On Impossibility of Simple Modular Translations of Concurrent Calculi

Manfred Schmidt-Schauß
Goethe-University Frankfurt

David Sabel
LMU Munich

WPTE 2020
June 29, 2020
We are interested in the **correctness of translations** between programming languages \( \tau \). In particular, we consider **concurrent** programming languages. We focus correctness w.r.t. **observational semantics**.

Motivations for considering these questions:
- **expressivity**: can language B express language A?
- **correctness of implementations**: is the implementation of concurrency primitives of A in language B correct?
Motivation and Overview of this Work

- **open problem** in previous work:
  - is there a particular small correct translation from the $\pi$-calculus into Concurrent Haskell?
- the **conjecture** was that such a translation does not exist, but we did not find a proof
Motivation and Overview of this Work

- **open problem** in previous work:
  - is there a particular small correct translation from the $\pi$-calculus into Concurrent Haskell?
- the **conjecture** was that such a translation **does not exist**, but we did not find a proof

**In this work:**

- we prove the conjecture
- method: consider a simpler problem using simpler languages
- we show impossibility of a correct translation for the simple languages
- this implies impossibility of a correct translation for the original problem
In previous work, we analyzed translations from the $\pi$-calculus to Concurrent Haskell
The Original Problem

In previous work, we analyzed translations from the $\pi$-calculus to Concurrent Haskell.

$\pi$-calculus with Stop

- process calculus
- message-passing model
- synchronous communication
- sending message $z$ over channel $x$:

$$\overline{xz}.P \mid x(y).Q \rightarrow P \mid Q[z/y]$$

- Stop-constant to signal success

CH (core language of Concurrent Haskell)

- functional language extended by threads and MVars for communication and synchronization
- shared-memory model
- MVars are one-place buffers: full or empty
- monadic operations on MVars:
  - $\text{takeMVar} \ x \mid x \text{m} \ e \rightarrow \text{return} \ e \mid x \text{m} -$
  - $\text{putMVar} \ x \ e \mid x \text{m} - \rightarrow \text{return} \ () \mid x \text{m} e$
  - $\text{takeMVar} \ x \mid x \text{m} - \text{blocks}$
  - $\text{putMVar} \ x \ e \mid x \text{m} e \text{blocks}$
The Original Problem (2)

Our correct translation encodes communication $\overline{xz}.P \parallel x(y).Q \rightarrow P \parallel Q[z/y]$ using

- one MVar for exchanging the message
- two additional check-MVars for synchronization
- check-MVar: MVar with content ()

**Conjecture [SSS2020]**

Two check-MVars are necessary.

In this work: we prove the conjecture, by transferring the problem:
The Simple Language: PISIMPLE

Subprocesses: \( U ::= 0 \) (silent process) \\
| 1 (success) \\
| \( !U \) (output) \\
| \( ?U \) (input)

Processes: \( P ::= U \mid U \mid P \) (parallel composition) \\
where \( \mid \) is associative and commutative and \( 0 \mid P \equiv P \)

Operational semantics: \( !U_1 \mid ?U_2 \mid P \xrightarrow{PIS} U_1 \mid U_2 \mid P \)

Successful process: \( 1 \mid P \)

Examples: \( ?!0 \mid !!1 \mid ?0 \xrightarrow{PIS} !0 \mid !1 \mid ?0 \xrightarrow{PIS} 0 \mid !1 \mid 0 \) not successful \\
\( ?!0 \mid !!1 \mid ?0 \xrightarrow{PIS} ?!0 \mid !1 \mid 0 \xrightarrow{PIS} !0 \mid 1 \mid 0 \) successful
The Simple Languages: CHSIMPLE

Subprocesses $U ::= 0$ (silent process) |
| $1$ (success) |
| $SU$ (send) |
| $RU$ (receive) |
| $PU$ (put) |
| $TU$ (take) 

Processes: $P ::= U | U \mid P$ (parallel composition) 
where $\mid$ is associative and commutative and $0 \mid P \equiv P$

State: $(P, M_1, M_2)$ where $M_1, M_2 \in \{\text{full, } \emptyset\}$ 
$M_1$ is the send-receive-MVar, 
$M_2$ is the check-MVar
Successful state: \((1 \mid \mathcal{P}, M_1, M_2)\)

