Conservative Ambiguity Detection in Context-Free Grammars

Sylvain Schmitz

Laboratoire I3S, Université de Nice - Sophia Antipolis & CNRS, France
schmitz@i3s.unice.fr

Abstract. The ability to detect ambiguities in context-free grammars is vital for their use in several fields, but the problem is undecidable in the general case. We present a safe, conservative approach, where the approximations cannot result in overlooked ambiguous cases. We analyze the complexity of the verification, and provide formal comparisons with several other ambiguity detection methods.

1 Introduction

Syntactic ambiguity allows a sentence to have more than one syntactic interpretation. A classical example is the sentence "She saw the man with a telescope.", where the phrase "with a telescope" can be associated to "saw" or to "the man". The presence of ambiguities in a context-free grammar (CFG) can severely hamper the reliability or the performance of the tools built from it. Sensitive fields, where CFGs are used to model the syntax, include for instance language acquisition [1], RNA analysis [2,3], controlled natural languages [4], or programming languages [5,6,7].

While proven undecidable [8,9], the problem of testing a context-free grammar for ambiguity can still be tackled approximatively. The approximations may result in two types of errors: *false negatives* if some ambiguities are left undetected, or *false positives* if some detected "ambiguities" are not actual ones.

In this paper, we present a framework for the conservative detection of ambiguities, only allowing false positives. Our general approach is that of the verification of an infinite system: we build a finite approximation of the grammar (Section 3) and check for ambiguities in this abstract structure (Section 4). The driving purpose of the paper is to establish the following theoretical results:

- An approximation model for CFGs, based on the quotienting of a graph of all the derivation trees of the grammar, which we call its *position graph*, into a nondeterministic finite automaton (NFA) (Section 3.2).
- The soundness of the verification we run on the resulting NFA. Although the ambiguity of our NFA is already a conservative test for ambiguities in the original grammar (Section 4.1), our verification improves on this immediate approach by ignoring some spurious paths (Section 4.2). The complexity of

L. Arge et al. (Eds.): ICALP 2007, LNCS 4596, pp. 692–703, 2007.

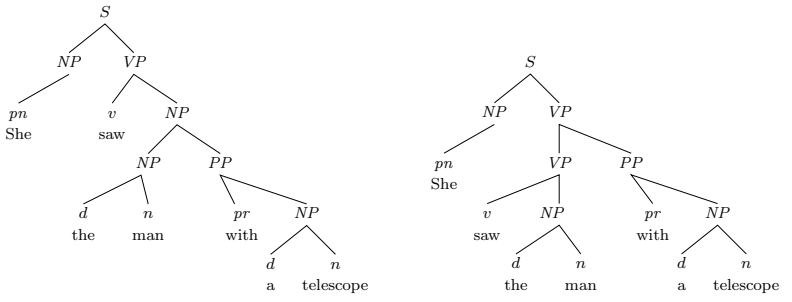

Fig. 1. Two trees yielding the sentence "She saw the man with a telescope." with $\mathcal{G}_1$.

the algorithm is bounded by a quadratic function of the size of our NFA (Section 4.4).

– Formal comparisons with several ambiguity checking methods: the bounded-length detection schemes [10,1,6,11] (which are not conservative tests), the LR-Regular condition [12], and the horizontal and vertical ambiguity condition [3] (Section 5); these comparisons rely on the generality of our approximation model.

The proofs of our results can be found in a companion research report [13], and we report on the experimental results of a prototype implementation of our algorithm in a different publication [7]. Let us proceed with an overview of our approach to ambiguity detection in the coming section.

2 Outline

Ambiguity in a CFG is characterized as a property of its derivation trees: if two different derivation trees yield the same sentence, then we are facing an ambiguity. Considering again the classical ambiguous sentence "She saw the man with a telescope.", a simple English grammar $\mathcal{G}_1 = \langle N, T, P, S \rangle$ that presents this ambiguity could have the rules in P

$$S \to NP\ VP,\ NP \to d\ n \mid pn \mid NP\ PP,\ VP \to v\ NP \mid VP\ PP,\ PP \to pr\ NP, \quad (\mathcal{G}_1)$$

where the nonterminals in N, namely S, NP, VP, and PP, stand respectively for a sentence, a noun phrase, a verb phrase, and a preposition phrase, whereas the terminals in T, namely d, n, v, pn, and pr, denote determinants, nouns, verbs, pronouns, and prepositions.[1] The two interpretations of our sentence are mirrored in the two derivation trees of Figure 1.

[1] We denote in general terminals in T by a, b, ..., terminal strings in T^* by u, v, ..., nonterminals by A, B, ..., symbols in $V = T \cup N$ by X, Y, ..., strings in V^* by α, β, ..., and rules in P by i, j or by indices 1, 2, ...; ε denotes the empty string, and $k : x$ the prefix of length k of the string x.

2.1 Bracketed Grammars

Tree structures are easier to handle in a flat representation, where the structural information is described by a bracketing [14]: each rule $i = A \xrightarrow{i} \alpha$ of the grammar is surrounded by a pair of opening and closing brackets d_i and r_i.

Formally, our *bracketed grammar* of a context-free grammar $\mathcal{G} = \langle N, T, P, S \rangle$ is the context-free grammar $\mathcal{G}_b = \langle N, T_b, P_b, S \rangle$ where $T_b = T \cup T_d \cup T_r$ with $T_d = \{d_i \mid i \in P\}$ and $T_r = \{r_i \mid i \in P\}$, and $P_b = \{A \xrightarrow{i} d_i \alpha r_i \mid A \xrightarrow{i} \alpha \in P\}$. We denote derivations in $\mathcal{G}_b$ by $\Rightarrow_b$. We define the homomorphism h from V_b^* to V^* by $h(d_i) = \varepsilon$ and $h(r_i) = \varepsilon$ for all i in P, and $h(X) = X$ otherwise, and denote by δ_b (resp. w_b) a string in V_b^* (resp. T_b^*) such that $h(\delta_b) = \delta$ (resp. $h(w_b) = w$).

