Chomsky Normal Form & CYK Algorithm

Summary of the conversion procedure to Chomsky normal form (CNF):

1) Remove unproductive symbols. (optional)
2) Remove unreachable symbols. (optional)
3) Make terminals occur alone on right-hand side.
4) Reduce arity of every production to 2 or less.
5) Remove epsilons.
6) Remove unit productions.
7) Remove unproductive symbols.
8) Remove unreachable symbols.

Exercise 1

Convert the following grammar to Chomsky normal form.

```
S ::= P
P ::= A | B | if B then P else P | P ; P | ε
A ::= U A | A + A | A * A
U ::= + | -
B ::= true | false | B && B
```

Solution:

After step 1) (A is unproductive)

```
S ::= P
P ::= B | if B then P else P | P ; P | ε
U ::= + | -
B ::= true | false | B && B
```

After step 2) (U is unreachable)

```
S ::= P
P ::= B | if B then P else P | P ; P | ε
B ::= true | false | B && B
```

After step 3)

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

After step 4)

$S ::= P$
$P ::= B \mid I_1\ I_2 \mid P\ I_3 \mid \varepsilon$
$I_1 ::= T_{if}\ I_4$
$I_2 ::= P\ I_5$
$I_3 ::= T_{;}\ P$
$I_4 ::= B\ T_{then}$
$I_5 ::= T_{else}\ P$
$B ::= \texttt{true} \mid \texttt{false} \mid B\ I_6$
$I_6 ::= T_{\&\&}\ B$
$T_{if} ::= \texttt{if}$
$T_{then} ::= \texttt{then}$
$T_{else} ::= \texttt{else}$
$T_{;} ::= \texttt{;}$
$T_{\&\&} ::= \texttt{\&\&}$

After step 5)

$S ::= P \mid \varepsilon$
$P ::= B \mid I_1\ I_2 \mid P\ I_3 \mid I_3$
$I_1 ::= T_{if}\ I_4$
$I_2 ::= P\ I_5 \mid I_5$
$I_3 ::= T_{;}\ P \mid T_{;}$
$I_4 ::= B\ T_{then}$
$I_5 ::= T_{else}\ P \mid T_{else}$
$B ::= \texttt{true} \mid \texttt{false} \mid B\ I_6$
$I_6 ::= T_{\&\&}\ B$
$T_{if} ::= \texttt{if}$
$T_{then} ::= \texttt{then}$
$T_{else} ::= \texttt{else}$
$T_{;} ::= \texttt{;}$
$T_{\&\&} ::= \texttt{\&\&}$

After step 6)

$S ::= true \mid false \mid B\ I_6 \mid I_1\ I_2 \mid P\ I_3 \mid T_;\ P \mid ; \mid \varepsilon$

$P ::= true \mid false \mid B\ I_6 \mid I_1\ I_2 \mid P\ I_3 \mid T_;\ P \mid ;$

$I_1 ::= T_{if}\ I_4$

$I_2 ::= P\ I_5 \mid T_{else}\ P \mid else$

$I_3 ::= T_;\ P \mid ;$

$I_4 ::= B\ T_{then}$

$I_5 ::= T_{else}\ P \mid else$

$B ::= true \mid false \mid B\ I_6$

$I_6 ::= T_{\&\&}\ B$

$T_{if} ::= if$

$T_{then} ::= then$

$T_{else} ::= else$

$T_; ::= ;$

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

After step 7)

Unchanged.

After step 8)

Unchanged.

Exercise 2

Convert the following grammar to Chomsky normal form.

$S ::= a\ S\ b \mid \varepsilon$

Solution:

First, we introduce a new start symbol, since we do not want it to appear on the right-hand side of any rule.

$S_0 ::= S$
$S ::= a\ S\ b \mid \varepsilon$

After steps 1) and 2), the grammar is unchanged.
After step 3)

$S_0 ::= S$
$S ::= T_a\ S\ T_b \mid \varepsilon$
$T_a ::= a$
$T_b ::= b$

After step 4)

$S_0 ::= S$
$S ::= T_a\ I \mid \varepsilon$
$I ::= S\ T_b$
$T_a ::= a$
$T_b ::= b$

After step 5)

$S_0 ::= S \mid \varepsilon$
$S ::= T_a\ I$
$I ::= S\ T_b \mid T_b$
$T_a ::= a$
$T_b ::= b$

After step 6)

$S_0 ::= T_a\ I \mid \varepsilon$
$S ::= T_a\ I$
$I ::= S\ T_b \mid b$
$T_a ::= a$
$T_b ::= b$

After step 7) and 8), the grammar is unchanged.

