IN4303 — Compiler Construction

Exam

April 04, 2011

---

1. Answer every question on its own page (A sheet has four pages).
2. Put your name, your student ID and the number of the question on top of each page.
3. If you need more than one sheet to answer a question, number your pages and state the overall number of pages for this question on the first page.
4. Take care of your time. This exam has 13 questions, for a total of 150 points. Try to answer a question worth 10 points in 10 minutes.
5. Keep your answers short and precise. Don't waste your time on essay writing.
6. Hand in the answers together with this form (including the questions).

Good luck!

Name: _____ Student ID: _____

Question	Points	Score
Language	15	
Formal grammars	10	
Syntax trees	10	
Term rewriting	10	
Static analysis	10	
Java Virtual Machine	10	
Polymorphism	10	
Calling conventions	10	
Liveness analysis	20	
Register allocation	20	
Garbage collection	5	
Lexical analysis	10	
LL parsing	10	
Total	150	

Question 1: Language (15 points)

(a) According to Edward Sapir, what is language? (5)

(b) What is the formal language $L(G)$ specified by a formal grammar G? Give a definition in English. (5)

(c) Where are aspects from Sapir's definition of language reflected in the definition of $L(G)$? Which aspects are not reflected at all? (5)

Question 2: Formal grammars (10 points)

Let G_1 be a formal grammar with nonterminal symbols S, and P, terminal symbols '**f**', '**x**', ',', '(' and ')', start symbol S, and the following production rules:

$$\begin{aligned} S &\rightarrow \mathbf{f}\ (\ P\ ) \\ P &\rightarrow \mathbf{x} \\ P &\rightarrow P\ ,\ \mathbf{x} \\ P &\rightarrow \mathbf{x}\ ,\ P \end{aligned}$$

(a) Is G_1 context-free? Why (not)? (1)

(b) Describe the language defined by G_1 in English. (2)

(c) Give a left-most derivation for the sentence $\mathbf{f}(\mathbf{x},\mathbf{x},\mathbf{x})$ according to G_1. (3)

(d) Use $\mathbf{f}(\mathbf{x},\mathbf{x})$ as an example to explain why G_1 is ambigous. (4)

Question 3: Syntax trees (10 points)

(a) Why do we need syntax trees when constructing compilers? (2)

(b) What are the fundamental differences between parse trees and abstract syntax trees? (3)

(c) How can we represent trees as terms? Illustrate your explanation with an example. (5)

Question 4: Term rewriting (10 points)

Stratego provides a strategy `inverse` with the following implementation:

```
inverse(|a): []     -> a
inverse(|a): [x|xs] -> <inverse(|[x|a])> xs
```

(a) Explain the semantics of `inverse` in English. (2)

(b) What is the result of applying `inverse(|[])` to the term `[1,2,3]`? Show each step of computation. (4)

(c) Based on the definition of `inverse`, explain how an accumulator is used. (4)

Question 5: Static analysis (10 points)

(a) How does static analysis contribute to a compiler w.r.t. its architecture? (2)

(b) Explain the generic approach of performing static analysis in rename/map/project/check phases. Use the example of type checking Mini Java. (8)

Question 6: Java Virtual Machine (10 points)

Execute the bytecode instructions of `A/main()V`, starting with an empty stack:

`A/main()V`

```
aload_0
bipush 5
iconst_4
isub
invokevirtual A/m(I)V
```

`A/m(I)V`

```
    goto l2
l1: iinc 1 -1
l2: iload_1
    ifne l1
    return
```

Hint: `iinc 1 -1`

```
iload_1
ldc -1
iadd
istore_1
```

The initial value of local variable 0 is `4242 4103`, pointing to an object of class `A`. Show stacks and local variables after each instruction.

Question 7: Polymorphism (10 points)

(a) Identify three examples of polymorphism in the following Java expression: (6)

```
"1" + ((2 + 4) + 3.5)
```

Which kinds of polymorphism do they represent? Explain the differences.