Using the rule indices as subscripts for the brackets, the two trees of Figure 1 are represented by the following two sentences of the bracketed grammar for $\mathcal{G}_1'$:[2]

$$d_1\ d_2\ d_4\ pn\ r_4\ d_6\ v\ d_5\ d_3\ d\ n\ r_3\ d_8\ pr\ d_3\ d\ n\ r_3\ r_8\ r_5\ r_6\ r_2\ \$\ r_1 \tag{1}$$

$$d_1\ d_2\ d_4\ pn\ r_4\ d_7\ d_6\ v\ d_3\ d\ n\ r_3\ r_6\ d_8\ pr\ d_3\ d\ n\ r_3\ r_8\ r_7\ r_2\ \$\ r_1. \tag{2}$$

The existence of an ambiguity can be verified by checking that the image of these two different sentences by h is the same string $pn\ v\ d\ n\ pr\ d\ n$.

2.2 Super Languages

In general, an ambiguity in a grammar $\mathcal{G}$ is thus the existence of two different sentences w_b and w_b' of $\mathcal{G}_b$ such that $w = w'$. Therefore, we can design a conservative ambiguity verification if we approximate the language $\mathcal{L}(\mathcal{G}_b)$ with a super language and look for such sentences in the super language.

There exist quite a few methods that return a regular superset of a context-free language [15]; we present in the next section a very general model for such approximations. We can then verify on the NFA we obtain whether the original grammar might have contained any ambiguity. In Section 4, we exhibit some shortcomings of regular approximations, and present how to compute a more accurate context-free super language instead.

3 Position Graphs and Their Quotients

3.1 Position Graph

Let us consider again the two sentences (1) and (2) and how we can read them step by step on the trees of Figure 1. This process is akin to a left to right walk in the trees, between *positions* to the immediate left or immediate right of a tree node. For instance, the dot in

$$d_1\ d_2\ d_4\ pn\ r_4\ d_6\ v\ d_5\ d_3\ d\ n\ r_3 \bullet d_8\ pr\ d_3\ d\ n\ r_3\ r_8\ r_5\ r_6\ r_2\ \$\ r_1 \tag{3}$$

identifies a position between NP and PP in the middle of the left tree of Figure 1.

[2] The *extended* version $\mathcal{G}' = \langle N', T', P', S' \rangle$ of a CFG $\mathcal{G} = \langle N, T, P, S \rangle$ adds a new start symbol S' to N, an end of sentence symbol \$ to T, and a new rule $S' \xrightarrow{1} S\$$ to P.

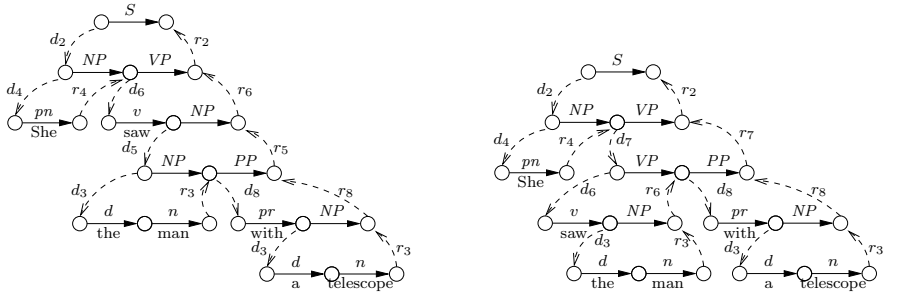

Fig. 2. Portions of the position graph of $\mathcal{G}_1$ corresponding to the two trees of Figure 1

Transitions from one position to the other can then be performed upon reading the node label, upon deriving from this node, or upon returning from such a derivation. We have thus three types of transitions: symbol transitions $\overset{X}{\rightarrowtail}$, derivation transitions $\overset{d_i}{\rightarrowtail}$, and reduction transitions $\overset{r_i}{\rightarrowtail}$, where i is a rule number. The set of all these positions in all parse trees along with the transition relation is a *position graph*. Figure 2 presents two portions of the position graph for $\mathcal{G}_1$; the position identified by the dot in (3) is now a vertex in the left graph.

Although a dotted sentence of $\mathcal{G}_b$ like (3) suffices to identify a unique position in the derivation tree for that sentence, it is convenient to know that this position is immediately surrounded by the NP and PP symbols. We therefore denote by $x_b d_i(\begin{smallmatrix}\alpha\\ u_b\end{smallmatrix} \bullet \begin{smallmatrix}\alpha'\\ u'_b\end{smallmatrix}) r_i x'_b$ the position identified by $x_b d_i u_b \bullet u'_b r_i x'_b$ such that the derivations

$$S' \Rightarrow_b^* x_b A x'_b \overset{i}{\Rightarrow}_b x_b d_i \alpha\alpha' r_i x'_b,\ \alpha \Rightarrow_b^* u_b \text{ and } \alpha' \Rightarrow_b^* u'_b \tag{4}$$

hold in $\mathcal{G}'_b$. Using this notation, the position identified by (3) is denoted by

$$d_1\ d_2\ d_4\ pn\ r_4\ d_6\ v\ d_5(\begin{smallmatrix}NP\\ d_3\ d\ n\ r_3\end{smallmatrix} \bullet \begin{smallmatrix}PP\\ d_8\ pr\ d_3\ d\ n\ r_3\ r_8\end{smallmatrix}) r_5\ r_6\ r_2\ \$\ r_1. \tag{5}$$

