Computer Language Processing

Exercise Sheet 04

October 18, 2022

Welcome to the fourth exercise session of CS320! We have selected exercises from the 2019 edition of this class (ex. 1, 3), as well as exercise 2 from grammar.epfl.ch - which we warmly recommend for additional problems on CYK parsing.

Exercise 1 | Chomsky Normal Form

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

Here is a summary of the conversion procedure to Chomsky normal form (CNF), which you may find useful:

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 2 | CYK parsing

For this exercise, you do not need to convert the given grammars to Chomsky normal form.

a) Write the CYK parse table for the word **A -> B S a** of the following grammar[1]:

C ::= NT C1

R1 ::= t2 R

P ::= Sym P | S | A | B | a | b

R ::= P R1 | Sym P | S | A | B | a | b

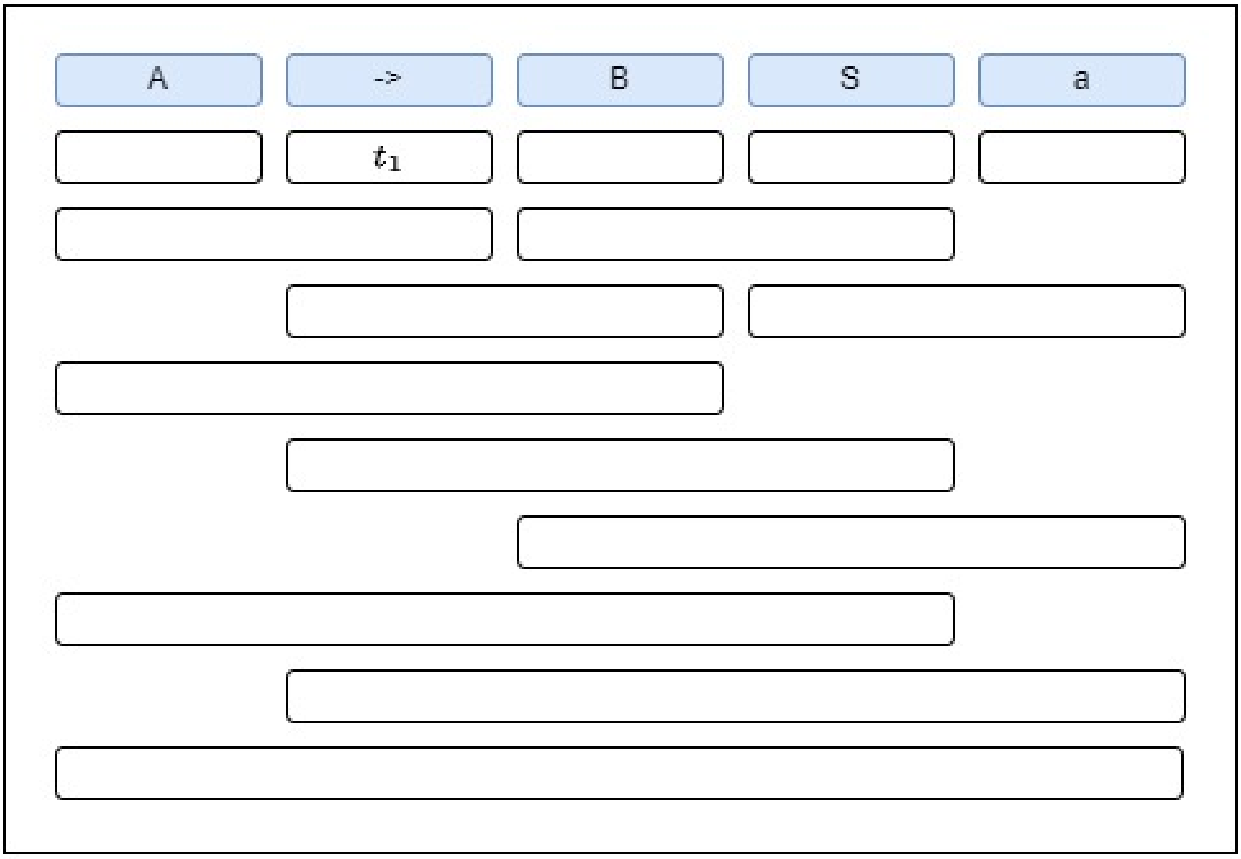
t1 ::= ->

Sym ::= S | A | B | a | b

NT ::= S | A | B

C1 ::= t1 R

t2 ::= '|'



[1] "Self-describing" on grammar.epfl.ch

b) - optional, for extra practice

Write the CYK parse table for the word **Int , Int , Int** => **Int $** of following grammar[2]:

