Modelling, Specification and Verification of Reactive Systems

Strong and Weak Bisimulation Equivalence

- Behavioural equivalences
- Strong bisimilarity and bisimulation games
- Properties of strong bisimilarity
- Ditto for weak bisimilarity
- Example: a communication protocol and its modelling in CCS
- Concurrency workbench (CWB)
**Behavoural Equivalence**

**Implementation**

\[
CM \overset{\text{def}}{=} \text{coin.coffee.CM} \\
CS \overset{\text{def}}{=} \text{pub.coin.coffee.CS} \\
Uni \overset{\text{def}}{=} (CM | CS) \setminus \{\text{coin, coffee}\}
\]

**Specification**

\[
Spec \overset{\text{def}}{=} \text{pub.Spec}
\]

**Question**

Are the processes \(Uni\) and \(Spec\) “behaviourally equivalent”?  

\[Uni \equiv Spec\]
Behavoural Equivalence

**Implementation**

\[
CM \overset{\text{def}}{=} \text{coin}\cdot\text{coffee}\cdot CM
\]

\[
CS \overset{\text{def}}{=} \text{pub}\cdot\text{coin}\cdot\text{coffee}\cdot CS
\]

\[
Uni \overset{\text{def}}{=} (CM | CS) \setminus \{\text{coin, coffee}\}
\]

**Specification**

\[
Spec \overset{\text{def}}{=} \text{pub}\cdot\text{Spec}
\]

**Question**

Are the processes \(Uni\) and \(Spec\) “behaviourally equivalent”?

\[Uni \equiv Spec\]
Behavourial Equivalence

**Implementation**

\[ CM \overset{\text{def}}{=} \text{coin.coffee.CM} \]
\[ CS \overset{\text{def}}{=} \text{pub.coin.coffee.CS} \]
\[ Uni \overset{\text{def}}{=} (CM \mid CS) \setminus \{\text{coin, coffee}\} \]

**Specification**

\[ Spec \overset{\text{def}}{=} \text{pub.Spec} \]

**Question**

Are the processes \( Uni \) and \( Spec \) “behaviourally equivalent”?

\[ Uni \equiv Spec \]
Goals

What should a reasonable behavioural equivalence satisfy?

- Abstract from states (consider only the behaviour – actions)
- Abstract from nondeterminism
- Abstract from internal behaviour

What else should a reasonable behavioural equivalence satisfy?

- Reflexivity: \( P \equiv P \) for each process \( P \)
- Transitivity: \( \text{Spec}_0 \equiv \text{Spec}_1 \equiv \text{Spec}_2 \equiv \cdots \equiv \text{Impl} \) gives that \( \text{Spec}_0 \equiv \text{Impl} \)
- Symmetry: \( P \equiv Q \) iff \( Q \equiv P \)
Goals

What should a reasonable behavioural equivalence satisfy?

- Abstract from states (consider only the behaviour – actions)
- Abstract from nondeterminism
- Abstract from internal behaviour

What else should a reasonable behavioural equivalence satisfy?

- **Reflexivity:** \( P \equiv P \) for each process \( P \)
- **Transitivity:** \( \text{Spec}_0 \equiv \text{Spec}_1 \equiv \text{Spec}_2 \equiv \cdots \equiv \text{Impl} \) gives that \( \text{Spec}_0 \equiv \text{Impl} \)
- **Symmetry:** \( P \equiv Q \) iff \( Q \equiv P \)
Congruence Property

\[ P \equiv Q \text{ implies that } C(P) \equiv C(Q) \]
Congruence Property

\[ P \equiv Q \text{ implies that } C(P) \equiv C(Q) \]
Trace Equivalence

Let \((\text{Proc}, \text{Act}, \{ \xrightarrow{a} | a \in \text{Act} \})\) be an LTS.

