McMaster University Comp Sci 4TB3/6TB3, Winter Term 2017/18 — Lab 2
For the Labs on January 16 - 19,
Due Monday, January 22, 11 pm

Eden Burton, Spencer Park, Jenny Wang

- Solutions are to be submitted electronically on Avenue under Assessments → Assignments. In this lab, you are asked to write the submission with LaTeX and submit one .tex and one .pdf file. The files should be named yourmacid.tex and yourmacid.pdf, where yourmacid is your McMaster e-mail address, not your student number. Please do not compress these files or submit a compressed directory with the files. The .tex file should be self-contained, i.e. not include any other file, except the style files mentioned below, and should compile with `pdflatex yourmacid.tex`.
- In this lab, you are allowed to work in pairs, provided that you split the work equally and arrive at a common understanding of the solution. However, in that case you must state in your submission the person you worked with, such that similarities in the solution will not be construed as Academic Dishonesty. Working in groups of three or larger is not allowed and will be considered Academic Dishonesty. If you look for someone to work with, we will try to find a match, please contact the TAs.
- You are allowed and encouraged to talk to everyone in the course to get a common understanding of the problem, but you can share a solution only with your collaborator, if you work in a pair.
- If you work on own computer, you will need a TeX distribution, like TeX Live at `tug.org`; for Macs, MacTeX is recommended; on Windows, MiKTeX is popular. You might need to install packages from `ctan.org`. You can also run LaTeX remotely on either `moore.mcmaster.ca` or `mills.mcmaster.ca`. Another option is working in the cloud-based LaTeX compiler such as ShareLaTex.
- This lab asks you to draw syntax diagrams. For this, you have to use a LaTeX package: `mdwtools` is a is included in TeX Live and MiKTeX, but you can use another package of `ctan.org/topic/syntax`. Do not use a package that is not in `ctan.org`. See the web site under Further Material for an example.
- The Tutorial Exercises will be presented in the tutorials; you need to submit only answers to the Lab Questions. In the labs, the solution to last week's lab questions are discussed and you can get help with this week's lab questions. Attendance at the labs is not checked.

Tutorial Exercise 1 (Syntax Diagrams)**.** Consider following grammar for arithmetic expressions:

$$\begin{aligned} \textit{expression} &::= [\text{"}+\text{"} \mid \text{"}-\text{"}]\ \textit{term}\ \{(\text{"}+\text{"} \mid \text{"}-\text{"})\ \textit{term}\} && (1)\\ \textit{term} &::= \textit{factor}\ \{(\text{"}*\text{"} \mid \text{"}/\text{"})\ \textit{factor}\} && (2)\\ \textit{factor} &::= \text{number} \mid \text{identifier} \mid \text{"(" } \textit{expression} \text{ ")"} && (3) \end{aligned}$$

Use the `mdwtools` package to (a) pretty-print above grammar, (b) to draw the syntax diagrams of the three nonterminals and (c) to draw the complete productions with syntax trees.

Answer.

a.
```
\begin{grammar}
<expression> ::= ['$+$' | '$-$'] <term> \{('$+$' | '$-$' ) <term>\}

<term> ::= <factor> \{('$*$' | '$/$') <factor>\}

<factor> ::= "number" | "identifier"  | '(' <expression> ')'
\end{grammar}
```

produces:

⟨*expression*⟩ ::= ['$+$' | '$-$'] ⟨*term*⟩ {('$+$' | '$-$') ⟨*term*⟩}

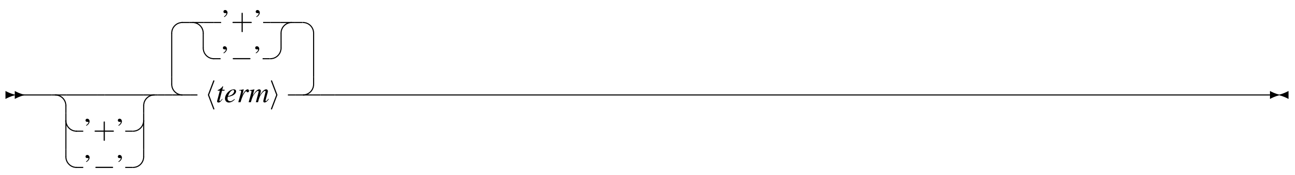
⟨*term*⟩ ::= ⟨*factor*⟩ {('$*$' | '/') ⟨*factor*⟩}

⟨*factor*⟩ ::= `number` | `identifier` | '(' ⟨*expression*⟩ ')'

b. Syntax of *expression*:

```
\begin{syntdiag}
  \begin{stack}\\'$+$'\\'$-$'\end{stack}
  \begin{rep}
    <term>\\
    \begin{stack}'$+$'\\'$-$'\end{stack}
  \end{rep}
\end{syntdiag}
```

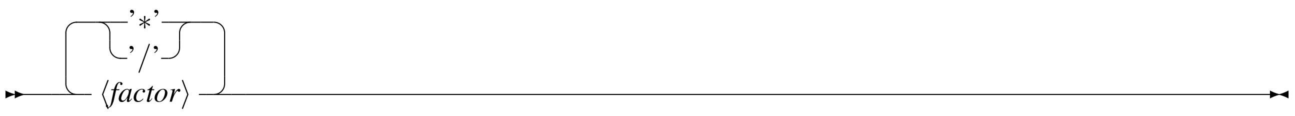
produces:



Syntax of *term*:

```
\begin{syntdiag}
  \begin{rep}
    <factor>\\
    \begin{stack}'$*$'\\'$/$'\end{stack}
  \end{rep}
\end{syntdiag}
```

