

LABORATOIRE



INFORMATIQUE, SIGNAUX ET SYSTÈMES
DE SOPHIA ANTIPOLIS
UMR 6070

CONSERVATIVE AMBIGUITY DETECTION IN CONTEXT-FREE GRAMMARS

Sylvain Schmitz

Projet LANGAGES

Rapport de recherche
ISRN I3S/RR-2006-30-FR

Octobre 2006

RÉSUMÉ :

La capacité de détecter les ambiguïtés dans les grammaires algébriques est vitale pour leur utilisation dans plusieurs domaines, mais le problème est indécidable dans le cas général. Nous présentons une approche sûre, où les approximations employées ne peuvent résulter en l'oubli de certaines ambiguïtés. Nous analysons la complexité de son utilisation, et nous la comparons avec les autres méthodes de détection d'ambiguïtés.

MOTS CLÉS :

Langage formel, ambiguïté, grammaire algébrique, graphe de positions

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 approximations cannot result in overlooked ambiguous cases. We analyze the complexity of its use, and compare it with other ambiguity detection methods.

KEY WORDS :

Formal language, ambiguity, context-free grammar, position graph

Conservative Ambiguity Detection in Context-Free Grammars

Sylvain Schmitz

Laboratoire I3S, Université de Nice - Sophia Antipolis, 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 approximations cannot result in overlooked ambiguous cases. We analyze the complexity of its use, and compare it with other ambiguity detection methods.

Key words: Formal language, ambiguity, context-free grammar, position graph

ACM categories: F.3.1 [*Logics and Meanings of Programs*]: Specifying and Verifying and Reasoning about Programs; F.4.2 [*Mathematical Logic and Formal Languages*]: Grammars and Other Rewriting Systems

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”. Ambiguity ought to be detected in several fields where context-free grammars are used to model the syntax, for instance language acquisition [7], RNA analysis [25, 5], controlled natural languages [1], or programming languages [27, 26]. The presence of ambiguities in a context-free grammar hampers the reliability or the performance of the tools build from it.

While proven undecidable [6, 8], 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.

We present in this paper 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 2) and check for ambiguities in this abstract structure (Section 3). More precisely,

- we quotient the *position graph* of all the parse trees of the grammar into a nondeterministic finite automaton (NFA) (Section 2.2),

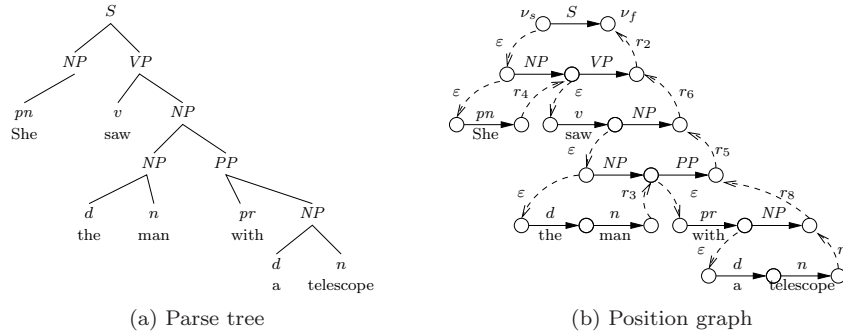


Figure 1: One possible parse for the sentence “She saw the man with a telescope.”.

- all ambiguities of the grammar are detected on this NFA (Section 3); not all paths are considered, and the approximative language we use is in fact a context-free one,
- the complexity of the algorithm depends on the chosen equivalence relation (Section 3.3), and
- we provide formal comparisons with other techniques that allow the detection of ambiguities, notably with the LR-Regular [10] condition (Section 4).

We assume the reader is familiar with context-free grammars and formal languages in general; we adhere to the (classical) notation of Sippu and Soisalon-Soininen [28]. All our grammars $\mathcal{G} = \langle N, T, P, S \rangle$ are context-free and reduced. We denote by \mathcal{G}' the grammar augmented with the rule $S' \rightarrow \$S\$$.

2 Grammatical Representation

The grammar with rules

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

is a simple English grammar that allows to derive the sentence “She saw the man with a telescope.”. One of the two possible interpretations of the sentence is described by the parse tree in Figure 1a; the other possibility would have been to use the rule $VP \rightarrow VP PP$.

The strategy we adopt for the detection of ambiguities is to first build a finite approximated representation of the grammar, and then to look for ambiguities in it. While a context-free grammar is already a finite representation of its generated language, in general it does not bend itself to static checking: the very context-freeness of the language makes the use of some stack mandatory, and thus only a dynamic check could be performed accurately.

As explained in Section 3, rather than exploit directly the regular language described by our finite structure, we restrict the allowed paths and consider in fact a context-free language that includes the original context-free language.

2.1 Position Graph

Let us consider the forest of all parse trees for a given grammar. In each of its trees, a left to right walk can be performed. We identify during this walk *positions* to the immediate left or immediate right of a derivation tree node. Transitions from one position to the other can then be performed upon reading the tree node symbol, upon deriving from this node, and upon returning from such a derivation. We have thus three types of transitions: symbol transitions \xrightarrow{X} , and two kinds of ε -transitions: derivation transitions $\xrightarrow{\varepsilon}$ and reduction transitions $\xrightarrow{r_i}$ 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 1b presents the portion of the position graph for \mathcal{G}_1 corresponding to the tree of Figure 1a.

Bracketed Grammar The position graph is a fully expanded representation of a bracketed context-free grammar, where rules $i = A \rightarrow \alpha$ of \mathcal{G} are surrounded by ε and r_i . Formally, our *bracketed grammar* of a context-free grammar \mathcal{G} is the context-free grammar $\mathcal{G}_b = \langle N, T_b, P_b, S \rangle$ where $T_b = T \cup \{r_i \mid i \in P\}$ and $P_b = \{i_b = A \rightarrow \alpha r_i \mid i = A \rightarrow \alpha \in P\}$. A sentence of \mathcal{G}_b represents a single parse tree of \mathcal{G} . We define the homomorphism h from V_b^* to V^* by $h(r_i) = \varepsilon$ for all i in P , and $h(X) = X$ otherwise. We sometimes write δ_b to denote a bracketed string in V_b^* such that $h(\delta_b) = \delta$. A bracketed grammar is never ambiguous.

Positions In order to uniquely identify a single position in a single parse tree, we define *valid positions* for a grammar \mathcal{G} as $\nu = x_b(\frac{\alpha}{u_b} \cdot \frac{\alpha'}{u'_b})r_i x'_b$ such that the derivations

$$S' \Rightarrow^* x_b A x'_b \xrightarrow{i} x_b \alpha \alpha' r_i x'_b, \quad \alpha \Rightarrow^* u_b \text{ and } \alpha' \Rightarrow^* u'_b \quad (1)$$

hold in \mathcal{G}'_b . The position immediately after “with” in the position graph of Figure 1b is identified by $\$ p n r_4 v d n r_3 (\frac{pr}{pr} \cdot \frac{NP}{dnr_3}) r_8 r_5 r_6 r_2 \$ r_1$.

