Logics for Contravariant Simulations

Ignacio Fábregas David de Frutos Escrig
Miguel Palomino

Departamento de Sistemas Informáticos y Computación, UCM

ICE-TCS Workshop on Logic and Concurrency

Outline

1. Introduction
   - Motivation
2. Definitions and results
   - New simulation notions
   - Examples revisited
   - Logical characterizations
3. Conclusions and future work

Plain simulations

- Compare processes based on the simple premise:

  “*you are better if you can do as much as me, and perhaps some additional new things*”

- Assume that all the executable actions are controlled by the user
  - No distinction between input and output actions.
  - More possibilities means less control (non-determinism).

Motivating examples

Machines

$$
\begin{array}{rcl}
\text{onecoke} & : & \text{coin?} \to \text{coke!} \to 0 \\
\text{cokeorlemonade} & : & \text{coin?} \to ((\text{coke!} \to 0) + (\text{lemonade!} \to 0))
\end{array}
$$

- onecoke $\lesssim_S$ cokeorlemonade.
- coke! and lemonade! should be considered as output actions.
    - I cannot choose between them!

Our proposal

We defined two new notions of simulations and their logical characterizations in terms of a Henessy-Milner logic.

Definitions

- Distinguishing between input and output actions.

Covariant-contravariant simulations

Given $P = (P, A, \rightarrow_P)$ and $Q = (Q, A, \rightarrow_Q)$, and $\{A^r, A^l, A^{bi}\}$ a partition of A, pSq implies

- for all $a \in A^r \cup A^{bi}$ and all $p \xrightarrow{a} p'$ there exists $q \xrightarrow{a} q'$ with $p'Sq'$.
- for all $a \in A^l \cup A^{bi}$, and all $q \xrightarrow{a} q'$ there exists $p \xrightarrow{a} p'$ with $p'Sq'$.

- If $A^r = A$ we have plain simulation.
- If $A^l = A$ we have plain “anti-simulation”.
- If $A^{bi} = A$ we have bisimulation.

Definitions

- Reduction of non-determinism: $ap \gtrsim ap + aq$.

Conformance simulations

Given $P = (P, A, \rightarrow_P)$ and $Q = (Q, A, \rightarrow_Q)$, pRq implies:

- For all $a \in A$, if $p \xrightarrow{a}$, then $q \xrightarrow{a}$.
- For all $a \in A$ such that $q \xrightarrow{a} q'$ and $p \xrightarrow{a}$, there exists some p' with $p \xrightarrow{a} p'$ and $p' R q'$.

- $I(p) \subseteq I(q)$ guarantees that Q has at least all the behaviors of P:

$$0 \lesssim_{CS} a \;,\; a \lesssim_{CS} a + b.$$

- The second clause is contravariant and establishes that a process can be "improved" by reducing the nondeterminism in it.

$$a + a \lesssim_{CS} a.$$

Definitions

- Reduction of non-determinism: $ap \gtrsim ap + aq$.

Conformance simulations

Given $P = (P, A, \rightarrow_P)$ and $Q = (Q, A, \rightarrow_Q)$, pRq implies:

- For all $a \in A$, if $p \xrightarrow{a}$, then $q \xrightarrow{a}$.
- For all $a \in A$ such that $q \xrightarrow{a} q'$ and $p \xrightarrow{a}$, there exists some p' with $p \xrightarrow{a} p'$ and $p'Rq'$.

- $I(p) \subseteq I(q)$ guarantees that Q has at least all the behaviors of P:

  $$0 \lesssim_{CS} a\ ,\ a \lesssim_{CS} a + b.$$

- The second clause is contravariant and establishes that a process can be "improved" by reducing the nondeterminism in it.

  $$a + a \lesssim_{CS} a.$$

Definitions

- Reduction of non-determinism: $ap \gtrsim ap + aq$.