Trace Set for \(s \in \text{Proc}\)

\[
\text{Traces}(s) = \{ w \in \text{Act}^* | \exists s' \in \text{Proc}. \, s \xrightarrow{w} s' \}
\]

Let \(s \in \text{Proc}\) and \(t \in \text{Proc}\).

Trace Equivalence

We say that \(s\) and \(t\) are trace equivalent \((s \equiv_t t)\) if and only if

\[
\text{Traces}(s) = \text{Traces}(t)
\]

Is this a “good” behavioural equivalence?
Trace Equivalence

Let \((Proc, Act, \{a \rightarrow \mid a \in Act}\})\) be an LTS.

### Trace Set for \(s \in Proc\)

\[
\text{Traces}(s) = \{w \in Act^* \mid \exists s' \in Proc. s \xrightarrow{w} s'\}
\]

Let \(s \in Proc\) and \(t \in Proc\).

### Trace Equivalence

We say that \(s\) and \(t\) are trace equivalent \((s \equiv_t t)\) if and only if

\[
\text{Traces}(s) = \text{Traces}(t)
\]

Is this a “good” behavioural equivalence?
Main Idea

Two processes are behaviorally equivalent if and only if an external observer cannot tell them apart.
Main Idea

Two processes are behaviorally equivalent if and only if an external observer cannot tell them apart.
Main Idea

Two processes are behaviorally equivalent if and only if an external observer cannot tell them apart.
Black-Box Experiments

<table>
<thead>
<tr>
<th>Experiment in A</th>
<th>Experiment in B</th>
<th>Experiment in B</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>coin</strong></td>
<td><strong>tea</strong></td>
<td><strong>coffee</strong></td>
</tr>
<tr>
<td><strong>coin</strong></td>
<td><strong>tea</strong></td>
<td><strong>coffee</strong></td>
</tr>
</tbody>
</table>

Main Idea

Two processes are behaviorally equivalent if and only if an external observer cannot tell them apart.
Black-Box Experiments

Experiment in $A$

\[
\begin{array}{c}
\text{press coin} \\
\text{coin} \quad \text{tea} \quad \text{coffee}
\end{array}
\]

Experiment in $B$

\[
\begin{array}{c}
\text{press coin} \\
\text{coin} \quad \text{tea} \quad \text{coffee}
\end{array}
\]

Experiment in $B$

\[
\begin{array}{c}
\text{press coin} \\
\text{coin} \quad \text{tea} \quad \text{coffee}
\end{array}
\]

Main Idea

Two processes are behaviorally equivalent if and only if an external observer cannot tell them apart.
Strong Bisimilarity

Let \((\text{Proc}, \text{Act}, \{ \xrightarrow{a} \mid a \in \text{Act} \})\) be an LTS.

**Strong Bisimulation**

A binary relation \(R \subseteq \text{Proc} \times \text{Proc}\) is a strong bisimulation iff whenever \((s, t) \in R\) then for each \(a \in \text{Act}\):

- if \(s \xrightarrow{a} s'\) then \(t \xrightarrow{a} t'\) for some \(t'\) such that \((s', t') \in R\)
- if \(t \xrightarrow{a} t'\) then \(s \xrightarrow{a} s'\) for some \(s'\) such that \((s', t') \in R\).

Two processes \(p_1, p_2 \in \text{Proc}\) are strongly bisimilar \((p_1 \sim p_2)\) if and only if there exists a strong bisimulation \(R\) such that \((p_1, p_2) \in R\).

\[\sim = \bigcup \{R \mid R \text{ is a strong bisimulation}\}\]
Strong Bisimilarity

Let \((Proc, Act, \{\xrightarrow{a} \mid a \in Act\})\) be an LTS.

**Strong Bisimulation**

A binary relation \(R \subseteq Proc \times Proc\) is a strong bisimulation iff whenever \((s, t) \in R\) then for each \(a \in Act\):
- if \(s \xrightarrow{a} s'\) then \(t \xrightarrow{a} t'\) for some \(t'\) such that \((s', t') \in R\)
- if \(t \xrightarrow{a} t'\) then \(s \xrightarrow{a} s'\) for some \(s'\) such that \((s', t') \in R\).

