A Unifying Model for Lookahead LR Parsing

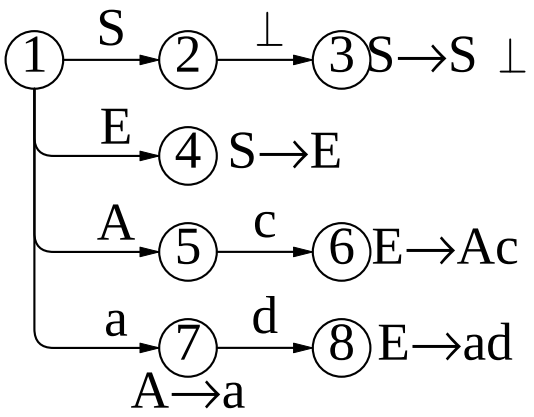
Manuel E. Bermudez

University of Florida
U.S.A.

LR Parser Construction

1.- Build LR(0) Automaton:

$S \rightarrow S \perp$ $\quad E \rightarrow Ac$ $\quad A \rightarrow a$
$S \rightarrow E$ $\quad E \rightarrow ad$

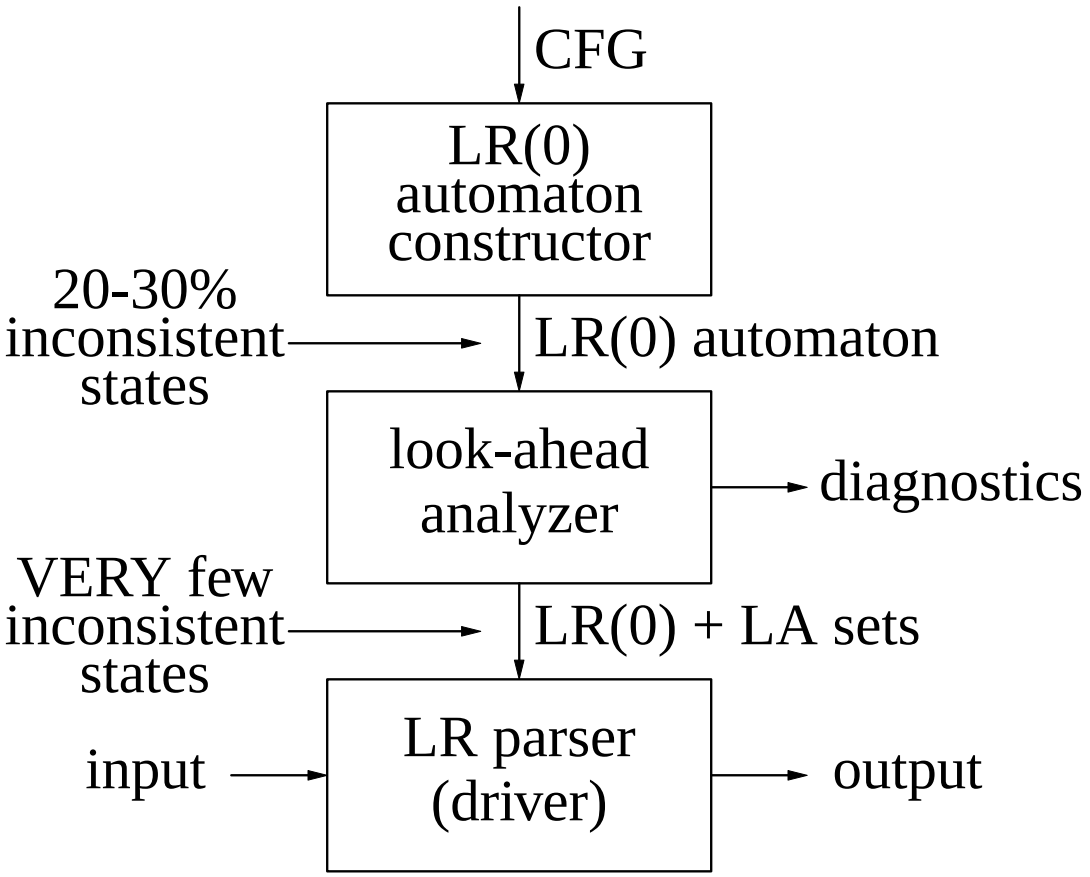


2.- Add lookahead sets:

LA (7,A→a) = {c}

LA (7,d) = {d}

Disjoint. Grammar is LALR(1).



