



Section

Compila 19

Chapter 0 ""

Course "Compiler Construction"

Martin Steffen

Spring 2019

Oblig 1



INF5110 – Oblig
1 + 2

Compila 19

Tools

Official

- material (also for oblig 2) based on previous years, including contributions from Eyvind W. Axelsen, Henning Berg, Fredrik Sørensen, and others
- see also the course web-page, containing links to “resources”

Goal (of oblig 1)



INF5110 – Oblig
1 + 2

Compila 19

Tools

Official

Parsing

Determine if programs written in *Compila 19* are syntactically correct:

- scanner
 - parser
-
- first part of a compiler, oblig 2 will add to it
 - language spec provided separatly

Learning outcomes



INF5110 – Oblig
1 + 2

Compila 19

Tools

Official

- using **tools** for parser/scanner generation
 - JFlex
 - CUP
- variants of a grammar for the same languages
 - **transforming** one form (EBNF) to another (compatible with the used tools)
 - controlling **precedence** and **associativity**
- designing and implementing an **AST** data structure
 - using the parsing tools to build such trees
 - pretty-printing such trees

Compila language at a glance



INF5110 – Oblig
1 + 2

Compila 19

Tools

Official

```
program MyProgram
begin
    struct complex {          // record data type, but
        re: float;           // no subtyping, polymorphism ...
        im: float
    }
end;

proc add (a: complex, b: complex) : complex
begin
    var retval : complex;
in
    retval := new complex;
    retval.re := a.re + b.re;
    retval.im := a.im + b.im;
    return retval
end;

proc main()                  // execution start here
begin
    var c1: complex;
    var c2: complex;
    result := add (c1, c2);
    ...
    return
end
end
```

Another glance

```
proc swap (a: ref(int), b: ref(int)) // passed as reference
begin
  var tmp: int;
  tmp := deref(a); // dereferencing
  deref(a) := deref(b); // deref can be used both
  deref(b) := tmp // left and right of
                  // an assignment.
end;
```

Grammar (1): declarations

PROGRAM
"end"

-> "program" NAME "begin" [DECL { ";" DECL }]

DECL

-> VAR_DECL | PROC_DECL | REC_DECL

VAR_DECL

-> "var" NAME ":" TYPE

PROC_DECL

-> "proc" NAME
"(" [PARAMFIELD_DECL { "," PARAMFIELD_DECL }]
[":" TYPE]
"begin" [DECL { ";" DECL }] "in" [STMT { ";" STMT }]

REC_DECL

-> "struct" NAME "{" [PARAMFIELD_DECL
{ ";" PARAMFIELD_DECL }]
"}"

Grammar (2): declarations

PARAMFIELD_DECL -> NAME ":" TYPE

EXP -> EXP LOG_OP EXP
 | "not" EXP
 | EXP REL_OP EXP
 | EXP ARIT_OP EXP
 | LITERAL
 | CALL_STMT
 | "new" NAME
 | VAR
 | REF_VAR
 | Deref_VAR
 | "(" EXP ")"

REF_VAR -> "ref" "(" VAR ")"

DEREF_VAR -> "deref" "(" VAR ")" | "deref" "(" Deref_VAR ")"

VAR -> NAME | EXP "." NAME

LOG_OP -> "&&" | "||"

REL_OP -> "<" | "<=" | ">" | ">=" | "=" | "<>"

ARIT_OP -> "+" | "-" | "*" | "/" | "^"

Grammar (3): statements and types

LITERAL	-> FLOAT_LITERAL INT_LITERAL STRING_LITERAL "true" "false" "null"
STMT	-> ASSIGN_STMT IF_STMT WHILE_STMT RETURN_STMT CALL_STMT
ASSIGN_STMT	-> VAR ":"=" EXP Deref_VAR ":"=" EXP
IF_STMT	-> "if" EXP "then" { STMT ";" } ["else" { STMT ";" }] "fi"
WHILE_STMT	-> "while" EXP "do" { STMT ";" } "od"
RETURN_STMT	-> "return" [EXP]
CALL_STMT	-> NAME "(" [EXP { "," EXP }] ")"
TYPE	-> "float" "int" "string" "bool" NAME "ref" "(" TYPE ")"



Section

Tools

Chapter 0 ""

Course "Compiler Construction"

Martin Steffen

Spring 2019



- scanner generator (or lexer generator) tool
 - **input**: lexical specification
 - **output**: scanner program in Java
- lexical spec written as `.lex` file
- consists of **3 parts**
 - user code
 - options and macros
 - lexical rules

Sample lex code



INF5110 – Oblig
1 + 2

Compila 19

Tools

Official

User code

```
package oblig1parser;  
import java_cup.runtime.*;
```

Copied to the generated class, before
the class definition

```
%%
```

Options/
macros

```
%class Lexer Options (class name, unicode support,  
%unicode CUP integration)  
%cup
```

```
%{  
    private Symbol symbol(int type) {  
        return new Symbol(type, yyline, yycolumn);  
    }  
}
```

Defined in package
java_cup.runtime.

