Conservative Ambiguity Detection in Context-Free Grammars

Sylvain Schmitz

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.

Key words: Ambiguity, context-free grammar, verification, 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". 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 [7], RNA analysis [34, 5], controlled natural languages [1], or programming languages [25, 38, 37].

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.

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:

*Published in Lars Arge et al., editors, *ICALP'07*, volume 4596 of *Lecture Notes in Computer Science*, pages 692–703.

ISRN I3S/RR-2006-30-FR

- 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 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 [15, 7, 38, 21] (which are not conservative tests), the LR-Regular condition [10], and the horizontal and vertical ambiguity condition [5] (Section 5); these comparisons rely on the generality of our approximation model.

We report on the experimental results of a prototype implementation of our algorithm in a different publication [37]. 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

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

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.

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^*

[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.

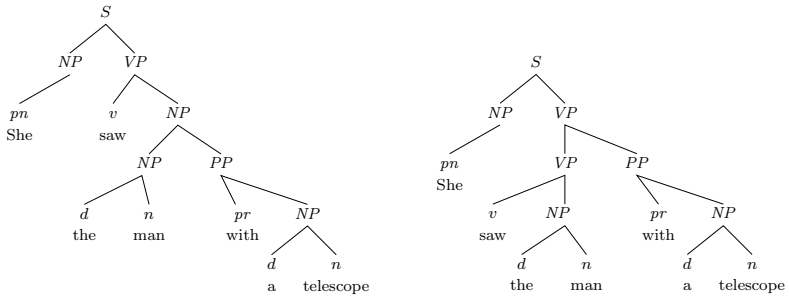

Figure 1: Two trees yielding the sentence "She saw the man with a telescope." with $\mathcal{G}_1$.

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 [29]; 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.

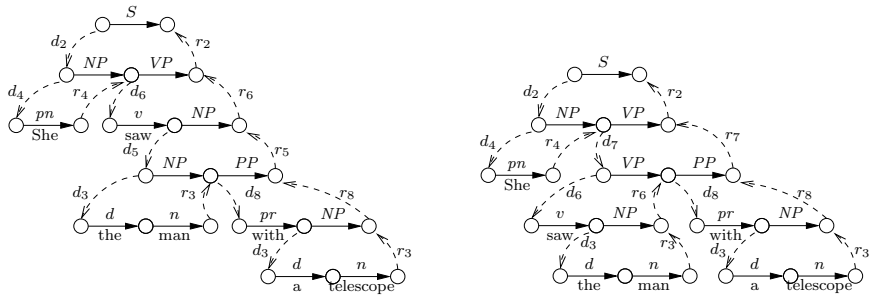

Figure 2: Portions of the position graph of $\mathcal{G}_1$ corresponding to the two trees of Figure 1.

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 $\mu_s = \{d_1(\begin{smallmatrix}S\$\\ w_b\$\end{smallmatrix}) r_1 \mid S \Rightarrow_b^* w_b\}$ 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, $[NP \overset{5}{\rightarrow} NP \bullet PP]$ is the LR(0) item [24] 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 [39].

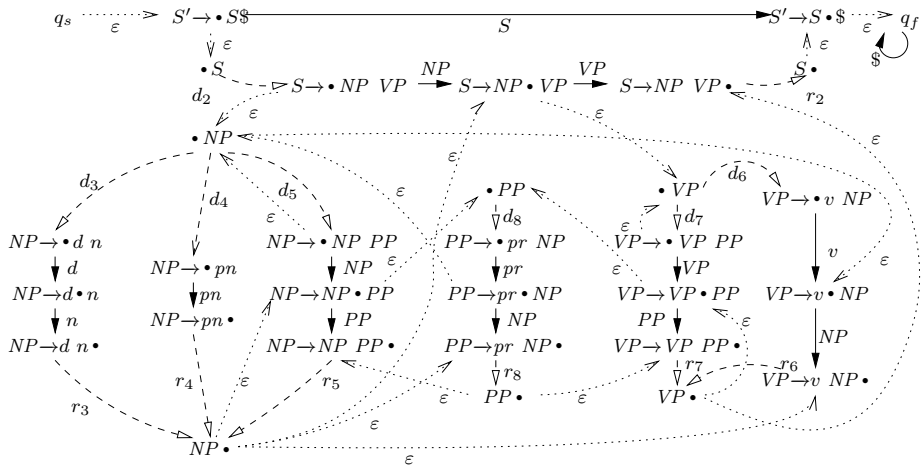

Figure 3: The nondeterministic position automaton for $\mathcal{G}_1$ using item_0.

