Proofs about Network Communication: For Humans and Machines

Wolfgang Jeltsch    Javier Díaz

16th Interaction and Concurrency Experience

Lisbon, Portugal
19 June 2023
Concurrent and distributed systems are often safety-critical

Machine-checked proofs can provide a high degree of assurance

Our research program:
- Targets verification of design refinements
- Centers on the Ouroboros blockchain consensus protocols
- Uses the Isabelle proof assistant

Previous achievement:
- A machine-checked correctness proof of broadcast via multicast

Issue with this proof:
- Relies on fundamental but unproved bisimilarity statements

Now we are delivering the missing proofs

And show you how to conduct such proofs so that they are:
- Concise
- Human-friendly
- Machine-checked
The $\Pi$-calculus

- A process calculus
- Our language for describing concurrent and distributed systems
  - Protocol specifications
  - Protocol implementations
  - Protocol environments
- Developed by us as part of our research program
- Key properties:
  - General
  - Minimal
  - Suitable for machine-checked proofs
- Similar to the asynchronous $\pi$-calculus
- Additional features:
  - Arbitrary data
  - Computation
  - Conditional execution
- Embedded in Isabelle/HOL
The $P$-calculus in detail

- **Processes:**
  - $0$ does nothing
  - $a \triangleleft x$ sends value $x$ to channel $a$
  - $a \triangleright x. P x$ receives a value $x$ from channel $a$, performs process $P x$
  - $p \parallel q$ performs processes $p$ and $q$ in parallel
  - $\nu a. P a$ introduces a local channel $a$, performs process $P a$

- Constructs capture just the key features of process calculi
  - Concurrency
  - Communication

- For other features we utilize the host language (Isabelle/HOL)
  - Using higher-order abstract syntax (HOAS)
    - Name binding
    - Arbitrary data
    - Computation
    - Conditional execution
  - Using coinduction
    - Repetition
Proofs about coinductively defined processes tend to be low-level

Solution:
- Define just a single, general repetition construct via coinduction
- Show fundamental properties of this construct for later use in proofs

Repeated receive:

\[ a \triangleright^\infty x. \ P x \] repeatedly receives values \( x \) from channel \( a \), initiates the execution of \( P x \) for each received \( x \)

Definition:

\[ a \triangleright^\infty x. \ P x = a \triangleright x. (P x \parallel a \triangleright^\infty x. P x) \]
Repeated receive idempotency

- Repeated receive is idempotent
  - With respect to parallel composition
  - Up to bisimilarity
- Formally:

\[ a \triangleright^\infty x. \ P \ x \parallel a \triangleright^\infty x. \ P \ x \sim a \triangleright^\infty x. \ P \ x \]

- This fact is used in our correctness proof of broadcast via multicast
- Its proof exemplifies the proof style we advocate here
Background of the proof of repeated receive idempotency

- **P-calculus transition rules about ∡, ▷, and ∥:**

  \[
  a \triangleleft x \xrightarrow{a \triangleleft x} 0 \quad (\triangleleft)
  \]

  \[
  a \triangleright x. P x \xrightarrow{a \triangleright x} P x \quad (\triangleright)
  \]

  \[
  p \xrightarrow{a \triangleleft x} p' \quad q \xrightarrow{a \triangleright x} q' \quad (\tau_{\rightarrow})
  \]

  \[
  p \parallel q \xrightarrow{\tau} p' \parallel q' \quad (\tau_{\rightarrow})
  \]

  \[
  p \xrightarrow{\alpha} p' \quad (\parallel_1)
  \]

  \[
  p \parallel q \xrightarrow{\alpha} p' \parallel q \quad (\parallel_2)
  \]

