Liveness of Communicating Transactions

Edsko de Vries

(joint work with Vasileios Koutavas and Matthew Hennessy)
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: “Nothing bad will happen” [Lamport’77]

- A safety property can be formulated as a safety test $T^\circ$ which signals on channel $\diamond$ when it detects the bad behaviour.
- $P$ passes the safety test $T^\circ$ when $P \mid T^\circ$ cannot output on $\diamond$.
  - This is the negation of passing a “may test” [DeNicola-Hennessy’84].
Liveness: “Something good will eventually happen” [Lamport’77]

- A liveness property can be formulated as a liveness test $T^\omega$ which detects and reports good behaviour on $\omega$.
- $P$ passes the liveness test $T^\omega$ when all future states of $P | T^\omega$ can output on $\omega$
  - 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: \[ P, Q ::= \sum \mu_i.P_i \text{ guarded choice} \]
\[ P \parallel Q \text{ parallel} \]
\[ \nu a.P \text{ hiding} \]
\[ \mu X.P \text{ recursion} \]
\[ \begin{bmatrix} P \triangleright_k Q \end{bmatrix} \text{ transaction (}k\text{ bound in } P) \]
\[ \text{co } k \text{ commit} \]
TransCCS [CONCUR 2010]

Syntax: \( P, Q \ ::= \sum \mu_i.P_i \) guarded choice

\( P | Q \) parallel

\( \nu a.P \) hiding

\( \mu X.P \) recursion

\( [P \triangleright_k Q] \) transaction (\( k \) bound in \( P \))

\( \text{co } k \) commit
TransCCS [CONCUR 2010]

Syntax: \[ P, Q ::= \sum \mu_i.P_i \quad \text{guarded choice} \]
\[ P | Q \quad \text{parallel} \]
\[ \nu a.P \quad \text{hiding} \]
\[ \mu X.P \quad \text{recursion} \]
\[ [P \triangleright_k Q] \quad \text{transaction (}k\text{ bound in }P) \]
\[ \text{co } k \quad \text{commit} \]
Syntax: \[ P, Q ::= \sum \mu_i.P_i \quad \text{guarded choice} \]
\[ P \parallel Q \quad \text{parallel} \]
\[ \nu a.P \quad \text{hiding} \]
\[ \mu X.P \quad \text{recursion} \]
\[ \llbracket P \triangleright_k Q \rrbracket \quad \text{transaction} (k \text{ bound in } P) \]
\[ \text{co } k \quad \text{commit} \]
TransCCS [CONCUR 2010]

Syntax: $P, Q ::= \sum \mu_i.P_i$ guarded choice

| $P \mid Q$ parallel |
| $\nu a.P$ hiding |
| $\mu X.P$ recursion |
| $[P \triangleright_k Q]$ transaction ($k$ bound in $P$) |
| $\text{co } k$ commit |
Overview of TransCCS
Safety and Liveness Theory

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 \( P \))
\( \mid \text{co } k \) commit

Main reductions:

R-Comm
\[ a_i = \overline{b_j} \]
\[ \sum_{i \in I} a_i.P_i \mid \sum_{j \in J} b_j.Q_j \to P_i \mid Q_j \]

R-CO
\[ [P \mid \text{co } k \triangleright_k Q] \to P \]

R-Emb
\[ k \notin R \]
\[ [P \triangleright_k Q] \mid R \to [P \mid R \triangleright_k Q \mid R] \]

R-Ab
\[ [P \triangleright_k Q] \to Q \]
Overview of TransCCS
Safety and Liveness Theory

TransCCS [CONCUR 2010]

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

Main reductions:

\[ R-\text{Comm} \]

\[ a_i = \overline{b_j} \]

\[ \sum_{i \in I} a_i.P_i \parallel \sum_{j \in J} b_j.Q_j \rightarrow P_i \parallel Q_j \]

\[ R-\text{Co} \]

\[ [P | co k \triangleright_k Q] \rightarrow P \]

\[ R-\text{Emb} \]

\[ k \notin R \]

\[ [P \triangleright_k Q] | R \rightarrow [P | R \triangleright_k Q | R] \]

\[ R-\text{Ab} \]

\[ [P \triangleright_k Q] \rightarrow Q \]
Overview of TransCCS
Safety and Liveness Theory

