The Formal System $\lambda\delta$ and the "Three Problems"

Ferruccio Guidi

University of Bologna, Italy

ferruccio.guidi@unibo.it

June 25, 2014

1. Overview

- $\lambda\delta$ is a typed λ-calculus inspired by Λ_∞ (van Benthem Jutting, 1984).
- $\lambda\delta$ is intended to underlie foundations of Mathematics requiring a theory of expressions, e.g. MTT (Maietti, 2009) and its predecessors.
- $\lambda\delta$ is developed as a machine-checked digital specification, that isn't the formal counterpart of some previously published informal material.
- $\lambda\delta$ comes in two versions so far: (1) formalized in Coq 7, published in 2008; (2) formalized in Matita (current version), under development.
- Following Automath, the "three problems" are: confluence (Church-Rosser), strong normalization, and preservation (subject reduction).
- The "problems" are solved for $\lambda\delta$ version 1; our aim is to discuss these "problems" for $\lambda\delta$ version 2, where more terms are typable.
- The "problems" are solved for Λ_∞ as well, but $\lambda\delta$ is more complex!

2. Terms and redexes

- Terms: $\star i$ (sort), $\# i$ (reference), $\lambda W.T$ (typed abstraction), $\delta W.T$ (abbreviation), $@V.T$ (application), and $©V.T$ (type annotation).

- By $@V.T$ we mean "$(T\ \ V)$" and by $©V.T$ we mean "$(T : V)$".
By $\delta W.T$ we mean "let $\#0$ be W in T" (explicit/delaied substitution).

- Sorts are numbered from 0, References are by index (starting at 0),

- ${}_d{\uparrow}^e$ is the term relocating function (lift of depth d and height e).

- Envs: $\star$ (empty), $L.\lambda W$ (typed declaration), and $L.\delta W$ (definition).

- ${}_d{\downarrow}^e$ is the env slicing function (drop of depth d and height e).
${}_d{\downarrow}^e L$ removes e entries after the fist d entries of L, relocating them.

- Redexes: $@V.\lambda W.T \xrightarrow{\beta} \delta(©W.V).T$ • $\delta_0{\uparrow}^1 V.T \xrightarrow{\zeta} T$ • $©V.T \xrightarrow{\epsilon} T$
$@V.\delta W.T \xrightarrow{\theta} \delta W.@({}_0{\uparrow}^1 V).T$ • $L \vdash \# i \xrightarrow{\delta} ({}_0{\uparrow}^{i+1} W)$ if ${}_0{\downarrow}^i L = K.\delta W$

- In $\lambda\delta$ version 1, the β-reductum is simpler: $@V.\lambda W.T \xrightarrow{\beta} \delta V.T$.

3. Context Sensitive Reduction

- As opposed to $\lambda\delta$ version 1, we start from $L \vdash T_1 \Rightarrow T_2$ meaning that T_1 produces T_2 by one step of parallel reduction in the env L.

$$\frac{L \vdash W_1 \Rightarrow W_2 \quad L.\delta/\lambda \boxed{W_1} \vdash T_1 \Rightarrow T_2}{L \vdash \delta/\lambda W_1.T_1 \Rightarrow \delta/\lambda W_2.T_2} \qquad \frac{{}_0{\downarrow}^i L = K.\delta W_1 \quad K \vdash W_1 \Rightarrow W_2}{L \vdash \#i \Rightarrow {}_0{\uparrow}^{i+1} W_2}\ \delta$$

- The env allows full parallelism in δ since each $\#i$ can be expanded with a different reduct of W_1. Compare with the "envless" version:

$$\frac{W_1 \Rightarrow W_2 \quad T_1 \Rightarrow T_2}{\delta W_1.T_1 \Rightarrow \delta W_2.[\boxed{W_2}/\#0]\,T_2}\ \delta \text{ with substitution } (\lambda\delta \text{ version } 1)$$

- With this approach, reduction correctly commutes with subclosure.