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* $\{q\chi \vdash q' \mid \exists \nu \in q$ *and* $\nu' \in q', \nu \overset{\chi}{\rightarrow} \nu'\} \cup \{q_s\varepsilon \vdash [\nu_s]_\equiv \mid \nu_s \in \mu_s\} \cup \{[\nu_f]_\equiv \varepsilon \vdash q_f \mid \nu_f \in \mu_f\} \cup \{q_f\$ \vdash q_f\}$*, 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 [19, 20]—the main difference being the presence of the r_i transitions—is item_0 defined by

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

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 [19]. Our position (5) is now in the equivalence class represented by the state labeled by $NP \to NP \bullet PP$ in Figure 3.

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.

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

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

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 [24] 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.

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

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 [42].

4.3 Accessibility Relations

We implement the idea of common prefixes with conflicts in the mutual accessibility relations classically used to find common prefixes [41, 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.

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 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* $\neg \lfloor q_1$ *or* $\neg \lfloor q_2$*, and then* $\neg \lfloor q_3$ *and* $\neg \lfloor q_4$*,*

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* $\neg \lfloor q_3$*, 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* $\neg \lfloor q_4$*.*

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 [15, 7, 38, 21]. 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]

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

Provided that $\mathcal{G}$ is not left recursive, 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

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

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

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 [34]. The LR(k) testing algorithms [24, 19, 20] 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).

One of the strongest ambiguity tests available is the LR-Regular condition [10, 17]: 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 $A{\to}\alpha = B{\to}\beta$, $\delta = \gamma$ 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

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

[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 $|x| \leq m$.

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

Let us consider now the grammar with rules

$$S{\to}AC\,|\,BCb,\ A{\to}a,\ B{\to}a,\ C{\to}cCb\,|\,cb. \qquad (\mathcal{G}_5)$$

Grammar $\mathcal{G}_5$ is not LRR: the right contexts $c^nb^n\$$ and $c^nb^{n+1}\$$ of the reductions using $A{\to}a$ and $B{\to}a$ 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.* [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. 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 [28].

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 $i = A{\to}\alpha$ 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, y_b and y'_b in T_b^* with $v = v'$, $w = w'$, $y = y'$, $|y| \geq 1$ and $v_b y_b w_b \neq v'_b y'_b w'_b$ 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.* [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 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] *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. In Miroslav Balík and Jan Holub, editors, *CIAA'07*, 2007. URL `http://www.brics.dk/~brabrand/grambiguity/`. To appear in *Lecture Notes in Computer Science*.

[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 Donnely 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] Seymour Ginsburg and Michael A. Harrison. Bracketed context-free languages. *Journal of Computer and System Sciences*, 1:1–23, 1967. ISSN 0022-0000.

[15] 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.

[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] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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/`.

[22] Stanislaw Jarzabek and Tomasz Krawczyk. LL-Regular grammars. *Information Processing Letters*, 4(2):31–37, 1975. ISSN 0020-0190. doi: 10.1016/0020-0190(75)90009-5.

[23] Brian W. Kernighan and Dennis M. Ritchie. *The C Programming Language*. Prentice-Hall, 1988. ISBN 0-13-110362-8.

[24] 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.

[25] Werner Kuich. Systems of pushdown acceptors and context-free grammars. *Elektronische Informationsverarbeitung und Kybernetik*, 6(2):95–114, 1970. ISSN 0013-5712.

[26] Scott McPeak and George C. Necula. Elkhound: A fast, practical GLR parser generator. In Evelyn Duesterwald, editor, *CC'04*, volume 2985 of *Lecture Notes in Computer Science*, pages 73–88. Springer, 2004. ISBN 3-540-21297-3. doi: 10.1007/b95956.

[27] Robin Milner, Mads Tofte, Robert Harper, and David MacQueen. *The definition of Standard ML*. MIT Press, revised edition, 1997. ISBN 0-262-63181-4.

[28] 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`.

[29] 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`.

[30] Anton Nijholt. On the parsing of LL-Regular grammars. In Antoni Mazurkiewicz, editor, *MFCS'76*, volume 45 of *Lecture Notes in Computer Science*, pages 446–452. Springer, 1976. ISBN 3-540-07854-1. doi: 10.1007/3-540-07854-1_213.

[31] Terence J. Parr. *ANTLR 3.0 Lookahead Analysis*, 2006. URL `http://www.antlr.org/blog/antlr3/lookahead.tml`.

[32] David A. Poplawski. On LL-Regular grammars. *Journal of Computer and System Sciences*, 18(3):218–227, 1979. ISSN 0022-0000. doi: 10.1016/0022-0000(79)90031-X.

