Well-typed programs can't be blamed

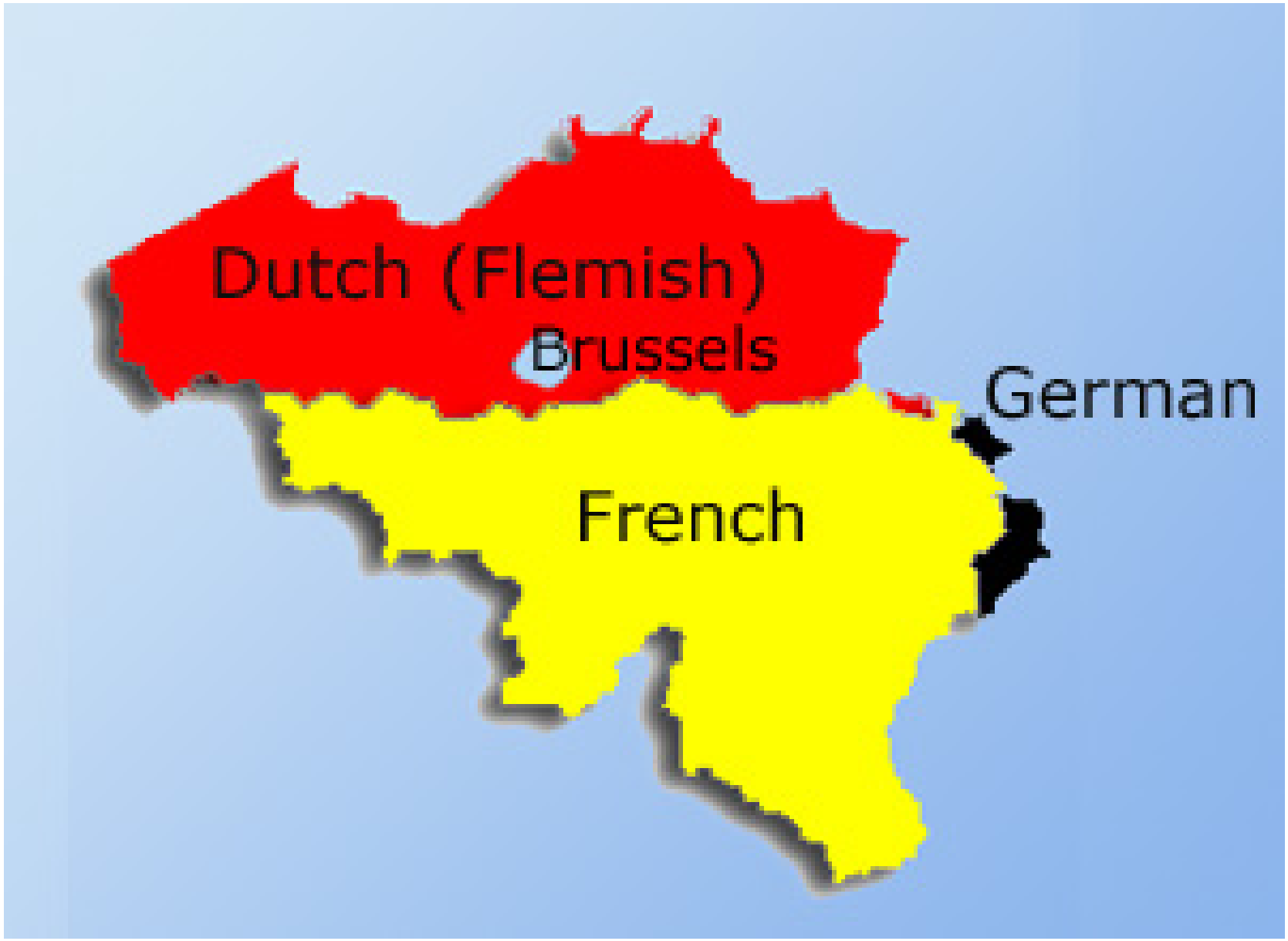
Philip Wadler
University of Edinburgh

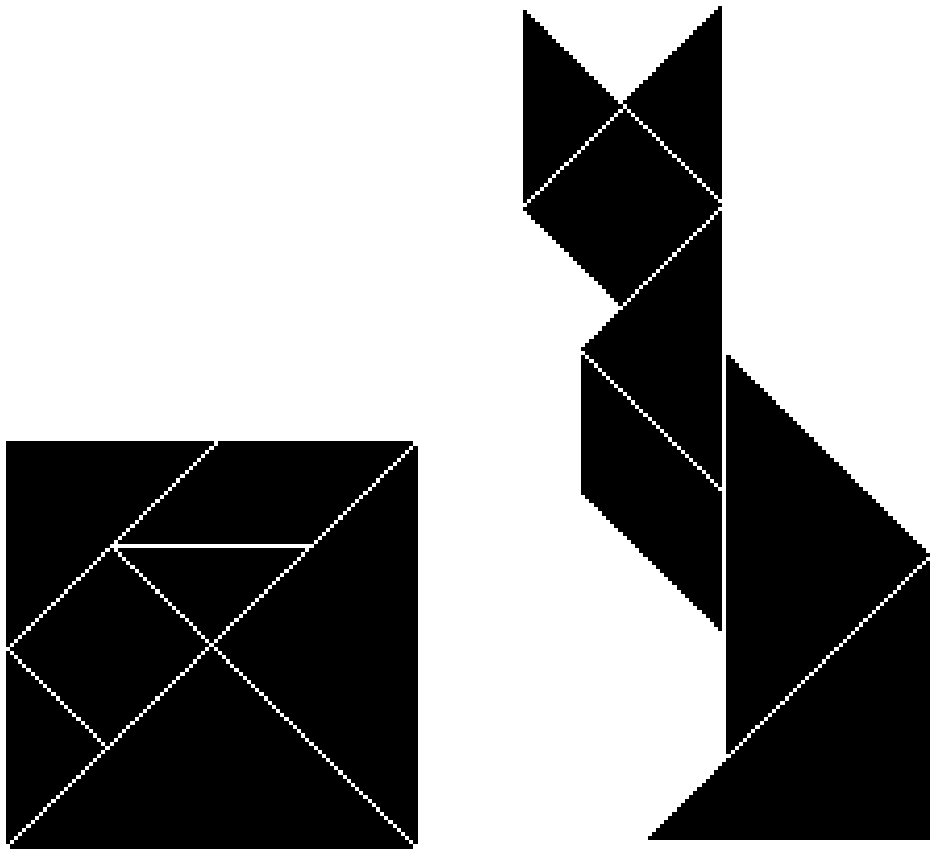
Robert Bruce Findler
University of Chicago

“The mathematics of programming languages is deep and elegant”

examples other than Curry-Howard?

terms other than ‘deep’, ‘elegant’?


Dutch (Flemish)
Brussels
German
French



A repeated theme

Thatte (1988):

Partial types

Henglein (1994):

Dynamic typing

Findler and Felleisen (2002):

Contracts

Flanagan (2006):

Hybrid types

Siek and Taha (2006):

Gradual types

A repeated theme

Visual Basic 9.0

Perl 6.0

ECMAScript 4.0

Evolving a program

An untyped program

$$
\begin{array}{l}
\lceil \texttt{let} \\
\quad x = 2 \\
\quad f = \lambda y.\ y + 1 \\
\quad h = \lambda g.\ g\ (g\ x) \\
\texttt{in} \\
\quad h\ f \rceil \\
\longrightarrow \\
\lceil 4 \rceil
\end{array}
$$

A typed program

$$
\begin{array}{l}
\texttt{let} \\
\quad x = 2 \\
\quad f = \lambda y : \texttt{Int}.\ y + 1 \\
\quad h = \lambda g : \texttt{Int} \to \texttt{Int}.\ g\,(g\,x) \\
\texttt{in} \\
\quad h\ f \\
\longrightarrow \\
\quad 4 : \texttt{Int}
\end{array}
$$