$$\frac{}{L \vdash \star/\#i \Rightarrow \star/\#i} \qquad \frac{L \vdash V_1 \Rightarrow V_2 \quad L \vdash T_1 \Rightarrow T_2}{L \vdash ©/@V_1.T_1 \Rightarrow ©/@V_2.T_2} \qquad \frac{L \vdash T_1 \Rightarrow T_2}{L \vdash \delta W.{}_0{\uparrow}^1 T_1 \Rightarrow T_2}\ \zeta \qquad \frac{L \vdash T_1 \Rightarrow T_2}{L \vdash ©V.T_1 \Rightarrow T_2}\ \epsilon$$

$$\frac{L \vdash V_1/W_1 \Rightarrow V_2/W_2 \quad L.\lambda \boxed{W_1} \vdash T_1 \Rightarrow T_2}{L \vdash @V_1.\lambda W_1.T_1 \Rightarrow \delta(©W_2.V_2).T_2}\ \beta \qquad \frac{L \vdash V_1/W_1 \Rightarrow V_2/W_2 \quad L.\delta \boxed{W_1} \vdash T_1 \Rightarrow T_2}{L \vdash @V_1.\delta W_1.T_1 \Rightarrow \delta W_2.@({}_0{\uparrow}^1 V_2).T_2}\ \theta$$

4. Pointwise Reduction, Confluence, Refinement

- Reduction on terms induces a reduction on envs by which entries are reduced in parallel. One step is denoted by: $L_1 \Rightarrow L_2$

$$\frac{}{\star \Rightarrow \star} \qquad \frac{K_1 \Rightarrow K_2 \quad K_1 \vdash W_1 \Rightarrow W_2}{K_1.\delta/\lambda W_1 \Rightarrow K_2.\delta/\lambda W_2}$$

- Confluence of reduction is easily achieved via "strip" lemma and "diamond" property. This must be proved in the general form:

$$\frac{L_0 \vdash T_0 \Rightarrow T_1 \quad L_0 \vdash T_0 \Rightarrow T_2 \quad L_0 \Rightarrow L_1 \quad L_0 \Rightarrow L_2}{\exists T. \quad L_1 \vdash T_1 \Rightarrow T \quad L_2 \vdash T_2 \Rightarrow T} \text{diamond}$$

- The proof is by induction on the subclosures of $\langle L_0, T_0 \rangle$ (next slide).

- In the β-cases, we must know that reduction is respected by the "refinement": $L_1 \dot{\subseteq} L_2$ and $L_2 \vdash T_1 \Rightarrow T_2$ yields $L_1 \vdash T_1 \Rightarrow T_2$.

$$\frac{}{L \dot{\subseteq} \star} \qquad \frac{K_1 \dot{\subseteq} K_2}{K_1.\delta/\lambda W \dot{\subseteq} K_2.\delta/\lambda W} \qquad \frac{K_1 \dot{\subseteq} K_2}{K_1.\delta(©W.V) \dot{\subseteq} K_2.\lambda W}$$

5. The Order on Subclosures

- When proof by structural induction fails, proof by induction on the relation "being a subterm" provides for a good alternative in general.

- In $\lambda\delta$ version 2 we need the stronger relation of "being a subclosure". One step of this relation is denoted by $\langle L_1, T_1\rangle \sqsupset \langle L_2, T_2\rangle$:

1. $\langle L, \delta/\lambda/\copyright/@V.T\rangle \sqsupset \langle L, V\rangle$ • $\langle L, \copyright/@V.T\rangle \sqsupset \langle L, T\rangle$ • $\langle L, \delta/\lambda W.T\rangle \sqsupset \langle L.\delta/\lambda W, T\rangle$

2 $\langle K.\delta/\lambda W, \#0\rangle \sqsupset \langle K, W\rangle$ • $\langle L, {}_0{\uparrow}^{e+1}\, T\rangle \sqsupset \langle {}_0{\downarrow}^{e+1}\, L, T\rangle$ (depth 0 is crucial here)

- Reduction and the order on subclosures commute as follows. The pointwise reduction is needed when $L_1 = K.\lambda V_1$ and $T_1 = \#0$.

$$\frac{\langle L_1, T_1\rangle \sqsupset \langle K, V_1\rangle \quad K \vdash V_1 \Rightarrow V_2}{\exists L_2, T_2. \quad L_1 \Rightarrow L_2 \quad L_1/L_2 \vdash T_1 \Rightarrow T_2 \quad \langle L_2, T_2\rangle \sqsupset \langle K, V_2\rangle}$$

- Pointwise reduction and the order on subclosures commute as follows.

$$\frac{\langle L_1, T_1\rangle \sqsupset \langle K_1, V\rangle \quad K_1 \Rightarrow K_2}{\exists L_2, T_2. \quad L_1 \Rightarrow L_2 \quad L_1 \vdash T_1 \Rightarrow T_2 \quad \langle L_2, T_2\rangle \sqsupset \langle K_2, V\rangle}$$

6. Typing

- In $\lambda\delta$ version 1, we start from $L \vdash T :_h U$ meaning that U is a type of T in the env L for the "sort hierarchy" h: a parameter of the calculus.