TransCCS [CONCUR 2010]

Syntax: \[ P, Q ::= \sum_{i} \mu_{i}.P_{i} \text{ guarded choice} \]
\[ P | Q \text{ parallel} \]
\[ \nu a.P \text{ hiding} \]
\[ \mu X.P \text{ recursion} \]
\[ [P \triangleright_{k} Q] \text{ transaction (} k \text{ bound in } P \text{)} \]
\[ \text{co } k \text{ commit} \]

Main reductions:

R-Comm
\[ a_{i} = \overline{b_{j}} \]
\[ \sum_{i \in I} a_{i}.P_{i} | \sum_{j \in J} b_{j}.Q_{j} \rightarrow P_{i} | Q_{j} \]

R-Co
\[ [P | \text{co } k \triangleright_{k} Q] \rightarrow P \]

R-Emb
\[ k \notin R \]
\[ [P \triangleright_{k} Q] | R \rightarrow [P | R \triangleright_{k} Q | R] \]

R-Ab
\[ [P \triangleright_{k} Q] \rightarrow Q \]
Simple Example

\[ a.c.\omega + e.\omega \mid [\bar{a}.\bar{c}.\bar{c} \ k + e \ \triangleright_k r] \]
Simple Example

\[ a.c.\omega + e.\omega \mid [\overline{a.c.co} \ k + \overline{e} \triangleright_k r] \]
Simple Example

\[ a.c.\omega + e.\omega \mid \bar{a}.\bar{c}.co \ k + e \ \triangleright_k \ r \]

\[ \text{R-EMB} \]

\[ \text{Liveness of Communicating Transactions} \]
Simple Example

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

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

\[ \xrightarrow{R-\text{COMM}} [c.\omega \mid \overline{c}.co k \triangleright_k a.c.\omega + e.\omega \mid r] \]
Simple Example

\[
\begin{align*}
\text{R-EMB} & \quad \to & \quad [a.c.\omega + e.\omega \mid \overline{a.c}.co \ k + \overline{e} \triangleright_k \ r] \\
\text{R-COMM} & \quad \to & \quad [c.\omega \mid \overline{c}.co \ k \triangleright_k a.c.\omega + e.\omega \mid r] \\
\text{R-COMM} & \quad \to & \quad [\omega \mid co \ k \triangleright_k a.c.\omega + e.\omega \mid r]
\end{align*}
\]
Simple Example

a.c.ω + e.ω | [\bar{a}.c.co k + e \triangleright_k r]

\[\text{R-EMB} \quad \frac{}{[a.c.ω + e.ω | \bar{a}.c.co k + e \triangleright_k a.c.ω + e.ω | r]}\]

\[\text{R-COMM} \quad \frac{}{[c.ω | \bar{c}.co k \triangleright_k a.c.ω + e.ω | r]}\]

\[\text{R-COMM} \quad \frac{}{[ω | co k \triangleright_k a.c.ω + e.ω | r]}\]

\[\text{R-Co} \quad \frac{}{ω}\]
Simple Example

\[
\begin{align*}
    & a.c.\omega + e.\omega \\
    \xrightarrow{R-\text{EMB}} & [a.c.co \ k + e \ △_k \ r] \\
    \xrightarrow{R-\text{COMM}} & [c.\omega \ | \ \overline{c}.co \ k \ △_k \ a.c.\omega + e.\omega \ | \ r] \\
    \xrightarrow{R-\text{COMM}} & [\omega \ | \ co \ k \ △_k \ a.c.\omega + e.\omega \ | \ r] \\
    \xrightarrow{R-\text{Co}} & \omega
\end{align*}
\]
Simple Example (a second trace)

\[ a \cdot c \cdot \omega + e \cdot \omega \mid [\overline{a} \cdot \overline{c} \cdot \text{co} \ k + \overline{e} \ \triangleright_k \ r] \]
Simple Example (a second trace)

