40-414 Compiler Design

Other Bottom-Up Parsing Methods

Lecture 15

Outline

- Operator Precedence
- Simple Precedence

Operator Precedence

- Problems in shift/reduce parsing:
  - Deciding when to perform which operation (shift, reduce, etc.)
  - Identifying HANDLE in the reduce operation.
  - operator grammars: a class of grammars where handle identification and conflict resolution is easy.

- Operator Grammars: no production right side is $\in$ or has two adjacent non-terminals.

$$E \rightarrow E - E \mid E + E \mid E * E \mid E / E \mid E \,\hat{}\, E \mid - E \mid (E) \mid id$$

- note: this is typically ambiguous grammar.

Basic Technique

- For the terminals of the grammar, define the relations <. .> and .=.
- a <. b means that *a* yields precedence to *b*
- a .=. b means that *a* has the same precedence as *b*.
- a .> b means hat *a* takes precedence over *b*
- E.g. * .> + or + <. *
- Many handles are possible. We will use <. .=. and .> to find the correct handle (i.e., the one that respects the precedence).

Using Operator-Precedence Relations

- GOAL: delimit the handle of a right sentential form
- <. will mark the beginning, .> will mark the end and .=. will be in between.
- Since no two adjacent non-terminals appear in the RHS of any production, the general form sentential forms is as:

  $\beta_0\ a_1\ \beta_1\ a_2\ \beta_2\ \ldots\ a_n\ \beta_n$, where each β_i is either a nonterminal or the empty string.
- At each step of the parse, the parser considers the top most terminal of the parse stack (i.e., either top or top-1), say a, and the current token, say b, and looks up their precedence relation, and decides what to do next.

Operator-Precedence Parsing

1. If a .=. b, then shift b into the parse stack
2. If a <. b, then shift <. And then shift b into the parse stack
3. If a .> b, then find the top most <. relation of the parse stack; the string between this relation (with the non-terminal underneath, if there exists) and the top of the parse stack is the handle (the handle should match (weakly) with the RHS of at least one grammar rule); replace the handle with a typical non-terminal

Example

STACK	INPUT	Relation
$	id + id * id $	$ <. id

	+	*	(	)	id	$
+	.>	<.	<.	.>	<.	.>
*	.>	.>	<.	.>	<.	.>
(	<.	<.	<.	.=.	<.	
)	.>	.>		.>		.>
id	.>	.>		.>		.>
$	<.	<.	<.		<.	.=.

Parse Table

1-2 E $\rightarrow$ E + T | T
3-4 T $\rightarrow$ T * F | F
5-6 F $\rightarrow$ (E) | id

Example (Cont.)

STACK	INPUT	Relation
$	id + id * id $	$ <. id
$ <. id	+ id * id $	id >. +

	+	*	(	)	id	$
+	.>	<.	<.	.>	<.	.>
*	.>	.>	<.	.>	<.	.>
(	<.	<.	<.	.=.	<.	
)	.>	.>		.>		.>
id	.>	.>		.>		.>
$	<.	<.	<.		<.	.=.

Parse Table

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

Example (Cont.)

STACK	INPUT	Relation
$	id + id * id $	$ <. id
$ <. id	+ id * id $	id >. +
$ E	+ id * id $	$ <. +

	+	*	(	)	id	$
+	.>	<.	<.	.>	<.	.>
*	.>	.>	<.	.>	<.	.>
(	<.	<.	<.	.=.	<.	
)	.>	.>		.>		.>
id	.>	.>		.>		.>
$	<.	<.	<.		<.	.=.

Parse Table

1-2 $E \rightarrow E + T \mid T$

3-4 $T \rightarrow T * F \mid F$

5-6 $F \rightarrow (E) \mid id$

Example (Cont.)

STACK	INPUT	Relation
$	**id + id * id $**	**$ <. id**
$ <. id	**+ id * id $**	**id >. +**
$ E	**+ id * id $**	**$ <. +**
$ E <. +	**id * id $**	**+ <. id**

	+	*	(	)	id	$
+	.>	<.	<.	.>	<.	.>
*	.>	.>	<.	.>	<.	.>
(	<.	<.	<.	.=.	<.	
)	.>	.>		.>		.>
id	.>	.>		.>		.>
$	<.	<.	<.		<.	.=.