Conformance simulations

Given $P = (P, A, \rightarrow_P)$ and $Q = (Q, A, \rightarrow_Q)$, pRq implies:

- For all $a \in A$, if $p \xrightarrow{a}$, then $q \xrightarrow{a}$.
- For all $a \in A$ such that $q \xrightarrow{a} q'$ and $p \xrightarrow{a}$, there exists some p' with $p \xrightarrow{a} p'$ and $p'Rq'$.

- $I(p) \subseteq I(q)$ guarantees that Q has at least all the behaviors of P:

  $0 \lesssim_{CS} a$, $a \lesssim_{CS} a + b$.

- The second clause is contravariant and establishes that a process can be "improved" by reducing the nondeterminism in it.

  $a + a \lesssim_{CS} a$.

Examples revisited

Machines

$$\begin{aligned} \text{onecoke} &: \text{coin?} \to \text{coke!} \to 0 \\ \text{cokeorlemonade} &: \text{coin?} \to \big((\text{coke!} \to 0) + (\text{lemonade!} \to 0)\big) \end{aligned}$$

Covariant-contravariant simulations

- cokeorlemonade $\lesssim_{CC}$ onecoke.
    - Considering coke! and lemonade! as outputs (that is, $A^l = \{$coke!,lemonade!$\}$).

Conformance simulations

- onecoke $\lesssim_{CS}$ cokeorlemonade.
    - But we do not consider coke! and lemonade! as outputs.

Examples revisited

Machines

$$
\begin{array}{lcl}
\text{onecoke} & : & \text{coin?} \rightarrow \text{coke!} \rightarrow 0 \\
\text{choice\_coke\_lemonade} & : & (\text{coin?} \rightarrow \text{coke!} \rightarrow 0)+ \\
 & & (\text{coin?} \rightarrow \text{lemonade!} \rightarrow 0)
\end{array}
$$

Conformance simulations

- choice_coke_lemonade $\lesssim_{CS}$ onecoke.
  - onecoke is more deterministic.
    ... But we still do not consider coke! or lemonade! outputs.
- $0 \lesssim_{CS}$ choice_coke_lemonade.

Covariant-contravariant simulations

Plain simulation logic

- $p \models \mathrm{tt}$.
- $p \models \bigwedge_{i \in I} \varphi_i$ if $p \models \varphi_i$ for all $i \in I$.
- $p \models \langle a \rangle \varphi$, if there exists $p \xrightarrow{a} p'$ and $p' \models \varphi$.

Plain "anti-simulation" logic

- $p \not\models \mathrm{ff}$.
- $p \models \bigvee_{i \in I} \varphi_i$ if $p \models \varphi_i$ for some $i \in I$.
- $p \models [b]\varphi$, if $p' \models \varphi$ for all p' such that $p \xrightarrow{b} p'$.

Covariant-contravariant simulations

Plain simulation logic

- $p \models \mathrm{tt}$.
- $p \models \bigwedge_{i \in I} \varphi_i$ if $p \models \varphi_i$ for all $i \in I$.
- $p \models \langle a \rangle \varphi$, if there exists $p \xrightarrow{a} p'$ and $p' \models \varphi$.

Plain "anti-simulation" logic

- $p \not\models \mathrm{ff}$.
- $p \models \bigvee_{i \in I} \varphi_i$ if $p \models \varphi_i$ for some $i \in I$.
- $p \models [b]\varphi$, if $p' \models \varphi$ for all p' such that $p \xrightarrow{b} p'$.

Covariant-contravariant simulations

Plain simulation logic

- $p \models \mathrm{tt}$.
- $p \models \bigwedge_{i \in I} \varphi_i$ if $p \models \varphi_i$ for all $i \in I$.
- $p \models \langle a \rangle \varphi$, if there exists $p \xrightarrow{a} p'$ and $p' \models \varphi$.

Plain "anti-simulation" logic

- $p \not\models \mathrm{ff}$.
- $p \models \bigvee_{i \in I} \varphi_i$ if $p \models \varphi_i$ for some $i \in I$.
- $p \models [b]\varphi$, if $p' \models \varphi$ for all p' such that $p \xrightarrow{b} p'$.

