1 Markers for FPH

The following extension for FPH also allows impredicative instantiation in the expression of a binding, if the poly type to instantiate to appears only in derivations where all instantiations have been done. In this case, the impredicative instantiation cannot result in an ambiguous type for the binding, as these ambiguities only arise when some of the instantiations do not have to be performed.

In order to do this, we indicate that some of the polymorphic types are safe, by using a specially marked universal quantifier $\forall_*$:

$$\begin{array}{lll} \sigma & ::= & \forall \overline{a}.\rho \mid \forall_* \overline{a}.\rho \\ \sigma' & ::= & \forall \overline{a}.\rho' \mid \forall_* \overline{a}.\rho' \end{array}$$

These marked universal quantifiers play a role during unboxing. Since we allow impredicative instantiation for these "safe" types, we can move the box over the qualifier, as indicated by the following extra unboxing rule:

$$\frac{\boxed{\rho} \sqsubseteq \rho'}{\boxed{\forall_* \overline{a}.\rho} \sqsubseteq \forall_* \overline{a}.\rho'}\,\text{LIFT}$$

These marked universal quantifiers (short: *markers*) have to be introduced somewhere. This is done by an extra rule for expressions:

$$\frac{\Gamma \vdash e : \rho'}{\Gamma \vdash e : m(\rho')}\,\text{MARK}$$

This rule applies the type of the expression to the relation m, but only if it is a ρ'-type. This is *essential*, as we only allow universal quantifiers to be marked safe if in the derivation that produces it, all instantiations have been performed. Since a ρ' type does not have any outermost quantifiers, this means that no instantiation can be performed anymore. Then it is safe to to use the relation m:

$$m(\sigma') = \begin{cases} \boxed{\sigma} & \text{if } \sigma' = \boxed{\sigma} \\ a & \text{if } \sigma' = a \\ m(\sigma'_1) \to m(\sigma'_2) & \text{if } \sigma' = \sigma'_1 \to \sigma'_2 \\ \forall \overline{a}.m(\sigma'') \text{ or } \forall_* \overline{a}.m(\sigma'') & \text{if } \sigma' = \forall \overline{a}.\sigma'' \end{cases}$$

It traverses the type structure recursively. When it encounters a universal quantifier, it *may* change it to a marked quantifier. This is a declarative aspect: later in the derivation process it may turn out that that its only possible to construct a derivation if the quantifier was not marked. The relation does not traverse into boxes either, as the contents of a box must be guessed properly directly during instantiation. This subtlety is shown in one of the examples later.

Finally, in order to get rid of the markers, there is the last extra rule for expressions:

$$\frac{\Gamma \vdash e : \sigma'}{\Gamma \vdash e : u(\sigma')}\,\text{UNMARK}$$

Again, it delegates to the u function, defined as follows:

$$u(\sigma') = \begin{cases} \boxed{\sigma} & \text{if } \sigma' = \boxed{\sigma} \\ a & \text{if } \sigma' = a \\ u(\sigma'_1) \to u(\sigma'_2) & \text{if } \sigma' = \sigma'_1 \to \sigma'_2 \\ \forall \overline{a}.u(\sigma'') & \text{if } \sigma' = \forall \overline{a}.\sigma'' \text{ or } \sigma' = \forall_* \overline{a}.\sigma'' \end{cases}$$

The u function recursively traverses the type and turns all marked universal quantifiers in plain ones.

Finally, we demand that types in the environment do not have these markers. The reason for this is that it ensures that markers that matter can only occur by means of the mark rule. To demand this, we introduce the type σ_Γ that does not have the marked universal quantifiers, and change the definition of the environment Γ accordingly:

$$\begin{array}{lll} \Gamma & ::= & \Gamma,\ (x : \sigma_\Gamma) \ \mid \ \cdot \\ \sigma_\Gamma & ::= & \forall \overline{a}.\rho \end{array}$$

The intended use is that the mark rule is applied just after the inst rule, and the unmark rule just at the end of typing a binding. In fact, instead of separate rules, the m and u can be moved to the instantiation and let rule respectively.

2 Inference for the $runST$ example

It is possible to type the runST example with the proposed extension. Assume that the environment is:

$$\Gamma = ((\$) \ : \forall ab.(a \to b) \to a \to b), (runST : \forall c.(\forall s.STs\,c) \to c), \cdot$$

Then this is the derivation (cut into pieces otherwise it doesn't fit):

$$\dfrac{\dfrac{\dfrac{\dfrac{\Delta_{\text{apply}} \qquad \Delta_{\text{runST}}}{\Gamma \vdash (\$)\ runST : \boxed{\forall_* s.STs\,r} \to r}\ \text{APP} \qquad \Delta_{\text{unbox}}}{\Gamma \vdash (\$)\ runST : (\forall_* s.STs\,r) \to r}\ \text{SUBS}}{\Gamma \vdash (\$)\ runST : (\forall s.STs\,r) \to r \qquad \{r\}\#\Gamma}\ \text{GEN}}{\Gamma \vdash (\$)\ runST : \forall r.(\forall s.STs\,r) \to r}\ \text{UNMARK}$$