[33] Paul Purdom. The size of LALR(1) parsers. *BIT Numerical Mathematics*, 14(3):326–337, 1974. ISSN 0006-3835. doi: 10.1007/BF01933232.

[34] 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.

[35] Daniel J. Rosenkrantz and Richard E. Stearns. Properties of deterministic top-down grammars. *Information and Control*, 17(3):226–256, 1970. ISSN 0019-9958. doi: 10.1016/S0019-9958(70)90446-8.

[36] 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`.

[37] Sylvain Schmitz. An experimental ambiguity detection tool. In Anthony Sloane and Adrian Johnstone, editors, *LDTA'07*, 2007. URL `http://www.i3s.unice.fr/~mh/RR/2006/RR-06.37-S.SCHMITZ.pdf`. To appear in *Electronic Notes in Theoretical Computer Science*.

[38] Friedrich Wilhelm Schröer. AMBER, an ambiguity checker for context-free grammars. Technical report, compilertools.net, 2001. URL `http://accent.compilertools.net/Amber.html`.

[39] Klaas Sikkel. *Parsing Schemata - a framework for specification and analysis of parsing algorithms*. Texts in Theoretical Computer Science - An EATCS Series. Springer, 1997. ISBN 3-540-61650-0.

[40] Seppo Sippu and Eljas Soisalon-Soininen. On LL(k) parsing. *Information and Control*, 53(3):141–164, 1982. ISSN 0019-9958. doi: 10.1016/S0019-9958(82)91016-6.

[41] 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.

[42] Thomas G. Szymanski and John H. Williams. Noncanonical extensions of bottom-up parsing techniques. *SIAM Journal on Computing*, 5(2):231–250, 1976. ISSN 0097-5397. doi: 10.1137/0205019.

[43] 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.

[44] 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.

A Omitted Material

We gather in the appendices A.3, A.4 and A.5 the technical material and the proofs omitted in the sections 3, 4 and 5 of main body of the paper. We complete our discussion with more related work in Appendix A.6.

A.3 Position Graphs and their Quotients

A.3.1 Position Graphs

Our first lemma relates the various paths between two positions of the position graph with the possible derivations in the original grammar.

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

Proof. Suppose $\delta_b \Rightarrow_b{}^n \gamma_b$; 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 $\delta_b \Rightarrow_b{}^{n-1} \rho_b A \sigma_b \overset{i}{\Rightarrow}_b \rho_b \alpha r_i \sigma_b = \gamma_b$ with $\nu \overset{\rho_b}{\rightarrow} \nu_1 \overset{A}{\rightarrow} \nu_2 \overset{\sigma_b}{\rightarrow} \nu'$. By Definition 1, $\nu_1 \overset{d_i}{\rightarrow} \nu_3 \overset{\alpha}{\rightarrow} \nu_4 \overset{r_i}{\rightarrow} \nu_2$ and thus $\nu \overset{\gamma_b}{\rightarrow} \nu'$.

A similar procedure yields a proof if $\gamma_b \Rightarrow_b^* \delta_b$.□ □

A.3.2 Position Equivalences

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

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

Proposition 1 *Let ν, ν' be positions in $\mathcal{N}$ and δ_b a string in V_b^*. If $\nu \overset{\delta_b}{\rightarrow} \nu'$, 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 $\Gamma/\equiv$, 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 $\nu_s = d_1(\ \bullet\ {}^{S\$}_{w_b\$})\$r_1$ and $\nu_f = d_1({}^{S}_{w_b}\ \bullet\ {}^{\$}_{\$})r_1$; $\nu_s \overset{S}{\rightarrow} \nu_f$. Using the derivation $S \Rightarrow^* w_b$ in $\mathcal{G}_b$ and Lemma 1, we know that $\nu_s \overset{w_b}{\rightarrow} \nu_f$. 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 $\Gamma/\equiv$, *i.e.* w_b is in $\mathcal{L}(\Gamma/\equiv) \cap T_b^*$.□ □

Lattice of Equivalence Relations The usual partial order on $\mathrm{Eq}(\mathcal{N})$—the complete lattice of all equivalence relations on $\mathcal{N}$—is the inclusion relation $\subseteq$. The largest element in $\mathrm{Eq}(\mathcal{N})$, 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 $\bot$. The lattice structure provides two operations for combining equivalence relations into new ones: for any two elements $\equiv_a$ and $\equiv_b$ of $\mathrm{Eq}(\mathcal{N})$,

$\equiv_a \vee \equiv_b$ 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 $\equiv_a \vee \equiv_b = (\equiv_a \circ \equiv_b)^+$ and $\equiv_a \wedge \equiv_b = \equiv_a \cap \equiv_b$.

