et al., 2013).

The usefulness of two types of hyperedge replacement grammar (HRG, see Habel (1992); Drewes et al. (1997)) for AMR generation was investigated by Jonsson (2016a) (see also (Jonsson, 2016b)), namely the predictive top-down (PTD) parsable grammar (Drewes et al., 2015) and the restricted directed acyclic graph (rDAG) grammar (Björklund et al., 2016). Both are of particular interest because their study was, among other possible application areas, motivated by AMR generation. A specific advantage of these special cases of HRGs is that their membership problem is solvable in polynomial time. However, Jonsson (2016a) concludes that neither of them is able to generate the complete set of AMRs over a given concept domain.

Unrestricted HRGs allow for better coverage at the expense of greater computational complexity. However, a general disadvantage of hyperedge replacement remains. The nonterminal items in an HRG are hyperedges – edges that may be attached to more (or fewer) than two nodes. Replacement of a hyperedge inserts a new subgraph in its place,

[1]We use the term AMR to refer not only to the concept of Abstract Meaning Representation as such (Langkilde and Knight, 1998; Banarescu et al., 2013), but also to its individual graphs.

[2]Although AMR is to some extent language independent, it is biased towards English (Banarescu et al., 2013), and therefore not truly an interlingua (Xue et al., 2014).

connecting it to the host graph via the nodes the replaced hyperedge was incident on. Intuitively, nonterminal hyperedges keep track of a number of potentially relevant nodes for the purpose of being able to attach new edges to them later on in the derivation. This process is well known (and easily seen) to generate graph languages of bounded treewidth. As shall be illustrated in Section 6 the ability of hyperedges to keep track of a bounded number of previously generated nodes can be used to ensure structural properties such as those caused by control verbs. However, it appears that other types of reentrancies, like those arising from the use of pronouns, are of a different nature. If, for example, several instances of the concept `boy` have been generated, any of them can in principle be referred to from anywhere else in the AMR. As a consequence, there is no reasonable a priori bound on the treewidth of the graph. Nonterminal hyperedges generating other parts of the AMR would have to keep track of all `boy` instances to accomplish full coverage. On the one hand, this is not possible in an HRG. On the other hand, it does not seem to be desirable either, because keeping track of every `boy` instance individually would enable a level of control far beyond what is needed.

Here we consider contextual hyperedge replacement grammars (CHRGs) (Drewes et al., 2012; Drewes and Hoffmann, 2015) to learn whether they can be used to overcome these disadvantages. CHRGs are also based on hyperedge replacement, but the left-hand side of a rule can contain so-called contextual nodes. This provides access to nodes other than those immediately controlled by the nonterminal hyperedge, thus enabling rules to establish connections of the type discussed in the previous paragraph. The additional ability is severely limited, far below true context-sensitivity in power, because nodes are terminal items and derivation steps cannot distinguish between contextual nodes with the same label. For instance, in the situation sketched above a rule application would just pick *any* occurrence of `boy` elsewhere in the host graph. As a consequence, however, the treewidth of generated graphs is not necessarily bounded anymore.

In the present paper we study and illustrate the advantages of CHRGs over HRGs for AMR generation by looking at an example concept domain in a theoretical case study. To this end, we build a CHRG that generates AMRs over a restricted domain and argue that it exhibits perfect coverage. The baseline domain is the one introduced by Braune et al. (2014), consisting of the concepts `boy`, `girl`, `want` and `believe` along with two basic relations (called `arg0` and `arg1`) that are used to bind the concepts together and correspond to the agent and patient of a `want` or `believe` event. We also consider the construction of CHRGs for more general AMRs to explore the advantages of the more generous rule format. Therefore we add a small set of possible modifiers, allow an arbitrary number of boys and girls to appear in an AMR,[3] and discuss how to handle control verbs.

The conclusion of our study is that contextual hyperedge replacement is indeed a promising formalism for describing sets of AMRs. On the one hand, AMRs contain the mentioned local structures that must satisfy certain well-formedness constraints, such as in the case of control verbs. This can be implemented like it would in an HRG, using a nonterminal hyperedge to keep track of the involved nodes. On the other hand, contextual nodes can be used to implement the kinds of coreferences which may occur anywhere without following strict local rules, such as those relating to the use of pronouns. As discussed above, the latter creates problems in HRGs because nonterminal hyperedges would have to keep track of potential antecedents, which seems inappropriate for various reasons: it is restricted by the rank of hyperedges, provides an unnecessarily detailed level of control (thus creating the risk of overfitting), and leads to a huge number of rules to account explicitly for all the possible nondeterministic choices arising from the (exponentially) many ways in which coreferences can be inserted.

The obvious downside of using CHRGs is that computational problems may potentially become more difficult. However, recent results on shift-reduce parsing for both HRGs and CHRGs (Drewes et al., 2017)[4] indicate that this may not be the case. In fact, as the rank of hyperedges and the number of rules are central parameters in the complexity of membership algorithms for both unrestricted HRGs and CHRGs, it may even pay off to turn to CHRGs since this leads to smaller ranks and much fewer rules, the latter

[3]Braune et al. (2014) only consider at most one boy and at most one girl.