- $h : \mathtt{N} \to \mathtt{N}$ satisfies the "strict monotonicity" condition: $i < h(i)$.

- The rules for the type judgment are: (note the generic term V in place of a sort in 2 and 3, note the λ-typing in 3)

$$\frac{}{K \vdash \star i :_h \star h(i)}\ 1 \qquad \frac{{}_0{\downarrow}^i\, L = K.\delta/\lambda W \quad K \vdash W :_h V}{L \vdash \#i :_h {}_0{\uparrow}^{i+1}(V/W)}\ 2 \qquad \frac{K \vdash W :_h V \quad K.\delta/\lambda W \vdash T :_h U}{K \vdash \delta/\lambda W.T :_h \delta/\lambda W.U}\ 3$$

$$\frac{L \vdash T :_h U}{L \vdash ©U.T :_h U}\ 4 \qquad \frac{L \vdash T :_h U_1 \quad L \vdash U_1 \Leftrightarrow^* U_2 \quad L \vdash U_2 :_h T_2}{L \vdash T :_h U_2}\ 5$$

- The rule for application is too week, in $\lambda\delta$ version 2 we want 6 and 7:

$$\frac{L \vdash V :_h W \quad L \vdash T :_h \lambda W.U}{L \vdash @V.T :_h @V.\lambda W.U}\ \text{PTS-style } (\lambda\delta \text{ version 1})$$

$$\frac{L \vdash V :_h W \quad L \vdash \lambda W.T :_h \lambda W.U}{L \vdash @V.\lambda W.T :_h @V.\lambda W.U}\ 6 \qquad \frac{L \vdash T :_h U \quad L \vdash @V.U :_h W}{L \vdash @V.T :_h @V.U}\ 7$$

7. Meaning of λ-Typing and @-Typing

- Let $L \vdash T :_h U :_h S$ be $L \vdash T :_h U$ and $L \vdash U :_h S$. By λ-typing (3) $L.\lambda W \vdash T :_h U :_h S$ implies $L \vdash \lambda W.T :_h \lambda W.U :_h \lambda W.S$.

- $\lambda W.T$ is a function, $\lambda W.U$ is function space, $\lambda W.S$ is a collection of function spaces with common domain W and codomains in S.

- The implicit function is legal: $L \vdash \#i :_h \lambda W.U :_h \lambda W.S$ since by the "start" rule (2), the kind (i.e. type of type) of $\#i$ may differ from a sort.

- The function (implicit or not) may receive an argument (even by the PTS-style "application" rule): $L \vdash @V.\#i :_h @V.\lambda W.U :_h @V.\lambda W.S$.

- The implicit space is legal: $L \vdash \#i :_h \#j :_h \lambda W.S$ and the function may be applied by @-typing (7): $L \vdash @V.\#i :_h @V.\#j :_h @V.\lambda W.S$.

- If we reject @-typing, we can still η-expand the function space $\#j$: $L \vdash \#i :_h \lambda W.@(\#0).\#(j+1) :_h \lambda W.S$ (PTS: η-conversion with Π).

8. Preservation Analyzed

- The "preservation of type" (subject reduction) is stated as follows:

$L \vdash T_1 :_h U$ and $L \vdash T_1 \Rightarrow T_2$ yield $L \vdash T_2 :_h U$.

- Usual proof: by induction on $L \vdash T_1 \Rightarrow T_2$ inverting $L \vdash T_1 :_h U$.

The inversion lemma for @ involves the "iterated type" judgment:

$$\frac{L \vdash @V.T :_h X}{\exists W, Y, U. \quad L \vdash V :_h W \quad L \vdash T :_h Y \ :_h^* \ \lambda W.U \quad L \vdash @V.Y \Leftrightarrow^* X} \text{ inversion for } @$$

- Type is modulo conversion: a conversion (e.g. a multistep reduction) is allowed at each step of the type chain $L \vdash Y :_h \ldots :_h \lambda W.U$.

- So a **mutual recursion** emerges between single step preservation and multiple step preservation at a higher level in the type hierarchy.

