DISCRIMINANT REVERSE LR PARSING OF CONTEXT-FREE GRAMMARS

Jacques Farré
Laboratoire I3S – CNRS and Université de Nice - Sophia Antipolis
jf@essi.fr

1 Introduction

In *Discriminant Reverse LR(k)* parsing[1], actions are determined by scanning a stack *suffix*, thanks to a small deterministic finite automaton (dfa). Thus, opposite to (direct) LR parsers, DRLR parsers do not need the whole content of the stack in order to determine actions. DRLR parsers can, in few cases, deeply explore the stack, but it has been proven they are linear in time.

When the grammar is not of the required class, this feature of DRLR parsers allows to defer actions in conflict and to push a *conflict mark* onto the stack. The DRLR parser can keep on parsing the input right of the conflict, that is to say it can perform shifts *and reductions* as long as these actions are determined by a stack suffix above the mark. Otherwise, the action cannot be chosen and an *induced* conflict occurs; a new mark is pushed, and parsing of the remaining input can resume.

For most grammars, the stack configurations in which marks resolve the LR conflict, and how to resolve it, are computed at construction. For other grammars, the conflict is resolved at parsing time by comparing the stack content with the candidate derivations (which are given by the marks), but the stack configurations in which marks trigger these comparisons are computed at construction.

One can argue that conflict marks are an implementation of multiple stacks. But the stack is not explored below a conflict mark, and Extended DRLR needs a single stack and a single parser. Thus, it is not comparable to GLR. In particular, Extended DRLR parsers for which conflicts resolution can be computed at construction are linear in time and size.

2 Principles of (Extended) DRLR construction

This presentation will rely on a DRLR(0) parser. In a DRLR(0) item $[i, A \to \alpha{\bullet}\beta]$, i is a parsing action (shift if i =3D 0, reduce according tőth $i$production if $i > 0$); the core $A \to \alpha{\bullet}\beta$ means that a stack suffix σ has been scanned, and $\beta \overset{*}{\underset{rm}{\Rightarrow}} \sigma x$. The states of the DRLR automaton are the states of the *dfa* which recognizes these σ's by reading them from right to left. If $[i, A \to \alpha X{\bullet}\beta]$ belongs to a state q and if no $[i', A' \to \alpha' X{\bullet}\beta']$, i' ≠3D i, belongs to q, the suffix $X\sigma$ determines action i; else more stack must be read, and a transition by X to q', which will contain $[i, A \to \alpha{\bullet}X\beta]$, is done; $[i, A \to {\bullet}\alpha]$ infers (closure computation) $[i, B \to \alpha'{\bullet}A\beta']$, that is a step back in a rightmost derivation chain.

A LR(0) conflict occurs when distinct rigthmost derivations produce a same prefix ϕ. These derivations (or *proto*-derivations when ϕ belongs to a language $\alpha\beta^*\gamma$) can be computed from the states of the DRLR(0) automaton. Each step of these derivations defines *mark positions.* For example, let

$$S \overset{*}{\underset{rm}{\Rightarrow}} \pi Ax \underset{rm}{\Rightarrow} \pi\alpha B\beta x \overset{*}{\underset{rm}{\Rightarrow}} \pi\alpha Byx \underset{rm}{\Rightarrow} \pi\alpha\gamma C\delta yx \overset{*}{\underset{rm}{\Rightarrow}} \pi\alpha\gamma Czyx \underset{rm}{\Rightarrow} \phi zyx$$

$$S \overset{*}{\underset{rm}{\Rightarrow}} \pi'A'x' \underset{rm}{\Rightarrow} \pi'\alpha'B'\beta'x' \overset{*}{\underset{rm}{\Rightarrow}} \pi'\alpha'B'y'x' \underset{rm}{\Rightarrow} \pi'\alpha'\gamma'C'\delta'y'x' \overset{*}{\underset{rm}{\Rightarrow}} \pi'\alpha'\gamma'C'z'y'x' \underset{rm}{\Rightarrow} \phi z'y'x'$$

be two derivations in conflict: a mark is pushed onto stack after ϕ, and it will be found instead of C or C', but only the C and C' of $B \overset{i}{\rightarrow} \gamma C\delta$ and $B' \overset{i'}{\rightarrow} \gamma'C'\delta'$. The cores $B \rightarrow \gamma C{\bullet}\delta$ and $B' \rightarrow \gamma'C'{\bullet}\delta'$ will be the mark positions.