Operational Semantics:

\[
\begin{align*}
(SU \mid \mathcal{P}, \emptyset, M_2) & \xrightarrow{CS} (U \mid \mathcal{P}, full, M_2) \\
(RU \mid \mathcal{P}, full, M_2) & \xrightarrow{CS} (U \mid \mathcal{P}, \emptyset, M_2) \\
(PU \mid \mathcal{P}, M_1, \emptyset) & \xrightarrow{CS} (U \mid \mathcal{P}, M_1, full) \\
(TU \mid \mathcal{P}, M_1, full) & \xrightarrow{CS} (U \mid \mathcal{P}, M_1, \emptyset)
\end{align*}
\]

Example:

\[
(\text{ST}0 \mid \text{RP}1, \emptyset, \emptyset) \xrightarrow{CS} (\text{T}0 \mid \text{RP}1, full, \emptyset) \xrightarrow{CS} (\text{T}0 \mid \text{P}1, \emptyset, \emptyset) \xrightarrow{CS} (\text{T}0 \mid 1, \emptyset, full) \text{ success}
\]
A modular translation $\tau : \text{PISIMPLE} \rightarrow \text{CHSIMPLE}$ is a homomorphism on the languages, and defined by the mappings:

$$
\tau(!) = s_{out} \quad \tau(?) = r_{in} \quad \tau(|) = | \quad \tau(0) = 0 \quad \tau(1) = 1
$$

where $s_{out}$ is a string over $\{P, T, S\}$, and $r_{in}$ is a string over $\{P, T, R\}$.

$\tau$ is an SRU-translation iff

- $s_{out}$ contains exactly one occurrence of $S$ and
- $r_{in}$ contains exactly one occurrence of $R$

A modular translation can be described by a translation pair $(\tau(!), \tau(?)) = (s_{out}, r_{in})$

Example: $(\tau(!), \tau(?)) = (SPP, RTT)$

Then, for instance $\tau(!0 | ?!1 | !0) = SPPRTT0 | RTTSPP0 | SPP0$
Correctness w.r.t. Observational Semantics

Observations: May- and Should-Convergence

PISIMPLE-process $P$ is

- may-convergent iff $P \xrightarrow{PIS,*} 1 \mid P'$
- should-convergent iff $\forall P' : P \xrightarrow{PIS,*} P' \implies P'$ is may-convergent

Analogous notions are defined for CHSIMPLE processes $P$ using $\xrightarrow{CS}$.

Correctness of Translations

A translation $\tau$ is correct, if it is convergence equivalent, i.e. for all $P \in$ PISIMPLE:

- $P$ is may-convergent iff $\tau(P)$ is may-convergent, and
- $P$ is should-convergent iff $\tau(P)$ is should-convergent.
Examples

Example 1: Let $\tau(!) = S$, $\tau(?) = R$

- the process $!?1$ is deadlocked in PISIMPLE
- $\tau(!?1) = SR1$ is should-convergent in CHSIMPLE:
  $(SR1, \emptyset, \emptyset) \xrightarrow{CS} (R1, full, \emptyset) \xrightarrow{CS} (1, full, \emptyset)$
- thus $\tau$ is not correct

Example 2: Let $\tau(!) = SPP$, $\tau(?) = RTT$.

- a smallest counter-example for correctness is $!0 | ?0 | !?!1$
- neither may- nor should-convergent (and thus must-divergent) in PISIMPLE
- translation $SPP0 | RTT0 | SPPRTTSPP1$ is may-convergent in CHSIMPLE:
  order of command-execution:
  
  $S P P 0 | R T T 0 | S P P R T T S P P 1$
  
  6 9 3 4 13 1 2 5 7 8 10 11 12 14
Main Result: Impossibility of a Correct Translation

Main Theorem
There are no modular correct SRU-translations from PISIMPLE into CHSIMPLE.

Proof: Illustrated in the remainder of the talk.

Corollary
There are no modular correct translations from the pi-calculus with Stop into $CH$, where the translations uses only one check-MVar per channel.

This holds, since a correct translation could be transformed into a correct SRU-translation from PISIMPLE to CHSIMPLE which does not exist.
Refuting Correctness of All SRU-Translations

The proof of impossibility is supported by our implemented tool:

**Refute-Regex** ([https://gitlab.com/davidsabel/refute-regex](https://gitlab.com/davidsabel/refute-regex))

- can execute PISIMPLE and CHSIMPLE programs
- can refute correctness of translations by searching for counter-examples
- can refute whole sets of translations represented by regular expressions
  (by executing prefixes of the translations and partial unfolding of the regular expressions)
- regular expressions are built by
  \( \lambda, P, T, S, R, 0, 1, w_1w_2, w^+, w^*, w_1|w_2, M \) for “more” (representing \((P|T)^*\))
- uses an external regex library to check containment of regular expressions
Some general properties of correct SRU-translations $\tau$ are used in all other proofs:

- The number of $P$-s is the same as the number of $T$-s in the multiset-union $\tau(!) \cup \tau(?)$.