\[
\begin{align*}
R-\text{EMB} & \quad [a.c.\omega + e.\omega \mid [\overline{a.c}.co \ k + \overline{e} \triangleright_k r]] \\
& \quad [a.c.\omega + e.\omega \mid \overline{a.c}.co \ k + \overline{e} \triangleright_k a.c.\omega + e.\omega \mid r]
\end{align*}
\]
Simple Example (a second trace)

\[ a.c.\omega + e.\omega \mid [\overline{a.c}.co \ k + \overline{e} \ ▽_k \ r] \]

\[ \xrightarrow{R-EMB} \ [a.c.\omega + e.\omega \mid \overline{a.c}.co \ k + \overline{e} \ ▽_k \ a.c.\omega + e.\omega \mid r] \]

\[ \xrightarrow{R-COMM} \ [\omega \ ▽_k \ a.c.\omega + e.\omega \mid r] \]
Simple Example (a second trace)

\[ a.c.\omega + e.\omega \mid \left[ \overline{a.c}.co \ k + \overline{e} \ \triangleright_k \ r \right] \]

R-\text{EMB} \quad \begin{array}{c}
R-\text{COMM} \quad \left[ \begin{array}{c}
\left[ a.c.\omega + e.\omega \mid \overline{a.c}.co \ k + \overline{e} \ \triangleright_k \ a.c.\omega + e.\omega \mid r \right]
\end{array} \right]
\end{array}

\left[ \begin{array}{c}
\left[ a.c.\omega + e.\omega \mid r \right]
\end{array} \right]
Simple Example (a second trace)

\[ a.c.\omega + e.\omega \mid \begin{array}{l}
\ll a.c.\text{co } k + \bar{e} \triangleright_k r \\
\end{array} \]

\[ \frac{\text{R-EMB}}{\longrightarrow} \begin{array}{l}
[a.c.\omega + e.\omega \mid \ll a.c.\text{co } k + \bar{e} \triangleright_k a.c.\omega + e.\omega \mid r]
\end{array} \]

\[ \frac{\text{R-COMM}}{\longrightarrow} \begin{array}{l}
\ll \omega \triangleright_k a.c.\omega + e.\omega \mid r
\end{array} \]

\[ \frac{\text{R-AB}}{\longrightarrow} a.c.\omega + e.\omega \mid r \]
Overview of TransCCS
Safety and Liveness Theory

Simple Example (a second trace)

\[
\begin{align*}
    & a.c.\omega + e.\omega \mid [\bar{a}.\bar{c}.co \ k + \bar{e} \ △_k \ r] \\
\xrightarrow{\text{R-EMB}} & [a.c.\omega + e.\omega \mid \bar{a}.\bar{c}.co \ k + \bar{e} \ △_k \ a.c.\omega + e.\omega \mid r] \\
\xrightarrow{\text{R-COMM}} & \left[\begin{array}{c}
\omega \\
\end{array}\right] \quad △_k \ a.c.\omega + e.\omega \mid r \\
\xrightarrow{\text{R-Ab}} & a.c.\omega + e.\omega \mid r \quad \text{(The environment is restored)}
\end{align*}
\]
Simple Example (all traces)

\[ a.c.\omega + e.\omega \mid [\overline{a.c.co \; k + e} \triangleright_k r] \xrightarrow{R-AB} a.c.\omega + e.\omega \mid r \]
Edsko de Vries   Liveness of Communicating Transactions
Restarting transactions

\[ a.c.\omega + e.\omega \mid \mu X. [\overline{a.c}.co \; k + e \triangleright_k X] \]
Restarting transactions

\[ a . c . \omega + e . \omega \mid \mu X. \left[ \overline{a . c . c o} k + \overline{e} \triangleright_k X \right] \]