[4]See `https://www.unibw.de/inf2/grappa/` for the extension to CHRGs.

because the use of contextual nodes removes the necessity to implement nondeterministic choices explicitly by creating a separate rule for each.

In Section 2, we lay the ground for the rest of the paper with some basic definitions. The CHRG is defined in Section 3, and the subset of AMR to be considered here is discussed in Section 4. The construction of a CHRG for this domain is described in Section 5. In Section 6, we indicate how to generalise it to larger domains, and in particular how control verbs can be added. Finally, the results are discussed in Section 7 followed by the conclusions and future work in Section 8.

Acknowledgement We thank the reviewers for useful comments that helped us clarify the line of argumentation (as we hope).

2 Preliminaries

For a set A, we write A^* to denote the set of finite sequences or strings over A, and $A^{\circledast}$ for the set of strings over A in which no element is repeated; ε denotes the empty sequence. Elements of A are identified with strings of length 1 over A, and thus subsets of A are string languages at the same time.

Furthermore, we let 2^A denote the power set of A, i.e., the set of all subsets of A. The extension of a function $f\colon A \to A'$ to sequences $a_1, \ldots, a_n$ where $a_i \in A$ for $0 \leq i \leq n$ is denoted $f^*\colon A^* \to A'^*$ and defined by $f^*(a_0, \ldots, a_n) = f(a_1) \cdots f(a_n)$. Concatenation of strings is denoted by simple juxtaposition, and elementwise concatenation of two string languages L, L' is denoted by $L \cdot L'$, i.e., $L \cdot L' = \{uv \mid u \in L, v \in L'\}$.

A *labelling alphabet* is a set Σ partitioned into three mutually disjoint sets Σ_V, Σ_E and Σ_N on which an *arity* function $\mathit{arity}\colon \Sigma_E \uplus \Sigma_N \to 2^{\Sigma_V^*}$ is defined. (See Section 5 for an example of an alphabet and its arity function.) The sets Σ_V, Σ_E and Σ_N are referred to as *node labels*, *(hyper)edge labels* and *nonterminal labels*, respectively.

A *hypergraph* is a generalisation of directed graphs by the usage of edges that can connect an arbitrary number of nodes. Here, we consider node- and edge-labelled hypergraphs.

Definition 1 (Hypergraph (Drewes et al., 2012)). *A* labelled hypergraph (hypergraph, *for short) over a labelling alphabet* Σ *is a tuple* $G = (V, E, \mathit{att}, \mathit{label}_V, \mathit{label}_E)$ *such that*

- V *is a finite set of* nodes.
- E *is a finite set of* hyperedges.
- $\mathit{att} : E \to V^{\circledast}$ *is the* attachment of hyperedges.
- $\mathit{label}_V : V \to \Sigma_V$ *is the* labelling of nodes.
- $\mathit{label}_E : E \to \Sigma_E \cup \Sigma_N$ *with* $\mathit{label}^*_V(\mathit{att}(e)) \in \mathit{arity}(\mathit{label}_E(e))$ *for all* $e \in E$ *is the* labelling of hyperedges.[5]

The *rank* of a hyperedge e is $|\mathit{arity}(\mathit{label}_E(e))|$. Hyperedges with labels in Σ_N are called *nonterminals*; $\mathcal{G}_\Sigma$ denotes the set of all hypergraphs over Σ. For a hypergraph G and a hyperedge $e \in E$, the hypergraph resulting from removing e from G is denoted by $G - e$. The empty hypergraph is denoted by $()$.

In illustrations, nodes and hyperedges are drawn as ellipses and squares, respectively, with inscribed labels. The attachment of a hyperedge is shown by lines, and the attachment order is depicted using numbers (these can be left out if the attachment order is clear from the context or irrelevant). If a hyperedge connects exactly two nodes (i.e., it is binary), it can be drawn as an arrow directed from the first node of the attachment to the second with its label next to it. See Section 5 for various examples of hypergraphs. Note that a hypergraph containing only binary hyperedges is equivalent to an ordinary directed graph; this is the case in e.g. Figure 1.

3 Contextual Hyperedge Replacement

Given a hypergraph containing nonterminals, rules can be applied to it in order to generate a new hypergraph. A set of such rules along with a fixed hypergraph to which they are to be applied forms a grammar. The grammar type considered here was proposed in (Drewes et al., 2012; Drewes and Hoffmann, 2015) and uses the following rule type.

Definition 2 (Contextual Rule). *A* contextual hyperedge replacement rule *(or* contextual rule*) is a pair* (L, R) *where* L *and* R *are hypergraphs over the labelling alphabet* Σ *such that*

- L *(the* left-hand side*) contains exactly one hyperedge* e *that must be a nonterminal, and*
- R *(the* right-hand side*) is an arbitrary supergraph of* $L - e$.

A contextual rule for which all nodes in the left-hand side are connected to e is called *context-free*. The nodes that are not connected to e are referred to as *contextual nodes*.

[5]The arity function used differs from the one in (Drewes et al., 2012), but the resulting hypergraph definition remains the same.

