Compilers

3er course
Spring Term

Alfonso Ortega: alfonso.ortega@uam.es
Enrique Alfonseca: enrique.alfonseca@uam.es

Chapter 4: Syntactic Analyser

Part 4g: Simple *Precedence Grammars*

Precedence grammars

Introduction

- A *precedence grammar* bases the analysis in the precedence relationships among the symbols in the language.
- Using those relationships, the parser will try to decide which is the symbol with the highest precedence, to start building the bottom-up parse.

Precedence grammars

Previous example

The purpose is to build the grammar in such a way that some symbols have precedence over other symbols.

`i+i*i`

For instance, the following grammar does not give precedence of * over +:

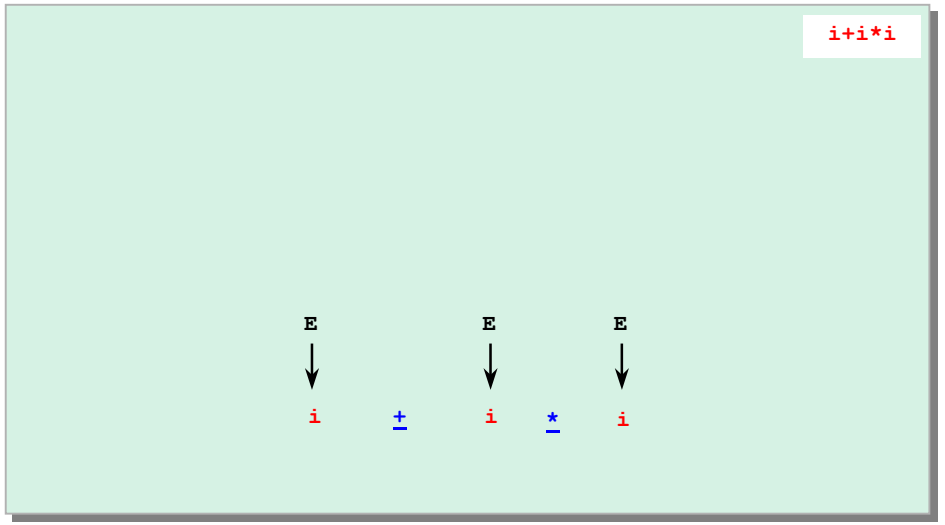
```
E ::= E+E | E*E | i
```

but the rule `E ::= i` has precedence over the other two rules, because they cannot apply if the `i`'s haven't been reduced to `E`'s