Covariant-contravariant simulations

Plain simulation logic

- $p \models \mathrm{tt}$.
- $p \models \bigwedge_{i \in I} \varphi_i$ if $p \models \varphi_i$ for all $i \in I$.
- $p \models \langle a \rangle \varphi$, if there exists $p \xrightarrow{a} p'$ and $p' \models \varphi$.

Plain "anti-simulation" logic

- $p \not\models \mathrm{ff}$.
- $p \models \bigvee_{i \in I} \varphi_i$ if $p \models \varphi_i$ for some $i \in I$.
- $p \models [b]\varphi$, if $p' \models \varphi$ for all p' such that $p \xrightarrow{b} p'$.

Covariant-contravariant simulations

Plain simulation logic

- $p \models \mathrm{tt}$.
- $p \models \bigwedge_{i \in I} \varphi_i$ if $p \models \varphi_i$ for all $i \in I$.
- $p \models \langle a \rangle \varphi$, if there exists $p \xrightarrow{a} p'$ and $p' \models \varphi$.

Plain "anti-simulation" logic

- $p \not\models \mathrm{ff}$.
- $p \models \bigvee_{i \in I} \varphi_i$ if $p \models \varphi_i$ for some $i \in I$.
- $p \models [b]\varphi$, if $p' \models \varphi$ for all p' such that $p \xrightarrow{b} p'$.

Covariant-contravariant simulations

Plain simulation logic

$p \lesssim_S q$ if and only if $p \models \varphi$ implies $q \models \varphi$

Plain "anti-simulation" logic

$p \lesssim_{\bar{S}} q$ if and only if $p \models \varphi$ implies $q \models \varphi$

Covariant-contravariant simulations

- Making the union of plain simulation and plain anti-simulation logics we obtain covariant-contravariant logic.

Covariant-contravariant logic: Syntax

The class $\mathcal{L}_{CC}$ is defined recursively by:

- tt and ff are in $\mathcal{L}_{CC}$.
- If I is a set and $\varphi_i \in \mathcal{L}_{CC}$ for all $i \in I$ then $\bigwedge_{i \in I} \varphi_i \in \mathcal{L}_{CC}$, $\bigvee_{i \in I} \varphi_i \in \mathcal{L}_{CC}$.
- If $\varphi \in \mathcal{L}_{CC}$ and $a \in A^r \cup A^{bi}$ then $\langle a \rangle \varphi \in \mathcal{L}_{CC}$.
- If $\varphi \in \mathcal{L}_{CC}$ and $a \in A^l \cup A^{bi}$ then $[a]\varphi \in \mathcal{L}_{CC}$.

Covariant-contravariant simulations

- Making the union of plain simulation and plain anti-simulation logics we obtain covariant-contravariant logic.

Covariant-contravariant logic: Semantic

- $p \models \mathrm{tt}$ and $p \not\models \mathrm{ff}$.
- $p \models \bigwedge_{i \in I} \varphi_i$ if $p \models \varphi_i$ for all $i \in I$.
- $p \models \bigvee_{i \in I} \varphi_i$ if $p \models \varphi_i$ for some $i \in I$.
- $p \models \langle a \rangle \varphi$ if there exists $p \xrightarrow{a} p'$ and $p' \models \varphi$.
- $p \models [a]\varphi$ if $p' \models \varphi$ for all p' such that $p \xrightarrow{a} p'$.

Proposition

$p \lesssim_{CC} q$ if and only if $p \models \varphi \implies q \models \varphi$ for all φ.

Conformance simulations

Conformance logic: Syntax

The class $\mathcal{L}_{CS}$ is defined recursively by:

- $\mathrm{tt} \in \mathcal{L}_{CS}$.
- If I is a set and $\varphi_i \in \mathcal{L}_{CS}$ for all $i \in I$ then $\bigwedge_{i\in I} \varphi_i, \in \mathcal{L}_{CS}$, $\bigvee_{i\in I} \varphi_i \in \mathcal{L}_{CS}$.
- If $\varphi \in \mathcal{L}_{CS}$ and $a \in A$ then $a\varphi \in \mathcal{L}_{CS}$.

