On Impossibility of Simple Modular Translations of Concurrent Calculi

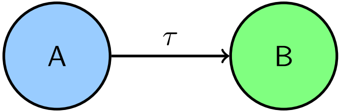
Manfred Schmidt-Schauß
Goethe-University Frankfurt

David Sabel
LMU Munich

WPTE 2020
June 29, 2020

General Motivation

- We are interested in the **correctness of translations** between programming languages



- 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 π-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 π-calculus into Concurrent Haskell?

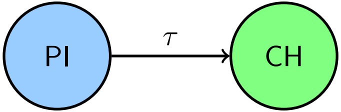
- the **conjecture** was that such a translation **does not exist**,
  but we did not find a proof

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

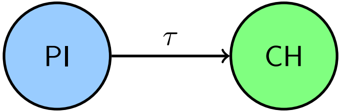
The Original Problem

In previous work, we analyzed translations from the π-calculus to Concurrent Haskell



The Original Problem

In previous work, we analyzed translations from the π-calculus to Concurrent Haskell



π-calculus with Stop

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

$$\underbrace{\overline{x}z.P}_{\text{sender}} \mid \underbrace{x(y).Q}_{\text{receiver}} \to P \mid Q[z/y]$$

- 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:
  - $\texttt{takeMVar}\ x \mid x\,\mathbf{m}\,e \to \texttt{return}\ e \mid x\,\mathbf{m}\,-$
  - $\texttt{putMVar}\ x\ e \mid x\,\mathbf{m}\,- \to \texttt{return}\ () \mid x\,\mathbf{m}\,e$
  - $\texttt{takeMVar}\ x \mid x\,\mathbf{m}\,-$ blocks
  - $\texttt{putMVar}\ x\ e \mid x\,\mathbf{m}\,e$ blocks

The Original Problem (2)

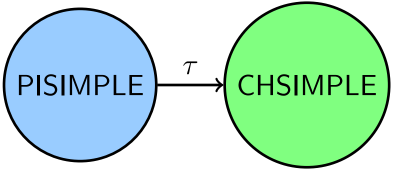
Our correct translation encodes communication $\overline{x}z.P \mid x(y).Q \to P \mid Q[z/y]$ using

- 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	$\mathcal{U}$	$::=$	0	(silent process)
		$\mid$	1	(success)
		$\mid$	$!\mathcal{U}$	(output)
		$\mid$	$?\mathcal{U}$	(input)

Processes: $\mathcal{P} ::= \mathcal{U} \mid \mathcal{U} \,\|\, \mathcal{P}$ (parallel composition)

where $\|$ is associative and commutative and $0 \,\|\, \mathcal{P} \equiv \mathcal{P}$

Operational semantics: $!\mathcal{U}_1 \,\|\, ?\mathcal{U}_2 \,\|\, \mathcal{P} \xrightarrow{PIS} \mathcal{U}_1 \,\|\, \mathcal{U}_2 \,\|\, \mathcal{P}$

Successful process: $1 \,\|\, \mathcal{P}$

Examples: $?!0 \,\|\, !!1 \,\|\, ?0 \xrightarrow{PIS} !0 \,\|\, !1 \,\|\, ?0 \xrightarrow{PIS} 0 \,\|\, !1 \,\|\, 0$ not successful

$?!0 \,\|\, !!1 \,\|\, ?0 \xrightarrow{PIS} ?!0 \,\|\, !1 \,\|\, 0 \xrightarrow{PIS} !0 \,\|\, 1 \,\|\, 0$ successful

The Simple Languages: CHSIMPLE

Subprocesses	$\mathcal{U}$	$::=$	0	(silent process)
		$\mid$	1	(success)
		$\mid$	$S\mathcal{U}$	(send)
		$\mid$	$R\mathcal{U}$	(receive)
		$\mid$	$P\mathcal{U}$	(put)
		$\mid$	$T\mathcal{U}$	(take)