- Unfortunately this recursion involves other participants as well.

- We also need **simultaneous induction** on four axes: subclosures, computation's length (terms/envs), degree (level in the type hierarchy).

9. Preservation Analyzed Farther

- Proving preservation splits in two: (1) which are the participants to the mutual recursion? (2) is the simultaneous induction well founded?
- (1) was solved in 5 months. (2) was solved two days ago after 15 months. In the literature (2) is the "big tree" theorem (solved for Λ_∞).
- The "big tree" of a closure $\langle L_1, T_1 \rangle$ comprises the closures $\langle L_2, T_2 \rangle$ reachable from $\langle L_1, T_1 \rangle$ following the four axes in any way.
- Subclosures are finite, as well as finite-length computations on terms. However, we can avoid the use of length by observing the following:
- $\langle L_1, T_1 \rangle$ is typed so the computations from T_1 are finite (strong normalization holds). Therefore we use the axis of reducts instead.
- Contrary to CIC and MTT, the pointwise computations from L_1 (n.a. in Λ_∞) are finite only for the entries of L_1 referred by T_1. See Rule (2).

10. Static Type Assignment

- Important: if $L \vdash T :_h U$ then there exists U_0 such that $L \vdash T :_h U_0$ is proved without the "conversion" Rule (5) (solved for version 1).

- U_0 is the "canonical" or "static" type of T. This property holds in a PTS with delayed Π-reduction (Kamareddine, Bloo, Nederpelt, 1999).

- The static type assignment $L \vdash T \bullet_h U$ is defined by:

$$\frac{}{L \vdash \star i \bullet_h \star h(i)} \qquad \frac{L \vdash T \bullet_h U}{L \vdash @V.T \bullet_h @V.U} \qquad \frac{L \vdash T \bullet_h U}{L \vdash ©V.T \bullet_h U}$$

$$\frac{{}_0{\downarrow}^i\, L = K.\delta/\lambda W \quad K \vdash W \bullet_h V}{L \vdash \#i \bullet_h {}_0{\uparrow}^{i+1}\, V/W} \qquad \frac{L.\delta/\lambda W \vdash T \bullet_h U}{L \vdash \delta/\lambda W.T \bullet_h \delta/\lambda W.U}$$

- $\langle L, T \rangle$ has a static type iff the head variable reference of T is hereditarily bound in L. An equivalent condition is on the next slide.

11. Degree Assignment

- Contrary to Λ_∞, by Rule (1), $\lambda\delta$ has infinite type levels. The degree must be assigned in a parametric reference system, termed g hereafter.

- $g : \mathtt{N} \to \mathtt{N}$ sets the degree of sorts. It satisfies the "compatibility" condition: $g(h(i)) = g(i) - 1$. It is formalized as functional relation.

- The rules for assigning degree l to $\langle L, T\rangle$ (write $L \vdash T \blacksquare_{h,g}\, l$) are:

$$\frac{g(i) = l}{L \vdash \star i \blacksquare_{h,g}\, l} \qquad \frac{{}_0{\downarrow}^i\, L = K.\delta/\lambda W \quad K \vdash W \blacksquare_{h,g}\, l}{L \vdash \# i \blacksquare_{h,g}\, l/(l+1)} \qquad \frac{L.\delta/\lambda W \vdash T \blacksquare_{h,g}\, l}{L \vdash \delta/\lambda W.T \blacksquare_{h,g}\, l} \qquad \frac{L \vdash T \blacksquare_{h,g}\, l}{L \vdash \copyright/@V.T \blacksquare_{h,g}\, l}$$

- Given $L \vdash T \bullet_h U$ with $L \vdash T \blacksquare_{h,g}\, l > 0$, the transition from $\langle L, T\rangle$ to $\langle L, U\rangle$ is a "step" along the axis of "static typing".

- $\langle L, T\rangle$ has a degree for some g iff it has a static type.

- If $L \vdash T \bullet_h U$ and $L \vdash T \blacksquare_{h,g}\, l$, then $L \vdash U \blacksquare_{h,g}\, (l-1)$.

12. Stratified Validity

- Type is defined modulo conversion: preservation is more difficult.
Validity (having or being a type) is not: preservation is less difficult.

- How are these linked? T is valid when it has a type, vice versa the types or a valid T are the valid U's convertible to the static type of T.