**Strong Bisimilarity**

Two processes \(p_1, p_2 \in Proc\) are strongly bisimilar \((p_1 \sim p_2)\) if and only if there exists a strong bisimulation \(R\) such that \((p_1, p_2) \in R\).

\[
\sim = \bigcup \{R \mid R \text{ is a strong bisimulation}\}
\]
Basic Properties of Strong Bisimilarity

Theorem

\sim \text{ is an equivalence relation (reflexive, symmetric and transitive)}

Theorem

\sim \text{ is the largest strong bisimulation}

Theorem

s \sim t \text{ if and only if for each } a \in \text{Act:}

- if \(s \xrightarrow{a} s'\) then \(t \xrightarrow{a} t'\) for some \(t'\) such that \(s' \sim t'\)
- if \(t \xrightarrow{a} t'\) then \(s \xrightarrow{a} s'\) for some \(s'\) such that \(s' \sim t'\).
Basic Properties of Strong Bisimilarity

Theorem

\( \sim \) is an equivalence relation (reflexive, symmetric and transitive)

Theorem

\( \sim \) is the largest strong bisimulation

Theorem

\( s \sim t \) if and only if for each \( a \in \text{Act} \):

- if \( s \xrightarrow{a} s' \) then \( t \xrightarrow{a} t' \) for some \( t' \) such that \( s' \sim t' \)
- if \( t \xrightarrow{a} t' \) then \( s \xrightarrow{a} s' \) for some \( s' \) such that \( s' \sim t' \).
Basic Properties of Strong Bisimilarity

Theorem

\(\sim\) is an equivalence relation (reflexive, symmetric and transitive)

Theorem

\(\sim\) is the largest strong bisimulation

Theorem

\(s \sim t\) if and only if for each \(a \in \text{Act}\):

- if \(s \xrightarrow{a} s'\) then \(t \xrightarrow{a} t'\) for some \(t'\) such that \(s' \sim t'\)
- if \(t \xrightarrow{a} t'\) then \(s \xrightarrow{a} s'\) for some \(s'\) such that \(s' \sim t'\).
How to Show Nonbisimilarity?

To prove that $s \not\sim t$:

- Enumerate all binary relations and show that none of them at the same time contains $(s, t)$ and is a strong bisimulation. (Expensive: $2^{|Proc|^2}$ relations.)
- Make certain observations which enable us to disqualify many bisimulation candidates in one step.
- Use the game characterization of strong bisimilarity.
How to Show Nonbisimilarity?

To prove that $s \not\sim t$:

- Enumerate all binary relations and show that none of them at the same time contains $(s, t)$ and is a strong bisimulation. (Expensive: $2^{\left|Proc\right|^2}$ relations.)

- Make certain observations which enable us to disqualify many bisimulation candidates in one step.

- Use the game characterization of strong bisimilarity.
How to Show Nonbisimilarity?

To prove that \( s \not\sim t \):

- Enumerate all binary relations and show that none of them at the same time contains \((s, t)\) and is a strong bisimulation. (Expensive: \(2^{|Proc|^2}\) relations.)
- Make certain observations which enable us to disqualify many bisimulation candidates in one step.
- Use the game characterization of strong bisimilarity.
How to Show Nonbisimilarity?

To prove that $s \not\sim t$:

- Enumerate all binary relations and show that none of them at the same time contains $(s, t)$ and is a strong bisimulation. (Expensive: $2^{|Proc|^2}$ relations.)
- Make certain observations which enable us to disqualify many bisimulation candidates in one step.
- Use the game characterization of strong bisimilarity.
Strong Bisimulation Game

Let \((\mathit{Proc}, \mathit{Act}, \{a \xrightarrow{a} \mid a \in \mathit{Act}\})\) be an LTS and \(s, t \in \mathit{Proc}\).