Inserted into
generated class

Variables holding
current line/column

```
%}  
LineTerminator = \r|\n|\r\n
```

Macros, defined as
regular expressions

```
%%
```

Lexical
rules

```
<YYINITIAL> The following rules are applicable from the initial state  
{  
    "program" { return symbol(sym.PROGRAM); }  
    "class" { return symbol(sym.CLASS); }  
    "begin" { return symbol(sym.BEGIN); }  
    "end" { return symbol(sym.END); }  
    "var" { return symbol(sym.VAR); }  
    ""  
}
```

Refers to names in
the .cup file (next
slides)

Lexical rules

CUP: Construction of useful parsers (for Java)



INF5110 – Oblig
1 + 2

Compila 19

Tools

Official

- a tool to easily (yymv) generate *parsers*
- reads tokens from the scanner using `next_token()`
- the `%cup` option (previous slide) makes that work

Input

grammar in BNF with **action** code

```
var_decl ::= VAR ID:name COLON type:vtype  
{: RESULT = new VarDecl(name, vtype); :};
```

- **output:** parser program (in Java)

Sample CUP code



INF5110 – Oblig
1 + 2

Compila 19

Tools

Official

Package/ imports	<pre>package oblig1parser; import java_cup.runtime.*; import syntaxtree.*;</pre>	<p>Package name for generated code and imports of packages we need</p> <p>The syntaxtree package contains our own AST classes</p>
-----------------------------	--	---

User code	<pre>parser code { : ;};</pre>	<p>Code between { : and : } is inserted directly into the generated class (parser.java)</p>
------------------	--------------------------------	---

Symbol list	<pre>terminal PROGRAM, CLASS; terminal BEGIN, END; ... terminal String ID; terminal String STRING_LITERAL; non terminal Program program; non terminal List<ClassDecl> decl_list; non terminal ClassDecl class_decl, decl;</pre>	<p>Terminals and non-terminals are defined here. They can also be given a Java type for the "value" that they carry, e.g. a node in the AST</p>
------------------------	--	---

Precedence	<pre>precedence left AND;</pre>	<p>Precedence declarations are listed in ascending order, last = highest</p>
-------------------	----------------------------------	--

Grammar	<pre>program := PROGRAM BEGIN decl_list:dl END SEMI { : RESULT = new Program(dl); :} ; decl_list := decl:d { : List<ClassDecl> l = new LinkedList<ClassDecl>(); l.add(d); RESULT = l; :} ; decl := class_decl:sd { : RESULT = sd; :} ; class_decl := CLASS ID:name BEGIN END { : RESULT = new ClassDecl(name); :} ;</pre>	<p>AST is built during parsing. The left hand side of each production is implicitly labeled RESULT.</p>
----------------	---	---

Build tool: ant



- Java-based build tool (think “make”)
- config in `build.xml`
- can contain different **targets**

typical general targets

- test
 - clean
 - build
 - run
-
- supplied configuration should take care of calling `jflex`, `cup`, and `javadoc` for you



INF5110 – Oblig
1 + 2

Compila 19

Tools

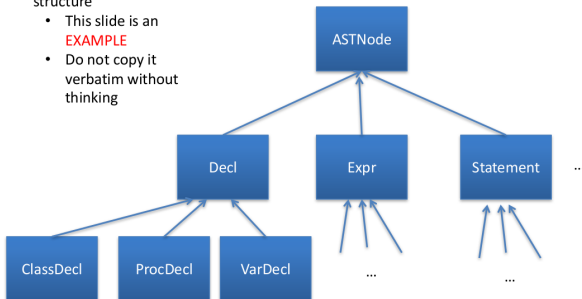
Official

AST data structure



INF5110 – Oblig
1 + 2

- Make a reasonable structure
 - This slide is an **EXAMPLE**
 - Do not copy it verbatim without thinking