Definition 1. *The* position graph $\Gamma = \langle \mathcal{N}, \rightarrowtail \rangle$ *of a grammar* $\mathcal{G}$ *associates the set* $\mathcal{N}$ *of positions with the relation* $\rightarrowtail$ *labeled by elements of* V_b*, defined by*

$$x_b d_i(\begin{smallmatrix}\alpha\\ u_b\end{smallmatrix} \bullet \begin{smallmatrix}X\alpha'\\ v_b u'_b\end{smallmatrix}) r_i x'_b \overset{X}{\rightarrowtail} x_b d_i(\begin{smallmatrix}\alpha X\\ u_b v_b\end{smallmatrix} \bullet \begin{smallmatrix}\alpha'\\ u'_b\end{smallmatrix}) r_i x'_b \qquad \textit{iff } X \in V, X \Rightarrow_b^* v_b,$$

$$x_b d_i(\begin{smallmatrix}\alpha\\ u_b\end{smallmatrix} \bullet \begin{smallmatrix}B\alpha'\\ v_b u'_b\end{smallmatrix}) r_i x'_b \overset{d_j}{\rightarrowtail} x_b d_i u_b d_j(\bullet \begin{smallmatrix}\beta\\ v_b\end{smallmatrix}) r_j u'_b r_i x'_b \qquad \textit{iff } B \overset{j}{\rightarrow} \beta \textit{ and } \beta \Rightarrow_b^* v_b,$$

$$x_b d_i u_b d_j(\begin{smallmatrix}\beta\\ v_b\end{smallmatrix} \bullet) r_j u'_b r_i x'_b \overset{r_j}{\rightarrowtail} x_b d_i(\begin{smallmatrix}\alpha B\\ u_b v_b\end{smallmatrix} \bullet \begin{smallmatrix}\alpha'\\ u'_b\end{smallmatrix}) r_i x'_b \ \textit{ iff } B \overset{j}{\rightarrow} \beta, \alpha \Rightarrow_b^* u_b \textit{ and } \alpha' \Rightarrow_b^* u'_b.$$

We label paths in Γ by the sequences of labels on the individual transitions. We denote the two sets of positions at the beginning and end of the sentences by $\mu_s = \{d_1(\begin{smallmatrix}S\$\\ w_b\$\end{smallmatrix} \bullet) r_1 \mid S \Rightarrow_b^* w_b\}$ and $\mu_f = \{d_1(\begin{smallmatrix}S\\ w_b\end{smallmatrix} \bullet \begin{smallmatrix}\$\\ \$\end{smallmatrix}) r_1 \mid S \Rightarrow_b^* w_b\}$. For each sentence w_b of $\mathcal{G}_b$, a ν_s in μ_s is related to a ν_f in μ_f by $\nu_s \overset{S}{\rightarrowtail} \nu_f$.

The parsing literature classically employs *items* to identify positions in grammars; for instance, $[NP \xrightarrow{5} NP \bullet PP]$ is the LR(0) item [16] corresponding to position (5). There is a direct connection between these two notions: items can be viewed as equivalence classes of positions—a view somewhat reminiscent of the tree congruences of Sikkel [17].

3.2 Position Equivalences

In order to look for ambiguities in our grammar, we need a finite structure instead of our infinite position graph. This is provided by an equivalence relation between the positions of the graph, such that the equivalence classes become the states of a nondeterministic automaton.

Definition 2. *The* nondeterministic position automaton $\Gamma/\equiv$ *of a context-free grammar* $\mathcal{G}$ *using the equivalence relation* $\equiv$ *is a tuple* $\langle Q, V_b', R, q_s, \{q_f\}\rangle$ *where*

- $Q = [\mathcal{N}]_\equiv \cup \{q_s, q_f\}$ *is the state alphabet, where* $[\mathcal{N}]_\equiv$ *is the set of equivalence classes* $[\nu]_\equiv$ *over* $\mathcal{N}$ *modulo the equivalence relation* $\equiv$*,*
- V_b' *is the input alphabet,*
- R *in* $Q\,(V_b' \cup \{\varepsilon\}) \times Q$ *is the set of rules* $\{q\chi \vdash q' \mid \exists \nu \in q \text{ and } \nu' \in q', \nu \xrightarrow{\chi} \nu'\} \cup \{q_s\varepsilon \vdash [\nu_s]_\equiv \mid \nu_s \in \mu_s\} \cup \{[\nu_f]_\equiv \varepsilon \vdash q_f \mid \nu_f \in \mu_f\} \cup \{q_f\$ \vdash q_f\}$*, and*
- q_s *and* q_f *are respectively the initial and the final state.*

If the chosen equivalence relation is of finite index, then the nondeterministic position automaton is finite. For instance, an equivalence relation that results in a NFA similar to a nondeterministic LR(0) automaton [18,19]—the main difference being the presence of the r_i transitions—is item_0 defined by

$$x_b d_i(\,{}^{\alpha}_{u_b} \bullet {}^{\alpha'}_{u_b'})r_i x_b' \;\; \mathsf{item}_0 \;\; y_b d_j(\,{}^{\beta}_{v_b} \bullet {}^{\beta'}_{v_b'})r_j y_b' \text{ iff } i = j \text{ and } \alpha' = \beta'. \tag{6}$$

The equivalence classes in $[\mathcal{N}]_{\mathsf{item}_0}$ are the LR(0) items. Figure 3 presents the nondeterministic automaton for $\mathcal{G}_1$ resulting from the use of item_0 as equivalence relation. Some plain ε-transitions and states of form $\bullet A$ and $A\bullet$ were added in order to reduce clutter in the figure. The addition of these states and transitions results in a $\mathcal{O}(|\mathcal{G}|)$ bound on the size of Γ/item_0 [18]. Our position (5) is now in the equivalence class represented by the state labeled by $NP \to NP \bullet PP$.

