

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/>

Ongoing courses

- SOEN 487, 2018 Winter
- **COMP 442 / 6421, 2018 Winter**





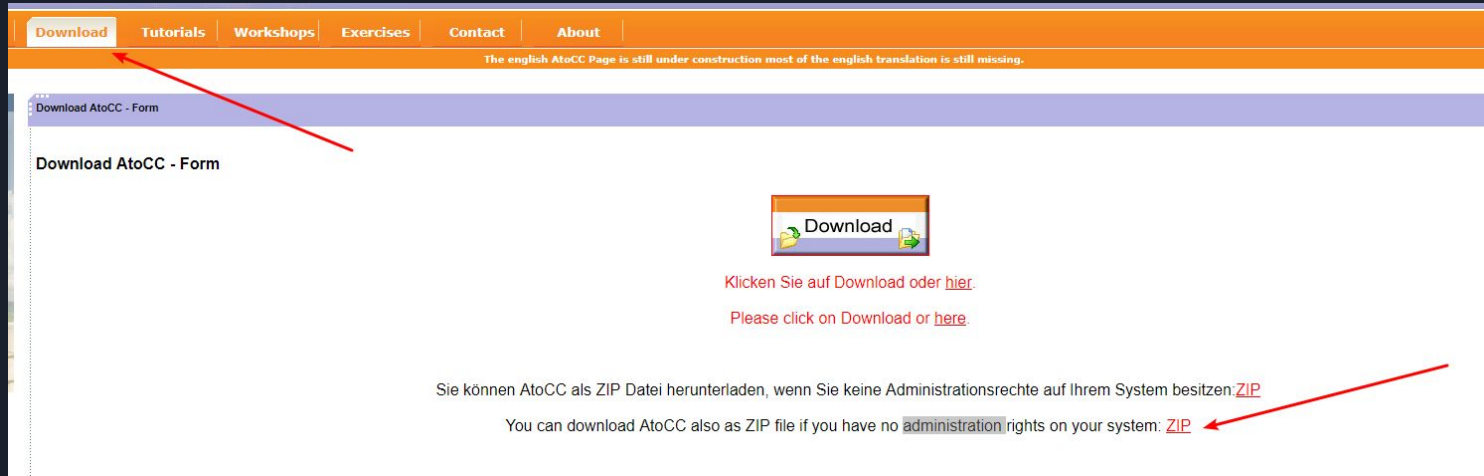
You don't have Window machine?

Check the following link out:

http://atocc.de/AtoCCFAQ/index.php?option=com_content&task=category§ionid=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 ?



The screenshot shows the AtoCC website interface. At the top, there is a navigation bar with tabs: Download, Tutorials, Workshops, Exercises, Contact, and About. A red arrow points to the 'Download' tab. Below the navigation bar, there is a message: "The english AtoCC Page is still under construction most of the english translation is still missing." The main content area is titled "Download AtoCC - Form". In the center, there is a "Download" button with a green arrow icon. Below the button, there is text in German: "Klicken Sie auf Download oder [hier](#)." and in English: "Please click on Download or [here](#)." At the bottom, there is text in German: "Sie können AtoCC als ZIP Datei herunterladen, wenn Sie keine Administrationsrechte auf Ihrem System besitzen: [ZIP](#)" and in English: "You can download AtoCC also as ZIP file if you have no [administration](#) rights on your system: [ZIP](#)". A red arrow points to the 'ZIP' link in the English text.

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

Another powerful tool

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

Introduction

Try converting the given context free grammar to LL(k) class by performing left-factoring then eliminating left recursion.

Supported grammars

- $A \rightarrow A c \mid A a d \mid b d \mid \epsilon$
(All tokens must be separated by space characters)
- $A \rightarrow A c$
 - | $A a d$
 - | $b d$
 - | ϵ
- $S \rightarrow A a \mid b$
- $A \rightarrow A c \mid S d \mid \epsilon$
- (Copy ϵ to input if needed)

Examples

- $S \rightarrow S S + \mid S S * \mid a$
- $S \rightarrow 0 S 1 \mid 0 1$
- $S \rightarrow + S S \mid * S S \mid a$
- $S \rightarrow S (S) S \mid \epsilon$
- $S \rightarrow S + S \mid S S \mid (S) \mid S * \mid a$
- $S \rightarrow (L) \mid a L \rightarrow L , S \mid S$
- $S \rightarrow a S b S \mid b S a S \mid \epsilon$
- $bexpr \rightarrow bexpr \text{ or } bterm \mid bterm$
 $bterm \rightarrow bterm \text{ and } bfactor \mid bfactor$
 $bfactor \rightarrow \text{not } bfactor \mid (bexpr) \mid \text{true} \mid \text{false}$

Input: $A \rightarrow A c \mid A a d \mid b d \mid \epsilon$

Output:

CONVERT



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)

Assume you was given a grammar as following:

commaSeparatedList -> a {,a} | EPSILON

You should remove the EBNF format and come up with the following grammar:

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?

