Reward Testing Equivalences for Processes

Rob van Glabbeek

Data61, CSIRO, Sydney, Australia

University of New South Wales, Sydney, Australia

24 September 2019

Presented at IFIP WG2.2 meeting in Vienna

The corresponding paper is dedicated to Rocco De Nicola, on the occasion of his 65^{th} birthday.

Rocco's work has been a source of inspiration to my own.

The general theory of testing

It assumes

- a set of processes $\mathbb{P}$,
- a set of tests $\mathbb{T}$, which can be applied to processes,
- a set of outcomes $\mathbb{O}$ of applying a test to a process, and
- a function $\mathcal{A}pply : \mathbb{T} \times \mathbb{P} \rightarrow \mathscr{P}^{+}(\mathbb{O})$, representing the possible results of applying a specific test to a specific process,
- a partial order $\leq$ on $\mathbb{O}$: some outcomes are better than others.

The general theory of testing

It assumes

- a set of processes $\mathbb{P}$,
- a set of tests $\mathbb{T}$, which can be applied to processes,
- a set of outcomes $\mathbb{O}$ of applying a test to a process, and
- a function $\mathcal{A}pply : \mathbb{T} \times \mathbb{P} \to \mathscr{P}^+(\mathbb{O})$, representing the possible results of applying a specific test to a specific process,
- a partial order $\leq$ on $\mathbb{O}$: some outcomes are better than others.

$O_1 \sqsubseteq_{\mathrm{Ho}} O_2$ if $\forall o_1 \in O_1\ \exists o_2 \in O_2$ such that $o_1 \leq o_2$
$O_1 \sqsubseteq_{\mathrm{Sm}} O_2$ if $\forall o_2 \in O_2\ \exists o_1 \in O_1$ such that $o_1 \leq o_2$.

The general theory of testing

It assumes

- a set of processes $\mathbb{P}$,
- a set of tests $\mathbb{T}$, which can be applied to processes,
- a set of outcomes $\mathbb{O}$ of applying a test to a process, and
- a function $\mathcal{A}pply : \mathbb{T} \times \mathbb{P} \to \mathscr{P}^+(\mathbb{O})$, representing the possible results of applying a specific test to a specific process,
- a partial order $\leq$ on $\mathbb{O}$: some outcomes are better than others.

  $O_1 \sqsubseteq_{\mathrm{Ho}} O_2$ if $\forall o_1 \in O_1\ \exists o_2 \in O_2$ such that $o_1 \leq o_2$
  $O_1 \sqsubseteq_{\mathrm{Sm}} O_2$ if $\forall o_2 \in O_2\ \exists o_1 \in O_1$ such that $o_1 \leq o_2$.

$P \sqsubseteq_{\mathrm{may}} Q$ iff $\mathcal{A}pply(T, P) \sqsubseteq_{\mathrm{Ho}} \mathcal{A}pply(T, Q)$ for every test T.
$P \sqsubseteq_{\mathrm{must}} Q$ iff $\mathcal{A}pply(T, P) \sqsubseteq_{\mathrm{Sm}} \mathcal{A}pply(T, Q)$ for every test T.

CCS

$$\alpha.E \xrightarrow{\alpha} E \qquad \frac{E_j \xrightarrow{\alpha} E'_j}{\sum_{i\in I} E_i \xrightarrow{\alpha} E'_j}\ (j \in I)$$

$$\frac{E \xrightarrow{\alpha} E'}{E|F \xrightarrow{\alpha} E'|F} \qquad \frac{E \xrightarrow{a} E',\ F \xrightarrow{\bar{a}} F'}{E|F \xrightarrow{\tau} E'|F'} \qquad \frac{F \xrightarrow{\alpha} F'}{E|F \xrightarrow{\alpha} E|F'}$$

$$\frac{E \xrightarrow{\alpha} E'}{E\backslash L \xrightarrow{\alpha} E'\backslash L}\ (\alpha, \bar{\alpha} \notin L) \qquad \frac{E \xrightarrow{\alpha} E'}{E[f] \xrightarrow{f(\alpha)} E'[f]} \qquad \frac{\mathbf{fix}(\!|S_X{:}S|\!) \xrightarrow{\alpha} E}{\mathbf{fix}(\!|X{:}S|\!) \xrightarrow{\alpha} E}$$

α ranges over $Act = \mathscr{A} \uplus \bar{\mathscr{A}} \uplus \{\tau\}$.

May and Must Testing for CCS

$Act^{\omega} := Act \cup \{\omega\}$. $\omega \notin Act$ is a special action reporting success.