Problems:

- Good diagnostics are difficult to produce.
- Required changes contort semantic processing.
- Remaining conflicts are few and difficult to correct.
- Many different lookahead algorithms.
- Systems are completely incompatible.
- No unifying algorithms.

LAR(M,C,L) Parsers

Central Premise: Lookahead information can be obtained by ‘‘simulating-ahead’’ the LR(0) parser’s moves, with three degrees of freedom:

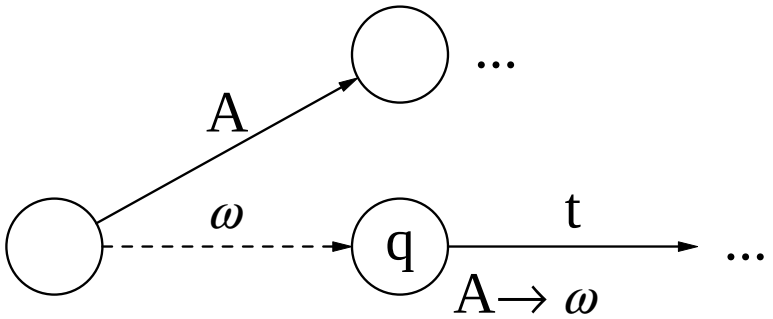
M: How much of the stack is to be simulated.

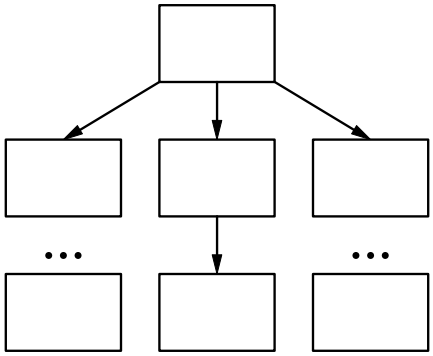
C: Whether the context implied by the inconsistent state is to be taken into consideration during the simulation.

L: The amount of lookahead, i.e. the duration of the simulation.

Strategy

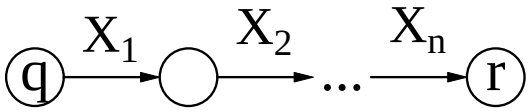
Build one "Lookahead" FSA for each inconsistent LR(0) state.





Notation:

- $[q{:}X_1 \ldots X_n]$: a sequence of states:

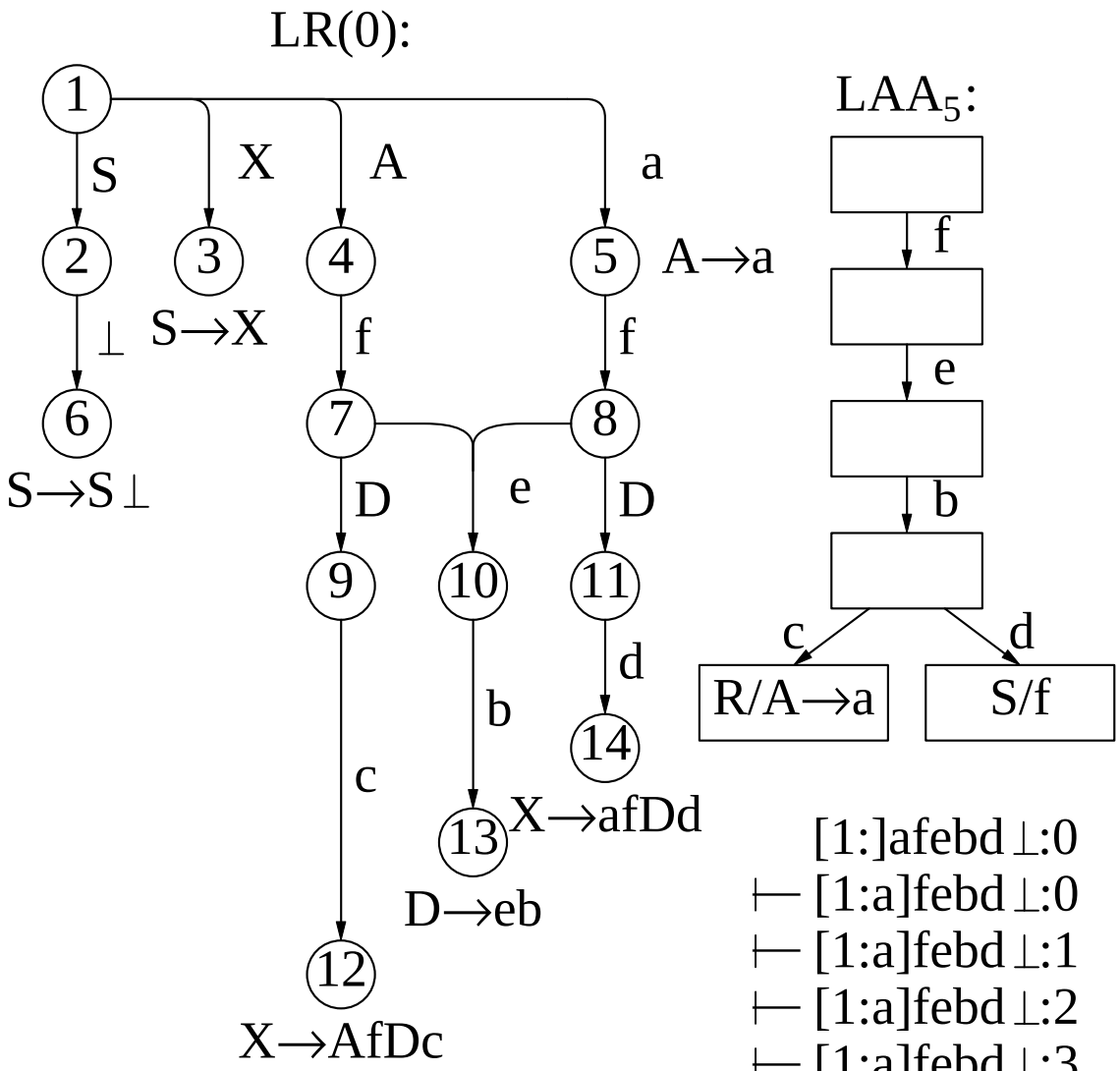


- $\mathrm{Top}[q{:}X_1 \ldots X_n]$ denotes r.
- $[X_1 \ldots X_n]$ denotes $[\mathrm{Start}{:}X_1 \ldots X_n]$
- $[\alpha]x{:}\,n$ is a "[stack]input:lookahead" **configuration**.

LAR(M,C,L) parser moves:

- **Shift**: $[\alpha]tz{:}\,0 \vdash [\alpha t]z{:}\,0$
- **Reduce**: $[\alpha\omega]z{:}\,0 \vdash [\alpha A]z{:}\,0$
- **LA-Scan**: $[\alpha]wtz{:}\,n \vdash [\alpha]wtz{:}\,n+1$
- **LA-Shift**: $[\alpha]tz{:}\,n \vdash [\alpha t]z{:}\,0$
- **LA-Reduce**: $[\alpha\omega]z{:}\,n \vdash [\alpha A]z{:}\,0$

Example



[1:]afebd⊥:0
⊢ [1:a]febd⊥:0
⊢ [1:a]febd⊥:1
⊢ [1:a]febd⊥:2
⊢ [1:a]febd⊥:3
⊢ [1:a]febd⊥:4
⊢ [1:af]ebd⊥:0
⊢ [1:afe]bd⊥:0
⊢ [1:afeb]d⊥:0
⊢ [1:afD]d⊥:0
⊢ [1:afDd]⊥:0
⊢ [1:X]⊥:0
⊢ [1:S]⊥:0

Construction of Lookahead Automata

- "Simulate-ahead" LR(0) parser's actions, starting at the conflicting state.

- Simulation keeps track of top (at most M) states of LR(0) parser's stack (as best it can).

- Begin by "forcing" each conflicting action.