We define a two-player game of an ‘attacker’ and a ‘defender’ starting from \(s\) and \(t\).

- The game is played in rounds, and configurations of the game are pairs of states from \(\mathit{Proc} \times \mathit{Proc}\).
- In every round exactly one configuration is called current. Initially the configuration \((s, t)\) is the current one.

**Intuition**

The defender wants to show that \(s\) and \(t\) are strongly bisimilar while the attacker aims at proving the opposite.
Let \((\text{Proc}, \text{Act}, \{ \xrightarrow{a} | a \in \text{Act}\})\) be an LTS and \(s, t \in \text{Proc}\).

We define a two-player game of an ‘attacker’ and a ‘defender’ starting from \(s\) and \(t\).

- The game is played in rounds, and configurations of the game are pairs of states from \(\text{Proc} \times \text{Proc}\).
- In every round exactly one configuration is called current. Initially the configuration \((s, t)\) is the current one.

**Intuition**

The defender wants to show that \(s\) and \(t\) are strongly bisimilar while the attacker aims at proving the opposite.
Rules of the Bisimulation Games

Game Rules

In each round the players change the current configuration as follows:

1. the attacker chooses one of the processes in the current configuration and makes an $a \rightarrow$-move for some $a \in Act$, and
2. the defender must respond by making an $a \rightarrow$-move in the other process under the same action $a$.

The newly reached pair of processes becomes the current configuration. The game then continues by another round.

Result of the Game

- If one player cannot move, the other player wins.
- If the game is infinite, the defender wins.
Rules of the Bisimulation Games

**Game Rules**

In each round the players change the current configuration as follows:

1. the attacker chooses one of the processes in the current configuration and makes an $\xrightarrow{a}$-move for some $a \in \text{Act}$, and
2. the defender must respond by making an $\xrightarrow{a}$-move in the other process under the same action $a$.

The newly reached pair of processes becomes the current configuration. The game then continues by another round.

**Result of the Game**

- If one player cannot move, the other player wins.
- If the game is infinite, the defender wins.
Game Characterization of Strong Bisimilarity

Theorem

- States $s$ and $t$ are strongly bisimilar if and only if the defender has a **universal** winning strategy starting from the configuration $(s, t)$.
- States $s$ and $t$ are not strongly bisimilar if and only if the attacker has a **universal** winning strategy starting from the configuration $(s, t)$.

Remark

The bisimulation game can be used to prove both bisimilarity and nonbisimilarity of two processes. It very often provides elegant arguments for the negative case.
Game Characterization of Strong Bisimilarity

**Theorem**

- States $s$ and $t$ are strongly bisimilar if and only if the defender has a **universal** winning strategy starting from the configuration $(s, t)$.
- States $s$ and $t$ are not strongly bisimilar if and only if the attacker has a **universal** winning strategy starting from the configuration $(s, t)$.

**Remark**

The bisimulation game can be used to prove both bisimilarity and nonbisimilarity of two processes. It very often provides elegant arguments for the negative case.
Theorem

Let $P$ and $Q$ be CCS processes such that $P \sim Q$. Then

- $\alpha.P \sim \alpha.Q$ for each action $\alpha \in \text{Act}$
- $P + R \sim Q + R$ and $R + P \sim R + Q$ for each CCS process $R$
- $P \parallel R \sim Q \parallel R$ and $R \parallel P \sim R \parallel Q$ for each CCS process $R$
- $P[f] \sim Q[f]$ for each relabelling function $f$
- $P \mid L \sim Q \mid L$ for each set of labels $L$. 
Other Properties of Strong Bisimilarity

The Following Properties Hold for all CCS Processes $P$, $Q$, $R$

- $P + Q \sim Q + P$
- $P | Q \sim Q | P$
- $P + \text{Nil} \sim P$
- $P | \text{Nil} \sim P$
- $(P + Q) + R \sim P + (Q + R)$
- $(P | Q) | R \sim P | (Q | R)$
Problem with Internal Actions