A *test* $T \in \mathbb{T}_{\mathrm{CCS}}$ is a CCS process, but with α ranging over Act^{ω}.

May and Must Testing for CCS

$Act^\omega := Act \cup \{\omega\}$. $\omega \notin Act$ is a special action reporting success.

A *test* $T \in \mathbb{T}_{\mathrm{CCS}}$ is a CCS process, but with α ranging over Act^ω.

So a CCS process is a special kind of CCS test.

May and Must Testing for CCS

$Act^\omega := Act \cup \{\omega\}$. $\omega \notin Act$ is a special action reporting success.

A *test* $T \in \mathbb{T}_{\mathrm{CCS}}$ is a CCS process, but with α ranging over Act^ω.

So a CCS process is a special kind of CCS test.

To apply test T to process P one runs them in parallel: $T|P$.

May and Must Testing for CCS

$Act^\omega := Act \cup \{\omega\}$. $\omega \notin Act$ is a special action reporting success.

A *test* $T \in \mathbb{T}_{\mathrm{CCS}}$ is a CCS process, but with α ranging over Act^ω.

So a CCS process is a special kind of CCS test.

To apply test T to process P one runs them in parallel: $T|P$.

A *computation* $\pi \in \mathbb{T}^*_{\mathrm{CCS}}$ is a sequence $T_0, T_1, T_2, \ldots$ of tests, s.t.
(i) if T_n is the final element, then $T_n \stackrel{\tau}{\not\longrightarrow}$, and
(ii) otherwise $T_n \stackrel{\tau}{\longrightarrow} T_{n+1}$.

It is *successful* if it contains a state T with $T \stackrel{\omega}{\longrightarrow}$.

May and Must Testing for CCS

$Act^\omega := Act \cup \{\omega\}$. $\omega \notin Act$ is a special action reporting success.

A *test* $T \in \mathbb{T}_{\mathrm{CCS}}$ is a CCS process, but with α ranging over Act^ω.

So a CCS process is a special kind of CCS test.

To apply test T to process P one runs them in parallel: $T|P$.

A *computation* $\pi \in \mathbb{T}^*_{\mathrm{CCS}}$ is a sequence $T_0, T_1, T_2, \ldots$ of tests, s.t.
(i) if T_n is the final element, then $T_n \stackrel{\tau}{\not\longrightarrow}$, and
(ii) otherwise $T_n \stackrel{\tau}{\longrightarrow} T_{n+1}$.

It is *successful* if it contains a state T with $T \stackrel{\omega}{\longrightarrow}$.

Comp(T, P) is the set of computations whose starting from $T|P$.

$$\mathcal{A}pply(T,P) := \{\top \mid \exists \text{ successful } \pi \in Comp(T,P)\} \cup \{\bot \mid \exists \text{ unsuccessful } \pi \in Comp(T,P)\}.$$

May and Must Testing for CCS

$Act^\omega := Act \cup \{\omega\}$. $\omega \notin Act$ is a special action reporting success.

A *test* $T \in \mathbb{T}_{\mathrm{CCS}}$ is a CCS process, but with α ranging over Act^ω.

So a CCS process is a special kind of CCS test.

To apply test T to process P one runs them in parallel: $T|P$.

A *computation* $\pi \in \mathbb{T}^*_{\mathrm{CCS}}$ is a sequence $T_0, T_1, T_2, \ldots$ of tests, s.t.
(i) if T_n is the final element, then $T_n \stackrel{\tau}{\not\longrightarrow}$, and
(ii) otherwise $T_n \stackrel{\tau}{\longrightarrow} T_{n+1}$.

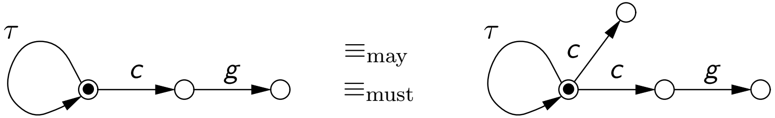
It is *successful* if it contains a state T with $T \stackrel{\omega}{\longrightarrow}$.

$Comp(T, P)$ is the set of computations whose starting from $T|P$.

$$\mathcal{A}pply(T, P) := \{\top \mid \exists \text{ successful } \pi \in Comp(T, P)\} \cup \{\bot \mid \exists \text{ unsuccessful } \pi \in Comp(T, P)\}.$$

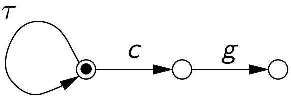
Now $P \sqsubseteq_{\mathrm{may}} Q$ holds unless $\exists T$ such that $T|P$ has a successful computation but Q has not.
Likewise $P \sqsubseteq_{\mathrm{must}} Q$ holds unless there is a test T such that $T|P$ has only successful computations but Q has not.