The result of the derivation is a non-boxy type without any marked universal quantifiers. The marker is introduced in order to get rid of the box, as given in the Δ_{unbox} part of the derivation:

$$\dfrac{\dfrac{\dfrac{}{\boxed{STs\,r} \sqsubseteq STs\,r}\ \text{TBOX}}{\boxed{\forall_* s.STs\,r} \sqsubseteq \forall_* s.STs\,r}\ \text{LIFT} \qquad \dfrac{}{r \sqsubseteq r}\ \text{REFL}}{\boxed{\forall_* s.STs\,r} \to r \sqsubseteq (\forall_* s.STs\,r) \to r}\ \text{CONG}$$

To get this marker there, it has to be guessed during instantiation of the type for (\$). Since this is just a normal σ-type, this is allowed, as long as it is wrapped in a box. The Δ_{apply} part of the derivation shows this:

$$\dfrac{\dfrac{((\$) \ : \ \forall ab.(a \to b) \to a \to b) \in \Gamma}{\Gamma \vdash (\$) \ : \forall ab.(a \to b) \to a \to b}\ \textsc{var}}{\Gamma \vdash (\$) \ : (\boxed{\forall_* s.STs\ r} \to r) \to \boxed{\forall_* s.STs\ r} \to r}\ \textsc{inst}$$

However, this marker results in the following obligation: the type of *runST* now also needs to have this marker. Since the environment does not contain any markers, the only way to get this is by using the mark rule. The mark rule requires the type of *runST* to be a ρ'-type, which requires that all instantiations have been performed in the Δ_{runST} derivation:

$$\dfrac{\dfrac{\dfrac{(runST \ : \ \forall c.(\forall s.STs\ c) \to c) \in \Gamma}{\Gamma \vdash runST : \forall c.(\forall s.STs\ c) \to c}\ \textsc{var}}{\Gamma \vdash runST : (\forall s.STs\ r) \to r}\ \textsc{inst}}{\Gamma \vdash runST : (\forall_* s.STs\ r) \to r}\ \textsc{mark}$$

3 Inference for higher-ranked *choose id*

A non-boxy rank-1 type can be inferred for *choose id*. A higher-rank type can be inferred as well, but then it must contain a box. Is this still the case with this extension? Here is a snipped of the derivation:

$$\dfrac{\dfrac{\dfrac{\cdots}{choose : \boxed{\forall a.a \to a} \to \boxed{\forall a.a \to a} \to \boxed{\forall a.a \to a}}\ \textsc{Inst} \qquad \dfrac{\cdots}{id : \forall a.a \to a}\ \textsc{var}}{choose\ id : \boxed{\forall a.a \to a} \to \boxed{\forall a.a \to a}}\ \textsc{App}$$

In order to abuse the extension to get rid of these boxes, we could try to get a marked universal quantifier. However, this is not possible. The mark rule cannot be applied to the type of *id*, because the type is not in ρ'. Also, the universal quantifiers after the application cannot be changed via the mark rule, because the m relation does not traverse into boxes.

4 A non-inference

The following example is taken from the FPH paper, and should end up with a box in the type. This is the case for the FPH type system, but what about the extension? In this attempt, we try to use the mark rule to get a marked universal quantifier, such that we can later use the extra unbox rule to remove the box. However, it will turn out that this is not possible.

$$
\begin{array}{ll}
f & :: \forall a.a \to [a] \to [a] \\
ids & :: [\forall a.a \to a] \\
id & :: \forall a.a \to a
\end{array}
$$

The following is an attempt to construct a derivation:

$$
\dfrac{\dfrac{\dfrac{\dfrac{\cdots}{f : \boxdot \to [\boxdot] \to [\boxdot]}\,\textsc{inst} \qquad \dfrac{\cdots}{id : \forall a.\mathbf{X}}\,\textsc{var} \qquad \dfrac{\dfrac{\cdots}{ids : [\forall a.a \to a]}\,\textsc{var}}{ids : [\forall_* a.a \to a]}\,\textsc{mark}}{f\ id\ ids : [\boxed{\forall_* a.a \to a}]}\,\textsc{app}}{f\ id\ ids : [\forall_* a.a \to a]}\,\textsc{subs}}{f\ id\ ids : [\forall a.a \to a]}\,\textsc{mark}
$$

This attempt fails because the type of *id* has to have a marked universal quantifier too. This is not possible, because the mark rule does not apply here, since the type of id is not in ρ'.

However, note that it is possible to construct the derivation with the box present. Either by omitting the mark rule, or by choosing not to change the quantifier in the m relation. This is essential, because we would for example want to be able to type $const' 3' (f\ id\ ids)$.

5 Implementation

Although the proposed extension is seemingly only a small change, it is likely that actually implementing this extension is somewhat challenging. The difficulty is caused by the declarative nature of the m relation. It's implementation needs to deal with the problem that during type inferencing not all type information is known, that not all boxes have been resolved, and that instantiation may take place later.