Optimising Compilers

Consider the ML-like language given by abstract syntax

\[ e ::= x \mid n \mid \lambda x.e \mid e_1 e_2 \mid \text{if } e_1 \text{ then } e_2 \text{ else } e_3 \mid \text{store } e \text{ in } r \mid \text{load } e \text{ from } r \]

where \( x \) ranges over variable names, \( n \) over integer constants, and \( r \) over global names for disjoint areas of memory known as regions. This language allows values to be stored inside regions: \( \text{store } e \text{ in } r \) writes the value of \( e \) at some newly allocated memory location within region \( r \) and returns a pointer to this new location; the complementary operation \( \text{load } e \text{ from } r \) reads the value which \( e \) points to (provided \( e \) is indeed a pointer into region \( r \), otherwise the operation fails without accessing \( r \)).

Types have syntax

\[ \tau ::= \text{int} \mid \tau \rightarrow \tau \mid * \tau \text{ in } r \]

where \( * \tau \text{ in } r \) is the type of a pointer to a \( \tau \)-typed value stored in region \( r \). Note that there is no polymorphism and that \( \text{if-then-else} \) uses an integer (rather than boolean) condition.

(a) Give an effect system (also known as an annotated type system) in which we can derive judgements of the form

\[ \Gamma \vdash e : t, \varphi \]

where \( t \) is an extended form of \( \tau \) and \( \Gamma \) is a set of assumptions of the form \( x : t \). Effects \( \varphi \) are sets of region names representing the regions which \( e \) may need to access (i.e. write into or read from) during its execution.

[12 marks]

(b) Give types and effects for the following expressions, commenting briefly on any problems your scheme encounters and how they may be resolved. (Assume that \( r \) and \( s \) are region names, \( x \) is a variable of type \( * \text{int} \text{ in } r \), and \( p \) is a variable of type \( * \text{int} \text{ in } s \).)

\( (i) \ \ \ \ \ \text{if load } x \text{ from } r \text{ then } \text{store } 42 \text{ in } s \text{ else } p \) \hspace{1cm} [2 marks]

\( (ii) \ \lambda y. \text{if load } x \text{ from } r \text{ then } \text{store } y \text{ in } s \text{ else } p \) \hspace{1cm} [2 marks]

\( (iii) \ \ \text{if load } x \text{ from } r \text{ then } \lambda y. \text{store } y \text{ in } s \text{ else } \lambda y. p \) \hspace{1cm} [4 marks]