Testing does not capture conditional liveness



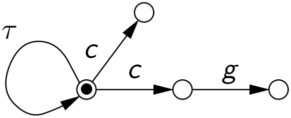
Processes identified by may and must testing, but with different conditional liveness properties

Testing does not capture conditional liveness



$\equiv_{\text{may}}$

$\equiv_{\text{must}}$



Processes identified by may and must testing, but with different conditional liveness properties

A conditional liveness property says that

> *under certain conditions something good will eventually happen.*

Reward Testing

A *reward test* is a CCS process, but with α ranging over $Act \times \mathbb{R}$. Such a *valued action* is tagged with a real number, the *reward* for executing this action. A negative reward is a penalty.

Reward Testing

A *reward test* is a CCS process, but with α ranging over $Act \times \mathbb{R}$. Such a *valued action* is tagged with a real number, the *reward* for executing this action. A negative reward is a penalty.

$$\frac{P \xrightarrow{a,r} P',\ \ Q \xrightarrow{\bar{a},r'} Q'}{P|Q \xrightarrow{\tau,r+r'} P'|Q'} \qquad \frac{P \xrightarrow{\alpha,r} P'}{P\backslash L \xrightarrow{\alpha,r} P'\backslash L}\ (\alpha, \bar{\alpha} \notin L) \qquad \frac{P \xrightarrow{\alpha,r} P'}{P[f] \xrightarrow{f(\alpha),r} P'[f]}$$

In all other rules of CCS, α is simply replaced by α, r, with $r \in \mathbb{R}$.

Reward Testing

A *reward test* is a CCS process, but with α ranging over $Act \times \mathbb{R}$. Such a *valued action* is tagged with a real number, the *reward* for executing this action. A negative reward is a penalty.

$$\frac{P \xrightarrow{a,r} P', \;\; Q \xrightarrow{\bar{a},r'} Q'}{P|Q \xrightarrow{\tau,r+r'} P'|Q'} \qquad \frac{P \xrightarrow{\alpha,r} P'}{P\backslash L \xrightarrow{\alpha,r} P'\backslash L} \;\; (\alpha, \bar{\alpha} \notin L) \qquad \frac{P \xrightarrow{\alpha,r} P'}{P[f] \xrightarrow{f(\alpha),r} P'[f]}$$

In all other rules of CCS, α is simply replaced by α, r, with $r \in \mathbb{R}$.

A valued action $\alpha, 0$ is denoted α, so a CCS process is a special CCS reward test: one in which all rewards are 0.

Reward Testing

A *reward test* is a CCS process, but with α ranging over $Act \times \mathbb{R}$. Such a *valued action* is tagged with a real number, the *reward* for executing this action. A negative reward is a penalty.

$$\frac{P \xrightarrow{a,r} P',\ Q \xrightarrow{\bar{a},r'} Q'}{P|Q \xrightarrow{\tau,r+r'} P'|Q'} \quad \frac{P \xrightarrow{\alpha,r} P'}{P\backslash L \xrightarrow{\alpha,r} P'\backslash L}\ (\alpha, \bar{\alpha} \notin L) \quad \frac{P \xrightarrow{\alpha,r} P'}{P[f] \xrightarrow{f(\alpha),r} P'[f]}$$

In all other rules of CCS, α is simply replaced by α, r, with $r \in \mathbb{R}$.

A valued action $\alpha, 0$ is denoted α, so a CCS process is a special CCS reward test: one in which all rewards are 0.

A *reward computation* π is a sequence $T_0, r_1, T_1, r_2, T_2, \ldots$, s.t.:
(i) if T_n is the final element, then $T_n \overset{\tau,r}{\not\longrightarrow}$, and
(ii) otherwise $T_n \xrightarrow{\tau,r_{n+1}} T_{n+1}$.

