Taking Control Over The Multiverse

Marius Weidner

Chair of Programming Languages, University of Freiburg
weidner@cs.uni-freiburg.de

Abstract.

0.1 Syntax

$$
\begin{array}{lll}
t, \ell, A, B ::= & x & \\
& \mid \lambda(x : A) \to t & \\
& \mid t_1\ t_2 & \\
& \mid \forall(x : A) \to B & \\
& \mid t_1 \equiv_A t_2 & \\
& \mid \texttt{refl}\ t & \\
& \mid A \uplus B & \\
& \mid \texttt{inj}_1\ t & \\
& \mid \texttt{inj}_2\ t & \\
& \mid \texttt{case}\ \ t\ \ \texttt{of} & \\
& \quad \texttt{inj}_1\ t \to t_1 & \\
& \quad \texttt{inj}_2\ t \to t_2 & \\
& \mid \bot & \\
& \mid \texttt{Level} & \\
& \mid 0 & \\
& \mid \omega \uparrow \ell_1 +_t \ell_2 & :: (\ell_1 : \texttt{Level}) \to (\ell_2 : Level) \to (t : \uparrow \ell_2 \leq \ell_1) \\
& \mid \texttt{case}_\ell\ t\ \ \texttt{of} & \texttt{\{-\# OPTIONS --undecidable-type-checking \#-\}} \\
& \quad 0 \to t_1 & \\
& \quad \omega \uparrow \ell_1 +_t \ell_2 \to t_2 & \\
& \mid \texttt{suc}\ \ \ell & \\
& \mid \uparrow \ell & \\
& \mid \ell_1 \sqcup \ell_2 & \\
& \mid \ell_1 <_\ell \ell_2 & \texttt{\{-\# OPTIONS --undecidable-type-checking \#-\}} \\
& \mid\ <_{\ell_1} & :: 0 <_\ell \omega \uparrow \ell_1 +_t \ell_2 \\
& \mid\ <_{\ell_2}\ \ t & :: \ell_{11} <_\ell \ell_{21} \to (\omega \uparrow \ell_{11} +_t \ell_{12}) <_\ell (\omega \uparrow \ell_{21} +_t \ell_{22}) \\
& \mid\ <_{\ell_3}\ \ t\ t' & :: \ell_{11} \equiv \ell_{21} \to \ell_{21} <_\ell \ell_{22} \to (\omega \uparrow \ell_{11} +_t \ell_{12}) <_\ell (\omega \uparrow \ell_{21} +_t \ell_{22}) \\
& \mid \texttt{case}_{<_\ell}\ t\ \ \texttt{of} & \texttt{\{-\# OPTIONS --undecidable-type-checking \#-\}} \\
& \quad <_{\ell_1} \to t_1 & \\
& \quad <_{\ell_2}\ \ t \to t_2 & \\
& \quad <_{\ell_2}\ \ t\ t' \to t_3 & \\
& \mid \texttt{Level}_\ell & \\
& \mid \ell ,_\ell t & \\
& \mid \texttt{proj}_\ell\ t & \\
& \mid \texttt{proj}_{<_\ell}\ t & \\
& \mid \texttt{Set}_\ell & \\
& \mid \texttt{Set}_{\varepsilon_0 + i} & \texttt{for all}\ \ i \in \mathbb{N}
\end{array}
$$

We write `Set` for $\texttt{Set}_0$.

We also write $\ell_1 \leq_\ell \ell_2$ as shorthand for $\ell_1 <_\ell \ell_2 \uplus \ell_1 \equiv \ell_2$ and $\ell_1 > \ell_2$ for $\ell_2 < \ell_1$ as well as $\ell_1 \geq \ell_2$ for $\ell_2 \leq \ell_1$.

We might omit the proof t that $\ell_1 \geq\uparrow \ell_2$ in the constructor $\omega \uparrow \ell_1 +_t \ell_2$ if it follows from context.

