1 Heap-based evaluation

Consider the case of the proof for the rule regarding singleton intervals below:

$$\frac{\Gamma \vdash A : B..C}{\Gamma \vdash A : A..A}\ \textsc{K-Sing}$$

By the inductive hypothesis, we have that there exists some α, n such that

$$H \vdash A \Downarrow^n \alpha$$

and

$$\alpha \in [\![B..C]\!]_\Gamma$$

Our obligation is to show that $\alpha \in [\![A..A]\!]_\Gamma$.

The obvious way to do this is to show that $\Gamma \vdash \alpha \leq A$. However, I don't think this is possible.

Let

- $A = X$ (e.g., a variable)
- B, C and τ
- Γ be a context containing $X : B..C$ such that $\Gamma \vdash \tau : B..C$
- H be a heap containing $X = \tau$

Then $H \vdash A \Downarrow^0 \tau$, and we know $\tau \in B..C$. However, there's no way to show that $\tau \leq X$, because the subtyping rules never interact directly with variables, except in the reflexivity rule $\Gamma \vdash X \leq X$. Instead, the β subtyping rules use substitution.

1.1 Ideas

1.1.1 Heap regularity

One idea might be to include a premise of $\Gamma \vdash X \leq \tau$ in the definition of consistent environments $\Gamma \Vdash H$. However, this only defers the problem, as it now becomes impossible to ever construct a proof that $\Gamma \Vdash H$.

1.1.2 Amending the subtyping rules

Another idea would be to adjust the subtyping rules, then show that Stucki's original formulation is equivalent to ours. However, it's not clear to me what changes to actually make (or that doing so would be easier than simply fixing the proof to use substitution rather than a heap in the first place). Changing the subtyping rules to themselves use a heap would work, but feels very inelegant.

2 Totality (RESOLVED)

My proof uses a denotation function mapping kinds to the set of normalizing type expressions of that kind as follows:

$$
\begin{aligned}
[\![*]\!]_\Gamma &= \{\langle H, \tau\rangle\} \\
[\![A..B]\!]_\Gamma &= \{\langle H, \tau\rangle \mid \Gamma \vdash A \le \tau : *, \Gamma \vdash \tau \le B : *, \} \\
[\![\Pi(X : J).K]\!]_\Gamma &= \{\langle H, \lambda(X : J).A\rangle \mid \forall \tau_x \in [\![J]\!]_\Gamma . \langle H(X \mapsto \tau_x), A\rangle \in \mathscr{E}[\![K[\tau_x/X]]\!]_\Gamma\} \\
\mathscr{E}[\![K]\!]_\Gamma &= \{\langle H, A\rangle \mid \exists \tau . H \vdash A \Downarrow \tau \wedge \langle H, \tau\rangle \in [\![K]\!]_\Gamma\}
\end{aligned}
$$

so strong normalization can be stated as

$$\frac{\Gamma \vdash A : K \qquad \Gamma \models H}{\langle H, A\rangle \in \mathscr{E}[\![K]\!]_\Gamma}\ \textsc{Strong-Normalize}$$

which is proven by induction on the judgment $\Gamma \vdash A : K$.

3 The Issue

Consider the variable case $A = X$, with the relevant rules being

$$\frac{\Gamma\ \texttt{ctx} \qquad \Gamma(X) = K}{\Gamma \vdash X : K}\ \textsc{K-Var} \qquad\qquad \frac{H(X) = \tau}{H \vdash X \Downarrow^0 \tau}\ \textsc{Eval-var}$$

To show that $\langle H, X\rangle \in [\![K]\!]_\Gamma$, we need to show $\tau \in [\![K]\!]_\Gamma$. Even if we assume a strict call-by-value semantics, we only know that $\Gamma \vdash \tau : K$ and $H \vdash \tau\ \texttt{val}$.

One solution would be to bake in $[\![\cdot]\!]_\Gamma$ to the definition of the heap H, but that feels like a mistake.

So we need to prove

$$\frac{\Gamma \vdash \tau : K \qquad H \vdash \tau\ \texttt{val}}{\langle H, \tau\rangle \in [\![K]\!]_\Gamma}\ \textsc{Denot-Spec}$$

Ideally, this would be trivial, as the entire point of $[\![\cdot]\!]_\Gamma$ is for this to be true, and it is a mostly straightforward induction on the judgment $H \vdash \tau\ \texttt{val}$. However, we run into an issue in the case that $\tau = \lambda(X : K).A$, with relevant rules/clauses:

$$\frac{}{H \vdash \lambda(X : K).A\ \texttt{val}}$$

$$[\![\Pi(X : J).K]\!]_\Gamma = \{\langle H, \lambda(X : J).A\rangle \mid \forall \tau_x \in [\![J]\!]_\Gamma . \langle H(X \mapsto \tau_x), A\rangle \in$$

$$\mathscr{E}[\![K[\tau_x/X]]\!]_\Gamma\}$$

The issue arises when showing that $\langle H(X \mapsto \tau_x), A\rangle \in \mathscr{E}[\![K]\!]_\Gamma$. Intuitively, this is simple; we just use the main strong normalization proof. But now we're doing some mutual induction that I'm not convinced is well-founded – we invoke Strong-Normalize with A (which is smaller than $\lambda(X : K).A$) and $H, X = \tau_x$ (which is larger than H). But then the proof of Strong-Normalize invokes Denot-Spec with an *arbitrary* τ in the var case.

My best guess is that I need to somehow use some measure of the heap size and term size combined, but I don't know if that actually works – τ can be arbitrarily large, and we don't shrink the heap when we pass it back to Denot-Spec.

3.1 Resolution

Change the definition of $\Gamma \Vdash H$ to include $\tau \in \mathscr{E}[\![K]\!]_\Gamma$.