COMP 442 / 6421 Compiler Design

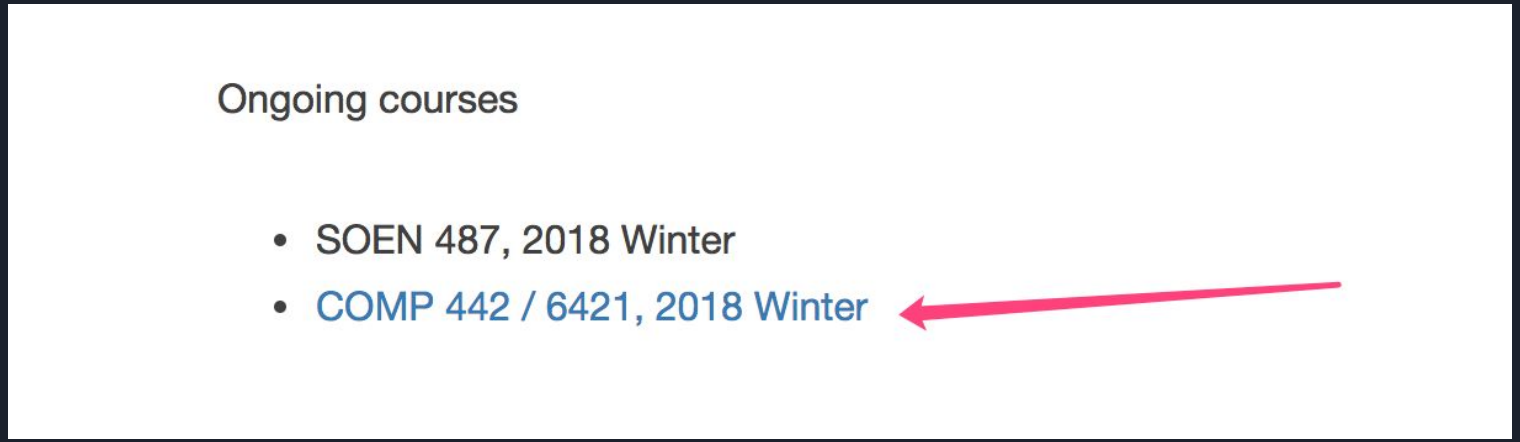
Tutorial 2

Instructor:	Dr. Joey Paquet	paquet@cse.concordia.ca
TAs:	Haotao Lai	h_lai@encs.concordia.ca
	Jashanjot Singh	s_jashan@cs.concordia.ca

Tutorial Slides

You can access the tutorial slide set through the following link:

http://laihaotao.me/ta/



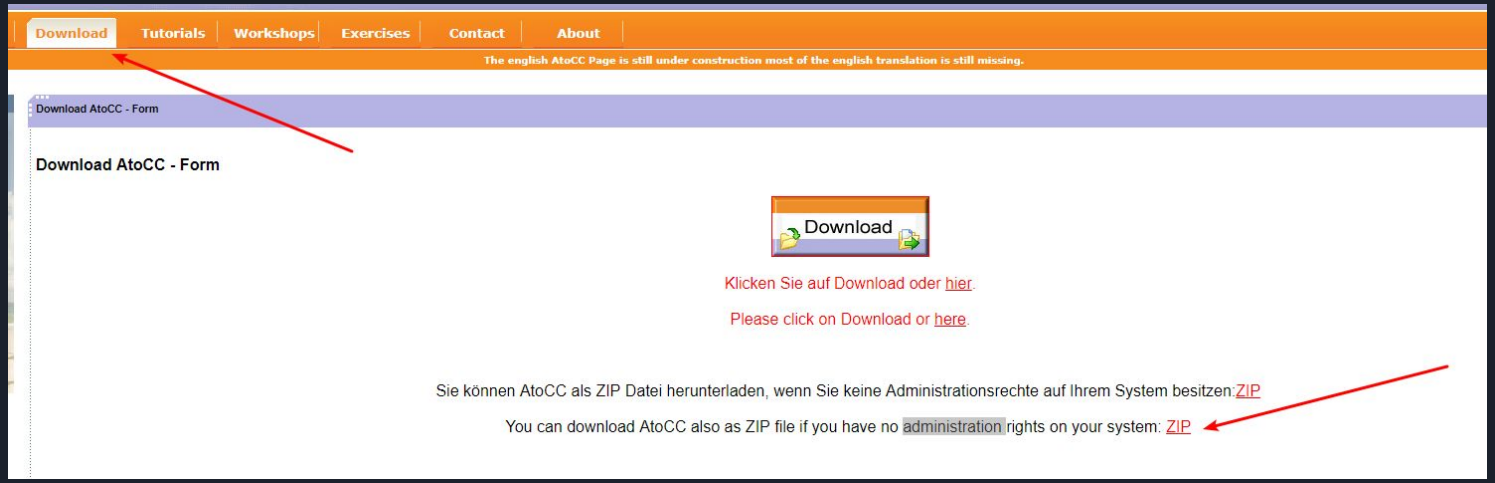
You don’t have Window machine?