If the languages of δ and δ' are disjoint and not prefix, z and z' can be reduced respectively to δ and δ' without scanning the stack until the mark. As, in this case, $\delta \neq$3D, $\delta[i, B \rightarrow \gamma C{\bullet}\delta]$ and $[i', B' \rightarrow \gamma'C'{\bullet}\delta']$ cannot be in the same state, and the mark can determine action i or action i'. Choosing i or i' means the LR(0) conflict must be resolved by reducing respectively to C or to C'.

But, if for example δ =3D, $\delta[i, B \rightarrow \gamma C{\bullet}\delta]$ and $[i', B' \rightarrow \gamma'C'{\bullet}\delta']$ will be in a same state. The mark cannot decide between i and i'. The new mark for this induced conflict can be found in stack instead of B or B', but only in positions $A \rightarrow \alpha B{\bullet}\beta$ for reduction to C and $A' \rightarrow \alpha'B'{\bullet}\beta'$ for reduction to C'. The LR conflict can be resolved if the languages of β and β' are discriminant.

Conflicts cannot be resolved at construction if a mark has the positions $A \rightarrow \alpha B{\bullet}\beta$ and $A \rightarrow \alpha'B'{\bullet}\beta$, which mean that $\pi\alpha B \overset{*}{\Rightarrow} w$, $\pi'\alpha'B' \overset{*}{\Rightarrow} w$. This does not mean that the grammar is ambiguous, as shown by the example below.

3 A working example

The example will rely on the grammar $S \overset{2}{\rightarrow} aAb$, $S \overset{3}{\rightarrow} aBbb$, $A \overset{4}{\rightarrow} aAb$, $A \overset{5}{\rightarrow} c$, $B \overset{6}{\rightarrow} aBbb$, $B \overset{7}{\rightarrow} c$, which produces the non deterministic language $a^n cb^n \cup a^n cb^{2n}, n > 0$. DRLR(0) augments the initial grammar with $S' \overset{1}{\rightarrow} \vdash S \dashv$.

Reductions 5 and 7 are in conflict. A mark m_0 with positions $S \rightarrow aA{\bullet}b$ and $A \rightarrow aA{\bullet}b$ for 5, with positions $S \rightarrow aB{\bullet}bb$ and $B \rightarrow aB{\bullet}bb$ for 7, is pushed onto the stack. It decides a shift for any of its position in q_0, but it produces an induced conflict between $[2, S \rightarrow aA{\bullet}b]$, $[4, A \rightarrow aA{\bullet}b]$, $[0, S \rightarrow aB{\bullet}bb]$ and $[0, B \rightarrow aB{\bullet}bb]$ in some q_i for the stack suffix $m_0 b$.

The new mark m_1 has positions $S' \rightarrow \vdash S{\bullet}\dashv$ for action 2, $S \rightarrow aA{\bullet}b$ and $A \rightarrow aA{\bullet}b$ for 4, $S \rightarrow aBb{\bullet}b$ and $B \rightarrow aBb{\bullet}b$ for 0. It resolves the LR conflict in favor of 5 with the stack suffix $m_1 \dashv$, and it produces another induced conflict between $[2, S \rightarrow aA{\bullet}b]$, $[4, A \rightarrow aA{\bullet}b]$, $[3, S \rightarrow aBb{\bullet}b]$ and $[6, B \rightarrow aBb{\bullet}b]$.

The mark m_2 has positions $S' \rightarrow \vdash S{\bullet}\dashv$ for actions 2 and 3, $S \rightarrow aA{\bullet}b$ and $A \rightarrow aA{\bullet}b$ for 4, $S \rightarrow aB{\bullet}bb$ and $B \rightarrow aB{\bullet}bb$ for 6. When the stack suffix is $m_2 \dashv$, the conflict cannot be resolved at construction. Otherwise (suffix =3D$_2 b$), m_2 has the same positions as m_0, and pushes m_1.

The mark m_2 means that an even number of b has been read, while m_1 means that an odd number of b has been read. The suffix $m_2 \dashv$ decides a conflict resolution at parsing time, while $m_1 \dashv$ can decide at construction to resolve the conflict in favor of 5. But if we have $S \overset{2}{\rightarrow} aAba$ instead of $S \overset{2}{\rightarrow} aAb$, the conflict resolution can be decided at construction in any case : the position $S' \rightarrow \vdash S{\bullet}\dashv$ of m_2 is no more compatible with the reduction in conflict 5, and it can decide the reduction 7.

References

[1] Fortes-Gálvez, J. 1996. A practical small LR parser with action decision through minimal stack suffix scanning. in *Developments in Language Theory II*, World Scientific.