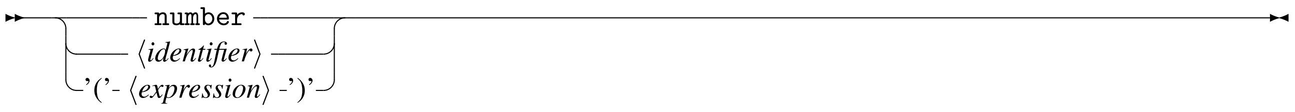
produces:



Syntax of *factor*:

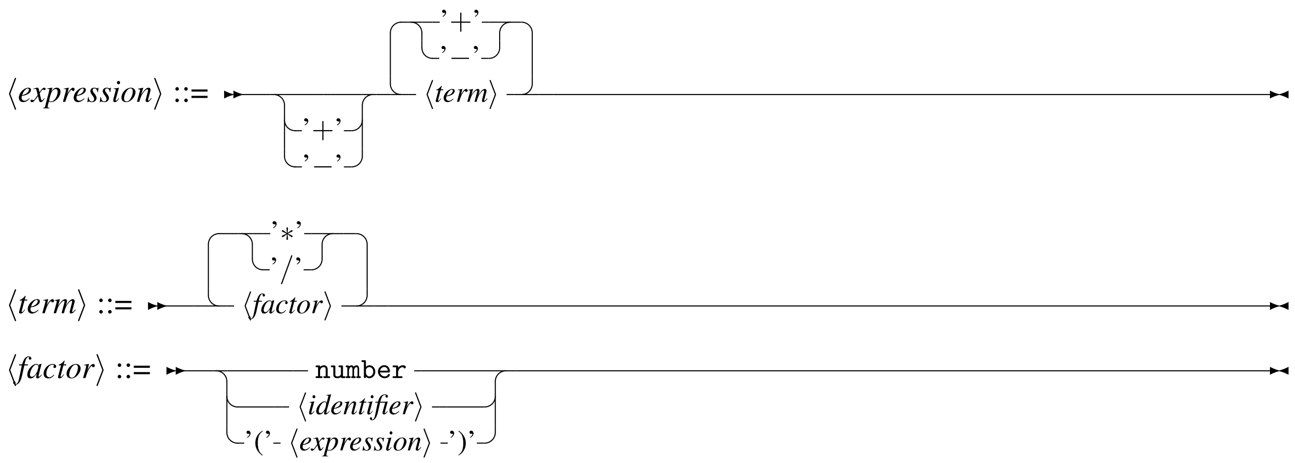
```
\begin{syntdiag}
  \begin{stack} "number"\\<identifier>\\'(' <expression>')'\end{stack}
\end{syntdiag}
```

produces:



c.
```
\begin{grammar}
<expression> ::= \[[
  \begin{stack}\\'$+$'\\'$-$'\end{stack}
  \begin{rep} <term>\\ \begin{stack}'$+$'\\'$-$'\end{stack}\end{rep}
\]]
<term> ::= \[[
  \begin{rep} <factor>\\ \begin{stack}'$*$'\\'$/$'\end{stack}\end{rep}
\]]
<factor> ::= \[[
\begin{stack} "number"\\<identifier>\\'(' <expression>')'\end{stack}
\]]
\end{grammar}
```

produces:



Tutorial Exercise 2 (Describing Regular Expressions)**.** In this question, we use $\{\ldots\}$ for sets and * for repetition. Describe the languages generated by following regular expressions.

a. $0(0 \mid 1)^*0$

Answer. The set of strings over $\{0, 1\}$ of length 2 or longer and starting and ending with 0.

b. $([0]1)^*$

Answer. The set of strings over $\{0, 1\}$ ending in 1 (if length ≥ 1) and such that any two 0's are separated by at least one 1.

Lab Question 1 (Syntax Diagrams)**.** Draw syntax diagrams ("railroad diagrams") for the hexadecimal numerals in Java as defined at `https://docs.oracle.com/javase/specs/jls/se8/jls8.pdf` (see section 3.10.1), using `mdwtools`. Simpify by assuming ranges where appropriate. For example, it is not necessarily to include a separate path for each digit from 0 to 9, a single path for 0..9 is sufficent.

Lab Question 2 (Describing Regular Expressions)**.** Describe the languages generated by following regular expressions.

a. $(a^*b^*)^*$

b. $(a^*ba^*b)^*a^*$

c. $(a^*[ba^*c])^*$

Lab Question 3 (Equivalence of Regular Expressions)**.** Are the following regular expressions equivalent? Either argue why or give a counterexample!

a. $bd \mid bc^*d$ and bc^*d

b. $ab \mid bd^*$ and $[a]bd^*$

c. $(a \mid ba)^*$ and $(a \mid b)^*$

Lab Question 4 (Regular Grammar to Regular Expressions)**.** Transform following regular grammar to a regular expression by following the procedure from the course notes.

$$\begin{array}{lcl} A & \rightarrow & aA \mid bB \\ B & \rightarrow & bB \mid cB \mid \varepsilon \end{array}$$

Lab Question 5 (Regular Expression to Regular Grammar)**.** Give a regular grammar that generates the language described by following regular expression:

$$a^* \mid b^* \mid (ab)^*$$

Lab Question 6 (Non-Deterministic Finite State Machines)**.** Convert the following non-deterministic finite state machine (NFA) into a deterministic one (DFA) using the algorithim described in the notes. Explicitly show your steps. Draw the generated DFA using a LaTex drawing tool such as `tikz`.