Compila 19

Tools

Official

Overview over the directory + first steps

- see the Readme at/from the `github.uio.no`

```
/uio/kant/ifi-ansatt-u00/msteffen/TMP/compila:
total used in directory 60 available 52814464
drwxrwxr-x. 11 msteffen ifi 2048 Feb 18 08:04 .
drwxrwxr-x. 3 msteffen ifi 2048 Feb 18 08:04 ..
drwxrwxr-x. 8 msteffen ifi 2048 Feb 18 08:04 .git
-rw-rw-r-- 1 msteffen ifi 66 Feb 18 08:04 .gitignore
-rw-rw-r-- 1 msteffen ifi 5267 Feb 18 08:04 Readme.org
drwxrwxr-x. 3 msteffen ifi 2048 Feb 18 08:05 build
-rwxrwxr-x. 1 msteffen ifi 3231 Feb 18 08:04 build.xml
drwxrwxr-x. 5 msteffen ifi 2048 Feb 18 08:04 doc
drwxrwxr-x. 2 msteffen ifi 2048 Feb 18 08:04 lib
drwxrwxr-x. 5 msteffen ifi 2048 Feb 18 08:04 material
drwxrwxr-x. 4 msteffen ifi 2048 Feb 18 08:04 previoussemesters
drwxrwxr-x. 8 msteffen ifi 2048 Feb 18 08:04 src
drwxrwxr-x. 3 msteffen ifi 2048 Feb 18 08:05 src-gen
drwxrwxr-x. 3 msteffen ifi 2048 Feb 18 08:04 tmp
```

```
/uio/kant/ifi-ansatt-u00/msteffen/TMP/compila/lib:
total used in directory 280 available 52814464
drwxrwxr-x. 2 msteffen ifi 2048 Feb 18 08:04 .
drwxrwxr-x. 11 msteffen ifi 2048 Feb 18 08:04 ..
-rwxrwxr-x. 1 msteffen ifi 179102 Feb 18 08:04 JFlex.jar
-rwxrwxr-x. 1 msteffen ifi 96121 Feb 18 08:04 java-cup-11a.jar
```

```
/uio/kant/ifi-ansatt-u00/msteffen/TMP/compila/src:
total used in directory 32 available 52814464
drwxrwxr-x. 8 msteffen ifi 2048 Feb 18 08:04 .
drwxrwxr-x. 11 msteffen ifi 2048 Feb 18 08:04 ..
drwxrwxr-x. 2 msteffen ifi 2048 Feb 18 08:04 compiler
drwxrwxr-x. 6 msteffen ifi 2048 Feb 18 08:04 doc
drwxrwxr-x. 2 msteffen ifi 2048 Feb 18 08:04 grammars
drwxrwxr-x. 2 msteffen ifi 2048 Feb 18 08:04 org
drwxrwxr-x. 2 msteffen ifi 2048 Feb 18 08:04 syntaxtree
drwxrwxr-x. 6 msteffen ifi 2048 Feb 18 08:04 tests
```

```
/uio/kant/ifi-ansatt-u00/msteffen/TMP/compila/src-gen:
total used in directory 16 available 52814464
drwxrwxr-x. 3 msteffen ifi 2048 Feb 18 08:05 .
drwxrwxr-x. 11 msteffen ifi 2048 Feb 18 08:04 ..
-rw-rw-r-- 1 msteffen ifi 13 Feb 18 08:04 .gitignore
drwxrwxr-x. 2 msteffen ifi 2048 Feb 18 08:05 parser
```



INF5110 – Oblig
1 + 2

Compila 19

Tools

Official

Building: putting it together

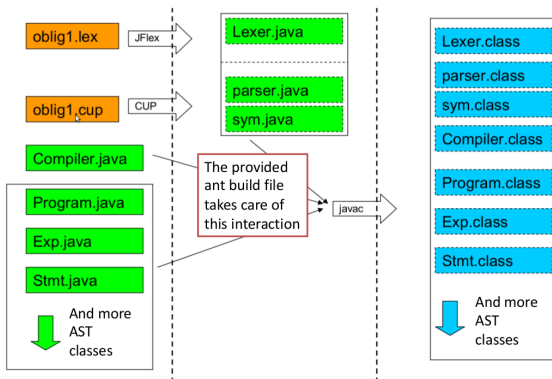


INF5110 – Oblig
1 + 2

Compila 19

Tools

Official





Section

Official

Chapter 0 ""
Course "Compiler Construction"
Martin Steffen
Spring 2019

Deadline

Friday 15. 03. 2019, 23:59

- don't miss the deadline
- for extensions, administration needs to agree (studadm), contact them if sick etc
- even if not 100% finished
 - deliver what you have
 - contact early when problems arise

Compila 19

Tools

Official

- see also the “handout”

Deliverables (1)

- working **parser**
 - parse the supplied sample programs
 - printout the resulting AST
- **two** grammars (two `.cup`-files)
 - one unambiguous
 - one ambiguous, where ambiguities resolved through precedence declarations in *CUP*, e.g.

precedence left AND;



Deliverables (2)

- report (with name(s) and UiO user name(s))
 - discussion of the solution (see handout for questions)
 - in particular: comparison of the two grammars
 - “Readme”
-
- the code must *build* (with ant) and run
 - test it on the UiO RHEL (linux) platform

Ask

If problems, **ask in time** (**NOT** Friday at the deadline)

Hand-in procedure



INF5110 – Oblig
1 + 2

Compila 19

Tools

Official

- this year we try *git*
- `https://github.uio.no` resp.
`https://github.uio.no/msteffen/compila`
- you need
 - a login
 - send me emails that you want to do oblig (+ potential partner) \Rightarrow I tell you group number
 - create a project `compila<n>` (n = group number)
 - add collaborator + (at some point me)
- see also the handout