- $\tau(!) \parallel \tau(?)$ can be executed without any deadlock until the process is empty.

- There are no correct translations $\tau$ with $|\tau(!)| + |\tau(?)| \leq 10$
  
  (this is shown by Refute-Regex, 12193 translations are refuted, using 10 counter-example processes)
Fix the notation for an SRU-translation $\tau(!) = s_1 S s_2$ and $\tau(?) = r_1 R r_2$.

The proof argues on the form of the prefixes $s_1$ and $r_1$.
Outline of the Proof (2)

Fix the notation for an SRU-translation $\tau(!) = s_1 S s_2$ and $\tau(?) = r_1 R r_2$.

The proof argues on the form of the prefixes $s_1$ and $r_1$

Initially, everything is possible.

allowed forms for $s_1, r_1$:

$s_1, r_1 \in \{P, T\}^*$
Outline of the Proof (2)

Fix the notation for an SRU-translation \( \tau(!) = s_1 S s_2 \) and \( \tau(?) = r_1 R r_2 \).

The proof argues on the form of the prefixes \( s_1 \) and \( r_1 \)

Initially, everything is possible.

**Proposition:** If \( \tau \) is correct, then neither \( PP \) nor \( TT \)
occurs in \( s_1 \) or \( r_1 \)

Allowed forms for \( s_1, r_1 \):

\[ s_1, r_1 \in \{P, T\}^* \]

Proof uses generic counter-example processes of the form

\[
\begin{align*}
!1 & \mid \ldots \mid !1 & \mid ?0 \\
\text{sufficiently many copies of } !1 \\
\end{align*}
\]

and

\[
\begin{align*}
?1 & \mid \ldots \mid ?1 & \mid !0 \\
\text{sufficiently many copies of } ?1 \\
\end{align*}
\]
Outline of the Proof (2)

Fix the notation for an SRU-translation \( \tau(!) = s_1 S s_2 \) and \( \tau(?) = r_1 R r_2 \).

The proof argues on the form of the prefixes \( s_1 \) and \( r_1 \)

Initially, everything is possible.

Proposition: If \( \tau \) is correct, then neither \( PP \) nor \( TT \) occurs in \( s_1 \) or \( r_1 \)

allowed forms for \( s_1, r_1 \):

\[
s_1, r_1 \in \{P, T\}^*
\]

\[
s_1, r_1 \in \{(PT)^*, (TP)^*, (PT)^*P, (TP)^*T\}
\]
Fix the notation for an SRU-translation \( \tau (!) = s_1 S s_2 \) and \( \tau (?) = r_1 R r_2 \).

The proof argues on the form of the prefixes \( s_1 \) and \( r_1 \)

Initially, everything is possible.

Proposition: If \( \tau \) is correct, then neither \( PP \) nor \( TT \) occurs in \( s_1 \) or \( r_1 \)

Proposition: If \( \tau \) is correct, \( s_1 \notin \{(PT)^nP, (TP)^nT\} \), and \( r_1 \notin \{(PT)^nP, (TP)^nT\} \)

allowed forms for \( s_1, r_1 \):

\[
\begin{align*}
  s_1, r_1 & \in \{P, T\}^* \\
  s_1, r_1 & \in \{(PT)^*, (TP)^*, (PT)^*P, (TP)^*T\}
\end{align*}
\]
Outline of the Proof (2)

Fix the notation for an SRU-translation $\tau(!) = s_1 S s_2$ and $\tau(?) = r_1 R r_2$.

The proof argues on the form of the prefixes $s_1$ and $r_1$

Initially, everything is possible.

**Proposition:** If $\tau$ is correct, then neither $PP$ nor $TT$ occurs in $s_1$ or $r_1$

**Proposition:** If $\tau$ is correct, $s_1 \not\in \{(PT)^nP, (TP)^nT\}$, and $r_1 \not\in \{(PT)^nP, (TP)^nT\}$

allowed forms for $s_1, r_1$:

$s_1, r_1 \in \{P, T\}^*$

$s_1, r_1 \in \{(PT)^*, (TP)^*, (PT)^*P, (TP)^*T\}$

$s_1, r_1 \in \{(PT)^*, (TP)^*\}$
Fix the notation for an SRU-translation $\tau(!) = s_1 S s_2$ and $\tau(?) = r_1 R r_2$.