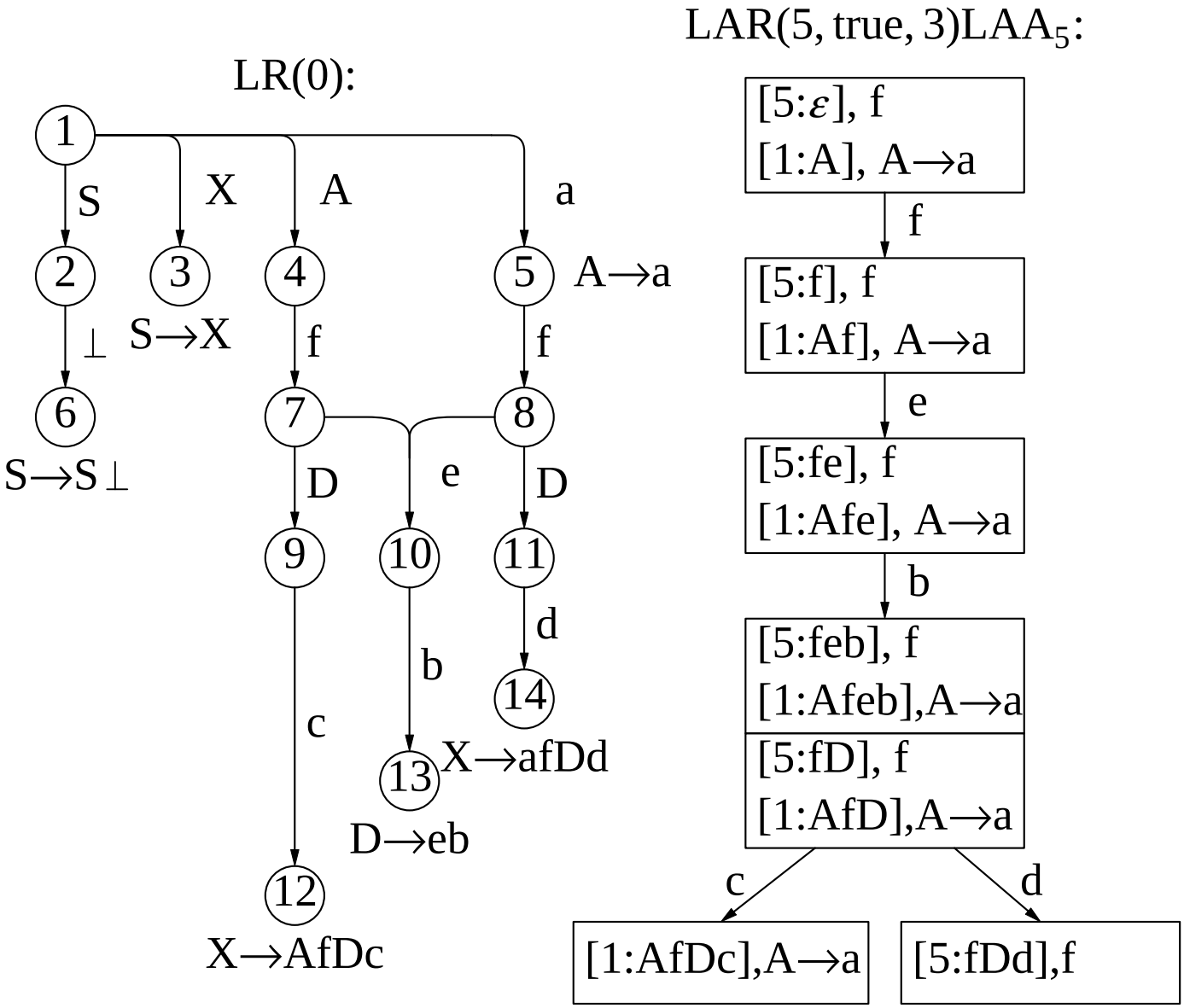
- Simulate all applicable moves:

  -- Reduction chain "collected" in LA state.