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 LR(k) precision, see Theorem 5—can be described as $\mathsf{item}_0 \wedge \mathsf{look}_k$ for any $k \geq 0$ where

$$x_b d_i \binom{\alpha}{u_b} \bullet \binom{\alpha'}{u'_b} r_i x'_b \ \mathsf{look}_k \ y_b d_j \binom{\beta}{v_b} \bullet \binom{\beta'}{v'_b} r_j y'_b \text{ iff } k : x' = k : y'. \tag{15}$$

Size of the Nondeterministic Automaton Let us 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 therefore $\Gamma/\top$, of size $|V| + 2|P| + 2$.

A.4 Ambiguity Detection

A.4.1 Common Prefixes with Conflicts

In general, we 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 with the d_i symbols ignored 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.

Using the example of Section 4.2 but keeping the d_i symbols, the pairs of shortest common prefixes with one conflict are successively

$$u_{b,1} = d_1\ d_2\ d_4\ pn\ r_4\ d_6\ v\ d_5\ d_3\ d\ n\ r_3 \text{ and } u'_{b,1} = d_1\ d_2\ d_4\ pn\ r_4\ d_7\ d_6\ v\ d_3\ d\ n\ r_3\ r_6,$$
$$u_{b,2} = d_8\ pr\ d_3\ d\ n\ r_3\ r_8\ r_5 \text{ and } u'_{b,2} = d_8\ pr\ d_3\ d\ n\ r_3\ r_8,$$
$$u_{b,3} = r_6 \text{ and } u'_{b,3} = \varepsilon,$$
$$u_{b,4} = \varepsilon \text{ and } u'_{b,4} = r_7,$$

at which point there only remains a common suffix $v_b = r_2\ \$\ r_1$. With explicit d_i symbols, one can verify that the d_1 and d_2 symbols matching the r_1 and r_2 symbols of v_b are not in v_b, and thus that no reduction could occur inside v_b. Our initial sentences (1) and (2) are decomposed as $u_{b,1}u_{b,2}u_{b,3}u_{b,4}v_b$ and $u'_{b,1}u'_{b,2}u'_{b,3}u'_{b,4}v_b$.

The decomposition is not unique, but that does not hamper the soundness of our algorithm. The following proposition formalizes the decomposition we just performed.

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^*: $w = w'$. 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, and v_b is a common suffix of w_b and w'_b.*□

A.4.2 Accessibility Relations

Let us denote by map the union $\mathsf{mas} \cup \mathsf{mae} \cup \mathsf{mar}$. 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) $\nu_s \xrightarrow{u_{b,1}\cdots u_{b,t-1}u_b} \nu_r \xrightarrow{r_i} \nu_t$ *and* $\nu'_s \xrightarrow{u'_{b,1}\cdots u'_{b,t-1}u_b} \nu'_t$,

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

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

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 $([\nu_s]_\equiv, [\nu'_s]_\equiv)\ (\mathsf{mas} \cup \mathsf{mae})^*\ ([\nu_r]_\equiv, [\nu'_1]_\equiv)$. If this length is zero, then $([\nu_s]_\equiv, [\nu'_s]_\equiv)\ \mathsf{mae}\ ([\nu_r]_\equiv, [\nu'_1]_\equiv)$ and thus *(ii)* holds. We then consider three atomic ways to increase this length while keeping u_b a common prefix: add an a symbol, a d_i, or an r_i symbol. The first two cases are clearly handled by mas and mae. In the third 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 $\nu'_s \xrightarrow{v_b} \nu'_A \xrightarrow{A} \nu'_1$. Applying the induction hypothesis, we see that $([\nu_s]_\equiv, [\nu'_s]_\equiv)\ (\mathsf{mas} \cup \mathsf{mae})^*\ ([\nu_A]_\equiv, [\nu'_A]_\equiv)$, and since furthermore $([\nu_A]_\equiv, [\nu'_A]_\equiv)\ \mathsf{mas}\ ([\nu_r]_\equiv, [\nu'_1]_\equiv)$, *(ii)* holds for ν_r and ν'_1.