Conformance simulations

Conformance logic: Semantic

- $p \models \mathrm{tt}$.
- $p \models \bigwedge_{i \in I} \varphi_i$ if $p \models \varphi_i$ for all $i \in I$.
- $p \models \bigvee_{i \in I} \varphi_i$ if $p \models \varphi_i$ for some $i \in I$.
- $p \models a\varphi$ if $p \xrightarrow{a}$ and $p' \models \varphi$ for all $p \xrightarrow{a} p'$.

- Captures $ap \gtrsim_{CS} ap + aq$.
- It is equivalent to $[a] \wedge \langle a \rangle$.

Proposition

$p \lesssim_{CS} q$ if and only if $p \models \varphi \implies q \models \varphi$ for all φ.

Conformance simulations

Conformance logic: Semantic

- $p \models \mathrm{tt}$.
- $p \models \bigwedge_{i \in I} \varphi_i$ if $p \models \varphi_i$ for all $i \in I$.
- $p \models \bigvee_{i \in I} \varphi_i$ if $p \models \varphi_i$ for some $i \in I$.
- $p \models a\varphi$ if $p \xrightarrow{a}$ and $p' \models \varphi$ for all $p \xrightarrow{a} p'$.

- Captures $ap \gtrsim_{CS} ap + aq$.
- It is equivalent to $[a] \wedge \langle a \rangle$.

Proposition

$p \lesssim_{CS} q$ if and only if $p \models \varphi \implies q \models \varphi$ for all φ.

Examples: covariant-contravariant simulation

Machines

$$\begin{aligned} \text{onecoke} &: \text{coin?} \rightarrow \text{coke!} \rightarrow 0 \\ \text{cokeorlemonade} &: \text{coin?} \rightarrow ((\text{coke!} \rightarrow 0) + (\text{lemonade!} \rightarrow 0)) \end{aligned}$$

- cokeorlemonade $\lesssim_{CC}$ onecoke.

  with $A^r = \{coin?\}$ and $A^l = \{coke!, lemonade!\}$.

Logical formulas

- onecoke $\models \langle \text{coin?} \rangle$ [lemonade!] ff.
  - Given a coin, we are sure that we are not going to get a lemonade.
- cokeorlemonade $\not\models \langle \text{coin?} \rangle$ [lemonade!] ff.

Examples: conformance simulation

Machines

onecoke	:	coin? $\rightarrow$ coke! $\rightarrow$ 0
choice_coke_lemonade	:	(coin? $\rightarrow$ coke! $\rightarrow$ 0)+
		(coin? $\rightarrow$ lemonade! $\rightarrow$ 0)

- choice_coke_lemonade $\lesssim_{CS}$ onecoke.

Logical formulas

- onecoke $\models$ coin? coke! tt.
  - We always get a coke for any coin (Non-trivially!).
  - 0 $\not\models$ coin? coke! tt.
- choice_coke_lemonade $\not\models$ coin? coke! tt.

- Plain simulation is not realistic if there are outputs.
- We define two new simulations.
  - We study their logical characterizations.

Machines

$$\begin{array}{lll} \text{slot\_machine} & : & (\text{coin?} \to ((\text{million\$!} \to 0) + (\text{souvenir!} \to 0)) + \\ & & (\text{coin?} \to \text{souvenir!} \to 0)) \\ \text{fake\_machine} & : & \text{coin?} \to \text{souvenir!} \to 0 \end{array}$$

- *fake_machine* $\lesssim_{CS}$ *slot_machine*,
- *slot_machine* $\lesssim_{CS}$ *fake_machine*.

  ... But we cannot win the big pot.

- We will continue with the study of these two notions.