- Stratified validity of $\langle L, T\rangle$ (write $L \vdash T\ !_{h,g}$) is defined as follows:

$$\frac{}{L \vdash \star i\ !_{h,g}} \qquad \frac{{}_0{\downarrow}^i L = K.\delta/\lambda W \quad K \vdash W\ !_{h,g}}{L \vdash \#i\ !_{h,g}} \qquad \frac{L \vdash W\ !_{h,g} \quad L.\delta/\lambda W \vdash T\ !_{h,g}}{L \vdash \delta/\lambda W.T\ !_{h,g}}$$

$$\frac{L \vdash V\ !_{h,g} \quad L \vdash T\ !_{h,g} \quad L \vdash T \blacksquare_{h,g} (l+1) \quad L \vdash T \bullet_h U \quad L \vdash U \Leftrightarrow^* V}{L \vdash \copyright V.T\ !_{h,g}}$$

$$\frac{L \vdash V\ !_{h,g},\ L \vdash T\ !_{h,g},\ L \vdash V \blacksquare_{h,g} (l+1),\ L \vdash V \bullet_h W,\ L \vdash W \Rightarrow^* W_0,\ L \vdash T \bullet^*\!\Rightarrow^*_{h,g} \lambda W_0.U}{L \vdash @V.T\ !_{h,g}}$$

- Decomposed extended computation (write $L \vdash T_1 \bullet^*\!\Rightarrow^*_{h,g} T_2$) is:
$\exists T, l_1, l_2.\quad l_2 \leqslant l_1,\ L \vdash T_1 \blacksquare_{h,g} l_1,\ L \vdash T_1 \bullet_h^{*(l_2)} T,\ L \vdash T \Rightarrow^* T_2.$

13. The Mutual Recursion

- Preservation of validity needs a mutual recursion with 4 participants:

1 $L_1 \vdash T_1 \,!_{h,g}$ and $L_1 \vdash T_1 \Rightarrow T_2$ and $L_1 \Rightarrow L_2$ implies $L_2 \vdash T_2 \,!_{h,g}$;

2 $L_1 \vdash T_1 \,!_{h,g}$ and $L_1 \vdash T_1 \blacksquare_{h,g}\, l$ and $L_1 \vdash T_1 \Rightarrow T_2$ and $L_1 \Rightarrow L_2$ implies $L_2 \vdash T_2 \blacksquare_{h,g}\, l$;

3 $L \vdash T_1 \,!_{h,g}$ and $l_2 \leqslant l_1$ and $L \vdash T_1 \blacksquare_{h,g}\, l_1$ and $L \vdash T_1 \bullet_h^{*(l_2)} T_2$ implies $L \vdash T_2 \,!_{h,g}$;

4 if $L_1 \vdash T_1 \,!_{h,g}$ and $l_2 \leqslant l_1$ and $L_1 \vdash T_1 \blacksquare_{h,g}\, l_1$ and $L_1 \vdash T_1 \bullet_h^{*(l_2)} U_1$ and $L_1 \vdash T_1 \Rightarrow T_2$ and $L_1 \Rightarrow L_2$, then there exists U_2 such that $L_2 \vdash T_2 \bullet_h^{*(l_2)} U_2$ and $L_2 \vdash U_1 \Leftrightarrow^* U_2$.

- Every Participant depends on all participants (including itself), except for Participant 2 that does not depend on Participant 4.

- Suitable "refinements" appear in the β-cases of Participants 1, 2, 4.

- Given the characterization of typing through validity (previous slide), preservation of type follows immediately from Participants 1 and 4.

- Important properties not depending on the mutual recursion: valid terms have a static type and are strongly normalizing.

14. The Simultaneous Induction: Normal Forms

- Axis of subclosures: $\langle L, T\rangle$ is in n.f. when L and T are atomic.
- Axis of static types: $\langle L, T\rangle$ is in normal form when $L \vdash T \blacksquare_{h,g} 0$.
- Axis of term reducts: $\langle L, T_1\rangle$ is in normal form when $L \vdash T_1 \Rightarrow T_2$ implies $T_1 = T_2$. Reduction steps are too short for single-step cycles.
- Axis of env reducts: $\langle L_1, T\rangle$ is in normal form when $L_1 \Rightarrow L_2$ implies $L_1 = L_2$ considering just the entries referred by T.
- To this end we introduce lazy equivalence (write $L_1\ {}_d{\equiv}_U\ L_2$):