**Question**
Does $a.\tau.\text{Nil} \sim a.\text{Nil}$ hold? **NO!**

**Problem**
Strong bisimilarity does not abstract away from $\tau$ actions.

**Example: SmUni $\not\sim$ Spec**

- $\text{SmUni} \not\sim \text{Spec}$
- $\text{SmUni} \not\sim \text{Spec}$
- $(CM | CS_1) \setminus \{\text{coin, coffee}\}$
- $(CM_1 | CS_2) \setminus \{\text{coin, coffee}\}$
- $(CM | CS) \setminus \{\text{coin, coffee}\}$

Diagram:

- $\text{SmUni} \not\sim \text{Spec}$
- $(CM | CS_1) \setminus \{\text{coin, coffee}\}$
- $(CM_1 | CS_2) \setminus \{\text{coin, coffee}\}$
- $(CM | CS) \setminus \{\text{coin, coffee}\}$
Problems with Internal Actions

Question
Does \( a.\tau.\text{Nil} \sim a.\text{Nil} \) hold? **NO!**

Problem
Strong bisimilarity does not abstract away from \( \tau \) actions.

Example: SmUni \( \not\sim \) Spec

\[
\begin{align*}
\text{SmUni} & \xrightarrow{pub} \text{Spec} \\
(CM | CS_1) & \xrightarrow{\tau} (CM_1 | CS_2) \\
& \xrightarrow{\tau} (CM | CS) \\
& \setminus \{\text{coin, coffee}\} \\
\end{align*}
\]
Question

Does $a.\tau.Nil \sim a.Nil$ hold? NO!

Problem

Strong bisimilarity does not abstract away from $\tau$ actions.

Example: SmUni $\not\sim$ Spec

$$\begin{align*}
\text{SmUni} & \not\sim \text{Spec} \\
\text{SmUni} & \not\sim \text{Spec} \\
\text{SmUni} & \not\sim \text{Spec}
\end{align*}$$
Questions

Does \( a.\tau.\text{Nil} \sim a.\text{Nil} \) hold? NO!

Problem

Strong bisimilarity does not abstract away from \( \tau \) actions.

Example: SmUni \( \not\sim \) Spec

\[
\begin{align*}
\text{SmUni} & \xrightarrow{\text{pub}} \text{Spec} \\
(CM \mid CS_1) & \setminus \{\text{coin, coffee}\} \xrightarrow{\tau} \\
(CM_1 \mid CS_2) & \setminus \{\text{coin, coffee}\} \xrightarrow{\tau} \\
(CM \mid CS) & \setminus \{\text{coin, coffee}\}
\end{align*}
\]
Let \((\text{Proc}, \text{Act}, \{ \frac{ \rightarrow a }{ a \in \text{Act}} \})\) be an LTS such that \(\tau \in \text{Act}\).

**Definition of Weak Transition Relation**

\[
\frac{\rightarrow a}{a} = \begin{cases} 
(\frac{\rightarrow \tau}{\tau})^* \circ \frac{\rightarrow a}{a} \circ (\frac{\rightarrow \tau}{\tau})^* & \text{if } a \neq \tau \\
(\frac{\rightarrow \tau}{\tau})^* & \text{if } a = \tau
\end{cases}
\]

**What does** \(s \frac{\rightarrow a}{a} t\) **informally mean?**

- If \(a \neq \tau\) then \(s \frac{\rightarrow a}{a} t\) means that from \(s\) we can get to \(t\) by doing zero or more \(\tau\) actions, followed by the action \(a\), followed by zero or more \(\tau\) actions.

- If \(a = \tau\) then \(s \frac{\rightarrow \tau}{\tau} t\) means that from \(s\) we can get to \(t\) by doing zero or more \(\tau\) actions.
Let \((\text{Proc}, \text{Act}, \{ \xrightarrow{a} \mid a \in \text{Act} \})\) be an LTS such that \(\tau \in \text{Act}\).