A partly typed program—narrowing

$$
\begin{array}{l}
\texttt{let} \\
\quad x = 2 \\
\quad f = \langle \texttt{Int} \to \texttt{Int} \Leftarrow \texttt{Dyn} \rangle^{p} \lceil \lambda y.\, y + 1 \rceil \\
\quad h = \lambda g : \texttt{Int} \to \texttt{Int}.\, g\,(g\,x) \\
\texttt{in} \\
\quad h\,f \\
\longrightarrow \\
\quad 4 : \texttt{Int}
\end{array}
$$

A partly typed program—narrowing

$$
\begin{array}{l}
\texttt{let} \\
\quad x = 2 \\
\quad f = \langle \texttt{Int} \to \texttt{Int} \Leftarrow \texttt{Dyn} \rangle^{p} \, \lceil \lambda y.\, \texttt{'b'} \rceil \\
\quad h = \lambda g : \texttt{Int} \to \texttt{Int}.\, g\,(g\,x) \\
\texttt{in} \\
\quad h\ f \\
\longrightarrow \\
\quad \texttt{blame}\, p
\end{array}
$$

Positive (covariant): blame the term contained in the cast

Another partly typed program—widening

$$
\begin{array}{l}
\texttt{let} \\
\quad x = \lceil 2 \rceil \\
\quad f = \langle \texttt{Dyn} \Leftarrow \texttt{Int} \to \texttt{Int} \rangle^{p}\ (\lambda y : \texttt{Int}.\ y + 1) \\
\quad h = \lceil \lambda g.\ g\ (g\ x) \rceil \\
\texttt{in} \\
\quad \lceil h\ f \rceil \\
\longrightarrow \\
\quad \lceil 4 \rceil
\end{array}
$$

Another partly typed program—widening

$$
\begin{array}{l}
\texttt{let} \\
\quad x = \lceil \texttt{'a'} \rceil \\
\quad f = \langle \texttt{Dyn} \Leftarrow \texttt{Int} \to \texttt{Int} \rangle^{p}\ (\lambda y : \texttt{Int}.\ y + 1) \\
\quad h = \lceil \lambda g.\ g\ (g\ x) \rceil \\
\texttt{in} \\
\quad \lceil h\ f \rceil \\
\longrightarrow \\
\quad \texttt{blame}\ \bar{p}
\end{array}
$$

Negative (contravariant): blame the context containg the cast

The Blame Game

Blame

$$\langle \texttt{Int} \Leftarrow \texttt{Dyn} \rangle^{p}\ \lceil 2 \rceil$$

$$\longrightarrow$$

$$2$$

$$\langle \texttt{Int} \Leftarrow \texttt{Dyn} \rangle^{p}\ \lceil \texttt{'a'} \rceil$$

$$\longrightarrow$$

$$\texttt{blame}\ p$$

The Blame Game—widening

$$(\langle \texttt{Dyn} \to \texttt{Dyn} \Leftarrow \texttt{Int} \to \texttt{Int} \rangle^{p}\ (\lambda y : \texttt{Int}.\, y + 1))\ \lceil 2 \rceil$$

$$\longrightarrow$$

$$\langle \texttt{Dyn} \Leftarrow \texttt{Int} \rangle^{p}\ ((\lambda y : \texttt{Int}.\, y + 1)\ (\langle \texttt{Int} \Leftarrow \texttt{Dyn} \rangle^{\bar{p}}\ \lceil 2 \rceil))$$

$$\longrightarrow$$

$$\lceil 3 \rceil$$

The Blame Game—widening

$$(\langle \texttt{Dyn} \to \texttt{Dyn} \Leftarrow \texttt{Int} \to \texttt{Int} \rangle^{p}\ (\lambda y : \texttt{Int}.\, y + 1))\ \lceil \texttt{'a'} \rceil$$

