SLR parsing (simplest LR parse)

Grammar $G = (\underline{\Sigma}, \underline{V}, \underline{P}, \underline{S})$

Clean it: Introduce new start symbol S' and add one production:

$$S' \to S$$

Guarantees S' appears nowhere else. Resulting grammar is G'.

Recall: an LR(0) item is an expr of the form

$$[A \to \alpha . \beta]$$

where α, β are strings of grammar symbols and $A \to \alpha\beta$ is a production of G'.

A state of the parser is a set of LR(0) items

Ex: $A \to b\,c\,d$

$A \to b\,c\,e$

After ~~seeing~~ consuming b and seeing c on lookahead, state will be

$$\{[A \to b.cd], [A \to b.ce]\}$$

- start state
- build a set of states
- transition function

trans(s, X) $\Longrightarrow$ s' (X: grammar symbol; s, s': states)

[book: goto fcn]

Apr 13-11:04 AM

Apr 13-11:15 AM

- action table

action[s, a] (a: terminal symbol or \$ ← end of input marker)

Start state

$I_0 = \{[S' \to .S], \ldots\}$ (only kernel item)

A kernel item is either $[S' \to .S]$ or of the form $[A \to \alpha . \beta]$ for some production $A \to \alpha\beta$ with $\alpha \neq \varepsilon$.

Everything else is called a non-kernel item.

Let I be a set of ~~states~~ LR(0) items. The closure of I $cl(I)$ is the smallest set of LR(0) items such that:

If $[A \to \alpha . B\beta] \in cl(I)$ where B is a nonterminal, and $B \to \gamma$ is some production with head B, then $[B \to .\gamma] \in cl(I)$.

nonkernel item

When building a state, start with kernel items, then take the closure, which may add some nonkernel items.

$I_0 := cl(\{[S' \to .S]\})$

Apr 13-11:15 AM

Example:

$E' \to E$

$E \to E + T \mid T$

$T \to T * F \mid F$

$F \to (E) \mid c$

$I_0 = \{[E' \to .E], [E \to .E+T], [E \to .T],$
$[T \to .T*F], [T \to .F],$
$[F \to .(E)], [F \to .c]\}$

Let s be a state (s is a set of LR(0) items)

Let X be some grammar symbol (either terminal or nonterminal).

Define trans(s, X) (goto) as follows:

0. trans(s, X) := $\emptyset$
1. For every item in s of the form $[A \to \alpha . X\beta]$ for some α, β, A, put $[A \to \alpha X . \beta]$ into trans(s, X) ← kernel item
2. take the closure: trans(s, X) := cl(trans(s, X)) to include nonkernel items

trans(I_0, E) = $\{[E' \to E.],$
$[E \to E.+T]\} =: I_1$

trans(I_0, T) = $\{[E \to T.],$
$[T \to T.*F]\} =: I_2$

Apr 13-11:27 AM

trans(I_0, F)
$= \{[T \to F.]\} =: I_3$

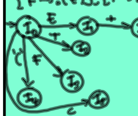
trans(I_0, '(')
$= \{[F \to (.E)], [E \to .E+T],$
$[E \to .T], [T \to .T*F],$
$[T \to .F], [F \to .(E)], [F \to .c]\}$
$=: I_4$

trans(I_0, c) =
$\{[F \to c.]\} =: I_5$

All other trans(I_0, X) sets are empty.

Recall: $I_1 = \{[E' \to E.], [E \to E.+T]\}$

trans(I_1, +) = $\{[E \to E+.T],$
$[T \to .T*F], [T \to .F],$
$[F \to .(E)], [F \to .c]\} =: I_6$



Actions

types:

1. shift s' ~~and push it onto stack~~
read one token, then goto state s' (push s' onto stack)
2. reduce $A \to \beta$
pop $|\beta|$ many items off of stack, exposing some state t. Push trans(t, A) onto stack

Apr 13-11:44 AM

action types (cont.)

3. accept (successful completion of parse)
4. error (unable to parse — no parse tree exists for input)

Initially: stack has one item: start state I_0

Build action table as follows:

Let s be some state and let a be some terminal or \$.

1. If $[S' \to S.] \in s$ then set action[s, \$] := "accept" (only place accept should occur)
2. If $[A \to \alpha . a\beta] \in s$ the set action[s, a] := shift s' where s' = trans(s, a)
3. If $[A \to \gamma .] \in s$ and $a \in$ FOLLOW(A) set action[s, a] = "reduce $A \to \gamma$"
4. Any as yet unset action table entries are now set to error.

Conflicts:

action[s, a] is set to two or more things for some s, a.

A conflict means the grammar is not an SLR grammar.

Apr 13-12:03 PM