Processes: $\mathcal{P} ::= \mathcal{U} \mid \mathcal{U} \,\mathbf{|}\, \mathcal{P}$ (parallel composition)

where $\mathbf{|}$ is associative and commutative and $0 \,\mathbf{|}\, \mathcal{P} \equiv \mathcal{P}$

State: $(\mathcal{P}, M_1, M_2)$ where $M_1, M_2 \in \{full, \emptyset\}$

M_1 is the send-receive-MVar,

M_2 is the check-MVar

The Simple Languages: CHSIMPLE (2)

Successful state: $(1 \mid \mathcal{P}, M_1, M_2)$

Operational Semantics:

$$\begin{array}{lcl}
(S\mathcal{U} \mid \mathcal{P}, \emptyset, M_2) & \xrightarrow{CS} & (\mathcal{U} \mid \mathcal{P}, full, M_2) \\
(R\mathcal{U} \mid \mathcal{P}, full, M_2) & \xrightarrow{CS} & (\mathcal{U} \mid \mathcal{P}, \emptyset, M_2) \\
(P\mathcal{U} \mid \mathcal{P}, M_1, \emptyset) & \xrightarrow{CS} & (\mathcal{U} \mid \mathcal{P}, M_1, full) \\
(T\mathcal{U} \mid \mathcal{P}, M_1, full) & \xrightarrow{CS} & (\mathcal{U} \mid \mathcal{P}, M_1, \emptyset)
\end{array}$$

Example:

$(ST0 \mid RP1, \emptyset, \emptyset) \xrightarrow{CS} (T0 \mid RP1, full, \emptyset) \xrightarrow{CS} (T0 \mid P1, \emptyset, \emptyset) \xrightarrow{CS} (T0 \mid 1, \emptyset, full)$ success

Simple Modular Translations

A modular translation τ : PISIMPLE $\rightarrow$ CHSIMPLE is a homomorphism on the languages, and defined by the mappings:

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

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

τ 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 $\tau(!?0 \mid ?!1 \mid !0) = SPPRTT0 \mid RTTSPP0 \mid SPP0$

Correctness w.r.t. Observational Semantics

Observations: May- and Should-Convergence

PISIMPLE-process $\mathcal{P}$ is

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

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

Correctness of Translations

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

- $\mathcal{P}$ is may-convergent iff $\tau(\mathcal{P})$ is may-convergent, and
- $\mathcal{P}$ is should-convergent iff $\tau(\mathcal{P})$ 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 τ is not correct

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

- a smallest counter-example for correctness is $!0 \mid ?0 \mid !?!1$
- neither may- nor should-convergent (and thus must-divergent) in PISIMPLE
- translation $SPP0 \mid RTT0 \mid SPPRTTSPP1$ is may-convergent in CHSIMPLE:
  order of command-execution:

$$\begin{array}{cccccccccccccccccc} S & P & P & 0 \mid & R & T & T & 0 \mid & S & P & P & R & T & T & S & P & P & 1 \\ 6 & 9 & & & 3 & 4 & 13 & & 1 & 2 & 5 & 7 & 8 & 10 & 11 & 12 & 14 & \end{array}$$

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`)

- 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 λ,P,T,S,R,0,1,$w_1 w_2$,w^+,w^*,$w_1|w_2$, M for "more" (representing $(P|T)^*$)
- uses an external regex library to check containment of regular expressions

Outline of the Proof

Some general properties of correct SRU-translations τ are used in all other proofs:

- The number of P-s is the same as the number of T-s in the multiset-union $\tau(!) \cup \tau(?)$.
- $\tau(!)$ | $\tau(?)$ can be executed without any deadlock until the process is empty.
- There are no correct translations τ with $|\tau(!)| + |\tau(?)| \leq 10$

  (this is shown by **Refute-Regex**, 12193 translations are refuted, using 10 counter-example processes)

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

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

allowed forms for s_1, r_1:

Initially, everything is possible. $s_1, r_1 \in \{P, T\}^*$

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

allowed forms for s_1, r_1:

Initially, everything is possible.