Definition 1 The position graph $\Gamma = \langle \mathcal{N}, \succ \rangle$ of a grammar \mathcal{G} associates the (potentially infinite) set \mathcal{N} of valid positions for \mathcal{G} with the relation \succ labeled by elements of $V_b \cup \{\varepsilon\}$, defined by

$$\begin{aligned} x_b(\frac{\alpha}{u_b} \cdot \frac{X\alpha'}{v_b u'_b})r_i x'_b &\xrightarrow{X} x_b(\frac{\alpha X}{u_b v_b} \cdot \frac{\alpha'}{u'_b})r_i x'_b && \text{iff } X \in V, X \Rightarrow^* v_b \\ x_b(\frac{\alpha}{u_b} \cdot \frac{B\alpha'}{v_b u'_b})r_i x'_b &\xrightarrow{\varepsilon} x_b u_b (\frac{\beta}{v_b})r_j u'_b r_i x'_b && \text{iff } B \xrightarrow{j} \beta \text{ and } \beta \Rightarrow^* v_b \\ x_b u_b (\frac{\beta}{v_b} \cdot) r_j u'_b r_i x'_b &\xrightarrow{r_j} x_b(\frac{\alpha B}{u_b v_b} \cdot \frac{\alpha'}{u'_b})r_i x'_b && \text{iff } B \xrightarrow{j} \beta, \alpha \Rightarrow^* u_b \text{ and } \alpha' \Rightarrow^* u'_b. \end{aligned}$$

We label paths in Γ by the sequences of labels on the individual transitions. Let $1 = S' \rightarrow \$ S \$$; two sets of valid positions are of specific interest: $\mu_s = \{\$(\cdot w_b) S \$ r_1 \mid S \Rightarrow^* w_b\}$ and $\mu_f = \{\$ S (w_b \cdot) \$ r_1 \mid S \Rightarrow^* w_b\}$, where, 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 $\nu_s \xrightarrow{S} \nu_f$.

Lemma 1 Let ν and ν' be positions in \mathcal{N} , and δ_b and γ_b be strings in V_b^* with $\nu \xrightarrow{\delta_b} \nu'$. If $\delta_b \Rightarrow^* \gamma_b$ or $\gamma_b \Rightarrow^* \delta_b$ in \mathcal{G}_b , then $\nu \xrightarrow{\gamma_b} \nu'$.

Proof. Suppose $\delta_b \Rightarrow^n \gamma_b$; we proceed by induction on n the number of individual derivation steps. If $n = 0$, then $\delta_b = \gamma_b$ and the property holds. Let now

$\delta_b \Rightarrow^{n-1} \rho_b A \sigma_b \xrightarrow{i} \rho_b \alpha r_i \sigma_b = \gamma_b$ with $\nu \xrightarrow{\rho_b} \nu_1 \xrightarrow{A} \nu_2 \xrightarrow{\sigma_b} \nu'$. By Definition 1, $\nu_1 \xrightarrow{\varepsilon} \nu_3 \xrightarrow{\alpha} \nu_4 \xrightarrow{r_i} \nu_2$ and thus $\nu \xrightarrow{\gamma_b} \nu'$.

A similar procedure yields a proof if $\gamma_b \Rightarrow^* \delta_b$. \square

2.2 Position Equivalences

Nondeterministic Position Automaton 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. The equivalence classes can then be seen as states of a nondeterministic automaton.

Definition 2 *The nondeterministic position automaton Γ/\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,
- $R = \{q\chi \vdash q' \mid \exists \chi \in V_b \cup \{\varepsilon\}, \nu \in q \text{ and } \nu' \in q', \nu \xrightarrow{\chi} \nu'\} \cup \{q_s \varepsilon \vdash [\nu_s]_{\equiv} \mid \nu_s \in \mu_s\} \cup \{[\nu_f]_{\equiv} \varepsilon \vdash q_f \mid \nu_f \in \mu_f\} \cup \{q_f \$ \vdash q_f\}$ is the set of rules, and
- q_s and q_f are respectively the initial and the final state.

A straightforward induction on the number of individual steps in $\nu \xrightarrow{\delta_b} \nu'$ yields the following proposition.

Proposition 1 *Let ν, ν' be positions in \mathcal{N} and δ_b a string in V_b^* . If $\nu \xrightarrow{\delta_b} \nu'$, then $[\nu]_{\equiv} \delta_b \vDash^* [\nu']_{\equiv}$.*

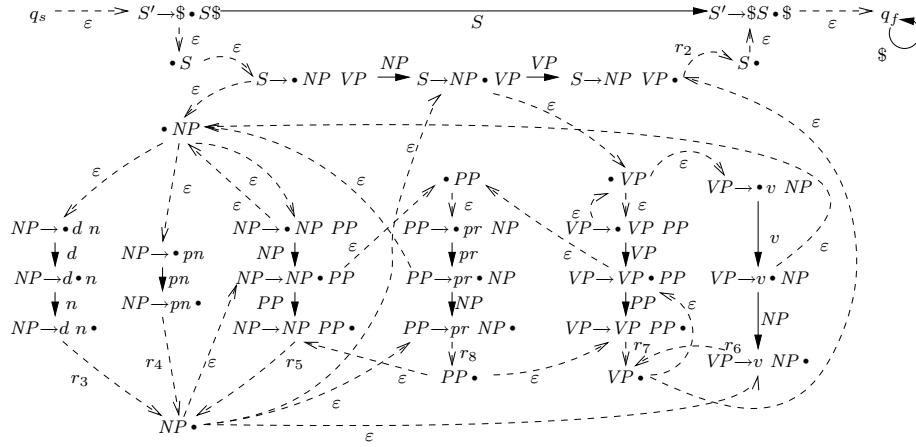
Theorem 1 *Let \mathcal{G} be a context-free grammar and \equiv an equivalence relation on \mathcal{N} . The language generated by \mathcal{G}_b is included in the terminal language recognized by Γ/\equiv , i.e. $\mathcal{L}(\mathcal{G}_b) \subseteq \mathcal{L}(\Gamma/\equiv) \cap T_b^*$.*

Proof. Let w_b be a sentence in $\mathcal{L}(\mathcal{G}_b)$ (and thus in T_b^*). Let us further consider the positions $\nu_s = \$(\cdot w_b)S\r_1 and $\nu_f = \$S(w_b \cdot)\r_1 ; $\nu_s \xrightarrow{S} \nu_f$. Using the derivation $S \Rightarrow^* w_b$ in \mathcal{G}_b and Lemma 1, we know that $\nu_s \xrightarrow{w_b} \nu_f$. Therefore, by Proposition 1, $[\nu_s]_{\equiv} w_b \vDash^* [\nu_f]_{\equiv}$. By Definition 2, $q_s \varepsilon \vdash [\nu_s]_{\equiv}$ and $[\nu_f]_{\equiv} \varepsilon \vdash q_f$ are in R , thus w_b is accepted by Γ/\equiv , i.e. w_b is in $\mathcal{L}(\Gamma/\equiv) \cap T_b^*$. \square

If the chosen equivalence relation is of finite index, then the nondeterministic position automaton is finite. For instance, an equivalence relation that will result in a NFA similar to a nondeterministic LR(0) automaton [18, 19] is item_0 defined by

$$x_b \left(\begin{smallmatrix} \alpha \\ u_b \end{smallmatrix} \cdot \begin{smallmatrix} \alpha' \\ u'_b \end{smallmatrix} \right) r_i x'_b \quad \text{item}_0 \quad y_b \left(\begin{smallmatrix} \beta \\ v_b \end{smallmatrix} \cdot \begin{smallmatrix} \beta' \\ v'_b \end{smallmatrix} \right) r_j y'_b \quad \text{iff } i = j \text{ and } \alpha' = \beta'. \quad (2)$$

The equivalence classes in $[\mathcal{N}]_{\text{item}_0}$ are the LR(0) items. Figure 2 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 $\cdot A$ and $A \cdot$ were added in order to reduce clutter in the figure; the other states are represented by LR(0) items. Our position right after “with” is now represented by the state labeled by $PP \rightarrow pr \cdot NP$ in the middle of Figure 2.