Let us denote by $\vDash$ the relation between configurations of a NFA $\mathcal{A} = \langle Q, \Sigma, R, q_s, F\rangle$, such that $qaw \vDash q'w$ if and only if there exists a rule $qa \vdash q'$ in R. The language recognized by $\mathcal{A}$ is then $\mathcal{L}(\mathcal{A}) = \{w \in \Sigma^* \mid \exists q_f \in F, q_s w \vDash^* q_f\}$.

Theorem 1. *Let* $\mathcal{G}$ *be a context-free grammar and* $\equiv$ *an equivalence relation on* $\mathcal{N}$*. The language generated by* $\mathcal{G}_b$ *is included in the terminal language recognized by* $\Gamma/\equiv$*, i.e.* $\mathcal{L}(\mathcal{G}_b) \subseteq \mathcal{L}(\Gamma/\equiv) \cap T_b^*$.

4 Ambiguity Detection

We are now in position to detect ambiguities on a finite, regular structure that approximates our initial grammar.

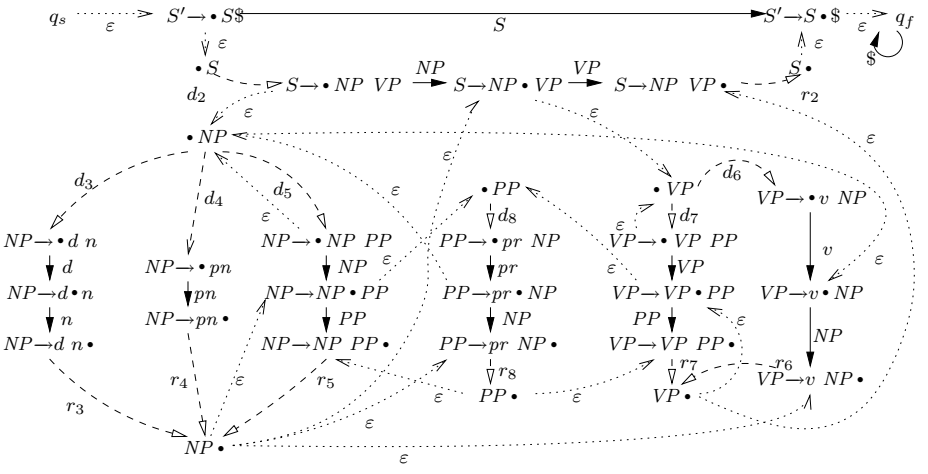

Fig. 3. The nondeterministic position automaton for $\mathcal{G}_1$ using item_0

4.1 Regular Ambiguity Detection

Our first conservative ambiguity checking procedure relies on Theorem 1. Following the arguments developed in Section 2.2, an ambiguity in $\mathcal{G}$ implies the existence of two sentences w_b and w'_b in the regular super language $\mathcal{L}(\Gamma/\equiv) \cap T_b^*$ such that $w = w'$. We call a CFG with no such pair of sentences *regular* $\equiv$-*unambiguous*, or RU($\equiv$) for short.

The existence of such a pair of sentences can be tested in $\mathcal{O}(|\Gamma/\equiv|^2)$ using accessibility relations like the ones developped in Section 4.3. How good is this algorithm? Being conservative is not enough for practical uses; after all, a program that always answers that the tested grammar is ambiguous is a conservative test. The regular ambiguity test sketched above performs unsatisfactorily: when using the item_0 equivalence, it finds some LR(0) grammars ambiguous, like for instance $\mathcal{G}_2$ with rules

$$S \to aAa \,|\, bAa,\ A \to c. \qquad (\mathcal{G}_2)$$

The sentences $d_2 a d_4 c r_4 a r_2$ and $d_2 a d_4 c r_4 a r_3$ are both in $\mathcal{L}(\Gamma_2/\ \mathsf{item}_0) \cap T_b^*$.

The LR algorithm [16] hints at a solution: we could consider nonterminal symbols in our verification and thus avoid spurious paths in the NFA. A single direct step using a nonterminal symbol represents *exactly* the context-free language derived from it, much more accurately than any regular approximation we could make for this language.

4.2 Common Prefixes with Conflicts

Let us consider again the two sentences (1) and (2), but let us dismiss all the d_i symbols; the two sentences (7) and (8) we obtain are still different:

$$pn\ r_4\ v\ d\ n\ r_3\ pr\ d\ n\ r_3\ r_8\ r_5\ r_6\ r_2\ \$\ r_1 \qquad (7)$$

$$pn\ r_4\ v\ d\ n\ r_3\ r_6\ pr\ d\ n\ r_3\ r_8\ r_7\ r_2\ \$\ r_1. \qquad (8)$$

They share a longest common prefix $pn\, r_4\, v\, d\, n\, r_3$ before a *conflict*[3] between pr and r_6.

Observe that the two positions in conflict could be reached more directly in a NFA by reading the prefix $NP\ v\ NP$. We obtain the two sentential forms

$$NP\ v\ NP\ pr\ d\ n\ r_3\ r_8\ r_5\ r_6\ r_2\ \$\ r_1 \tag{9}$$
$$NP\ v\ NP\ r_6\ pr\ d\ n\ r_3\ r_8\ r_7\ r_2\ \$\ r_1. \tag{10}$$

We cannot however reduce our two sentences to two identical sentential forms: our common prefix with one conflict $pn\, r_4\, v\, d\, n\, r_3\, r_6$ would reduce to a different prefix $NP\ VP$, and thus we do not reduce the conflicting reduction symbol r_6.

