Models and Semantics of Computation

Jorge A. Pérez
Bernoulli Institute
University of Groningen

October 9, 2023

CCS: A Calculus of Communicating Systems (II)

- value passing CCS
- translations vs encodings
- strong and weak bisimilarity: definition, games, properties

Acknowledgment

This set of slides was originally produced by Jiri Srba, and makes part of the course material for the book

Reactive Systems: Modelling, Specification and Verification
by L. Aceto, A. Ingolfsdottir, K. G. Larsen and J. Srba
URL: http://rsbook.cs.aau.dk

I have adapted them slightly for the purposes of this course.

Value Passing CCS

Main Idea

Handshake synchronization is extended with the possibility to exchange integer values.

$$\overline{pay(6)}.Nil \parallel pay(x).\overline{save(x/2)}.Nil$$

Value Passing CCS

Main Idea

Handshake synchronization is extended with the possibility to exchange integer values.

$$\overline{pay(6)}.Nil \;\|\; pay(x).\overline{save(x/2)}.Nil$$

$$\downarrow \tau$$

$$Nil \;\|\; \overline{save(3)}.Nil$$

Value Passing CCS

Main Idea

Handshake synchronization is extended with the possibility to exchange integer values.

$$\overline{pay(6)}.Nil \parallel pay(x).\overline{save(x/2)}.Nil$$

$$\downarrow \tau$$

$$Nil \parallel \overline{save(3)}.Nil$$

Parametrized Process Constants

For example: $Bank(total) \stackrel{\text{def}}{=} save(x).Bank(total + x)$.

Value Passing CCS

Main Idea

Handshake synchronization is extended with the possibility to exchange integer values.

$$\overline{pay(6)}.Nil \parallel pay(x).\overline{save(x/2)}.Nil \parallel Bank(100)$$

$$\downarrow \tau$$

$$Nil \parallel \overline{save(3)}.Nil \parallel Bank(100)$$

Parametrized Process Constants

For example: $Bank(total) \stackrel{\text{def}}{=} save(x).Bank(total + x)$.

Value Passing CCS

Main Idea

Handshake synchronization is extended with the possibility to exchange integer values.

$$\overline{pay(6)}.Nil \parallel pay(x).\overline{save(x/2)}.Nil \parallel Bank(100)$$

$$\downarrow \tau$$

$$Nil \parallel \overline{save(3)}.Nil \parallel Bank(100)$$

$$\downarrow \tau$$

$$Nil \parallel Nil \parallel Bank(103)$$

Parametrized Process Constants

For example: $Bank(total) \stackrel{\text{def}}{=} save(x).Bank(total + x)$.

Translation of Value Passing CCS to Standard CCS

Value Passing CCS (CCS^v)

$$C \stackrel{\text{def}}{=} in(x).C'(x)$$

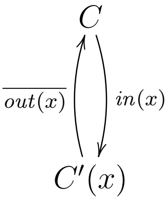
$$C'(x) \stackrel{\text{def}}{=} \overline{out(x)}.C$$

$$\longrightarrow$$

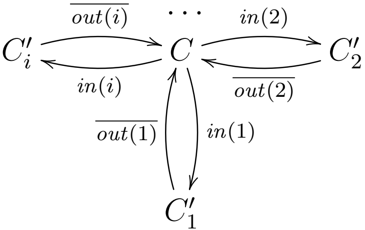
Standard CCS (CCS)

$$C \stackrel{\text{def}}{=} \sum_{i \in \mathbb{N}} in(i).C'_i$$

$$C'_i \stackrel{\text{def}}{=} \overline{out(i)}.C$$



symbolic LTS



infinite LTS

Language Translations and Encodings

- We would like to express the previous intuitions about the translation in a formal manner.
- We will rely on the concepts of translations and encodings. In a way, an encoding is a "compiler" between two different process calculi

[For now, we will rely on some basic notions.
Later on the course we will be more precise about translations.]

Language Translations and Encodings

- We would like to express the previous intuitions about the translation in a formal manner.
- We will rely on the concepts of translations and encodings. In a way, an encoding is a "compiler" between two different process calculi

[For now, we will rely on some basic notions.
Later on the course we will be more precise about translations.]

Language Translations and Encodings

Assume a source language $\mathcal{P}_{\mathsf{s}}$, and a target language $\mathcal{P}_{\mathsf{t}}$

A translation is a function $[\![\cdot]\!] : \mathcal{P}_{\mathsf{s}} \to \mathcal{P}_{\mathsf{t}}$

To be meaningful, translations should satisfy some criteria

- syntactic criteria ensuring, e.g., that the translation enjoys some compositionality principles
  Ex: homormorphism wrt parallel, i.e., $[\![P \parallel Q]\!] = [\![P]\!] \parallel [\![Q]\!]$
- semantic criteria ensuring, e.g., that the behavior of the translated target term is related to the behavior of the corresponding source term
  Ex: operational correspondence, i.e.,
  $P \xrightarrow{\alpha} P'$ implies $[\![P]\!] \xrightarrow{\tau}\xrightarrow{\alpha}\xrightarrow{\tau} [\![P']\!]$

Language Translations and Encodings

Assume a source language $\mathcal{P}_{\mathsf{s}}$, and a target language $\mathcal{P}_{\mathsf{t}}$

A translation is a function $[\![\cdot]\!] : \mathcal{P}_{\mathsf{s}} \to \mathcal{P}_{\mathsf{t}}$

