Liveness of Communicating Transactions

Edsko de Vries

(joint work with Vasileios Koutavas and Matthew Hennessy)

TRINITY COLLEGE DUBLIN
Coláiste na Tríonóide, Baile Átha Cliath

science foundation ireland
fondúireacht eolaíochta éireann

Dublin Concurrency Workshop 2011

Traditional Transactions

- Transactions provide an abstraction for error recovery in a concurrent setting.

Traditional Transactions

- Transactions provide an abstraction for error recovery in a concurrent setting.
- The transactional system guarantees:
  - **Atomicity**: Each transaction will either run in its entirety or not at all
  - **Consistency**: Faults caused by a transaction are automatically detected and rolled-back
  - **Isolation**: The effects of a transaction are concealed from the rest of the system until the transaction commits
  - **Durability**: After a transaction commits, its effects are permanent.

Traditional Transactions

- Transactions provide an abstraction for error recovery in a concurrent setting.
- The transactional system guarantees:
  - **Atomicity**: Each transaction will either run in its entirety or not at all
  - **Consistency**: Faults caused by a transaction are automatically detected and rolled-back
  - **Isolation**: The effects of a transaction are concealed from the rest of the system until the transaction commits
  - **Durability**: After a transaction commits, its effects are permanent.
- However, **isolation limits concurrency**
  - The semantics of traditional transactions is sequential schedules
  - Traditional transactions do not offer an abstraction for recovery from distributed errors (e.g. deadlocks)

Communicating Transactions

- We drop isolation to increase concurrency
  - There is no limit on the communication between a transaction and its environment
- The transactional system guarantees:
  - **Atomicity**: Each transaction will either run in its entirety or not at all
  - **Consistency**: Faults caused by a transaction are automatically detected and rolled-back, together with all effects of the transaction to its environment
  - **Durability**: After all transactions that have interacted commit, their effects are permanent (coordinated checkpointing)
- We are interested in safety and especially liveness properties
  - First theory of liveness in the presence of transactions
  - We have studied the transactional properties of communicating transactions in [CONCUR'2010]

Safety

Safety: “Nothing bad will happen” [Lamport’77]

- A safety property can be formulated as a safety test T^{ω} which signals on channel ω when it detects the bad behaviour
- P passes the safety test T^{ω} when $P \mid T^{\omega}$ cannot output on ω
    - This is the negation of passing a “may test” [DeNicola-Hennessy’84]

Liveness

Liveness: "Something good will eventually happen" [Lamport'77]

- A liveness property can be formulated as a liveness test T^ω which detects and reports good behaviour on ω.
- P passes the liveness test T^ω when all future states of $P \mid T^\omega$ can output on ω
  - This is a "should test" [Binksma-Rensink-Vogler'95, Rensink-Vogler'07]
  - It excludes pathological traces
- We will later see why "must testing" [DeNicola-Hennessy'84] is not appropriate for transactions

TransCCS [CONCUR 2010]

Syntax:	P, Q	$::=$	$\sum \mu_i . P_i$	guarded choice
		$\mid$	$P \mid Q$	parallel
		$\mid$	$\nu a.P$	hiding
		$\mid$	$\mu X.P$	recursion
		$\mid$	$[\![P \rhd_k Q]\!]$	transaction (k bound in P)
		$\mid$	$\mathrm{co}\ k$	commit

TransCCS [CONCUR 2010]

Syntax:	P, Q	$::=$	$\sum \mu_i . P_i$	guarded choice
		$\mid$	$P \mid Q$	parallel
		$\mid$	$\nu a . P$	hiding
		$\mid$	$\mu X . P$	recursion
		$\mid$	$[\![P \triangleright_k Q]\!]$	transaction (k bound in P)
		$\mid$	$\mathrm{co}\ k$	commit

TransCCS [CONCUR 2010]

Syntax:	P, Q	$::=$	$\sum \mu_i . P_i$	guarded choice
		$\mid$	$P \mid Q$	parallel
		$\mid$	$\nu a . P$	hiding
		$\mid$	$\mu X . P$	recursion
		$\mid$	$\llbracket P \triangleright_k Q \rrbracket$	transaction (k bound in P)
		$\mid$	$\text{co } k$	commit

Default

TransCCS [CONCUR 2010]

Syntax:	P, Q	$::=$	$\sum \mu_i . P_i$	guarded choice
		$\mid$	$P \mid Q$	parallel
		$\mid$	$\nu a . P$	hiding
		$\mid$	$\mu X . P$	recursion
		$\mid$	$[\![P \rhd_k Q]\!]$	transaction (k bound in P)
		$\mid$	co k	commit

Alternative

TransCCS [CONCUR 2010]

Syntax:

$$\begin{array}{llll}
P, Q & ::= & \sum \mu_i . P_i & \text{guarded choice} \\
 & \mid & P \mid Q & \text{parallel} \\
 & \mid & \nu a . P & \text{hiding} \\
 & \mid & \mu X . P & \text{recursion} \\
 & \mid & [\![P \rhd_k Q]\!] & \text{transaction } (k \text{ bound in } P) \\
 & \mid & \text{co } k & \text{commit}
\end{array}$$