By an abuse of notation we may write $\texttt{f} : \forall(\ell : \texttt{Level}_{\ell'}) \to \texttt{Set}\ \ell$ and $\texttt{f}\ \ell\ \{\ell < \ell'\}$ instead of $\texttt{f} : \forall(\ell : \texttt{Level}_{\ell'}) \to \texttt{Set}\ (\texttt{proj}_\ell\ \ell)$ and $\texttt{f}\ (\ell,_\ell \ell < \ell')$ which is closer to what we believe should be implemented.

Note that $\texttt{suc}\ \ell$ and $\uparrow \ell$ are essentially just definitions possible when `{-# OPTIONS --undecidable-type-checking #-}` is enabled. We could implement them in an manually checked unsafe module and mark them for the compiler similar to the constructors of Level.

- All syntax constructs marked with `{-# OPTIONS --undecidable-type-checking #-}` should only be visible to the compiler / some manually checked prelude module that included the least definitions introduced above.
- Note that in the case of level quantification the user *sees* $\_ <_\ell \_$ *indirectly* in a secure way.
- IDEA: Can we allow $\_ <_\ell \_$ to appear in the *return type* of a function without breaking decidability of typechecking? Further: We could allow *fully generalized* $\ell_1 <_\ell \ell_2$ as argument.
- Enabling the option for use in any other module, enables the user to break decidability of typechecking but also allows to add custom laws.

0.2 Laws

Idempotence: $\ell \sqcup \ell \equiv \ell$
Associativity: $(\ell_1 \sqcup \ell_2) \sqcup \ell_3 \equiv \ell_1 \sqcup (\ell_2 \sqcup \ell_3)$
Commutativity: $\ell_1 \sqcup \ell_2 \equiv \ell_2 \sqcup \ell_1$
Distributivity$_1$: suc $(\ell_1 \sqcup \ell_2) \equiv \texttt{suc}\ \ell_1 \sqcup \texttt{suc}\ \ell_2$
Distributivity$_2$: $\omega \uparrow \ell +_t (\ell_1 \sqcup \ell_2) \equiv \omega \uparrow \ell +_{t_1} \ell_1 \sqcup \omega \uparrow \ell +_{t_2} \ell_2$
Distributivity$_3$: $\uparrow (\ell_1 \sqcup \ell_2) \equiv\uparrow \ell_1 \sqcup \uparrow \ell_2$
Neutrality: $\ell \sqcup 0 \equiv \ell$
Subsumption$_1$: $\ell \sqcup \texttt{suc}^n\ \ell \equiv \texttt{suc}^n\ \ell$
Subsumption$_2$: $\ell \sqcup \omega \uparrow \ell_1 + .. + \omega \uparrow \ell_n + \texttt{suc}^n\ \ell \equiv \omega \uparrow \ell_1 + .. + \omega \uparrow \ell_n + \texttt{suc}^n\ \ell$
Subsumption$_3$: $\ell \sqcup \uparrow^n \ell \equiv \ell$

- All laws should be provable when `{-# OPTIONS --undecidable-type-checking #-}` is enabled
- With `{-# OPTIONS --undecidable-type-checking #-}` enabled you can add more reduction rules (either applying them explicitly or by using `{-# REWRITE #-}`
- The rewrite system is ‘best effort’, i.e. *not* complete..
- .. it might be confluent though (it probably even *needs* to be?)

We should probably also include a library of manually checked equations enabling inequality-reasoning that make use of `{-# OPTIONS --undecidable-type-checking #-}`
Transitivity: $\ell_1 <_\ell \ell_2 \to \ell_2 <_\ell \ell_3 \to \ell_1 <_\ell \ell_3$
Subsumption: $\ell_1 <_\ell \ell_2 \to \ell_1 <_\ell \ell_2 \sqcup \ell_3$

0.3 Typing

$$\frac{(x : T) \in \Gamma}{\Gamma \vdash x : T}\ \text{T-Var}$$

todo: add context well formedness(?)