i + i * i

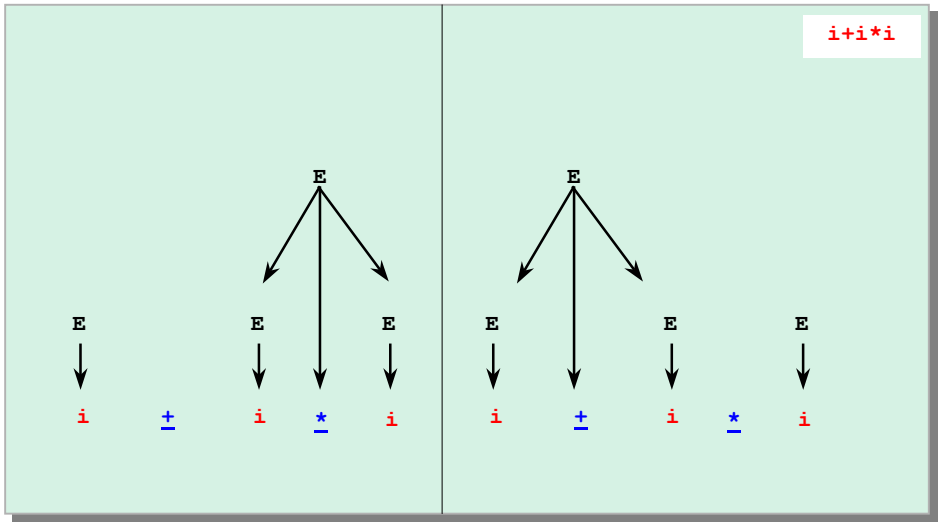
Precedence grammars

Previous example



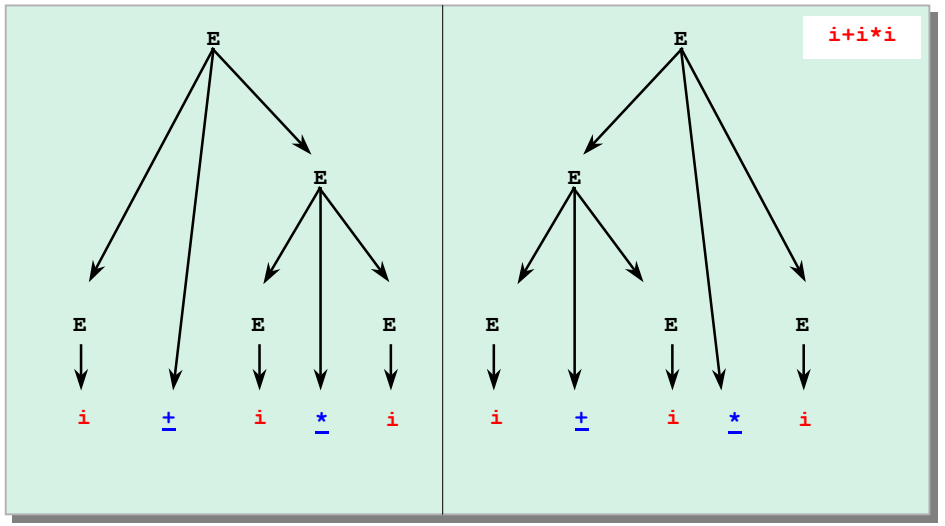
Precedence grammars

Previous example



Precedence grammars

Previous example



Precedence grammars

Idea

In the following slides we shall formalise some relations of precedence between the symbols of the grammar:

- Which symbols have equal precedence.
- Which pairs of symbols are such that the first one has more precedence than the second one.

HEAD relationship

Definition

Let $G=\{\Sigma_N, \Sigma_T, P, A\}$ be a context-independent grammar.

If **U** is a non-terminal symbol in the grammar, we shall call **head(U)** the set of symbols that can be at the beginning of the strings derived from U.

- If there is a rule **U ::= V$x_1x_2...x_n$**, V∈head(U)
- If V∈head(U), and , W∈head(V), then W∈head(U)

HEAD relationship

Matrix representation

E ::= E+T
E ::= T
T ::= T * F
T ::= F
F ::= i
F ::= (E)

	Σ_T					Σ_N		
	i	+	*	(	)	E	T	F
i	0	0	0	0	0	0	0	0
+	0	0	0	0	0	0	0	0
*	0	0	0	0	0	0	0	0
(	0	0	0	0	0	0	0	0
)	0	0	0	0	0	0	0	0
E	1	0	0	1	0	1	1	1
T	1	0	0	1	0	0	1	1
F	1	0	0	1	0	0	0	0

TAIL relationship

Definition

Let $G=\{\Sigma_N, \Sigma_T, P, A\}$ be a context-independent grammar.

If **U** is a non-terminal symbol in the grammar, we shall call **tail(U)** the set of symbols that can be at the end of the strings derived from U.

- If there is a rule **U ::= $x_1x_2...x_n$V**, V∈tail(U)
- If V∈tail(U), and , W∈tail(V), then W∈tail(U)

TAIL relationship

Matrix representation

E ::= E+T
E ::= T
T ::= T * F
T ::= F
F ::= i
F ::= (E)

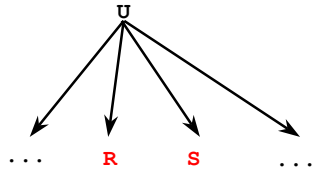
	Σ_T					Σ_N		
	i	+	*	(	)	E	T	F
i	0	0	0	0	0	0	0	0
+	0	0	0	0	0	0	0	0
*	0	0	0	0	0	0	0	0
(	0	0	0	0	0	0	0	0
)	0	0	0	0	0	0	0	0
E	1	0	0	0	1	0	1	1
T	1	0	0	0	1	0	0	1
F	1	0	0	0	1	0	0	0

Precedence grammars

Precedence relationships

- Two symbols R and S, in a grammar, have equal precedence (=.) if and only if there is any rule in the grammar in which they appear consecutive

`R =. S` if and only if there is a rule (`U ::= xRSy`)



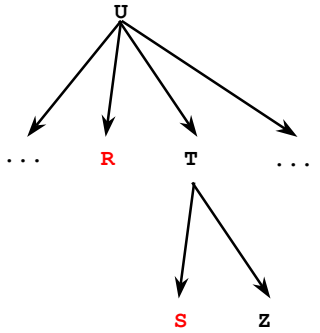
Precedence grammars

Precedence relationships

- A symbol R has less precedence than a symbol S (R <. S) if and only if there is a rule in the grammar

