Static Type Inference for Jimple Local Variables

Etienne M. Gagnon (gagnon@sable.mcgill.ca)

September 20, 2000

If you want a description of the *type inference* algorithm used in Soot, as well as the supporting theory, please read the SAS 2000 paper entitled *Efficient Inference of Static Types for Java Bytecode* that you can find on the `http://www.sable.mcgill.ca/publications`, Sable publications web page.

1 Where is the actual code?

You can find the implementation of typing in the following packages:

- soot.jimple.toolkits.typing: classes for infering *reference types*.
- soot.jimple.toolkits.typing.integer: classes for infering *integer types* (like int, short, char, byte, boolean).

The actual algorithm is fully described on the paper referenced above.

2 Pitfalls

While programming Soot, you *should not* assume the original `.class` files were generated by a Java compiler. You may only assume verifiable bytecode.

This leads to somewhat weird sitations, as you may end up with the following typed `.jimple` code (generated from a handcoded *verifiable* class file):

```
private static boolean error1(int)
{
    int i0;
    boolean z0;
    i0 := @parameter0: int;
    z0 = (boolean) i0;
    return z0;
}
```

Notice the (**z0 = (boolean) i0;**) statement. It contains a type cast from *int* to *boolean*. This is not possible in Java. This type cast is simply discarded when generating class files, but should be transformed into:

```
z0 = (i0 != 0);
```

when decompiling back to Java source code.

The inverse is also possible, and should translate as follows:

```
i0 = (int) z0;
```

should become

```
i0 = z0 ? 1 : 0;
```

3 Questions?

If you have *carefully* read the paper referenced above, but still have more questions related to type inference, you may ask your questions on the Soot mailing-list (see information on the `http://www.sable.mcgill.ca/soot`, Soot web site).

History

- October 6, 2000 : Initial version.