BIGTYPES in ML

Bruce J. McAdam
bjm@dcs.ed.ac.uk
The University of Edinburgh
Supervisor: Stephen Gilmore stg@dcs.ed.ac.uk

Motivation

Despite the assistance which a typed functional programming language gives programmers, programming is still a process fraught with problems. In fact, sometimes problems are *caused* by programmers relying too heavily on correct typing as an indication of a program correctness[3].

BIGTYPES are a system devised to provide extra type information to programmers in languages using Hindley-Milner type inference (e.g. Standard ML[4]). By making extra information available, the programmer will discover more errors without the need to break apart a structured program in order to investigate local definitions.

The extra information in a BIGTYPE explains every use of a name in a program (e.g. declarations of new functions and uses of library functions). This helps the programmer by: letting them know if a library function had a different type from that expected; giving the type of imperative operations; reminding a programmer of the types in a program.

An Example

The BIGTYPE has two parts, a DEEPTYPE describing free names (e.g. those referring to libraries or built into the language) and a WIDETYPE describing new names (e.g. functions defined by the programmer). This example below shows a simple but incorrect program which could be debugged with the aid of BIGTYPES.

```
fun len (Cons(h,t)) = 1 + len(t)
  | len (Empty)     = 0
```

The BIGTYPE tells the programmer that `Empty` is a pattern (variable) with type `List`. This is not what was expected, it should be a *constructor* with type `List`.

In this case, the programmer has not realised that lists are 'fat-lists' implemented with three constructors: `Cons`, `App` and `Nil`. `Empty` is treated as matching lists with structure `App(a,b)` or `Nil`. This consequently gives incorrect answers for lists formed with the append constructor.

The BIGTYPE for this example is sufficient to show the programmer the error by telling him how the names have been interpreted.

ICFP '97 Amsterdam, ND

Implementation — W'

The BIGTYPES are derived using an extension of the standard algorithm for Hindley-Milner type-inference[2], W. We call this extended algorithm W'.

W has clauses for each of the type-derivation rules, W' retains these but also inserts into the BIGTYPE information which algorithm W would discard. Though the implementation was for ML, the close relationship between the semantic rules and W' mean that BIGTYPES can be implemented for any similarly typed language.

Algorithm W' has been added to the ML-Kit compiler[1] to demonstrate the use of BIGTYPES. There was no noticeable effect on run-time.

The implementation used a simple pretty-printer for displaying the results. Information could be presented better using a more sophisticated printing routine, for example an interactive printer which allows the programmer to pick which parts to display (so the programmer can focus on a particular area and not be overwhelmed by extra information).

The Poster

The poster illuminates the concept with examples of programs and their BIGTYPES. The algorithm is also given in detail with an explanation of its relationship to the semantic rules to show how easily it can be added to an existing type checker.

References

[1] Lars Birkedal, Nick Rothwell, Mads Tofte, and David N. Turner. *The ML Kit.* DIKU, March 1993.

[2] Luis Damas and Robin Milner. Principal type-schemes for functional programs. In *Ninth Annual Symposium on Principles of Programming Languages*, 1982.

[3] Stephen Gilmore. Designing for proof. In *Mathematics of Software Quality*, 1995.

[4] Robin Milner, Mads Tofte, and Robert Harper. *The Definition of Standard ML.* MIT Press, 1990.