`U ::=` $w_1 R T w_2$, where $w_1, w_2 \in (\Sigma_N+\Sigma_T)^*$

and `S ∈ Head(T)`



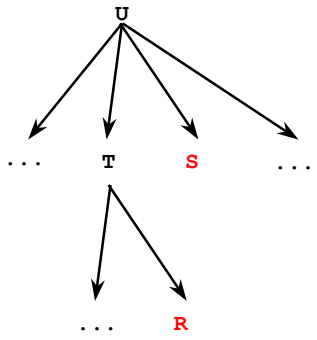
Precedence grammars

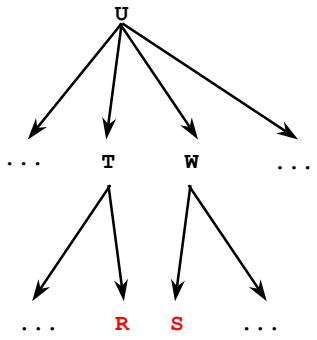
Precedence relationships

- A symbol R has more precedence than a symbol S (R >. S) if and only if there is a rule in the grammar

`U ::=` $w_1 T W w_2$, where $w_1, w_2 \in (\Sigma_N+\Sigma_T)^*$

- `R ∈ Tail(T)`
- Either `S ∈ Head(W)`, or `S=W`.





Precedence grammars

Precedence relationships

- A grammar is a “simple precedence grammar” if, for every pair of symbols, there is only one precedence relationship.
- If we have a sentential form, we can calculate the precedence relationships between each pair of consecutive symbols:

`x1 <. x2 =. x3 =. x4 >. x5 <. x6 =. x7 =. x8 >. x9`

- A handle will always start with a <. relationship, will possibly continue with any number of =. relationships, and will always end with a >. relationship.
- We are going to see now how to build the precedence matrix showing the relationships.

Precedence grammars

Example: =. relationship

S ::= aSb

S ::= c

	S	a	b	c
S			1	
a	1			
b				
c				

Precedence grammars

Example: HEAD relationship

S ::= aSb

S ::= c

	S	a	b	c
S		1		1
a				
b				
c				

Precedence grammars

Example: TAIL relationship

S ::= aSb

S ::= c

	S	a	b	c
S			1	1
a				
b				
c				

Precedence grammars

Example: <. relationship

	S	a	b	c
S			1	
a	1			
b				
c				

x

	S	a	b	c
S		1		1
a				
b				
c				

=

	S	a	b	c
S				
a		1		1
b				
c				

•It can be calculated as the (Boolean) matricial product

=. × Head

S ::= aSb

S ::= c

Precedence grammars

Example: identity relationship

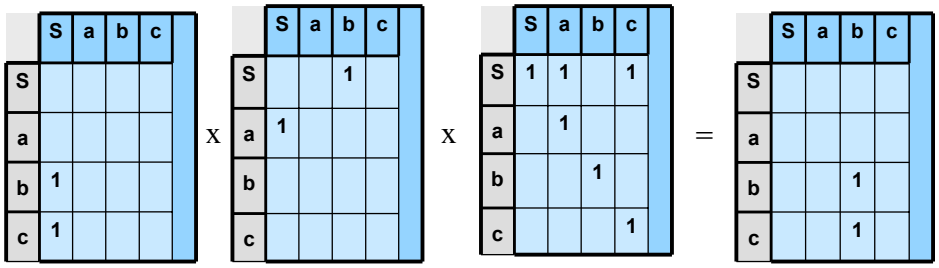
S ::= aSb

S ::= c

	S	a	b	c
S	1			
a		1		
b			1	
c				1

Precedence grammars

Example: >. relationship



•It can be calculated as the (Boolean) matricial product:

(TAIL)^t × =. × (HEAD OR Identity)

S ::= aSb

S ::= c

Precedence grammars

Precedence relationships

S ::= aSb

S ::= c

	S	a	b	c
S			=.	
a	=.	<.		<.
b			>.	
c			>.	

Precedence grammars

Precedence relationships

S ::= aSb

S ::= c

	S	a	b	c
S			=.	
a	=.	<.		<.
b			>.	
c			>.	

If we have the word:

aacbb

the relationships are:

a <. a **<. c >.** b >. b

with the handle marked in red.

Precedence grammars

Precedence relationships

S ::= aSb

S ::= c

	S	a	b	c	END
S			=.		>.
a	=.	<.		<.	>.
b			>.		>.
c			>.		>.
START	<.	<.	<.	<.	