Parse Table

1-2 $E \rightarrow E + T \mid T$

3-4 $T \rightarrow T * F \mid F$

5-6 $F \rightarrow (E) \mid id$

Example (Cont.)

STACK	INPUT	Relation
$	id + id * id $	$ <. id
$ <. id	+ id * id $	id >. +
$ E	+ id * id $	$ <. +
$ E <. +	id * id $	+ <. id
$ E <. + <. id	* id $	id >. *

	+	*	(	)	id	$
+	.>	<.	<.	.>	<.	.>
*	.>	.>	<.	.>	<.	.>
(	<.	<.	<.	.=.	<.	
)	.>	.>		.>		.>
id	.>	.>		.>		.>
$	<.	<.	<.		<.	.=.

Parse Table

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

Example (Cont.)

STACK	INPUT	Relation
$	id + id * id $	$ <. id
$ <. id	+ id * id $	id >. +
$ E	+ id * id $	$ <. +
$ E <. +	id * id $	+ <. id
$ E <. + <. id	* id $	id >. *
$ E <. + E	* id $	+ <. *

	+	*	(	)	id	$
+	.>	<.	<.	.>	<.	.>
*	.>	.>	<.	.>	<.	.>
(	<.	<.	<.	.=.	<.	
)	.>	.>		.>		.>
id	.>	.>		.>		.>
$	<.	<.	<.		<.	.=.

Parse Table

1-2 E $\rightarrow$ E + T | T
3-4 T $\rightarrow$ T * F | F
5-6 F $\rightarrow$ (E) | id

Example (Cont.)

STACK	INPUT	Relation
$	id + id * id $	$ <. id
$ <. id	+ id * id $	id >. +
$ E	+ id * id $	$ <. +
$ E <. +	id * id $	+ <. id
$ E <. + <. id	* id $	id >. *
$ E <. + E	* id $	+ <. *
$ E <. + E <. *	id $	* <. id

	+	*	(	)	id	$
+	.>	<.	<.	.>	<.	.>
*	.>	.>	<.	.>	<.	.>
(	<.	<.	<.	.=.	<.	
)	.>	.>		.>		.>
id	.>	.>		.>		.>
$	<.	<.	<.		<.	.=.

Parse Table

1-2 $E \rightarrow E + T \mid T$

3-4 $T \rightarrow T * F \mid F$

5-6 $F \rightarrow (E) \mid id$

Example (Cont.)

STACK	INPUT	Relation
$	id + id * id $	$ <. id
$ <. id	+ id * id $	id >. +
$ E	+ id * id $	$ <. +
$ E <. +	id * id $	+ <. id
$ E <. + <. id	* id $	id >. *
$ E <. + E	* id $	+ <. *
$ E <. + E <. *	id $	* <. id
$ E <. + E <. * <. id	$	id >. $

	+	*	(	)	id	$
+	.>	<.	<.	.>	<.	.>
*	.>	.>	<.	.>	<.	.>
(	<.	<.	<.	.=.	<.	
)	.>	.>		.>		.>
id	.>	.>		.>		.>
$	<.	<.	<.		<.	.=.

Parse Table

1-2 $E \rightarrow E + T \mid T$
3-4 $T \rightarrow T * F \mid F$
5-6 $F \rightarrow (E) \mid id$

Example (Cont.)

STACK	INPUT	Relation
$	id + id * id $	$ <. id
$ <. id	+ id * id $	id >. +
$ E	+ id * id $	$ <. +
$ E <. +	id * id $	+ <. id
$ E <. + <. id	* id $	id >. *
$ E <. + E	* id $	+ <. *
$ E <. + E <. *	id $	* <. id
$ E <. + E <. * <. id	$	id >. $
$ E <. + E <. * E	$	* >. $

	+	*	(	)	id	$
+	.>	<.	<.	.>	<.	.>
*	.>	.>	<.	.>	<.	.>
(	<.	<.	<.	.=.	<.	
)	.>	.>		.>		.>
id	.>	.>		.>		.>
$	<.	<.	<.		<.	.=.