Name

TransCCS [CONCUR 2010]

Syntax:

$$\begin{array}{rcll} P, Q & ::= & \sum \mu_i.P_i & \text{guarded choice} \\ & \mid & P \mid Q & \text{parallel} \\ & \mid & \nu a.P & \text{hiding} \\ & \mid & \mu X.P & \text{recursion} \\ & \mid & [\![P \triangleright_k Q]\!] & \text{transaction } (k \text{ bound in } P) \\ & \mid & \mathsf{co}\ k & \text{commit} \end{array}$$

Main reductions:

R-COMM

$$\frac{a_i = \overline{b}_j}{\sum_{i \in I} a_i.P_i \mid \sum_{j \in J} b_j.Q_j \to P_i \mid Q_j}$$

R-EMB

$$\frac{k \notin R}{[\![P \triangleright_k Q]\!] \mid R \to [\![P \mid R \triangleright_k Q \mid R]\!]}$$

R-CO

$$\frac{}{[\![P \mid \mathsf{co}\ k \triangleright_k Q]\!] \to P}$$

R-AB

$$\frac{}{[\![P \triangleright_k Q]\!] \to Q}$$

TransCCS [CONCUR 2010]

Syntax:

P, Q	$::=$	$\sum \mu_i.P_i$	guarded choice
	$\mid$	$P \mid Q$	parallel
	$\mid$	$\nu a.P$	hiding
	$\mid$	$\mu X.P$	recursion
	$\mid$	$[\![P \rhd_k Q]\!]$	transaction (k bound in P)
	$\mid$	$\mathrm{co}\ k$	commit

Main reductions:

R-Comm
$$\frac{a_i = \overline{b}_j}{\sum_{i\in I} a_i.P_i \mid \sum_{j\in J} b_j.Q_j \to P_i \mid Q_j}$$

R-Emb
$$\frac{k \notin R}{[\![P \rhd_k Q]\!] \mid R \to [\![P \mid R \rhd_k Q \mid R]\!]}$$

R-Co
$$\frac{}{[\![P \mid \mathrm{co}\ k \rhd_k Q]\!] \to P}$$

R-Ab
$$\frac{}{[\![P \rhd_k Q]\!] \to Q}$$

TransCCS [CONCUR 2010]

Syntax:

P, Q	$::=$	$\sum \mu_i.P_i$	guarded choice
	$\mid$	$P \mid Q$	parallel
	$\mid$	$\nu a.P$	hiding
	$\mid$	$\mu X.P$	recursion
	$\mid$	$[\![P \triangleright_k Q]\!]$	transaction (k bound in P)
	$\mid$	$\mathsf{co}\ k$	commit

Main reductions:

R-Comm

$$\frac{a_i = \overline{b}_j}{\sum_{i\in I} a_i.P_i \mid \sum_{j\in J} b_j.Q_j \to P_i \mid Q_j}$$

R-Emb

$$\frac{k \notin R}{[\![P \triangleright_k Q]\!] \mid R \to [\![P \mid R \triangleright_k Q \mid R]\!]}$$

R-Co

$$\frac{}{[\![P \mid \mathsf{co}\ k \triangleright_k Q]\!] \to P}$$

R-Ab

$$\frac{}{[\![P \triangleright_k Q]\!] \to Q}$$

Simple Example

$$a.c.\omega + e.\mho \mid \llbracket \overline{a}.\overline{c}.\mathrm{co}\ k + \overline{e} \triangleright_k r \rrbracket$$

Simple Example

$$a.c.\omega + e.\mho \mid \llbracket \overline{a}.\overline{c}.\mathsf{co}\ k + \overline{e} \triangleright_k r \rrbracket$$

Simple Example

$$a.c.\omega + e.\mho \mid \llbracket \overline{a}.\overline{c}.\mathsf{co}\ k + \overline{e} \rhd_k\ r \rrbracket$$

$$\xrightarrow{\text{R-EMB}} \llbracket a.c.\omega + e.\mho \mid \overline{a}.\overline{c}.\mathsf{co}\ k + \overline{e} \rhd_k\ a.c.\omega + e.\mho \mid r \rrbracket$$

Simple Example

$$a.c.\omega + e.\mho \mid [\![\overline{a}.\overline{c}.\mathsf{co}\ k + \overline{e} \rhd_k r]\!]$$

$$\xrightarrow{\text{R-Emb}} [\![a.c.\omega + e.\mho \mid \overline{a}.\overline{c}.\mathsf{co}\ k + \overline{e} \rhd_k a.c.\omega + e.\mho \mid r]\!]$$

$$\xrightarrow{\text{R-Comm}} [\![\ c.\omega \mid \overline{c}.\mathsf{co}\ k \rhd_k a.c.\omega + e.\mho \mid r]\!]$$