$$\frac{\Gamma, x : A \vdash t : B \quad \Gamma \vdash A : \texttt{Set}_\ell}{\Gamma \vdash \lambda(x : A) \to t : \forall(x : A) \to B}\ \text{T-Abs}$$

$$\frac{\Gamma \vdash t_1 : \forall(x : A) \to B \quad \Gamma \vdash t_2 : A}{\Gamma \vdash t_1\ t_2 : B[x/t_2]}\ \text{T-App}$$

$$\frac{\Gamma \vdash A : \texttt{Set}_{\ell_1} \quad \Gamma, x : A \vdash B : \texttt{Set}_{\ell_2}}{\Gamma \vdash \forall(x : A) \to B : \texttt{Set}_{\ell_1 \sqcup \ell_2}}\ \text{T-All}$$

$$\frac{\Gamma \vdash t_2 : A_2 \quad \Gamma \vdash A_1 = A_2 : \texttt{Set}_\ell}{\Gamma \vdash t_1 : A_1}\ \text{T-Conv}$$

todo: add definitional equality rules(?)

$$\frac{\Gamma \vdash \ell : \texttt{Level}}{\Gamma \vdash \texttt{Set}_\ell : \texttt{Set}_{\texttt{suc}\ \ell}}\ \text{T-Set}$$

todo: add context well formedness(?)

$$\frac{\Gamma \vdash t_1 : A \quad \Gamma \vdash t_2 : A \quad \Gamma \vdash A : \texttt{Set}_\ell}{\Gamma \vdash t_1 \equiv_A t_2 : \texttt{Set}_\ell}\ \text{T-Eq}$$

$A \uplus B$, $\texttt{inj}_1$, $\texttt{inj}_2$, `case_of_` missing

$$\frac{}{\Gamma \vdash \texttt{Level} : \texttt{Set}_{\varepsilon_0}}\ \text{T-Level}$$

$$\frac{}{\Gamma \vdash 0 : \texttt{Level}}\ \text{T-Zero}$$

$$\frac{\Gamma \vdash \ell_1 : \texttt{Level} \quad \Gamma \vdash \ell_2 : \texttt{Level} \quad \Gamma \vdash t :\uparrow \ell_2 \leq_\ell \ell_1}{\Gamma \vdash \omega \uparrow \ell_1 +_t \ell_2 : \texttt{Level}}\ \text{T-CNF}$$

$\texttt{case}_\ell\texttt{\_of\_}$ missing

$$\frac{\Gamma \vdash \ell : \texttt{Level}}{\Gamma \vdash \texttt{suc}\ \ \ell : \texttt{Level}}\ \text{T-Suc}$$

$$\frac{\Gamma \vdash \ell : \texttt{Level}}{\Gamma \vdash \uparrow \ell : \texttt{Level}} \text{ T-Exp}$$

$$\frac{\Gamma \vdash \ell_1 : \texttt{Level} \quad \Gamma \vdash \ell_2 : \texttt{Level}}{\Gamma \vdash \ell_1 \sqcup \ell_2 : \texttt{Level}} \text{ T-LUB}$$

$$\frac{\Gamma \vdash \ell_1 : \texttt{Level} \quad \Gamma \vdash \ell_2 : \texttt{Level}}{\Gamma \vdash \ell_1 <_\ell \ell_2 : \texttt{Set}} \text{ T-LT}$$

$$\frac{\Gamma \vdash \ell_1 : \texttt{Level} \quad \Gamma \vdash \ell_2 : \texttt{Level} \quad \Gamma \vdash t : \uparrow \ell_2 \leq_\ell \ell_1}{\Gamma \vdash <_{\ell_1} : 0 <_\ell \omega \uparrow \ell_1 +_t \ell_2} \text{ T-LTZero}$$

$$\frac{\Gamma \vdash \ell_{1..4} : \texttt{Level} \quad \Gamma \vdash t_1 : \uparrow \ell_2 \leq_\ell \ell_1 \quad \Gamma \vdash t_2 : \uparrow \ell_4 \leq_\ell \ell_3 \quad \Gamma \vdash t : \ell_1 <_\ell \ell_3}{\Gamma \vdash <_{\ell_1} t : \omega \uparrow \ell_1 +_{t\ 1} \ell_2 <_\ell \omega \uparrow \ell_3 +_{t\ 2} \ell_4} \text{ T-LTExp}$$

$$\frac{\Gamma \vdash \ell_{1..4} : \texttt{Level} \quad \Gamma \vdash t_1 : \uparrow \ell_2 \leq_\ell \ell_1 \quad \Gamma \vdash t_2 : \uparrow \ell_4 \leq_\ell \ell_3 \quad \Gamma \vdash t : \ell_1 \equiv \ell_3 \quad \Gamma \vdash t' : \ell_2 <_\ell \ell_4}{\Gamma \vdash <_{\ell_1} tt' : \omega \uparrow \ell_1 +_{t\ 1} \ell_2 <_\ell \omega \uparrow \ell_3 +_{t\ 2} \ell_4} \text{ T-LTCons}$$