Apply the CYK algorithm to parse the following inputs.

a a a b b b

Solution:

a	a	a	b	b	b
T_a	T_a	T_a	T_b, I	T_b, I	T_b, I
		S, S_0			
		I			
	S, S_0				
	I				
S, S_0					

a b a b

Solution:

a	b	a	b
T_a	T_b, I	T_a	T_b, I
S, S_0		S, S_0	

Exercise 3

Convert the following grammar to Chomsky normal form.

```
S ::= P ;
P ::= I | I ; P
I ::= if E then P R | print E
R ::= else P | ε
W ::= while E do P
E ::= L | E or E | C
C ::= C and E | E and C
L ::= true | false
```

Solution:

After step 1) (C is unproductive)

```
S ::= P ;
P ::= I | I ; P
I ::= if E then P R | print E
R ::= else P | ε
W ::= while E do P
E ::= L | E or E
L ::= true | false
```

After step 2) (W is unreachable)

```
S ::= P ;
P ::= I | I ; P
I ::= if E then P R | print E
R ::= else P | ε
E ::= L | E or E
L ::= true | false
```

After step 3)

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

After step 4)

$S ::= P\ T_{;}$
$P ::= I \mid I\ P_1$
$P_1 ::= T_{;}\ P$
$I ::= T_{if}\ I_1 \mid T_{print}\ E$
$I_1 ::= E\ I_2$
$I_2 ::= T_{then}\ I_3$
$I_3 ::= P\ R$
$R ::= T_{else}\ P \mid \varepsilon$
$E ::= L \mid E\ E_1$
$E_1 ::= T_{or}\ E$
$L ::= \texttt{true} \mid \texttt{false}$
$T_{;} ::= \texttt{;}$
$T_{if} ::= \texttt{if}$
$T_{then} ::= \texttt{then}$
$T_{print} ::= \texttt{print}$
$T_{else} ::= \texttt{else}$
$T_{or} ::= \texttt{or}$

After step 5)

$S ::= P\ T_{;}$
$P ::= I \mid I\ P_1$
$P_1 ::= T_{;}\ P$
$I ::= T_{if}\ I_1 \mid T_{print}\ E$
$I_1 ::= E\ I_2$
$I_2 ::= T_{then}\ I_3$
$I_3 ::= P\ R \mid P$
$R ::= T_{else}\ P$
$E ::= L \mid E\ E_1$
$E_1 ::= T_{or}\ E$
$L ::= \text{true} \mid \text{false}$
$T_{;} ::= ;$
$T_{if} ::= \text{if}$
$T_{then} ::= \text{then}$
$T_{print} ::= \text{print}$
$T_{else} ::= \text{else}$
$T_{or} ::= \text{or}$

After step 6)

$S ::= P\ T_{;}$
$P ::= T_{if}\ I_1 \mid T_{print}\ E \mid I\ P_1$
$P_1 ::= T_{;}\ P$
$I ::= T_{if}\ I_1 \mid T_{print}\ E$
$I_1 ::= E\ I_2$
$I_2 ::= T_{then}\ I_3$
$I_3 ::= P\ R \mid T_{if}\ I_1 \mid T_{print}\ E \mid I\ P_1$
$R ::= T_{else}\ P$
$E ::= \text{true} \mid \text{false} \mid E\ E_1$
$E_1 ::= T_{or}\ E$
$L ::= \text{true} \mid \text{false}$
$T_{;} ::= ;$
$T_{if} ::= \text{if}$
$T_{then} ::= \text{then}$
$T_{print} ::= \text{print}$
$T_{else} ::= \text{else}$
$T_{or} ::= \text{or}$

After step 7)

Unchanged.

After step 8) (L unreachable)

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

$P ::= T_{if}\ I_1 \mid T_{print}\ E \mid I\ P_1$

$P_1 ::= T_{;}\ P$

$I ::= T_{if}\ I_1 \mid T_{print}\ E$

$I_1 ::= E\ I_2$

$I_2 ::= T_{then}\ I_3$

$I_3 ::= P\ R \mid T_{if}\ I_1 \mid T_{print}\ E \mid I\ P_1$

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

$E ::= \texttt{true} \mid \texttt{false} \mid E\ E_1$

$E_1 ::= T_{or}\ E$

$T_{;} ::= ;$

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

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

$T_{print} ::= \texttt{print}$

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

$T_{or} ::= \texttt{or}$

Apply the CYK algorithm to parse the following input. Count the number of possible parse trees.

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

Solution:

We will use the notation `:n` to denote how many times we can produce a non-terminal (that is, the number of parse tree 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.

Without actually running the CYK algorithm, count the number of parse trees of the following inputs.

Solution:

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

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 ;
```

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 ;
```

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 entirely of `print true ; print false ; print true` is guarded.