$$L_1\ {}_d{\equiv}_U\ L_2 \text{ iff } |L_1| = |L_2| \text{ and for every } K_1, K_2, W_1, W_2, i,\ \ d \leqslant i \text{ and } \forall T.\ {}_i{\uparrow}^1 T \neq U \text{ and}$$

$$_0{\downarrow}^i L_1 = K_1.\delta/\lambda W_1 \text{ and } _0{\downarrow}^i L_2 = K_2.\delta/\lambda W_2 \text{ imply } W_1 = W_2 \text{ and } K_1\ {}_0{\equiv}_{W_1}\ K_2.$$

- $|L|$ counts the entries of L and the "depth" d allows to prove:

$$L_1\ {}_d{\equiv}_W\ L_2 \text{ and } L_1.\delta/\lambda W\ {}_{(d+1)}{\equiv}_T\ L_2.\delta/\lambda W \text{ imply } L_1\ {}_d{\equiv}_{\delta/\lambda W.U}\ L_2.$$

- So, last axis: $\langle L_1, T\rangle$ is in n.f. when $L_1 \Rightarrow L_2$ implies $L_1\ {}_0{\equiv}_T\ L_2$.

15. The Simultaneous Induction: Extension

- Leading idea: if we could rearrange a computation t in the "big tree" grouping the steps along each axis, we could prove that t is finite.

- Unfortunately: static type assignment commutes neither with reduction (participant 4), nor with the order on subclosures.

- Reduction and static type assignment are not separable in "big trees". We generalize both by extending reduction with "type inference" steps.

- Extended redexes: $L \vdash \star i \xrightarrow{s} \star h(i)$ if $g(i) > 0$ • $©V.T \xrightarrow{t} V$ $L \vdash \# i \xrightarrow{l} ({}_0{\uparrow}^{i+1} W)$ if ${}_0{\downarrow}^i L = K.\lambda W$ (the "δ-redex for λ").

- Hereafter the relations $L \vdash T_1 \Rightarrow_{h,g} T_2$ and $L_1 \Rightarrow_{h,g} L_2$ denote one step of extended reduction on terms and environments respectively.

- No single-step cycles: unchanged halting conditions on these axes.

- No confluence in general (ϵ-step vs. t-step). May hold on valid terms.

16. The Simultaneous Induction: Decomposition

- Note: we can prove that $L \vdash T_1 \bullet^* \!\!\Rightarrow^*_{h,g} T_2$ implies $L \vdash T_1 \Rightarrow^*_{h,g} T_2$.
- Extended reduction, subclosures, and lazy equivalence commute thus:

$$\frac{\langle L, T_1 \rangle \sqsupset \langle K, V_1 \rangle;\ K \vdash V_1 \Rightarrow_{h,g} V_2}{\exists T_2.\ L \vdash T_1 \Rightarrow_{h,g} T_2;\ \langle L, T_2 \rangle \sqsupset \langle K, V_2 \rangle}\ (A) \qquad \frac{L_1 \Rightarrow_{h,g} L_2;\ L_2 \vdash T_1 \Rightarrow_{h,g} T_2}{L_1 \vdash T_1 \Rightarrow^*_{h,g} T_2}\ (B)$$

$$\frac{L_1\ {}_0{\equiv}_{T_1}\ L_2;\ L_2 \vdash T_1 \Rightarrow_{h,g} T_2}{L_1 \vdash T_1 \Rightarrow_{h,g} T_2}\ (C) \qquad \frac{L_1 \Rightarrow_{h,g} L_2;\ \langle L_2, T_2 \rangle \sqsupset \langle K_2, V \rangle}{\exists K_1, T.\ L_1 \vdash T_2 \Rightarrow_{h,g} T;\ \langle L_1, T \rangle \sqsupset \langle K_1, V \rangle;\ K_1 \Rightarrow_{h,g} K_2}\ (D)$$

$$\frac{L_1\ {}_0{\equiv}_T\ L_2;\ \langle L_2, T \rangle \sqsupset \langle K_2, V \rangle}{\exists K_1.\ \langle L_1, T \rangle \sqsupset \langle K_1, V \rangle;\ K_1\ {}_0{\equiv}_V\ K_2}\ (E) \qquad \frac{L_1\ {}_d{\equiv}_T\ L_2;\ L_2 \Rightarrow_{h,g} K_2}{\exists K_1.\ L_1 \Rightarrow_{h,g} K_1;\ K_1\ {}_d{\equiv}_T\ K_2}\ (F)$$