The matrix can be extended with a START and an END symbol

aacbb

The relationships will be:

START <. a <. a **<. c >.** b >. b >. END

with the handle marked in red.

Precedence grammars

Parsing algorithm

Parse

1. Initialise a stack with the START symbol.
2. Compare top(stack) with the next element in the string, X.
   1. If top(stack) is not related to X, notify a syntactic error.
   2. If top(stack) <. X, push(X)
   3. If top(stack) =. X, push(X)
   4. If top(stack) >. X, pop from the stack until we find the <. relationship. The symbols extracted from the stack are the handle. We look for the handle in the right-hand-sides of the grammar rules, and substitute it by the left-hand-side, which will be pushed in the stack.

Precedence grammars

Parsing algorithm

	S	a	b	c	END
S			=.		>.
a	=.	<.		<.	>.
b			>.		>.
c			>.		>.
START	<.	<.	<.	<.	

a	a	c	b	b	END	

STA RT						

Precedence grammars

Parsing algorithm

	S	a	b	c	END
S			=.		>.
a	=.	<.		<.	>.
b			>.		>.
c			>.		>.
START	<.	<.	<.	<.	

a	c	b	b	END	

a,<	STA RT					

Precedence grammars

Parsing algorithm

	S	a	b	c	END
S			=.		>.
a	=.	<.		<.	>.
b			>.		>.
c			>.		>.
START	<.	<.	<.	<.	

c	b	b	END		

a,<	a,<	STA RT					

Precedence grammars

Parsing algorithm

	S	a	b	c	END
S			=.		>.
a	=.	<.		<.	>.
b			>.		>.
c			>.		>.
START	<.	<.	<.	<.	

b	b	END		

c,<	a,<	a,<	STA RT				

Precedence grammars

Parsing algorithm

	S	a	b	c	END
S			=.		>.
a	=.	<.		<.	>.
b			>.		>.
c			>.		>.
START	<.	<.	<.	<.	

S	b	b	END		

a,<	a,<	STA RT					

Precedence grammars

Parsing algorithm

	S	a	b	c	END
S			=.		>.
a	=.	<.		<.	>.
b			>.		>.
c			>.		>.
START	<.	<.	<.	<.	

b	b	END		

S,=	a,<	a,<	STA RT				

Precedence grammars

Parsing algorithm

	S	a	b	c	END
S			=.		>.
a	=.	<.		<.	>.
b			>.		>.
c			>.		>.
START	<.	<.	<.	<.	

b	END	

b, =	S,=	a,<	a,<	STA RT		

Precedence grammars

Parsing algorithm

	S	a	b	c	END
S			=.		>.
a	=.	<.		<.	>.
b			>.		>.
c			>.		>.
START	<.	<.	<.	<.	

S	b	END

a,<	STA RT					

Precedence grammars

Parsing algorithm

	S	a	b	c	END
S			=.		>.
a	=.	<.		<.	>.
b			>.		>.
c			>.		>.
START	<.	<.	<.	<.	

b	END	

S,=	a,<	STA RT				

Precedence grammars

Parsing algorithm

	S	a	b	c	END
S			=.		>.
a	=.	<.		<.	>.
b			>.		>.
c			>.		>.
START	<.	<.	<.	<.	

END		

b,=	S,=	a,<	STA RT			

Precedence grammars

Parsing algorithm

	S	a	b	c	END
S			=.		>.
a	=.	<.		<.	>.
b			>.		>.
c			>.		>.
START	<.	<.	<.	<.	

S	END					

STA RT						

Precedence grammars

Parsing algorithm

	S	a	b	c	END
S			=.		>.
a	=.	<.		<.	>.
b			>.		>.
c			>.		>.
START	<.	<.	<.	<.	

END						

S,<	STA RT					

ACCEPTED

Precedence grammars

Parsing algorithm

	S	a	b	c	END
S			=.		>.
a	=.	<.		<.	>.
b			>.		>.
c			>.		>.
START	<.	<.	<.	<.	

a	a	b	b	END		

STA RT						

Precedence grammars

Parsing algorithm

	S	a	b	c	END
S			=.		>.
a	=.	<.		<.	>.
b			>.		>.
c			>.		>.
START	<.	<.	<.	<.	

a	b	b	END		

a,<	STA RT					

Precedence grammars

Parsing algorithm

	S	a	b	c	END
S			=.		>.
a	=.	<.		<.	>.
b			>.		>.
c			>.		>.
START	<.	<.	<.	<.	

b	b	END		

a,<	a,<	START				

ERROR: their is no precedence between “a” and “b”, so the word is rejected.

