

Computer Language Processing

Exercise Sheet 04 - Solutions

October 20, 2022

N.B.: There are many equivalent ways to do step 4) of the conversion to Chomsky normal form (conversion to binary productions) - our solution shows just one way. For example, $\mathbf{P\ Q}$ $\mathbf{R\ S}$ can be represented as either $\mathbf{X\ Y}$, where

$$X ::= P\ Q$$
$$Y ::= R\ S$$

or it can be represented as $\mathbf{P\ U}$, where

$$U ::= Q\ V$$
$$V ::= R\ S$$

Exercise 1

After step 1) (A is unproductive):

$$S ::= P$$
$$P ::= B \mid \text{if } B \text{ then } P \text{ else } P \mid P ; P \mid \varepsilon$$
$$U ::= + \mid -$$
$$B ::= \text{true} \mid \text{false} \mid B \ \&\& \ B$$

After step 2) (U is unreachable):

$$S ::= P$$
$$P ::= B \mid \text{if } B \text{ then } P \text{ else } P \mid P ; P \mid \varepsilon$$
$$B ::= \text{true} \mid \text{false} \mid B \ \&\& \ B$$

After step 3):

$$S ::= P$$
$$P ::= B \mid T_{if} \ B \ T_{then} \ P \ T_{else} \ P \mid P \ T; \ P \mid \varepsilon$$
$$B ::= \text{true} \mid \text{false} \mid B \ T_{\&\&} \ B$$
$$T_{if} ::= \text{if}$$
$$T_{then} ::= \text{then}$$

$T_{else} ::= \text{else}$

$T_{;} ::= ;$

$T_{\&\&} ::= \&\&$

After step 4):

$S ::= P$

$P ::= B \mid I1 \ I2 \mid P \ I3 \mid \varepsilon$

$I1 ::= T_{if} \ I4$

$I2 ::= P \ I5$

$I3 ::= T_{;} \ P$

$I4 ::= B \ T_{then}$

$I5 ::= T_{else} \ P$

$B ::= \text{true} \mid \text{false} \mid B \ I6$

$I6 ::= T_{\&\&} \ B$

$T_{if} ::= \text{if}$

$T_{then} ::= \text{then}$

$T_{else} ::= \text{else}$

$T_{;} ::= ;$

$T_{\&\&} ::= \&\&$

After step 5):

$S ::= P \mid \varepsilon$

$P ::= B \mid I1 \ I2 \mid P \ I3 \mid I3$

$I1 ::= T_{if} \ I4$

$I2 ::= P \ I5 \mid I5$

$I3 ::= T_{;} \ P \mid T_{;} \ P$

$I4 ::= B \ T_{then}$

$I5 ::= T_{else} \ P \mid T_{else} \ P$

$B ::= \text{true} \mid \text{false} \mid B \ I6$

$I6 ::= T_{\&\&} \ B$

$T_{if} ::= \text{if}$

$T_{then} ::= \text{then}$

$T_{else} ::= \text{else}$

$T_{;} ::= ;$

$T_{\&\&} ::= \&\&$

After step 6):

$S ::= \text{true} \mid \text{false} \mid B \ I6 \mid I1 \ I2 \mid P \ I3 \mid T; P \mid ; \mid \varepsilon$

$P ::= \text{true} \mid \text{false} \mid B \ I6 \mid I1 \ I2 \mid P \ I3 \mid T; P \mid ;$

$I1 ::= T_{if} \ I4$

$I2 ::= P \ I5 \mid T_{else} \ P \mid \text{else}$

$I3 ::= T; P \mid ;$

$I4 ::= B \ T_{then}$

$I5 ::= T_{else} \ P \mid \text{else}$

$B ::= \text{true} \mid \text{false} \mid B \ I6$

$I6 ::= T_{\&\&} \ B$

$T_{if} ::= \text{if}$

$T_{then} ::= \text{then}$

$T_{else} ::= \text{else}$

$T; ::= ;$

$T_{\&\&} ::= \&\&$

After step 7):

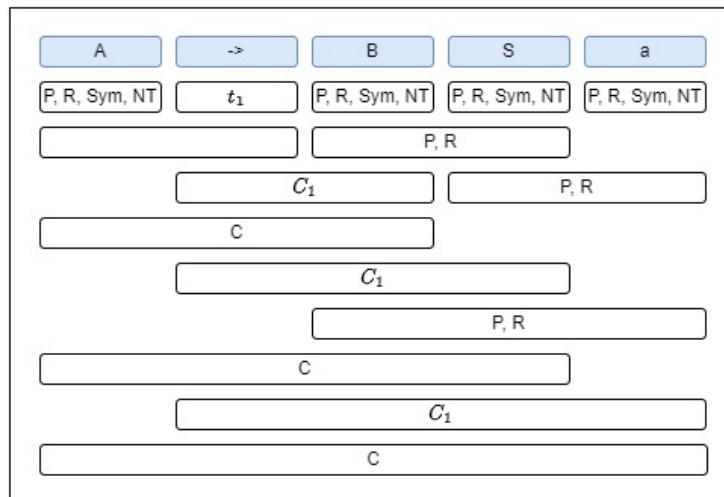
Unchanged.