Simple Example

$$a.c.\omega + e.\mho \mid [\![\overline{a}.\overline{c}.\mathsf{co}\ k + \overline{e} \rhd_k\ r]\!]$$

$$\xrightarrow{\text{R-EMB}} [\![a.c.\omega + e.\mho \mid \overline{a}.\overline{c}.\mathsf{co}\ k + \overline{e} \rhd_k\ a.c.\omega + e.\mho \mid r]\!]$$

$$\xrightarrow{\text{R-COMM}} [\![\ c.\omega \mid \overline{c}.\mathsf{co}\ k \rhd_k\ a.c.\omega + e.\mho \mid r]\!]$$

$$\xrightarrow{\text{R-COMM}} [\![\ \omega \mid \mathsf{co}\ k \rhd_k\ a.c.\omega + e.\mho \mid r]\!]$$

Simple Example

$$
\begin{aligned}
& a.c.\omega + e.\mho \mid \llbracket \overline{a}.\overline{c}.\mathsf{co}\ k + \overline{e} \triangleright_k r \rrbracket \\
\xrightarrow{\text{R-EMB}}\ & \llbracket a.c.\omega + e.\mho \mid \overline{a}.\overline{c}.\mathsf{co}\ k + \overline{e} \triangleright_k a.c.\omega + e.\mho \mid r \rrbracket \\
\xrightarrow{\text{R-COMM}}\ & \llbracket\ c.\omega \mid \overline{c}.\mathsf{co}\ k \triangleright_k a.c.\omega + e.\mho \mid r \rrbracket \\
\xrightarrow{\text{R-COMM}}\ & \llbracket\ \omega \mid \mathsf{co}\ k \triangleright_k a.c.\omega + e.\mho \mid r \rrbracket \\
\xrightarrow{\text{R-CO}}\ & \omega
\end{aligned}
$$

Simple Example

$$
\begin{array}{rl}
 & a.c.\omega + e.\mho \mid \llbracket \overline{a}.\overline{c}.\mathsf{co}\ k + \overline{e} \triangleright_k\ r \rrbracket \\
\xrightarrow{\text{R-EMB}} & \llbracket a.c.\omega + e.\mho \mid \overline{a}.\overline{c}.\mathsf{co}\ k + \overline{e} \triangleright_k\ a.c.\omega + e.\mho \mid r \rrbracket \\
\xrightarrow{\text{R-COMM}} & \llbracket\ c.\omega \quad\quad \mid \quad \overline{c}.\mathsf{co}\ k \quad\quad \triangleright_k\ a.c.\omega + e.\mho \mid r \rrbracket \\
\xrightarrow{\text{R-COMM}} & \llbracket\ \ \omega \quad\quad\ \ \mid \quad\ \ \mathsf{co}\ k \quad\quad \triangleright_k\ a.c.\omega + e.\mho \mid r \rrbracket \\
\xrightarrow{\text{R-CO}} & \omega
\end{array}
$$

Simple Example (a second trace)

$$a.c.\omega + e.\mho \mid \llbracket \overline{a}.\overline{c}.\mathsf{co}\ k + \overline{e} \rhd_k r \rrbracket$$

Simple Example (a second trace)

$$a.c.\omega + e.\mho \mid \llbracket \overline{a}.\overline{c}.\mathsf{co}\ k + \overline{e} \rhd_k\ r \rrbracket$$

$$\xrightarrow{\text{R-EMB}} \llbracket a.c.\omega + e.\mho \mid \overline{a}.\overline{c}.\mathsf{co}\ k + \overline{e} \rhd_k\ a.c.\omega + e.\mho \mid r \rrbracket$$

Simple Example (a second trace)

$$a.c.\omega + e.\mho \mid [\![\overline{a}.\overline{c}.\mathsf{co}\ k + \overline{e} \rhd_k\ r]\!]$$

$$\xrightarrow{\text{R-EMB}} [\![a.c.\omega + e.\mho \mid \overline{a}.\overline{c}.\mathsf{co}\ k + \overline{e} \rhd_k\ a.c.\omega + e.\mho \mid r]\!]$$

$$\xrightarrow{\text{R-COMM}} [\![\quad \mho \quad \rhd_k\ a.c.\omega + e.\mho \mid r]\!]$$

Simple Example (a second trace)

$$a.c.\omega + e.\mho \mid \llbracket \overline{a}.\overline{c}.\mathrm{co}\ k + \overline{e} \rhd_k r \rrbracket$$

$$\xrightarrow{\text{R-EMB}} \llbracket a.c.\omega + e.\mho \mid \overline{a}.\overline{c}.\mathrm{co}\ k + \overline{e} \rhd_k a.c.\omega + e.\mho \mid r \rrbracket$$

$$\xrightarrow{\text{R-COMM}} \llbracket \quad \mho \quad \rhd_k a.c.\omega + e.\mho \mid r \rrbracket$$