$\texttt{case}_{<_\ell}\texttt{\_of\_}$ missing

$$\frac{\Gamma \vdash \ell : \texttt{Level}}{\Gamma \vdash \texttt{Level}_\ell : \texttt{Set}\ \ell} \text{ T-BoundLevel}$$

$$\frac{\Gamma \vdash \ell : \texttt{Level} \quad \Gamma \vdash \ell' : \texttt{Level} \quad \Gamma \vdash t : \ell <_\ell \ell'}{\Gamma \vdash \ell ,_\ell t : \texttt{Level}_{\ell'}} \text{ T-LevelPair}$$

$$\frac{\Gamma \vdash t : \texttt{Level}_\ell}{\Gamma \vdash \texttt{proj}_\ell\ t : \texttt{Level}} \text{ T-LevelBoundProj}$$

$$\frac{\Gamma \vdash t : \texttt{Level}_\ell}{\Gamma \vdash \texttt{proj}_{<_\ell}\ t : (\texttt{proj}_\ell\ t) <_\ell \ell} \text{ T-LevelBoundProofProj}$$

$$\frac{\Gamma \vdash \ell : \texttt{Level}}{\Gamma \vdash \texttt{Set}_\ell : \texttt{Set}_{\texttt{suc}\ \ell}} \text{ T-Set}$$

$$\frac{}{\Gamma \vdash \texttt{Set}_{\varepsilon_0 + i} : \texttt{Set}_{\varepsilon_0 + i + 1}\ \texttt{for all}\ i \in \mathbb{N}} \text{ T-SetEps}$$

0.4 Semantics

$$\frac{}{\texttt{suc}\ 0 \hookrightarrow \omega \uparrow 0 + 0} \ \beta\text{-suc-0}$$

$$\frac{}{\texttt{suc}\ \omega \uparrow \ell_1 + \ell_2 \hookrightarrow \omega \uparrow \ell_1 + \texttt{suc}\ \ell_2} \ \beta\text{-suc-}\omega$$

$$\frac{}{\uparrow 0 \hookrightarrow 0}\ \beta\text{-}\uparrow\text{-}0$$

$$\frac{}{\uparrow (\omega \uparrow \ell_1 + \ell_2) \hookrightarrow \ell_1}\ \beta\text{-}\uparrow\text{-}\omega$$

0.5 Metatheory

Theorem 1 (Intrinsic Level Properties). *The laws from Section 0.2 are correct, i.e. can be proven by induction when `{-# OPTIONS --undecidable-type-checking #-}` is enabled. Furthermore the resulting rewrite system is confluent, i.e. satisfies the diamond property.*

Theorem 2 (Soundness). *The system is sound, i.e. progress and subject reduction hold. Progress hold if we have $\emptyset \vdash t : A$ then either t is in weak head normal form or $\exists t'.t \hookrightarrow t'$. Subject reduction holds if reduction preserves typing, i.e. if $\Gamma \vdash t : A$ and $t \hookrightarrow t'$ then $\Gamma \vdash t' : A$.*

Theorem 3 (Logical Consistency). *The system is logical consistent if $\emptyset \vdash t : \bot$ is not derivable. This proof requires an logical relation. This should hold even when `{-# OPTIONS --undecidable-type-checking #-}` is enabled.*

Theorem 4 (Decidability of Type Checking). *With `{-# OPTIONS --undecidable-type-checking #-}` disabled (and no usage of `{-# TERMINATING #-}` or similar) the type checking procedure terminates. When enabled, type checking may run forever.*