Parse Table

1-2 $E \rightarrow E + T \mid T$

3-4 $T \rightarrow T * F \mid F$

5-6 $F \rightarrow (E) \mid id$

Example (Cont.)

STACK	INPUT	Relation
$	id + id * id $	$ <. id
$ <. id	+ id * id $	id >. +
$ E	+ id * id $	$ <. +
$ E <. +	id * id $	+ <. id
$ E <. + <. id	* id $	id >. *
$ E <. + E	* id $	+ <. *
$ E <. + E <. *	id $	* <. id
$ E <. + E <. * <. id	$	id >. $
$ E <. + E <. * E	$	* >. $
$ E <. + E	$	+ >. $

	+	*	(	)	id	$
+	.>	<.	<.	.>	<.	.>
*	.>	.>	<.	.>	<.	.>
(	<.	<.	<.	.=.	<.	
)	.>	.>		.>		.>
id	.>	.>		.>		.>
$	<.	<.	<.		<.	.=.

Parse Table

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

Example (Cont.)

STACK	INPUT	Relation
$	id + id * id $	$ <. id
$ <. id	+ id * id $	id >. +
$ E	+ id * id $	$ <. +
$ E <. +	id * id $	+ <. id
$ E <. + <. id	* id $	id >. *
$ E <. + E	* id $	+ <. *
$ E <. + E <. *	id $	* <. id
$ E <. + E <. * <. id	$	id >. $
$ E <. + E <. * E	$	* >. $
$ E <. + E	$	+ >. $
$ E	$	accept

	+	*	(	)	id	$
+	.>	<.	<.	.>	<.	.>
*	.>	.>	<.	.>	<.	.>
(	<.	<.	<.	.=.	<.	
)	.>	.>		.>		.>
id	.>	.>		.>		.>
$	<.	<.	<.		<.	.=.

Parse Table

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

Producing Parse Table

- $FirstTerm(A) = \{a \mid A \Rightarrow^{+} a\alpha \text{ or } A \Rightarrow^{+} Ba\alpha\}$
- $LastTerm(A) = \{a \mid A \Rightarrow^{+} \alpha a \text{ or } A \Rightarrow^{+} \alpha aB\}$

- a .=. b iff $\exists\ U \rightarrow \alpha ab\beta$ or $\exists\ U \rightarrow \alpha aBb\beta$

- a <. b iff $\exists\ U \rightarrow \alpha aB\beta$ and $b \in FirsTerm(B)$

- a .> b iff $\exists\ U \rightarrow \alpha Bb\beta$ and $a \in LastTerm(B)$

Example:

- FirstTerm (E) = {+, *, id, (}
- FirstTerm (T) = {*, id, (}
- FirstTerm (F) = {id, (}

- LastTerm (E) = {+, *, id,)}
- LastTerm (T) = {*, id,)}
- LastTerm (F) = {id,)}

1-2 $E \rightarrow E + T \mid T$
3-4 $T \rightarrow T * F \mid F$
5-6 $F \rightarrow (E) \mid id$

Precedence Functions vs Relations

	+	-	*	/	$\uparrow$	(	)	id	$
f	2	2	4	4	4	0	6	6	0
g	1	1	3	3	5	5	0	5	0

- f(a) < g(b) whenever a <. b
- f(a) = g(b) whenever a .=. b
- f(a) > g(b) whenever a .> b