Edsko de Vries

Liveness of Communicating Transactions
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: $[P \triangleright_k Q]$. 
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: $[[P \triangleright_k Q]]$

\[
\begin{array}{c|c|c|c}
\text{emb } k & \text{emb } k & \text{emb } k \\
\hline
(a.c.\omega + e.\omega) & [a.c.\omega + e.\omega] & [a.c.\omega + e.\omega] \\
\hline
k(a) & \triangleright_k a.c.\omega + e.\omega & a.c.\omega + e.\omega \\
\hline
k(c) & \triangleright_k a.c.\omega + e.\omega & a.c.\omega + e.\omega \\
\hline
\text{co } k & \triangleright_k a.c.\omega + e.\omega & a.c.\omega + e.\omega \\
\hline
\omega & [\omega] & a.c.\omega + e.\omega \\
\hline
\end{array}
\]

\[
\begin{array}{c|c|c|c}
\hline
\text{emb } k & \text{emb } k & \text{emb } k \\
\hline
[a.c.\omega + e.\omega] & [a.c.\omega + e.\omega] & [a.c.\omega + e.\omega] \\
\hline
k(\bar{a}) & k(\bar{a}) & k(\bar{a}) \\
\hline
k(\bar{c}) & k(\bar{c}) & k(\bar{c}) \\
\hline
\text{co } k & \text{co } k & \text{co } k \\
\hline
\omega & [\omega] & [\omega] \\
\hline
\end{array}
\]
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:
  \[
  \left[ P \triangleright_k Q \right]^\circ
  \]

\[
\begin{align*}
\text{emb } k & \quad \Rightarrow \quad a.c.\omega + e.\top \\
\text{emb } k & \quad \Rightarrow \quad \left[ a.c.\omega + e.\top \triangleright_k a.c.\omega + e.\top \right]^\circ \\
\text{emb } k & \quad \Rightarrow \quad \left[ a.c.\omega + e.\top \triangleright_k a.c.\omega + e.\top \right]^\circ \\
\text{emb } k & \quad \Rightarrow \quad \left[ a.c.\omega + e.\top \triangleright_k a.c.\omega + e.\top \right]^\circ \\
ab k & \quad \Rightarrow \quad a.c.\omega + e.\top \\
\end{align*}
\]
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
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

\[
\mathcal{L}(\lbrack a.c.co \ k + e \ \triangleright_k \ r \rbrack) = \{\epsilon, \ a, \ c, \ r\}
\]
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

\[ \mathcal{L}(\llbracket a.c.co \; k + e \upharpoonright_k r \rrbracket) = \{\epsilon, a, c, r\} \]  (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

\[ L(\llbracket a.c.co \ k + e \triangleright_k r \rrbracket) = \{\epsilon, \ a, \ c, \ r\} \quad \text{(Non-prefix-closed set)} \]

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

\[
\mathcal{L}(\mu X. \boxed{a.c.co\ k + e \triangleright_k X}) = \{\epsilon, a, c\} \quad \text{(Atomicity: all-or-nothing)}
\]

\[
\mathcal{L}(\boxed{a.c.co\ k + e \triangleright_k r}) = \{\epsilon, a, c, r\} \quad \text{(Non-prefix-closed set)}
\]
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

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

\[ \mathcal{L}(\mu X. \llbracket a.c.co \ k + e \triangleright_k X \rrbracket) = \{ \epsilon, a \ c \} \quad (\text{Atomicity: all-or-nothing}) \]

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

Definition (Basic Observable)

\[ P \downarrow_a \text{ iff } \exists P' \text{ such that } P \rightarrow^* P' \mid a \]

- Basic observable actions are permanent
Overview of TransCCS
Safety and Liveness Theory

Safety

Definition (Basic Observable)

\[ P \downarrow_a \iff \exists P' \text{ such that } P \xrightarrow{\ast} P' \mid a \]

- Basic observable actions are permanent

Definition (\( P \) passes safety test \( T^{(\omega)} \))

\[ P \text{ cannot } T^{(\omega)} \text{ when } P \mid T^{(\omega)} \downarrow^{(\omega)} \]
Overview of TransCCS
Safety and Liveness Theory

Safety

Definition (Basic Observable)

\[ P \downarrow_a \iff \text{there exists } P' \text{ such that } P \rightarrow^* P' \mid a \]

- Basic observable actions are permanent

Definition (\( P \) passes safety test \( T^{(\omega)} \))

\[ P \text{ cannot } T^{(\omega)} \text{ when } P \mid T^{(\omega)} \downarrow^{(\omega)} \]

