Type Preservation as a Confluence Problem

Aaron Stump, Garrin Kimmell, Ruba El Haj Omar
Computational Logic Center
The University of Iowa
Iowa City, Iowa, USA

Funding from U.S. National Science Foundation
TRELYS project (Tim Sheard, Stephanie Weirich)
The Standard Approach to Typing

Syntax.

\[
\begin{align*}
types T & ::= \ A \mid T_1 \Rightarrow T_2 \\
terms t & ::= \ x \mid t_1 \ t_2 \mid \lambda x : T . t
\end{align*}
\]

Operational semantics.

\[t \rightarrow_c t', \text{ small-step reduction relation (concrete)}.\]

Typing relation.

\[\Gamma \vdash t : T, \text{ with } \Gamma \text{ a context.}\]

Type Preservation.

\[\Gamma \vdash t : T \land t \rightarrow_c t' \implies \Gamma \vdash t' : T\]
Alternative: The Rewriting Approach to Typing

Proposed by Kuan, MacQueen, and Findler.


Define typing as small-step abstract reduction $\rightarrow_a$.

Terms rewrite to their types.

$$\lambda y : A.y \quad \rightarrow^*_{a} \quad A \Rightarrow A$$

$$\left(\lambda x : A \Rightarrow A.x\right) \lambda y : A.y \quad \rightarrow^*_{a} \quad A \Rightarrow A$$

$$\lambda x : A.\lambda y : A.x \quad \rightarrow^*_{a} \quad A \Rightarrow A \Rightarrow A$$

Formulate metatheory in term-rewriting terms.
This Work

Build on Kuan et al.’s work by applying term-rewriting methods.

Prove confluence of combined reduction $\rightarrow_{ca} = \rightarrow_c \cup \rightarrow_a$.

Use decreasing diagrams.

Type Preservation is a corollary.

Ultimate goal: simplify metatheory of programming languages.
The Rewriting Approach to STLC
Syntax

To rewrite a term to type, need intermediate mixed terms.

\[
\begin{align*}
types \ T & \ ::= \ A \mid T_1 \Rightarrow T_2 \\
mixed \ terms \ m & \ ::= \ x \mid m \ m' \mid \lambda x : T. \ m \mid A \mid T \Rightarrow m \\
mixed \ values \ u & \ ::= \ \lambda x : T. \ m \mid T \Rightarrow m \mid A
\end{align*}
\]

\( T \Rightarrow m \) is viewed as an abstract function.

Can be applied to arguments (concrete or abstract).

Example mixed terms:

\[
\begin{align*}
\lambda x : A.x \\
A \Rightarrow A \\
A \Rightarrow \lambda x : A.x \\
(\lambda x : A.x) \ A \\
(A \Rightarrow A) \ A
\end{align*}
\]
Concrete and Abstract Reduction Relations

Concrete reduction ($\rightarrow_c$) is call-by-value.

Abstract reduction ($\rightarrow_a$) is nondeterministic.

\[
E_c[(\lambda x : T. m) \ u] \rightarrow_c E_c[[u/x]m] \quad c(\beta)
\]

\[
E_a[\lambda x : T. m] \rightarrow_a E_a[T \Rightarrow [T/x]m] \quad a(\lambda)
\]

\[
E_a[(T \Rightarrow m) \ T] \rightarrow_a E_a[m] \quad a(\beta)
\]

where:

\[
E_c ::= \ast \mid (E_c \ t) \mid (u \ E_c)
\]

\[
E_a ::= \ast \mid (E_a \ m) \mid (m \ E_a) \mid \lambda x : T. E_a \mid T \Rightarrow E_a
\]
An Example Abstract Reduction Sequence

Reduce a term to its type (redexes underlined):

\[
\lambda x : (A \Rightarrow A). \lambda y : A. (x (x y)) \rightarrow_a \\
\lambda x : (A \Rightarrow A). A \Rightarrow (x (x A)) \rightarrow_a \\
(A \Rightarrow A) \Rightarrow A \Rightarrow ((A \Rightarrow A) ((A \Rightarrow A) A)) \rightarrow_a \\
(A \Rightarrow A) \Rightarrow A \Rightarrow ((A \Rightarrow A) A) \rightarrow_a \\
(A \Rightarrow A) \Rightarrow A \Rightarrow A
\]

Use \textit{a(}\lambda\textit{)} rule to turn \lambda to \Rightarrow.

\[
\lambda y : A. (x (x y)) \rightarrow_a A \Rightarrow (x (x A))
\]

Use \textit{a(}\beta\textit{)} rule to reduce applications of abstract functions.

\[
((A \Rightarrow A) A) \rightarrow_a A
\]
Metatheory: Relation to Standard Typing

Define standard terms $t$.

$$ t ::= x \mid \lambda x : T . t \mid t \ t' $$

**Theorem.** $x_1 : T_1, \ldots, x_n : T_n \vdash t : T$ iff $[T_1/x_1, \ldots, T_n/x_n]t \rightarrow^* T$.

Proof is similar to relating big-step and small-step concrete reduction.

