Syntax Analysis Or Parsing

Lecture 08

LR(1) Item

- To avoid some of invalid reductions, the states need to carry more information.

- Extra information is put into a state by including a terminal symbol as a second component in an item.

- A LR(1) item is:

  $A \rightarrow \alpha \bullet \beta, a$ where **a** is the look-head of the LR(1) item

  (**a** is a terminal or end-marker.)

LR (1) ITEMS

Just like before, except...

- Look-ahead symbol
- Terminal symbol from grammar

The look-ahead symbol

Grammar:

1. $E \rightarrow E + T$
2. $E \rightarrow T$
3. $T \rightarrow T * F$
4. $T \rightarrow F$
5. $F \rightarrow (E)$
6. $F \rightarrow \underline{\texttt{id}}$

Examples:

$E \rightarrow \bullet E + T\ ,\ )$
$E \rightarrow \bullet E + T\ ,\ \$$
$E \rightarrow E \bullet + T\ ,\ )$
$E \rightarrow E \bullet + T\ ,\ \$$
. . .

Intuition behind LR (1) Items

1. E → E + T
2. E → T
3. T → T * F
4. T → F
5. F → (E)
6. F → id

F → (• E) ,)

We were hoping / expecting to see an F next, followed by a)
and we have already seen a (.
We are on the path to finding an F, followed by a) .
Using rule 5, one way to find an F is to find (E) next.
So now we are looking for E) , followed by a) .

F → (E) • ,)

We were looking for an F, followed by a)
and we have found (E)
If a) comes next then the parse is going great!
... Now reduce, using rule F → (E)

INTUITION BEHIND LR (1) ITEMS

1. E → E + T
2. E → T
3. T → T * F
4. T → F
5. F → (E)
6. F → id

F → • (E) ,)

It would be legal at this point in the parse
to see an F, followed by a) .
Using rule 5, one way to find an F is to find (E) next.
So, among other possibilities, we are looking for (E) , followed by a) .
If a (comes next, then let's scan it and keep going,
looking for E) , followed by a) .
If we get E) later, then we will be able to reduce it to F
... but we may get something different (although perfectly legal).

E → • T ,)

It would be legal at this point in the parse
to see an E, followed by a) .
Using rule 2, one way to find an E is to find T next.
So, among other possibilities, we are looking for a T followed by a) .
And how can we find a T followed by a) ?

T → • T * F ,)

T → • F ,)

The CLOSURE Function

0. S' → E
1. E → E + T
2. E → T
3. T → T * F
4. T → F
5. F → (E)
6. F → id

Let's say we have this item:

E → • T,)

What are the ways to find a T?

T → F

T → T * F

We are looking for a T, followed by a) , so we'll need to add these items:

T → • F,)

T → • T * F,)

We can find a T followed by a) if we find an F following by a) .
How can we find that?

F → • (E),)

F → • id ,)

We can also find a T followed by a) if we find an T * F following by a) .
To find that, we need to first find another T, but followed by *.

T → • F, *

T → • T * F, *

So we should also look for a F followed by a *.

F → • (E), *

F → • id , *

THE CLOSURE FUNCTION

Given:

I = a set of items

Output:

CLOSURE(I) = a new set of items

Algorithm:

```
result = {}
add all items in I to result
repeat
  for every item A → β • C δ , a in result do
    for each rule C → γ in the grammar do
      for each b in FIRST(δa) do
        add C → • γ , b to result
      endFor
    endFor
  endFor
until we can't add anything more to result
```

LR (1) Example

- S -> CC
- C -> cC
- C -> d

THE GOTO FUNCTION

Let I be a set of items...

Let X be a grammar symbol (terminal or non-terminal)...

```
function GOTO(I,X) returns a set of items
   result = {}
   look at all items in I...
     if A → α•Xδ,a is in I
        then add A → αX•δ,a to result
   result = CLOSURE(result)
```

In other words, move the dot past the X in any items where it is in front of an X

...and take the CLOSURE of whatever items you get

Intuition:

- I is a set of items indicating where we are so far, after seeing some prefix γ of the input.
- I describes what we might legally see next.
- Assume we get an X next.
- Now we have seen some prefix γX of the input.
- GOTO(I, X) tells what we could legally see after that.
- GOTO(I, X) is the set of all items that are "valid" for prefix γX.