Check the following link out:

http://atocc.de/AtoCCFAQ/index.php?option=com_content&task=category&sectionid=11&id=25&Itemid=34

I tried it on macOS High Sierra version 10.13.1 and it worked!

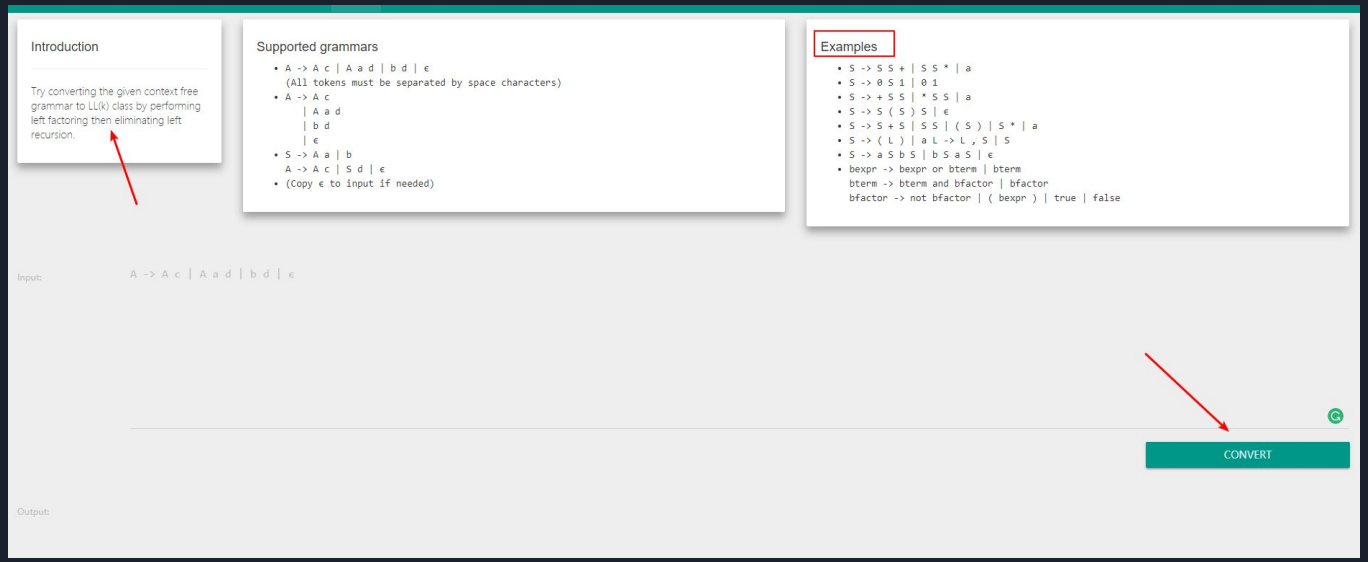
How to access AtoCC without administration right ?



There is a portable version, but sometime it will crash, save your work frequently!

