Examination in Compilers, EDAN65

Department of Computer Science, Lund University

2020–10–28, 09.00-12.00

***Note!* Your exam will be marked only if you have completed all six programming lab assignments in advance.**

Start each solution (1, 2, 3, 4) on a separate sheet of paper. Write only on one side of each sheet. Write your name and signature on every sheet or paper. Write clearly and legibly. Try to find clear, readable solutions with meaningful names. Unnecessary complexity will result in point reduction.

The following documents may be used during the exam:

- *Reference manual for JastAdd2*
- *x86 Cheat Sheet*

Max points: 36
For grade 3: Min 18
For grade 4: Min 24
For grade 5: Min 30

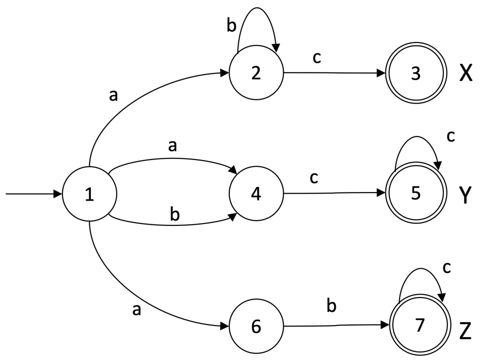
Good luck!

1 Lexical analysis

A token specification over the alphabet `{a, b, c}` includes definitions of tokens `X`, `Y`, and `Z`, where the usual disambiguation rules of rule priority and longest match apply:

```
X = ...
Y = ...
Z = ...
```

The following NFA has been derived from this specification:



a) Construct regular expressions for each of the tokens. (3p)

b) Construct a DFA that is equivalent to the NFA. Mark each DFA state with the state numbers from the corresponding states in the NFA. Mark each final state by the appropriate token. (5p)

2 Grammars

Consider the following context-free grammar G with start symbol `L` and terminal symbols `ID`, `"="`, `","`, `"("`, `")"`, and `$`.

```
L  → S $
S  → A
S  → C
A  → ID "=" E
C  → ID "(" EL ")"
E  → ID
E  → C
EL  → ϵ
EL  → E RL
RL  → ϵ
RL  → "," E RL
```

a) Consider the program `ID(ID,ID)$`. Prove that this program belongs to $L(G)$ by drawing a parse tree.

(3p)

b) The grammar G is a bit difficult to read because of its canonical form. Construct an equivalent grammar on EBNF form, where the `EL` and `RL` nonterminals have been eliminated.

Hint! If you don't solve this problem right away, it can help to draw parse trees for a couple of more examples to understand what language the grammar defines.

(4p)

c) G is not LL(1). Construct an equivalent grammar that is LL(1). The grammar should be on canonical form. (5p)

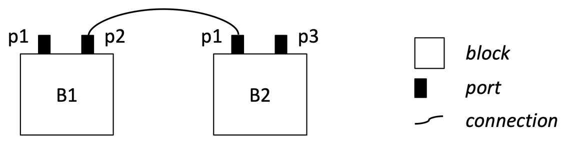
d) Consider again the program `ID(ID,ID)$`. Write down the sequence of steps that an LR parser would take for parsing this program according to the grammar G. For each step, show the stack contents, the remaining input, and the shift/reduce/accept action taken as in the following table:

stack	•	*remaining input*	*action*
	•	`ID "(" ID "," ID ")" $`	...
...	•	...	...

(5p)

3 Program analysis

A visual language has blocks with ports, and connections between ports, as illustrated below.



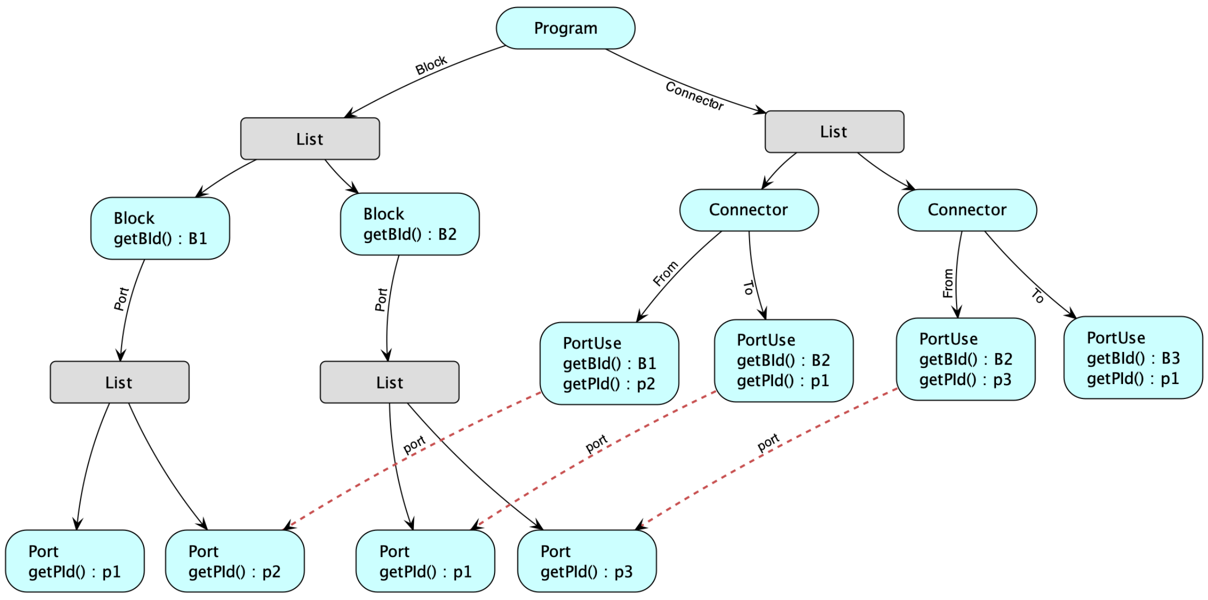
An abstract grammar for the language is shown below.

```
Program ::= Block* Connector*;
Block ::= <BId:String> Port*;
Port ::= <PId:String>;
Connector ::= From:PortUse To:PortUse;
PortUse ::= <BId:String> <PId:String>;
```

Blocks and ports have identifier names (`BId` and `PId`). A port name is local to the block, so different blocks can have ports with the same name.

A connection endpoint is represented by a `PortUse(BId, PId)`, where `BId` is a block name and `PId` is a port name on that block.

A reference attribute `PortUse.port()` refers to the appropriate `Port` object, if there is one, and is otherwise `null`. This is illustrated in the figure below. Note that the value of `port` in the `PortUse (B3, p1)` is not shown, because it is `null` (since there is no block with the name `B3`).



Note! In the problems below, use JastAdd syntax for the attributes. You may neither use `instanceof` nor the `getParent()` method.

a) Implement the following two attributes:

```
syn Block Program.localLookupBlock(String s);
syn Port Block.localLookupPort(String s);
```

The `localLookupBlock` attribute should look up a block of the name `s`, and return `null` if there is no such block.

The `localLookupPort` attribute should similarly look up a port of the name `s` in a given block, and return `null` if there is no such port in the block.

(4p)

b) Implement the attribute `port` discussed earlier:

```
syn Port PortUse.port();
```

In the implementation, use the two attributes defined in a), and introduce additional synthesized and/or inherited attributes as needed.

(5p)

4 Runtime systems

Explain what is meant by a *root pointer* and what a garbage collector can use root pointers for. (2p)