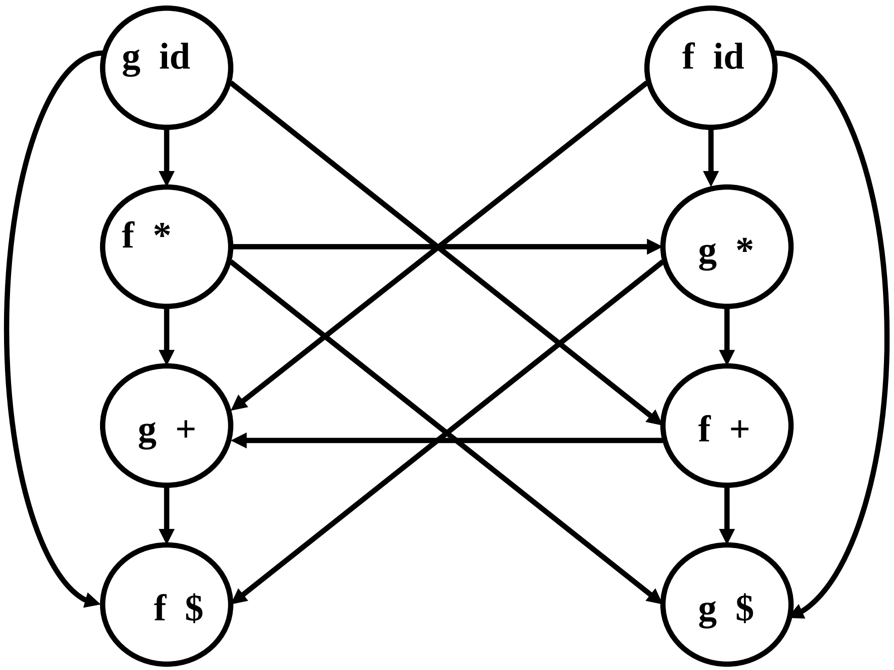
1-3 $E \rightarrow E + T \mid E - T \mid T$

4-6 $T \rightarrow T * F \mid T / F \mid F$

7-8 $F \rightarrow P \uparrow F \mid P$

9-10 $P \rightarrow (E) \mid id$

Constructing Precedence Functions



	+	*	id	$
f	2	4	4	0
g	1	3	5	0

1-2 $E \rightarrow E + T \mid T$
3-4 $T \rightarrow T * F \mid F$
5-6 $F \rightarrow (E) \mid id$

Handling Errors During Reductions

- Suppose *abEc* is poped and there is no production right hand side that matches *abEc*
- If there were a rhs *aEc*, we might issue message
  illegal b on line x
- If the rhs is *abEdc*, we might issue message
  missing d on line x
- If the found rhs is *abc*, the error message could be
  illegal E on line x,
  where E stands for an appropriate syntactic category represented by non-terminal E

Handling Shift/Reduce Errors

e1: /* called when whole expression is missing */

insert id onto the input

print "missing operand

e2: /* called when expression begins with a right parenthesis */

delete) from the input

print "unbalanced right parenthesis"

e3": /* called when id or) is followed by id or (*/

insert + onto the input

print "missing operator

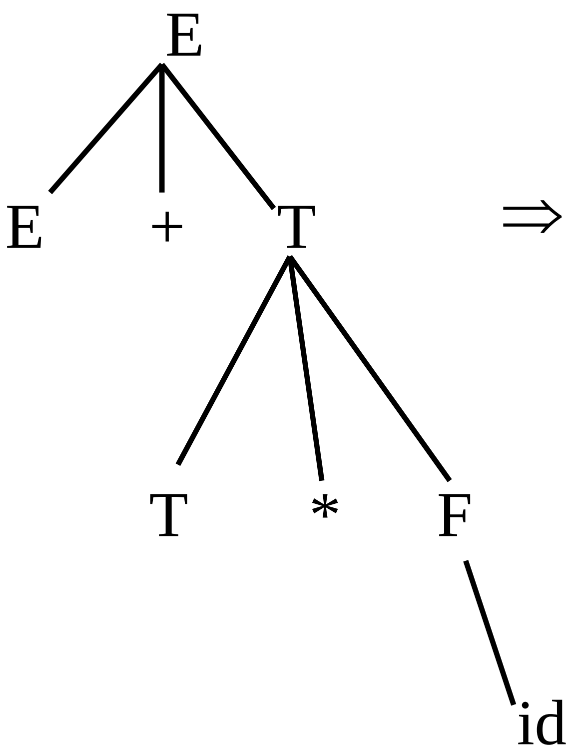
e4: /* called when expression ends with a left parenthesis */

pop (from the stack

print "missing right parenthesis"