To be meaningful, translations should satisfy some criteria

- syntactic criteria ensuring, e.g., that the translation enjoys some compositionality principles
  Ex: homormorphism wrt parallel, i.e., $[\![P \parallel Q]\!] = [\![P]\!] \parallel [\![Q]\!]$
- semantic criteria ensuring, e.g., that the behavior of the translated target term is related to the behavior of the corresponding source term
  Ex: operational correspondence, i.e.,
  $P \xrightarrow{\alpha} P'$ implies $[\![P]\!] \xrightarrow{\tau}\xrightarrow{\alpha}\xrightarrow{\tau} [\![P']\!]$

Language Translations and Encodings

Assume a source language $\mathcal{P}_{\mathsf{s}}$, and a target language $\mathcal{P}_{\mathsf{t}}$

A translation is a function $[\![\cdot]\!] : \mathcal{P}_{\mathsf{s}} \to \mathcal{P}_{\mathsf{t}}$

To be meaningful, translations should satisfy some criteria

- syntactic criteria ensuring, e.g., that the translation enjoys some compositionality principles
  Ex: homormorphism wrt parallel, i.e., $[\![P \parallel Q]\!] = [\![P]\!] \parallel [\![Q]\!]$
- semantic criteria ensuring, e.g., that the behavior of the translated target term is related to the behavior of the corresponding source term
  Ex: operational correspondence, i.e.,
  $P \xrightarrow{\alpha} P'$ implies $[\![P]\!] \xrightarrow{\tau}\xrightarrow{\alpha}\xrightarrow{\tau} [\![P']\!]$

Language Translations and Encodings

Assume a source language $\mathcal{P}_{\mathsf{s}}$, and a target language $\mathcal{P}_{\mathsf{t}}$

A translation is a function $[\![\cdot]\!] : \mathcal{P}_{\mathsf{s}} \to \mathcal{P}_{\mathsf{t}}$

To be meaningful, translations should satisfy some criteria

- syntactic criteria ensuring, e.g., that the translation enjoys some compositionality principles
  Ex: homormorphism wrt parallel, i.e., $[\![P \parallel Q]\!] = [\![P]\!] \parallel [\![Q]\!]$
- semantic criteria ensuring, e.g., that the behavior of the translated target term is related to the behavior of the corresponding source term
  Ex: operational correspondence, i.e.,
  $P \xrightarrow{\alpha} P'$ implies $[\![P]\!] \xrightarrow{\tau}\xrightarrow{\alpha}\xrightarrow{\tau} [\![P']\!]$

Language Translations and Encodings

Assume a source language $\mathcal{P}_{\mathsf{s}}$, and a target language $\mathcal{P}_{\mathsf{t}}$

Translation

A function $[\![\cdot]\!] : \mathcal{P}_{\mathsf{s}} \rightarrow \mathcal{P}_{\mathsf{t}}$

Encoding

A translation enjoying some syntactic and semantic criteria

Translation of Value Passing CCS to Standard CCS

Let V be a set of values.

Intuitively, each label l in CCS^v is a set $\{l_v \mid v \in V\}$ in CCS.

The translation $[\![\cdot]\!]_{\mathcal{V}} : \mathrm{CCS}^v \to \mathrm{CCS}$ is defined inductively as:

$$\begin{array}{rcl}
[\![a(x).P]\!]_{\mathcal{V}} & = & \sum_{v \in V} a_v.[\![P\{v/x\}]\!]_{\mathcal{V}} \\
[\![\overline{a}(e).P]\!]_{\mathcal{V}} & = & \overline{a}_e.[\![P]\!]_{\mathcal{V}} \\
[\![P_1 \parallel P_2]\!]_{\mathcal{V}} & = & [\![P_1]\!]_{\mathcal{V}} \parallel [\![P_2]\!]_{\mathcal{V}} \\
[\![\sum_{i \in I} P_i]\!]_{\mathcal{V}} & = & \sum_{i \in I} [\![P_i]\!]_{\mathcal{V}} \\
[\![(\nu \tilde{a})P]\!]_{\mathcal{V}} & = & (\nu \tilde{b})[\![P]\!]_{\mathcal{V}} \quad \text{with } \tilde{b} = \{a_v \mid a \in \tilde{b}, v \in V\} \\
[\![A\langle e_1, \ldots, e_n \rangle]\!]_{\mathcal{V}} & = & A_{e_1, \ldots, e_n}
\end{array}$$

Also, the definition $A(\tilde{x}) \stackrel{\text{def}}{=} P$, with $\tilde{x} = x_1, \ldots, x_n$, is translated into a set of definitions $\{A_{\tilde{v}} \stackrel{\text{def}}{=} [\![P\{\tilde{v}/\tilde{x}\}]\!]_{\mathcal{V}} \mid \tilde{v} \in V^n\}$