Figure 2: The nondeterministic position automaton for \mathcal{G}_1 using item_0 .

Lattice of Equivalence Relations The usual partial order on $\text{Eq}(\mathcal{N})$ —the complete lattice of all equivalence relations on \mathcal{N} —is the inclusion relation \subseteq . The largest element in $\text{Eq}(\mathcal{N})$, denoted by \top , always results in a single equivalence class, while the smallest (and finest) equivalence relation is the identity on \mathcal{N} , denoted by \perp . The lattice structure provides two operations for combining equivalence relations into new ones: for any two elements \equiv_a and \equiv_b of $\text{Eq}(\mathcal{N})$, $\equiv_a \vee \equiv_b$ is the least upper bound or *join* of \equiv_a and \equiv_b , and $\equiv_a \wedge \equiv_b$ is the greatest lower bound or *meet*. These operations are defined as

$$\equiv_a \vee \equiv_b = (\equiv_a \circ \equiv_b)^+ \quad (3)$$

$$\equiv_a \wedge \equiv_b = \equiv_a \cap \equiv_b . \quad (4)$$

This very ability to combine equivalence relations makes our grammatical representation highly generic, and allows for various trade-offs. For instance, finer equivalence relations are obtained when using the meet of two equivalence relations; they result in larger nondeterministic position automata. The item_k equivalence relation, comparable in effect to a $\text{LR}(k)$ precision, can be described as $\text{item}_0 \wedge \text{look}_k$ for any $k \geq 0$ where

$$x_b \binom{\alpha}{u_b} \cdot \binom{\alpha'}{u'_b} r_i x'_b \text{look}_k y_b \binom{\beta}{v_b} \cdot \binom{\beta'}{v'_b} r_j y'_b \text{ iff } k : u'x' = k : v'y' . \quad (5)$$

Size of the Nondeterministic Automaton We consider here the size of the automaton as the maximum between $|Q|$ and $|R|$; it obviously depends on the chosen \equiv . Let us first remark that our lattice of equivalence relations is connected to a lattice of nondeterministic position automata sizes: if $\equiv_a \subseteq \equiv_b$, then $|\mathcal{N}_{\equiv_a}| \geq |\mathcal{N}_{\equiv_b}|$, thus $|\Gamma/\equiv_a| \geq |\Gamma/\equiv_b|$. The smallest nondeterministic position automaton is thus Γ/\top , of size $|V| + 2|P| + 2$.

The item_k equivalence relations result in nondeterministic position automata of size $|\Gamma/\text{item}_k| = \mathcal{O}(|\mathcal{G}| |T|^{2k} |P|)$, or $\mathcal{O}(|\mathcal{G}|)$ for item_0 if we use additional ε -transitions as in Figure 2 to reduce the overall number of transitions [18].

3 Ambiguity Detection

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

Formally, an ambiguity in the context-free grammar can be defined as the existence of two parse trees deriving the same sentence. Using the bracketed grammar, an ambiguity is thus the existence of two different sentences w_b and w'_b of \mathcal{G}_b such that $w = w'$. We only have to check whether there exist such two sentences in the language accepted by Γ/\equiv in order to detect this ambiguity; this operation can be performed in $\mathcal{O}(|\Gamma/\equiv|^2)$ time. By Theorem 1, no ambiguity can ever be overlooked by this test.

How good is this algorithm? Being conservative is not enough for practical uses; after all a program which always answers that the tested grammar is ambiguous is a conservative test. The quality of a conservative detection scheme is a measure of the number of false positives it will return. In this regard, the test sketched above performs unsatisfactorily: when using the item_0 equivalence, it will sometimes find some LR(0) grammars ambiguous (*e.g.* \mathcal{G}_2 , Section 4.1).

The above test does not follow any transition on a nonterminal symbol. Some of the information provided by the NFA is thus lost: as formulated in Lemma 1, following a nonterminal transition is equivalent to using the entire subautomaton accepting its derived language, at least with respect to the original grammar. Interestingly enough, following a single transition avoids many approximations: a nonterminal derives a context-free language, whereas using terminals only we are limited to regular languages. Our strategy thus consists in using nonterminal transitions as often as possible.

3.1 Common Prefixes with Conflicts

Let us consider two different sentences w_b and w'_b of \mathcal{G}_b such that $w = w'$. They share a longest common prefix u_b such that $w_b = u_b r_i v_{b,1}$ and $w'_b = u_b v'_{b,1}$ with $r_i \neq 1 : v'_{b,1}$. Let us call $u_{b,1} = u_b r_i$ and $u'_{b,1} = u_b$ the shortest common prefixes with one conflict. The remaining portions $v_{b,1}$ and $v'_{b,1}$, if different, also have a pair of shortest common prefixes $u_{b,2}$ and $u'_{b,2}$ with one conflict, so that $u_{b,1} u_{b,2}$ and $u'_{b,1} u'_{b,2}$ are shortest common prefixes with two conflicts.

If we consider for instance the bracketed sentences exhibiting an ambiguity in \mathcal{G}_1 $w_b = pnr_4 v dnr_3 pr dnr_3 r_8 r_5 r_6 r_2$ and $w'_b = pnr_4 v dnr_3 r_6 pr dnr_3 r_8 r_7 r_2$, we see that they have shortest common prefixes with one conflict $u_{b,1} = pnr_4 v dnr_3$ and $u'_{b,1} = pnr_4 v dnr_3 r_6$, followed by $u_{b,2} = pr dnr_3 r_8 r_5$ and $u'_{b,2} = pr dnr_3 r_8$, $u_{b,3} = \varepsilon$ and $u'_{b,3} = r_7$, $u_{b,4} = r_6$ and $u'_{b,4} = \varepsilon$ before the longest common suffix r_2 . The following proposition establishes the decomposition we just performed.¹

Proposition 2 *Let w be a sentence of a context-free grammar \mathcal{G} with two different parse trees represented by strings w_b and w'_b in V_b^* : $w = w'$. Let v_b be the longest common suffix of these two strings. Then there exists $t \geq 1$ such that $w_b = u_{b,1} \cdots u_{b,t} v_b$ and $w'_b = u'_{b,1} \cdots u'_{b,t} v_b$ where $u_{b,1} \cdots u_{b,t}$ and $u'_{b,1} \cdots u'_{b,t}$ are shortest common prefixes of w_b and w'_b with t conflicts.*

The interest of common prefixes with conflicts is that the common parts in T_b^* up to the conflicts could just as easily be read using nonterminals. In

¹Note that such a decomposition is not unique: we could have chosen for instance $u_{b,3} = r_6$ and $u'_{b,3} = \varepsilon$. This is not an issue since we can define a canonical order.

our previous example, we could reach the same points as with $u_{b,1}$ and $u'_{b,1}$ by reading the strings $NP v NP$ and $NP v NP r_6$ respectively.

3.2 Position Accessibility

We implement the idea of prefixes with t conflicts in the mutual accessibility relations classically used to find common prefixes [28, Chapter 10]. Mutual accessibility relations are defined on couples of states, and used to identify couples of states accessible upon reading the same language. We only have to find couples accessible 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, avoiding the potential exponential blowup of determinizing our position automaton.

General Relations Lemma 1 allows us to disregard a reduction transition on a r_i symbol, for a given $i = A \rightarrow \alpha$, provided we considered the transition on the nonterminal A in our mutual accessibility relations. In order to be able to invoke Lemma 1, we have to make sure that we really were in position to follow this transition on the nonterminal A . Due to the conflicts, this is not always granted.

