# Compilers: Bison

a topic in

DM565 – Formal Languages and Data Processing

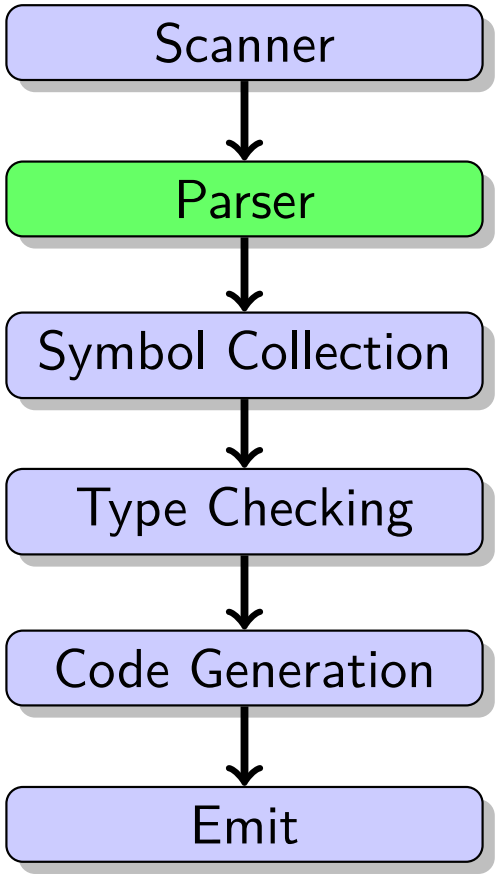
Kim Skak Larsen

Department of Mathematics and Computer Science (IMADA)
University of Southern Denmark (SDU)

*kslarsen@imada.sdu.dk*

October, 2023

# Syntax Analysis: parsers



SDU

## The Parsing Problem

We have a grammar $G$ (partially) defining the programming language and a string $s$ in the form of the user's program.

We want to know if the user program is correct, i.e., if $s \in L(G)$.

We consider the tool `bison` that implements an LALR(1)-parser.

# Syntax Analysis: parsers

SDU

## Input to phase

A stream of tokens (keywords, numbers, identifiers, symbols) from `flex`

## Output from phase

An abstract syntax tree (AST)

# Bison

`bison` – successor to Yet Another Compiler Compiler (`yacc`)

Tool available for many programming languages: C, Java, ...

Same functionality available in Python with native syntax.

# Bison

## Format

```
%{
  C DEFINITIONS
%}
  BISON DEFINITIONS
%%
  GRAMMAR AND ACTIONS
%%
  C CODE
```

# Bison

## How to Run Bison

```
> bison FILENAME.y                    -- makes FILENAME.tab.c
```

See the `make` file for how to run with `flex`.

Option **`--report=all`** gives `FILENAME.output` with DFA states.
Option **`--defines`** gives `FILENAME.tab.h` with tokens for `flex`.

SDU

## Bison Examples

- first example (`one_variable.y`)
- ingredients
- error report – we inspect State 12 – example input 3*4+5
- fixes
  - rewrite grammar (`three_variables.y`)
  - use precedence directives (`precedence.y`) – we inspect States 12 and 14
    - *left* associative $\equiv$ reduce (`%left`)
    - *right* associative $\equiv$ shift (`%right`)
    - *non*-associative $\equiv$ error (`%nonassoc`)

SDU

## A Stack Machine

- larger example (`StackMachine`: `flex`, `bison` files, etc.)
- when are actions executed?
- what is `yylval`?
- what does `$1` etc. mean?
- what does `yyparse` do?
- run on `a*(b-17)+5/c`

This is actually a compiler!

SDU

## Building the AST

- AST example (`TinyExpressions`: `flex`, `bison` files, etc.)
- what is `extern`?
- what is `union`?
- what is `token`/`type`?
- run on `3+4-5*6/7+8`
- run by hand on `3+4*5`

SDU

## Adding to the AST

- what does it take to add line numbers?
- just an example – also `bison` support via `yylineno`.

SDU

## Parser Construction in Python

In Python, the same functionality known from `bison` is realized using a more native Python code style using the module `ply.yacc`.

See how it is done in `scil`...

- rules in documentation strings
- `$$`, `$1`, `$2`, etc. is `t[0]`, `t[1]`, `t[2]`, etc.

# Other Tools

SDU

**ANTLR**

"ANother Tool for Language Recognition"

Implements a type of LL-parsing with unbounded lookahead – LL($*$).

- Has become quite popular in recent years – more modern interface.
- Can handle more than any LL($k$)-parser.
- Less clear when and why a grammar fails to be parsable using this technique.
- Necessary for some languages – possibly due to poor language design.
- Slower on average than LALR(1)-parsing.
- Exponential worst-case running time.