In a world

Conor McBride

December 31, 2014

Abstract

In a world where modelling information flow (to allow or prevent it) is becoming increasingly critical, type systems which account only for *what* stuff consists of, not *where* it is, *who* knows it, *when* it is available, or other such considerations of time and space, are sadly separated from key concerns. Dependent type systems generally require a Kripke semantics to account for their contextualisation, so it might as well be an interesting Kripke semantics. There should probably be a sentence between the first two sentences and this sentence should probably not exist. There should probably be a sentence after this sentence but it does not exist yet.

1 Introduction

I'll write this when I know what I'm talking about.

2 Worlds and Quantifiers

Different things exist in different *worlds*. Your computer has different data from my computer. The data available to the typechecker of my program are different from the data available to my program at runtime. Because I am a dependently typed programmer, abstract approximants to all runtime data are available to the typechecker, but I might like to ensure that information flow from typechecker to runtime is restricted, in order to obtain healthy *erasure* properties and acknowledge the fact that quantities which exist only to sustain specifications need not occupy memory once the hard work starts.

Accordingly, let W be a preorder of 'worlds' u, v, w with arrows $\twoheadrightarrow$. (Of course, W is thus a category, and I shall prefer to use categorical terminology.) These worlds represent different places where data can exist and computation can happen. For example[1], $W = \{\bullet, *\}$ with $\bullet \twoheadrightarrow *$ is the simplest nontrivial such system, modelling has runtime data on 'earth', $\bullet$, and typechecking in 'heaven', $*$, with information flowing only upwards.

The preorder structure tells us between which worlds information can flow without explicit communication: that is, which worlds are *in scope* for which other worlds. Worlds will show up in judgments: everything we construct will be constructed in a particular world. Worlds will show up in contexts: every variable we assume must be somewhere, as well as some kind of thing. Context entries look like $u\,x{:}S$, where u is the world x comes from and S is its type. The key rule is this:

$$\frac{\Gamma, u\,x{:}S, \Delta \vdash \textsc{valid} \quad u \twoheadrightarrow w}{\Gamma, u\,x{:}S, \Delta \vdash w\,x \in S}$$

[1] I use the word 'example', despite my preference for categorical terminology.

That is, x can be used for constructions in any world w to which x may travel from its 'home' world u.

Many type systems have some implicit separation of worlds baked into them, and it is very often the case that the worlds something may inhabit are closely related with the type of thing it is. It is liberating to discard this correlation and treat *what* and *where* as separable concerns.

An *upset* of W is a subset $U \subseteq W$ such that whenever $u \in U$ and $u \twoheadrightarrow w$, $w \in U$. Let $\bigstar$ be an upset of W, namely the subset of worlds in which the construction of types is permitted. If we want to construct types at all, we should probably ensure that $\bigstar$ is nonempty. The extent to which a system is 'dependently typed' is in some way characterized by the information flow to worlds in $\bigstar$. In our example system, take $\bigstar = \{*\} \subseteq \{\bullet, *\} = W$. This system has 'full dependent types', because all things are in scope for typechecking, but supports erasure in that information which exists *only* for typechecking purposes cannot flow to runtime.

We shall need a monotone operator $\bigstar\cdot : W \to \bigstar$ which maps each world w to the world in which the types of w-constructions are constructed.

Let us consider functions and their types. There is a set Q of *quantifiers*, characterising the varieties of functional abstraction which are possible. Each quantifiers acts as an endofunctor on worlds, qw: If $u \twoheadrightarrow w$, then $qu \twoheadrightarrow qw$. This action explains how functions change their usage as they move between worlds. We should expect that addition at runtime demands two runtime numbers, but we should be sorely disappointed if the addition function, promoted to the type level, could not also be used for numbers existing only at the type level. The key rules are, roughly speaking:

$$\frac{\Gamma, qw\, x{:}S \vdash w\, t : T}{\Gamma \vdash w\, \lambda x \mapsto t : (q\, x{:}S) \to T} \qquad \frac{\Gamma \vdash w\, f : (q\, x{:}S) \to T \qquad \Gamma \vdash qw\, s : S}{\Gamma \vdash w\, f\, s : T[s{:}S/x]}$$

In our example system, let us take

$$Q = \{\cap, \Pi\} \qquad \begin{array}{c|cc} qw & \bullet & * \\ \hline \Pi & \bullet & * \\ \cap & * & * \end{array}$$

and thus deliver that all functions used in types take type-level values.