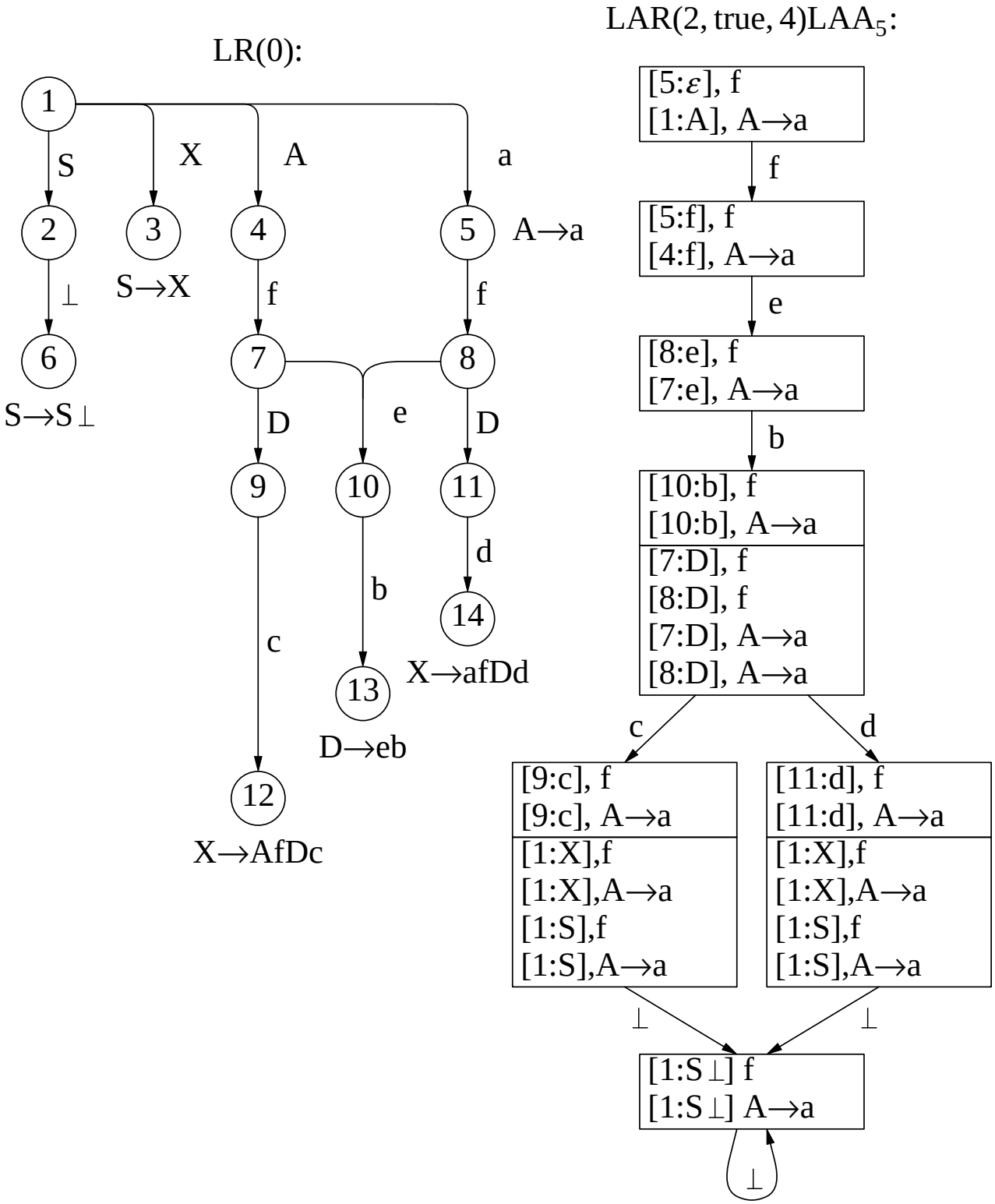
  -- Shift-moves are transitions in LA Automaton.

- Stop if L is exceeded, or if conflict is resolved.

Case I. L too small: LAA not reduced



Case II. M too small: LAA not acyclic


LR(0):
1
S
X
A
a
2
3
4
5
A→a
⊥
S→X
f
f
6
7
8
S→S⊥
D
e
D
9
10
11
d
b
14
c
13
X→afDd
D→eb
12
X→AfDc
LAR(2, true, 4)LAA5:
[5:ε], f
[1:A], A→a
f
[5:f], f
[4:f], A→a
e
[8:e], f
[7:e], A→a
b
[10:b], f
[10:b], A→a
[7:D], f
[8:D], f
[7:D], A→a
[8:D], A→a
c
d
[9:c], f
[9:c], A→a
[1:X],f
[1:X],A→a
[1:S],f
[1:S],A→a
[11:d], f
[11:d], A→a
[1:X],f
[1:X],A→a
[1:S],f
[1:S],A→a
⊥
⊥
[1:S⊥] f
[1:S⊥] A→a
⊥

Definition:

A grammar is LAR(M,C,L) if all lookahead automata are **acyclic** and **reduced**, i.e. if final states are always reachable.

Theorem:

If G is LAR(M,C,L), then

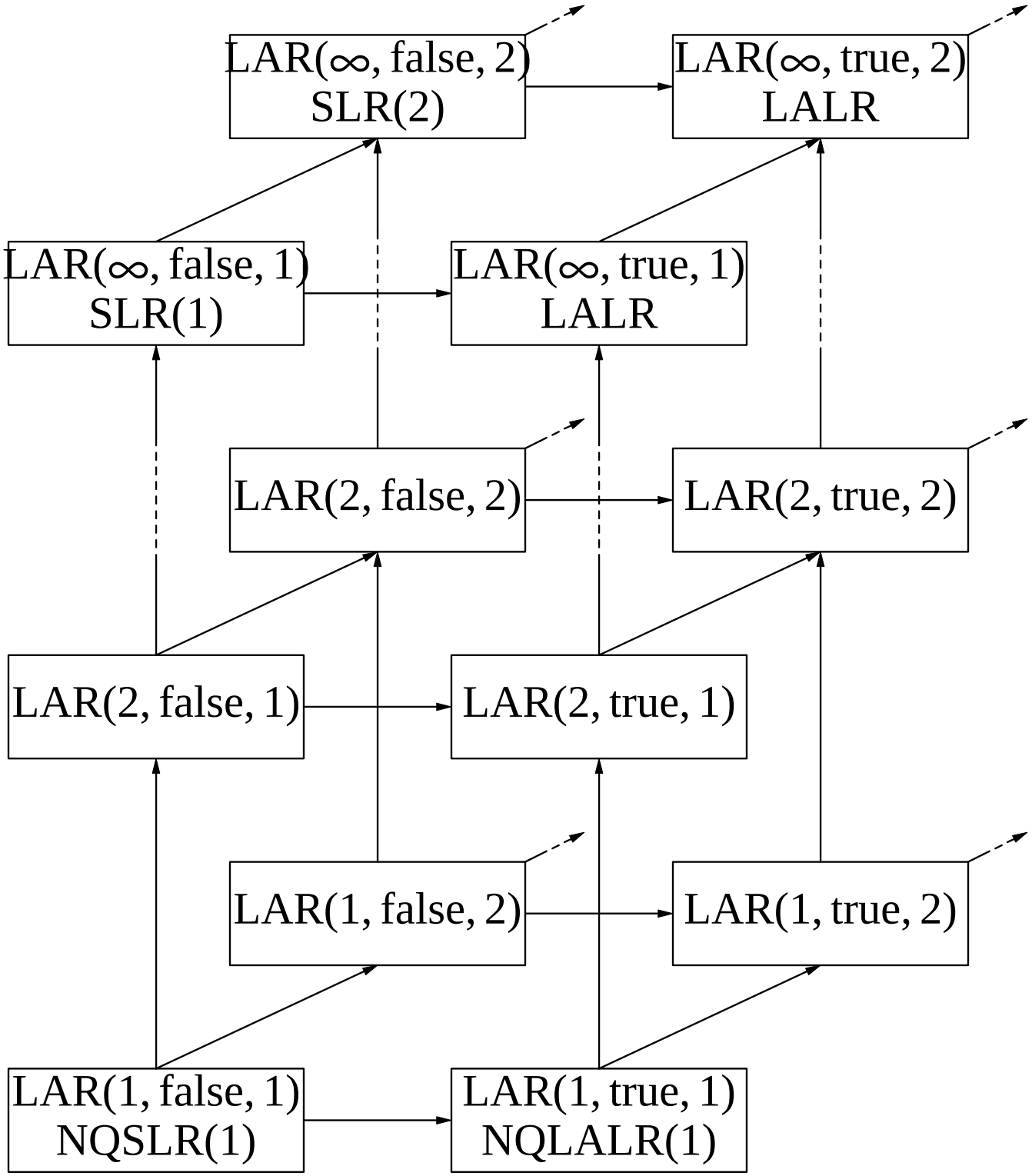
$$L(\ LR(0)\ ) = L(\ LAR(M,C,L)\ ),$$

i.e. the parser is correct.

Theorem:

- LAR(∞,true,k) $\equiv$ LALR(k).
- LAR(M,true,k) $\equiv$ NQLALR(k).
- LAR(∞,false,k) $\equiv$ SLR(k)
- LAR(M,false,k) $\equiv$ NQSLR(k).

Grammar Class Relationships


LAR(∞, false, 2)
SLR(2)
LAR(∞, true, 2)
LALR
LAR(∞, false, 1)
SLR(1)
LAR(∞, true, 1)
LALR
LAR(2, false, 2)
LAR(2, true, 2)
LAR(2, false, 1)
LAR(2, true, 1)
LAR(1, false, 2)
LAR(1, true, 2)
LAR(1, false, 1)
NQSLR(1)
LAR(1, true, 1)
NQLALR(1)

Conclusions

- LAR(M,C,L) allows single and multiple symbol lookahead.
- Can obtain LALR(k), SLR(k), NQLALR(k), etc. via appropriate settings of M,C, and L.
- Can manipulate the parameters to suit the grammar, rather than the other way around.
- Can progressively attempt larger amounts of lookahead.
- Better understood relationships among grammar classes.
- Simultaneous availability of several techniques.