**Definition of Weak Transition Relation**

\[
\xrightarrow{a} = \begin{cases}
(\xrightarrow{\tau})^* \circ \xrightarrow{a} \circ (\xrightarrow{\tau})^* & \text{if } a \neq \tau \\
(\xrightarrow{\tau})^* & \text{if } a = \tau
\end{cases}
\]

**What does \(s \xrightarrow{a} t\) informally mean?**

- If \(a \neq \tau\) then \(s \xrightarrow{a} t\) means that from \(s\) we can get to \(t\) by doing zero or more \(\tau\) actions, followed by the action \(a\), followed by zero or more \(\tau\) actions.
- If \(a = \tau\) then \(s \xrightarrow{\tau} t\) means that from \(s\) we can get to \(t\) by doing zero or more \(\tau\) actions.
Let \((\text{Proc}, \text{Act}, \{\xrightarrow{a} \mid a \in \text{Act}\})\) be an LTS such that \(\tau \in \text{Act}\).

**Weak Bisimulation**

A binary relation \(R \subseteq \text{Proc} \times \text{Proc}\) is a **weak bisimulation** iff whenever \((s, t) \in R\) then for each \(a \in \text{Act}\) (including \(\tau\)):

- if \(s \xrightarrow{a} s'\) then \(t \xrightarrow{a} t'\) for some \(t'\) such that \((s', t') \in R\)
- if \(t \xrightarrow{a} t'\) then \(s \xrightarrow{a} s'\) for some \(s'\) such that \((s', t') \in R\).

**Weak Bisimilarity**

Two processes \(p_1, p_2 \in \text{Proc}\) are **weakly bisimilar** \((p_1 \approx p_2)\) if and only if there exists a weak bisimulation \(R\) such that \((p_1, p_2) \in R\).

\[\approx = \bigcup \{R \mid R \text{ is a weak bisimulation}\}\]
Let \((\text{Proc}, \text{Act}, \{\xrightarrow{a}| a \in \text{Act}\})\) be an LTS such that \(\tau \in \text{Act}\).

**Weak Bisimulation**

A binary relation \(R \subseteq \text{Proc} \times \text{Proc}\) is a weak bisimulation iff whenever \((s, t) \in R\) then for each \(a \in \text{Act}\) (including \(\tau\)):

- if \(s \xrightarrow{a} s'\) then \(t \xrightarrow{a} t'\) for some \(t'\) such that \((s', t') \in R\)
- if \(t \xrightarrow{a} t'\) then \(s \xrightarrow{a} s'\) for some \(s'\) such that \((s', t') \in R\).

**Weak Bisimilarity**

Two processes \(p_1, p_2 \in \text{Proc}\) are weakly bisimilar \((p_1 \approx p_2)\) if and only if there exists a weak bisimulation \(R\) such that \((p_1, p_2) \in R\).

\[
\approx = \bigcup \{R \mid R \text{ is a weak bisimulation}\}
\]
**Weak Bisimulation Game**

**Definition**

All the same except that

- **defender** can now answer using \( \overset{a}{\rightarrow} \) moves.

The attacker is still using only \( \overset{a}{\rightarrow} \) moves.

**Theorem**

- States \( s \) and \( t \) are weakly bisimilar if and only if the **defender** has a *universal* winning strategy starting from the configuration \((s, t)\).

- States \( s \) and \( t \) are not weakly bisimilar if and only if the **attacker** has a *universal* winning strategy starting from the configuration \((s, t)\).
Weak Bisimulation Game

**Definition**

All the same except that

- **defender** can now answer using $\xrightarrow{a}$ moves.

The attacker is still using only $\rightarrow$ moves.

**Theorem**

- States $s$ and $t$ are weakly bisimilar if and only if the defender has a **universal** winning strategy starting from the configuration $(s, t)$.