Simple Example (a second trace)

$$a.c.\omega + e.\mho \mid [\![\overline{a}.\overline{c}.\mathsf{co}\ k + \overline{e} \triangleright_k\ r]\!]$$

$$\xrightarrow{\text{R-EMB}} [\![a.c.\omega + e.\mho \mid \overline{a}.\overline{c}.\mathsf{co}\ k + \overline{e} \triangleright_k\ a.c.\omega + e.\mho \mid r]\!]$$

$$\xrightarrow{\text{R-COMM}} [\![\quad \mho \quad \triangleright_k\ a.c.\omega + e.\mho \mid r]\!]$$

$$\xrightarrow{\text{R-AB}} a.c.\omega + e.\mho \mid r$$

Simple Example (a second trace)

$$a.c.\omega + e.\mho \mid \llbracket \overline{a}.\overline{c}.\mathrm{co}\ k + \overline{e} \triangleright_k r \rrbracket$$

$$\xrightarrow{\text{R-EMB}} \llbracket a.c.\omega + e.\mho \mid \overline{a}.\overline{c}.\mathrm{co}\ k + \overline{e} \triangleright_k a.c.\omega + e.\mho \mid r \rrbracket$$

$$\xrightarrow{\text{R-COMM}} \llbracket \quad \mho \quad \triangleright_k a.c.\omega + e.\mho \mid r \rrbracket$$

$$\xrightarrow{\text{R-AB}} a.c.\omega + e.\mho \mid r \quad \text{(The environment is restored)}$$

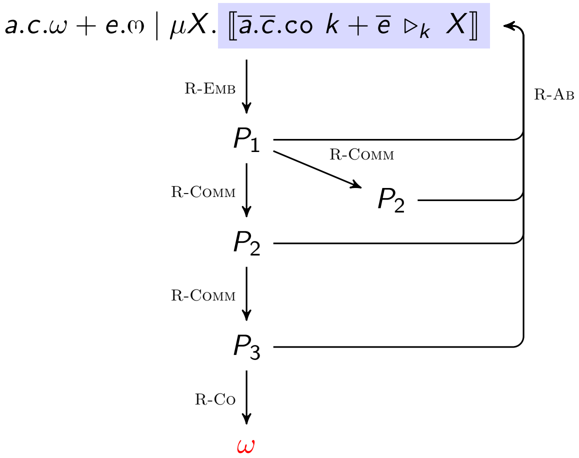
Simple Example (all traces)

$$\begin{tikzcd}
a.c.\omega + e.\mho \mid \llbracket \overline{a}.\overline{c}.\mathsf{co}\ k + \overline{e} \rhd_k\ r \rrbracket \arrow[rr, "\text{R-Ab}"] \arrow[d, "\text{R-Emb}"'] & & {\color{red} a.c.\omega + e.\mho \mid r} \\
P_1 \arrow[d, "\text{R-Comm}"'] \arrow[dr, "\text{R-Comm}"] \arrow[urr, "\text{R-Ab}"'] & & \\
& P_2 \arrow[uur] & \\
P_2 \arrow[d, "\text{R-Comm}"'] \arrow[uuurr] & & \\
P_3 \arrow[d, "\text{R-Co}"'] \arrow[uuuurr] & & \\
{\color{red} \omega} & &
\end{tikzcd}$$

Restarting transactions

$$a.c.\omega + e.\mho \mid \mu X.\,[\![\overline{a}.\overline{c}.\mathsf{co}\ k + \overline{e} \rhd_k X]\!]$$

Restarting transactions



$$\begin{tikzcd}
a.c.\omega + e.\mho \mid \mu X.\,[\![\overline{a}.\overline{c}.\mathrm{co}\ k + \overline{e} \rhd_k X]\!] \arrow[d, "\text{R-Emb}"'] & \\
P_1 \arrow[d, "\text{R-Comm}"'] \arrow[dr, "\text{R-Comm}"] \arrow[u, bend right=60, "\text{R-Ab}"'] & \\
& P_2 \arrow[ul, bend right=60] \\
P_2 \arrow[d, "\text{R-Comm}"'] \arrow[uuu, bend right=60] & \\
P_3 \arrow[d, "\text{R-Co}"'] \arrow[uuuu, bend right=60] & \\
\omega &
\end{tikzcd}$$

Restarting transactions

$$\begin{tikzcd}
a.c.\omega + e.\mho \mid \mu X. [\![\overline{a}.\overline{c}.\mathsf{co}\ k + \overline{e} \rhd_k X]\!] \arrow[d, "\text{R-Emb}"'] & \\
P_1 \arrow[d, "\text{R-Comm}"'] \arrow[r, "\text{R-Comm}"] \arrow[u, bend right, "\text{R-Ab}"'] & P_2 \arrow[ul, bend right, "\text{R-Ab}"'] \\
P_2 \arrow[d, "\text{R-Comm}"'] \arrow[uu, bend right, "\text{R-Ab}"'] & \\
P_3 \arrow[d, "\text{R-Co}"'] \arrow[uuu, bend right, "\text{R-Ab}"'] & \\
\omega &
\end{tikzcd}$$

Infinitely aborting loop

Compositional Semantics

- The embedding rule is simple but entangles the processes
- We need to reason about the behaviour of $P|Q$ in terms of P and Q
- We introduce a compositional Labelled Transition System that uses secondary transactions: $[\![P \triangleright_k Q]\!]^\circ$