$$\longrightarrow$$

$$\langle \texttt{Dyn} \Leftarrow \texttt{Int} \rangle^{p}\ ((\lambda y : \texttt{Int}.\, y + 1)\ (\langle \texttt{Int} \Leftarrow \texttt{Dyn} \rangle^{\bar{p}}\ \lceil \texttt{'a'} \rceil))$$

$$\longrightarrow$$

$$\texttt{blame}\ \bar{p}$$

Widening can give rise to negative blame, but never positive blame

The Blame Game—narrowing

$$(\langle \mathtt{Int} \to \mathtt{Int} \Leftarrow \mathtt{Dyn} \to \mathtt{Dyn} \rangle^{p}\ (\lambda y : \mathtt{Dyn}.\ \lceil y + 1 \rceil))\ 2$$

$$\longrightarrow$$

$$\langle \mathtt{Int} \Leftarrow \mathtt{Dyn} \rangle^{p}\ ((\lambda y : \mathtt{Dyn}.\ \lceil y + 1 \rceil)\ (\langle \mathtt{Dyn} \Leftarrow \mathtt{Int} \rangle^{\bar{p}}\ 2))$$

$$\longrightarrow$$

$$3$$

The Blame Game—narrowing

$$(\langle \texttt{Int} \to \texttt{Int} \Leftarrow \texttt{Dyn} \to \texttt{Dyn} \rangle^{p}\ (\lambda y : \texttt{Dyn}.\ \lceil \texttt{'b'} \rceil))\ \texttt{2}$$

$$\longrightarrow$$

$$\langle \texttt{Int} \Leftarrow \texttt{Dyn} \rangle^{p}\ ((\lambda y : \texttt{Dyn}.\ \lceil \texttt{'b'} \rceil)\ (\langle \texttt{Dyn} \Leftarrow \texttt{Int} \rangle^{\bar{p}}\ \texttt{2}))$$

$$\longrightarrow$$

$$\texttt{blame}\ p$$

Narrowing can give rise to positive blame, but never negative blame

Untyped and supertyped

Untyped = Uni-typed

$$
\begin{array}{lcl}
\lceil x \rceil & = & x \\
\lceil n \rceil & = & \langle \mathtt{Dyn} \Leftarrow \mathtt{Int} \rangle\, n \\
\lceil \lambda x.\, N \rceil & = & \langle \mathtt{Dyn} \Leftarrow \mathtt{Dyn} \to \mathtt{Dyn} \rangle\, (\lambda x : \mathtt{Dyn}.\, \lceil N \rceil) \\
\lceil L\ M \rceil & = & (\langle \mathtt{Dyn} \to \mathtt{Dyn} \Leftarrow \mathtt{Dyn} \rangle\, \lceil L \rceil)\, \lceil M \rceil
\end{array}
$$

(slogan due to Bob Harper)

Contracts

$$\texttt{Nat} = \{x : \texttt{Int} \mid x \geq 0\}$$

$$\begin{array}{l}
\texttt{let} \\
\quad x = \langle \texttt{Nat} \Leftarrow \texttt{Int} \rangle\ 2 \\
\quad f = \langle \texttt{Nat} \to \texttt{Nat} \Leftarrow \texttt{Int} \to \texttt{Int} \rangle\ (\lambda y : \texttt{Int}.\ y + 1) \\
\quad h = \lambda g : \texttt{Nat} \to \texttt{Nat}.\ g\ (g\ x) \\
\texttt{in} \\
\quad h\ f \\
\longrightarrow \\
\quad 4_{\texttt{Nat}} : \texttt{Nat}
\end{array}$$