Reward Testing

A *reward test* is a CCS process, but with α ranging over $Act \times \mathbb{R}$. Such a *valued action* is tagged with a real number, the *reward* for executing this action. A negative reward is a penalty.

$$\frac{P \xrightarrow{a,r} P',\ \ Q \xrightarrow{\bar{a},r'} Q'}{P|Q \xrightarrow{\tau,r+r'} P'|Q'} \qquad \frac{P \xrightarrow{\alpha,r} P'}{P\backslash L \xrightarrow{\alpha,r} P'\backslash L}\ (\alpha,\bar{\alpha} \notin L) \qquad \frac{P \xrightarrow{\alpha,r} P'}{P[f] \xrightarrow{f(\alpha),r} P'[f]}$$

In all other rules of CCS, α is simply replaced by α, r, with $r \in \mathbb{R}$.

A valued action $\alpha, 0$ is denoted α, so a CCS process is a special CCS reward test: one in which all rewards are 0.

A *reward computation* π is a sequence $T_0, r_1, T_1, r_2, T_2, \ldots$, s.t.:
(i) if T_n is the final element, then $T_n \stackrel{\tau,r}{\not\longrightarrow}$, and
(ii) otherwise $T_n \xrightarrow{\tau,r_{n+1}} T_{n+1}$.

The *reward* of π: $\sum_{i=1}^n r_i$ or $\displaystyle\inf_{n\to\infty} \sum_{i=1}^{n} r_i \in \mathbb{R} \cup \{-\infty, \infty\}$.

Let $\mathcal{A}pply(T,P) := \{reward(\pi) \mid \pi \in Comp^R(T,P)\}$.

Must versus may

$P \sqsubseteq^{\text{must}}_{\text{reward}} Q$ holds iff under any reward test, the worst possible reward for Q is better than the worst possible reward for P.

$P \sqsubseteq^{\text{may}}_{\text{reward}} Q$ holds iff under any reward test, the best possible reward for Q is better than the best possible reward for P.

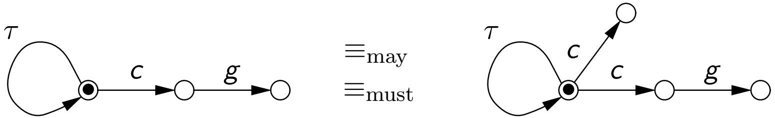
Must versus may

$P \sqsubseteq^{\text{must}}_{\text{reward}} Q$ holds iff under any reward test, the worst possible reward for Q is better than the worst possible reward for P.

$P \sqsubseteq^{\text{may}}_{\text{reward}} Q$ holds iff under any reward test, the best possible reward for Q is better than the best possible reward for P.

Theorem: $P \sqsubseteq^{\text{may}}_{\text{reward}} Q$ iff $Q \sqsubseteq^{\text{must}}_{\text{reward}} P$.

Reward testing captures conditional liveness



Processes identified by may and must testing, but with different conditional liveness properties

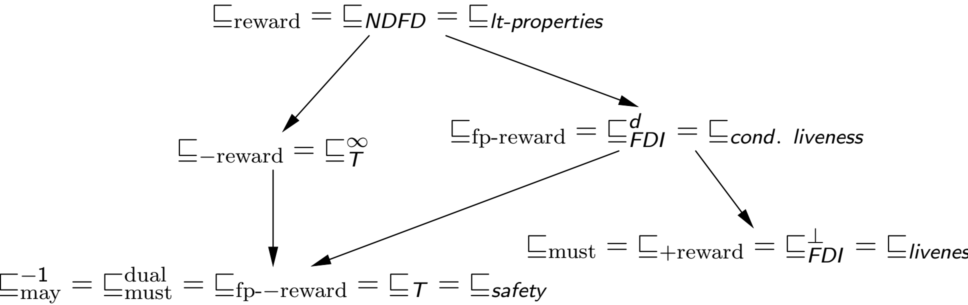
They are distinguished by reward testing

A conditional liveness property says that

> *under certain conditions something good will eventually happen.*

Theorem: Reward testing respects conditional liveness properties.

A spectrum of testing preorders



Congruence properties

$\sqsubseteq_{\mathrm{reward}}$ is a congruence for all operators of CCS except $+$.

Congruence properties

$\sqsubseteq_{\mathrm{reward}}$ is a congruence for all operators of CCS except $+$.

Let *stable* be the predicate that holds for a process P iff $P \stackrel{\tau}{\not\longrightarrow}$.

Write $P \sqsubseteq_X^\tau Q$ iff $P \sqsubseteq_X Q \wedge (\mathit{stable}(P) \Rightarrow \mathit{stable}(Q))$.

$\sqsubseteq_{\mathrm{reward}}^\tau$ is a congruence for all operators of CCS.