Compositional Semantics

- The embedding rule is simple but entangles the processes
- We need to reason about the behaviour of $P|Q$ in terms of P and Q
- We introduce a compositional Labelled Transition System that uses secondary transactions: $\boxed{[\![P \rhd_k Q]\!]^\circ}$

$$
\begin{array}{lll|lll}
 & a.c.\omega + e.\mho & & & [\![\overline{a}.\overline{c}.\mathrm{co}\ k + \overline{e} \rhd_k r]\!] \\
\xrightarrow{\mathtt{emb}\ k} & [\![a.c.\omega + e.\mho \rhd_k a.c.\omega + e.\mho]\!]^\circ & & \xrightarrow{\mathtt{emb}\ k} & [\![\overline{a}.\overline{c}.\mathrm{co}\ k + \overline{e} \rhd_k r]\!] \\
\xrightarrow{k(a)} & [\![\ c.\omega \rhd_k a.c.\omega + e.\mho]\!]^\circ & & \xrightarrow{k(\overline{a})} & [\![\ \overline{c}.\mathrm{co}\ k \rhd_k r]\!] \\
\xrightarrow{k(c)} & [\![\ \omega \rhd_k a.c.\omega + e.\mho]\!]^\circ & & \xrightarrow{k(\overline{c})} & [\![\ \mathrm{co}\ k \rhd_k r]\!] \\
\xrightarrow{\mathrm{co}\ k} & \omega & & \xrightarrow{\mathrm{co}\ k} & \mathbf{0}
\end{array}
$$

Compositional Semantics

- The embedding rule is simple but entangles the processes
- We need to reason about the behaviour of $P|Q$ in terms of P and Q
- We introduce a compositional Labelled Transition System that uses secondary transactions: $\boxed{[\![P \rhd_k Q]\!]^\circ}$

$$
\begin{array}{lll|lll}
 & a.c.\omega + e.\mho & & & [\![\overline{a}.\overline{c}.\mathrm{co}\ k + \overline{e} \rhd_k r]\!] \\
\xrightarrow{\texttt{emb}\ k} & [\![a.c.\omega + e.\mho \rhd_k a.c.\omega + e.\mho]\!]^\circ & & \xrightarrow{\texttt{emb}\ k} & [\![\overline{a}.\overline{c}.\mathrm{co}\ k + \overline{e} \rhd_k r]\!] \\
\xrightarrow{k(e)} & [\![\ c.\omega \rhd_k a.c.\omega + e.\mho]\!]^\circ & & \xrightarrow{k(\overline{e})} & [\![\ \rhd_k r]\!] \\
\xrightarrow{\texttt{ab}\ k} & a.c.\omega + e.\mho & & \xrightarrow{\texttt{ab}\ k} & r
\end{array}
$$

Compositional Semantics (2)

The behaviour of processes in TransCCS can be understood by CCS-like "Clean" traces derived by the LTS that:

- consider only traces where all actions are eventually committed
- ignore transactional annotations on the traces

Compositional Semantics (2)

The behaviour of processes in TransCCS can be understood by CCS-like "Clean" traces derived by the LTS that:

- consider only traces where all actions are eventually committed
- ignore transactional annotations on the traces

$$\mathcal{L}(\ [\![a.c.\mathrm{co}\ k + e \rhd_k\ r]\!]\ ) = \{\epsilon,\ \mathbf{a}\,\mathbf{c},\ \mathbf{r}\}$$

Compositional Semantics (2)

The behaviour of processes in TransCCS can be understood by CCS-like "Clean" traces derived by the LTS that:

- consider only traces where all actions are eventually committed
- ignore transactional annotations on the traces

$$\mathcal{L}(\llbracket a.c.\mathrm{co}\ k + e \triangleright_k r \rrbracket) = \{\epsilon,\ \mathbf{a\,c},\ \mathbf{r}\}$$ (Non-prefix-closed set)

Compositional Semantics (2)

The behaviour of processes in TransCCS can be understood by CCS-like "Clean" traces derived by the LTS that:

- consider only traces where all actions are eventually committed
- ignore transactional annotations on the traces

$$\mathcal{L}(\llbracket a.c.co\ k + e \triangleright_k\ r \rrbracket) = \{\epsilon,\ \mathbf{a\,c},\ \mathbf{r}\} \qquad \text{(Non-prefix-closed set)}$$

$$\mathcal{L}(\mu X.\llbracket a.c.co\ k + e \triangleright_k\ X \rrbracket) = \{\epsilon,\ \mathbf{a\,c}\}$$

Compositional Semantics (2)

The behaviour of processes in TransCCS can be understood by CCS-like “Clean” traces derived by the LTS that:

- consider only traces where all actions are eventually committed
- ignore transactional annotations on the traces

$$\mathcal{L}(\ \llbracket a.c.co\ k + e\ \rhd_k\ r \rrbracket\ ) = \{\epsilon,\ \mathbf{a}\,\mathbf{c},\ \mathbf{r}\} \qquad \text{(Non-prefix-closed set)}$$

$$\mathcal{L}(\mu X.\ \llbracket a.c.co\ k + e\ \rhd_k\ X \rrbracket\ ) = \{\epsilon,\ \mathbf{a}\,\mathbf{c}\} \qquad \text{(Atomicity: all-or-nothing)}$$

Compositional Semantics (2)

The behaviour of processes in TransCCS can be understood by CCS-like "Clean" traces derived by the LTS that:

- consider only traces where all actions are eventually committed
- ignore transactional annotations on the traces

$$\mathcal{L}(\ [\![a.c.co\ k + e\ \rhd_k\ r]\!]\ ) = \{\epsilon,\ \mathbf{a\,c},\ \mathbf{r}\} \qquad \text{(Non-prefix-closed set)}$$

$$\mathcal{L}(\mu X.\ [\![a.c.co\ k + e\ \rhd_k\ X]\!]\ ) = \{\epsilon,\ \mathbf{a\,c}\} \qquad \text{(Atomicity: all-or-nothing)}$$

- enable compositional reasoning:
  - $\mathcal{L}(P \mid Q) = \mathcal{L}(P)\ \mathbf{zip}\ \mathcal{L}(Q)$
  - $\mathcal{L}(P) \subseteq \mathcal{L}(Q) \quad \text{implies} \quad \mathcal{L}(P \mid R) \subseteq \mathcal{L}(Q \mid R)$

Safety

Definition (Basic Observable)

$P\Downarrow_a$ iff there exists P' such that $P \to^* P' \mid a$

- Basic observable actions are permanent

Safety

Definition (Basic Observable)

$P{\Downarrow}_a$ iff there exists P' such that $P \rightarrow^* P' \mid a$

- Basic observable actions are permanent

Definition (P passes safety test T^{ω})

$P \ \mathrm{cannot}\ T^{\omega}$ when $P \mid T^{\omega}{\not\Downarrow}_{\omega}$

Safety

Definition (Basic Observable)

$P{\Downarrow}_a$ iff there exists P' such that $P \rightarrow^* P' \mid a$

- Basic observable actions are permanent

Definition (P passes safety test T^{ω})

$P \,\mathrm{cannot}\, T^{\omega}$ when $P \mid T^{\omega} \not\Downarrow_{\omega}$

Definition (Safety preservation)

$S \underset{\sim}{\sqsubset}_{\mathrm{safe}} I$ when $\forall T^{\omega}.\ S \,\mathrm{cannot}\, T^{\omega}$ implies $I \,\mathrm{cannot}\, T^{\omega}$

Safety

Definition (Basic Observable)

$P{\Downarrow}_a$ iff there exists P' such that $P \to^* P' \mid a$

- Basic observable actions are permanent

Definition (P passes safety test T^{ω})

$P \,\mathrm{cannot}\, T^{\omega}$ when $P \mid T^{\omega} \not\Downarrow_{\omega}$

Definition (Safety preservation)

$S \sqsubsetapprox_{\mathrm{safe}} I$ when $\forall T^{\omega}.$ $S \,\mathrm{cannot}\, T^{\omega}$ implies $I \,\mathrm{cannot}\, T^{\omega}$

Theorem (Characterization of safety preservation)

$S \sqsubsetapprox_{\mathrm{safe}} I$ *iff* $\mathcal{L}(S) \supseteq \mathcal{L}(I)$

Liveness

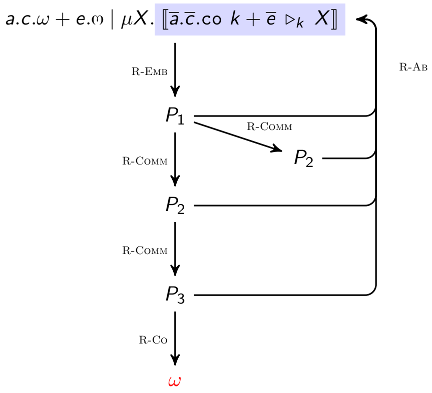
Definition (P Passes liveness Test T^ω [Rensink-Vogler'07])

$P \,\mathrm{shd}\, T^\omega$ when $\forall R.$ $P \mid T^\omega \rightarrow^* R$ implies $R{\Downarrow_\omega}$

Liveness

Definition (P Passes liveness Test T^ω [Rensink-Vogler'07])

$P \operatorname{shd} T^\omega$ when $\forall R.\ \ P \mid T^\omega \rightarrow^* R$ implies $R{\Downarrow_\omega}$



$$\begin{tikzcd}
a.c.\omega + e.\varpi \mid \mu X.\, \llbracket \overline{a}.\overline{c}.\mathrm{co}\ k + \overline{e} \rhd_k X \rrbracket \arrow[d, "\textsc{R-Emb}"'] & \\
P_1 \arrow[d, "\textsc{R-Comm}"'] \arrow[dr, "\textsc{R-Comm}"] \arrow[u, bend right=60, "\textsc{R-Ab}"'] & \\
& P_2 \arrow[uul, bend right=60] \\
P_2 \arrow[d, "\textsc{R-Comm}"'] \arrow[uuu, bend right=60] & \\
P_3 \arrow[d, "\textsc{R-Co}"'] \arrow[uuuu, bend right=60] & \\
\omega &
\end{tikzcd}$$

Liveness

Definition (P Passes liveness Test T^ω [Rensink-Vogler'07])