After step 8):

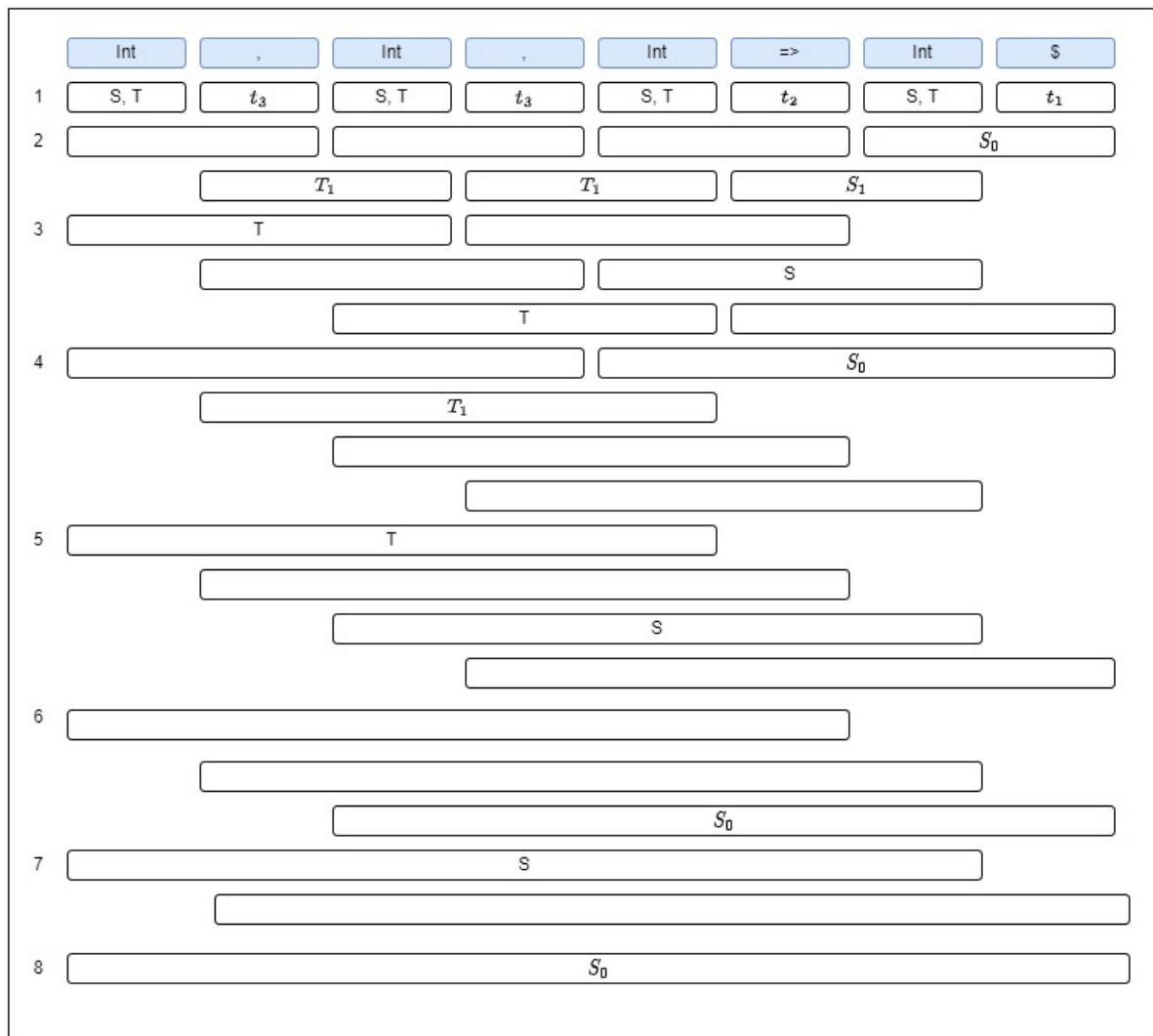
Unchanged.

Exercise 2

a) Self-describing grammar



b) Function types grammar



Exercise 3

a) Conversion to Chomsky normal form

After step 1) (C is unproductive):

$S ::= P ;$

$P ::= I \mid I ; P$

$I ::= \text{if } E \text{ then } P \text{ R} \mid \text{print } E$

$R ::= \text{else } P \mid \varepsilon$

$W ::= \text{while } E \text{ do } P$

$E ::= L \mid E \text{ or } E$

$L ::= \text{true} \mid \text{false}$

After step 2) (W is unreachable):

$S ::= P ;$
 $P ::= I \mid I ; P$
 $I ::= \text{if } E \text{ then } P \ R \mid \text{print } E$
 $R ::= \text{else } P \mid \varepsilon$
 $E ::= L \mid E \text{ or } E$
 $L ::= \text{true} \mid \text{false}$