Subtyping

Subtype

$$\frac{}{\texttt{Dyn} <: \texttt{Dyn}}$$

$$\frac{S' <: S \qquad T <: T'}{S \to T <: S' \to T'}$$

$$\frac{s \text{ implies } t}{\{x : B \mid s\} <: \{x : B \mid t\}}$$

Example:

$$\texttt{Dyn} \to \texttt{Int} <: \texttt{Int} \to \texttt{Dyn}$$

$$\texttt{Int} \to \texttt{Nat} <: \texttt{Nat} \to \texttt{Int}$$

Positive subtype—widening

$$\frac{}{S <:^{+} \texttt{Dyn}}$$

$$\frac{S' <:^{-} S \qquad T <:^{+} T'}{S \to T <:^{+} S' \to T'}$$

$$\frac{s \text{ implies } t}{\{x : B \mid s\} <:^{+} \{x : B \mid t\}}$$

Examples:

$$\texttt{Int} \to \texttt{Int} <:^{+} \texttt{Dyn} \to \texttt{Dyn}$$

$$\texttt{Nat} \to \texttt{Nat} <:^{+} \texttt{Int} \to \texttt{Int}$$

Negative subtype—narrowing

$$\frac{}{\texttt{Dyn} <:^{-} T}$$

$$\frac{S' <:^{+} S \qquad T <:^{-} T'}{S \to T <:^{-} S' \to T'}$$

$$\frac{}{\{x : B \mid s\} <:^{-} \{x : B \mid t\}}$$

Examples:

$$\texttt{Dyn} \to \texttt{Dyn} <:^{-} \texttt{Int} \to \texttt{Int}$$

$$\texttt{Int} \to \texttt{Int} <:^{-} \texttt{Nat} \to \texttt{Nat}$$

Naive subtype

$$\frac{}{S <:_n \texttt{Dyn}}$$

$$\frac{S <:_n S' \qquad T <:_n T'}{S \to T <:_n S' \to T'}$$

$$\frac{s \text{ implies } t}{\{x : B \mid s\} <:_n \{x : B \mid t\}}$$

Example:

$$\texttt{Int} \to \texttt{Int} <:_n \texttt{Dyn} \to \texttt{Dyn}$$

$$\texttt{Nat} \to \texttt{Nat} <:_n \texttt{Int} \to \texttt{Int}$$

And now . . . a theorem!

The Blame Theorem

Consider a source program, where p appears only once.

- If $S <:^{+} T$ then $\langle T \Leftarrow S \rangle^{p} s \not\longrightarrow \texttt{blame}\ p$.
- If $S <:^{-} T$ then $\langle T \Leftarrow S \rangle^{p} s \not\longrightarrow \texttt{blame}\ \bar{p}$.

The Blame Lemma

Let t be a well-typed term and p be a blame label, and consider all subterms of t containing p. If

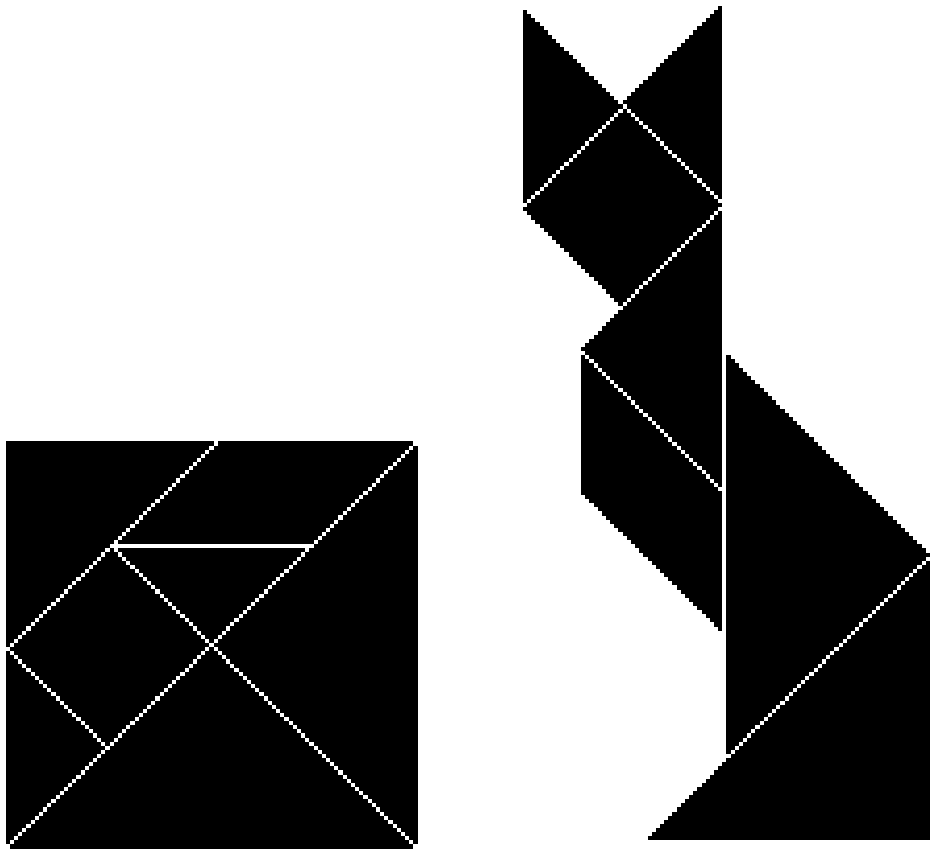
- every cast with label p is a positive subtype,