	id	(	)	$
id	e3	e3	.>	.>
(	<..	<.	.=.	e4
)	e3	e3	.>	.>
$	<.	<.	e2	e1

Extracting Precedence relations from parse trees

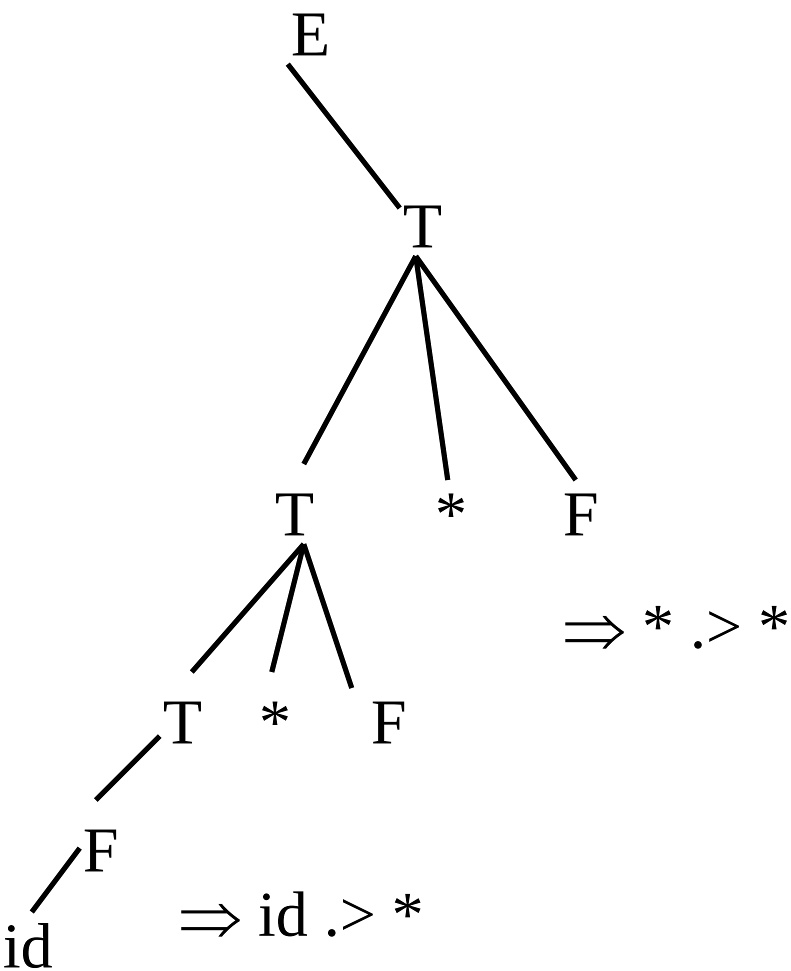


$\Rightarrow + <. *$

$\Rightarrow * <. id$

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

Extracting Precedence relations from parse trees (Cont.)



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

Pros and Cons

- + simple implementation
- + small parse table
- - weak (too restrictive for not allowing two adjacent non-terminals
- - not very accurate (some syntax errors are not detected due weak treatment of non-terminals)
- Simple Precedence parsing is an improved form of operator precedence that doesn't have these weaknesses

Other Parsing Methods

Bottom-Up Parsing Methods (Cont.)

Simple Precedence

Simple Precedence

- Less restricted than Operator precedence
  - We may have adjacent not-terminals on rhs
- More accurate than Operator Precedence
  - All syntax errors are found
  - However, unlike LL(1) and LR(1), illegal inputs may be shifted onto the stack before that the parser recognizes the error
- Precedence relations are defined between all symbols (i.e., both terminals and non-terminals)
- Restrictions:
  - no production right side is $\in$
  - Rules cannot have the same rhs.

Simple Precedence

- For all symbols of the grammar, define the relations $⋖$, $⋗$ and $≐$
- X $⋖$ Y means that *X* yields precedence to *Y*
- X $≐$ Y means that *X* has the same precedence as *Y*
- X $⋗$ Y means hat *X* takes precedence over *Y*
- Similar to OP, we will use these relations to find the correct handle

Parsing Algorithm

- Let X be the top most symbol in stack and b be the current token,
- Look up their precedence relation, and decide what to do next:
  - If X = b, then shift b into the parse stack
  - If X < b, then shift < and then shift b into the parse stack
  - If X > b, then find the top most < relation of the parse stack; the string between this relation and the top of the stack is the handle (the handle should match (exactly) with the RHS of at least one grammar rule); let *top* be the top symbol of the stack after deleting the handle; let *LHS* be the left hand of the rule whose rhs has matched the handle; look up the precedence relation between *top* and *LHS* and perform the following:

Parsing Algorithm (Cont.)