After step 3):

$S ::= P \ T_;$
 $P ::= I \mid I \ T_ ; P$
 $I ::= T_{if} \ E \ T_{then} \ P \ R \mid T_{print} \ E$
 $R ::= T_{else} \ P \mid \varepsilon$
 $E ::= L \mid E \ T_{or} \ E$
 $L ::= \text{true} \mid \text{false}$
 $T_ ::= ;$
 $T_{if} ::= \text{if}$
 $T_{then} ::= \text{then}$
 $T_{print} ::= \text{print}$
 $T_{else} ::= \text{else}$
 $T_{or} ::= \text{or}$

After step 4):

$S ::= P \ T_;$
 $P ::= I \mid I \ P1$
 $P1 ::= T_ ; P$
 $I ::= T_{if} \ I1 \mid T_{print} \ E$
 $I1 ::= E \ I2$
 $I2 ::= T_{then} \ I3$
 $I3 ::= P \ R$
 $R ::= T_{else} \ P \mid \varepsilon$
 $E ::= L \mid E \ E1$
 $E1 ::= T_{or} \ E$
 $L ::= \text{true} \mid \text{false}$

$T_;$::= ;

T_{if} ::= if

T_{then} ::= then

T_{print} ::= print

T_{else} ::= else

T_{or} ::= or

After step 5):

S ::= P $T_;$

P ::= I | I P1

$P1$::= $T_;$ P

I ::= T_{if} I1 | T_{print} E

$I1$::= E I2

$I2$::= T_{then} I3

$I3$::= P R | P

R ::= T_{else} P

E ::= L | E E1

$E1$::= T_{or} E

L ::= true | false

$T_;$::= ;

T_{if} ::= if

T_{then} ::= then

T_{print} ::= print

T_{else} ::= else

T_{or} ::= or

After step 6):

S ::= P $T_;$

P ::= T_{if} I1 | T_{print} E | I P1

$P1$::= $T_;$ P

I ::= T_{if} I1 | T_{print} E

$I1$::= E I2

$I2$::= T_{then} I3

$I3$::= P R | T_{if} I1 | T_{print} E | I P1

$R ::= T_{else} P$
 $E ::= \text{true} \mid \text{false} \mid E E1$
 $E1 ::= T_{or} E$
 $L ::= \text{true} \mid \text{false}$
 $T_{,} ::= ;$
 $T_{if} ::= \text{if}$
 $T_{then} ::= \text{then}$
 $T_{print} ::= \text{print}$
 $T_{else} ::= \text{else}$
 $T_{or} ::= \text{or}$

After step 7):

Unchanged.

After step 8) (L unreachable):

$S ::= P T_{,}$
 $P ::= T_{if} I1 \mid T_{print} E \mid I P1$
 $P1 ::= T_{,} P$
 $I ::= T_{if} I1 \mid T_{print} E$
 $I1 ::= E I2$
 $I2 ::= T_{then} I3$
 $I3 ::= P R \mid T_{if} I1 \mid T_{print} E \mid I P1$
 $R ::= T_{else} P$
 $E ::= \text{true} \mid \text{false} \mid E E1$
 $E1 ::= T_{or} E$
 $T_{,} ::= ;$
 $T_{if} ::= \text{if}$
 $T_{then} ::= \text{then}$
 $T_{print} ::= \text{print}$
 $T_{else} ::= \text{else}$
 $T_{or} ::= \text{or}$

b) Parsing and counting the number of parse trees

We will use the notation $:n$ to denote how many times we can produce a non-terminal (that is, the number of parse trees at that point).

if	true	then	print	true	;	print	false	;
$T_{if}:1$	$E:1$	$T_{then}:1$	$T_{print}:1$	$E:1$	$T_{;}:1$	$T_{print}:1$	$E:1$	$T_{;}:1$
			$I:1$ $I_3:1$ $P:1$			$I:1$ $I_3:1$ $P:1$		
		$I_2:1$	$S:1$		$P_1:1$	$S:1$		
	$I_1:1$							
$I:1$ $I_3:1$ $P:1$			$P:1$ $I_3:1$					
$S:1$		$I_2:1$	$S:1$					
	$I_1:1$							
$P:2$ $I:1$ $I_3:2$								
$S:2$								

There are 2 parse trees. Intuitively, one parse tree represents the program where **print false** is meant to be part of the **then** branch of the **if**, while the second parse tree represents the program where **print false** is meant to be completely after the **if** statement.

c) Counting the number of parse trees

if true or false or true then print true or false or true ;

There are 2 different valid ways to interpret true or false or true. Therefore, in total 4 different ways to interpret that string.

if true then if false then print true else print false ;

There are 2 valid ways to parse, one which considers the else to be part of the first if, the other one which considers the else to be part of the second, innermost, if.

if true then print true ; print false ; print true ;

There are 3 valid ways to parse:

- 1) print true only is guarded by the condition,
- 2) print true ; print false is guarded by the condition,
- 3) The entirety of print true ; print false ; print true is guarded.