We therefore remember whenever we followed an ε -transition after a conflict. We consider for our mutual accessibility relations pairs over $\mathbb{B} \times Q$ instead of Q : the boolean value tells whether we followed an ε -transition after 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 $\setminus q$ 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.

Definition 3 *The primitive mutual accessibility relations defined over $\mathbb{B} \times Q$ are*

shift mas defined by $(q_1, q_2) \text{mas} (q_3, q_4)$ 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 $\text{mae} = \text{mael} \cup \text{maer}$ where $(q_1, q_2) \text{mael} (q_3, q_4)$ if and only if $q_1 d_i \vdash q_3$ and $\setminus q_3$ and symmetrically for maer , $(q_1, q_2) \text{maer} (q_1, q_4)$ if and only if $q_2 d_i \vdash q_4$, and $\setminus q_4$,

reduction mar defined by $(q_1, q_2) \text{mar} (q_3, q_4)$ 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 $\neg \setminus q_1$ or $\neg \setminus q_2$, and then $\neg \setminus q_3$ and $\neg \setminus q_4$,

conflict $\text{mac} = \text{macl} \cup \text{macr}$ with $(q_1, q_2) \text{macl} (q_3, q_2)$ if and only if there exist i in P , q_4 in Q and a in $T'_b - \{r_i\}$ such that $q_1 r_i \vdash q_3$, $q_2 a \vDash^+ q_4$ and $\neg \setminus q_3$, and symmetrically for macr , $(q_1, q_2) \text{macr} (q_1, q_4)$ if and only if there exist i in P , q_3 in Q and a in $T'_b - \{r_i\}$ such that $q_2 r_i \vdash q_4$, $q_1 a \vDash^+ q_3$, and $\neg \setminus q_4$.

The prefix mutual accessibility relation map is defined as the union $\text{mas} \cup \text{mae} \cup \text{mar}$; the global mutual accessibility relation ma is defined as $\text{ma} = \text{map} \cup \text{mac}$.

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 between a reduction and another action. More precisely, the conditions of **mac** are adapted from the LR(k) conflict conditions using the EFF_k sets of Aho and Ullman [2].

Ambiguity Detection We explicit the relation between **ma** and common prefixes with t conflicts in the following lemma.

Lemma 2 *Let w_b and w'_b be two different sentences of \mathcal{G}_b with a pair of shortest common prefixes with t conflicts $u_{b,1} \cdots u_{b,t}$ and $u'_{b,1} \cdots u'_{b,t}$. Furthermore, let ν_s and ν'_s be the corresponding starting positions in μ_s , and $u_{b,t} = u_b r_i$ and $u'_{b,t} = u_b$.*

Then, there exist ν_r , ν_t and ν'_t in \mathcal{N} with

$$(i) \ \nu_s \xrightarrow{u_{b,1} \cdots u_{b,t-1} u_b} \nu_r \xrightarrow{r_i} \nu_t \text{ and } \nu'_s \xrightarrow{u'_{b,1} \cdots u'_{b,t-1} u_b} \nu'_t,$$

$$(ii) \ ([\nu_s]_{\equiv}, [\nu'_s]_{\equiv}) (\text{mas} \cup \text{mae})^* \circ (\text{mac} \circ \text{map}^*)^{t-1} ([\nu_r]_{\equiv}, [\nu'_t]_{\equiv}), \text{ and}$$

$$(iii) \ ([\nu_r]_{\equiv}, [\nu'_t]_{\equiv}) \text{mac} ([\nu_t]_{\equiv}, [\nu'_t]_{\equiv}).$$

Proof. We first note that (i) always holds in Γ , and that together with the fact that $u_{b,1} \cdots u_{b,t}$ and $u'_{b,1} \cdots u'_{b,t}$ are longest common prefixes with t conflicts, it implies that (iii) holds. Let us then prove (ii) by induction on t .

We can show using a simple induction on the length $|u_b|$ that, for $t = 1$, the common prefix u_b is such that $([\nu_s]_{\equiv}, [\nu'_s]_{\equiv}) (\text{mas} \cup \text{mae})^* ([\nu_r]_{\equiv}, [\nu'_t]_{\equiv})$. If this length is zero, then $([\nu_s]_{\equiv}, [\nu'_s]_{\equiv}) \text{mae}^* ([\nu_r]_{\equiv}, [\nu'_t]_{\equiv})$ and thus (ii) holds. We then consider two atomic ways to increase this length while keeping u_b a common prefix: add an a symbol or an r_i symbol. The first case is clearly handled by **mas**. In the second case, using Lemma 1, there exist ν_A and ν'_A in \mathcal{N} such that $\nu_s \xrightarrow{v_b} \nu_A \xrightarrow{A} \nu_r$ and $\nu'_s \xrightarrow{v_b} \nu'_A \xrightarrow{A} \nu'_t$. Applying the induction hypothesis, we see that $([\nu_s]_{\equiv}, [\nu'_s]_{\equiv}) (\text{mas} \cup \text{mae})^* ([\nu_A]_{\equiv}, [\nu'_A]_{\equiv})$, and since furthermore $([\nu_A]_{\equiv}, [\nu'_A]_{\equiv}) \text{mas} ([\nu_r]_{\equiv}, [\nu'_t]_{\equiv})$, (ii) holds for ν_r and ν'_t .

Let us now prove the induction step for $t > 1$. By induction hypothesis and (iii), $([\nu_s]_{\equiv}, [\nu'_s]_{\equiv}) (\text{mas} \cup \text{mae})^* \circ (\text{mac} \circ \text{map}^*)^{t-2} \circ \text{mac} ([\nu_{t-1}]_{\equiv}, [\nu'_{t-1}]_{\equiv})$, and we only need to prove that $([\nu_{t-1}]_{\equiv}, [\nu'_{t-1}]_{\equiv}) \text{map}^* ([\nu_r]_{\equiv}, [\nu'_t]_{\equiv})$. We proceed again by induction on the length of the common prefix u_b . The initial step for $|u_b| = 0$ is clear, and the induction step where we add an a symbol also. The case where we add an r_i symbol triggers the use of **mar** if at least one of the two states verifies $\neg \downarrow q$. Otherwise, we did not follow any r_i transition since the last ε one, and thus Lemma 1 applies. In all cases, (ii) holds. \square

We just need to combine Lemma 2 with Proposition 2 in order to prove our main result:

Definition 4 *Let \mathcal{G} be a context-free grammar and Γ/\equiv its nondeterministic position automaton using \equiv as position equivalence relation; \mathcal{G} is \equiv -ambiguous if $(q_s, q_s) (\text{mae} \cup \text{mas})^* \circ \text{mac} \circ \text{ma}^* (q_f, q_f)$ holds in Γ/\equiv .*

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 \equiv -ambiguous.*

3.3 Complexity

The complexity of our algorithm depends mostly of the equivalence relation we choose to quotient the position graph. Supposing that we are in the favorable case where \equiv is of finite index and of decidable computation of complexity $\mathcal{C}(\Gamma/\equiv)$, we then need to build the image $\text{ma}^* (\{(q_s, q_s)\})$. 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 $\mathcal{O}(\mathcal{C}(\Gamma/\equiv) + |\Gamma/\equiv|^2 |P|^2)$.

The complexity $\mathcal{C}(\Gamma/\text{item}_0)$ 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 $\mathcal{O}(|\mathcal{G}|^2)$.

4 Formal Comparisons

Our detection scheme forbids any false negative result, but in practice we would also like to avoid as many false positives as possible. Many equivalence relations give rather poor results in this regard, for they allow very different positions to end in the same equivalence class. The relation item_0 is reasonably accurate, and if necessary finer equivalence relations will return less false positives.