The remaining suffixes $pr\ d\ n\, r_3\, r_8\, r_5\, r_6\, r_2\, \$\, r_1$ and $pr\ d\ n\, r_3\, r_8\, r_7\, r_2\, \$\, r_1$ share again a longest common prefix $pr\ d\ n\, r_3\, r_8$ before a conflict between r_5 and r_7; the common prefix reduces to PP, and we have the sentential forms

$$NP\ v\ NP\ PP\ r_5\ r_6\ r_2\ \$\ r_1 \tag{11}$$
$$NP\ v\ NP\ r_6\ PP\ r_7\ r_2\ \$\ r_1. \tag{12}$$

Keeping the successive conflicting reduction symbols r_5, r_6 and r_7, we finally reach a common suffix $r_2\ \$\ r_1$ that cannot be reduced any further, since we need to keep our conflicting reductions. The image of our two different reduced sentential forms (11) and (12) by h is a common sentential form $NP\ v\ NP\ PP\ \$$, which shows the existence of an ambiguity in our grammar.

We conclude from our small example that, in order to give preference to the more accurate direct path over its terminal counterpart, we should only follow the r_i transitions in case of conflicts or in case of a common factor that cannot be reduced due to the earlier conflicts. This general behavior is also the one displayed by noncanonical parsers [20].

4.3 Accessibility Relations

We implement the idea of common prefixes with conflicts in the mutual accessibility relations classically used to find common prefixes [21, Chapter 10]. Mutual accessibility relations are used to identify couples of states accessible upon reading the same language from the starting couple (q_s, q_s), which brings the complexity of the test down to a quadratic function in the number of transitions, and avoids the potential exponential blowup of a NFA determinization.

The case where reduction transitions should be followed after a conflict is handled by considering pairs over $\mathbb{B} \times Q$ instead of Q: the boolean tells whether we followed a d_i transition since the last conflict. In order to improve readability, we write $q\chi \vdash q'$ for q and q' in $\mathbb{B} \times Q$ if their states allow this transition to occur. The predicate $\backslash q$ in $\mathbb{B}$ denotes that we are allowed to ignore a reduction transition. Our starting couple (q_s, q_s) has its boolean values initially set to true.

[3] Our notion of conflict coincides with that of LR(0) conflicts when one employs item_0.

Definition 3. *The primitive mutual accessibility relations over* $(\mathbb{B} \times Q)^2$ *are*

shift. mas *defined by* $(q_1, q_2)\ \mathsf{mas}\ (q_3, q_4)$ *if and only if there exists* X *in* V *such that* $q_1 X \vdash q_3$ *and* $q_2 X \vdash q_4$

epsilon. $\mathsf{mae}{=}\mathsf{mael} \cup \mathsf{maer}$ *where* $(q_1, q_2)\ \mathsf{mael}\ (q_3, q_2)$ *if and only if* $q_1 d_i \vdash q_3$ *or* $q_1 \varepsilon \vdash q_3$ *and* $\backslash q_3$ *and symmetrically for* maer, $(q_1, q_2)\ \mathsf{maer}\ (q_1, q_4)$ *if and only if* $q_2 d_i \vdash q_4$ *or* $q_2 \varepsilon \vdash q_4$, *and* $\backslash q_4$,

reduction. mar *defined by* $(q_1, q_2)\ \mathsf{mar}\ (q_3, q_4)$ *if and only if there exists* i *in* P *such that* $q_1 r_i \vdash q_3$ *and* $q_2 r_i \vdash q_4$, *and furthermore* $\neg \backslash q_1$ *or* $\neg \backslash q_2$, *and then* $\neg \backslash q_3$ *and* $\neg \backslash q_4$,

conflict. $\mathsf{mac}{=}\mathsf{macl} \cup \mathsf{macr}$ *with* $(q_1, q_2)\ \mathsf{macl}\ (q_3, q_2)$ *if and only if there exist* i *in* P, q_4 *in* Q *and* z *in* $T_d^* \cdot T'$ *such that* $q_1 r_i \vdash q_3$, $q_2 z \vDash^+ q_4$ *and* $\neg \backslash q_3$, *and symmetrically for* macr, $(q_1, q_2)\ \mathsf{macr}\ (q_1, q_4)$ *if and only if there exist* i *in* P, q_3 *in* Q *and* z *in* $T_d^* \cdot T'$ *such that* $q_2 r_i \vdash q_4$, $q_1 z \vDash^+ q_3$, *and* $\neg \backslash q_4$.

The global mutual accessibility relation ma *is defined as* $\mathsf{mas} \cup \mathsf{mae} \cup \mathsf{mar} \cup \mathsf{mac}$.

These relations are akin to the item construction of a LR parser: the relation mas corresponds to a shift, the relation mae to an item closure, the relation mar to a goto, and the relation mac to a LR conflict.

Let us call a grammar $\mathcal{G}$ such that $(q_s, q_s)\ (\mathsf{mae} \cup \mathsf{mas})^* \circ \mathsf{mac} \circ \mathsf{ma}^*\ (q_f, q_f)$ does not hold in $\Gamma/{\equiv}$ *noncanonically* $\equiv$*-unambiguous*, or NU($\equiv$) for short.

