IPS 2023

Eksamensnoter

Naomi Knudsen & Gustav Brandsen

Table of Contents

[Regular languages, automata, and lexical analysis 2](#_Toc137731825)

[Draw NFA from Regex (see section 1.3, p. 10-11 in book) 2](#_Toc137731826)

[From NFA to DFA (see section 1.5.2, p. 17 in book) 3](#_Toc137731827)

[Example 3](#_Toc137731828)

[Regular languages 7](#_Toc137731829)

[Regular expressions (Regex) 7](#_Toc137731830)

[Example 8](#_Toc137731831)

[Context-free grammars and syntax analysis 9](#_Toc137731832)

[Regex to context free grammar 9](#_Toc137731833)

[Ambiguous vs. Unambiguous 9](#_Toc137731834)

[Nullable nonterminal (see p. 56-57 in book) 10](#_Toc137731835)

[Example 10](#_Toc137731836)

[First set (see p. 56-58 in book) 12](#_Toc137731837)

[Example 12](#_Toc137731838)

[Follow set (see p. 61-62 in book) 13](#_Toc137731839)

[Example constraints 13](#_Toc137731840)

[Example least solution constraints 15](#_Toc137731841)

[Look-Ahead Sets and LL(1) Grammars 16](#_Toc137731842)

[Example (continued from earlier Nullable, First and Follow example) 16](#_Toc137731843)

[Recursive descent parser for Grammar (see p. 66 in book) 17](#_Toc137731844)

[Example: 17](#_Toc137731845)

[Interpretation and type checking 18](#_Toc137731846)

[Code generation 19](#_Toc137731847)

[IL code to MIPS/RISC-V 19](#_Toc137731848)

[Example 19](#_Toc137731849)

[RISC-V code generation in FASTO compiler 20](#_Toc137731850)

[Example: generate like in CodeGen.fs 20](#_Toc137731851)

[Liveness analysis and register allocation 22](#_Toc137731852)

[Succ, gen and kill sets for instructions 22](#_Toc137731853)

[Example 22](#_Toc137731854)

[In and out sets for instructions 23](#_Toc137731855)

[Example exercise 23](#_Toc137731856)

[Interference 24](#_Toc137731857)

# Regular languages, automata, and lexical analysis

## Draw NFA from Regex (see section 1.3, p. 10-11 in book)

A picture containing drawing, sketch, diagram, text

Description automatically generated

## 

## From NFA to DFA (see section 1.5.2, p. 17 in book)

* Find initial state {the nodes you can get to from start} node, only using epsilons.
* Then use
  + c is an example of a letter in NFA. Should use all the letters in the NFA.
  + the nodes you can get to following c from

, nodes following epsilon}

if new set then name new set else do nothing

* + Note: the empty set of NFA states is not an DFA state
* Repeat step 2 on new sets.
* Make table that shows the connections between the different states.
* Draw DFA from table.

### Example

A picture containing drawing, sketch, circle, line

Description automatically generated

With our incomplete being

With our incomplete now being

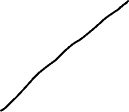
With our incomplete now being

With our incomplete now being

The constructions of is now done, and

1. From the move functions, we can create a table that shows how the different states are connected (the arrow meaning the initial state, and the stars meaning the final states):

|  |  |  |  |
| --- | --- | --- | --- |
|  | x | y | z |
| \* |  |  |  |
| \* |  |  |  |
| \* |  |  |  |
| \* |  |  |  |



## Regular languages

**Regular language:** a language that can be expressed by a regular expression.

* A set of strings is called a **language.**
* We use the notation L(s) to denote the language (i.e., set of strings) described by the regular expression s.

**To determine if regular language:**

* IMPORTANT: It has to be a finite size.
  + Uses *finite automaton* as explanation (see p. 7 in book)
* Look if language can be created by the difference between two languages.
* Can be explained, if you can create the regular expression.

### A picture containing text, screenshot, font, number Description automatically generatedA picture containing text, receipt, screenshot, font Description automatically generatedRegular expressions (Regex)

The following typical programming language elements can be described with regular expressions (see 1.1.2 Examples in book p. 5):

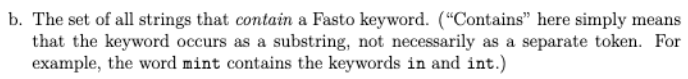
* Non-negative integer constants
* Keywords
* Variable names
* Integers
* Floats
* String constants

### Example

A picture containing text, font, white, receipt

Description automatically generatedA screen shot of a computer screen

Description automatically generated with low confidence

A screen shot of a computer screen

Description automatically generated with low confidence

A picture containing text, font, white

Description automatically generated

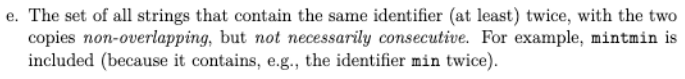
A screen shot of a computer screen

Description automatically generated with low confidence



A screen shot of a computer screen

Description automatically generated with low confidence



A screen shot of a computer screen

Description automatically generated with low confidence

# Context-free grammars and syntax analysis

* **Terminal:** The small letters and symbols in context free grammar (a and + in example). Also known as the symbols in the alphabet.
* **Nonterminal:** The capital letters in context free grammar (S, A and P in the example). Also known the name by which as set of string that is denoted.

**Example of context free grammar:**

A picture containing origami, white, design, typography

Description automatically generated

## Regex to context free grammar

A picture containing text, screenshot, font, number

Description automatically generated

## Ambiguous vs. Unambiguous

* **Ambiguous** grammar means that you can draw more than one tree and get the same result.
  + Grammar is always ambiguous if it has form:

A picture containing font, typography, text, graphics

Description automatically generated

* **Unambiguous** means that there is only one possible way of drawing the tree to get one result.

