Parallel Generation of LR Parsers

by

Manuel E. Bermudez

George Logothetis

Richard Newman-Wolfe

University of Florida

Gainesville, FL 32611

Generation of LR Parsers.

1) Generate LR(0) Parser.
2) Add Lookahead Information.
3) Encode into Parse Table.

ACTION Table:

	t_1	t_2	. . .	t_n
p_1	S/q			
. . .				
p_m			R/A$\rightarrow \omega$	

Four types of entries:

- ‘‘S/q’’**:** Shift to state q.
- ‘‘R/A$\rightarrow \omega$’’**:** Reduce using A$\rightarrow \omega$.
- Accept the input.
- Error.

	t_1	t_2	. . .	t_n
p_1	S/q			
. . .				
p_m			R/A$\rightarrow \omega$	

Criteria for filling table:

- ACTION(p,t) = "S/q" if $p \xrightarrow{t} q$.
- ACTION(p,t) = "R/A$\rightarrow \omega$" if t$\in$Follow(A), in the SLR(1) technique.
- Follow(A) = {t | S =>* αAtx}.
- LALR(1) similar; different Follow sets.

Critical Observations:

- Follow sets are not independent.
- Rows in ACTION table are not independent.
- Can fill table **by column**, not by row.
- Each column is **completely** independent.
- Can assign one processor to each column.
- Speedup factor: min (processors,terminals).

Decomposition of Follow sets:

Follow(A) = IFollow(A) $\cup$ DFollow(A),
IFollow(A) = $\cup$ {Follow(B) | B$\rightarrow \alpha$Aγ, γ =>* ε},
DFollow(A)= $\cup$ {First(X) | B$\rightarrow \alpha$AγXδ, γ =>* ε},
First(A) = $\cup$ {First(X) | A$\rightarrow \gamma$Xδ, γ =>* ε},
First(t) = {t}.

In relational form,

- X ff A if A$\rightarrow\gamma$Xδ and γ =>* ε.
- X fF A if B$\rightarrow \alpha$AγXδ and γ =>* ε.
- B FF A if B$\rightarrow \alpha$Aγ and γ =>* ε.

Theorem: t$\in$Follow(A) iff t (ff$^*\circ$ fF $\circ$ FF*) A.

```
Algorithm Compute_SLR_Action_Table(t):
        Input:      LR(0) automaton, ff, fF, FF;
        Output:     ACTION: vector indexed by states;
var     t_in_First: bit vector indexed by symbols;
        t_in_Follow: bit vector indexed by nonterminals;

procedure Follow_to_Follow(A):
        begin
                if t_in_Follow[A] then return;
                set t_in_Follow[A];
                for each (q,A→ ω) do
                        Add "Reduce/A→ ω" to ACTION[q];
{FF*}                   for each B such that A FF B do
                        Follow_to_Follow(B);
        end;

procedure First_to_First(X):
        begin
                if t_in_First[X] then return;
                set t_in_First[X];
{fF}            for each A such that X fF A do
                        Follow_to_Follow(A);
                for each Y such that X ff Y do
{ff*}                   First_to_First(Y);
        end;


begin
        clear t_in_First[X], for each symbol X;
        clear t_in_Follow[A], for each nonterminal A;
        First_to_First(t);      {ff* ◦ fF ◦ FF*}.
end;
```

Conclusions.

- SLR(1) parsers can be generated in a totally parallel fashion.
- LALR(1) can be similarly generated.
- A fairly large number of processors can be productively used.
- Speedup factor: min(processors,terminals).
- The storage requirements of each parallel processor are reasonable.