- **Definition of repeated receive again:**

  \[
  a \triangleright^\infty x. P x = a \triangleright x. (P x \parallel a \triangleright^\infty x. P x)
  \]
lemma repeated_receive_idempotency:
shows a ▷∞ x. P x || a ▷∞ x. P x ~ a ▷∞ x. P x

proof coinduction case (forward_simulation α s) ⟨...⟩
next case (backward_simulation α s) from ‹a ▷∞ x. P x α −→ s›
obtain x where α = a ▷ x and s = P x || a ▷∞ x. P x
⟨proof⟩ with ‹a ▷∞ x. P x α −→ s›
have a ▷∞ x. P x a ▷ x −−−→ P x || a ▷∞ x. P x
⟨proof⟩ then have a ▷∞ x. P x || a ▷∞ x. P x
a ▷ x −−−→ P x || a ▷∞ x. P x
⟨proof⟩ then show ?case ⟨proof⟩
qed
lemma repeated_receive_idempotency:
   shows $a \triangleright^\infty x. P x \parallel a \triangleright^\infty x. P x \sim a \triangleright^\infty x. P x$

proof coinduction
   case (forward_simulation $\alpha$ s)

next
   case (backward_simulation $\alpha$ s)

qed
lemma repeated_receive_idempotency:
  shows $a \triangleright \infty x. P x \parallel a \triangleright \infty x. P x \sim a \triangleright \infty x. P x$

proof coinduction
  case (forward_simulation $\alpha$ $s$)
  ⟨...⟩

next
  case (backward_simulation $\alpha$ $s$)

qed
Proving repeated receive idempotency

\textbf{lemma} \texttt{repeated\_receive\_idempotency}:  
\begin{align*}
\text{shows } a \triangleright^\infty x. \; P \; x \parallel a \triangleright^\infty x. \; P \; x & \sim a \triangleright^\infty x. \; P \; x \\
\text{proof} \text{ coinduction} & \\
\text{case (forward\_simulation } \alpha \; s) & \\
\langle \ldots \rangle \\
\text{next} & \\
\text{case (backward\_simulation } \alpha \; s) & \\
\text{from } \langle a \triangleright^\infty x. \; P \; x \xrightarrow{\alpha} s \rangle & \\
\text{obtain } x \; \text{where } \alpha = a \triangleright x \; \text{and } s = P \; x \parallel a \triangleright^\infty x. \; P \; x & \\
\langle \text{proof} \rangle & \\
\text{qed}
\end{align*}
lemma repeated_receive_idempotency:
  shows \( a \triangleright^\infty x. \ P x \parallel a \triangleright^\infty x. \ P x \sim a \triangleright^\infty x. \ P x \)

proof coinduction
  case (forward_simulation \( \alpha \ s \))
  ⟨...⟩

next
  case (backward_simulation \( \alpha \ s \))
  from ⟨\( a \triangleright^\infty x. \ P x \xrightarrow{\alpha} s \)⟩
  obtain \( x \) where \( \alpha = a \triangleright x \) and \( s = P x \parallel a \triangleright^\infty x. \ P x \)
  ⟨proof⟩
  with ⟨\( a \triangleright^\infty x. \ P x \xrightarrow{\alpha} s \)⟩ have \( a \triangleright^\infty x. \ P x \xrightarrow{a \triangleright x} P x \parallel a \triangleright^\infty x. \ P x \)
  ⟨proof⟩