- If top $\circledequal$ LHS, then shift LHS into the parse stack
- If top $\lessdot$ LHS, then shift $\lessdot$ and then shift LHS into the parse stack
- If top $\gtrdot$ LHS, a syntax error has occurred!
- In fact, no symbol can have a higher precedence than a non-terminal (Why?)

Example

STACK	INPUT	Relation
$	(c c) $	$ < (

	S	(	)	c	$
S	=	<	=	<	=
(	=	<		<	
)	>	>	>	>	>
c	>	>	>	>	>
$	=	<		<	

Parse Table

S → (S S) | c

Example (Cont.)

STACK	INPUT	Remark
$	(c c) $	$ < (
$ < (	c c) $	(< c

	S	(	)	c	$
S	=	<	=	<	=
(	=	<		<	
)	>	>	>	>	>
c	>	>	>	>	>
$	=	<		<	

Parse Table

$$S \rightarrow (\, S\, S\,) \mid c$$

Example (Cont.)

STACK	INPUT	Remark
$	(c c) $	$ < (
$ < (	c c) $	(< c
$ < (< c	c) $	c > c

	S	(	)	c	$
S	=	<	=	<	=
(	=	<		<	
)	>	>	>	>	>
c	>	>	>	>	>
$	=	<		<	

Parse Table

$$S \rightarrow (S S) \mid c$$

Example (Cont.)

STACK	INPUT	Remark
$	(c c) $	$ < (
$ < (	c c) $	(< c
$ < (< c	c) $	c > c
$ < (S	c) $	(= S, S < c

top = LHS

	S	(	)	c	$
S	=	<	=	<	=
(	=	<		<	
)	>	>	>	>	>
c	>	>	>	>	>
$	=	<		<	

Parse Table

$$S \rightarrow (\, S\ S\,) \mid c$$

Example (Cont.)

STACK	INPUT	Remark
$	(c c) $	$ < (
$ < (	c c) $	(< c
$ < (< c	c) $	c > c
$ < (S	c) $	(= S, S < c
$ < (S < c	) $	c >)

	S	(	)	c	$
S	=	<	=	<	=
(	=	<		<	
)	>	>	>	>	>
c	>	>	>	>	>
$	=	<		<	

Parse Table

$$S \rightarrow (S\ S) \mid c$$

Example (Cont.)

STACK	INPUT	Remark
$	(c c) $	$ < (
$ < (	c c) $	(< c
$ < (< c	c) $	c > c
$ < (S	c) $	(= S, S < c
$ < (S < c	) $	c >)
$ < (S S	) $	S = S, S =)

top = LHS

	S	(	)	c	$
S	=	<	=	<	=
(	=	<		<	
)	>	>	>	>	>
c	>	>	>	>	>
$	=	<		<	

Parse Table

$$S \rightarrow (S\ S) \mid c$$

Example (Cont.)

STACK	INPUT	Remark
$	(c c) $	$ < (
$ < (	c c) $	(< c
$ < (< c	c) $	c > c
$ < (S	c) $	(= S, S < c
$ < (S < c	) $	c >)
$ < (S S	) $	S = S, S =)
$ < (S S)	$	) > $

	S	(	)	c	$
S	=	<	=	<	=
(	=	<		<	
)	>	>	>	>	>
c	>	>	>	>	>
$	=	<		<	

Parse Table

$$S \rightarrow (S \ S) \mid c$$

Example (Cont.)

STACK	INPUT	Remark
$	(c c) $	$ < (
$ < (	c c) $	(< c
$ < (< c	c) $	c > c
$ < (S	c) $	(= S, S < c
$ < (S < c	) $	c >)
$ < (S S	) $	S = S, S =)
$ < (S S)	$	) > $
$ S	$	$ = S, accept

top = LHS

	S	(	)	c	$
S	=	<	=	<	=
(	=	<		<	
)	>	>	>	>	>
c	>	>	>	>	>
$	=	<		<	

Parse Table

$$S \rightarrow (S \; S) \mid c$$

Example (Cont.)