Axioms

$$
\left\{
\begin{array}{rcl}
\tau.X + Y & \equiv & \tau.X + \tau.(X + Y) \\
\alpha.X + \tau.(\alpha.Y + Z) & \equiv & \tau(\alpha.X + \alpha.Y + Z) \\
\alpha.(\tau.X + \tau.Y) & \equiv & \alpha.X + \alpha.Y \\
\tau.X + Y & \sqsubseteq & \tau.(X + Y) \\
\tau.X + Y & \sqsubseteq & X \\
\tau.\Delta X + Y & \equiv & \Delta(X + Y)
\end{array}
\right\}
$$

$\Delta P := \mathbf{fix}(\!|X\colon X \stackrel{def}{=} \tau.X + P\ |\!)$

Must testing: $\Delta X = \Delta Y$.

Congruence for recursion?

An equivalence $\sim$ is a (full) congruence for recursion if

$$\frac{S_Y \sim T_Y \quad \text{for all } Y \in dom(S)}{\mathbf{fix}(\!|X{:}S|\!) \sim \mathbf{fix}(\!|X{:}T|\!)} \qquad \frac{E \sim F}{\mu X.E \sim \mu X.F}$$

Congruence for recursion?

An equivalence $\sim$ is a (full) congruence for recursion if

$$\frac{S_Y \sim T_Y \quad \text{for all } Y \in dom(S)}{\mathbf{fix}(\!|X{:}S|\!) \sim \mathbf{fix}(\!|X{:}T|\!)} \qquad \frac{E \sim F}{\mu X.E \sim \mu X.F}$$

$\equiv^{\tau}_{\mathrm{reward}}$ fails to be a congruence for recursion in CCS.

Congruence for recursion?

An equivalence $\sim$ is a (full) congruence for recursion if

$$\frac{S_Y \sim T_Y \quad \text{for all } Y \in dom(S)}{\mathbf{fix}(\!|X{:}S|\!) \sim \mathbf{fix}(\!|X{:}T|\!)} \qquad \frac{E \sim F}{\mu X.E \sim \mu X.F}$$

$\equiv^{\tau}_{\mathrm{must}}$ and $\equiv^{\tau}_{\mathrm{reward}}$ fail to be congruences for recursion in CCS.

Congruence for recursion?

An equivalence $\sim$ is a (full) congruence for recursion if

$$\frac{S_Y \sim T_Y \quad \text{for all } Y \in dom(S)}{\mathbf{fix}(\!|X{:}S|\!) \sim \mathbf{fix}(\!|X{:}T|\!)} \qquad \frac{E \sim F}{\mu X.E \sim \mu X.F}$$

$\equiv^{\tau}_{\mathrm{must}}$ and $\equiv^{\tau}_{\mathrm{reward}}$ fail to be congruences for recursion in CCS.

$$\tau.X + Y \equiv^{\tau}_{must} \tau.X + \tau.(X + Y)$$

Congruence for recursion?

An equivalence $\sim$ is a (full) congruence for recursion if

$$\frac{S_Y \sim T_Y \quad \text{for all } Y \in dom(S)}{\mathbf{fix}(\!|X{:}S|\!) \sim \mathbf{fix}(\!|X{:}T|\!)} \qquad \frac{E \sim F}{\mu X.E \sim \mu X.F}$$

$\equiv^{\tau}_{\mathrm{must}}$ and $\equiv^{\tau}_{\mathrm{reward}}$ fail to be congruences for recursion in CCS.

$$\tau.X + Y \equiv^{\tau}_{must} \tau.X + \tau.(X + Y)$$
$$\tau.\mathbf{0} + Y \equiv^{\tau}_{must} \tau.\mathbf{0} + \tau.Y$$

Congruence for recursion?

An equivalence $\sim$ is a (full) congruence for recursion if

$$\frac{S_Y \sim T_Y \quad \text{for all } Y \in dom(S)}{\mathbf{fix}(\!|X{:}S|\!) \sim \mathbf{fix}(\!|X{:}T|\!)} \qquad \frac{E \sim F}{\mu X.E \sim \mu X.F}$$

$\equiv^{\tau}_{\mathrm{must}}$ and $\equiv^{\tau}_{\mathrm{reward}}$ fail to be congruences for recursion in CCS.

$$\tau.X + Y \equiv^{\tau}_{must} \tau.X + \tau.(X + Y)$$
$$\tau.\mathbf{0} + Y \equiv^{\tau}_{must} \tau.\mathbf{0} + \tau.Y$$

$$\mu Y.(\tau.\mathbf{0} + Y) \equiv^{\tau}_{must} \mu Y.(\tau.\mathbf{0} + \tau.Y)$$

Congruence for recursion?