Let us now prove the induction step for $t > 1$. By induction hypothesis and *(iii)*, $([\nu_s]_\equiv, [\nu'_s]_\equiv)\ (\mathsf{mas} \cup \mathsf{mae})^* \circ (\mathsf{mac} \circ \mathsf{map}^*)^{t-2} \circ \mathsf{mac}\ ([\nu_{t-1}]_\equiv, [\nu'_{t-1}]_\equiv)$, and we only need to prove that $([\nu_{t-1}]_\equiv, [\nu'_{t-1}]_\equiv)\ \mathsf{map}^*\ ([\nu_r]_\equiv, [\nu'_t]_\equiv)$. 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 or a d_i 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 $\neg \setminus q$. Otherwise, we did not follow any r_i transition since the last d_i one, and thus Lemma 1 applies. In all cases, *(ii)* holds.□ □

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

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($\equiv$).*□

A.5 Formal Comparisons

The lattice of context-free grammar classes inclusions presented in Figure 4 sums up the results of our comparisons.

A.5.1 Regular Ambiguity

The simple algorithm discussed in Section 4.1 and by the author in [36] is based on Theorem 1: any ambiguity in the original grammar is reflected in the

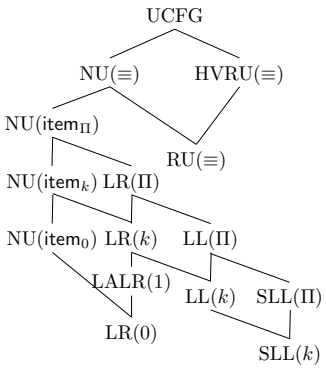

Figure 4: Context-free grammar classes inclusions. The classes parameterized by k, Π and $\equiv$ denote the full classes for any fixed k, any finite regular partition Π of T^*, and any position equivalence relation $\equiv$ with finite index respectively.

regular superset $\mathcal{L}(\Gamma/\equiv) \cap T_b^*$. The quotienting into $\Gamma/\equiv$ is a generalization of the techniques used to build regular superset approximations of context-free languages [29]. Such approximations have been applied for instance to LR-Regular lookahead computations [3, 12] or to guided parsing [4].

Lemma 3 *Let q_1, q_2, q_3 and q_4 be states in Q such that (q_1, q_2) 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) 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) ma (q_4, 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, $v_b = v'_b = d_i$ 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.□ □

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

Proof. Since the relation $(\mathsf{mae} \cup \mathsf{mas})^* \circ \mathsf{mac} \circ \mathsf{ma}^*$ that defines noncanonical $\equiv$-ambiguity is included in ma^*, Lemma 3 also applies to it. Therefore, if $\mathcal{G}$ is noncanonically $\equiv$-ambiguous, then there are two strings u_b and u'_b in T_b^* such that $u = u'$ and $q_s u_b \vDash^* q_f$ and $q_s u'_b \vDash^* q_f$, *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.□ □

A.5.2 Bounded Length Detection Schemes

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

Proof. We proceed by induction on the number n of steps in $q_s w_b \vDash^n q$. 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 $q_s w_b \chi \vDash^{n-1} q\chi \vDash q'$ with χ in T_b. For all ν' in q', there exists ν such that $\nu \overset{\chi}{\rightarrowtail} \nu'$. 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 $\nu_s \overset{w_b}{\rightarrowtail} \nu \overset{\chi}{\rightarrowtail} \nu'$.□ □

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

A.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 [34]. The LR(k) testing algorithms [24, 19, 20] 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.

The stronger LRR condition relies on a partition Π that 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). Then, a grammar $\mathcal{G}$ is LR(Π) if and only if

$$S \underset{\text{rm}}{\Longrightarrow}{}^* \delta A x \underset{\text{rm}}{\Longrightarrow} \delta\alpha x, S \underset{\text{rm}}{\Longrightarrow}{}^* \gamma B y \underset{\text{rm}}{\Longrightarrow} \gamma\beta y = \delta\alpha z \text{ and } x \cong z \,(\text{mod}\,\Pi) \qquad (16)$$

implies

$$A{\rightarrow}\alpha = B{\rightarrow}\beta, \delta = \gamma \text{ and } y = z. \qquad (17)$$

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.

The two following lemmas show some immediate properties of look_Π and item_0.

Lemma 5 *Let $\nu = x_b d_i(\begin{smallmatrix}\alpha \\ u_b\end{smallmatrix} \bullet \begin{smallmatrix}\alpha' \\ u'_b\end{smallmatrix}) r_i x'_b$ be a position in $\mathcal{N}$ and w_b a bracketed string in T_b^*. If $[\nu]_{\mathsf{look}_\Pi} w_b \vDash^* q_f$ in Γ/look_Π, then $w\$ \cong u'x'$ (mod Π).*

Proof. We proceed by induction on n the number of steps in $[\nu]_{\mathsf{look}_\Pi} w_b \vDash^n q_f$. 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 $[\nu]_{\mathsf{look}_\Pi} \chi w_b \vDash q w_b \vDash^n q_f$, where χ is in T_b. Using the induction hypothesis, any $\nu' = y_b d_j(\begin{smallmatrix}\beta \\ v_b\end{smallmatrix} \bullet \begin{smallmatrix}\beta' \\ v_b\end{smallmatrix}) r_j y' b$ in q is such that $w\$ \cong v'y'$ (mod Π).

