Simple LR

SLR parser.

$E \rightarrow E+T$

$E \rightarrow T$

$T \rightarrow T*F$

$T \rightarrow F$

$F \rightarrow (E)$

$F \rightarrow id$

Augmented Grammar

$E' \rightarrow E$

$E \rightarrow E+T$

$E \rightarrow T$

$T \rightarrow T*F$

$T \rightarrow F$

$F \rightarrow (E)$

$F \rightarrow id$

Cannonoical LR(0) Collection:

I_0: $E' \rightarrow \cdot E$
$E \rightarrow \cdot E+T$
$E \rightarrow \cdot T$
$T \rightarrow \cdot T*F$
$T \rightarrow \cdot F$
$F \rightarrow \cdot (E)$
$F \rightarrow \cdot id$

goto (I_0, E)

I_1: $E' \rightarrow E \cdot$
$E \rightarrow E \cdot +T$

goto (I_0, T)

I_2:
$E \rightarrow T \cdot$
$T \rightarrow T \cdot *F$

goto (I_0, F)

I_3: $T \rightarrow F \cdot$

goto $(I_0, ( )$

I_4: $F \rightarrow (\cdot E)$
$E \rightarrow \cdot E+T$
$E \rightarrow \cdot T$
$T \rightarrow \cdot T*F$
$T \rightarrow \cdot F$
$F \rightarrow \cdot (E)$
$F \rightarrow \cdot id$

goto (I_0, id)

I_5: $F \rightarrow id \cdot$

goto $(I_1, +)$

I_6 $E \rightarrow E+ \cdot T$
$T \rightarrow \cdot T*F$
$T \rightarrow \cdot F$
$F \rightarrow \cdot (E)$
$F \rightarrow \cdot id$

goto $(I_2, *)$

I_7: $T \rightarrow T* \cdot F$
$F \rightarrow \cdot (E)$
$F \rightarrow \cdot id$

goto (I_4, E)

I_8: $F \rightarrow (E \cdot)$
$E \rightarrow E \cdot +T$

goto (I_4, T)

I_9: $E \rightarrow T \cdot$
$T \rightarrow T \cdot *F$

goto (I_4, F)

I_{10}: $T \rightarrow F \cdot$

goto (I_7, F)

I_{11}: $T \rightarrow T*F \cdot$

goto(I_6, T)

I_9: E → E+T·

T → T·*F

goto (I_7, F)

I_{10}: T → T*F·

goto(I_8,))

I_{11}: F → (E)·

In Cat-2 Question

11. Three address code, types and its implementation
12. Boolean expression
13. Optimization of basic blocks
14. Simple code generation & DAG construction
15. Issue in design of code generation
16. SLR Parser

In goto
see the
Production
(rightside)

~~code~~

2m

1. Compare three kinds of LR Parser
2. What is Handle pruning?
3. Application of DAG
4. Various representation of intermediate code.
5. Define closure & goto operation
6. Statement, convert it into 3-address code
7. Construct syntax tree for the
8. Code motion & Dead code elimination
9. Describe register & Address descriptor
10. Define basic blocks & flowgraphs.

& Instruction codst