Precedence grammars

Simplification with precedence functions

- The precedence matrix has a size of N by N.
- We can represent it with less space using a couple of functions, f and g, called “precedence functions”. They have the following properties:
  - f(x) = g(y) if and only if x =. y
  - f(x) < g(y) if and only if x <. y
  - f(x) > g(y) if and only if x >. y
- Example:

	START	S	a	b	c	END
f	0	2	2	3	3	-
g	-	2	3	2	3	0

	S	a	b	c	END
S			=.		>.
a	=.	<.		<.	>.
b			>.		>.
c			>.		>.
START	<.	<.	<.	<.	

Precedence grammars

Simplification with precedence functions

Advantages:

- The relations of precedence can be stored in a matrix of size 2xN, rather than NxN.

Disadvantages:

- We lose some information, such as the empty cells in the original matrix.
- In some cases, it will be impossible to find such functions. For instance, consider the following matrix:

	a	b
a	=.	>.
b	=.	=.

  - The restrictions are:
    - f(a) = g(a)
    - f(a) > g(b)
    - f(b) = g(a)
    - f(b) = g(b),

    which are impossible to satisfy.

Precedence grammars

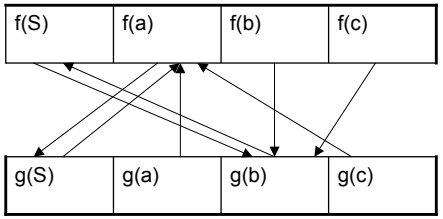
Procedure for calculating the precedence functions

The precedence functions can be calculated with the following procedure:

1. Represent in a diagram:
   - Function f, applied to all the symbols in the grammar, in a row.
   - Function g, applied to every symbol, in a row below.
2. For every pair (x,y),
   - Draw an arc from f(x) to g(y) if x >. y or x =. y
   - Draw an arc from g(y) to f(x) y x <. y or x =. y
3. Finally, for every cell f(x) or g(x):
   - We count how many cells are accessible from it, in zero, one or several steps.
   - That number will be the value of the function for that symbol.

Precedence grammars

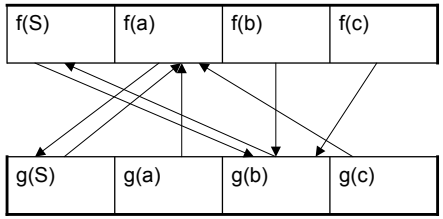
Procedure for calculating the precedence functions



	S	a	b	c	END
S			=.		>.
a	=.	<.		<.	>.
b			>.		>.
c			>.		>.
START	<.	<.	<.	<.	

Precedence grammars

Procedure for calculating the precedence functions



1. We start from f(S). There are two nodes accesible (including itself): f(S) and g(b). Therefore, we set

   f(S) = 2
2. From f(a), we can go to f(a) and g(S). Therefore, we set, again,

   f(a) = 2
3. From f(b), we can go to {f(b), g(b), f(S)} -> f(b) = 3
4. From f(c), we can go to {f(c), g(b), f(S)} -> f(c) = 3
5. From g(S), we can go to {g(S), f(a)} -> g(S) = 2
6. From g(a), we can go to {g(a), f(a), g(S)} -> g(a) = 3
7. From g(b), we can go to {g(b), f(S)} -> g(b) = 2
8. From g(c), we can go to {g(c), f(a), g(S)} -> g(c) = 3

Precedence grammars

Procedure for calculating the precedence functions

If x =. y, then

- f(x) = g(y), because every node accessible from f(x) will be accessible from g(y).

If x >. y, then

- There is an arrow from f(x) to g(y), but there is not an inverse arrow.

  Therefore, we cannot go back from g(y) to f(x).

  This means that from f(x) we'll be able to access every node accessible from g(y), plus f(x), plus every other node accessible from f(x).
  - f(x) will be strictly higher than g(y)

Finally, if x <. y, then,

- There is an arrow from g(y) to f(x), but not the inverse.

  Therefore, the number of nodes accessible from g(y) will be strictly higher than the number accessible from f(x) -> g(y) > f(x).