- States $s$ and $t$ are not weakly bisimilar if and only if the attacker has a **universal** winning strategy starting from the configuration $(s, t)$.
Weak Bisimilarity – Properties

Properties of $\approx$

- an equivalence relation
- the largest weak bisimulation
- validates lots of natural laws, e.g.
  - $a.\tau.P \approx a.P$
  - $P + \tau.P \approx \tau.P$
  - $a.(P + \tau.Q) \approx a.(P + \tau.Q) + a.Q$
  - $P + Q \approx Q + P$
  - $P|Q \approx Q|P$
  - $P + Nil \approx P$

- strong bisimilarity is included in weak bisimilarity ($\sim \subseteq \approx$)
- abstracts from $\tau$ loops
Case Study: Communication Protocol

\[
\begin{align*}
\text{Send} & \quad \text{def} = \text{acc.Send} \\
\text{Sending} & \quad \text{def} = \text{send.Wait} \\
\text{Wait} & \quad \text{def} = \text{ack.Send + error.Sending} \\
\text{Rec} & \quad \text{def} = \text{trans.Del} \\
\text{Del} & \quad \text{def} = \text{del.Ack} \\
\text{Ack} & \quad \text{def} = \text{ack.Rec} \\
\text{Med} & \quad \text{def} = \text{send.Med}' \\
\text{Med}' & \quad \text{def} = \tau.\text{Err} + \text{trans.Med} \\
\text{Err} & \quad \text{def} = \text{error.Med}
\end{align*}
\]
Case Study: Communication Protocol

\[\text{Send} \overset{\text{def}}{=} \text{acc}.\text{Sending} \quad \text{Rec} \overset{\text{def}}{=} \text{trans}.\text{Del}\]

\[\text{Sending} \overset{\text{def}}{=} \text{send}.\text{Wait} \quad \text{Del} \overset{\text{def}}{=} \text{del}.\text{Ack}\]

\[\text{Wait} \overset{\text{def}}{=} \text{ack}.\text{Send} + \text{error}.\text{Sending} \quad \text{Ack} \overset{\text{def}}{=} \text{ack}.\text{Rec}\]

\[\text{Med} \overset{\text{def}}{=} \text{send}.\text{Med}’ \quad \text{Med}’ \overset{\text{def}}{=} \tau.\text{Err} + \text{trans}.\text{Med}\]

\[\text{Err} \overset{\text{def}}{=} \text{error}.\text{Med}\]
Verification Question

\[ \text{Impl} \overset{\text{def}}{=} (\text{Send} \mid \text{Med} \mid \text{Rec}) \setminus \{\text{send, trans, ack, error}\} \]

\[ \text{Spec} \overset{\text{def}}{=} \text{acc.del.Spec} \]

Question

1. Draw the LTS of Impl and Spec and prove (by hand) the equivalence.
2. Use Concurrency WorkBench (CWB).
\[ \text{Impl} \overset{\text{def}}{=} (\text{Send} \mid \text{Med} \mid \text{Rec}) \setminus \{\text{send}, \text{trans}, \text{ack}, \text{error}\} \]

\[ \text{Spec} \overset{\text{def}}{=} \text{acc.del.Spec} \]

\text{Question}

\begin{itemize}
  \item Draw the LTS of Impl and Spec and prove (by hand) the equivalence.
  \item Use Concurrency WorkBench (CWB).
\end{itemize}
Impl $\overset{\text{def}}{=} (\text{Send} \mid \text{Med} \mid \text{Rec}) \setminus \{\text{send}, \text{trans}, \text{ack}, \text{error}\}$

Spec $\overset{\text{def}}{=} \text{acc}.\text{del}.\text{Spec}$

**Question**

1. Draw the LTS of Impl and Spec and prove (by hand) the equivalence.
2. Use Concurrency WorkBench (CWB).
**Verification Question**

\[
\text{Impl} \overset{\text{def}}{=} (\text{Send} \mid \text{Med} \mid \text{Rec}) \setminus \{\text{send, trans, ack, error}\}
\]

\[
\text{Spec} \overset{\text{def}}{=} \text{acc.del.Spec}
\]