Definition (Safety preservation)

\[ S \prec_{\text{safe}} I \text{ when } \forall T^{(\omega)}. \ S \text{ cannot } T^{(\omega)} \text{ implies } I \text{ cannot } T^{(\omega)} \]
Safety

Definition (Basic Observable)

\[ P \downarrow_a \text{ iff there exists } P' \text{ such that } P \rightarrow^* P' \mid a \]

- Basic observable actions are permanent

Definition (\( P \) passes safety test \( T^\omega \))

\[ P \text{ cannot } T^\omega \text{ when } P \mid T^\omega \downarrow^\omega_a \]

Definition (Safety preservation)

\[ S \models_{\text{safe}} I \text{ when } \forall T^\omega. \ S \text{ cannot } T^\omega \text{ implies } I \text{ cannot } T^\omega \]

Theorem (Characterization of safety preservation)

\[ S \models_{\text{safe}} I \text{ iff } \mathcal{L}(S) \supseteq \mathcal{L}(I) \]
Liveness

**Definition (P Passes liveness Test $T^\omega$ [Rensink-Vogler’07])**

$P \text{ shd } T^\omega \quad \text{when} \quad \forall R. \quad P \mid T^\omega \rightarrow^* R \quad \text{implies} \quad R\downarrow^\omega$
Definition (\(P\) Passes liveness Test \(T^\omega\) [Rensink-Vogler’07])

\[ P \text{ shd } T^\omega \text{ when } \forall R. \ P \mid T^\omega \rightarrow^* R \text{ implies } R \Downarrow^\omega \]

\[ a.c.\omega + e.\omega \mid \mu X. [\overline{a.c}.co k + e \triangleright_k X] \]
Liveness

**Definition (P Passes liveness Test \( T^\omega \) [Rensink-Vogler’07])**

\( P \text{ shd } T^\omega \) when \( \forall R. \ P \mid T^\omega \rightarrow^* R \) implies \( R \Downarrow^\omega \)

\[ a.c.\omega + e.\omega \mid \mu X. [\overline{a.c}.\text{co} k + \overline{e} \triangleright_k X] \]

must testing would consider the infinite loop
Liveness

**Definition (\(P\) passes liveness test \(T^\omega\) [Rensink-Vogler’07])**

\[ P \text{ shd } T^\omega \text{ when } \forall R. \ P \divides T^\omega \rightarrow^* R \implies R \downarrow^\omega \]

**Definition (Tree Failures [Rensink-Vogler’07])**

\((t, \text{Ref})\) is a **tree failure** of \(P\) when

\[ \exists P'. \ P \xrightarrow{t} \mathcal{CL} P' \text{ and } \mathcal{L}(P') \cap \text{Ref} = \emptyset \]

\[ \mathcal{F}(P) = \{(t, \text{Ref}) \text{ tree failure of } P\} \]
Liveness

Definition (\(P\) passes liveness test \(T^\omega\) [Rensink-Vogler’07])

\(P\) shd \(T^\omega\) when \(\forall R. \ P \mid T^\omega \rightarrow^* R\) implies \(R \downarrow^\omega\)

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

\((t, \text{Ref})\) is a tree failure of \(P\) when

\(\exists P'. \ P \xrightarrow{t} CL P'\) and \(\mathcal{L}(P') \cap \text{Ref} = \emptyset\)

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

- \(\text{Ref}\) is generally non-prefix-closed
**Liveness**

**Definition (P passes liveness test \(T^\omega\) [Rensink-Vogler’07])**

\(P\) **shd** \(T^\omega\) when \(\forall R. \ P \mid T^\omega \rightarrow^* R\) implies \(R \downarrow^\omega\)

**Definition (Tree Failures [Rensink-Vogler’07])**

\((t, \text{Ref})\) is a **tree failure** of \(P\) when

\[\exists P'. \ P \xrightarrow{t} \text{CL} \ P' \text{ and } \mathcal{L}(P') \cap \text{Ref} = \emptyset\]