Precedence grammars

Procedure for calculating the precedence functions

We can automate the procedure in the following way:

1. Build the matrix B:

0	>=
$(<=)^t$	0

2. Calculate the matrix $B^* = B^0$ OR B OR B^2 OR B^3 OR ...
3. f(x) will be the addition of each row in the top half of the matrix.
4. g(x) will be the addition of each row in the lower half of the matrix.

Precedence grammars

Procedure for calculating the precedence functions

	S	a	b	c	END
S			=.		>.
a	=.	<.		<.	>.
b			>.		>.
c			>.		>.
Start	<.	<.	<.	<.	

<=	S	a	b	c
S			1	
a	1	1		1
b				
c				

>=	S	a	b	c
S			1	
a	1			
b			1	
c			1	

Precedence grammars

Procedure for calculating the precedence functions

B	s	a	b	c	S	a	b	c
S							1	
a					1			
b							1	
c							1	
S		1						
a		1						
b	1							
c		1						

Precedence grammars

Procedure for calculating the precedence functions

B*	S	a	b	c	S	a	b	c
S	1						1	
a		1			1			
b	1		1				1	
c	1			1			1	
S		1			1			
a		1			1	1		
b	1						1	
c		1			1			1

Precedence grammars

Procedure for calculating the precedence functions

f(S)	2	Σ
f(a)	2	Σ
f(b)	3	Σ
f(c)	3	Σ
g(S)	2	Σ
g(a)	3	Σ
g(b)	2	Σ
g(c)	3	Σ

B*	S	a	b	c	S	a	b	c
S	1						1	
a		1			1			
b	1		1				1	
c	1			1			1	
S		1			1			
a		1			1	1		
b	1						1	
c		1			1			1

Precedence grammars

Complete example

```
E ::= E+T|T
T ::= F*T|F
F ::= i
```

=	E	T	F	+	*	i
E				1		
T						
F					1	
+		1				
*		1				
i						

Precedence grammars

Example: HEAD relationship

```
E ::= E+T|T
T ::= F*T|F
F ::= i
```

Head	E	T	F	+	*	i
E	1	1	1			1
T			1			1
F						1
+						
*						
i						

Precedence grammars

Example: TAIL relationship

```
E ::= E+T|T
T ::= F*T|F
F ::= i
```

Tail	E	T	F	+	*	i
E		1	1			1
T		1	1			1
F						1
+						
*						
i						

Precedence grammars

Example: <. relationship

=	E	T	F	+	*	i
E				1		
T						
F					1	
+		1				
*		1				
i						

x

H	E	T	F	+	*	i
E	1	1	1			1
T			1			1
F						1
+						
*						
i						

=

<	E	T	F	+	*	i
E						
T						
F						
+			1			1
*			1			1
i						

=. × Head

Precedence grammars

Example: identity relationship

```
E ::= T+E|T
T ::= F*T|F
F ::= i
```

Id	E	T	F	+	*	i
E	1					
T		1				
F			1			
+				1		
*					1	
i						1

Precedence grammars

Example: >. relationship

T^t	E	T	F	+	*	i
E						
T	1	1				
F	1	1				
+						
*						
i	1	1	1			

x

=	E	T	F	+	*	i
E				1		
T						
F					1	
+		1				
*		1				
i						

x

H	E	T	F	+	*	i
E	1	1	1			1
T		1	1			1
F			1			1
+				1		
*					1	
i						1

=

>	E	T	F	+	*	i
E						
T				1		
F				1		
+						
*						
i				1	1	

$(TAIL)^t \times =. \times (HEAD\ OR\ Identity)$

Precedence grammars

Precedence relationships

```
S ::= aSb
S ::= c
```

M	E	T	F	+	*	i
E				=.		
T				>.		
F				>.	=.	
+		=.	<.			<.
*		=.	<.			<.
i				>.	>.	

Precedence grammars

Parsing algorithm

M	E	T	F	+	*	i	END
E				=.			>.
T				>.			>.
F				>.	=.		>.
+		=.	<.			<.	>.
*		=.	<.			<.	>.
i				>.	>.		>.
St	<.	<.	<.	<.	<.	<.	

i	*	i	+	i	END	

STA RT						

Precedence grammars

Parsing algorithm

M	E	T	F	+	*	i	END
E				=.			>.
T				>.			>.
F				>.	=.		>.
+		=.	<.			<.	>.
*		=.	<.			<.	>.
i				>.	>.		>.
St	<.	<.	<.	<.	<.	<.	