We denote a contextual rule by letting $::=$ separate the left- and right-hand sides. Moreover, we allow rules that share the same left-hand side to be drawn more compactly; in this case, the left-hand side is only drawn once, and a vertical line is used to separate the right-hand sides from each other. To save further space, we use rule schemata in which labels may be variables ranging over a specified subset of the set of all labels. As an example of a set of contextual rules, consider the rules in (iii) in Figure 4 of Section 5. Every choice of z, u, v and a_1, a_2 in the range specified beneath the rules yields three rules. Each has the nonterminal N_1 in its left-hand side, and the node labelled u is a contextual node. In addition, the third right-hand side contains a newly generated node labelled v.

A contextual rule (L, R) can be applied to a hypergraph G containing an isomorphic copy of L, i.e., a subgraph that is equal to L up to renaming of nodes and hyperedges. Suppose for simplicity that L is a subgraph of G. Then the application of the rule works in the following manner:

1. Remove e from G, yielding $G - e$.
2. Add R to $G - e$, disjointly.
3. Identify the nodes in $L - e$ with the corresponding nodes in R.

The resulting hypergraph is denoted by $G[R/e]$. Now, we can formally define the grammar type that makes use of contextual rules.

Definition 3 (Contextual Hyperedge Replacement)**.** *A* contextual hyperedge replacement grammar (CHRG) *is a triple* $\Gamma = (\Sigma, \mathcal{R}, Z)$ *where*

- Σ *is a finite labelling alphabet,*
- $\mathcal{R}$ *is a finite set of contextual rules, and*
- $Z \in \mathcal{G}_\Sigma$ *is a start hypergraph.*

If $G' = G[R/e]$ for some contextual rule $(L, R) \in \mathcal{R}$, we say that G' is *derived* from G in Γ, and we write $G \Rightarrow_\mathcal{R} G'$. The language generated by Γ is $\mathcal{L}(\Gamma) = \{G \in \mathcal{G}_{\Sigma\setminus N} \mid Z \Rightarrow^*_\mathcal{R} G\}$ where $\Rightarrow^*$ is the reflective and transitive closure of $\Rightarrow$. Two CHRGs Γ_1 and Γ_2 are *equivalent* if $\mathcal{L}(\Gamma_1) = \mathcal{L}(\Gamma_2)$, i.e., if they generate the same language. If all of the rules of Γ are context-free, then Γ is a *hyperedge replacement grammar* (HRG). Thus, CHRG is a generalisation of HRG through the extension of context-free rules to contextual rules. Intuitively, the difference between the two is that CHRGs can nondeterministically pick a previously generated node with a specified label without that node being connected to the replaced nonterminal. HRGs do not have this ability.

The graph languages generated by CHRGs are in NP (Drewes and Hoffmann, 2015), and can thus be NP-complete, as this already holds for HRGs. Hence, unless P $=$ NP there are CHRGs which do not admit a polynomial membership test. For HRGs, there exist polynomial membership algorithms for nontrivial special cases such as PTD parsable, shift-reduce parsable, and rDAG HRGs. The fact that membership testing is not harder for CHRG than for HRG (at least in theory) strengthens the hope that there are subclasses of CHRG with efficient membership tests. Indeed, this has partially been confirmed: the membership algorithms for PTD and shift-reduce parsable HRGs can be extended to CHRGs.[4]

4 Abstract Meaning Representation

Abstract meaning representation (AMR) (Langkilde and Knight, 1998; Banarescu et al., 2013) denotes sentence meaning as directed, rooted, acyclic graphs with node and edge labels. To the extent possible, AMR aims to provide a unique representation of semantics, i.e., while numerous sentences can express the same meaning, they should all map to the same AMR. The idea is that the nodes of the graph represent the concepts identifiable in the sentence, and the edges represent the relations between the concepts. Intuitively, the subgraph rooted at any one given node represents an event, a fact, or an entity. See Figure 1 for example AMRs that can be realised into the English sentences “The boy wants the girl to believe him” or “[. . .] to believe the other boy.”

The previous example highlights that every event or entity represented in an AMR should occur once and only once. In fact, this is the major difference between AMRs and syntax trees, in which several subtrees may refer to the same semantic thing. In the second AMR in Figure 1, the fact that the wantee is not represented by the same node as the believee implies that these two are distinct. Representing the first sentence by the second AMR (or the second one by the first) is an error.

Thus, to achieve complete coverage, a grammar for generating AMRs over the given domain must generate both graphs in Figure 1. Figures 2 and 3 show another pair of AMRs, of which the former correctly represents the semantics of the sentence “The boy wants the girl to believe in herself and this is what the girl wants, too.” The interpretation of the latter is less obvious. We do not endeav-

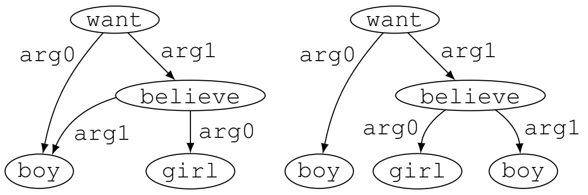


Figure 1: AMRs representing an event `want`, where the wanter is a `boy` and the wantee is the event `believe` for which the believer is a `girl` and the believee is either the formerly mentioned `boy` (left) or a different one (right).

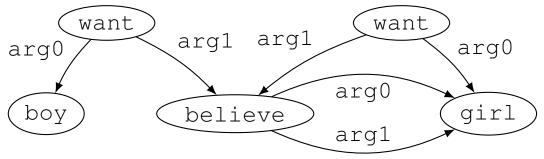


Figure 2: Another AMR.