(b) Explain the difference between method overloading and method overriding. Illustrate your explanation with an example in Java. (4)

Question 8: Calling conventions (10 points)

A compiler translates a function call and a function body to the following instructions for a register-based machine:

function call

```
mov  AX 21
mov  DX 42
call _f@8
```

function body

```
push BP
mov  BP SP
add  AX DX
pop  BP
ret
```

(a) Which calling convention do these instructions follow? (1)

(b) What are the benefits of this calling convention? (1)

(c) How are calls handled by callers and by callees according to this convention? Base your explanation on the given instructions. (8)

Question 9: Liveness analysis (20 points)

Consider the following intermediate code:

```
     c := r3
     a := r1
     b := r2
     d := 0
     e := a
l1:  d := d + b
     e := e - 1
     if e > 0 goto l1
     r1 := d
     r3 := c
     return
```

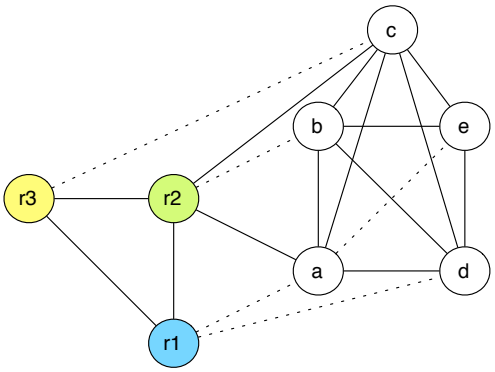
(a) Construct the control graph. (2)

(b) Calculate successor nodes, defined variables, and used variables for each node in the control graph. (3)

(c) Assume `r1` and `r3` to be live-out on the return instruction. Calculate live-ins and live-outs for each node in the control graph. Present your results in a table. (15)

Question 10: Register allocation (20 points)

You have to colour the following interference graph with three colours (`r1`, `r2`, and `r3` are already precoulored):



(a) Explain why the next step in the colouring has to be a *spill*. (5)

(b) Spill node `c` and continue the graph colouring until you can decide if this spill is an actual one. (12)

(c) Is node `c` an actual spill? (1)

(d) Perform the spill on the intermediate code from the previous question. (2)

Question 11: Garbage collection (5 points)

(a) Explain the general ideas behind garbage collection by reference counting. (3)

(b) What are the back draws of this strategy? (2)

Question 12: Lexical analysis (10 points)

Let G_2 be a formal grammar with nonterminal symbols S and D, terminal symbols '**b**', '**0**' and '**1**', start symbol S, and the following production rules:

$$\begin{aligned} S &\rightarrow \mathbf{b}\ D \\ D &\rightarrow \mathbf{0}\ D \\ D &\rightarrow \mathbf{1}\ D \\ D &\rightarrow \mathbf{0} \\ D &\rightarrow \mathbf{1} \end{aligned}$$

(a) Is G_2 regular? Why (not)? (1)

(b) Describe the language defined by G_2 in English. (2)

(c) Turn G_2 *systematically* into a finite automaton. (4)

(d) Use G_2 to generate a word with at least five letters. Show each derivation step. Use the automaton to recognise this word. Enumerate the states passed during the recognition. (3)

Question 13: LL parsing (10 points)

Let G_3 be a formal grammar with nonterminal symbols S, T, E and E', terminal symbols '**x**', '+' and '$', start symbol S, and the following production rules:

$$\begin{aligned} S &\rightarrow E\ \$ \\ E &\rightarrow T\ E' \\ E' &\rightarrow +\ T\ E' \\ E' &\rightarrow \\ T &\rightarrow \mathbf{x} \end{aligned}$$

(a) Construct an LL(0) parse table for the grammar. Calculate FIRST and FOLLOW sets as needed. (8)

(b) Use the parse table to recognise the sentence $\mathbf{x} + \mathbf{x}$. Show the stack and the remaining input after each step. (2)