Translation of Value Passing CCS to Standard CCS

Let V be a set of values.
Intuitively, each label l in CCS^v is a set $\{l_v \mid v \in V\}$ in CCS.
The translation $[\![\cdot]\!]_{\mathcal{V}} : \mathrm{CCS}^v \to \mathrm{CCS}$ is defined inductively as:

$$
\begin{aligned}
[\![a(x).P]\!]_{\mathcal{V}} &= \sum_{v \in V} a_v.[\![P\{^v/_x\}]\!]_{\mathcal{V}} \\
[\![\overline{a}(e).P]\!]_{\mathcal{V}} &= \overline{a}_e.[\![P]\!]_{\mathcal{V}} \\
[\![P_1 \parallel P_2]\!]_{\mathcal{V}} &= [\![P_1]\!]_{\mathcal{V}} \parallel [\![P_2]\!]_{\mathcal{V}} \\
[\![\sum_{i \in I} P_i]\!]_{\mathcal{V}} &= \sum_{i \in I} [\![P_i]\!]_{\mathcal{V}} \\
[\![(\nu \tilde{a})P]\!]_{\mathcal{V}} &= (\nu \tilde{b})[\![P]\!]_{\mathcal{V}} \quad \text{with } \tilde{b} = \{a_v \mid a \in \tilde{b}, v \in V\} \\
[\![A\langle e_1, \ldots, e_n\rangle]\!]_{\mathcal{V}} &= A_{e_1, \ldots, e_n}
\end{aligned}
$$

Also, the definition $A(\tilde{x}) \stackrel{\mathrm{def}}{=} P$, with $\tilde{x} = x_1, \ldots, x_n$, is translated into a set of definitions $\{A_{\tilde{v}} \stackrel{\mathrm{def}}{=} [\![P\{^{\tilde{v}}/_{\tilde{x}}\}]\!]_{\mathcal{V}} \mid \tilde{v} \in V^n\}$

Translation of Value Passing CCS to Standard CCS

Let V be a set of values.

Intuitively, each label l in CCS^v is a set $\{l_v \mid v \in V\}$ in CCS.

The translation $[\![\cdot]\!]_{\mathcal{V}} : \mathrm{CCS}^v \to \mathrm{CCS}$ is defined inductively as:

$$
\begin{array}{rcl}
[\![a(x).P]\!]_{\mathcal{V}} & = & \sum_{v\in V} a_v.[\![P\{^v/_x\}]\!]_{\mathcal{V}} \\
[\![\overline{a}(e).P]\!]_{\mathcal{V}} & = & \overline{a}_e.[\![P]\!]_{\mathcal{V}} \\
[\![P_1 \parallel P_2]\!]_{\mathcal{V}} & = & [\![P_1]\!]_{\mathcal{V}} \parallel [\![P_2]\!]_{\mathcal{V}} \\
[\![\sum_{i\in I} P_i]\!]_{\mathcal{V}} & = & \sum_{i\in I} [\![P_i]\!]_{\mathcal{V}} \\
[\![(\nu\tilde{a})P]\!]_{\mathcal{V}} & = & (\nu\tilde{b})[\![P]\!]_{\mathcal{V}} \quad \text{with } \tilde{b} = \{a_v \mid a \in \tilde{b}, v \in V\} \\
[\![A\langle e_1, \ldots, e_n\rangle]\!]_{\mathcal{V}} & = & A_{e_1,\ldots,e_n}
\end{array}
$$

Also, the definition $A(\tilde{x}) \stackrel{\mathrm{def}}{=} P$, with $\tilde{x} = x_1, \ldots, x_n$, is translated into a set of definitions $\{A_{\tilde{v}} \stackrel{\mathrm{def}}{=} [\![P\{^{\tilde{v}}/_{\tilde{x}}\}]\!]_{\mathcal{V}} \mid \tilde{v} \in V^n\}$

Translation of Value Passing CCS to Standard CCS

- The encoding shows that CCS^v can be safely used in examples; CCS is enough for theoretical purposes
- The encoding is possible because we have (guarded) choice on possibly infinite sums
- Recall the translations for parallel composition and sum:

$$\begin{aligned} [\![P_1 \parallel P_2]\!]_{\mathcal{V}} &= [\![P_1]\!]_{\mathcal{V}} \parallel [\![P_2]\!]_{\mathcal{V}} \\ [\![\sum_{i\in I} P_i]\!]_{\mathcal{V}} &= \sum_{i\in I} [\![P_i]\!]_{\mathcal{V}} \end{aligned}$$

  This kind of translations are often called homomorphic, as they simply follow the structural meaning of the operator.
- For simplicity, operators translated homomorphically are often omitted in definitions

Translation of Value Passing CCS to Standard CCS

- The encoding shows that CCS^v can be safely used in examples; CCS is enough for theoretical purposes
- The encoding is possible because we have (guarded) choice on possibly infinite sums
- Recall the translations for parallel composition and sum:

$$
\begin{aligned}
[\![P_1 \parallel P_2]\!]_{\mathcal{V}} &= [\![P_1]\!]_{\mathcal{V}} \parallel [\![P_2]\!]_{\mathcal{V}} \\
[\![\sum_{i \in I} P_i]\!]_{\mathcal{V}} &= \sum_{i \in I} [\![P_i]\!]_{\mathcal{V}}
\end{aligned}
$$

  This kind of translations are often called homomorphic, as they simply follow the structural meaning of the operator.
- For simplicity, operators translated homomorphically are often omitted in definitions

CCS is Turing Complete

Another expressiveness result:

Fact

CCS can simulate a computation of any Turing machine.

Question: how would you show this?

Remark

Hence CCS is as expressive as any other programming language but its use is to rather describe the behaviour of reactive systems than to perform specific calculations.

CCS is Turing Complete

Another expressiveness result:

Fact

CCS can simulate a computation of any Turing machine.

Question: how would you show this?

Remark

Hence CCS is as expressive as any other programming language but its use is to rather describe the behaviour of reactive systems than to perform specific calculations.