Now, in the course of checking an argument which itself is an application, the action of quantifiers on worlds is *iterated*. Accordingly, the worlds we can reach from a given w by iterated quantification or type formation form a subcategory $\overline{Q}\, w$ of W, 'worlds *local* to w', whose objects are given by words in $\overline{q} \in (Q \cup \{\bigstar\})^*$, and whose arrows $\overline{Q}\, w(\overline{q}, \overline{q'})$ are exactly the arrows $\overline{q}w \twoheadrightarrow \overline{q'}w$.

The key additional requirement which makes typing survive transmission between worlds is that $\overline{Q}$ is itself a functor from W to preorders. Whenever $v \twoheadrightarrow w$, we need to map $\overline{Q}\, v(\overline{q}, \overline{q'})$ to $\overline{Q}\, w(\overline{q}, \overline{q'})$, so $\overline{q}v \twoheadrightarrow \overline{q'}v$ implies $\overline{q}w \twoheadrightarrow \overline{q'}w$. That is, embedding between worlds preserves their *local* preorder structure.

3 Syntax and Computation

Let us develop a more formal account of the situation. I propose to adopt a bidirectional approach, explicitly separating those syntactic forms from which types are synthesized and those for which

types proposed in advance are checked.

$$\begin{array}{rcl} s,t,R,S,T,U & ::= & \underline{e} \\ & | & \mathsf{Type} \\ & | & (q\,x{:}S) \to T \\ & | & \lambda x \mapsto t \\ e,f,E,F & ::= & (s{:}S) \\ & | & x \\ & | & f\ s \end{array}$$

Observe that worlds in W occur nowhere in the syntax of terms. Quantifiers from Q, interpreted differently in different worlds, are the things we see.

Substitution is defined as usual, but note that it makes sense only to substitute $[e/x]$, not $[s/x]$. Moreover, λ is unencumbered by type annotation, but that any β-redex necessarily requires a type annotation. Reduction is based on these two rules:

$$(\lambda x \mapsto t : (q\,x{:}S) \to T)\ s \rightsquigarrow t[(s{:}S)/x] \qquad \underline{(s{:}S)} \rightsquigarrow s$$

Ultimately, we can eliminate type annotation from the system and insist on computation in larger steps, by *hereditary* substitution. However, for now, let us work with a small-step system, in order to develop the rest of the metatheoretical apparatus without requiring the machinery of normalization. Indeed, to put my money where my mouth is and also to reduce digression on the topic of cumulative hierarchies (about which I have too much to say to say it here), I shall work in the pleasingly naïve world of Type-in-Type.

Computation, $t \twoheadrightarrow t'$, is the reflexive-transitive contextual closure of $\rightsquigarrow$. It is straightforward to show that $\twoheadrightarrow$ is stable under substitution. Moreover, as the contraction schemes contain no critical pairs, a Takahashi-style 'parallel reduction' argument shows that $\twoheadrightarrow$ has the 'diamond property': if $R \twoheadrightarrow S$ and $R \twoheadrightarrow T$, then for some R', $S \twoheadrightarrow R'$ and $T \twoheadrightarrow R'$.

4 Typing Rules

Typing contexts look like this

$$\Gamma, \Delta \ ::= \ \cdot \mid \Gamma, w\,x{:}S$$

Let us have judgment forms

$$\Gamma \vdash \qquad \Gamma \vdash w\ T \ni t \qquad \Gamma \vdash w\ e \in S$$

Context validity is as follows.

$$\frac{}{\cdot \vdash} \qquad \frac{\Gamma \vdash w\ \mathsf{Type} \ni S \quad w \in \bigstar}{\Gamma, w\,x{:}S \vdash}$$

I write $S \leq T$ for the relation 'an S can be used wherever a T is needed'. Locally, I define this by the rather symmetrical property of computational joinability, $\exists U.\, S \twoheadrightarrow U \wedge T \twoheadrightarrow U$, but we can imagine deploying standard methods to add η-rules. In a cumulative setting, symmetry would evaporate, but a directed inclusion is enough.