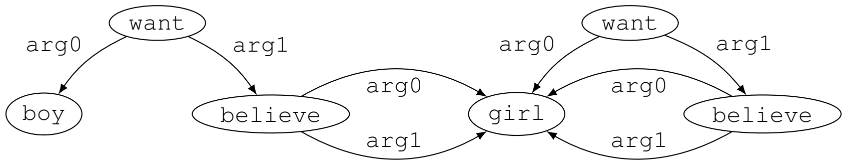


Figure 3: An AMR similar to the one in Figure 2, but with two distinct `believe` events.

our to discuss whether this AMR is meaningful at all, but it certainly seems to be less probable. Unfortunately, it turns out that structures such as the one in Figure 3 are easy to generate by a HRG and even by the aforementioned PTD parsable HRGs, whereas trying to include the more desirable one in Figure 2 meets severe difficulties. This is an instance of the problems mentioned in the introduction: a HRG generating structures like the one in Figure 2 (even one that is not PTD parsable) would have to generate the `believe` node early on and then keep track of it in its nonterminal hyperedges to establish the desired relations later on, when the two `want` nodes are generated.[6] The nondeterministic choices this creates seem to destroy PTD parsability. Further, even if PTD parsability is abandoned in favour of generative power, the desired effect can only be approximated: as the number of `believe` nodes grows, it eventually exceeds the number of nodes that the nonterminal hyperedges have been designed to keep track of.

4.1 The Boy-Girl AMR Corpus

The *boy-girl AMR corpus* is a set of 10 000 AMRs over a restricted domain that was presented in (Braune et al., 2014). Each AMR of this corpus fulfils the following conditions:

- The node label alphabet consists of the concept names `boy`, `girl`, `want`, and `believe`.
- The edge label alphabet consists of the relation names `arg0` and `arg1`.
- The node labels `boy` and `girl` occur at most once each, and label the leaves of the graph.
- For each `want` and `believe` node, the outgoing edges carry distinct labels and all incoming edges are labelled `arg1`.

The relation `arg0` is used for marking the agent of an action expressed by a concept in the form of a verb, and the patient of the same action is given by the concept pointed to by `arg1`. The above restrictions simply give us the domain and tell us that a person cannot be used as a verb, and that verbs cannot be agents, but that an event (a subgraph with a verb concept as root) can act as a patient. The left AMR in Figure 1 is a boy-girl AMR, whereas the left one is not, as it contains two copies of `boy`.

To make things more interesting, we remove the restriction that there can only be one girl and one boy, and extend the concept domain by months, weekdays and the words `happy` and `angry`. Let Σ_V denote this extended domain. Finally, we add the relations `manner`, `month` and `day`, which together with `arg0` and `arg1` form Σ_E.

5 Construction of a Boy-Girl CHRG

Let us now discuss how to construct a CHRG that generates the complete language of boy-girl AMRs. The alphabet used is that of Section 4.1, enlarged by $\Sigma_N = \{S, N, N_1, M\}$ and with the

[6]This assumes for simplicity that a bottom-up generation strategy is employed. However, the difficulties arising depend only marginally on the choice of strategy.

arity function given as follows: for $A \in \Sigma_E$, $\mathit{arity}(A) = \{\texttt{want}, \texttt{believe}\} \cdot \mathit{TAR}_A$ where

$$\begin{aligned}
\mathit{TAR}_{\texttt{arg0}} &= \{\texttt{boy}, \texttt{girl}\}\\
\mathit{TAR}_{\texttt{arg1}} &= \Sigma_V \setminus \{\texttt{happy}, \texttt{angry}\}\\
\mathit{TAR}_{\texttt{manner}} &= \{\texttt{happy}, \texttt{angry}\}\\
\mathit{TAR}_{\texttt{month}} &= \{\texttt{Jan}, \ldots, \texttt{Dec}\}\\
\mathit{TAR}_{\texttt{day}} &= \{\texttt{Mon}, \ldots, \texttt{Sun}\}.
\end{aligned}$$

Furthermore, $\mathit{arity}(S) = \mathit{arity}(N) = \varepsilon$ and $\mathit{arity}(N_1) = \mathit{arity}(M) = \{\texttt{want}, \texttt{believe}\}$. The start hypergraph Z consists of a single nonterminal labelled S. The rules of the boy-girl CHRG can be seen in Figure 4.

