Type Safety for the Simply Typed Lambda Calculus with Recursive Types Using Logical Relations

Tom de Jong

23 May

1 Recap on Type Safety Using Logical Relations

Simply Typed Lambda Calculus (STLC)

Types	$\tau ::= \text{bool} \mid \tau \to \tau$
Terms	$e ::= x \mid \text{true} \mid \text{false} \mid \text{if } e \text{ then } e \text{ else } e \mid \lambda x : \tau .\, e \mid e\, e$
Values	$v ::= \text{true} \mid \text{false} \mid \lambda x : \tau .\, e$
Evaluation contexts	$E ::= [] \mid \text{if } E \text{ then } e \text{ else } e \mid E\, e \mid v\, E$
Evaluations	$\text{if false then } e_1 \text{ else } e_2 \to e_2$ $(\lambda x : \tau .\, e)v \to e[v/x]$ $\dfrac{e \to e'}{E[e] \to E[e']}$
Typing contexts	$\Gamma ::= \bullet \mid \Gamma, x : \tau$
Typing rules	$\dfrac{}{\Gamma \vdash \text{false} : \text{bool}}$ T-false $\dfrac{}{\Gamma \vdash \text{true} : \text{bool}}$ T-true $\dfrac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau}$ T-var $\dfrac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x : \tau_1 .\, e : \tau_1 \to \tau_2}$ T-abs $\dfrac{\Gamma \vdash e_1 : \tau_2 \to \tau \qquad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\, e_2 : \tau}$ T-app $\dfrac{\Gamma \vdash e : \text{bool} \qquad \Gamma \vdash e_1 : \tau \qquad \Gamma \vdash e_2 : \tau}{\Gamma \vdash \text{if } e \text{ then } e_1 \text{ else } e_2 : \tau}$ T-if

Theorem 1.1 (Type safety)**.** *If $\bullet \vdash e : \tau$ and $e \to^* e'$, then e' is a value or there exists a term e'' such that $e' \to e''$.*

Definition 1.2. For a term e, we define $\text{irred}(e) \coloneqq \neg \exists e'' \, (e' \to e'')$ and $\text{safe}(e) \coloneqq \forall e' \, (e \to^* e' \Rightarrow (e' \text{ is a value} \vee \exists\, e''(e' \to e'')))$.

Definition 1.3. We define the *value interpretation* $\mathcal{V}[\![\tau]\!]$ *of a type* τ and the *term (or expression) interpretation* $\mathcal{E}[\![\tau]\!]$ *of a type* τ simultaneously by induction on τ:

$$\begin{aligned}
\mathcal{V}[\![\text{bool}]\!] &\coloneqq \{\text{true}, \text{false}\};\\
\mathcal{V}[\![\tau_1 \to \tau_2]\!] &\coloneqq \{\lambda x : \tau.\, e \mid \forall v \in \mathcal{V}[\![\tau_1]\!]\, e[v/x] \in \mathcal{E}[\![\tau_1]\!]\};\\
\mathcal{E}[\![\tau]\!] &\coloneqq \{e \mid \forall e'(e \to e' \wedge \text{irred}(e') \Rightarrow e' \in \mathcal{V}[\![\tau]\!])\}.
\end{aligned}$$

Definition 1.4. We define the *context interpretation* $\mathcal{G}[\![\Gamma]\!]$ *of a context* Γ by induction on Γ:

$$\begin{aligned}
\mathcal{G}[\![\bullet]\!] &\coloneqq \{\emptyset\};\\
\mathcal{G}[\![\Gamma, x : \tau]\!] &\coloneqq \{\gamma \cup \{x \mapsto v\} \mid \gamma \in \mathcal{G}[\![\tau]\!] \wedge v \in \mathcal{V}[\![\tau]\!]\}.
\end{aligned}$$

Definition 1.5. Finally, we define $\Gamma \models e : \tau \coloneqq \forall \gamma \in \mathcal{G}[\![\Gamma]\!]\, \gamma(e) \in \mathcal{E}[\![\tau]\!]$.

Theorem 1.6 (Fundamental property)**.** *If* $\Gamma \vdash e : \tau$*, then* $\Gamma \models e : \tau$*.*

Theorem 1.7. *If* $\bullet \models e : \tau$*, then* $\text{safe}(e)$*.*

Type safety is now obtained as a corollary of the previous two theorems.