Behavioural Equivalence

Implementation

$$CM \stackrel{\text{def}}{=} coin.\overline{coffee}.CM$$

$$CS \stackrel{\text{def}}{=} \overline{pub}.\overline{coin}.coffee.CS$$

$$Uni \stackrel{\text{def}}{=} (\nu\, coin, coffee)(CM \parallel CS)$$

Specification

$$Spec \stackrel{\text{def}}{=} \overline{pub}.Spec$$

Question

Are the processes Uni and $Spec$ behaviorally equivalent?

$$Uni \equiv Spec$$

where $\equiv$ is a binary relation on processes.

Behavioural Equivalence

Implementation

$$CM \stackrel{\text{def}}{=} coin.\overline{coffee}.CM$$

$$CS \stackrel{\text{def}}{=} \overline{pub}.\overline{coin}.coffee.CS$$

$$Uni \stackrel{\text{def}}{=} (\nu coin, coffee)(CM \parallel CS)$$

Specification

$$Spec \stackrel{\text{def}}{=} \overline{pub}.Spec$$

Question

Are the processes Uni and $Spec$ behaviorally equivalent?

$$Uni \equiv Spec$$

where $\equiv$ is a binary relation on processes.

Behavioural Equivalence

Implementation

$$CM \stackrel{\text{def}}{=} coin.\overline{coffee}.CM$$

$$CS \stackrel{\text{def}}{=} \overline{pub}.\overline{coin}.coffee.CS$$

$$Uni \stackrel{\text{def}}{=} (\nu coin, coffee)(CM \parallel CS)$$

Specification

$$Spec \stackrel{\text{def}}{=} \overline{pub}.Spec$$

Question

Are the processes Uni and $Spec$ behaviorally equivalent?

$$Uni \equiv Spec$$

where $\equiv$ is a binary relation on processes.

Goals

What should a reasonable behavioral equivalence satisfy?

- abstract from states (consider only the behavior – actions)
- abstract from nondeterminism
- abstract from internal behavior

What else?

- reflexivity $P \equiv P$ for any process P
- transitivity $Spec_0 \equiv Spec_1 \equiv Spec_2 \equiv \cdots \equiv Impl$ gives that
$$Spec_0 \equiv Impl$$
- symmetry $P \equiv Q$ iff $Q \equiv P$
- congruence

Goals

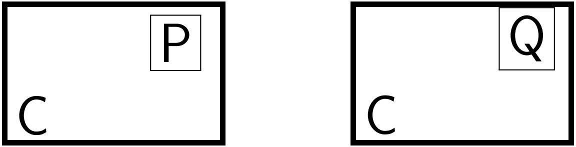
What should a reasonable behavioral equivalence satisfy?

- abstract from states (consider only the behavior – actions)
- abstract from nondeterminism
- abstract from internal behavior

What else?

- reflexivity $P \equiv P$ for any process P
- transitivity $\mathit{Spec}_0 \equiv \mathit{Spec}_1 \equiv \mathit{Spec}_2 \equiv \cdots \equiv \mathit{Impl}$ gives that
$$\mathit{Spec}_0 \equiv \mathit{Impl}$$
- symmetry $P \equiv Q$ iff $Q \equiv P$
- congruence

Congruence



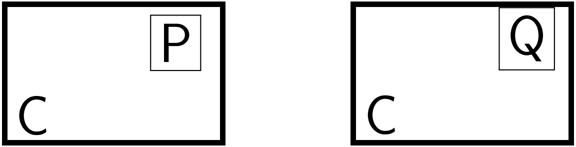
$C(P)$ $\qquad\qquad$ $C(Q)$

- We would like "equal" processes P and Q to "behave the same" under any context $C(\cdot)$.
- A context is a process with a hole. When the hole is filled in with a process P, we obtain another process (noted $C(P)$ or $C[P]$).

Congruence Property

$$P \equiv Q \text{ implies } C(P) \equiv C(Q)$$

Congruence



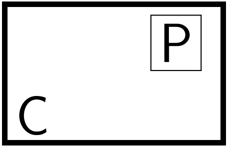
$C(P)$ $\qquad\qquad$ $C(Q)$

- We would like "equal" processes P and Q to "behave the same" under any context $C(\cdot)$.
- A context is a process with a hole. When the hole is filled in with a process P, we obtain another process (noted $C(P)$ or $C[P]$).

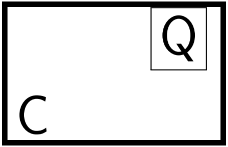
Congruence Property

$$P \equiv Q \text{ implies } C(P) \equiv C(Q)$$

Congruence



$C(P)$



$C(Q)$

- We would like "equal" processes P and Q to "behave the same" under any context $C(\cdot)$.
- A context is a process with a hole. When the hole is filled in with a process P, we obtain another process (noted $C(P)$ or $C[P]$).

Congruence Property

$$P \equiv Q \text{ implies } C(P) \equiv C(Q)$$

Ideally

- Ideally, we would like to identify two processes unless there is some sequence of 'interactions' than an 'observer' may have with them leading to different 'outcomes'
- There are many different choices — many notions of behavioral equivalences.