## Nullable nonterminal (see p. 56-57 in book)

* Only necessary to look at one elm, if has a ‘’(and) and one elm is false -> result=false
* Only necessary to look at one elm, if has a ‘’(or) and one elm is true -> result=true

A picture containing text, font, white, screenshot

Description automatically generated



A picture containing text, screenshot, font

Description automatically generated

### Example

-+f[HT) 
-+ 
д 

Note, should use 4th rule with:

First set (see p. 56-58 in book)

A picture containing text, font, screenshot, white

Description automatically generated



A picture containing text, screenshot, font, number

Description automatically generated

### Example

-+f[HT) 
-+ 
д 

Note, should use 4th rule with:

// is no

// taken from

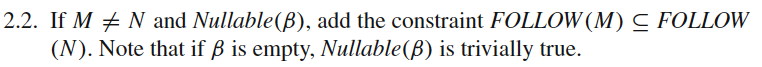
// is No

// is no

## Follow set (see p. 61-62 in book)

A picture containing text, font, screenshot, number

Description automatically generatedA picture containing text, screenshot, font, number

Description automatically generatedA picture containing text, screenshot, font, number

Description automatically generated

### Example constraints

-+f[HT) 
-+ 
д 

Set constraints for our grammar:

// For Follow rule, is no

Nonterminal, H:

// for istrue?yes

For Follow rule, is Nullabletrue? we calculate:

Answer no, so do nothing

Other nonterminal, T:

// For Follow rule, is no

Nonterminal, H:

// For follow rule, is yes

From follow rule:

Other nonterminal, T:

// is yes, but T=T, do nothing

// doesn’t matter, don’t add in table

Put in box:

|  |  |
| --- | --- |
| **Production** | **Constraints** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

### Example least solution constraints

Initialize Follow sets:

Taken from box. Put all constraint belonging to the follow in one (see p.62-63 in book)

Because we have:

// is already in else add

We can replace subset with equality:

## Look-Ahead Sets and LL(1) Grammars

To find LL(1) you need to have computations of:

* Nullable, First and Follow sets

A picture containing text, font, screenshot

Description automatically generatedA picture containing text, screenshot, font, number

Description automatically generated

### Example (continued from earlier Nullable, First and Follow example)

Look-Ahead Sets and LL(1):

// is

// is

// is

// is

// Look-Ahead Sets for H are disjoint (they don’t have any elements in common)

// is

// is

// Look-Ahead Sets for T are disjoint (they don’t have any elements in common)

## Recursive descent parser for Grammar (see p. 66 in book)

Notes:

* Try to do from other examples
* Use LL(1) from before
* Use ParseInsert\_nonterminal\_name() for nonterminals
* Use match(‘insert\_terminal\_name’) for terminals

### Example:

A black text on a white background

Description automatically generated with low confidence

From LL(1):

function ParseS() =

if input = ‘f’ then

ParseF(); match(‘$’)

else reportError()

function ParseT() =

if input = ‘+’ then

match(‘+’); ParseH(); ParseT()

else if input = ‘]’ then

(\* do nothing, just return \*) --- comes from

else reportError()

# Interpretation and type checking

Chapter 4 in text book

# Code generation

## IL code to MIPS/RISC-V

A picture containing text, receipt, document, number

Description automatically generated

### Example

:

A black text on a white background

Description automatically generated with low confidence

## RISC-V code generation in FASTO compiler

Look in CodeGen.fs and RiscV.fs

### Example: generate like in CodeGen.fs

A picture containing text, font, screenshot, document

Description automatically generated

| Expt (e1, e2, pos) ->

let t1 = newReg "expt\_L"

let t2 = newReg "expt\_R"

let t3 = newReg "zeroReg"

let t4 = newReg "resReg"

let code1 = compileExp e1 vtable t1

let code2 = compileExp e2 vtable t2

let code3 = compileExp (Constant (IntVal 0, pos)) vtable t3

let code4 = compileExp e1 vtable t4

let posLabel = newLab "positive"

let zeroLabel = newLab "zero"

let endLabel = newLab "end"

let loopLabel = newLab "loop"

code1 @ code3 @

[ LI (place, 0)

; BGE (t2, t3, zeroLabel) // bge is >=

; LA (Ra1, "m.IllegalExpt")

; JAL ("p.RuntimeError", [Ra1])

; LABEL zeroLabel

; BNE(t2, t3, posLabel) // bne is !=

; LI (place, 1)

; LABEL posLabel]

@ code2 @ code4 @

[ Label loopLabel

; BEQ t2,t3,endLabel

; MUL t4, t4, t1

; SUB t2, t2, 1

; J loopLabel

; Label endLabel

; MV (place, t4)]

**Exemplary answer:**

A person standing in front of a screen

Description automatically generated with medium confidence

# Liveness analysis and register allocation

## Succ, gen and kill sets for instructions

A picture containing text, font, screenshot, algebra

Description automatically generatedA picture containing text, screenshot, font

Description automatically generated

Label is always Ø for

### Example

A picture containing text, screenshot, font, number

Description automatically generated

## In and out sets for instructions

Note begin from bottom up and with :.

Use the first picture below to find and :

A picture containing text, font, white, typography

Description automatically generated

A picture containing text, font, white, screenshot

Description automatically generated

A picture containing text, screenshot, font

Description automatically generated

### Example exercise

## Interference

* If then there is no interference.
* If and then they interfere at not
* Write a note were x:=y

A picture containing text, screenshot, font

Description automatically generated

Create table:

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction |  |  | Interferes with |
|  |  |  |  |