CONSTRUCTION OF LR(1) PARSING TABLES

1. Construct the canonical collection of sets of LR(1) items for G'.
   $C \leftarrow \{I_0,\ldots,I_n\}$

2. Create the parsing action table as follows
   - If a is a terminal, $A \rightarrow \alpha \bullet a\beta,b$ in I_i and $goto(I_i,a)=I_j$ then action[i,a] is ***shift j.***
   - If $A \rightarrow \alpha \bullet ,a$ is in I_i , then action[i,a] is ***reduce*** $\boldsymbol{A \rightarrow \alpha}$ where $A \neq S'$.
   - If $S' \rightarrow S \bullet ,\$$ is in I_i , then action[i,$] is ***accept***.
   - If any conflicting actions generated by these rules, the grammar is not LR(1).

3. Create the parsing goto table
   - for all non-terminals A, if $goto(I_i,A)=I_j$ then goto[i,A]=j

4. All entries not defined by (2) and (3) are errors.

5. Initial state of the parser contains $S' \rightarrow .S,\$$

LALR Parsing Tables

- **LALR** stands for **LookAhead LR.**

- LALR parsers are often used in practice because LALR parsing tables are smaller than LR(1) parsing tables.
- The number of states in SLR and LALR parsing tables for a grammar G are equal.
- But LALR parsers recognize more grammars than SLR parsers.
- A state of LALR parser will be again a set of LR(1) items.

Creating LALR Parsing Tables

Canonical LR(1) Parser € LALR Parser

shrink # of states

- This shrink process may introduce a **reduce/reduce** conflict in the resulting LALR parser (so the grammar is NOT LALR)
- But, this shrink process does not produce a **shift/reduce** conflict.

The Core of A Set of LR(1) Items

- The core of a set of LR(1) items is the set of its first component.

Ex: $S \rightarrow L \cdot {=}R, \$$ $S \rightarrow L \cdot {=}R$ **Core**

$R \rightarrow L \cdot , \$$ $R \rightarrow L \cdot$

- We will find the states (sets of LR(1) items) in a canonical LR(1) parser with same cores. Then we will merge them as a single state.

$I_1: L \rightarrow id \cdot , =$ **Same Core**   A new state: $I_{12}: L \rightarrow id \cdot , =$

$L \rightarrow id \cdot , \$$

$I_2: L \rightarrow id \cdot , \$$ **Merge Them**

- We will do this for all states of a canonical LR(1) parser to get the states of the LALR parser.
- In fact, the number of the states of the LALR parser for a grammar will be equal to the number of states of the SLR

Creation of LALR Parsing Tables

- Create the canonical LR(1) collection of the sets of LR(1) items for the given grammar.
- Find each core; find all sets having that same core; replace those sets having same cores with a single set which is their union.

  $C=\{I_0,...,I_n\}$ € $C'=\{J_1,...,J_m\}$ where $m \leq n$
- Create the parsing tables (action and goto tables) same as the construction of the parsing tables of LR(1) parser.
  - Note that: If $J=I_1 \cup ... \cup I_k$ since $I_1,...,I_k$ have same cores

    € cores of $goto(I_1,X),...,goto(I_2,X)$ must be same.
  - So, $goto(J,X)=K$ where K is the union of all sets of items having same cores as $goto(I_1,X)$.
- If no conflict is introduced, the grammar is LALR(1) grammar. (We may only introduce reduce/reduce conflicts; we cannot introduce a shift/reduce conflict)

Shift/Reduce Conflict

- We say that we cannot introduce a shift/reduce conflict during the shrink process for the creation of the states of a LALR parser.
- Assume that we can introduce a shift/reduce conflict. In this case, a state of LALR parser must have:

$$A \rightarrow \alpha \,\blacksquare ,a \text{ and } \quad B \rightarrow \beta \,\blacksquare\, a\gamma ,b$$

- This means that a state of the canonical LR(1) parser must have:

$$A \rightarrow \alpha \,\blacksquare ,a \text{ and } \quad B \rightarrow \beta \,\blacksquare\, a\gamma ,c$$

But, this state has also a shift/reduce conflict. i.e. The original canonical LR(1) parser has a conflict.

(Reason for this, the shift operation does not depend on lookaheads)

Reduce/Reduce Conflict

- But, we may introduce a reduce/reduce conflict during the shrink process for the creation of the states of a LALR parser.

$I_1 : A \to \alpha \,\blacksquare\, ,a$ $\qquad$ $I_2 : A \to \alpha \,\blacksquare\, ,b$

$B \to \alpha \,\blacksquare\, ,b$ $\qquad$ $B \to \alpha \,\blacksquare\, ,c$

$$\Downarrow$$

$I_{12} : A \to \alpha \,\blacksquare\, ,a/b$ $\qquad$ € reduce/reduce conflict

$B \to \alpha \,\blacksquare\, ,b/c$

Any Questions ?