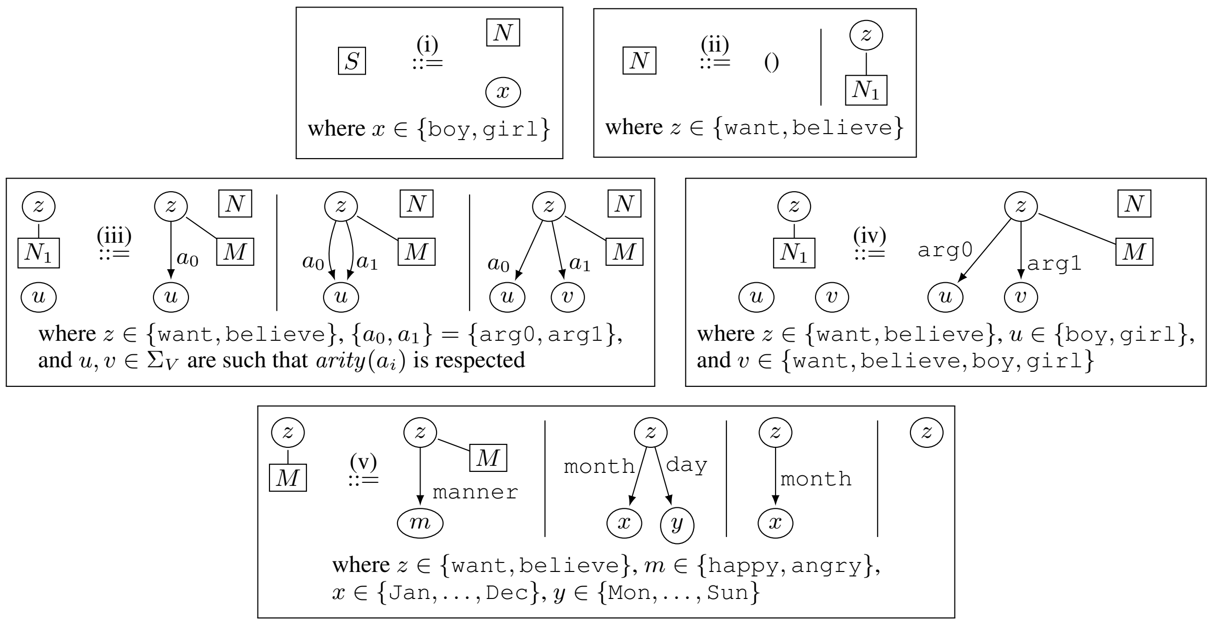
The initial rules of the grammar, the ones of schema (i), simply generate the first leaf of the graph. The rules of schema (ii) choose between terminating the derivation by generating the empty graph or continuing it by generating a non-leaf node. Schemata (iii) and (iv) connect the newly generated node with label z to at least one previously generated node. Moreover, these rules connect a nonterminal labelled M to the node, which makes it possible to add zero or more outgoing `manner` edges from the node currently being handled to suitable (new) leaves. In addition, at most one `month` and one `day` edge can be generated, and the latter only in connection with the former. We note here that these restrictions are not intended to be semantically particularly meaningful. They only serve to illustrate that this type of "regular control" can be used to put together the combination of outgoing relations a node shall have.

To restart the cycle of either generating another `want` or `believe` node or terminating the derivation, (iii) and (iv) also create a new nonterminal labelled N.

We can see that each node must be given all of its outgoing `arg0` and `arg1` edges before another one is generated, making sure that the resulting AMR is acyclic (because `manner`, `month`, and `day` edges only point to leaves). Every node generated by (iii), (iv), or (v) is immediately connected to an already existing node. Moreover, the new node generated by the second rule of (ii) is connected to a nonterminal labelled N_1 until that node, by (iii) or (iv), is connected to an older node. Using this, it follows by induction that only connected graphs are generated.

An example of a derivation using the boy-girl CHRG can be seen in Figure 5. The rule(s) used in every step are indicated above the derivation symbol ($\Rightarrow$) combined with the right-hand side index of the used rule (starting at 1). What variables are mapped to which labels throughout the derivation is shown implicitly. The resulting AMR is the previously discussed one in Figure 2.

It should be clear that this grammar generates the complete language of AMRs over our small domain: as we are only interested in generating acyclic graphs this is always possible by generat-


S
(i) ::=
N
x
where $x \in \{\texttt{boy}, \texttt{girl}\}$
N
(ii) ::=
()
z
N_1
where $z \in \{\texttt{want}, \texttt{believe}\}$
z
N_1
u
(iii) ::=
a_0
a_1
M
N
v
where $z \in \{\texttt{want}, \texttt{believe}\}$, $\{a_0, a_1\} = \{\texttt{arg0}, \texttt{arg1}\}$, and $u, v \in \Sigma_V$ are such that $\mathit{arity}(a_i)$ is respected
(iv) ::=
arg0
arg1
where $z \in \{\texttt{want}, \texttt{believe}\}$, $u \in \{\texttt{boy}, \texttt{girl}\}$, and $v \in \{\texttt{want}, \texttt{believe}, \texttt{boy}, \texttt{girl}\}$
(v) ::=
manner
m
month
day
x
y
where $z \in \{\texttt{want}, \texttt{believe}\}$, $m \in \{\texttt{happy}, \texttt{angry}\}$, $x \in \{\texttt{Jan}, \ldots, \texttt{Dec}\}$, $y \in \{\texttt{Mon}, \ldots, \texttt{Sun}\}$
Figure 4: A boy-girl CHRG exemplifying general rule structures. Rules are named for later reference by a superscript on the operator '::='.