We compare here our ambiguity detection algorithm with the other means to test a context-free grammar for ambiguity we are aware of.

4.1 Regular Approximations

The simple ambiguity detection algorithm discussed at the beginning of Section 3 and by the author in [26] is based on Theorem 1: any ambiguity in the original grammar is reflected in the regular superset $\mathcal{L}(\Gamma/\equiv) \cap T_b^*$. The quotienting into Γ/\equiv is a generalization of the techniques used to build regular superset approximations of context-free languages [23]. Such approximations have been applied for instance to LR-Regular lookahead computations [3, 12] or to guided parsing [4].

Let us show that our ambiguity detection algorithm performs better than the regular approximation one. We call a context-free grammar with two different sentences w_b and w'_b in $\mathcal{L}(\Gamma/\equiv) \cap T_b^*$ with $w = w'$ *regular \equiv -ambiguous*.

Lemma 3 *Let q_1, q_2, q_3 and q_4 be states in Q such that $(q_1, q_2) \text{ma}^* (q_3, q_4)$. Then, there exist u_b and u'_b in T_b^* such that $u = u'$, $q_1 u_b \vDash^* q_3$ and $q_2 u'_b \vDash^* q_4$.*

Proof. We proceed by induction on the number of steps n in $(q_1, q_2) \text{ma}^n (q_3, q_4)$. If $n = 0$, then $q_1 = q_3$ and $q_2 = q_4$, hence $u_b = u'_b = \varepsilon$ fit our requirements.

Let us prove the induction step. Suppose we have two states q_5 and q_6 such that $(q_3, q_4) \text{ma} (q_5, q_6)$, and, using the induction hypothesis, two strings u_b and u'_b in T_b^* such that $u = u'$, $q_1 u_b \vDash^* q_3$ and $q_2 u'_b \vDash^* q_4$. Let us find v_b and v'_b two strings in T_b^* such that $v = v'$, $q_3 v_b \vDash^* q_5$ and $q_4 v'_b \vDash^* q_6$ for each of the primitive mutual accessibility relations. For mas , $v_b = v'_b$ such that $X \Rightarrow^* v_b$ in \mathcal{G}_b fit; for mae , $v_b = v'_b = \varepsilon$ do; for mar , $v_b = v'_b = r_i$ do; at last, for macl , $v_b = r_i$ and $v'_b = \varepsilon$ do and symmetrically for macr . \square

Theorem 3 *If \mathcal{G} is \equiv -ambiguous, then it is also regular \equiv -ambiguous.*

Proof. Since the relation $(\text{mae} \cup \text{mas})^* \circ \text{mac} \circ \text{ma}^*$ of Definition 4 is included in ma^* , Lemma 3 also applies to it. Therefore, if \mathcal{G} is \equiv -ambiguous, then there are two strings u_b and u'_b in T_b^* such that $u = u'$ and $q_s u_b \models^* q_f$ and $q_s u'_b \models^* q_f$, i.e. u_b and u'_b are in $\mathcal{L}(\Gamma/\equiv) \cap T_b^*$. Note that the presence of the first occurrence of mac in the relation implies that the two bracketed strings u_b and u'_b are different, which concludes the proof. \square

The following grammar is LR(0) and not item_0 -ambiguous:

$$S \xrightarrow{2} aAa, S \xrightarrow{3} bAa, A \xrightarrow{4} c. \quad (\mathcal{G}_2)$$

Nevertheless, \mathcal{G}_2 is regular item_0 -ambiguous: the sentences $acr_4 ar_2$ and $acr_4 ar_3$ are in $\mathcal{L}(\Gamma_2/\text{item}_0) \cap T_b^*$. Our algorithm performs strictly better than the regular superset algorithm.

4.2 Bounded Length Detection Schemes

To the best of our knowledge, all the other algorithms that have been specifically designed so far for ambiguity detection look for ambiguities in all sentences up to some length [14, 7, 27, 20]. 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.

Prefix Equivalence Relations The same guarantee is offered by the equivalence relation prefix_m defined for any fixed length m by²

$$x_b \binom{\alpha}{u_b} \cdot \binom{\alpha'}{u'_b} r_i x'_b \text{ prefix}_m y_b \binom{\beta}{v_b} \cdot \binom{\beta'}{v'_b} r_j y'_b \text{ iff } m :_b x_b u_b = m :_b y_b v_b. \quad (6)$$

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

Lemma 4 *Let q be a state in Q and w_b be a string in T_b^* such that $q_s w_b \models^* q$ in Γ/prefix_m . If $|w| \leq m$, then for all ν in q , there exists ν_s in μ_s such that $\nu_s \xrightarrow{w_b} \nu$.*

Proof. We proceed by induction on the number n of steps in $q_s w_b \models^n q$. If $n = 1$, then $w_b = \varepsilon$ and any ν in q is also in μ_s , and the lemma holds.

For the induction step, we consider $q_s w_b \chi \models^{n-1} q \chi \models q'$ with χ in $T_b \cup \{\varepsilon\}$. For all ν' in q' , there exists ν such that $\nu \xrightarrow{\chi} \nu'$. Since $|w| \leq m$, ν is in q , and we can invoke the induction hypothesis to find an appropriate ν_s in μ_s such that $\nu_s \xrightarrow{w_b} \nu \xrightarrow{\chi} \nu'$. \square

Lemma 4 yields immediately the following theorem.

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

Our ambiguity detection algorithm being a refinement of regular \equiv -ambiguity, Theorem 4 implies that ambiguities in sentences of lengths smaller than m are always actual ambiguities.

² 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 $|x| \leq m$.

Discussion 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 with rules

$$S \rightarrow A | B_n, A \rightarrow Aaa | a, B_1 \rightarrow aa, B_2 \rightarrow B_1 B_1, \dots, B_n \rightarrow B_{n-1} B_{n-1} \quad (\mathcal{G}_3)$$

and \mathcal{G}_4^n with rules

$$S \rightarrow A | B_n a, A \rightarrow Aaa | a, B_1 \rightarrow aa, B_2 \rightarrow B_1 B_1, \dots, B_n \rightarrow B_{n-1} B_{n-1}, \quad (\mathcal{G}_4)$$

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 will correctly find \mathcal{G}_3^n unambiguous and \mathcal{G}_4^n ambiguous in time $\mathcal{O}(n^2)$ using item_0 .

4.3 LR(k) and LRR 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 [25]. The LR(k) testing algorithms [21, 18, 19] are much more efficient in the worst case and provided our initial inspiration. Our nondeterministic position automaton can be seen as a generalization of the item grammar or nondeterministic automaton of these works, and our test looks for ambiguities instead of LR conflicts.

One of the strongest ambiguity tests available is the LR-Regular condition [10, 15]: 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^* , a grammar \mathcal{G} is LR(Π) if and only if

$$S \xRightarrow{\text{rm}}^* \delta A x \xRightarrow{\text{rm}} \delta \alpha x, S \xRightarrow{\text{rm}}^* \gamma B y \xRightarrow{\text{rm}} \gamma \beta y = \delta \alpha z \text{ and } x \cong z \pmod{\Pi} \quad (7)$$

implies

$$A \rightarrow \alpha = B \rightarrow \beta, \delta = \gamma \text{ and } y = z. \quad (8)$$

Moreover, the condition relies on the fact that Π defines a *left congruence* \cong for string concatenation (if this is not the case, a refinement of Π which is also a left congruence can always be constructed and used instead).