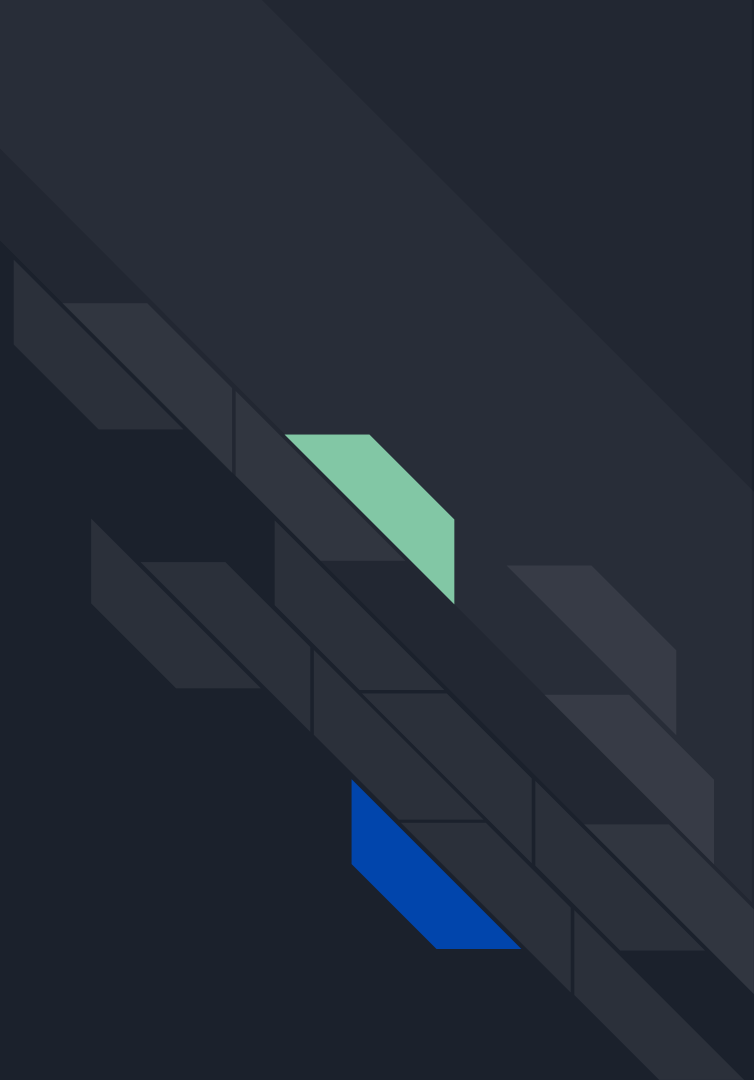
- You receive the initial grammar in EBNF in assignment 2 description already;
- You need to remove the EBNF since the compiler cannot understand this form;
- Perform left factoring (if necessary);
- Remove left recursion (if exist, unfortunately, they exist in the given grammar);