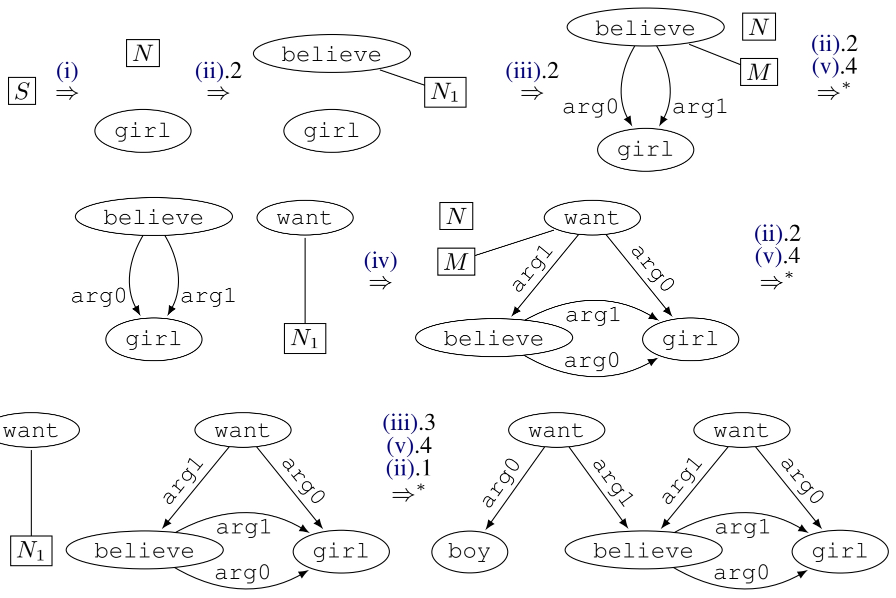


Figure 5: A derivation of an AMR using the boy-girl CHRG.

ing the nodes in reverse topological order. In other words, the CHRG constructed indeed generates the complete AMR language described above.

6 Generalisations

Let us now formulate some general rules about how to create a CHRG that generates AMRs over a given, finite domain of concepts and relations.

Let Σ_V contain the concept names of the domain and Σ_E its relations – these can be any sets as long as they are finite. Define the arity function of Σ as $\mathit{arity}(r) = \{c_i c_j \mid r \text{ is a valid relation from } c_i \text{ to } c_j \text{ for } c_i, c_j \in \Sigma_V\}$. The arity function is used to restrict which concepts can be connected using a particular relation. (For example, in the boy-girl case, we know that verbs cannot be agents and that persons cannot have agents. Thus, want boy and want girl are allowed in $\mathit{arity}(\texttt{arg0})$, but not want believe or girl want.)

As in the boy-girl CHRG, generation starts with the base case – a single leaf nondeterministically chosen from all the concept names that may appear as leaves. A nonterminal similar to N in the boy-girl case generates one new non-leaf node at a time. All of the outgoing edges to other non-leaf nodes are generated before returning to N. This guarantees acyclicity as it prevents nodes from being given outgoing edges to nodes generated posterior to them. As in the previous section, contextual rules are thus used to (1) enable the generation to refer back to previously generated nodes by adding incoming relations and to (2) make sure that the AMRs are connected. Further leaves can only be generated along with the generation of an outgoing relation from another node.

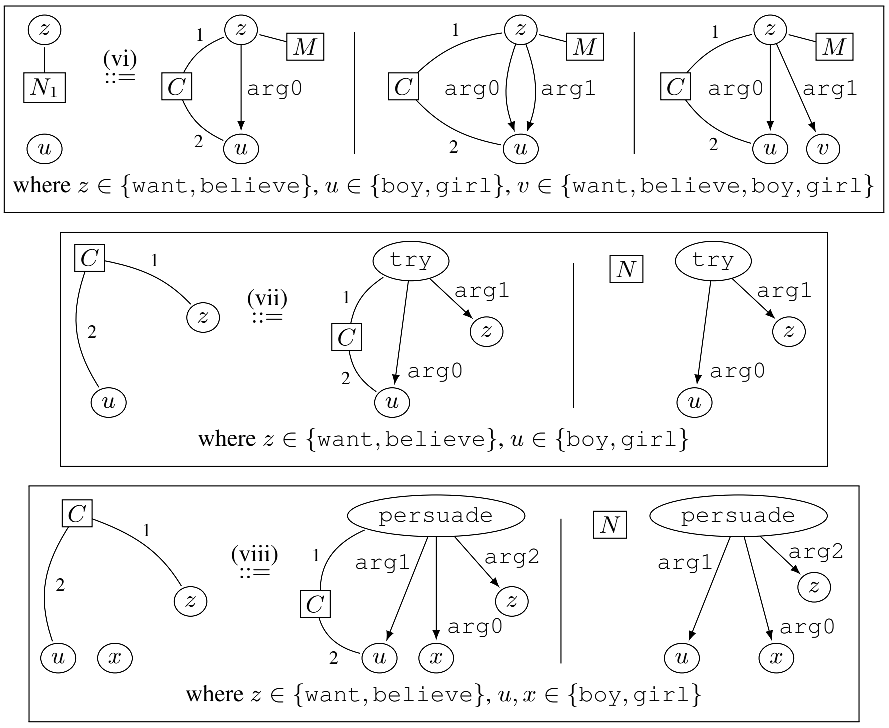
We may also want for a CHRG to generate various combinations of outgoing relations from the latest non-leaf node (in the boy-girl grammar represented as z). This can be done similarly to the generation of manner, month, and day edges by M in Figure 4.