- Rule (A) shows the gain of extended reduction over ordinary one (compare with slide 5). We did not try Rule (D) for ordinary reduction.
- The proof of rule (F) is hard and requires a dedicated apparatus.
- Given a computation t from $\langle L_1, T_1 \rangle$ to $\langle L_2, T_2 \rangle$, there are L_0, L, T s.t. $L_1 \vdash T_1 \Rightarrow^*_{h,g} T;\ \langle L_1, T \rangle \sqsupset^* \langle L, T_2 \rangle;\ L \Rightarrow^*_{h,g} L_0;\ L_0\ {}_0{\equiv}_{T_2}\ L_2$

17. Atomic Arity Assignment

- Strong normalization must be proved for extended reduction. We can adapt the proof working in $\lambda\delta$ version 1 for ordinary reduction.

- We use Tait's candidates of reducibility (CR) containing closures.

- An "arity" encodes the structure of a CR: a "simple" type in this case. $L \vdash T \mathrel{\vdots} A$ means that $\langle L, T\rangle$ may belong to the CR with arity A.

$$\frac{}{L \vdash \star i \mathrel{\vdots} \star} \qquad \frac{{}_0{\downarrow}^i L = K.\delta/\lambda W \quad K \vdash W \mathrel{\vdots} B}{L \vdash \# i \mathrel{\vdots} B} \qquad \frac{L \vdash W \mathrel{\vdots} B \quad L.\delta W \vdash T \mathrel{\vdots} A}{L \vdash \delta W.T \mathrel{\vdots} A}$$

$$\frac{L \vdash W \mathrel{\vdots} B \quad L.\lambda W \vdash T \mathrel{\vdots} A}{L \vdash \lambda W.T \mathrel{\vdots} B \to A} \qquad \frac{L \vdash V \mathrel{\vdots} B \quad L \vdash T \mathrel{\vdots} B \to A}{L \vdash @V.T \mathrel{\vdots} A} \qquad \frac{L \vdash V \mathrel{\vdots} A \quad L \vdash T \mathrel{\vdots} A}{L \vdash ©V.T \mathrel{\vdots} A}$$

- Important: $L \vdash T\ !_{h,g}$ implies $L \vdash T \mathrel{\vdots} A$ for some A. Valid terms are simply typed (replacing type annotations with their arities).

- $L \vdash T \mathrel{\vdots} A$ implies $L \vdash T \bullet_h U$ and $L \vdash U \mathrel{\vdots} A$ for some U.

If we take an extended reduction step, we remain in the same CR.

18. Some Points on Strong Normalization

- Idea of the proof. Given $L \vdash T : A$ we construct the CR $[\![A]\!]_{h,g}$ by induction on A, and prove $\langle L, T \rangle \in [\![A]\!]_{h,g}$ by induction on $L \vdash T : A$.

- The statement requires a suitable "refinement" relation to handle the β-cases, and a generalization of the relocating functions: ${}_d{\uparrow}^e$ and ${}_d{\downarrow}^e$.

- A CR must satisfy saturation conditions S0 to S7. S1 is Girard's CR1, S2 is Tait's iii, S0 is $\langle {}_d{\downarrow}^e L, T \rangle \in [\![A]\!]_{h,g}$ implies $\langle L, {}_d{\uparrow}^e T \rangle \in [\![A]\!]_{h,g}$.

- S3 (Tait's ii) is: $\langle L, @V_1. \ldots @V_n.\delta(©W.V).T \rangle \in [\![A]\!]_{h,g}$ implies $\langle L, @V_1. \ldots @V_n.@V.\lambda W.T \rangle \in [\![A]\!]_{h,g}$. Proving S3 for $[\![\star]\!]_{h,g}$ requires:

- $L \vdash @V.\lambda W.T \Rightarrow^*_{h,g} U$ (head) implies $L \vdash \delta(©W.V).T \Rightarrow^*_{h,g} U$. With extended reduction, $T = \#0$, $U = {}_0{\uparrow}^1 W$ is possible on the l.h.s.

- To handle this case on the r.h.s, we need W in the β-reductum (contrary to $\lambda\delta$ version 1), and we need the t-reduction step.

Thank you