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

- \(\text{Ref}\) is generally non-prefix-closed

**Theorem (Characterization of liveness preservation)**

\(S \xrightarrow{\sim} \text{live} \ I \text{ iff } \mathcal{F}(S) \supseteq \mathcal{F}(I)\)
Simple Examples

Let \( S_{ab} = \mu X. \ [a.b.co \ k \ X] \)

\[ \mathcal{L}(S_{ab}) = \{\epsilon, ab\} \]

\[ \mathcal{F}(S_{ab}) = \{ (\epsilon, S \setminus ab), (ab, S) \mid S \subseteq A^* \} \]
Simple Examples

Let \( S_{ab} = \mu X. [a.b.co k \triangleright_k X] \)

\[ \mathcal{L}(S_{ab}) = \{ \epsilon, ab \} \]

\[ \mathcal{F}(S_{ab}) = \{ (\epsilon, S \setminus ab), (ab, S) \mid S \subseteq A^* \} \]

\[ S_{ab} \tilde{\sim}_{\text{safe}} I_1 = \mu X. [a.b.co k \triangleright_k \emptyset] \]

\[ \mathcal{L}(I_1) = \{ \epsilon, ab \} \]

\[ \mathcal{F}(I_1) = \{ (\epsilon, S), (ab, S) \mid S \subseteq A^* \} \]
Simple Examples

Let \( S_{ab} = \mu X. \left[ a \cdot b \cdot \text{co } k \triangleright_k X \right] \)

\[ \mathcal{L}(S_{ab}) = \{ \epsilon, ab \} \]
\[ \mathcal{F}(S_{ab}) = \{ (\epsilon, S \setminus ab), (ab, S) \mid S \subseteq A^* \} \]

\[ \triangleright S_{ab} \sim_{\text{safe}} l_1 = \left[ a \cdot b \cdot \text{co } k \triangleright_k 0 \right] \]
\[ S_{ab} \not\sim_{\text{live}} l_1 \]

\[ \mathcal{L}(l_1) = \{ \epsilon, ab \} \]
\[ \mathcal{F}(l_1) = \{ (\epsilon, S), (ab, S) \mid S \subseteq A^* \} \]

\[ \triangleright S_{ab} \sim_{\text{safe}} l_2 = \mu X. \left[ a \cdot b \cdot \text{co } k + e \triangleright_k X \right] \]
\[ S_{ab} \sim_{\text{live}} l_2 \]

\[ \mathcal{L}(l_2) = \mathcal{L}(S_{ab}) \]
\[ \mathcal{F}(l_2) = \mathcal{F}(S_{ab}) \]
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.
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 $P \simeq_{\text{safe}} Q$ then also in TransCCS $P \simeq_{\text{safe}} Q$.
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** $P \overset{\text{safe}}{\sim} Q$ then also in **TransCCS** $P \overset{\text{safe}}{\sim} Q$.
- 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**, \( a \cdot (b \cdot c + b \cdot d) \stackrel{live}{\asymp} a \cdot b \cdot c + a \cdot b \cdot d \)
  - In **TransCCS**, \( a \cdot (b \cdot c + b \cdot d) \not\stackrel{live}{\asymp} a \cdot b \cdot c + a \cdot b \cdot d \)
    - \((a, \{bd\}) \notin \mathcal{F}(a \cdot (b \cdot c + b \cdot d))\)
    - \((a, \{bd\}) \in \mathcal{F}(a \cdot b \cdot c + a \cdot b \cdot d)\)
- **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** \( a.(b.c + b.d) \not\simlive a.b.c + a.b.d \)
  - In **TransCCS** \( a.(b.c + b.d) \not\simlive a.b.c + a.b.d \)
    - \( (a, \{bd\}) \notin \mathcal{F}(a.(b.c + b.d)) \)
    - \( (a, \{bd\}) \in \mathcal{F}(a.b.c + a.b.d) \)
  - **TransCCS** distinguishing liveness test in the paper
- Thus no sound translation from **TransCCS** to **CCS** that is the identity on CCS terms
- 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 :)

| Edsko de Vries | Liveness of Communicating Transactions |
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 are non-prefix-closed traces (Atomicity).