qed
lemma repeated_receive_idempotency:
  shows $a \triangleright^\infty x. \ P x \parallel a \triangleright^\infty x. \ P x \sim a \triangleright^\infty x. \ P x$
proof coinduction
  case (forward_simulation $\alpha \ s$)
  ⟨...⟩
next
  case (backward_simulation $\alpha \ s$)
  from ⟨$a \triangleright^\infty x. \ P x \xrightarrow{\alpha} s$⟩
  obtain $x$ where $\alpha = a \triangleright x$ and $s = P x \parallel a \triangleright^\infty x. \ P x$
  ⟨proof⟩
  with ⟨$a \triangleright^\infty x. \ P x \xrightarrow{\alpha} s$⟩ have $a \triangleright^\infty x. \ P x \xrightarrow{a \triangleright x} P x \parallel a \triangleright^\infty x. \ P x$
  ⟨proof⟩
  then have $a \triangleright^\infty x. \ P x \parallel a \triangleright^\infty x. \ P x \xrightarrow{a \triangleright x} (P x \parallel a \triangleright^\infty x. \ P x) \parallel a \triangleright^\infty x. \ P x$
  ⟨proof⟩
qed
lemma repeated_receive_idempotency:
  shows \( a \triangleright^{\infty} x \cdot P x \parallel a \triangleright^{\infty} x \cdot P x \sim a \triangleright^{\infty} x \cdot P x \)
proof coinduction
  case (forward_simulation \( \alpha s \))
  ⟨...⟩
next
  case (backward_simulation \( \alpha s \))
  from ⟨\( a \triangleright^{\infty} x \cdot P x \xrightarrow{\alpha} s \)⟩
  obtain \( x \) where \( \alpha = a \triangleright x \) and \( s = P x \parallel a \triangleright^{\infty} x \cdot P x \)
  ⟨proof⟩
  with ⟨\( a \triangleright^{\infty} x \cdot P x \xrightarrow{\alpha} s \)⟩ have \( a \triangleright^{\infty} x \cdot P x \xrightarrow{a\triangleright x} P x \parallel a \triangleright^{\infty} x \cdot P x \)
  ⟨proof⟩
then have \( a \triangleright^{\infty} x \cdot P x \parallel a \triangleright^{\infty} x \cdot P x \xrightarrow{a\triangleright x} P x \parallel (a \triangleright^{\infty} x \cdot P x \parallel a \triangleright^{\infty} x \cdot P x) \)
  ⟨proof⟩
qed
Proving repeated receive idempotency

**lemma** repeated_receive_idempotency:

shows $a \triangleright^\infty x \cdot P x \parallel a \triangleright^\infty x \cdot P x \sim a \triangleright^\infty x \cdot P x$

**proof** coinduction

*case* (forward_simulation $\alpha$ $s$)

⟨...⟩

*next*

*case* (backward_simulation $\alpha$ $s$)

*from* ⟨$a \triangleright^\infty x \cdot P x \xrightarrow{\alpha} s$⟩

*obtain* $x$ *where* $\alpha = a \triangleright x$ *and* $s = P x \parallel a \triangleright^\infty x \cdot P x$

⟨proof⟩

*with* ⟨$a \triangleright^\infty x \cdot P x \xrightarrow{\alpha} s$⟩ *have* $a \triangleright^\infty x \cdot P x \xrightarrow{a\triangleright x} P x \parallel a \triangleright^\infty x \cdot P x$

⟨proof⟩

*then* *have* $a \triangleright^\infty x \cdot P x \parallel a \triangleright^\infty x \cdot P x \xrightarrow{a\triangleright x} (P x \parallel a \triangleright^\infty x \cdot P x) \parallel a \triangleright^\infty x \cdot P x$

⟨proof⟩

qed
Proving repeated receive idempotency

**lemma** repeated_receive_idempotency:

shows $a \triangleright^\infty x. \ P \ x \ || \ a \triangleright^\infty x. \ P \ x \sim a \triangleright^\infty x. \ P \ x$

**proof** (coinduction rule: up_to_rule [where $\mathcal{F} = \{\sim\} \sim \mathcal{M}$])

- case (forward_simulation $\alpha \ s$)

  ⟨...⟩

- next

  case (backward_simulation $\alpha \ s$)

  from $\langle a \triangleright^\infty x. \ P \ x \xrightarrow{\alpha} s \rangle$

  obtain $x$ where $\alpha = a \triangleright x$ and $s = P \ x \ || \ a \triangleright^\infty x. \ P \ x$

  ⟨proof⟩

  with $\langle a \triangleright^\infty x. \ P \ x \xrightarrow{\alpha} s \rangle$ have $a \triangleright^\infty x. \ P \ x \xrightarrow{a\triangleright x} P \ x \ || \ a \triangleright^\infty x. \ P \ x$