In view of the previous discussion the reader may wonder whether one will ever have the need to use nonterminal labels A with $|\mathit{arity}(A)| > 1$. It might seem that arguments can always be picked using contextual nodes. However, this selects targets exclusively based on their labels and is thus inappropriate if finer structural control is required. To illustrate this, let us add the object control verb persuade and the subject control verb try to our concept set (i.e., to Σ_V). We also need a new relation arg2 to connect an occurrence of persuade to its indirect object, i.e., $\mathit{arity}(\texttt{arg2}) = \{\texttt{persuade}\} \cdot \Sigma_{\text{verb}}$ where $\Sigma_{\text{verb}} = \{\texttt{want}, \texttt{believe}, \texttt{persuade}, \texttt{try}\}$.

Recall that, whenever an arg0 edge is created in one of the rules in Figure 4, the subsequent creation of further want and believe nodes is taken care of by a nonterminal N generated at the same time. To implement control, we use variants of these rules which, instead of N, use a nonterminal C with $\mathit{arity}(C) = \Sigma_{\text{verb}} \cdot \{\texttt{boy}, \texttt{girl}\}$. This nonterminal is attached to the two nodes of the arg0 edge, thus remembering where the control should take place instead of floating freely.

Some of the new rules are illustrated in Figure 6. The rules in (vi) work like those in (iii), but create nonterminals labelled by C instead of N, in the way just described. A similar rule obtained from (iv) is left out to save space.

The remaining rules insert the control verbs: those in (vii) implement subject control by try whereas those in (viii) implement object control by persuade. Each of the rules corresponds to a succession of two rules in Figure 4, namely (iii).1 followed by rule (ii).2. The first of each pair of rules initiates another level of control whereas the second returns to the "uncontrolled" case. Note that we, for simplicity, drop the nonterminals M that should be attached to the control verbs to follow (iii).1 strictly. Also, there should be further rules corresponding to (iii).2, (iii).3, and (iv), which are omitted because they are constructed along the same lines as those shown in the figure.



(vi) ::=
(vii) ::=
(viii) ::=
arg0
arg1
arg2
try
persuade
where $z \in \{$want, believe$\}$, $u \in \{$boy, girl$\}$, $v \in \{$want, believe, boy, girl$\}$
where $z \in \{$want, believe$\}$, $u \in \{$boy, girl$\}$
where $z \in \{$want, believe$\}$, $u, x \in \{$boy, girl$\}$

Figure 6: Rules implementing subject control (vii) and object control (viii).

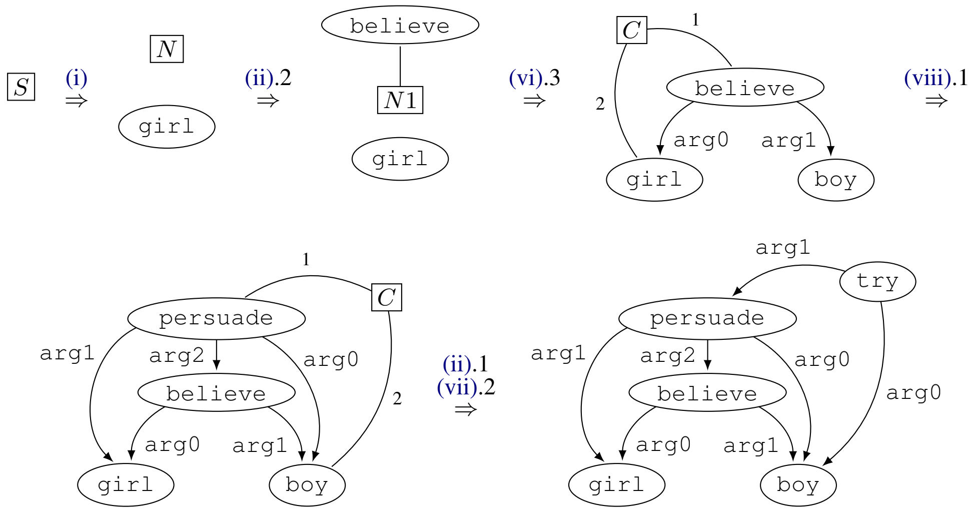


Figure 7: An example derivation using the rules in Figures 4 and 6.

Figure 7 shows an example derivation involving the new rules, the sentence being "The boy tries to persuade the girl to believe him." The reader may wish to add the remaining rules not shown in Figure 6, so that AMRs for sentences such as "The boy wants to persuade the girl to try to persuade the other girl to believe him" can be generated.

7 Discussion

Being able to generate complete AMR languages is a clear advantage of CHRGs compared to PTD parsable and rDAG grammars, and even over unrestricted HRGs. The latter can only generate graph languages of bounded treewidth, and despite the fact that real-world AMRs usually seem to be of small treewidth (Chiang et al., 2016) it does not seem to be justified to impose an a priori upper bound on their treewidth.

However, the advantage of CHRGs for modelling AMR languages exceeds the formal aspect of unlimited treewidth. In an HRG, nonterminal hyperedges have to keep track of all nodes to which edges shall (potentially) be attached later on in the process. This includes the implementation of non-local phenomena like anaphora, for which little if any structural control is required, thus resulting in an artificial increase not only in the rank of hyperedges but also in the number of rules. The latter may be significant, even exponential in the number of additional nodes to be kept track of, because in a right-hand side every nonterminal hyperedge would nondeterministically have to choose a subset of additional nodes to attach to. Even so, the number of nodes that can be kept track of is restricted by a constant depending on the rank of hyperedges. In contrast, CHRGs do not need to carry around such additional information at all as they can simply view antecedents as contextual nodes when it is time to insert a reference. The finer control provided by nonterminal hyperedges can be reserved for situations in which structural requirements must be met, such as implementing control verbs, quantifiers, and the like.