$P \,\mathrm{shd}\, T^\omega$ when $\forall R.\quad P \mid T^\omega \rightarrow^* R$ implies $R{\Downarrow_\omega}$

$$\begin{tikzcd}
a.c.\omega + e.\mho \mid \mu X.\, [\![\overline{a}.\overline{c}.\mathrm{co}\ k + \overline{e} \rhd_k X]\!] \arrow[d, "\textsc{R-Emb}"'] & \\
P_1 \arrow[u, bend right=60, "\textsc{R-Ab}"'] \arrow[rd, "\textsc{R-Comm}"] \arrow[d, "\textsc{R-Comm}"'] & \\
P_2 \arrow[uu, bend right=60, "\textsc{R-Ab}"'] \arrow[d, "\textsc{R-Comm}"'] & P_2 \arrow[luu, bend right=40, "\textsc{R-Ab}"'] \\
P_3 \arrow[uuu, bend right=60, "\textsc{R-Ab}"'] \arrow[d, "\textsc{R-Co}"'] & \\
\omega &
\end{tikzcd}$$

must testing
would consider
the infinite loop

Liveness

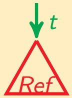
Definition (P passes liveness test T^ω [Rensink-Vogler'07])

$P \,\mathrm{shd}\, T^\omega$ when $\forall R.\ \ P \mid T^\omega \rightarrow^* R$ implies $R{\Downarrow_\omega}$

Definition (Tree Failures [Rensink-Vogler'07])

(t, Ref) is a **tree failure** of P when

$$\exists P'.\quad P \overset{t}{\Rightarrow}_{CL} P' \quad \text{and} \quad \mathcal{L}(P') \cap \mathit{Ref} = \emptyset$$



$\mathcal{F}(P) = \{(t, \mathit{Ref})$ tree failure of $P\}$

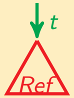
Liveness

Definition (P passes liveness test T^ω [Rensink-Vogler'07])

$P \,\mathrm{shd}\, T^\omega$ when $\forall R.\; P \mid T^\omega \rightarrow^* R$ implies $R{\Downarrow_\omega}$

Definition (Tree Failures [Rensink-Vogler'07])

(t, Ref) is a **tree failure** of P when
$\exists P'.\; P \overset{t}{\Rightarrow}_{CL} P'$ and $\mathcal{L}(P') \cap \mathit{Ref} = \emptyset$



$\mathcal{F}(P) = \{(t, \mathit{Ref})$ tree failure of $P\}$

- *Ref* is generally non-prefix-closed

Liveness

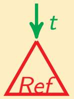
Definition (P passes liveness test T^ω [Rensink-Vogler'07])

$P \operatorname{shd} T^\omega$ when $\forall R.\ \ P \mid T^\omega \to^* R$ implies $R{\Downarrow_\omega}$

Definition (Tree Failures [Rensink-Vogler'07])

(t, Ref) is a **tree failure** of P when

$$\exists P'.\quad P \overset{t}{\Rightarrow}_{CL} P' \quad \text{and} \quad \mathcal{L}(P') \cap \mathit{Ref} = \emptyset$$



$\mathcal{F}(P) = \{(t, \mathit{Ref})$ tree failure of $P\}$

- Ref is generally non-prefix-closed

Theorem (Characterization of liveness preservation)

$S \sqsubseteq_{\text{live}} I$ *iff* $\mathcal{F}(S) \supseteq \mathcal{F}(I)$

Simple Examples

Let $S_{ab} = \mu X . \llbracket a.b.\mathrm{co}\ k \triangleright_k X \rrbracket$ $\quad \mathcal{L}(S_{ab}) = \{\epsilon, ab\}$

$$\mathcal{F}(S_{ab}) = \{(\epsilon, S \backslash ab), (ab, S) \mid S \subseteq A^*\}$$

Simple Examples

Let $S_{ab} = \mu X. \llbracket a.b.\mathsf{co}\ k \rhd_k X \rrbracket$ $\qquad \mathcal{L}(S_{ab}) = \{\epsilon, ab\}$

$$\mathcal{F}(S_{ab}) = \{(\epsilon, S \backslash ab), (ab, S) \mid S \subseteq A^*\}$$

- $S_{ab} \eqsim_{\text{safe}} I_1 = \llbracket a.b.\mathsf{co}\ k \rhd_k \mathbb{0} \rrbracket$ $\qquad \mathcal{L}(I_1) = \{\epsilon, ab\}$
  $S_{ab} \not\eqsim_{\text{live}} I_1$ $\qquad \mathcal{F}(I_1) = \{(\epsilon, S), (ab, S) \mid S \subseteq A^*\}$

Simple Examples

Let $S_{ab} = \mu X.\ [\![a.b.\text{co } k \triangleright_k X]\!]$ $\qquad \mathcal{L}(S_{ab}) = \{\epsilon, ab\}$

$$\mathcal{F}(S_{ab}) = \{(\epsilon, S \backslash ab), (ab, S) \mid S \subseteq A^*\}$$

- $S_{ab} \eqsim_{\text{safe}} I_1 = [\![a.b.\text{co } k \triangleright_k \mathbb{0}]\!]$ $\qquad \mathcal{L}(I_1) = \{\epsilon, ab\}$
  $S_{ab} \not\eqsim_{\text{live}} I_1$ $\qquad \mathcal{F}(I_1) = \{(\epsilon, S), (ab, S) \mid S \subseteq A^*\}$