We will explore two of them:

- Trace Equivalence
- Strong Bisimilarity

Trace Equivalence

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

Trace Set for $s \in Proc$

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

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

$$Traces(s) = Traces(t)$$

Trace Equivalence

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

Trace Set for $s \in Proc$

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

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

$$Traces(s) = Traces(t)$$

Trace Equivalence: Example

$$
\begin{tikzcd}
& s \arrow[d, "a"] & \\
& s_1 \arrow[dl, "b"'] \arrow[dr, "c"] & \\
s_2 & & s_3
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
& t \arrow[dl, "a"'] \arrow[dr, "a"] & \\
t_1 \arrow[d, "b"'] & & t_2 \arrow[d, "c"] \\
t_3 & & t_4
\end{tikzcd}
$$

We have $s \equiv_t t$ because

$$Traces(s) = Traces(t) = \{ab, ac\}$$

Black-Box Experiments

Main Idea

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

Strong Bisimilarity

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

Strong Bisimulation

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

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

Strong Bisimilarity

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

$$\sim = \bigcup \{\mathcal{R} \mid \mathcal{R} \text{ is a strong bisimulation}\}$$

Strong Bisimilarity

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

Strong Bisimulation

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

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

Strong Bisimilarity

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

$$\sim \; = \bigcup \{\mathcal{R} \mid \mathcal{R} \text{ is a strong bisimulation}\}$$

Basic Properties of Strong Bisimilarity

Theorem

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

Theorem

$\sim$ is the largest strong bisimulation

Theorem

$s \sim t$ if and only if for each $a \in Act$:

- *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 (reflexive, symmetric and transitive)*

Theorem

$\sim$ *is the largest strong bisimulation*

Theorem

$s \sim t$ *if and only if for each* $a \in Act$:

- *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 (reflexive, symmetric and transitive)*

Theorem

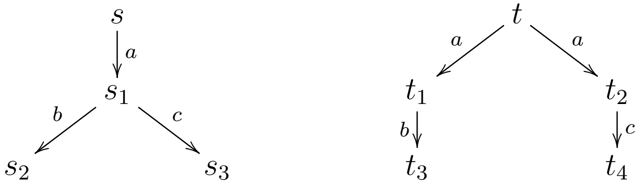
$\sim$ *is the largest strong bisimulation*

Theorem

$s \sim t$ *if and only if for each* $a \in Act$*:*

- *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* $s' \sim t'$.

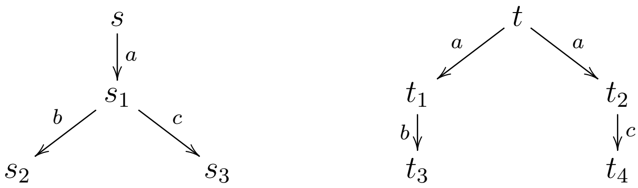
How to Show Nonbisimilarity?



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 will enable to disqualify many bisimulation candidates in one step.
- Use 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 will enable to disqualify many bisimulation candidates in one step.
- Use game characterization of strong bisimilarity.

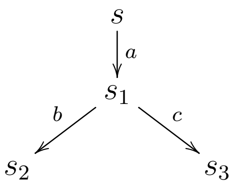
How to Show Nonbisimilarity?

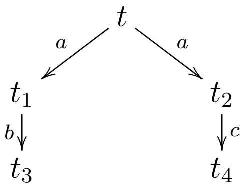
$$\begin{tikzcd}
& s \arrow[d, "a"] & \\
& s_1 \arrow[dl, "b"'] \arrow[dr, "c"] & \\
s_2 & & s_3
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
& t \arrow[dl, "a"'] \arrow[dr, "a"] & \\
t_1 \arrow[d, "b"'] & & t_2 \arrow[d, "c"] \\
t_3 & & t_4
\end{tikzcd}$$

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 will enable to disqualify many bisimulation candidates in one step.
- Use 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 will enable to disqualify many bisimulation candidates in one step.
- Use game characterization of strong bisimilarity.

Strong Bisimulation Game

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

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 $Proc \times Proc$.
- In every round exactly one configuration is called current. Initially the configuration (s, t) is the current one.

Intuition

The defender wants the show that s and t are strongly bisimilar while the attacker aims to prove the opposite.

Strong Bisimulation Game

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

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 $Proc \times Proc$.
- In every round exactly one configuration is called current. Initially the configuration (s, t) is the current one.

Intuition

The defender wants the show that s and t are strongly bisimilar while the attacker aims to prove 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 $\xrightarrow{a}$-move for some $a \in Act$, 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.

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 $a \in Act$, 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

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

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

Strong Bisimilarity is a Congruence for CCS Operations

Theorem

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

- $\alpha.P \sim \alpha.Q$ *for each action* $\alpha \in Act$
- $P + R \sim Q + R$ *and* $R + P \sim R + Q$ *for each CCS process* R
- $P \mid R \sim Q \mid R$ *and* $R \mid P \sim R \mid Q$ *for each CCS process* R
- $(\nu \mathsf{a})\, P \sim (\nu \mathsf{a})\, Q$ *for any* a.