Type checking works like this:

$$\textbf{(typ)}\ \frac{w \in \bigstar \quad \Gamma \vdash}{\Gamma \vdash w\ \mathsf{Type} \ni \mathsf{Type}} \qquad \textbf{(fun)}\ \frac{w \in \bigstar \quad \Gamma, qw\,x{:}S \vdash w\ \mathsf{Type} \ni T}{\Gamma \vdash w\ \mathsf{Type} \ni (q\,x{:}S) \to T}$$

$$\textbf{(syn)}\ \frac{\Gamma \vdash w\ e \in S \quad S \leq T}{\Gamma \vdash w\ T \ni \underline{e}} \qquad \textbf{(abs)}\ \frac{\Gamma, qw\,x{:}S \vdash w\ T \ni t}{\Gamma \vdash w\ (q\,x{:}S) \to T \ni \lambda x \mapsto t}$$

Type synthesis works like this:

$$\textbf{(chk)}\ \frac{\Gamma \vdash \bigstar w\ \mathsf{Type} \ni S \quad S \twoheadrightarrow T \quad \Gamma \vdash w\ T \ni s}{\Gamma \vdash w\ (s{:}S) \in S} \qquad \textbf{(var)}\ \frac{\Gamma, u\,x{:}S, \Delta \vdash \qquad u \rightarrowtriangle w}{\Gamma, u\,x{:}S, \Delta \vdash w\ x \in S}$$

$$\textbf{(app)}\ \frac{\Gamma \vdash w\ f \in R \quad R \twoheadrightarrow (q\,x{:}S) \to T \quad \Gamma \vdash qw\ S \ni s}{\Gamma \vdash w\ f\ s \in T[(s{:}S)/x]}$$

Context validity amounts to this:

$$\textbf{(emp)}\ \frac{}{\vdash} \qquad \textbf{(ext)}\ \frac{\Gamma \vdash \bigstar u\ \mathsf{Type} \ni S}{\Gamma, u\,x{:}S \vdash}$$

It is my habit to let the metavariable J range over anything which may extend a context to a judgment and to write $\Gamma \vdash J$ for that judgment. E.g., if $J = \Delta \vdash$, $\Gamma \vdash J$ is the judgment $\Gamma, \Delta \vdash$.

The following sanity property is admissible, by straightforward induction on derivations.

$$\frac{\Gamma \vdash J}{\Gamma \vdash}$$

With the usual remarks about variable freshness, we also have thinning admissible,

$$\frac{\Gamma \vdash w\ \mathsf{Type} \ni S \quad w \in \bigstar \quad \Gamma \vdash J}{\Gamma, u\,x{:}S \vdash J}$$

and thence a chance to establish stability under substitution by the usual method of glorified substitution

$$\frac{\Gamma, u\,x{:}S \vdash J \quad S \twoheadrightarrow T \quad \Gamma \vdash u\ T \ni s}{\Gamma \vdash J[(s{:}S)/x]}$$

simply because wherever in a derivation the variable rule is used, we shall be able to replace it by a suitably thinned synthesis for $(s : S) \in S$. The catch, however, is that the variable can be transported up the world preorder. We must show that its replacement can be correspondingly transported.

It might be nice to show the following admissible (and indeed they are)

$$\frac{\Gamma \vdash v\ T \ni t \quad v \rightarrowtriangle w}{\Gamma \vdash w\ T \ni t} \qquad \frac{\Gamma \vdash v\ e \in S \quad v \rightarrowtriangle w}{\Gamma \vdash w\ e \in S}$$

but the obvious strategy of induction on derivations will fail as soon as we try to go under a binder, putting vs and ws into the context.

We need a suitable generalization to make the induction go through. The key is to consider judgments-with-world-holes, $J(\cdot)$ where some quantifier-word-applied $\overline{q}\cdot$ can stand as a world: in particular, let us insist that the target world of a typing judgment is of that form. We can form a proper judgment $J(w)$ by substituting w for the $\cdot$ and computing the iterated quantifier action.