  ⟨proof⟩

  then have $a \triangleright^\infty x. \ P \ x \ || \ a \triangleright^\infty x. \ P \ x$ $\xrightarrow{a\triangleright x}$ $(P \ x \ || \ a \triangleright^\infty x. \ P \ x) \ || \ a \triangleright^\infty x. \ P \ x$

  ⟨proof⟩

qed
proving repeated receive idempotency

**Lemma** repeated_receive_idempotency:

shows $a \triangleright^{\infty} x. P x \parallel a \triangleright^{\infty} x. P x \sim a \triangleright^{\infty} x. P x$

**Proof** (coinduction rule: up_to_rule [where $\mathcal{F} = [\sim] \dashv \mathcal{M}$])

- case (forward_simulation $\alpha s$)

  ⟨...⟩

next

- case (backward_simulation $\alpha s$)

  from ⟨$a \triangleright^{\infty} x. P x \xrightarrow{\alpha} s$⟩

  obtain $x$ where $\alpha = a \triangleright x$ and $s = P x \parallel a \triangleright^{\infty} x. P x$

  ⟨proof⟩

  with ⟨$a \triangleright^{\infty} x. P x \xrightarrow{\alpha} s$⟩ have $a \triangleright^{\infty} x. P x \xrightarrow{a \triangleright x} P x \parallel a \triangleright^{\infty} x. P x$

  ⟨proof⟩

  then have $a \triangleright^{\infty} x. P x \parallel a \triangleright^{\infty} x. P x \xrightarrow{a \triangleright x} (P x \parallel a \triangleright^{\infty} x. P x) \parallel a \triangleright^{\infty} x. P x$

  ⟨proof⟩

**QED** respectful
lemma repeated_receive_idempotency:
  shows \( a \xrightarrow{\infty} x. P x \parallel a \xrightarrow{\infty} x. P x \sim a \xrightarrow{\infty} x. P x \)
proof (coinduction rule: up_to_rule [where \( F = [\sim] \circ M \)])
  case (forward_simulation \( \alpha s \))
  ⟨...⟩
next
  case (backward_simulation \( \alpha s \))
from ⟨\( a \xrightarrow{\infty} x. P x \xrightarrow{\alpha} s \)⟩
obtain x where \( \alpha = a \xrightarrow{\infty} x \) and \( s = P x \parallel a \xrightarrow{\infty} x. P x \)
  ⟨proof⟩
with ⟨\( a \xrightarrow{\infty} x. P x \xrightarrow{\alpha} s \)⟩ have \( a \xrightarrow{\infty} x. P x \xrightarrow{a\xrightarrow{\infty}x} P x \parallel a \xrightarrow{\infty} x. P x \)
  ⟨proof⟩
then have \( a \xrightarrow{\infty} x. P x \parallel a \xrightarrow{\infty} x. P x \xrightarrow{a\xrightarrow{\infty}x} (P x \parallel a \xrightarrow{\infty} x. P x) \parallel a \xrightarrow{\infty} x. P x \)
  ⟨proof⟩
then show ?case
  ⟨proof⟩
qed respectful
Tools for bisimulation proofs for humans and machines

- The Isabelle/Isar proof language
  - Closer to usual mathematics than proof terms and tactics scripts
  - Still precise and amenable to machine-checking
- A formalized algebra of “up to” methods
  - Concise bisimulation proofs that are machine-checked
  - Simple construction of custom “up to” methods
- Isabelle’s coinduction proof method
  - Structured coinductive proofs
  - Integration of “up to” methods via custom coinduction rules
- Higher-order abstract syntax
  - Less dealing with boring technicalities in proofs
Follow the development

- https://github.com/input-output-hk/equivalence-reasoner
- https://github.com/input-output-hk/transition-systems
- https://github.com/input-output-hk/thorn-calculus