Other Properties of Strong Bisimilarity

Following Properties Hold for any CCS Processes P, Q and R

- $P + Q \sim Q + P$
- $P \parallel Q \sim Q \parallel P$
- $P + Nil \sim P$
- $P \parallel Nil \sim P$
- $(P + Q) + R \sim P + (Q + R)$
- $(P \parallel Q) \parallel R \sim P \parallel (Q \parallel R)$

Example – Buffer

Buffer of Capacity 1

$$B_0^1 \stackrel{\text{def}}{=} in.B_1^1$$
$$B_1^1 \stackrel{\text{def}}{=} \overline{out}.B_0^1$$

Buffer of Capacity n

$$B_0^n \stackrel{\text{def}}{=} in.B_1^n$$
$$B_i^n \stackrel{\text{def}}{=} in.B_{i+1}^n + \overline{out}.B_{i-1}^n \quad \text{for } 0 < i < n$$
$$B_n^n \stackrel{\text{def}}{=} \overline{out}.B_{n-1}^n$$

Example: $B_0^2 \sim B_0^1 \parallel B_0^1$

Example – Buffer

Buffer of Capacity 1

$$B^1_0 \stackrel{\text{def}}{=} in.B^1_1$$
$$B^1_1 \stackrel{\text{def}}{=} \overline{out}.B^1_0$$

Buffer of Capacity n

$$B^n_0 \stackrel{\text{def}}{=} in.B^n_1$$
$$B^n_i \stackrel{\text{def}}{=} in.B^n_{i+1} + \overline{out}.B^n_{i-1} \quad \text{for } 0 < i < n$$
$$B^n_n \stackrel{\text{def}}{=} \overline{out}.B^n_{n-1}$$

Example: $B^2_0 \sim B^1_0 \parallel B^1_0$

Example – Buffer

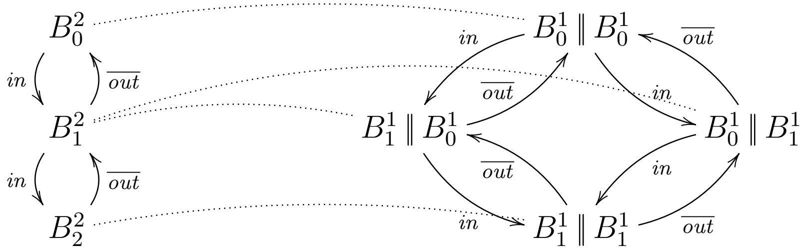
Buffer of Capacity 1

$$B_0^1 \stackrel{\text{def}}{=} in.B_1^1$$
$$B_1^1 \stackrel{\text{def}}{=} \overline{out}.B_0^1$$

Buffer of Capacity n

$$B_0^n \stackrel{\text{def}}{=} in.B_1^n$$
$$B_i^n \stackrel{\text{def}}{=} in.B_{i+1}^n + \overline{out}.B_{i-1}^n \quad \text{for } 0 < i < n$$
$$B_n^n \stackrel{\text{def}}{=} \overline{out}.B_{n-1}^n$$

Example: $B_0^2 \sim B_0^1 \parallel B_0^1$



Example – Buffer

Theorem

For all natural numbers n: $$B_0^n \sim \underbrace{B_0^1 \parallel B_0^1 \parallel \cdots \parallel B_0^1}_{n \text{ times}}$$

Proof.

The co-inductive proof method: to prove bisimilar processes, show an appropriate strong bisimulation that contains them. Construct the following binary relation where $i_1, i_2, \ldots, i_n \in \{0, 1\}$.

$$\mathcal{R} = \{\big(B_i^n,\ B_{i_1}^1 \parallel B_{i_2}^1 \parallel \cdots \parallel B_{i_n}^1\big) \mid \sum_{j=1}^{n} i_j = i\}$$

- $\big(B_0^n,\ B_0^1 \parallel B_0^1 \parallel \cdots \parallel B_0^1\big) \in R$
- $\mathcal{R}$ is strong bisimulation

Example – Buffer

Theorem

For all natural numbers n: $$B_0^n \sim \underbrace{B_0^1 \parallel B_0^1 \parallel \cdots \parallel B_0^1}_{n \text{ times}}$$

Proof.

The co-inductive proof method: to prove bisimilar processes, show an appropriate strong bisimulation that contains them. Construct the following binary relation where $i_1, i_2, \ldots, i_n \in \{0, 1\}$.

$$\mathcal{R} = \{\big(B_i^n,\ B_{i_1}^1 \parallel B_{i_2}^1 \parallel \cdots \parallel B_{i_n}^1\big) \mid \sum_{j=1}^{n} i_j = i\}$$

- $\big(B_0^n,\ B_0^1 \parallel B_0^1 \parallel \cdots \parallel B_0^1\big) \in R$
- $\mathcal{R}$ is strong bisimulation

Strong Bisimilarity – Summary

Properties of $\sim$

- an equivalence relation
- the largest strong bisimulation
- a congruence
- enough to prove some natural rules like
  - $P \parallel Q \sim Q \parallel P$
  - $P \parallel Nil \sim P$
  - $(P \parallel Q) \parallel R \sim Q \parallel (P \parallel R)$
  - $\ldots$

Question

Should we look any further???

Strong Bisimilarity – Summary

Properties of $\sim$

- an equivalence relation
- the largest strong bisimulation
- a congruence
- enough to prove some natural rules like
  - $P \parallel Q \sim Q \parallel P$
  - $P \parallel Nil \sim P$
  - $(P \parallel Q) \parallel R \sim Q \parallel (P \parallel R)$
  - $\cdots$

Question

Should we look any further???

Problems with Internal Actions

Question

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

Problem

Strong bisimilarity does not abstract away from τ actions.