S0 ::= S t1

T1 ::= t3 T

S ::= T S1 | Int

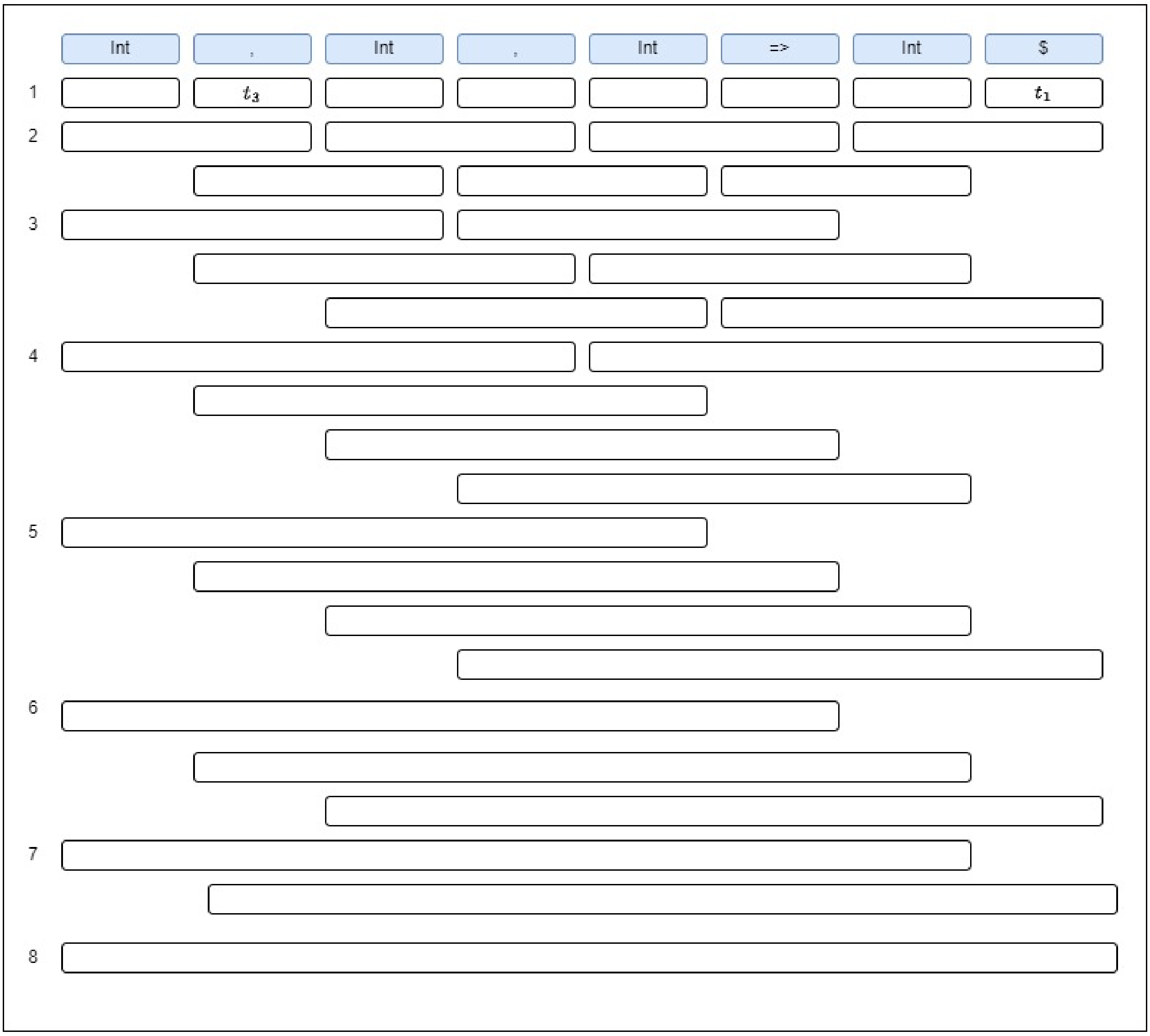
T ::= S T1 | Int

t3 ::= ,

t1 ::= $

S1 ::= t2 S

t2 ::= =>



	Int	,	Int	,	Int	=>	Int	$
1		t_3						t_1
2								
3								
4								
5								
6								
7								
8								

[2] "Function Types" on grammar.epfl.ch

Exercise 3 | CYK parsing; CNF

a) 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

b) Now, apply the CYK algorithm to the following input. Count the number of possible parse trees.

- **if true then print true ; print false ;**

c) Without actually running the CYK algorithm, count the number of parse trees of the following inputs:

- **if true or false or true then print true or false or true ;**
- **if true the if false then print true else print false ;**
- **if true then print true ; print false; print true ;**