Theorem 2. *Let* $\mathcal{G}$ *be a context-free grammar and* $\equiv$ *a position equivalence relation. If* $\mathcal{G}$ *is ambiguous, then* $\mathcal{G}$ *is not NU(*$\equiv$*).*

4.4 Complexity

The complexity of our algorithm depends mostly on the equivalence relation we choose to quotient the position graph. Supposing that we choose an equivalence relation $\equiv$ of finite index and of decidable computation of complexity $\mathcal{C}(\Gamma/{\equiv})$, then we need to build the image $\mathsf{ma}^*\ (\{(q_s, q_s)\})$. This step and the search for a conflict in this image can both be performed in time $\mathcal{O}(|\Gamma/{\equiv}|^2)$. The overall complexity of our algorithm is thus $\mathcal{O}(\mathcal{C}(\Gamma/{\equiv}) + |\Gamma/{\equiv}|^2)$.

The complexity $\mathcal{C}(\Gamma/\mathsf{item}_0)$ of the construction of the collapsed position graph Γ/item_0 is linear with the size of the resulting nondeterministic position automaton. The overall complexity of our ambiguity detection algorithm when one uses item_0 is therefore $\mathcal{O}(|\mathcal{G}|^2)$.

5 Formal Comparisons

We compare here our ambiguity detection algorithm with some of the other means to test a context-free grammar for ambiguity we are aware of. We first establish the edge of our algorithm over the regular ambiguity test of Section 4.1. The comparison with LR-Regular testing requires the full power of our method, and at last, the horizontal and vertical ambiguity detection technique is shown to be incomparable with our own.

5.1 Regular Ambiguity

Theorem 3, along with the example of $\mathcal{G}_2$, shows a strict improvement of our method over the simple algorithm discussed in Section 4.1.

Theorem 3. *If $\mathcal{G}$ is RU($\equiv$), then it is also NU($\equiv$).*

5.2 Bounded Length Detection Schemes

Many algorithms specifically designed for ambiguity detection look for ambiguities in all sentences up to some length [10,1,6,11]. As such, they fail to detect ambiguities beyond that length: they allow false negatives. Nonetheless, these detection schemes can vouch for the ambiguity of any string shorter than the given length; this is valuable in applications where, in practice, the sentences are of a small bounded length. The same guarantee is offered by the equivalence relation prefix_m defined for any fixed length m by[4]

$$x_b d_i(\begin{smallmatrix}\alpha\\ u_b\end{smallmatrix}\bullet\begin{smallmatrix}\alpha'\\ u'_b\end{smallmatrix})r_i x'_b \;\mathsf{prefix}_m\; y_b d_j(\begin{smallmatrix}\beta\\ v_b\end{smallmatrix}\bullet\begin{smallmatrix}\beta'\\ v'_b\end{smallmatrix})r_j y'_b \text{ iff } m :_b x_b u_b = m :_b y_b v_b. \tag{13}$$

Provided that $\mathcal{G}$ is acyclic, Γ/prefix_m is finite.

Theorem 4. *Let w_b and w'_b be two bracketed sentences in $\mathcal{L}(\Gamma/\mathsf{prefix}_m) \cap T_b^*$ with $w = w'$ and $|w| \leq m$. Then w_b and w'_b are in $\mathcal{L}(\mathcal{G}_b)$.*

Outside of the specific situation of languages that are finite in practice, bounded length detection schemes can be quite costly to use. The performance issue can be witnessed with the two families of grammars $\mathcal{G}_3^n$ and $\mathcal{G}_4^n$ with rules

$$S{\to}A\,|\,B_n,\ A{\to}Aaa\,|\,a,\ B_1{\to}aa,\ B_2{\to}B_1B_1,\ \ldots,\ B_n{\to}B_{n-1}B_{n-1} \tag{$\mathcal{G}_3^n$}$$

$$S{\to}A\,|\,B_n a,\ A{\to}Aaa\,|\,a,\ B_1{\to}aa,\ B_2{\to}B_1B_1,\ \ldots,\ B_n{\to}B_{n-1}B_{n-1}, \tag{$\mathcal{G}_4^n$}$$

where $n \geq 1$. In order to detect the ambiguity of $\mathcal{G}_4^n$, a bounded length algorithm would have to explore all strings in $\{a\}^*$ up to length $2^n + 1$. Our algorithm correctly finds $\mathcal{G}_3^n$ unambiguous and $\mathcal{G}_4^n$ ambiguous in time $\mathcal{O}(n^2)$ using item_0.

5.3 LR(k) and LR-Regular Testing

Conservative algorithms do exist in the programming language parsing community, though they are not primarily meant as ambiguity tests. Nonetheless, a full LALR or LR construction is often used as a practical test for non ambiguity [2]. The LR(k) testing algorithms [16,18,19] are much more efficient in the worst case and provided our initial inspiration. Our position automaton is a generalization of the item grammar or nondeterministic automaton of these works, and our test looks for ambiguities instead of LR conflicts. Let us consider again $\mathcal{G}_3^n$: it requires a LR(2^n) test for proving its unambiguity, but it is simply NU(item_0).

[4] The *bracketed prefix* $m :_b x_b$ of a bracketed string x_b is defined as the longest string in $\{y_b \mid x_b = y_b z_b \text{ and } |y| = m\}$ if $|x| > m$ or simply x_b if $|x| \leq m$.