Example: SmUni $\not\sim$ Spec

$$
\begin{tikzcd}
\text{SmUni} \arrow[d, "\overline{pub}"] & \not\sim & \text{Spec} \arrow[loop below, "\overline{pub}"] \\
(\nu\, coin, coffee)(CM \parallel CS_1) \arrow[d, "\tau"] & & \\
(\nu\, coin, coffee)(CM_1 \parallel CS_2) \arrow[d, "\tau"] & & \\
(\nu\, coin, coffee)(CM \parallel CS) \arrow[uu, bend right, "\overline{pub}"'] & &
\end{tikzcd}
$$

Problems with Internal Actions

Question

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

Problem

Strong bisimilarity does not abstract away from τ actions.

Example: SmUni $\not\sim$ Spec

$$
\begin{tikzcd}
\text{SmUni} \arrow[d, "\overline{pub}"] & \not\sim & \text{Spec} \arrow[loop below, "\overline{pub}"] \\
(\nu\, coin, coffee)(CM \parallel CS_1) \arrow[d, "\tau"] & & \\
(\nu\, coin, coffee)(CM_1 \parallel CS_2) \arrow[d, "\tau"] & & \\
(\nu\, coin, coffee)(CM \parallel CS) \arrow[uu, bend right, "\overline{pub}"'] & &
\end{tikzcd}
$$

Problems with Internal Actions

Question

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

Problem

Strong bisimilarity does not abstract away from τ actions.

Example: SmUni $\not\sim$ Spec

$$\begin{tikzcd}
\text{SmUni} \arrow[d, "\overline{pub}"] & \not\sim & \text{Spec} \arrow[loop below, "\overline{pub}"] \\
(\nu\, coin, coffee)(CM \parallel CS_1) \arrow[d, "\tau"] & & \\
(\nu\, coin, coffee)(CM_1 \parallel CS_2) \arrow[d, "\tau"] & & \\
(\nu\, coin, coffee)(CM \parallel CS) \arrow[uu, bend right, "\overline{pub}"'] & &
\end{tikzcd}$$

Problems with Internal Actions

Question

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

Problem

Strong bisimilarity does not abstract away from τ actions.

Example: SmUni $\not\sim$ Spec

$$
\begin{tikzcd}
\mathsf{SmUni} \arrow[d, "\overline{pub}"] & \not\sim & \mathsf{Spec} \arrow[loop below, "\overline{pub}"] \\
(\nu coin, coffee)(CM \parallel CS_1) \arrow[d, "\tau"] & & \\
(\nu coin, coffee)(CM_1 \parallel CS_2) \arrow[d, "\tau"] & & \\
(\nu coin, coffee)(CM \parallel CS) \arrow[uu, bend right=60, swap, "\overline{pub}"] & &
\end{tikzcd}
$$

Weak Transition Relation

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

Definition of Weak Transition Relation

Below, $\circ$ stands for function composition.

$$\stackrel{a}{\Longrightarrow} = \begin{cases} (\xrightarrow{\tau})^* \circ \xrightarrow{a} \circ (\xrightarrow{\tau})^* & \text{if } a \neq \tau \\ (\xrightarrow{\tau})^* & \text{if } a = \tau \end{cases}$$

What does $s \stackrel{a}{\Longrightarrow} t$ informally mean?

- If $a \neq \tau$ then $s \stackrel{a}{\Longrightarrow} t$ means that
  from s we can get to t by doing zero or more τ actions, followed by the action a, followed by zero or more τ actions.
- If $a = \tau$ then $s \stackrel{\tau}{\Longrightarrow} t$ means that
  from s we can get to t by doing zero or more τ actions.

Weak Transition Relation

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

Definition of Weak Transition Relation

Below, $\circ$ stands for function composition.

$$\overset{a}{\Longrightarrow} = \begin{cases} (\xrightarrow{\tau})^* \circ \xrightarrow{a} \circ (\xrightarrow{\tau})^* & \text{if } a \neq \tau \\ (\xrightarrow{\tau})^* & \text{if } a = \tau \end{cases}$$

What does $s \overset{a}{\Longrightarrow} t$ informally mean?

- If $a \neq \tau$ then $s \overset{a}{\Longrightarrow} t$ means that
  from s we can get to t by doing zero or more τ actions, followed by the action a, followed by zero or more τ actions.
- If $a = \tau$ then $s \overset{\tau}{\Longrightarrow} t$ means that
  from s we can get to t by doing zero or more τ actions.

Weak Bisimilarity

Let $(Proc, Act, \{\xrightarrow{a} | \; a \in Act\})$ be an LTS such that $\tau \in Act$.

Weak Bisimulation

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

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

Weak Bisimilarity

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

$$\approx = \bigcup \{\mathcal{R} \mid \mathcal{R} \text{ is a weak bisimulation}\}$$

Weak Bisimilarity

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

Weak Bisimulation

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

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

Weak Bisimilarity

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

$$\approx \; = \bigcup \{\mathcal{R} \mid \mathcal{R} \text{ is a weak bisimulation}\}$$

Weak Bisimulation Game

Definition

All the same except that

- defender can now answer using $\overset{a}{\Longrightarrow}$ moves.

The attacker is still using only $\xrightarrow{a}$ 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 $\overset{a}{\Longrightarrow}$ moves.

The attacker is still using only $\xrightarrow{a}$ moves.