STACK	INPUT	Remark
$	(c c) $	$ < (
$ < (	c c) $	(< c
$ < (< c	c) $	c > c
$ < (S	c) $	(= S, S < c
$ < (S < c	) $	c >)
$ < (S S	) $	S = S, S =)
$ < (S S)	$	) > $
$ S	$	$ = S, accept

	S	(	)	c	$
S	=	<	=	<	=
(	=	<		<	
)	>	>	>	>	>
c	>	>	>	>	>
$	=	<		<	

Parse Table

These should be deleted; because, nothing can be greater than a non-terminal why?

$$S \rightarrow (S S) \mid c$$

Producing Parse Table

- $\text{Head}(A) = \{X \mid A \Rightarrow^{+} X\alpha\}$
- $\text{Tail}(A) = \{X \mid A \Rightarrow^{+} \alpha X\}$

- $X \doteq Y$ iff $\exists\, U \rightarrow \alpha\, X\, Y\, \beta$

- $X \lessdot Y$ iff $\exists\, U \rightarrow \alpha\, X\, B\, \beta$ and $Y \in \text{Head}(B)$

- $X \gtrdot Y$ iff $\exists\, U \rightarrow \alpha\, B\, Y\, \beta$ and $X \in \text{Tail}(B)$ or

  $\exists\, U \rightarrow \alpha\, A\, B\, \beta$ and $X \in \text{Tail}(A)$ and $Y \in \text{Head}(B)$

Example

- Head (E) = {E, T, F, id, (}
- Head (T) = {T, F, id, (}
- Head (F) = {id, (}

- Tail (E) = {T, F, id,)}
- Tail (T) = {F, id,)}
- Tail (F) = {id,)}

1-2 $E \rightarrow E + T \mid T$

3-4 $T \rightarrow T * F \mid F$

5-6 $F \rightarrow (E) \mid id$

Problems with Left and Right Recursions

- if $\exists\ U \rightarrow U\ \gamma$ and $\exists\ U \rightarrow \alpha\ X\ U\ \beta$ then there would be a problem in the parsing table
  - $X \doteq U$ and $X \lessdot U$
- The problem can be resolved by introducing a new non-terminal W, and a new rule

  $W \rightarrow U$, and change the initial rule to $U \rightarrow \alpha\ X\ W\ \beta$
- Also, if $\exists\ U \rightarrow \gamma\ U$ and $\exists\ U \rightarrow \alpha\ U\ X\ \beta$ then there would be a problem in the parsing table
  - $U \doteq X$ and $U \gtrdot X$
- Again, the problem can be resolved by introducing a new non-terminal W, and a new rule

  $W \rightarrow U$, and change the initial rule to $U \rightarrow \alpha\ W\ X\ \beta$

Example

- $E \rightarrow E + T$ and $F \rightarrow (E)$; then
  - ($=$ E and ($<$ E
- W introduce a new non-terminal E1, and a new rule

  $E1 \rightarrow E$, and change the initial rule to $F \rightarrow (E1)$

- Also, $T \rightarrow T * F$ and $E \rightarrow E + T$; then
  - + $=$ T and + $<$ T
- Again, the problem can be resolved by introducing a new non-terminal W, and a new rule

  $W \rightarrow T$, and change the initial rule to $E \rightarrow E + E1$

1-2 $E \rightarrow E + T \mid T$
3-4 $T \rightarrow T * F \mid F$
5-6 $F \rightarrow (E) \mid id$

$\Longrightarrow$

1-2 $E \rightarrow E + T1 \mid T$
3-4 $T \rightarrow T * F \mid F$
5-6 $F \rightarrow (E1) \mid id$
7 $E1 \rightarrow E$
8 $T1 \rightarrow T$

Example (cont.)

- Now, there is a new problem! :-(
  - Rules number 2 and 8 have the same rhs
  - This problem is resolved by changing rule number 2 into $E \rightarrow T1$

1-2 $E \rightarrow E + T \mid T$
3-4 $T \rightarrow T * F \mid F$
5-6 $F \rightarrow (E) \mid id$

$\Rightarrow$

1-2 $E \rightarrow E + T1 \mid T1$
3-4 $T \rightarrow T * F \mid F$
5-6 $F \rightarrow (E1) \mid id$
7 $E1 \rightarrow E$
8 $T1 \rightarrow T$