One of the strongest ambiguity tests available is the LR-Regular condition [12,22]: instead of merely checking the k next symbols of lookahead, a LRR parser considers regular equivalence classes on the entire remaining input to infer its decisions. Given Π a finite regular partition of T^* that defines a left congruence $\cong$, a grammar $\mathcal{G}$ is LR(Π) if and only if $S \underset{\text{rm}}{\Longrightarrow}^* \delta A x \underset{\text{rm}}{\Longrightarrow} \delta\alpha x$, $S \underset{\text{rm}}{\Longrightarrow}^* \gamma B y \underset{\text{rm}}{\Longrightarrow} \gamma\beta y = \delta\alpha z$ and $x \cong z \pmod{\Pi}$ imply $A{\rightarrow}\alpha = B{\rightarrow}\beta$, $\delta = \gamma$ and $y = z$.

Our test for ambiguity is strictly stronger than the LR(Π) condition with the equivalence relation $\mathsf{item}_\Pi{=}\mathsf{item}_0 \cap \mathsf{look}_\Pi$, where look_Π is defined by

$$x_b d_i(\,{}^{\alpha}_{u_b}\bullet{}^{\alpha'}_{u'_b})r_i x'_b \;\; \mathsf{look}_\Pi \;\; y_b d_j(\,{}^{\beta}_{v_b}\bullet{}^{\beta'}_{v'_b})r_j y'_b \text{ iff } u'x' \cong v'y' \pmod{\Pi}. \tag{14}$$

Theorem 5. *If $\mathcal{G}$ is LR(Π), then it is also NU(item_Π).*

Let us consider now the grammar with rules

$$S{\rightarrow}AC\,|\,BCb,\;\; A{\rightarrow}a,\;\; B{\rightarrow}a,\;\; C{\rightarrow}cCb\,|\,cb. \tag{$\mathcal{G}_5$}$$

Grammar $\mathcal{G}_5$ is not LRR: the right contexts $c^n b^n\$$ and $c^n b^{n+1}\$$ of the reductions using $A{\rightarrow}a$ and $B{\rightarrow}a$ cannot be distinguished by regular covering sets. Nevertheless, our test on $\Gamma_5/\,\mathsf{item}_0$ shows that $\mathcal{G}_5$ is not ambiguous

5.4 Horizontal and Vertical Ambiguity

Brabrand *et al.* [3] recently proposed an ambiguity detection scheme also based on regular approximations of the grammar language. Its originality lies in the decomposition of the ambiguity problem into two (also undecidable) problems, namely the horizontal and vertical ambiguity problems. The detection method then relies on the fact that a context-free grammar is unambiguous if and only if it is horizontal and vertical unambiguous. The latter tests are performed on a regular approximation of the grammar [23].

Definition 4. *The automaton $\Gamma/{\equiv}$ is vertically ambiguous if and only if there exist an A in N with two different productions $A \xrightarrow{i} \alpha_1$ and $A \xrightarrow{j} \alpha_2$, and the bracketed strings x_b, x'_b, u_b, u'_b, w_b, and w'_b in T_b^* with $w = w'$ such that*

$$[x_b d_i(\,\bullet{}^{\alpha_1}_{u_b})r_i x'_b]_{\equiv} w_b \vdash^* [x_b d_i(\,{}^{\alpha_1}_{u_b}\bullet\,)r_i x'_b]_{\equiv} \text{ and}$$

$$[x_b d_j(\,\bullet{}^{\alpha_2}_{u'_b})r_j x'_b]_{\equiv} w'_b \vdash^* [x_b d_j(\,{}^{\alpha_2}_{u'_b}\bullet\,)r_j x'_b]_{\equiv}.$$

The automaton $\Gamma/{\equiv}$ is horizontally ambiguous if and only if there is a production $A \xrightarrow{i} \alpha$ in P, a decomposition $\alpha = \alpha_1\alpha_2$, and the bracketed strings x_b, x'_b, u_b, u'_b, v_b, v'_b, w_b, w'_b, and y_b in T_b^* with $v = v'$ and $w = w'$ such that

$$[x_b d_i(\,\bullet{}^{\alpha_1\alpha_2}_{u_b u'_b})r_i x'_b]_{\equiv} v_b y_b w_b \vdash^* [x_b d_i(\,{}^{\alpha_1}_{u_b}\bullet{}^{\alpha_2}_{u'_b})r_i x'_b]_{\equiv} y_b w_b \vdash^* [x_b d_i(\,{}^{\alpha_1\alpha_2}_{u_b u'_b}\bullet\,)r_i x'_b]_{\equiv}$$

$$[x_b d_i(\,\bullet{}^{\alpha_1\alpha_2}_{u_b u'_b})r_i x'_b]_{\equiv} v'_b y_b w'_b \vdash^* [x_b d_i(\,{}^{\alpha_1}_{u_b}\bullet{}^{\alpha_2}_{u'_b})r_i x'_b]_{\equiv} w'_b \vdash^* [x_b d_i(\,{}^{\alpha_1\alpha_2}_{u_b u'_b}\bullet\,)r_i x'_b]_{\equiv}.$$

Theorem 6. *Let $\mathcal{G}$ be a context-free grammar and $\Gamma/\equiv$ its position automaton. If $\mathcal{G}$ is RU($\equiv$), then $\Gamma/\equiv$ is horizontally and vertically unambiguous.*

Theorem 6 shows that the horizontal and vertical ambiguity criteria result in a better conservative ambiguity test than regular $\equiv$-ambiguity, although at a higher price: $\mathcal{O}(|\mathcal{G}|^5)$ in the worst case. Owing to these criteria, the technique of Brabrand *et al.* accomplishes to show that the palindrome grammar with rules

$$S \to aSa \,|\, bSb \,|\, a \,|\, b \,|\, \varepsilon \qquad (\mathcal{G}_6)$$

is unambiguous, which seems impossible with our scheme. On the other hand, even when they employ *unfolding* techniques, they are always limited to regular approximations, and fail to see that the LR(0) grammar with rules

$$S \to AA,\ A \to aAa \,|\, b \qquad (\mathcal{G}_7)$$

is unambiguous. The two techniques are thus incomparable, and could benefit from each other.