It remains to be seen whether the algorithmic properties of AMR-generating CHRGs are sufficiently good, especially when compared to HRGs. Since CHRGs generalise HRGs one may expect them to be algorithmically more demanding. However, the preceding discussion indicates that the converse may be true in practice. The efficiency of algorithms for analysing graphs with respect to a given (C)HRG depends most significantly on two things: the ranks of hyperedges and the number of rules (see in particular Chiang et al. (2013)). Thus, the greater algorithmic complexity of CHRGs may very well turn out to be outweighed by them requiring much smaller ranks and fewer rules, because the difference in size, as indicated above, will most likely be exponential in the desired number of potential antecedents.

The fact that CHRGs allow for structurally simpler rules may also make it possible to cast CHRGs like the one discussed here into a special form suitable for efficient analysis like shift-reduce parsing (Drewes et al., 2017) whereas the same may happen to be impossible for an HRG, even though the former has better coverage than the latter. Whether these possibilities can be realised is a question to be addressed by future work.

8 Conclusion

We have shown how CHRGs can generate complete AMR languages in cases where HRGs fail to do so because they do not provide appropriate means for the implementation of arbitrary coreferences. Whether CHRGs can generate complete AMR languages over arbitrary concept domains, including phenomena such as quantification while excluding structurally incorrect graphs, remains to be studied. In any case, the simplicity of the grammar discussed here seems to be promising. Future work, should investigate how efficiently problems such as the membership problem can be solved in practise for AMR-generating CHRGs. In this context, a better understanding of how quickly such CHRGs grow with the size of the input domain would also be valuable. If CHRGs indeed turn out to be a suitable device for AMR generation, a long-term goal should be to define a weighted version of CHRGs and to devise machine learning methods that make it possible to extract rule weights or even entire grammars from AMRbank.

Finally, it should be mentioned that there are other formalisms for defining languages of directed acyclic graphs that seem promising and should therefore be investigated for AMR modelling, e.g. DAG automata (Blum and Drewes, 2016, 2017; Chiang et al., 2016). In particular, it would be interesting to study the relative advantages and disadvantages of these options, and whether they can be combined in a fruitful way.

References

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*. pages 178–186.

Henrik Björklund, Frank Drewes, and Petter Ericson. 2016. Between a rock and a hard place — parsing for hyperedge replacement DAG grammars. In *10th International Conference on Language and Automata Theory and Applications*.

Johannes Blum and Frank Drewes. 2016. Properties of regular dag languages. In *10th International Conference on Language and Automata Theory and Applications*.

Johannes Blum and Frank Drewes. 2017. Language theoretic properties of regular DAG languages. To appear.

Fabienne Braune, Daniel Bauer, and Kevin Knight. 2014. Mapping between English strings and reentrant semantic graphs. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*. pages 4493–4498.

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)*. pages 924–932.

David Chiang, Frank Drewes, Daniel Gildea, Adam Lopez, and Giorgio Satta. 2016. Weighted DAG automata for semantic graphs. Submitted.

Frank Drewes and Berthold Hoffmann. 2015. Contextual hyperedge replacement. *Acta Informatica* 52(6):497–524.

Frank Drewes, Berthold Hoffmann, and Mark Minas. 2012. *Applications of Graph Transformations with Industrial Relevance: 4th International Symposium, Revised Selected and Invited Papers*, chapter Contextual Hyperedge Replacement, pages 182–197.

Frank Drewes, Berthold Hoffmann, and Mark Minas. 2015. Predictive top-down parsing for hyperedge replacement grammars. In *Proceedings of the 8th International Conference on Graph Transformation*. pages 19–34.

Frank Drewes, Berthold Hoffmann, and Mark Minas. 2017. Predictive shift-reduce parsing for hyperedge replacement grammars. In *Proc. 10th Intl. Conf. on Graph Transformation (ICGT'17)*. Lecture Notes in Computer Science.

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

Annegret Habel. 1992. *Hyperedge replacement: grammars and languages*, volume 643. Springer Science & Business Media.

Anna Jonsson. 2016a. *Generation of Abstract Meaning Representations by Hyperedge Replacement Grammars – A Case Study*. Master's thesis.

Anna Jonsson. 2016b. On the generation of abstract meaning representations using polynomial-time parsable hyperedge replacement grammars. The Sixth Swedish Language Technology Conference.

Kevin Knight and Jonathan Graehl. 2005. An overview of probabilistic tree transducers for natural language processing. In *International Conference on Intelligent Text Processing and Computational Linguistics*. pages 1–24.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*. pages 704–710.

Nianwen Xue, Ondrej Bojar, Jan Hajic, Martha Palmer, Zdenka Uresova, and Xiuhong Zhang. 2014. Not an interlingua, but close: Comparison of English AMRs to Chinese and Czech. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*. pages 1765–1772.