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).
• 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:

\[
\begin{array}{c|c}
\text{SYNTAX} & \text{SEMANTIC RULES} \\
T \rightarrow & T.type = \text{short}; T.size = 2 \\
| \text{int} & T.type = \text{int}; T.size = 4 \\
| \text{float} & T.type = \text{float}; T.size = 8 \\
| \text{byte} & T.type = \text{byte}; T.size = 1 \\
\end{array}
\]

1 Casting

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

\[
\begin{array}{c|c}
\text{SYNTAX} & \text{RULES} \\
S \rightarrow (T_1) S_2 & S_2.e = S.e; S.type = T_1.type \\
| x & S.type = \text{LookupType}(S.e, x) \\
T \rightarrow \ldots & (as above)
\end{array}
\]

Question 1.1 (5 points). Consider the cast expression \((\text{float})(\text{int})x\) 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:

\[
\begin{align*}
R & \rightarrow \text{class} \ id \ \{ \ Ds \} \\
Ds & \rightarrow D \ Ds \mid \epsilon \\
D & \rightarrow T \ id \ ; \\
T & \rightarrow \ldots \quad (\text{as above})
\end{align*}
\]

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

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

```java
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 multilpum of 8 bytes.