Coinduction All the Way Up

Damien Pous

Plume group, CNRS, ENS Lyon

Program Equivalence

Through equational reasoning

\[
\begin{align*}
!a \cdot a & = !(a \mid a) & (a \cdot a = a \mid a) \\
& = !a \mid !a & (!P \mid Q) = !P \mid !Q \\
& = !a & (!P \mid !P = !P)
\end{align*}
\]

(Axiomatic, syntax-based, inductive)
Program Equivalence

Through bisimulations

\[ \mathcal{R} \triangleq \{ \langle P \mid Q, Q \mid P \rangle \mid \forall P, Q \} \]

(Behavioural, LTS-based, coinductive)
Program Equivalence

Through bisimulations

\[ R \triangleq \{ \langle P | Q, Q | P \rangle \mid \forall P, Q \} \]

\[
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
P \mid Q \quad R
\end{array}
\end{array}
\end{array}
\rightarrow
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
(\nu a)(P' \mid Q') \quad R
\end{array}
\end{array}
\end{array}
\]
Program Equivalence

Through bisimulations

\[ \mathcal{R} \triangleq \{ \langle (\nu \tilde{a})(P \mid Q), (\nu \tilde{a})(Q \mid P) \mid \forall \tilde{a}, P, Q \rangle \} \]

\[
\begin{array}{ccc}
P \mid Q & \mathcal{R} & Q \mid P \\
\tau \downarrow & \mathcal{R} & \tau \downarrow \\
(\nu a)(P' \mid Q') & (\nu a)(Q' \mid P') & \mathcal{R}
\end{array}
\]
Program Equivalence

Through bisimulations up-to

$$\mathcal{R} \triangleq \{ \langle P \mid Q, Q \mid P \rangle \mid \forall P, Q \}$$

$$P \mid Q \xrightarrow{\tau} (\nu a)(P' \mid Q') \xrightarrow{\mathcal{R}} (\nu a)(Q' \mid P') \xrightarrow{\mathcal{R}} (\nu a)(\mathcal{R})$$
Program Equivalence

Through bisimulations up-to

\[ \mathcal{R} \triangleq \{ \langle !(P \mid Q), !P \mid !Q \rangle \} \]
Program Equivalence

Through bisimulations up-to

\[ \mathcal{R} \triangleq \{ \langle !(P \mid Q), !P \mid !Q \rangle \} \]

\[
\begin{array}{ccc}
!(P \mid Q) & \mathcal{R} & !P \mid !Q \\
\downarrow^{\alpha} & & \downarrow^{\alpha} \\
!(P \mid Q) \mid (P' \mid Q) & \mathcal{R} & (P' \mid P') \mid !Q
\end{array}
\]
Program Equivalence

Through bisimulations up-to

\[ R \triangleq \{ \langle !(P \mid Q), !P \mid !Q \rangle \} \]

\[
\begin{array}{ccc}
!(P \mid Q) & R & !P \mid !Q \\
\alpha \downarrow & & \alpha \downarrow \\
!(P \mid Q) \mid (P' \mid Q) & R & (P \mid P') \mid !Q \\
& & \sim \\
& & (P \mid P') \mid (Q \mid Q)
\end{array}
\]
Program Equivalence

Through bisimulations up-to

\[ \mathcal{R} \triangleq \{ \langle !(P \mid Q), !P \mid !Q \rangle \} \]

\[
\begin{array}{c}
!(P \mid Q) \quad \mathcal{R} \quad !P \mid !Q \\
\downarrow \alpha \\
!(P \mid Q) \mid (P' \mid Q) \quad \mathcal{R} \quad (!P \mid P') \mid !Q \\
\sim \\
(!P \mid P') \mid (!Q \mid Q) \\
\sim \\
(!P \mid !Q) \mid (P' \mid Q)
\end{array}
\]
Program Equivalence

Through bisimulations up-to

\[ \mathcal{R} \triangleq \{ \langle ! (P \mid Q), ! P \mid ! Q \rangle \} \]

\[ !(P \mid Q) \quad \mathcal{R} \quad ! P \mid ! Q \]

\[ \alpha \downarrow \quad \alpha \downarrow \]

\[ !(P \mid Q) \mid (P' \mid Q) \quad \mathcal{R} \quad (P \mid P') \mid ! Q \]

\[ \sim \]

\[ C(\mathcal{R}) \]

\[ (P \mid Q) \mid !(P' \mid Q) \sim (P' \mid Q) \]

\[ (P \mid P') \mid (Q) \langle ! P \mid ! Q \rangle \]
Program Equivalence

Through bisimulations up-to

\[ \mathcal{R} \triangleq \{ \langle !(P \mid Q), !P \mid !Q \rangle \} \]

\[ \begin{align*}
!(P \mid Q) \quad \mathcal{R} \quad !P \mid !Q \\
\alpha \downarrow \quad \alpha \downarrow \\
!(P \mid Q) \mid (P' \mid Q) \quad \mathcal{R} \quad (!P \mid P') \mid !Q \\
\end{align*} \]

\[ C(\mathcal{R}) \]

\[ \sim \]

\[ (!P \mid P') \mid (!Q \mid Q) \]

\[ \sim \]

\[ (!P \mid !Q) \mid (P' \mid Q) \]