Usual typing rules define big-step abstract reduction:

$$ \Gamma \vdash t : T' \Rightarrow T \quad \Gamma \vdash t' : T' \quad \Gamma \vdash t \ t' : T $$
Metatheory: Type Preservation

Standard formulation:

\[ \Gamma \vdash t : T \land t \rightarrow_c t' \quad \Rightarrow \quad \Gamma \vdash t' : T \]

Rewriting formulation:

\[ m \rightarrow^*_a T \land m \rightarrow_c m' \quad \Rightarrow \quad m' \rightarrow^*_a T \]

Follows from confluence of \( \rightarrow_{ca} \) for typable terms (prove now).
Confluence of Combined Reduction on Typable Terms
Initial Observations

Call $m$ typable iff $m \xrightarrow{a}^* T$, for some $T$.

Confluence fails for non-typable terms.

We will prove confluence for typable terms.
Basic Properties of Abstract Reduction

**Theorem.** \( \rightarrow_a \) is terminating.

**Theorem.** \( \rightarrow_a \) has the diamond property.

**Proof:**

No critical overlap of \( \rightarrow_a \)-redexes.

No duplication or deletion of \( \rightarrow_a \)-redexes.

\[
\frac{E_a[\lambda x : T. \, m] \rightarrow_a E_a[T \Rightarrow [T/x]m]}{a(\lambda)}
\]

\[
\frac{E_a[(T \Rightarrow m) \, T] \rightarrow_a E_a[m]}{a(\beta)}
\]
Recall: Decreasing Diagrams [van Oostrom]

Assign labels to steps $e \to e'$.

Order labels using some well-founded ordering $\prec$.

Confluent if all local peaks completable to locally decreasing diagrams.

Local peak has form

\[
S_1 \leftarrow_\alpha t \rightarrow_\beta S_2
\]

Valley has form

\[
S_1 \xrightarrow{\ast} \gamma \alpha \xrightarrow{\equiv} \beta \xrightarrow{\ast} (\gamma \alpha) \cup (\gamma \beta) \hat{t} \leftarrow (\gamma \alpha) \cup (\gamma \beta) \leftarrow_\alpha \equiv \leftarrow_\gamma \beta S_2
\]

where:

- $\gamma \alpha = \{ \alpha' | \alpha' \prec \alpha \}$
- $\rightarrow_A = \bigcup_{\alpha \in A} \rightarrow_\alpha$
Confluence of Combined Reduction

Main Theorem. \( \rightarrow_{ca} \) is confluent on typable terms.

To prove, apply decreasing diagrams with \( a < c \).

- \( aa \)-peaks completable since \( \rightarrow_{a} \) has diamond property.
- No non-trivial \( cc \)-peaks, by determinism of \( \rightarrow_{c} \).
- Two simple forms for \( ac \)-peaks, where \( m \rightarrow_{a} m', u \rightarrow_{a} u' \).

\[
\begin{align*}
E_{c}[(\lambda x : T. m') u] & \leftarrow_{a} E_{c}[(\lambda x : T. m) u] \rightarrow_{c} E_{c}[[u/x]m] \\
E_{c}[(\lambda x : T. m) u'] & \leftarrow_{a} E_{c}[(\lambda x : T. m) u] \rightarrow_{c} E_{c}[[u/x]m]
\end{align*}
\]

- One more complex peak (next).
Crucial Decreasing Diagram for Combined Reduction

\[ E_c[(\lambda x : T_1.m) u] \]

\[ a \leftarrow E_c[(T_1 \Rightarrow [T_1/x]m) u] \]

\[ E_c[[u/x]m] \]

\[ c \]

\[ E_c[(T_1 \Rightarrow [T_1/x]m) T_1] \]

\[ a \]

\[ a \leftarrow E_c[[T_1/x]m] \]

\[ u \rightarrow_{a}^{*} T_1, \text{ since redex typable.} \]

Locally decreasing, since \( c > a \).
Concluding Confluence

All local peaks completable to locally decreasing diagrams.

Therefore, $\rightarrow_{ca}$ is confluent for typable terms.

Type Preservation for STLC established by rewriting techniques.

New proof method.

In the paper, apply to other example systems:

$$\text{STLC+fix, STLC+}\forall, \text{Uniform-STC}$$

For Uniform-STC, we apply APROVE and ACP for $\rightarrow_a$. 
Quick Look: STLC with Type Inference

Extend types to include meta-variables $\alpha_i$.

Now $a(\beta)$ rule uses narrowing to instantiate $\alpha_i$:

$$
\begin{align*}
\sigma \text{ is mgu}(T_1, T_2) \\
E_a[(T_1 \Rightarrow m)T_2] \rightarrow_a \sigma(E_a[m])
\end{align*}
$$

Again establish confluence for typable terms.

Conclude Type Preservation.
Conclusion: A New Approach to Type Systems

Can use rewriting techniques to prove Type Preservation.

Proofs are different, qualitatively simpler, partly automatable.

A new agenda: redevelop Type Theory using rewriting approach.

1 Dependent types.
2 Curry-style systems.
   - Confluence fails.
   - Prove Type Preservation (rewriting form) directly.
3 Normalization proofs.
4 Soundness of symbolic simulation.
   - Natural operational formulation.
   - Soundness often not proved.