Another powerful tool

https://cyberzhg.github.io/toolbox/cfg2ll



Some examples of the grammar

Course Material and Resources

- Course outline
- ENCS Electronic Student Submission System
- AtoCC grammar files
- Moon processor simulator: code, documentation, libraries, examples

The Goal of Assignment 2

1. Convert the given CFG to LL(1) grammar
    a. Need to use tools to verify your converting procedure
    b. Remove the grammar from EBNF to non-EBNF presentation
    c. Remove ambiguity and left recursion
2. Implement a LL(1) parser
    a. Recursive descent predictive parsing
    b. Table-driven predictive parsing

Example (1)

<u>Assume you was given a grammar as following:</u>

commaSeparatedList -> a {,a} | EPSILON

<u>You should remove the EBNF format and come up with the following grammar:</u>

```
commaSeparatedList     -> a commaSeparatedListTail
                        | EPSILON
commaSeparatedListTail -> ,a commaSeparatedListTail
                        | EPSILON
```

Example (2)

After remove EBNF format, assume you have something like:

```
expr     ->        expr + term | term
term     ->        term * factor | factor
factor   ->        '(' expr ')' |    x
```

You will need to perform the operation: remove left recursion.

How to come up with the proper grammar?

- You receive the initial grammar in EBNF in assignment 2 description already;
- You need to remove the EBNF since the compiler cannot understand this form;
- Perform left factoring (if necessary);
- Remove left recursion (if exist, unfortunately, they exist in the given grammar);

Strongly suggest that every time you correct a grammar please use AtoCC to check whether you correction worked or not.

Don’t try to correct many errors within one shot, it is easy to get lost.

Example

--- How to use AtoCC for verification

kfG Edit

File Help

New Open Save | Validate Grammar is regular ? | Export Automaton Export Compiler

kfG Edit | Language | Grammar | Derivation | LL(1) conditions | Definition

kfG Edit
Define Grammar

Edit: Insert: Format: Transform: CNF Panels:

Grammar

```
1  E -> E + T
2     | E - T
3     | T
4
5  T -> T * F
6     | T / F
7     | F
8
9  F -> ( E )
10    | id
11
```

type your grammar here ... ...

Symbol List

- S E
- N T
- N F
- T -
- T (
- T)
- T *
- T /
- T +
- T id

How to define a grammar:

- You only need to define your production rules here!
- Terminals can also be written within ' '. Terminals will become black and non-terminals red.
- First non-terminal on the left side will automatically be the start symbol!
- A grammar example for palindroms over {a,b}*:
  S -> a S a | b S b | EPSILON
- For epsilon rules just leave a blank in a rule or write EPSILON:

kfG Edit

File Help

New Open Save | Validate Grammar | is regular ? | Export Automaton | Export Compiler

kfG Edit | Language | Grammar | Derivation | LL(1) conditions | Definition

kfG Edit
First&Follow

LL(1) Conditions:

- Check Condition 1
- Check Condition 2
- is LL(1) Grammar ?

$E \rightarrow \alpha_0 \mid \alpha_1 \mid \alpha_2$

with:
α_0 = T
α_1 = E - T
α_2 = E + T

First-Sets:
FIRST(α_0) = {(, id}
FIRST(α_1) = {(, id}
FIRST(α_2) = {(, id}

∩	α_0	α_1	α_2
α_0	-	{(, id}	{(, id}
α_1	{(, id}	-	{(, id}
α_2	{(, id}	{(, id}	-

$T \rightarrow \alpha_0 \mid \alpha_1 \mid \alpha_2$

with:
α_0 = F
α_1 = T / F
α_2 = T * F

First-Sets:
FIRST(α_0) = {(, id}
FIRST(α_1) = {(, id}
FIRST(α_2) = {(, id}

∩	α_0	α_1	α_2
α_0	-	{(, id}	{(, id}
α_1	{(, id}	-	{(, id}

$E \rightarrow \alpha_0 \mid \alpha_1 \mid \alpha_2$

with:

α_0 = T

α_1 = E - T

α_2 = E + T

First-Sets:

FIRST(α_0) = {(, id}

FIRST(α_1) = {(, id}

FIRST(α_2) = {(, id}

$\cap$	α_0	α_1	α_2
α_0	-	{(, id}	{(, id}
α_1	{(, id}	-	{(, id}
α_2	{(, id}	{(, id}	-

$T \rightarrow \alpha_0 \mid \alpha_1 \mid \alpha_2$

with:

α_0 = F

α_1 = T / F

α_2 = T * F

First-Sets:

