LR parsers with lookahead

In order to enlarge the class of grammars that can be parsed we need to consider more powerful parsing strategies. In particular we will study:

- LR(1) parsers
- LALR parsers

LR(1) items

LR(1) items structure

The very general idea is to encapsulate more information in the items of an automaton to decide when to reduce. The solution is to differentiate items on the base of lookaheads. As a result a general item follows now the template $[A \to \alpha \cdot \beta, a]$

LR(1) items and reductions

Given the new form on an item, the parser will call for a reduction $A \to \alpha$ only for item sets including the item $[A \to \alpha \cdot, a]$ and only for symbol a

LR(1) CLOSURE and GOTO functions

Closure of an item

If $[A \to \alpha \cdot B\beta, a]$ is in I then for each production $B \to \gamma$ and for each terminal b in FIRST(βa) add the item $[B \to \cdot\gamma, b]$

GOTO(I, X)

Let J initially empty. For each item $[A \to \alpha \cdot X\beta, a]$ in I add item $[A \to \alpha X \cdot \beta, a]$ to set J. Then compute CLOSURE(J)

Consider the starting item as the closure of the item $[S' \to S, \$]$.

Exercise

Compute the LR(1) item sets for the following grammar:
$S \to CC \quad C \to cC|d$

LR(1) parsing table

How to build the LR(1) parsing table

1. build the collection of sets of LR(1) items for the grammar
2. Parsing actions for state i are:
   1. if $[A \to \alpha \cdot a\beta, b]$ is in I_i and GOTO(I_i, a)$= I_j$ then set ACTION$[i, a]$ to shift J.
   2. if $[A \to \alpha\cdot, a]$ is in I_i $A \neq S'$ then set ACTION$[i, a]$ to reduce($A \to \alpha$)
   3. if $[S' \to S\cdot, \$]$ is in I_i then set ACTION$[i, \$]$ to accept
3. if $\texttt{GOTO}(I_i, A) = I_j$ then $\texttt{GOTO}[i, A] = j$
4. All entries not defined so far are mare "error"
5. The initial state of the parse is the one constructed from the set of items containing $[S' \to \cdot S, \$]$

Consider the following grammar and derive the LR(1) parsing table:
$S \to L = R | R \quad L \to *R | id \quad R \to L$

LR(1) parsing

Consider the following grammar and discuss applicability of LR(1) parsing:
$S \rightarrow aSa \mid a$

- Which is the language generated?
- Propose an alternative grammar parsable using an LR(1) parser

LR(1) parsing

Consider the following grammar and discuss applicability of LR(1) parsing:
$S \rightarrow aSa \mid a$

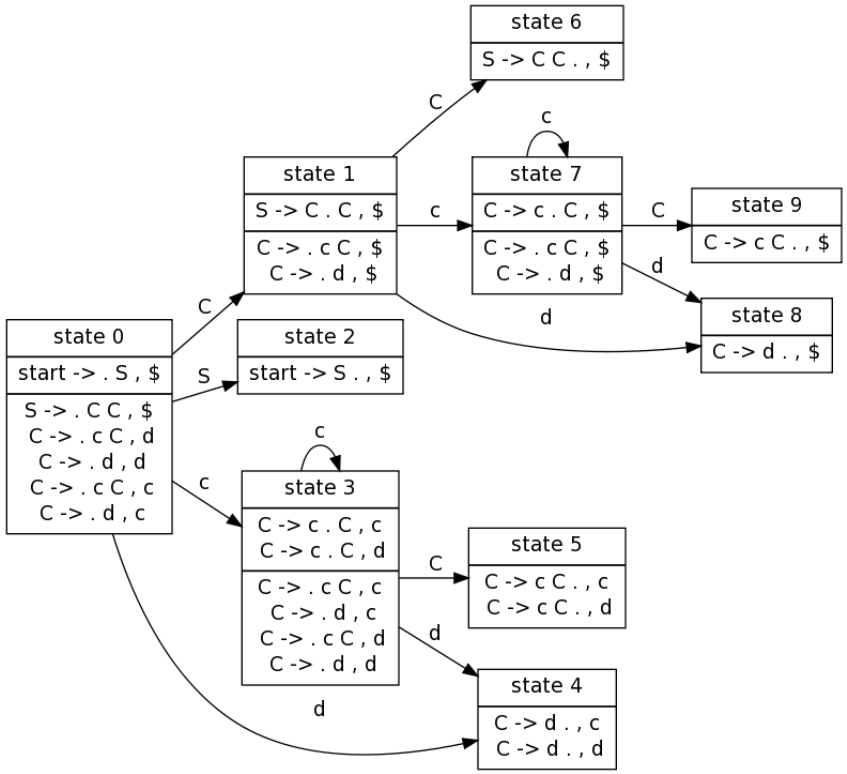
- Which is the language generated?
- Propose an alternative grammar parsable using an LR(1) parser