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

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

Proof uses generic counter-example processes of the form

$\underbrace{!1 \mid \ldots \mid !1}_{\text{sufficiently many copies of } !1} \mid ?0$ and $\underbrace{?1 \mid \ldots \mid ?1}_{\text{sufficiently many copies of } ?1} \mid !0$

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

	allowed forms for s_1, r_1:
Initially, everything is possible.	$s_1, r_1 \in \{P, T\}^*$
Proposition: If τ is correct, then neither PP nor TT occurs in s_1 or r_1	$s_1, r_1 \in \{(PT)^*, (TP)^*, (PT)^*P, (TP)^*T\}$

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

	allowed forms for s_1, r_1:
Initially, everything is possible.	$s_1, r_1 \in \{P, T\}^*$
Proposition: If τ is correct, then neither PP nor TT occurs in s_1 or r_1	$s_1, r_1 \in \{(PT)^*, (TP)^*, (PT)^*P, (TP)^*T\}$
Proposition: If τ is correct, $s_1 \notin \{(PT)^n P, (TP)^n T\}$, and $r_1 \notin \{(PT)^n P, (TP)^n T\}$	

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

	allowed forms for s_1, r_1:
Initially, everything is possible.	$s_1, r_1 \in \{P, T\}^*$
Proposition: If τ is correct, then neither PP nor TT occurs in s_1 or r_1	$s_1, r_1 \in \{(PT)^*, (TP)^*, (PT)^*P, (TP)^*T\}$
Proposition: If τ is correct, $s_1 \notin \{(PT)^n P, (TP)^n T\}$, and $r_1 \notin \{(PT)^n P, (TP)^n T\}$	$s_1, r_1 \in \{(PT)^*, (TP)^*\}$

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

Initi

Prop

Prop

and $r_1 \not\in \{(PT)$ ... $, (TP)$ ... $T\}$

Proof uses the lemmas:

Lemma: Let $\tau(!) = (PT)^n S P^k s_3$ and $\tau(?) = R T^h r_3$, 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 τ is not correct.

Lemma: Let $\tau(!) = (PT)^n S T^k s_3$ and $\tau(?) = R P^h r_3$, where $n \geq 0$, $h, k \geq 2$. Then τ is not correct.

Proposition: τ is not correct for the translation patterns

- $\tau(!) = (PT)^n S s_2$ and $\tau(?) = (PT)^m R r_2$,
- $\tau(!) = (PT)^n S s_2$ and $\tau(?) = (TP)^m R r_2$,
- $\tau(!) = (TP)^n S s_2$ and $\tau(?) = (PT)^m R r_2$,
- $\tau(!) = (TP)^n S s_2$ and $\tau(?) = (TP)^m R r_2$,

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

allowed forms for s_1, r_1:

Initially, everything is possible. $s_1, r_1 \in \{P, T\}^*$

Proposition: If τ is correct, then neither PP nor TT occurs in s_1 or r_1 $s_1, r_1 \in \{(PT)^*, (TP)^*, (PT)^*P, (TP)^*T\}$

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

Proposition: τ is not correct for the translation patterns

- $\tau(!) = (PT)^n S s_2$ and $\tau(?) = (PT)^m R r_2$,
- $\tau(!) = (PT)^n S s_2$ and $\tau(?) = (TP)^m R r_2$,
- $\tau(!) = (TP)^n S s_2$ and $\tau(?) = (PT)^m R r_2$,
- $\tau(!) = (TP)^n S s_2$ and $\tau(?) = (TP)^m R r_2$,

$s_1, r_1 \in \emptyset$ □

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 CHSIMPLE_i be like CHSIMPLE, but with i copies of P, T (each with their own MVar)

- A correct modular SRU-translation from PISIMPLE $\to$ CHSIMPLE_2 is $\tau(!) = P_1 S T_2 T_1$ and $\tau(?) = R P_2$.
- A correct modular PT-only translation from PISIMPLE $\to$ CHSIMPLE_3 is $\tau(!) = P_1 P_3 T_2 T_1$ and $\tau(?) = T_3 P_2$.

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?