FIRST(α_0) = {(, id}

FIRST(α_1) = {(, id}

FIRST(α_2) = {(, id}

$\cap$	α_0	α_1	α_2
α_0	-	{(, id}	{(, id}
α_1	{(, id}	-	{(, id}
α_2	{(, id}	{(, id}	-

first set intersection

$F \rightarrow \alpha_0 \mid \alpha_1$

with:

α_0 = **id**

α_1 = (**E**)

First-Sets:

FIRST(α_0) = {**id**}

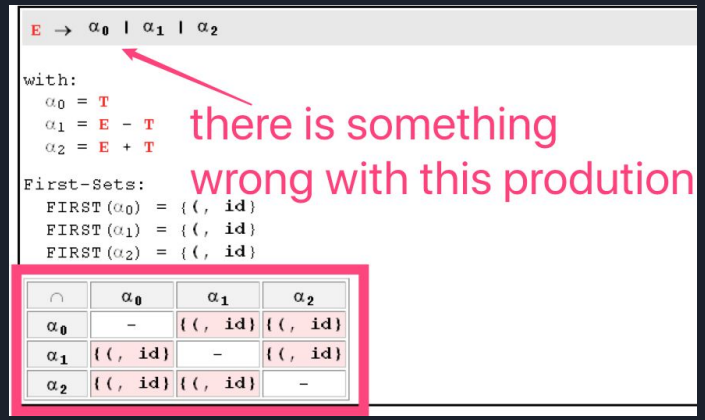
FIRST(α_1) = {(}

$\cap$	α_0	α_1
α_0	–	$\varnothing$
α_1	$\varnothing$	–

go to the very end of the page

LL(1) first condition not fulfilled!

What you should do?



1. Locate where the error, you can check the production
2. Copy the relate production put into the second tool we mention above (https://cyberzhg.github.io/toolbox/cfg2ll).
3. Copy the correction from the tool and paste it into AtoCC
4. Do some modification to adapt AtoCC format
5. Check the grammar again

Note: Don’t try to solve more than one production at a time. When you solve one production’s error, use the tool to check to make sure you are not bringing new errors.

```
E -> T E''
T -> F T''
F -> ( E )
   | id
E' -> + T
    | - T
T' -> * F
    | / F
E'' -> E' E''
     | ?
T'' -> T' T''
     | ?
```

result from the tool

```
E -> T ETailTail
T -> F TTailTail
F -> ( E )
   | id
ETail -> + T
       | - T
TTail -> * F
       | / F
ETailTail -> ETail ETailTail
           | EPSILON
TTailTail -> TTail TTailTail
           | EPSILON
```

after modification adapt to AtoCC

kfG Edit

File Help

New Open Save Validate Grammar is regular ? Export Automaton Export Compiler

kfG Edit | Language | Grammar | Derivation | LL(1) conditions | Definition

kfG Edit
First&Follow

LL(1) Conditions:

- Check Condition 1
- Check Condition 2
- is LL(1) Grammar ?

$E \rightarrow \alpha_0$

with:
α_0 = T ETialTial

First-Sets:
FIRST(α_0) = {(, id}

$T \rightarrow \alpha_0$

with:
α_0 = F TTialTial

First-Sets:
FIRST(α_0) = {(, id}

kfG Edit

LL(1) first condition fulfilled!
LL(1) second condition fulfilled!

OK

$F \rightarrow \alpha_0 \mid \alpha_1$

with:
α_0 = id
α_1 = (E)

First-Sets:
FIRST(α_0) = {id}
FIRST(α_1) = {(}

∩	α_0	α_1
α_0	–	∅
α_1	∅	–

$ETail \rightarrow \alpha_0 \mid \alpha_1$

with:

Genesis-X7 Software 2007 - 2008

LL(1) first condition fulfilled!

```
FIRST (ETailTail) = {+, -, EPSILON}
FOLLOW(ETailTail) = {$, )}
FIRST (ETailTail) ∩ FOLLOW(ETailTail) = ∅
```

```
FIRST (TTailTail) = {*, /, EPSILON}
FOLLOW(TTailTail) = {$, ), +, -}
FIRST (TTailTail) ∩ FOLLOW(TTailTail) = ∅
```

LL(1) second condition fulfilled!

Thanks