LALR parsing

- LR(1) for a real language a SLR parser has several hundred states. For the same language an LR(1) parser has several thousand states
- Can we produce a parser with power similar to LR(1) and table dimension similar to SLR?

LALR parsing

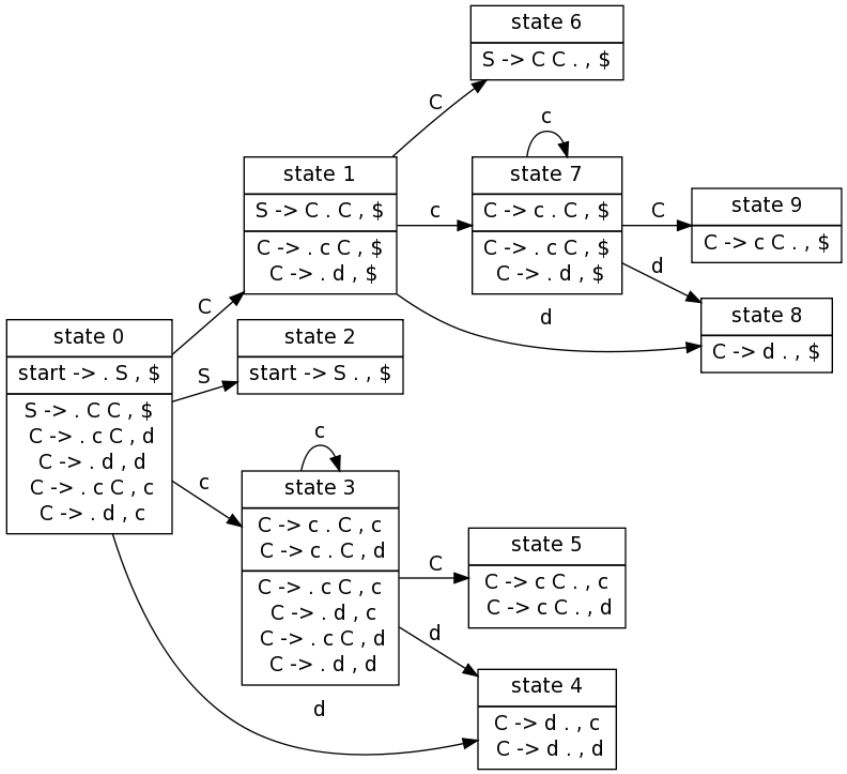
Let's consider the LR(1) automaton for the grammar
$S \to CC \quad C \to cC|d$



LALR table can be built from LR(1) automaton merging "similar" item sets.

LALR parsing

Let's consider the LR(1) automaton for the grammar
$S \to CC \quad C \to cC|d$



LALR table can be built from LR(1) automaton merging "similar" item sets.

Exercises

Consider the grammar:
$S \to Aa|bAc|dc|bda \quad A \to d$
show that is LALR(1) but not SLR(1)

Consider the grammar:
$S \to Aa|bAc|Bc|bBa \quad A \to d \quad B \to d$
show that is LR(1) but not LALR(1)

Exercises

Consider the grammar:
$S \to Aa|bAc|dc|bda \quad A \to d$
show that is LALR(1) but not SLR(1)

Consider the grammar:
$S \to Aa|bAc|Bc|bBa \quad A \to d \quad B \to d$
show that is LR(1) but not LALR(1)