Lemma (world subsumption). The following is admissible.

$$\frac{\Gamma \vdash J(v) \quad v \rightarrowtriangle w}{\Gamma \vdash J(w)}$$

Proof. Induction on derivations. Let us proceed rule by rule.

emp $J(\cdot)$ is $\vdash$, so the assumption and conclusion coincide

ext there is nothing to prove unless $J(\cdot) = \Delta(\cdot), \overline{q} \cdot x : S \vdash$; if so, we must have had the premise $\Gamma, \Delta(v) \vdash \bigstar \overline{q} v \ \mathsf{Type} \ni S$, so the inductive hypothesis tells us that $\Gamma, \Delta(w) \vdash \bigstar \overline{q} w \ \mathsf{Type} \ni S$, and we may deduce that $\Gamma, \Delta(w) \vdash \bigstar \overline{q} w \ \mathsf{Type} \ni S$

typ we have $\Gamma, \Delta(v) \vdash \overline{q} v \ \mathit{Type} \ni \mathsf{Type}$, so we must have had $\overline{q} v \in \bigstar$ and $\Gamma, \Delta(v) \vdash$; $\overline{q}\cdot$ is monotonic and $\bigstar$ is an upset, so $\overline{q} w \in \bigstar$; by inductive hypothesis, $\Gamma, \Delta(w) \vdash$; we thus deduce $\Gamma, \Delta(w) \vdash \overline{q} w \ \mathsf{Type} \ni \mathsf{Type}$

fun we have $\Gamma, \Delta(v) \vdash \overline{q} v \ \mathsf{Type} \ni (q\, x{:}S) \to T$, so we must have had $\Gamma, \Delta(v), q\overline{q}v\, x{:}S \vdash \ \mathsf{Type} \ni T$, with $\overline{q} v \in \bigstar$; as before, $\overline{q} w \in \bigstar$ and by induction $\Gamma, \Delta(w), q\overline{q}w\, x : S \vdash \ \mathsf{Type} \ni T$; hence $\Gamma, \Delta(w) \vdash \overline{q} w \ \mathsf{Type} \ni (q\, x{:}S) \to T$

syn induction

abs induction—as with **fun**, generalizing to $J(\cdot)$ lets us bind a variable in world $q\overline{q}\cdot$

chk induction

var there are two cases

1. $u\, x{:}S$ in $\Gamma, \Delta(\cdot)$ with no $\cdot$ in u; $\Gamma, \Delta(v) \vdash \overline{q} v \ x \in S$
   we must have $\Gamma, \Delta(v) \vdash$ and $u \twoheadrightarrow \overline{q} v$; by induction, $\Gamma, \Delta(w) \vdash$; monotonicity tells us that $\overline{q} v \twoheadrightarrow \overline{q} w$, so by transitivity, $u \twoheadrightarrow \overline{q} w$; we may conclude $\Gamma, \Delta(w) \vdash \overline{q} w \ x \in S$
2. $\overline{q'} \cdot x{:}S$ in $\Gamma, \Delta(\cdot)$; $\Gamma, \Delta(v) \vdash \overline{q} v \ x \in S$
   we must have $\Gamma, \Delta(v) \vdash$ and $\overline{q'} v \twoheadrightarrow \overline{q} v$; by induction, $\Gamma, \Delta(w) \vdash$; functoriality of $\overline{Q}$ tells us that $\overline{q'} w \twoheadrightarrow \overline{q} w$; we may conclude $\Gamma, \Delta(w) \vdash \overline{q} w \ x \in S$

app induction □

We now obtain stability under substitution, as stated above, by a straightforward induction on derivations. The crux comes when we are given $\Gamma, u\, x : S, \Delta \vdash w \ x \in S$ with $u \twoheadrightarrow v$, $S \twoheadrightarrow T$ and $\Gamma \vdash u \ T \ni s$ and asked to deduce $\Gamma, \Delta[s : S/x] \vdash w \ s : S \in S$. We have just what we need to establish $\Gamma \vdash u \ s : S \in S$, then world subsumption and thinning finish the job.