**Question**

1. Draw the LTS of Impl and Spec and prove (by hand) the equivalence.
2. Use Concurrency WorkBench (CWB).
Verification Question

\[
\text{Impl} \overset{\text{def}}{=} (\text{Send} \mid \text{Med} \mid \text{Rec}) \setminus \{\text{send}, \text{trans}, \text{ack}, \text{error}\}
\]

\[
\text{Spec} \overset{\text{def}}{=} \text{acc.del.Spec}
\]

Question

\[
\text{Impl} \ ? \ \approx \ \text{Spec}
\]

1. Draw the LTS of Impl and Spec and prove (by hand) the equivalence.

2. Use Concurrency WorkBench (CWB).
### CCS Definitions

<table>
<thead>
<tr>
<th>Definition</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Med</td>
<td>send.Med'</td>
</tr>
<tr>
<td>Err</td>
<td>error.Med</td>
</tr>
<tr>
<td>Spec</td>
<td>acc.del.Spec</td>
</tr>
</tbody>
</table>

### CWB Program (protocol.cwb)

```plaintext
agent Med = send.Med';
agent Med' = (tau.Err + 'trans.Med);
agent Err = 'error.Med;

set L = {send, trans, ack, error};
agent Impl = (Send | Med | Rec) \ L;
agent Spec = acc.'del.Spec;
```
CWB Session

[luca@vel5638 CWB]$ ./xccscwb.x86-linux

> help;

> input "protocol.cwb";

> vs(5,Impl);

> sim(Spec);

> eq(Spec,Impl);  ** weak bisimilarity **

> strongeq(Spec,Impl);  ** strong bisimilarity **
Is Weak Bisimilarity a Congruence for CCS?

**Theorem**

Let $P$ and $Q$ be CCS processes such that $P \approx Q$. Then

- $\alpha.P \approx \alpha.Q$ for each action $\alpha \in \text{Act}$
- $P | R \approx Q | R$ and $R | P \approx R | Q$ for each CCS process $R$
- $P[f] \approx Q[f]$ for each relabelling function $f$
- $P \setminus L \approx Q \setminus L$ for each set of labels $L$.

What about choice?

$\tau.a.\text{Nil} \approx a.\text{Nil}$ but $\tau.a.\text{Nil} + b.\text{Nil} \not\approx a.\text{Nil} + b.\text{Nil}$

**Conclusion**

Weak bisimilarity is **not** a congruence for CCS.
Is Weak Bisimilarity a Congruence for CCS?

**Theorem**

Let $P$ and $Q$ be CCS processes such that $P \approx Q$. Then

- $\alpha.P \approx \alpha.Q$ for each action $\alpha \in \text{Act}$
- $P | R \approx Q | R$ and $R | P \approx R | Q$ for each CCS process $R$
- $P[f] \approx Q[f]$ for each relabelling function $f$
- $P \setminus L \approx Q \setminus L$ for each set of labels $L$.

What about choice?

$\tau.a.Nil \approx a.Nil$  but  $\tau.a.Nil + b.Nil \not\approx a.Nil + b.Nil$

**Conclusion**

Weak bisimilarity is **not** a congruence for CCS.
Is Weak Bisimilarity a Congruence for CCS?

**Theorem**

Let $P$ and $Q$ be CCS processes such that $P \approx Q$. Then

- $\alpha.P \approx \alpha.Q$ for each action $\alpha \in \text{Act}$
- $P | R \approx Q | R$ and $R | P \approx R | Q$ for each CCS process $R$
- $P[f] \approx Q[f]$ for each relabelling function $f$
- $P \setminus L \approx Q \setminus L$ for each set of labels $L$.

What about choice?

$\tau.a.Nil \approx a.Nil$ but $\tau.a.Nil + b.Nil \not\approx a.Nil + b.Nil$

**Conclusion**

Weak bisimilarity is **not** a congruence for CCS.