If $\chi \in T_d$, any ν' in q is such that $\beta = v_b = \varepsilon$ and $v'y' \cong u'x'$(mod Π), and the lemma holds trivially by transitivity of $\cong$. If $\chi = a$, then $\nu = x_b d_i(\begin{smallmatrix}\alpha \\ u_b\end{smallmatrix} \bullet \begin{smallmatrix}a\alpha' \\ au'_b\end{smallmatrix}) r_i x'_b$ and any ν' in q is such that $v'y' \cong u'x'$ (mod Π). Since Π is a left congruence, $w\$ \cong u'x'$ (mod Π) implies that $aw\$ \cong au'x'$ (mod Π) and the lemma holds. If $\chi \in T_r$, then $\nu = x_b d_i(\begin{smallmatrix}\alpha \\ u_b\end{smallmatrix} \bullet) r_i x'_b$ and any ν' in q is such that $v'y' \cong x'$ (mod Π), and the lemma holds.□ □

Lemma 6 *Let $\nu = x_b d_i(\begin{smallmatrix}\alpha\\ u_b\end{smallmatrix}\bullet\begin{smallmatrix}\alpha'\\ u'_b\end{smallmatrix})r_i x'_b$ be a position in $\mathcal{N}$ and γ a string in $(V \cup T_d)^*$. If $q_s\gamma \vDash^* [\nu]_{\mathsf{item}_0}$ in Γ/item_0, then $A\xrightarrow{i}\alpha\bullet\alpha'$ is a valid LR(0) item for γ, i.e. $S\underset{\text{rm}}{\Longrightarrow}{}^*\delta Az\underset{\text{rm}}{\overset{i}{\Longrightarrow}}\delta\alpha\alpha' z = \gamma\alpha' z$ holds in $\mathcal{G}$.*

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

If $\chi = d_j$, then $\alpha' = B\alpha''$ and ν' is of form $y_b d_j(\begin{smallmatrix}\beta\\ v_b\end{smallmatrix}\bullet)r_j y'_b$ for some $B\xrightarrow{j}\beta$ in P. Then, $\gamma\alpha' z = \gamma B\alpha'' z\underset{\text{rm}}{\overset{j}{\Longrightarrow}}\gamma\beta\alpha'' z$ holds in $\mathcal{G}$ and $B\xrightarrow{j}\bullet\beta$ is a valid LR(0) item for $\gamma\chi$. If χ is in V, then $\alpha' = \chi\alpha''$ and ν' is of form $x_b d_i(\begin{smallmatrix}\alpha\chi\\ u_b v_b\end{smallmatrix}\bullet\begin{smallmatrix}\alpha''\\ u''_b\end{smallmatrix})r_i x'_b$. Then, $\gamma\alpha' z = \gamma\chi\alpha'' z$ and $A\xrightarrow{i}\alpha\chi\bullet\alpha''$ is a valid LR(0) item for $\gamma\chi$.□ □

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

Proof. Let us suppose that $\mathcal{G}$ is noncanonically item_Π-ambiguous. We have the relation

$$(q_s, q_s)\ (\mathsf{mas} \cup \mathsf{mae})^*\ (q_1, q_2)\ \mathsf{mac}\ (q_3, q_2)\ \mathsf{ma}^*\ (q_f, q_f). \tag{18}$$

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 $y_b d_j(\begin{smallmatrix}\beta\\ v_b\end{smallmatrix}\bullet\begin{smallmatrix}\beta'\\ v'_b\end{smallmatrix})r_j y'_b$. Relation (q_1, q_2) 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 $x_b d_i(\begin{smallmatrix}\alpha\\ u_b\end{smallmatrix}\bullet)r_i x'_b$. The relation also forbids r_i to be equal to $1 : v' r_j$.