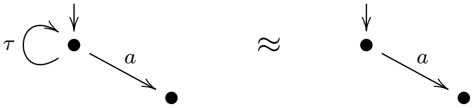
Theorem

- 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.
  - $a.\tau.P \approx a.P$
  - $P + \tau.P \approx \tau.P$
  - $a.(P + \tau.Q) \approx a.(P + \tau.Q) + a.Q$
  - $P + Q \approx Q + P \qquad P \parallel Q \approx Q \parallel P \qquad P + Nil \approx P \qquad \ldots$
- strong bisimilarity is included in weak bisimilarity ($\sim \subseteq \approx$)
- abstracts from τ loops



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* $\alpha \in Act$
- $P \mid R \approx Q \mid R$ *and* $R \mid P \approx R \mid Q$ *for each CCS process* R
- $(\nu a)\,P \approx (\nu a)Q$ *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* $\alpha \in Act$
- $P \mid R \approx Q \mid R$ *and* $R \mid P \approx R \mid Q$ *for each CCS process* R
- $(\nu a)\, P \approx (\nu a) Q$ *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* $\alpha \in Act$
- $P \mid R \approx Q \mid R$ *and* $R \mid P \approx R \mid Q$ *for each CCS process* R
- $(\nu a)\, P \approx (\nu a) Q$ *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.

Case Study: Communication Protocol

$$\begin{aligned}
\mathsf{Send} &\stackrel{\text{def}}{=} \mathsf{acc}.\mathsf{Sending} \\
\mathsf{Sending} &\stackrel{\text{def}}{=} \overline{\mathsf{send}}.\mathsf{Wait} \\
\mathsf{Wait} &\stackrel{\text{def}}{=} \mathsf{ack}.\mathsf{Send} + \mathsf{error}.\mathsf{Sending}
\end{aligned}
\qquad
\begin{aligned}
\mathsf{Rec} &\stackrel{\text{def}}{=} \mathsf{trans}.\mathsf{Del} \\
\mathsf{Del} &\stackrel{\text{def}}{=} \overline{\mathsf{del}}.\mathsf{Ack} \\
\mathsf{Ack} &\stackrel{\text{def}}{=} \overline{\mathsf{ack}}.\mathsf{Rec}
\end{aligned}$$

$$\begin{aligned}
\mathsf{Med} &\stackrel{\text{def}}{=} \mathsf{send}.\mathsf{Med}' \\
\mathsf{Med}' &\stackrel{\text{def}}{=} \tau.\mathsf{Err} + \overline{\mathsf{trans}}.\mathsf{Med} \\
\mathsf{Err} &\stackrel{\text{def}}{=} \overline{\mathsf{error}}.\mathsf{Med}
\end{aligned}$$

Case Study: Communication Protocol

$$
\begin{aligned}
\mathsf{Send} &\stackrel{\text{def}}{=} \mathsf{acc}.\mathsf{Sending} \\
\mathsf{Sending} &\stackrel{\text{def}}{=} \overline{\mathsf{send}}.\mathsf{Wait} \\
\mathsf{Wait} &\stackrel{\text{def}}{=} \mathsf{ack}.\mathsf{Send} + \mathsf{error}.\mathsf{Sending}
\end{aligned}
\qquad
\begin{aligned}
\mathsf{Rec} &\stackrel{\text{def}}{=} \mathsf{trans}.\mathsf{Del} \\
\mathsf{Del} &\stackrel{\text{def}}{=} \overline{\mathsf{del}}.\mathsf{Ack} \\
\mathsf{Ack} &\stackrel{\text{def}}{=} \overline{\mathsf{ack}}.\mathsf{Rec}
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{Med} &\stackrel{\text{def}}{=} \mathsf{send}.\mathsf{Med}' \\
\mathsf{Med}' &\stackrel{\text{def}}{=} \tau.\mathsf{Err} + \overline{\mathsf{trans}}.\mathsf{Med} \\
\mathsf{Err} &\stackrel{\text{def}}{=} \overline{\mathsf{error}}.\mathsf{Med}
\end{aligned}
$$

Verification Question

$$\mathsf{Impl} \stackrel{\text{def}}{=} (\nu\, \mathsf{send}, \mathsf{trans}, \mathsf{ack}, \mathsf{error})(\mathsf{Send} \parallel \mathsf{Med} \parallel \mathsf{Rec})$$

$$\mathsf{Spec} \stackrel{\text{def}}{=} \mathsf{acc}.\overline{\mathsf{del}}.\mathsf{Spec}$$

Question

$$\mathsf{Impl} \stackrel{?}{\approx} \mathsf{Spec}$$

Check it by yourself in CAAL!

Verification Question

$$\mathsf{Impl} \stackrel{\text{def}}{=} (\nu\, \mathsf{send}, \mathsf{trans}, \mathsf{ack}, \mathsf{error})(\mathsf{Send} \parallel \mathsf{Med} \parallel \mathsf{Rec})$$

$$\mathsf{Spec} \stackrel{\text{def}}{=} \mathsf{acc}.\overline{\mathsf{del}}.\mathsf{Spec}$$

Question

$$\mathsf{Impl} \stackrel{?}{\approx} \mathsf{Spec}$$

Check it by yourself in CAAL!

Verification Question

$$\mathsf{Impl} \stackrel{\text{def}}{=} (\nu\, \mathsf{send}, \mathsf{trans}, \mathsf{ack}, \mathsf{error})(\mathsf{Send} \parallel \mathsf{Med} \parallel \mathsf{Rec})$$

$$\mathsf{Spec} \stackrel{\text{def}}{=} \mathsf{acc}.\overline{\mathsf{del}}.\mathsf{Spec}$$

Question

$$\mathsf{Impl} \stackrel{?}{\approx} \mathsf{Spec}$$

Check it by yourself in CAAL!