This definition is a proper generalization of the LR(k) condition. Practical implementations [3, 12] of the LRR parsing method actually compute, for each inadequate LR state, a finite state automaton that attempts to discriminate between the x and z regular lookaheads. The final states of this automaton act as the partitions of Π .

LR(Π) Equivalence Relations We show here that our test for ambiguity is strictly stronger than the LR(Π) condition when one uses the equivalence relation $\text{item}_{\Pi} = \text{item}_0 \wedge \text{look}_{\Pi}$, where look_{Π} is defined by

$$x_b \binom{\alpha}{u_b} \cdot \binom{\alpha'}{u'_b} r_i x'_b \text{ look}_{\Pi} y_b \binom{\beta}{v_b} \cdot \binom{\beta'}{v'_b} r_j y'_b \text{ iff } u' x' \cong v' y' \pmod{\Pi}. \quad (9)$$

The two following lemmas show some immediate properties of look_{Π} and item_0 .

Lemma 5 *Let $\nu = x_b(\frac{\alpha}{u_b} \cdot \frac{\alpha'}{u'_b})r_i x'_b$ be a position in \mathcal{N} and w_b a bracketed string in T_b^* . If $[\nu]_{\text{look}_\Pi} w_b \models^* q_f$ in Γ/look_Π , then $w\$ \cong u'x' \pmod{\Pi}$.*

Proof. We proceed by induction on n the number of steps in $[\nu]_{\text{look}_\Pi} w_b \models^n q_f$. If $n = 1$, then ν is in μ_f , $w_b = u'_b = \varepsilon$ and $x'_b = \$$. We consider for the induction step the path $[\nu]_{\text{look}_\Pi} \chi w_b \models q w_b \models^n q_f$, where χ is in $T_b \cup \{\varepsilon\}$. Using the induction hypothesis, any $\nu' = y_b \beta (v_b \cdot v_b) \beta' r_j y' b$ in q is such that $w\$ \cong v'y' \pmod{\Pi}$.

If $\chi = \varepsilon$, any ν' in q is such that $\beta = v_b = \varepsilon$ and $v'y' \cong u'x' \pmod{\Pi}$, and the lemma holds trivially by transitivity of \cong . If $\chi = a$, then $\nu = x_b \alpha (u_b \cdot a u'_b) \alpha r_i x'_b$ and any ν' in q is such that $v'y' \cong u'x' \pmod{\Pi}$. Since Π is a left congruence, $w\$ \cong u'x' \pmod{\Pi}$ implies that $aw\$ \cong au'x' \pmod{\Pi}$ and the lemma holds. If $\chi = r_i$, then $\nu = x_b \alpha (u_b \cdot r_i) r_i x'_b$ and any ν' in q is such that $v'y' \cong x' \pmod{\Pi}$, and the lemma holds. \square

Lemma 6 *Let $\nu = x_b(\frac{\alpha}{u_b} \cdot \frac{\alpha'}{u'_b})r_i x'_b$ be a position in \mathcal{N} and γ a string in V^* . If $q_s \gamma \models^* [\nu]_{\text{item}_0}$ in Γ/item_0 , then $A \xrightarrow{i} \alpha \cdot \alpha'$ is a valid LR(0) item for γ , i.e. $S \xrightarrow{\text{rm}}^* \delta A z \xrightarrow{\text{rm}}^i \delta \alpha \alpha' z = \gamma \alpha' z$ holds in \mathcal{G} .*

Proof. We proceed by induction on the number n of steps in $q_s \gamma \models^n [\nu]_{\text{item}_0}$. If $n = 1$, then $S \rightarrow \cdot \alpha'$ is a valid LR(0) item for $\gamma = \varepsilon$. Let us consider for the induction step $q_s \gamma \chi \models^n [\nu]_{\text{item}_0}$ with χ in $V \cup \{\varepsilon\}$, where $S \xrightarrow{\text{rm}}^* \delta A z \xrightarrow{\text{rm}}^i \delta \alpha \alpha' z = \gamma \alpha' z$ holds in \mathcal{G} .

If $\chi = \varepsilon$, then $\alpha' = B \alpha''$ and ν' is of form $y_b(\cdot v_b) \beta r_j y'_b$ for some $B \xrightarrow{j} \beta$ in P . Then, $\gamma \alpha' z = \gamma B \alpha'' z \xrightarrow{\text{rm}}^j \gamma \beta \alpha'' z$ holds in \mathcal{G} and $B \xrightarrow{j} \cdot \beta$ is a valid LR(0) item for $\gamma \chi$. If χ is in V , then $\alpha' = \chi \alpha''$ and ν' is of form $x_b \alpha \chi (u_b v_b \cdot u'_b) \alpha'' r_i x'_b$. Then, $\gamma \alpha' z = \gamma \chi \alpha'' z$ and $A \xrightarrow{i} \alpha \chi \cdot \alpha''$ is a valid LR(0) item for $\gamma \chi$. \square

Theorem 5 *If \mathcal{G} is item_Π -ambiguous, then it is not LR(Π).*

Proof. Let us suppose that \mathcal{G} is item_Π -ambiguous. By Definition 4, we have the relation

$$(q_s, q_s) (\text{mas} \cup \text{mae})^* (q_1, q_2) \text{mac} (q_3, q_2) \text{ma}^* (q_f, q_f). \quad (10)$$

Let ν_1 and ν_2 be two positions in q_1 and q_2 at the origin of the mac relation. We suppose ν_2 is of general form $y_b(\frac{\beta}{v_b} \cdot \frac{\beta'}{v'_b})r_j y'_b$. Relation $(q_1, q_2) \text{mac} (q_3, q_2)$ indicates that $q_1 r_i \vdash q_3$ for some i in P : ν_1 is necessarily of form $x_b(\frac{\alpha}{u_b} \cdot \cdot) r_i x'_b$. The relation also forbids r_i to be equal to $1 : v' r_j$.

Clearly, the first part $(q_s, \cdot q_s) (\text{mas} \cup \text{mae})^* (q_1, q_2)$ of Equation (10) implies that $q_s \delta \alpha \models^* q_1$ and $q_s \gamma \beta \models^* q_2$ with $\delta \alpha = \gamma \beta$ for some δ and γ in V^* . By Lemma 6,

$$S \xrightarrow{\text{rm}}^* \delta A x' \xrightarrow{\text{rm}}^i \delta \alpha x' \text{ and } S \xrightarrow{\text{rm}}^* \gamma B y' \xrightarrow{\text{rm}}^j \gamma \beta \beta' y' = \delta \alpha \beta' y' \quad (11)$$

hold in \mathcal{G} .

Let us now consider the second part $(q_1, q_2) \text{mac} (q_3, q_2) \text{ma}^* (q_f, q_f)$ of Equation (10). By Lemma 3, there exist two bracketed strings w_b and w'_b with $w = w'$ such that $q_1 w_b \models^* q_f$ and $q_2 w'_b \models^* q_f$. By Lemma 5, $x' \cong w \pmod{\Pi}$ and $v'y' \cong w' \pmod{\Pi}$ and by transitivity

$$v'y' \cong x' \pmod{\Pi}. \quad (12)$$

We follow now the classical argument of Aho and Ullman [2, Theorem 5.9] and study the cases where β' is ε , in T^+ , or contains a nonterminal as a factor.

