Prof. Dr. Peter Thiemann
Matthias Keil

Winter Semester 2016/2017

Compiler Construction

http://proglang.informatik.uni-freiburg.de/teaching/compilerbau/2016ws/

Exercise Sheet 4

1 From MiniJava to Piglet (15 Points)

For the next step in our project, Piglet will serve as an intermediate representation. Piglet's grammar and semantics is defined as follows.

Grammar

Program	::=	**MAIN** *StmtList* **END** *Procedure**
StmtList	::=	(Label? *Stmt*)*
Procedure	::=	*Label* [*IntegerLiteral*] *StmtExp*
Stmt	::=	**NOOP**
	|	**ERROR**
	|	**CJUMP** *Exp Label*
	|	**JUMP** *Label*
	|	**HSTORE** *Exp IntegerLiteral Exp*
	|	**HLOAD** *Temp Exp IntegerLiteral*
	|	**MOVE** *Temp Exp*
	|	**PRINT** *Exp*
StmtExp	::=	**BEGIN** *StmtList* **RETURN** *Exp* **END**
Exp	::=	*StmtExp*
	|	**CALL** *Exp* (*Exp**)
	|	**HALLOCATE** *Exp*
	|	*Operator Exp Exp*
	|	*Temp*
	|	*IntegerLiteral*
	|	*Label*
Operator	::=	**LT**
	|	**PLUS**
	|	**MINUS**
	|	**TIMES**
Temp	::=	**TEMP** *IntegerLiteral*
IntegerLiteral	::=	⟨integer literal⟩
Label	::=	⟨identifier⟩

Semantics

Procedures

If we have a procedure p

```
p [5] s
```

which takes five arguments, they are accessible in the procedure body s as **TEMP 0**, **TEMP 1**, ..., **TEMP 4**. Other temporaries (**TEMP 5** and higher) are treated as local variables within the procedure.

NOOP

Does nothing.

ERROR

Terminates the program execution with an error message.

CJUMP e l

If e evaluates to 1, then continue with the next statement, otherwise jump to label l.

LT

This is the $<$ operator. It returns 0 for false and 1 for true. You also need to use this operator to test whether a memory address is null (the value 0), by asking whether the address is less than 1.

HALLOCATE e

The expression e should evaluate to an integer denoting to the number of bytes of heapspace which is then allocated. The address of the newly allocated memory block is returned as the result. Both integers and memory addresses (i.e., pointers) have a size of 4 bytes, so in general you will allocate memory in multiples of 4.

HSTORE e_1 i e_2

Stores the value of expression e_2 at address e_1 with offset i. The expression e_1 evaluates to an address, the integer i is an offset from this address.

HLOAD t e i

Loads the value at address e with offset i into the temporary t. The expression e evaluates to an address.

The semantics of all other constructs should be obvious.

Project - Part 3

Implement an AST transformation from MiniJava to Piglet.

- You may assume that there are arbitrarily many temporaries available.
- On the homepage, you will find a project template with a parser for Piglet and a pretty-printer for Piglet, which you may want to use.
- You will also find parts of a type checker to build a class table. You are encouraged to adapt it to your needs.
- When implementing the transformation, make sure that you construct new object instances for each node. Otherwise the automatic management of child-parent links damages the tree structure, which leads to NullPointerExceptions.
- You may assume that you are only translating type-correct MiniJava programs.
- Make sure that all labels are unique. For procedure labels, you might want to take a composition of class name and corresponding method name.
- Do not forget to initialize newly allocated arrays and fields with zero.
- When transforming operations on arrays and objects, insert code to check at run-time for "index out of bounds" exceptions or "null pointer" exceptions.

2 From Piglet to Spiglet (10 Points)

Spiglet is a subset of Piglet. It simplifies the IR by "flattening" Piglet's tree structure to lists of (labeled) statements.

Grammar

Program	::=	**MAIN** *StmtList* **END** *Procedure**
StmtList	::=	(Label? *Stmt*)*
Procedure	::=	*Label* [*IntegerLiteral*] *StmtExp*
Stmt	::=	**NOOP**
	|	**ERROR**
	|	**CJUMP** *Temp Label*
	|	**JUMP** *Label*
	|	**HSTORE** *Temp IntegerLiteral Temp*
	|	**HLOAD** *Temp Temp IntegerLiteral*
	|	**MOVE** *Temp Exp*
	|	**PRINT** *SimpleExp*
StmtExp	::=	**BEGIN** *StmtList* **RETURN** *SimpleExp* **END**
Exp	|	**CALL** *SimpleExp* (*Temp**)
	|	**HALLOCATE** *SimpleExp*
	|	*Operator Temp SimpleExp*
	|	*SimpleExp*
Operator	::=	**LT**
	|	**PLUS**
	|	**MINUS**
	|	**TIMES**
SimpleExp	|	*Temp*
	|	*IntegerLiteral*
	|	*Label*
Temp	::=	**TEMP** *IntegerLiteral*
IntegerLiteral	::=	⟨integer literal⟩
Label	::=	⟨identifier⟩

The grammar for Spiglet differs from that of Piglet in two ways:

- A list of *Stmt*s followed by an expression is not an *Exp* in Spiglet.
- In many places, *Exp* is replaced with *SimpleExp* or **Temp**.

Project - Part 4

Implement a transformation from Piglet to Spiglet.

- On the homepage, you will find a project template with a parser for SPiglet, and also a pretty-printer for SPiglet, which you can use.
- When implementing the transformation, make sure that you construct new object instances for each node.

Submission

- Deadline: **12.01.2017, 12:00 (noon)**. Late submissions will not be accepted.
- Submit your solution to the subversion repository. Your submission will consist of one folder (exercise4) which includes your solution.
- Rewrite method `minijava.topiglet.MiniJavaToPigletTranslator.translateProgram` so that it calls your Piglet transformation for the given MiniJava AST.
- Rewrite method `piglet.tospiglet.PigletToSpigletTranslator.translateProgram` so that it calls your Spiglet transformation for the given Piglet AST.
- Your solution will consist of: 1. a zip file `J2P2S.zip` as generated by `ant submission` with the implementation of the two translators, and 2. a pdf `intermediate-<your name>.pdf` with a description of the two transformations.
- You are strongly encouraged to test your solution with the provided test data. Add test cases as you might think necessary. You need not submit your own test cases.
- The description must be limited to two pages. Submitting more than one page will lead to reduction in points.
- The description may be either German or English. Clear and understandable style is required.