- $S_{ab} \eqsim_{\text{safe}} I_2 = \mu X.\ [\![a.b.\text{co } k + e \triangleright_k X]\!]$ $\qquad \mathcal{L}(I_2) = \mathcal{L}(S_{ab})$
  $S_{ab} \eqsim_{\text{live}} I_2$ $\qquad \mathcal{F}(I_2) = \mathcal{F}(S_{ab})$

Comparison with CCS (1)

Safety in **TransCCS** is characterized by non-prefix-closed sets of traces
Safety in **CCS** is characterized by prefix-closed sets of traces

Comparison with CCS (1)

Safety in **TransCCS** is characterized by non-prefix-closed sets of traces
Safety in **CCS** is characterized by prefix-closed sets of traces

- **TransCCS** safety tests have the same distinguishing power as **CCS** safety tests
    - If in **CCS** $P \sqsubset\!\!\!\sim_{\text{safe}} Q$ then also in **TransCCS** $P \sqsubset\!\!\!\sim_{\text{safe}} Q$

Comparison with CCS (1)

Safety in **TransCCS** is characterized by non-prefix-closed sets of traces
Safety in **CCS** is characterized by prefix-closed sets of traces

- **TransCCS** safety tests have the same distinguishing power as **CCS** safety tests
  - If in **CCS** $P \sqsubset\!\!\!\sim_{\text{safe}} Q$ then also in **TransCCS** $P \sqsubset\!\!\!\sim_{\text{safe}} Q$
- No way to encode non-prefix-closed traces in **CCS**; thus no fully-abstract translation from **TransCCS** to **CCS**

Comparison with CCS (2)

Liveness in **TransCCS** is characterized by tree failures
Liveness in **CCS** is characterized by a more complex model [Rensink-Vogler'07]

Comparison with CCS (2)

Liveness in **TransCCS** is characterized by tree failures
Liveness in **CCS** is characterized by a more complex model [Rensink-Vogler'07]

- **TransCCS** liveness tests have more distinguishing power than **CCS** liveness tests
  - In **CCS** $a.(b.c + b.d) \sqsubseteq\!\!\!\sim_{\text{live}} a.b.c + a.b.d$
  - In **TransCCS** $a.(b.c + b.d) \not\sqsubseteq\!\!\!\sim_{\text{live}} a.b.c + a.b.d$
    - $(a, \{bd\}) \notin \mathcal{F}(a.(b.c + b.d))$
    - $(a, \{bd\}) \in \mathcal{F}(a.b.c + a.b.d)$
  - **TransCCS** distinguishing liveness test in the paper

Comparison with CCS (2)

Liveness in **TransCCS** is characterized by tree failures
Liveness in **CCS** is characterized by a more complex model [Rensink-Vogler'07]

- **TransCCS** liveness tests have more distinguishing power than **CCS** liveness tests
  - In **CCS** $a.(b.c + b.d) \sqsubseteq_{\text{live}} a.b.c + a.b.d$
  - In **TransCCS** $a.(b.c + b.d) \not\sqsubseteq_{\text{live}} a.b.c + a.b.d$
    - $(a, \{bd\}) \notin \mathcal{F}(a.(b.c + b.d))$
    - $(a, \{bd\}) \in \mathcal{F}(a.b.c + a.b.d)$
  - **TransCCS** distinguishing liveness test in the paper
- Thus no sound translation from **TransCCS** to **CCS** that is the identity on CCS terms

Also in [APLAS 2010]

- Canonical class of tests for liveness and safety
- See how restarting transactions add fault tolerance to CCS (Ex. 6)
- A sound, but incomplete bisimulation proof method, using the “clean” LTS transitions
- Many examples

Conclusions

Communicating transactions:

- Traditional transactions without the isolation requirement
  - No limit on communication or concurrency
- Simple safety and liveness theory
  - First theory of liveness in the presence of transactions
- **Future directions:** Reference implementation/evaluation of the construct in a programming language.

Advertisement

Joint Trinity/Microsoft Research PhD on extending Haskell with communicating transactions. We need a good student :)

ACD Properties

$$\begin{tikzcd}
a.b.\omega + e.\omega \mid \mu X.\, [\![\overline{a}.\overline{b}.\mathrm{co}\ k + \overline{e} \triangleright_k X]\!] \arrow[d, "\text{R-Emb}"'] & \\
P_1 \arrow[d, "\text{R-Comm}"'] \arrow[r, "\text{R-Comm}"] \arrow[u, bend right=60, "\text{R-Ab}"'] & P_2 \arrow[ul, bend right, "\text{R-Ab}"'] \\
P_2 \arrow[d, "\text{R-Comm}"'] \arrow[uu, bend right=70, "\text{R-Ab}"'] & \\
P_3 \arrow[d, "\text{R-Co}"'] \arrow[uuu, bend right=80, "\text{R-Ab}"'] & \\
\omega &
\end{tikzcd}$$

A commit step makes the effects of the transaction permanent (**Durability**)

An abort step:

- restarts the transaction
- rolls-back embedded processes to their state before embedding (**Consistency**)
- does not roll-back actions that happened before embedding
- does not affect non-embedded processes

The semantics of transactions transactions are non-prefix-closed traces (**Atomicity**).