If $\beta' = \varepsilon$, then our Equations (11) and (12) fit the requirements of Equation (7). Nevertheless, $v' = \varepsilon$ and $r_i \neq 1 : v'_b r_j$ thus implies $i \neq j$, violating the requirements of Equation (8). If $\beta' = v'$ is in T^+ , then once again we fit the requirements of Equation (7). Nevertheless, in this case, $y' \neq v' y'$, hence violating the requirements of Equation (8). If there is at least one nonterminal in β' , then

$$S' \xrightarrow{\text{rm}}^* \gamma \beta v_1 C v_3 y' \xrightarrow{\text{rm}} \gamma \beta v_1 v_2 v_3 y' = \delta \alpha v_1 v_2 v_3 y' = \delta \alpha v' y'. \quad (13)$$

Remember that $r_i \neq 1 : v'_b r_j$, thus $v_1 v_2 \neq \varepsilon$ and Equation (8) cannot hold. \square

Let us consider now the grammar with rules

$$S \rightarrow AC \mid BCb, A \rightarrow a, B \rightarrow a, C \rightarrow cCb \mid cb. \quad (\mathcal{G}_5)$$

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 Γ_5 / item_0 will show that \mathcal{G}_5 is not ambiguous: our algorithm is thus strictly better than the LRR condition.

4.4 Horizontal and Vertical Ambiguity

Brabrand *et al.* [5] 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.

Definition 5 ([5]) *A context-free grammar is vertically unambiguous if and only if, for all A in N with two different productions $A \rightarrow \alpha_1$ and $A \rightarrow \alpha_2$ in P , $\mathcal{L}(\alpha_1) \cap \mathcal{L}(\alpha_2) = \emptyset$.*

It is horizontally unambiguous if and only if, for all productions $A \rightarrow \alpha$ in P , and for all decompositions $\alpha = \alpha_1 \alpha_2$, $\mathcal{L}(\alpha_1) \bowtie \mathcal{L}(\alpha_2) = \emptyset$, where \bowtie is the language overlap operator defined by $L_1 \bowtie L_2 = \{xay \mid x, xa \in L_1 \text{ and } y, ay \in L_2\}$.

The ambiguity detection method of Brabrand *et al.* 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. Let us compare the horizontal and vertical criteria with regular \equiv -ambiguity.

Definition 6 *The automaton Γ / \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*

$$\begin{aligned} [x_b(\cdot \overset{\alpha_1}{u_b})r_i x'_b]_{\equiv} w_b \models^* [x_b(\overset{\alpha_1}{u_b} \cdot) r_i x'_b]_{\equiv} \text{ and} \\ [x_b(\cdot \overset{\alpha_2}{u'_b})r_j x'_b]_{\equiv} w'_b \models^* [x_b(\overset{\alpha_2}{u'_b} \cdot) r_j x'_b]_{\equiv}. \end{aligned}$$

The automaton Γ / \equiv is horizontally ambiguous if and only if there exist a production $A \xrightarrow{i} \alpha$ in P , a decomposition $\alpha = \alpha_1 \alpha_2$, a symbol a in T , and the

bracketed strings $x_b, x'_b, u_b, u'_b, v_b, v'_b, w_b,$ and w'_b in T_b^* with $v = v'$ and $w = w'$ such that

$$\begin{aligned} [x_b(\cdot \begin{smallmatrix} \alpha_1 & \alpha_2 \\ u_b & u'_b \end{smallmatrix})r_i x'_b]_{\equiv} v_b a w_b \vDash^* [x_b(\begin{smallmatrix} \alpha_1 \\ u_b \end{smallmatrix} \cdot \begin{smallmatrix} \alpha_2 \\ u'_b \end{smallmatrix})r_i x'_b]_{\equiv} a w_b \vDash^* [x_b(\begin{smallmatrix} \alpha_1 & \alpha_2 \\ u_b & u'_b \end{smallmatrix} \cdot)r_i x'_b]_{\equiv} \text{ and} \\ [x_b(\cdot \begin{smallmatrix} \alpha_1 & \alpha_2 \\ u_b & u'_b \end{smallmatrix})r_i x'_b]_{\equiv} v'_b a w'_b \vDash^* [x_b(\begin{smallmatrix} \alpha_1 \\ u_b \end{smallmatrix} \cdot \begin{smallmatrix} \alpha_2 \\ u'_b \end{smallmatrix})r_i x'_b]_{\equiv} w'_b \vDash^* [x_b(\begin{smallmatrix} \alpha_1 & \alpha_2 \\ u_b & u'_b \end{smallmatrix} \cdot)r_i x'_b]_{\equiv}. \end{aligned}$$

Theorem 6 *Let \mathcal{G} be a context-free grammar and Γ/\equiv its position automaton. If Γ/\equiv is horizontally ambiguous or vertically ambiguous, then \mathcal{G} is regular \equiv -ambiguous.*

Proof. If Γ/\equiv is vertically ambiguous, then $x_b w_b r_i x'_b$ and $x_b w'_b r_j x'_b$ are two different sentences in $\mathcal{L}(\Gamma/\equiv) \cap T_b^*$ with $x w x' = x w' x'$, and thus \mathcal{G} is regular \equiv -ambiguous. If Γ/\equiv is horizontally ambiguous, then $x_b v_b a w_b r_i x'_b$ and $x_b v'_b a w'_b r_i x'_b$ are two different sentences in $\mathcal{L}(\Gamma/\equiv) \cap T_b^*$ with $x v a w x' = x v' a w' x'$, and thus \mathcal{G} is regular \equiv -ambiguous. \square

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, with the regular approximation of Mohri and Nederhof [22].

Owing to these criteria, the technique of Brabrand *et al.* accomplishes to show that the palindrome grammar \mathcal{G}_6 with rules

$$S \rightarrow a S a | b S b | a | b | \varepsilon \tag{\mathcal{G}_6}$$

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 \mathcal{G}_7 with rules

$$S \rightarrow A A, A \rightarrow a A a | b \tag{\mathcal{G}_7}$$

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

5 Related Work

5.1 Noncanonical Parsers

Noncanonical parser constructions are amongst the most powerful deterministic parser constructions available and as such provide very powerful tests for the ambiguity of a grammar. Grammar \mathcal{G}_5 for instance is NSLR(2) [29], and thus cannot be ambiguous.

Nevertheless, most noncanonical techniques are defined by their construction mechanisms: if the resulting parser is deterministic, then the grammar is not ambiguous. The computational cost of an attempt to build a deterministic parser can be exponential in the size of the grammar. Our strategy of using nonterminal transitions as often as possible is inspired by the noncanonical constructions, but we maintain a reasonable worst-case complexity by avoiding the determinization phase altogether. In this regard, practical tests ought to be performed in order to verify the edge of the mutual accessibility relations: the blowup of determinization is unlikely to occur in practice [24].

The techniques used in this paper have also found their application in the construction of noncanonical parsers: the nondeterministic position automaton can be determinized with a subset construction to yield a parser [13].

5.2 Grammatical Representations

The nondeterministic position automaton we introduced in this paper can be seen as a generalization of several similar representations defined in different contexts, for instance $\vee C$ -flow graphs [14], item grammars [16], transition diagrams or networks [9, 30], nondeterministic LR automata [18], or item graphs [12]. Besides testing, these representations have been applied to parser generation [16, 11, 13] and to regular approximations of context-free languages [23].

5.3 Verification of Programs

Context-free grammars can also model programs running mutually recursive processes—they are better known as normed Basic Process Algebra equations. The equivalence of such programs using the bisimulation semantics can be tested in polynomial time [17]. Nevertheless, ambiguity checking needs completed trace semantics, and it is unclear whether any suitable solution exists in the verification literature.

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 first conservative tests explicitly aimed towards ambiguity checking, along with the recent work of Brabrand *et al.* [5].

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 [13] where the amount of lookahead needed is not preset but computed for each parsing decision. We hope to see more applications of this general scheme 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.