A mix of

- behavioural, LTS-based, coinductive
- axiomatic, syntax-based, inductive
Abstract coinduction

Let $b$ a monotone function on a complete lattice

- Coinduction

\[ x \leq y \leq b(y) \]

\[ x \leq \nu b \]

- bisimulation candidate

- program equivalence

- law to be proved

Damien Pous, CNRS, ENS Lyon
Abstract coinduction

Let $b$ a monotone function on a complete lattice

- Coinduction
  
  \[
  x \leq y \leq b(y) \quad \frac{\text{law to be proved}}{\quad x \leq \nu b}
  \]

- Coinduction up to
  
  \[
  x \leq y \leq b(f(y)) \quad \frac{\text{an up-to technique}}{\quad x \leq \nu b}
  \]
Up-to techniques

The function $f$ can be

- sound: it just makes the rule valid
- respectful: $f \circ (b \cap id) \leq b \circ f$  
  [Sangiorgi’94]
- compatible: $f \circ b \leq b \circ f$  
  [me’07]

The latter two classes are closed under union and composition
The companion

Let $t$ be the largest compatible function

- We have
  1. $\text{id} \leq t$ and $t \circ t \leq t$, i.e., $t$ is a closure
  2. $\nu b = t(\perp)$
  3. $\nu b = t(\nu b)$
  4. $\nu b = \nu(b \circ t)$
  5. $t$ coincides with the largest respectful

- Intuitively $t(x)$ is “what can be deduced assuming $x$”

- Leads to a new presentation of parameterized coinduction
Parameterized Coinduction (Up-to)

\[
\begin{align*}
&\frac{y \leq t(\bot)}{\text{Init}} \\
&\frac{y \leq \nu b}{\text{Init}} \\
\frac{y \leq x}{y \leq t(x)} & \quad \text{Done} \\
\frac{y \leq f(t(x))}{\text{Up to } f} \quad f \leq t \\
\frac{y \leq t(x)}{\text{Up to } f} \\
\frac{y \leq b(t(y \cup x))}{\text{ColInd}} \\
\frac{y \leq t(x)}{\text{ColInd}}
\end{align*}
\]

law to be proved

program equivalence

reason inductively, using some valid up-to technique

use coinduction hypotheses

assume \( y \) by coinduction

Damien Pous, CNRS, ENS Lyon
The companion is itself a coinductive object

\[ t = \nu B \]

(for some \( B : (X \to X) \to (X \to X) \) whose post-fixpoints are the compatible functions)
Second order techniques

The companion is itself a coinductive object

\[ t = \nu B \]

(for some \( B : (X \to X) \to (X \to X) \) whose post-fixpoints are the compatible functions)

We can apply the previous results

\[ f \leq g \leq B(T(g)) \]

f \leq t
Back to process algebra

[Standard approach] The following function is respectful:

\[ C : R \mapsto \{ \langle C[\tilde{P}], C[\tilde{Q}] \rangle \mid \tilde{P} R \tilde{Q} \} \]
Back to process algebra

[Standard approach] The following function is respectful:

\[ C : \mathcal{R} \mapsto \{ \langle C[\tilde{P}], C[\tilde{Q}] \rangle \mid \tilde{P} \mathcal{R} \tilde{Q} \} \]

[New approach] The following functions are below \( t \):

\[ c_{(\nu a)} : \mathcal{R} \mapsto \{ \langle (\nu a)P, (\nu a)Q \rangle \mid P \mathcal{R} Q \} \]
\[ c| : \mathcal{R} \mapsto \{ \langle P_1 \mid P_2, Q_1 \mid Q_2 \rangle \mid P_i \mathcal{R} Q_i \} \]
\[ c! : \mathcal{R} \mapsto \{ \langle !P, !Q \rangle \mid P \mathcal{R} Q \} \]

\[ \ldots \]

(A routine check in each case, thanks to the second order companion)
Back to process algebra

[Standard approach] The following function is respectful:

\[ C : \mathcal{R} \mapsto \{ \langle \tilde{P}, \tilde{Q} \rangle \mid \tilde{P} \mathcal{R} \tilde{Q} \} \]

[New approach] The following functions are below \(t\):

\[ c_{(\nu a)} : \mathcal{R} \mapsto \{ \langle (\nu a)P, (\nu a)Q \rangle \mid P \mathcal{R} Q \} \]

\[ c_{|} : \mathcal{R} \mapsto \{ \langle P_1 \mid P_2, Q_1 \mid Q_2 \rangle \mid P_i \mathcal{R} Q_i \} \]

\[ c_{!} : \mathcal{R} \mapsto \{ \langle !P, !Q \rangle \mid P \mathcal{R} Q \} \]

\ldots

(A routine check in each case, thanks to the second order companion)
Huge simplification of the theory of up-to techniques...  
...just by focusing on the largest one

- parameterized coinduction through the companion
- second order techniques for the companion
- straightforward formalisation in Coq

In progress: categorical account