2 STLC Extended with Recursive Types

STLC with recursive types

Types	$\tau ::= \ldots \mid \mu\alpha.\, \tau$
Terms	$e ::= \ldots \mid \text{fold}\, e \mid \text{unfold}\, e$
Values	$v ::= \ldots \mid \text{fold}\, v$
Evaluation contexts	$E ::= \ldots \mid \text{fold}\, E \mid \text{unfold}\, E$
Evaluations	$\ldots + \text{unfold}(\text{fold}\, v) \to v$
Typing contexts	$\ldots$
Typing rules	$\ldots + \dfrac{\Gamma \vdash e : \tau[(\mu\alpha.\, \tau)/\alpha]}{\Gamma \vdash \text{fold}\, e : \mu\alpha.\, \tau}$ T-fold $\dfrac{\Gamma \vdash e : \mu\alpha.\, \tau}{\Gamma \vdash \text{unfold}\, e : \tau[(\mu\alpha.\, \tau/\alpha])}$ T-unfold

Example 2.1. *intlist* $= \mu\alpha.\ \mathbb{1} + (\mathit{int} \times \alpha)$; *bininttree* $= \mu\alpha.\ \mathbb{1} + (\mathit{int} \times \alpha \times \alpha)$

3 Type Safety for STLC with Recursive Types

Definition 3.1. We define value and term intpretations $\mathcal{V}_k[\![\tau]\!]$ and $\mathcal{E}_k[\![\tau]\!]$ indexed by natural numbers simultaneously by induction on (k, τ):

$$\begin{aligned}
\mathcal{V}_k[\![\text{bool}]\!] &\coloneqq \{\text{true}, \text{false}\};\\
\mathcal{V}_k[\![\tau_1 \to \tau_2]\!] &\coloneqq \{\lambda x : \tau_1.\, e \mid \forall j \le k\, \forall v \in \mathcal{V}_j[\![\tau_1]\!]\, e[v/x] \in \mathcal{E}_j[\![\tau_2]\!])\};\\
\mathcal{V}_k[\![\mu\alpha.\, \tau]\!] &\coloneqq \{\text{fold}\, v \mid \forall j < k\, (v \in \mathcal{V}_j[\![\tau[(\mu\alpha\, \tau)/\alpha]]\!])\};\\
\mathcal{E}_k[\![\tau]\!] &\coloneqq \{e \mid \forall j < k\, \forall e'(e \to^j e' \wedge \text{irred}(e') \Rightarrow e' \in \mathcal{V}_{k-j}[\![\tau]\!])\}.
\end{aligned}$$

Further, we define the context interpretation $\mathcal{G}_k[\![\Gamma]\!]$ indexed by natural numbers also by induction on (k, Γ):

$$\begin{aligned}
\mathcal{G}_k[\![\bullet]\!] &\coloneqq \{\emptyset\};\\
\mathcal{G}_k[\![\Gamma, x : \tau]\!] &\coloneqq \{\gamma \cup \{x \mapsto v\} \mid \gamma \in \mathcal{G}_k[\![\tau]\!] \wedge v \in \mathcal{V}_k[\![\tau]\!]\}.
\end{aligned}$$

Definition 3.2. Finally, we put $\Gamma \models e : \tau \coloneqq \forall k \ge 0\, \forall \gamma \in \mathcal{G}_k[\![\tau]\!]\, \gamma(e) \in \mathcal{E}_k[\![\tau]\!]$.

Lemma 3.3 (Monotonicity lemma)**.** *If τ is a type and $k, j \in \mathbb{N}$ with $j \le k$ and $v \in \mathcal{V}_k[\![\tau]\!]$, then $v \in \mathcal{V}_j[\![\tau]\!]$.*

Theorem 3.4 (Fundamental property)**.** *If $\Gamma \vdash e : \tau$, then $\Gamma \models e : \tau$.*

Theorem 3.5. *If $\bullet \models e : \tau$, then* safe(e)*.*

Remark 3.6. For the *intlist* example, we need product and sum types and a unit type. Of course, one may extend the STLC with such types. One can extend the value interpretations as follows:

$$\begin{aligned}
\mathcal{V}_k[\![\mathbb{1}]\!] &\coloneqq \{1\};\\
\mathcal{V}_k[\![\tau_1 \times \tau_2]\!] &\coloneqq \{\langle v_1, v_2\rangle \mid v_1 \in \mathcal{V}_k[\![\tau_1]\!] \wedge v_2 \in \mathcal{V}_k[\![\tau_2]\!]\};\\
\mathcal{V}_k[\![\tau_1 + \tau_2]\!] &\coloneqq \{\text{inl}\ v \mid v \in \mathcal{V}_k[\![\tau_1]\!]\} \cup \{\text{inr}\ v \mid v \in \mathcal{V}_k[\![\tau_2]\!]\}.
\end{aligned}$$