6 Conclusion

As a classical undecidable problem in formal languages, ambiguity detection in context-free grammars did not receive much practical attention. Nonetheless, the ability to provide a conservative test could be applied in many fields where context-free grammars are used. This paper presents one of the few conservative tests explicitly aimed towards ambiguity checking, along with the recent work of Brabrand *et al.* [3].

The ambiguity detection scheme we presented here provides some insights on how to tackle undecidable problems on approximations of context-free languages. The general method can be applied to different decision problems, and indeed has also been put to work in the construction of an original parsing method [24] where the amount of lookahead needed is not preset but computed for each parsing decision. We hope to see more applications of this model in the future.

Acknowledgements. The author is highly grateful to Jacques Farré for his invaluable help at all stages of the preparation of this work. The author also thanks the anonymous referees for their numerous helpful remarks.

References

1. Cheung, B.S.N., Uzgalis, R.C.: Ambiguity in context-free grammars. In: SAC'95, pp. 272–276. ACM Press, New York (1995), doi:10.1145/315891.315991
2. Reeder, J., Steffen, P., Giegerich, R.: Effective ambiguity checking in biosequence analysis. BMC Bioinformatics 6, 153 (2005)
3. Brabrand, C., Giegerich, R., Møller, A.: Analyzing ambiguity of context-free grammars. Technical Report RS-06-09, BRICS, University of Aarhus (May 2006)
4. AeroSpace and Defence Industries Association of Europe: ASD Simplified Technical English, Specification ASD-STE100 (2005)

5. Kuich, W.: Systems of pushdown acceptors and context-free grammars. Elektronische Informationsverarbeitung und Kybernetik 6(2), 95–114 (1970)
6. Schröer, F.W.: AMBER, an ambiguity checker for context-free grammars. Technical report, compilertools.net (2001)
7. Schmitz, S.: An experimental ambiguity detection tool. In: Sloane, A., Johnstone, A., eds.: LDTA'07, To appear in Electronic Notes in Theoretical Computer Science (2007)
8. Cantor, D.G.: On the ambiguity problem of Backus systems. Journal of the ACM 9(4), 477–479 (1962)
9. Chomsky, N., Schützenberger, M.P.: The algebraic theory of context-free languages. In: Braffort, P., Hirshberg, D. (eds.) Computer Programming and Formal Systems. Studies in Logic, pp. 118–161. North-Holland Publishing, Amsterdam (1963)
10. Gorn, S.: Detection of generative ambiguities in context-free mechanical languages. Journal of the ACM 10(2), 196–208 (1963)
11. Jampana, S.: Exploring the problem of ambiguity in context-free grammars. Master's thesis, Oklahoma State University (July 2005)
12. Culik, K., Cohen, R.: LR-Regular grammars—an extension of LR(k) grammars. Journal of Computer and System Sciences 7, 66–96 (1973)
13. Schmitz, S.: Conservative ambiguity detection in context-free grammars. Technical Report I3S/RR-2006-30-FR, Laboratoire I3S, Université de Nice - Sophia Antipolis & CNRS (September 2006)
14. Ginsburg, S., Harrison, M.A.: Bracketed context-free languages. Journal of Computer and System Sciences 1, 1–23 (1967)
15. Nederhof, M.J.: Regular approximation of CFLs: a grammatical view. In: Bunt, H., Nijholt, A. (eds.) Advances in Probabilistic and other Parsing Technologies, pp. 221–241. Kluwer Academic Publishers, Boston, MA (2000)
16. Knuth, D.E.: On the translation of languages from left to right. Information and Control 8(6), 607–639 (1965)
17. Sikkel, K. (ed.): Parsing Schemata - a framework for specification and analysis of parsing algorithms. Texts in Theoretical Computer Science - An EATCS Series. Springer, Heidelberg (1997)
18. Hunt III, H.B., Szymanski, T.G., Ullman, J.D.: Operations on sparse relations and efficient algorithms for grammar problems. In: 15th Annual Symposium on Switching and Automata Theory, pp. 127–132. IEEE Computer Society, Los Alamitos (1974)
19. Hunt III, H.B., Szymanski, T.G., Ullman, J.D.: On the complexity of LR(k) testing. Communications of the ACM 18(12), 707–716 (1975), doi:10.1145/361227.361232
20. Szymanski, T.G., Williams, J.H.: Noncanonical extensions of bottom-up parsing techniques. SIAM Journal on Computing 5(2), 231–250 (1976), doi:10.1137/0205019
21. Sippu, S., Soisalon-Soininen, E.: Parsing Theory, Vol. II: LR(k) and LL(k) Parsing. In: Simple Program Schemes and Formal Languages. LNCS, vol. 20, Springer, Heidelberg (1990)
22. Heilbrunner, S.: Tests for the LR-, LL-, and LC-Regular conditions. Journal of Computer and System Sciences 27(1), 1–13 (1983)
23. Mohri, M., Nederhof, M.J.: Regular approximations of context-free grammars through transformation. In: Junqua, J.C., van Noord, G. (eds.) Robustness in Language and Speech Technology, pp. 153–163. Kluwer Academic Publishers, Dordrecht (2001)
24. Gálvez, J.F., Schmitz, S., Farré, J.: Shift-resolve parsing: Simple, linear time, unbounded lookahead. In: Ibarra, O.H., Yen, H.-C. (eds.) CIAA 2006. LNCS, vol. 4094, pp. 253–264. Springer, Heidelberg (2006)