Compiler Construction/Fall 2014/Homework 6

Eva Rose
evarose@cs.nyu.edu

Kristoffer Rose
krisrose@cs.nyu.edu

Assigned Thursday 10/9/2014, due Thursday 10/16/2014 at 8am

Reading Assignments

- Lecture 6 on 10/9/2014 (this homework): Type Analysis. Dragon book 6.3, 6.5 (20p).
- Lecture 7 on 10/16/2014: Intermediate code generation. Dragon book 6.1, 6.2, 6.4, 6.6, 6.9 (34p).
- *Midterm exam on 10/23/2014*.

Homework Assignments

The following assignments should be submitted for a maximum of 25 points.

The assignments in this homework are based on the "primitive types" of Java as defined by the following SDD fragment:

SYNTAX	SEMANTIC RULES
$T \to$ `short`	$T.type =$ `short`$; T.size = 2$
$\mid$ `int`	$T.type =$ `int`$; T.size = 4$
$\mid$ `float`	$T.type =$ `float`$; T.size = 8$
$\mid$ `byte`	$T.type =$ `byte`$; T.size = 1$

1 Casting

In the following questions we consider the following SDD for cast expressions.

SYNTAX	RULES
$S \to$ `(` T_1 `)` S_2	$S_2.e = S.e; S.type = T_1.type$
$\mid x$	$S.type = \text{LookupType}(S.e, x)$
$T \to \ldots$	*(as above)*

Question 1.1 (5 points)**.** Consider the cast expression `(float)(int)x` and assume an environment $S.e$ attribute from the context where the type of `x` is `byte`. Draw the abstract syntax tree and insert the value of the $type$ attribute for every node. Which attributes are synthesized and inherited?

Question 1.2 (10 points)**.** We would like to have a check of whether each cast is safe. Types follow the Java conventions for widening, thus `byte` widens to `short` widens to `int` widens to `float`. Extend the SDD such that it *checks whether casts are safe*, and invokes the special action `Warning()` when a cast is unsafe.

2 Records

Given this grammar, which represents the way plain record types can be encoded in Java:

$$
\begin{aligned}
R &\to \texttt{class}\ \textbf{id}\ \texttt{\{}\ Ds\ \texttt{\}} \\
Ds &\to D\ Ds \mid \epsilon \\
D &\to T\ \textbf{id}\ \texttt{;} \\
T &\to \ldots \quad \textit{(as above)}
\end{aligned}
$$

Hint. For this exercise, you might enjoy rereading the Dragon book §§6.3.5-6.

Question 2.1 (5 points)**.** Consider the record declaration

```
class MyRecord { int x; byte b; float y; }
```

Draw how you would allocate an instance of `MyRecord` in memory and explain what the offset (distance in bytes from the beginning of the record) of each record field is, based on the $T.size$ attribute for each type defined above.

Question 2.2 (5 points)**.** Write an SDD that for every D node in the abstract syntax tree defines a $D.offset$ attribute with the offset of that field's bytes from the beginning of the record. It uses the auxiliary inherited attribute $D.next$ on R and .

Question 2.3 (5 bonus points)**.** Modify your SDD so it works for a computer architecture where all individually addressable units must be aligned to an address, which is a multiplum of 8 bytes.