An equivalence $\sim$ is a (full) congruence for recursion if

$$\frac{S_Y \sim T_Y \quad \text{for all } Y \in dom(S)}{\mathbf{fix}(\!|X{:}S|\!) \sim \mathbf{fix}(\!|X{:}T|\!)} \qquad \frac{E \sim F}{\mu X.E \sim \mu X.F}$$

$\equiv^{\tau}_{\mathrm{must}}$ and $\equiv^{\tau}_{\mathrm{reward}}$ fail to be congruences for recursion in CCS.

$$\tau.X + Y \equiv^{\tau}_{must} \tau.X + \tau.(X + Y)$$
$$\tau.\mathbf{0} + Y \equiv^{\tau}_{must} \tau.\mathbf{0} + \tau.Y$$

$$\tau.\mathbf{0} \equiv \qquad \mu Y.(\tau.\mathbf{0} + Y) \equiv^{\tau}_{must} \mu Y.(\tau.\mathbf{0} + \tau.Y)$$

Congruence for recursion?

An equivalence $\sim$ is a (full) congruence for recursion if

$$\frac{S_Y \sim T_Y \quad \text{for all } Y \in dom(S)}{\mathbf{fix}(\!|X{:}S|\!) \sim \mathbf{fix}(\!|X{:}T|\!)} \qquad \frac{E \sim F}{\mu X.E \sim \mu X.F}$$

$\equiv^{\tau}_{\mathrm{must}}$ and $\equiv^{\tau}_{\mathrm{reward}}$ fail to be congruences for recursion in CCS.

$$\tau.X + Y \equiv^{\tau}_{must} \tau.X + \tau.(X + Y)$$
$$\tau.\mathbf{0} + Y \equiv^{\tau}_{must} \tau.\mathbf{0} + \tau.Y$$

$$\tau.\mathbf{0} \equiv \qquad \mu Y.(\tau.\mathbf{0} + Y) \equiv^{\tau}_{must} \mu Y.(\tau.\mathbf{0} + \tau.Y) \equiv \Delta(\tau.\mathbf{0})$$

Congruence for recursion?

An equivalence $\sim$ is a (full) congruence for recursion if

$$\frac{S_Y \sim T_Y \quad \text{for all } Y \in dom(S)}{\mathbf{fix}(\!|X{:}S|\!) \sim \mathbf{fix}(\!|X{:}T|\!)} \qquad \frac{E \sim F}{\mu X.E \sim \mu X.F}$$

$\equiv^{\tau}_{\mathrm{must}}$ and $\equiv^{\tau}_{\mathrm{reward}}$ fail to be congruences for recursion in CCS.

$$\tau.X + Y \equiv^{\tau}_{must} \tau.X + \tau.(X + Y)$$
$$\tau.\mathbf{0} + Y \equiv^{\tau}_{must} \tau.\mathbf{0} + \tau.Y$$

$$\tau.\mathbf{0} \equiv \qquad \mu Y.(\tau.\mathbf{0} + Y) \equiv^{\tau}_{must} \mu Y.(\tau.\mathbf{0} + \tau.Y) \equiv \Delta(\tau.\mathbf{0})$$

$\uparrow$

strong bisimilarity

Congruence for recursion?

An equivalence $\sim$ is a (full) congruence for recursion if

$$\frac{S_Y \sim T_Y \quad \text{for all } Y \in dom(S)}{\mathbf{fix}(\!|X{:}S|\!) \sim \mathbf{fix}(\!|X{:}T|\!)} \qquad \frac{E \sim F}{\mu X.E \sim \mu X.F}$$

$\equiv^{\tau}_{\mathrm{must}}$ and $\equiv^{\tau}_{\mathrm{reward}}$ fail to be congruences for recursion in CCS.

$$\tau.X + Y \equiv^{\tau}_{must} \tau.X + \tau.(X + Y)$$
$$\tau.\mathbf{0} + Y \equiv^{\tau}_{must} \tau.\mathbf{0} + \tau.Y$$

$$\tau.\mathbf{0} \not\equiv^{\tau}_{must} \mu Y.(\tau.\mathbf{0} + Y) \equiv^{\tau}_{must} \mu Y.(\tau.\mathbf{0} + \tau.Y) \equiv \Delta(\tau.\mathbf{0})$$

Conclusion

I presented a new theory of testing yielding a finer equivalence, that respects conditional liveness properties.