References

- [1] *ASD Simplified Technical English*. AeroSpace and Defence Industries Association of Europe, 2005. Specification ASD-STE100.
- [2] Alfred V. Aho and Jeffrey D. Ullman. *The Theory of Parsing, Translation, and Compiling. Volume I: Parsing*. Series in Automatic Computation. Prentice Hall, 1972. ISBN 0-13-914556-7.
- [3] Manuel E. Bermudez and Karl M. Schimpf. Practical arbitrary lookahead LR parsing. *Journal of Computer and System Sciences*, 41(2):230–250, 1990. ISSN 0022-0000. doi: 10.1016/0022-0000(90)90037-L.

-
- [4] Pierre Boullier. Guided Earley parsing. In *IWPT'03*, pages 43–54, 2003. URL ftp://ftp.inria.fr/INRIA/Projects/Atoll/Pierre.Boullier/earley_final.pdf.
- [5] Claus Brabrand, Robert Giegerich, and Anders Møller. Analyzing ambiguity of context-free grammars. Technical Report RS-06-09, BRICS, University of Aarhus, May 2006. URL <http://www.brics.dk/~brabrand/grambiguity/>.
- [6] David G. Cantor. On the ambiguity problem of Backus systems. *Journal of the ACM*, 9(4):477–479, 1962. ISSN 0004-5411. doi: 10.1145/321138.321145.
- [7] Bruce S. N. Cheung and Robert C. Uzgalis. Ambiguity in context-free grammars. In *SAC'95*, pages 272–276. ACM Press, 1995. ISBN 0-89791-658-1. doi: 10.1145/315891.315991.
- [8] Noam Chomsky and Marcel Paul Schützenberger. The algebraic theory of context-free languages. In P. Braffort and D. Hirshberg, editors, *Computer Programming and Formal Systems*, Studies in Logic, pages 118–161. North-Holland Publishing, 1963.
- [9] Melvin E. Conway. Design of a separable transition-diagram compiler. *Communications of the ACM*, 6(7):396–408, 1963. ISSN 0001-0782. doi: 10.1145/366663.366704.
- [10] Karel Čulik and Rina Cohen. LR-Regular grammars—an extension of LR(k) grammars. *Journal of Computer and System Sciences*, 7:66–96, 1973. ISSN 0022-0000.
- [11] Charles Donnelly and Richard Stallman. *Bison version 2.1*, September 2005. URL <http://www.gnu.org/software/bison/manual/>.
- [12] Jacques Farré and José Fortes Gálvez. A bounded-connect construction for LR-Regular parsers. In Reinhard Wilhelm, editor, *CC'01*, volume 2027 of *Lecture Notes in Computer Science*, pages 244–258. Springer, 2001. URL <http://springerlink.com/content/e3e8g77kxevkyjfd>.
- [13] José Fortes Gálvez, Sylvain Schmitz, and Jacques Farré. Shift-resolve parsing: Simple, linear time, unbounded lookahead. In Oscar H. Ibarra and Hsu-Chun Yen, editors, *CIAA'06*, volume 4094 of *Lecture Notes in Computer Science*, pages 253–264. Springer, 2006. ISBN 3-540-37213-X. doi: 10.1007/11812128_24.
- [14] Saul Gorn. Detection of generative ambiguities in context-free mechanical languages. *Journal of the ACM*, 10(2):196–208, 1963. ISSN 0004-5411. doi: 10.1145/321160.321168.
- [15] Stephan Heilbrunner. Tests for the LR-, LL-, and LC-Regular conditions. *Journal of Computer and System Sciences*, 27(1):1–13, 1983. ISSN 0022-0000. doi: 10.1016/0022-0000(83)90026-0.
- [16] Stephan Heilbrunner. A parsing automata approach to LR theory. *Theoretical Computer Science*, 15(2):117–157, 1981. ISSN 0304-3975. doi: 10.1016/0304-3975(81)90067-0.

- [17] Yoram Hirshfeld, Mark Jerrum, and Faron Moller. A polynomial algorithm for deciding bisimilarity of normed context-free processes. *Theoretical Computer Science*, 158(1–2):143–159, 1996. ISSN 0304-3975. doi: 10.1016/0304-3975(95)00064-X.
- [18] Harry B. Hunt III, Thomas G. Szymanski, and Jeffrey D. Ullman. Operations on sparse relations and efficient algorithms for grammar problems. In *15th Annual Symposium on Switching and Automata Theory*, pages 127–132. IEEE Computer Society, 1974.
- [19] Harry B. Hunt III, Thomas G. Szymanski, and Jeffrey D. Ullman. On the complexity of LR(k) testing. *Communications of the ACM*, 18(12):707–716, 1975. ISSN 0001-0782. doi: 10.1145/361227.361232.
- [20] Saichaitanya Jampana. Exploring the problem of ambiguity in context-free grammars. Master’s thesis, Oklahoma State University, July 2005. URL <http://e-archive.library.okstate.edu/dissertations/AAI1427836/>.
- [21] Donald E. Knuth. On the translation of languages from left to right. *Information and Control*, 8(6):607–639, 1965. ISSN 0019-9958. doi: 10.1016/S0019-9958(65)90426-2.
- [22] Mehryar Mohri and Mark-Jan Nederhof. Regular approximations of context-free grammars through transformation. In Jean-Claude Junqua and Gertjan van Noord, editors, *Robustness in Language and Speech Technology*, chapter 9, pages 153–163. Kluwer Academic Publishers, 2001. ISBN 0-7923-6790-1. URL <http://citeseer.ist.psu.edu/mohri00regular.html>.
- [23] Mark-Jan Nederhof. Regular approximation of CFLs: a grammatical view. In H. Bunt and A. Nijholt, editors, *Advances in Probabilistic and other Parsing Technologies*, chapter 12, pages 221–241. Kluwer Academic Publishers, 2000. ISBN 0-7923-6616-6. URL <http://odur.let.rug.nl/~markjan/publications/2000d.pdf>.
- [24] Paul Purdom. The size of LALR(1) parsers. *BIT Numerical Mathematics*, 14(3):326–337, 1974. ISSN 0006-3835. doi: 10.1007/BF01933232.
- [25] Janina Reeder, Peter Steffen, and Robert Giegerich. Effective ambiguity checking in biosequence analysis. *BMC Bioinformatics*, 6:153, 2005. ISSN 1471-2105. doi: 10.1186/1471-2105-6-153.
- [26] Sylvain Schmitz. Modular syntax demands verification. Technical Report I3S/RR-2006-32-FR, Laboratoire I3S, Université de Nice - Sophia Antipolis, October 2006. URL <http://www.i3s.unice.fr/~mh/RR/2006/RR-06.32-S.SCHMITZ.pdf>.
- [27] Friedrich Wilhelm Schröder. AMBER, an ambiguity checker for context-free grammars. Technical report, compilertools.net, 2001. URL <http://accent.compilertools.net/Amber.html>.
- [28] Seppo Sippu and Eljas Soisalon-Soininen. *Parsing Theory, Vol. II: LR(k) and LL(k) Parsing*, volume 20 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1990. ISBN 3-540-51732-4.

- [29] Kuo-Chung Tai. Noncanonical SLR(1) grammars. *ACM Transactions on Programming Languages and Systems*, 1(2):295–320, 1979. ISSN 0164-0925. doi: 10.1145/357073.357083.
- [30] W. A. Woods. Transition network grammars for natural language analysis. *Communications of the ACM*, 13(10):591–606, 1970. ISSN 0001-0782. doi: 10.1145/355598.362773.