$$\langle T \Leftarrow S \rangle^{p}\ s \quad \text{has} \quad S <:^{+} T$$

- every cast with label $\bar{p}$ is a negative subtype,

$$\langle T \Leftarrow S \rangle^{\bar{p}}\ s \quad \text{has} \quad S <:^{-} T$$

then $t \not\longrightarrow^{*} \texttt{blame}\ p$.



The First Tangram Theorem

$$S <: T \text{ if and only if } S <:^{+} T \text{ and } S <:^{-} T$$

The Blame Corollary

Consider a source program, where p appears only once.

- If $S <: T$ then $\langle T \Leftarrow S \rangle^{p} s \not\longrightarrow \texttt{blame}\ p, \texttt{blame}\ \bar{p}$.

The Second Tangram Theorem

$$S <:_n T \text{ if and only if } S <:^+ T \text{ and } T <:^- S$$

The Blame Corollaries

Consider a source program, where p appears only once.

- If $S <:_n T$ then $\langle T \Leftarrow S \rangle^p s \not\longrightarrow \texttt{blame}\ p$.
- If $T <:_n S$ then $\langle T \Leftarrow S \rangle^p s \not\longrightarrow \texttt{blame}\ \bar{p}$.

And there’s more!

Merging casts

Three-place cast ($R <:_n S, R <:_n T$):

$$\langle T \overset{R}{\Leftarrow} S \rangle^p \; s = \langle T \Leftarrow R \rangle^p \; \langle R \Leftarrow S \rangle^p \; s$$

Greatest-lower bound:

$$\mathtt{Dyn} \wedge S = S = S \wedge \mathtt{Dyn}$$

$$(S \to T) \wedge (S' \to T') = (S \wedge S') \to (T \wedge T')$$

$$\{x : B \mid s\} \wedge \{x : B \mid t\} = \{x : B \mid s \wedge t\}$$

Every cast is a three-way cast:

$$\langle T \Leftarrow S \rangle^p \; s = \langle T \overset{S \wedge T}{\Leftarrow} S \rangle^p \; s$$

Two adjacent three-place casts can be merged:

$$\langle U \overset{R}{\Leftarrow} T \rangle^p \; \langle T \overset{Q}{\Leftarrow} S \rangle^p \; s = \langle U \overset{Q \wedge R}{\Leftarrow} S \rangle^p \; s$$

Conclusion

A new slogan for type safety

Milner (1978):

Well-typed programs can't go wrong.

Harper; Felleisen and Wright (1994):

Well-typed programs don't get stuck.

Wadler and Findler (2008):

Well-typed programs can't be blamed.