*	i	+	i	END	

i,<	START					

Precedence grammars

Parsing algorithm

M	E	T	F	+	*	i	END
E				=.			>.
T				>.			>.
F				>.	=.		>.
+		=.	<.			<.	>.
*		=.	<.			<.	>.
i				>.	>.		>.
St	<.	<.	<.	<.	<.	<.	

F	*	i	+	i	END	

START						

Precedence grammars

Parsing algorithm

M	E	T	F	+	*	i	END
E				=.			>.
T				>.			>.
F				>.	=.		>.
+		=.	<.			<.	>.
*		=.	<.			<.	>.
i				>.	>.		>.
St	<.	<.	<.	<.	<.	<.	

*	i	+	i	END	

F,<	START					

Precedence grammars

Parsing algorithm

M	E	T	F	+	*	i	END
E				=.			>.
T				>.			>.
F				>.	=.		>.
+		=.	<.			<.	>.
*		=.	<.			<.	>.
i				>.	>.		>.
St	<.	<.	<.	<.	<.	<.	

i	+	i	END	

*,=	F,<	START				

Precedence grammars

Parsing algorithm

M	E	T	F	+	*	i	END
E				=.			>.
T				>.			>.
F				>.	=.		>.
+		=.	<.			<.	>.
*		=.	<.			<.	>.
i				>.	>.		>.
St	<.	<.	<.	<.	<.	<.	

+	i	END	

i,<	*,=	F,<	STA RT			

Precedence grammars

Parsing algorithm

M	E	T	F	+	*	i	END
E				=.			>.
T				>.			>.
F				>.	=.		>.
+		=.	<.			<.	>.
*		=.	<.			<.	>.
i				>.	>.		>.
St	<.	<.	<.	<.	<.	<.	

F	+	i	END

*,=	F,<	STA RT			

Precedence grammars

Parsing algorithm

M	E	T	F	+	*	i	END
E				=.			>.
T				>.			>.
F				>.	=.		>.
+		=.	<.			<.	>.
*		=.	<.			<.	>.
i				>.	>.		>.
St	<.	<.	<.	<.	<.	<.	

+	i	END	

F,<	*,=	F,<	STA RT			

Precedence grammars

Parsing algorithm

M	E	T	F	+	*	i	END
E				=.			>.
T				>.			>.
F				>.	=.		>.
+		=.	<.			<.	>.
*		=.	<.			<.	>.
i				>.	>.		>.
St	<.	<.	<.	<.	<.	<.	

T	+	i	END

*,=	F,<	STA RT			

Precedence grammars

Parsing algorithm

M	E	T	F	+	*	i	END
E				=.			>.
T				>.			>.
F				>.	=.		>.
+		=.	<.			<.	>.
*		=.	<.			<.	>.
i				>.	>.		>.
St	<.	<.	<.	<.	<.	<.	

+	i	END	

T,=	*,=	F,<	STA RT			

Precedence grammars

Parsing algorithm

M	E	T	F	+	*	i	END
E				=.			>.
T				>.			>.
F				>.	=.		>.
+		=.	<.			<.	>.
*		=.	<.			<.	>.
i				>.	>.		>.
St	<.	<.	<.	<.	<.	<.	

T	+	I	END

STA RT						

Precedence grammars

Parsing algorithm

M	E	T	F	+	*	i	END
E				=.			>.
T				>.			>.
F				>.	=.		>.
+		=.	<.			<.	>.
*		=.	<.			<.	>.
i				>.	>.		>.
St	<.	<.	<.	<.	<.	<.	

+	i	END	

T,<	STA RT					

Precedence grammars

Parsing algorithm

M	E	T	F	+	*	i	END
E				=.			>.
T				>.			>.
F				>.	=.		>.
+		=.	<.			<.	>.
*		=.	<.			<.	>.
i				>.	>.		>.
St	<.	<.	<.	<.	<.	<.	

E	+	I	END

STA RT						

Precedence grammars

Parsing algorithm

M	E	T	F	+	*	i	END
E				=.			>.
T				>.			>.
F				>.	=.		>.
+		=.	<.			<.	>.
*		=.	<.			<.	>.
i				>.	>.		>.
St	<.	<.	<.	<.	<.	<.	

+	i	END	

E,<	STA RT					

Precedence grammars

Parsing algorithm

M	E	T	F	+	*	i	END
E				=.			>.
T				>.			>.
F				>.	=.		>.
+		=.	<.			<.	>.
*		=.	<.			<.	>.
i				>.	>.		>.
St	<.	<.	<.	<.	<.	<.	

i	END	

+,=	E,<	STA RT				

Precedence grammars

Parsing algorithm

M	E	T	F	+	*	i	END
E				=.			>.
T				>.			>.
F				>.	=.		>.
+		=.	<.			<.	>.
*		=.	<.			<.	>.
i				>.	>.		>.
St	<.	<.	<.	<.	<.	<.	

