[TYPES] System F omega with (equi-)recursive types

Francois Pottier Francois.Pottier at inria.fr
Mon Jun 6 17:00:41 EDT 2011

- Previous message: [TYPES] System F omega with (equi-)recursive types
- Next message: [TYPES] System F omega with (equi-)recursive types
- **Messages sorted by:** [date] [thread] [subject] [author]

Hello,

On Mon, Jun 06, 2011 at 06:33:00PM +0200, Florian Lorenzen wrote:
> *I have been looking for publications studying the properties (type*
> *safety, decidability of type checking, type checking algorithm) of*
> *System F omega combined with (equi-)recursive types but have,*
> *unfortunately, been quite unsuccessful. I am interested in an extension*
> *of F omega where the X in (mu X. T) ranges over proper types of kind *.*

First, do you know about the papers that concern System F with equi-recursive types? There is a paper by Neal Glew and one by Nadji Gauthier and I on efficiently deciding type equality in the presence of \forall and \mu quantifiers. You can find references to earlier work in there (e.g. work by Colazzo and Ghelli). Once you know how to check type equality, the rest (proof of type soundness, type-checking algorithm) is just as in System F; the presence of recursive types has essentially no impact.

You say that you are interested in System F_omega, but you restrict the \mu quantifier to variables of kind *. It seems to me that the resulting system remains fairly simple, because beta-reduction and the unfolding of \mu's cannot interact: unfolding a \mu cannot cause a beta-redex to appear. Thus, you should be able to first bring every type to a normal form, as in Ssytem F_omega, then compare types for equality, as in System F with recursive types.

If you allow \mu at every kind, things become more interesting. There is an old paper by Marvin Solomon at POPL 1978 where it is showed that in the presence of parameterized, recursive types definitions (that is, in System F_omega with \mu at every kind), deciding whether two types are equal is equivalent to the then-open problem of deciding whether two deterministic pushdown automata are equivalent. This problem has been shown to be decidable since then, but no practical/tractable algorithm exists, as far as I understand.

The more recent paper by Stone and Schoonmaker is probably very relevant (I haven't read it).

The fun becomes greater yet if you allow not just recursive types at every kind, but recursive kinds to begin with. Then, the Y fixed point combinator can be defined at the level of types (it is well-kinded), so the notion of a recursive type need not be considered primitive: \mu X.T can be viewed as syntactic sugar for Y (\lambda X.T). I have considered a system based on this

somewhat exotic idea in my POPL 2011 paper. Up to some restrictions (which ensure that every recursive type definition is productive in a certain sense) the problem of determining whether two types are equal is semi-decidable. (I don't know if it is decidable.) I have implemented a type-checker, which is available online. The proof of type soundness (which appears in the long version of the paper) is completely standard: again, it is not affected by the presence of recursive types.

Basically, all you need in order to prove subject reduction is injectivity of type constructors (e.g., A -> B = C -> D implies A = C and B = D), and all you need in order to prove progress is disjointness of type constructors (e.g., A -> B != C * D). These properties are usually easily shown to hold, even in the presence of recursive types. Oh, and you also need reflexivity and transitivity of type equality, which (depending on your definition of equality and depending on how non-trivial an equational theory you want to bake in) could be somewhat tricky to prove in the presence of recursive types.

It is very likely that there are important references concerning recursive types that I have not mentioned here; I apologize for that. I hope nevertheless that this is useful,

--
François Pottier
Francois.Pottier at inria.fr
http://gallium.inria.fr/~fpottier/

- Previous message: [TYPES] System F omega with (equi-)recursive types
- Next message: [TYPES] System F omega with (equi-)recursive types
- **Messages sorted by:** [date] [thread] [subject] [author]

More information about the Types-list mailing list