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





File Help

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

kgf Edit Language Grammar Derivation LL(1) conditions Definition

kgf Edit

Define Grammar

Edit: [undo] [redo] [insert] [delete] [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
```

Symbol List

- E
- T
- F
- .
- (
-)
- *
- /
- +
- id

type your grammar here

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 $\{a,b\}^*$:
 $S \rightarrow a S a \mid b S b \mid \text{EPSILON}$
- For epsilon rules just leave a blank in a rule or write EPSILON:

Genesis-X7 Software 2007 - 2008

File Help

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

kgf Edit Language Grammar Derivation **LL(1) conditions** Definition

kgf 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:

$\alpha_0 = T$
 $\alpha_1 = E - T$
 $\alpha_2 = E + T$

First-Sets:

$FIRST(\alpha_0) = \{ (, id \}$
 $FIRST(\alpha_1) = \{ (, id \}$
 $FIRST(\alpha_2) = \{ (, id \}$

\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:

$\alpha_0 = F$
 $\alpha_1 = T / F$
 $\alpha_2 = T * F$

First-Sets:

$FIRST(\alpha_0) = \{ (, id \}$
 $FIRST(\alpha_1) = \{ (, id \}$
 $FIRST(\alpha_2) = \{ (, id \}$

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

Genesis-X7 Software 2007 - 2008

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

with:

$\alpha_0 = T$

$\alpha_1 = E - T$

$\alpha_2 = E + T$

First-Sets:

$\text{FIRST}(\alpha_0) = \{ (, \text{id} \}$

$\text{FIRST}(\alpha_1) = \{ (, \text{id} \}$

$\text{FIRST}(\alpha_2) = \{ (, \text{id} \}$

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

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

with:

$\alpha_0 = F$

$\alpha_1 = T / F$

$\alpha_2 = T * F$

First-Sets:

$\text{FIRST}(\alpha_0) = \{ (, \text{id} \}$

$\text{FIRST}(\alpha_1) = \{ (, \text{id} \}$

$\text{FIRST}(\alpha_2) = \{ (, \text{id} \}$

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

first set intersection

$F \rightarrow \alpha_0 \mid \alpha_1$

with:

$\alpha_0 = \text{id}$

$\alpha_1 = (\text{E})$

First-Sets:

$\text{FIRST}(\alpha_0) = \{\text{id}\}$

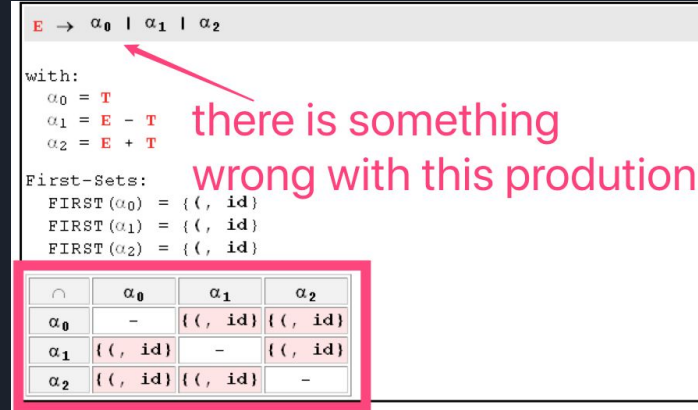
$\text{FIRST}(\alpha_1) = \{ (\}$

\cap	α_0	α_1
α_0	-	\emptyset
α_1	\emptyset	-

go to the very end of the page

LL(1) first condition not fulfilled!

What you should do?



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

with:

- $\alpha_0 = T$
- $\alpha_1 = E - T$
- $\alpha_2 = E + T$

First-Sets:

- $\text{FIRST}(\alpha_0) = \{ (, \text{id} \}$
- $\text{FIRST}(\alpha_1) = \{ (, \text{id} \}$
- $\text{FIRST}(\alpha_2) = \{ (, \text{id} \}$

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

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.


```

14 E -> T E ' '
15 T -> F T ' '
16 F -> ( E )
17   | id
18 E ' -> + T
19   | - T
20 T ' -> * F
21   | / F
22 E ' ' -> E ' E ' '
23   | ?
24 T ' ' -> T ' T ' '
25   | ?

```

result from the tool

```

1 E -> T ETailTail
2 T -> F TTailTail
3 F -> ( E )
4   | id
5 ETail -> + T
6       | - T
7 TTail -> * F
8       | / F
9 ETailTail -> ETail ETailTail
10           | EPSILON
11 TTailTail -> TTail TTailTail
12           | EPSILON
13

```

after modification adapt to AtoCC

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:
 $\alpha_0 = \text{ T ETialTial }$

First-Sets:
 $\text{FIRST}(\alpha_0) = \{ (, \text{ id})$

T $\rightarrow \alpha_0$

with:
 $\alpha_0 = \text{ F TTialTial }$

First-Sets:
 $\text{FIRST}(\alpha_0) = \{ (, \text{ id})$

F $\rightarrow \alpha_0 \mid \alpha_1$

with:
 $\alpha_0 = \text{ id }$
 $\alpha_1 = (\text{ E })$

First-Sets:
 $\text{FIRST}(\alpha_0) = \{ \text{ id} \}$
 $\text{FIRST}(\alpha_1) = \{ (\}$

\cap	α_0	α_1
α_0	-	\emptyset
α_1	\emptyset	-

ETail $\rightarrow \alpha_0 \mid \alpha_1$

with:

kfG Edit

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

OK

LL(1) first condition fulfilled!

FIRST (ETailTail) = {+, -, EPSILON}

FOLLOW(ETailTail) = {\$,)}

FIRST (ETailTail) \cap FOLLOW(ETailTail) = \emptyset

FIRST (TTailTail) = {*, /, EPSILON}

FOLLOW(TTailTail) = {\$,), +, -}

FIRST (TTailTail) \cap FOLLOW(TTailTail) = \emptyset

LL(1) second condition fulfilled!

Thanks