END	

i,<	+,=	E,<	STA RT			

Precedence grammars

Parsing algorithm

M	E	T	F	+	*	i	END
E				=.			>.
T				>.			>.
F				>.	=.		>.
+		=.	<.			<.	>.
*		=.	<.			<.	>.
i				>.	>.		>.
St	<.	<.	<.	<.	<.	<.	

F	END

+,=	E,<	STA RT			

Precedence grammars

Parsing algorithm

M	E	T	F	+	*	i	END
E				=.			>.
T				>.			>.
F				>.	=.		>.
+		=.	<.			<.	>.
*		=.	<.			<.	>.
i				>.	>.		>.
St	<.	<.	<.	<.	<.	<.	

END	

F,<	+,=	E,<	STA RT			

Precedence grammars

Parsing algorithm

M	E	T	F	+	*	i	END
E				=.			>.
T				>.			>.
F				>.	=.		>.
+		=.	<.			<.	>.
*		=.	<.			<.	>.
i				>.	>.		>.
St	<.	<.	<.	<.	<.	<.	

T	END

+,=	E,<	STA RT			

Precedence grammars

Parsing algorithm

M	E	T	F	+	*	i	END
E				=.			>.
T				>.			>.
F				>.	=.		>.
+		=.	<.			<.	>.
*		=.	<.			<.	>.
i				>.	>.		>.
St	<.	<.	<.	<.	<.	<.	

END	

T,=	+,=	E,<	STA RT			

Precedence grammars

Parsing algorithm

M	E	T	F	+	*	i	END
E				=.			>.
T				>.			>.
F				>.	=.		>.
+		=.	<.			<.	>.
*		=.	<.			<.	>.
i				>.	>.		>.
St	<.	<.	<.	<.	<.	<.	

E	END

STA RT						

Precedence grammars

Parsing algorithm

M	E	T	F	+	*	i	END
E				=.			>.
T				>.			>.
F				>.	=.		>.
+		=.	<.			<.	>.
*		=.	<.			<.	>.
i				>.	>.		>.
St	<.	<.	<.	<.	<.	<.	

END	

E	START					

ACCEPTED

Precedence grammars

Complete example

Note that the following versions of the same grammar are not simple-precedence grammars:

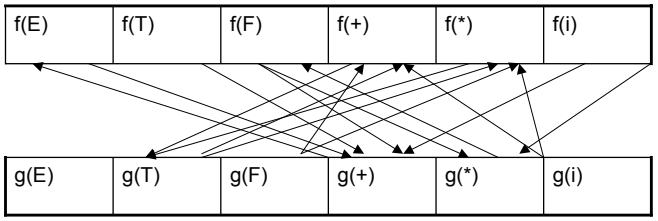
```
E ::= E+T | T
T ::= T * F | F
F ::= i

E ::= T + E | T
T ::= F * T | F
F ::= i
```

Why? (exercise)

Precedence grammars

Procedure for calculating the precedence functions



M	E	T	F	+	*	i
E				=.		
T				>.		
F				>.	=.	
+		=.	<.			<.
*		=.	<.			<.
i				>.	>.	

Precedence grammars

Complete example

Function	Accessible cells	Function value
f(E)	f(E), g(+)	2
f(T)	f(T), g(+), f(E)	3
f(F)	f(F), g(+), f(E), g(*)	4
f(+)	f(+), g(T), f(*)	3
f(*)	f(*), g(T), f(+)	3
f(i)	f(i), g(+), g(*), f(E), f(F)	5
g(E)	g(E)	1
g(T)	g(T), f(+), f(*)	3
g(F)	g(F), f(+), f(*), g(T)	4
g(+)	g(+), f(E)	2
g(*)	g(*), f(F), g(+), f(E)	4
g(i)	g(i), f(+), f(*), g(T)	4

Precedence grammars

Parsing algorithm

Function	Function value
f(E)	2
f(T)	3
f(F)	4
f(+)	3
f(*)	3
f(i)	5
g(E)	1
g(T)	3
g(F)	4
g(+)	2
g(*)	4
g(i)	4
f(START)=g(END)	0

i	*	i	+	i	END	

STA RT						

EXERCISE:

Repeat the analysis of the string using the precedence functions