The proof argues on the form of the prefixes $s_1$ and $r_1$ allowed forms for $s_1, r_1$:

$s_1, r_1 \in \{P, T\}^*$

$s_1, r_1 \in \{(PT)^n P, (TP)^n T\}$

$s_1, r_1 \in \emptyset$

Initially, everything is possible.

Proposition: If $\tau$ is correct, then neither $PP$ nor $TT$ occurs in $s_1$ or $r_1$.

Proposition: If $\tau$ is correct, $s_1 \not\in \{(PT)^n P, (TP)^n T\}$, and $r_1 \not\in \{(PT)^n P, (TP)^n T\}$.

Proposition: $\tau$ is not correct for the translation patterns

- $\tau(!) = (PT)^n S P^k s_3$ and $\tau(?) = RT^h r_3$, where $n \geq 0$, $h, k \geq 2$, $h + k \geq 5$, $s_3$ does not start with $P$, $r_3$ does not start with $T$. Then $\tau$ is not correct.

- $\tau(!) = (PT)^n ST^k s_3$ and $\tau(?) = RP^h r_3$, where $n \geq 0$, $h, k \geq 2$. Then $\tau$ is not correct.
Fix the notation for an SRU-translation $\tau(!) = s_1 S s_2$ and $\tau(?) = r_1 R r_2$.

The proof argues on the form of the prefixes $s_1$ and $r_1$

Initially, everything is possible.

Proposition: If $\tau$ is correct, then neither $PP$ nor $TT$ occurs in $s_1$ or $r_1$

Proposition: If $\tau$ is correct, $s_1 \notin \{(PT)^n P, (TP)^n T\}$, and $r_1 \notin \{(PT)^n P, (TP)^n T\}$

Proposition: $\tau$ is not correct for the translation patterns

- $\bullet \tau(!) = (PT)^n S s_2$ and $\tau(?) = (PT)^m R r_2$, $s_1,r_1 \in \emptyset$
- $\bullet \tau(!) = (PT)^n S s_2$ and $\tau(?) = (TP)^m R r_2$,
- $\bullet \tau(!) = (TP)^n S s_2$ and $\tau(?) = (PT)^m R r_2$,
- $\bullet \tau(!) = (TP)^n S s_2$ and $\tau(?) = (TP)^m R r_2$, allowed forms for $s_1, r_1$:

$s_1, r_1 \in \{P, T\}^*$

$s_1, r_1 \in \{(PT)^*, (TP)^*, (PT)^* P, (TP)^* T\}$

$s_1, r_1 \in \{(PT)^*, (TP)^*\}$

Proof uses generic counter-example processes of the form

\[ \vdots \]

Proof uses the lemmas:

Lemma: Let $\tau(!) = (PT)^n S s_2$ and $\tau(?) = (TP)^m R r_2$, where $n \geq 0$, $h,k \geq 2$, $h+k \geq 5$, $s_3$ does not start with $P$, $r_3$ does not start with $T$. Then $\tau$ is not correct.

Lemma: Let $\tau(!) = (PT)^n S s_2$ and $\tau(?) = (TP)^m R r_2$, where $n \geq 0$, $h,k \geq 2$. Then $\tau$ is not correct.
Variants of CHSIMPLE

**Theorem**
There are no correct PT-only translations, where in PT-only translation no $S$ and $R$ are permitted.

Proof: Similar case-distinction as in the previous proof.
Variants of CHSIMPLE

**Theorem**

There are no correct PT-only translations, where in PT-only translation no $S$ and $R$ are permitted.

Proof: Similar case-distinction as in the previous proof

**Theorem (Correct Translations)**

Let $\text{CHSIMPLE}_i$ be like CHSIMPLE, but with $i$ copies of $P, T$ (each with their own MVar)

- A correct modular SRU-translation from $\text{PISIMPLE} \rightarrow \text{CHSIMPLE}_2$ is
  $$\tau(!) = P_1ST_2T_1 \text{ and } \tau(?) = RP_2.$$

- A correct modular PT-only translation from $\text{PISIMPLE} \rightarrow \text{CHSIMPLE}_3$ is
  $$\tau(!) = P_1P_3T_2T_1 \text{ and } \tau(?) = T_3P_2.$$
Conclusion

- solved an open question on the existence/nonexistence of correct modular translations from the pi-calculus into CH, with special question on the number of check-MVars

- two check-MVars are sufficient, one is insufficient

- seems to be a sharp boundary between synchronous and asynchronous communication in concurrent calculi

Future work

- consider further cases and variations

- formulate the result more independent from CH, perhaps replace MVars by locks?