Clearly, the first part (q_s, q_s) $(\mathsf{mas} \cup \mathsf{mae})^*$ (q_1, q_2) of Equation (18) implies that $q_s\delta\alpha \vDash^* q_1$ and $q_s\gamma\beta \vDash^* q_2$ with $\delta\alpha = \gamma\beta$ for some δ and γ in $(V \cup T_d)^*$. By Lemma 6,

$$S\underset{\text{rm}}{\Longrightarrow}{}^*\delta Ax'\underset{\text{rm}}{\overset{i}{\Longrightarrow}}\delta\alpha x' \text{ and } S\underset{\text{rm}}{\Longrightarrow}{}^*\gamma By'\underset{\text{rm}}{\overset{j}{\Longrightarrow}}\gamma\beta\beta' y' = \delta\alpha\beta' y' \tag{19}$$

hold in $\mathcal{G}$.

Let us now consider the second part (q_1, q_2) mac (q_3, q_2) ma^* (q_f, q_f) of Equation (18). By Lemma 3, there exist two bracketed strings w_b and w'_b with $w = w'$ such that $q_1 w_b \vDash^* q_f$ and $q_2 w'_b \vDash^* q_f$. By Lemma 5, $x' \cong w \pmod{\Pi}$ and $v'y' \cong w' \pmod{\Pi}$ and by transitivity

$$v'y' \cong x' \pmod{\Pi}. \tag{20}$$

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 (19) and (20) fit the requirements of Equation (16). Nevertheless, $v' = \varepsilon$ and $r_i \neq 1 : v'_b r_j$ thus implies $i \neq j$, violating the requirements of Equation (17). If $\beta' = v'$ is in T^+, then once again we fit the requirements of Equation (16). Nevertheless, in this case, $y' \neq v'y'$, hence violating the requirements of Equation (17). If there is at least one nonterminal C in β', then

$$S'\underset{\text{rm}}{\Longrightarrow}{}^*\gamma\beta v_1 C v_3 y'\underset{\text{rm}}{\Longrightarrow}{}^*\gamma\beta v_1 v_2 v_3 y' = \delta\alpha v_1 v_2 v_3 y' = \delta\alpha v'y'. \tag{21}$$

Remember that $r_i \neq 1 : v'_b r_j$, thus $v_1 v_2 \neq \varepsilon$ and Equation (17) cannot hold.□□

We have implemented our algorithm [37] along with a LR and a LRR test in GNU bison [11]. Table 1 shows the number of initial LR(0) conflicts left

Table 1: Number of initial LR(0) conflicts remaining with each test.

Grammar	LR(0)	LRR	NU	SLR(1)	LRR	NU	LR(1)
ANSI C'	387	43	32	43	43	32	29
Standard ML	477	299	271	165	163	135	127
Elsa–	1379	278	226	66	63	58	64
Elsa	2094	-	308	91	88	77	-

unsolved with our tests using successively LR(0), SLR(1) and LR(1) items. More precisely, the first three columns present the number of LR(0) conflicts taken as pairs of LR(0) items in conflict, the number of such LR(0) conflicts that remain after a LRR exploration of the right context of each conflict, and the number of LR(0) conflicts that remain after a noncanonical exploration of the right context with our test. The three next columns do the same but employ a SLR(1) notion of conflicts, and the last column uses LR(1) items. The missing entries indicate memory exhaustions. The tested grammars are the ANSI C grammar [23] with its `typedef` ambiguity, the Standard ML grammar [27], and a simplified and the complete version of the C++ grammar of Elsa [26].

At the same level of precision, our algorithm performs better than the others, and is also an order of magnitude faster using SLR(1) items than the LR(1) test.

A.5.4 Horizontal and Vertical Ambiguity

We first recall the definition of horizontal and vertical ambiguity.

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

It is horizontally unambiguous *if and only if, for all productions $A{\to}\alpha$ in P, and for all decompositions $\alpha = \alpha_1\alpha_2$, $\mathcal{L}(\alpha_1) \between \mathcal{L}(\alpha_2) = \emptyset$, where $\between$ is the language* overlap *operator defined by $L_1 \between L_2 = \{xyz \mid x, xy \in L_1, y \in T^+,$ and $yz, z \in L_2\}$.*

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.*

Proof. If $\Gamma/{\equiv}$ 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 $xwx' = xw'x'$, and thus $\mathcal{G}$ is regular $\equiv$-ambiguous. If $\Gamma/{\equiv}$ is horizontally ambiguous, then $x_b v_b y_b w_b r_i x'_b$ and $x_b v'_b y_b w'_b r_i x'_b$ are two different sentences in $\mathcal{L}(\Gamma/{\equiv}) \cap T_b^*$ with $xvywx' = xv'yw'x'$, and thus $\mathcal{G}$ is regular $\equiv$-ambiguous. □

The intended application of the test of Brabrand *et al.* is the testing of grammars that describe the secondary structure of the RNA [34]. For completeness, we copy here the results of our tool [37] on these in Table 2.

A.5.5 LL-Regular Testing

In spite of their popularity as an alternative to the bottom-up parsers of the LR(k) family, top-down parser construction tests [35, 40] would not be very relevant for practical ambiguity detection: the class of LL(k) grammars is arguably

Table 2: Reported ambiguities in the RNA grammars from [34].

Grammar [34]	actual class	bison	HVRU [5]	NU(item_1)
$\mathcal{G}_1$	ambiguous	18+12	1+5	14
$\mathcal{G}_2$	ambiguous	19+14	1+6	13
$\mathcal{G}_3$	non-LR	4+0	0+0	2
$\mathcal{G}_4$	SLR(1)	0+0	0+0	0
$\mathcal{G}_5$	SLR(1)	0+0	0+0	0
$\mathcal{G}_6$	LALR(1)	0+0	0+0	0
$\mathcal{G}_7$	non-LR	5+0	0+0	3
$\mathcal{G}_8$	LALR(1)	0+0	0+0	0

not large enough. The exception is the class of LL-Regular grammars [22, 30, 32], defined similarly to LR-Regular grammars by generalizing the LL(k) condition. In particular, the Strong LL-Regular condition [32] has given birth to the LL(*) algorithm of the upcoming version 3 of ANTLR [31] and to the ambiguity detection tool shipped with its IDE, ANTLRWorks. These classes of grammars are strictly contained inside the class of LR-Regular grammars for left congruences [17], and are thus captured by our noncanonical ambiguity test when using item_Π.

In this section, we explicit the (non) relation between the LLR condition and the regular ambiguity test. A grammar $\mathcal{G}$ is LL(Π) if and only if

$$S \underset{\text{lm}}{\Longrightarrow}{}^* zA\delta \underset{\text{lm}}{\Longrightarrow} z\alpha\delta \underset{\text{lm}}{\Longrightarrow}{}^* zx, S \underset{\text{lm}}{\Longrightarrow}{}^* zA\delta \underset{\text{lm}}{\Longrightarrow} z\beta\delta \underset{\text{lm}}{\Longrightarrow}{}^* zy \text{ and } x \cong y \,(\mathrm{mod}\,\Pi) \tag{22}$$

imply

$$\alpha = \beta. \tag{23}$$

Let us consider the grammar with rules

$$S{\to}BAb,\ A{\to}a\,|\,aa,\ B{\to}bBa\,|\,b. \tag{$\mathcal{G}_8$}$$

It is SLL(2) but not RU(item_2), as witnessed by the two different sentences $d_2\,d_5\,b\,d_6\,b\,r_6\,a\,r_5\,d_3\,a\,r_3\,b\,r_2$ and $d_2\,d_5\,b\,d_6\,b\,r_6\,d_4\,a\,a\,r_4\,b\,r_2$ recognized by Γ/item_2. On the other hand, the grammar with rules

$$S{\to}aAb\,|\,aAa,\ A{\to}ab\,|\,a \tag{$\mathcal{G}_9$}$$

is not LL(2), but is RU(item_2).

A.6 More Related Work

A.6.1 Systems of Pushdown Acceptors and their Transformations

Systems of pushdown acceptors were defined by Kuich [25] as a representation of CFGs amenable to testing for *quasi determinism*. If a system is not immediately quasi deterministic, one of its language and ambiguity preserving transformations can be tested instead. The quasi determinism of a system or of one of its transformations implies the unambiguity of the original grammar.

The approach of Kuich is very close to our own: the systems of pushdown acceptors are another interpretation of Γ/item_0, and the transformations provide the same flexibility as the choice of an equivalence relation finer than item_0 for us. Nonetheless, the quasi determinism condition is very restrictive, and Kuich had to rely on extensive transformations in practice, including determinization

and identification of which nonterminals displayed self embedding or not. A formal comparison of the criteria is likely to be rather meaningless if we do not complete quasi determinism with some of the transformations to yield a more powerful criterion.

It is worth noting at this point that our own distinction between an equivalence relation and a criterion on the resulting NFA is somewhat blurry as well, and really dictated by the usual representations found in the parsing literature.

A.6.2 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) [43], 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 [33].

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].

A.6.3 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 [15], item grammars [16], transition diagrams or networks [9, 44], nondeterministic LR automata [19], 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 [29].

A.6.4 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 [18]. Nevertheless, ambiguity checking needs completed trace semantics, and it is unclear whether any suitable solution exists in the verification literature.