Weak bisimilarity (reprise) and weak bisimulation games
Properties of weak bisimilarity
Example: a communication protocol and its modelling in CCS
Concurrency workbench (CWB)
An introduction to Hennessy-Milner logic (HML)
Syntax and semantics of HML
Let \((Proc, Act, \{\xrightarrow{a} \mid a \in Act\})\) be an LTS such that \(\tau \in Act\).

**Definition of the Weak Transition Relations**

Let \(a\) be an action or \(\varepsilon\):

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

**Definition**

If \(a\) is an observable action, then \(\hat{a} = a\). On the other hand, \(\hat{\tau} = \varepsilon\).
Let \((\text{Proc}, \text{Act}, \{ \overset{a}{\rightarrow} \mid a \in \text{Act}\})\) 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 \(s \overset{a}{\rightarrow} s'\) then \(t \overset{\hat{a}}{\rightarrow} t'\) for some \(t'\) such that \((s', t') \in R\)
- if \(t \overset{a}{\rightarrow} t'\) then \(s \overset{\hat{a}}{\rightarrow} s'\) 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 \((p_1, p_2) \in R\).

\[
\approx = \bigcup \{ R \mid R \text{ is a weak bisimulation}\}
\]
Definition

Same as for the strong bisimulation game except that
- defender can now answer using $\xrightarrow{a}$ moves.
The attacker is still using only $\rightarrow a$ moves.

Let’s play!

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**

Same as for the strong bisimulation game except that:
- **defender** can now answer using $\overset{a}{\Rightarrow}$ moves.
- The **attacker** is still using only $\overset{a}{\rightarrow}$ moves.

Let’s play!

**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 $(s, t)$. 

Reactive Systems: Modelling, Specification and Verification
Properties of $\approx$

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

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

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

\[
\begin{align*}
\text{Send} & \overset{\text{def}}{=} \text{acc}.\text{Sending} \\
\text{Sending} & \overset{\text{def}}{=} \text{send}.\text{Wait} \\
\text{Wait} & \overset{\text{def}}{=} \text{ack}.\text{Send} + \text{error}.\text{Sending} \\
\text{Rec} & \overset{\text{def}}{=} \text{trans}.\text{Del} \\
\text{Del} & \overset{\text{def}}{=} \text{del}.\text{Ack} \\
\text{Ack} & \overset{\text{def}}{=} \text{ack}.\text{Rec} \\
\text{Med} & \overset{\text{def}}{=} \text{send}.\text{Med}' \\
\text{Med}' & \overset{\text{def}}{=} \tau.\text{Err} + \text{trans}.\text{Med} \\
\text{Err} & \overset{\text{def}}{=} \text{error}.\text{Med}
\end{align*}
\]
\[
\text{Impl} \overset{\text{def}}{=} (\text{Send} \mid \text{Med} \mid \text{Rec}) \setminus \{\text{send}, \text{trans}, \text{ack}, \text{error}\}
\]

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

**Question**

Impl \overset{?}{\approx} Spec

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

2. Use the Concurrency WorkBench (CWB).
Impl \stackrel{\text{def}}{=} (\text{Send} \mid \text{Med} \mid \text{Rec}) \setminus \{\text{send, trans, ack, error}\}

Spec \stackrel{\text{def}}{=} \text{acc.del.Spec}

Question

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

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 the 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

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

2. Use the Concurrency WorkBench (CWB).
Impl $\overset{\text{def}}{=} (\text{Send} | \text{Med} | \text{Rec}) \setminus \{\text{send, trans, ack, error}\}$

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

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

**CCS Definitions**

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

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

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

**CWB Program (protocol.cwb)**

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

set L = \{send, trans, ack, error\};
agent Impl = (Send \mid Med \mid Rec) \setminus L;
agent Spec = acc.'del.Spec;
[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?

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

Conclusion

Weak bisimilarity is not a congruence for CCS.
Theorem

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

- \( \alpha. P \approx \alpha. Q \) for each action \( \alpha \in \text{Act} \)
- \( P \mid R \approx Q \mid R \) and \( R \mid P \approx R \mid Q \) 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.
Let $Impl$ be an implementation of a system (e.g. in CCS syntax).

**Equivalence Checking Approach**

$$Impl \equiv Spec$$

- $\equiv$ is an abstract equivalence, e.g. $\sim$ or $\approx$
- $Spec$ is often expressed in the same language as $Impl$
- $Spec$ provides the full specification of the intended behaviour

**Model Checking Approach**

$$Impl \models Property$$

- $\models$ is the satisfaction relation
- $Property$ is a particular feature, often expressed via a logic
- $Property$ is a partial specification of the intended behaviour
Let $Impl$ be an implementation of a system (e.g. in CCS syntax).

**Equivalence Checking Approach**

\[ Impl \equiv Spec \]

- $\equiv$ is an abstract equivalence, e.g. $\sim$ or $\approx$
- $Spec$ is often expressed in the same language as $Impl$
- $Spec$ provides the full specification of the intended behaviour

**Model Checking Approach**

\[ Impl \models Property \]

- $\models$ is the satisfaction relation
- $Property$ is a particular feature, often expressed via a logic
- $Property$ is a partial specification of the intended behaviour
Our Aim

Develop a logic in which we can express interesting properties of reactive systems.
Modal Properties – what can happen now (possibility, necessity)

- drink a coffee (can drink a coffee now)
- does not drink tea
- drinks both tea and coffee
- drinks tea after coffee

Temporal Properties – behaviour in time

- never drinks any alcohol
  (safety property: nothing bad can happen)
- eventually will have a glass of wine
  (liveness property: something good will happen)

Can these properties be expressed using equivalence checking?
Modal Properties – what can happen now (possibility, necessity)

- drink a coffee (can drink a coffee now)
- does not drink tea
- drinks both tea and coffee
- drinks tea after coffee

Temporal Properties – behaviour in time

- never drinks any alcohol
  (safety property: nothing bad can happen)
- eventually will have a glass of wine
  (liveness property: something good will happen)

Can these properties be expressed using equivalence checking?
Modal Properties – what can happen now (possibility, necessity)

- drink a coffee (can drink a coffee now)
- does not drink tea
- drinks both tea and coffee
- drinks tea after coffee

Temporal Properties – behaviour in time

- never drinks any alcohol
  (safety property: nothing bad can happen)
- eventually will have a glass of wine
  (liveness property: something good will happen)

Can these properties be expressed using equivalence checking?
Syntax of the Formulae \((a \in \text{Act})\)

\[
F, \ G \ ::= \ tt \ | \ ff \ | \ F \land G \ | \ F \lor G \ | \ \langle a \rangle F \ | \ [a]F
\]

Intuition:
- \(tt\) all processes satisfy this property
- \(ff\) no process satisfies this property
- \(\land, \lor\) usual logical AND and OR
- \(\langle a \rangle F\) there is at least one \(a\)-successor that satisfies \(F\)
- \([a]F\) all \(a\)-successors have to satisfy \(F\)

Remark
Temporal properties like *always/never in the future* or *eventually* are not included.
Syntax of the Formulae ($a \in \text{Act}$)

$$F, G ::= \text{tt} | \text{ff} | F \land G | F \lor G | \langle a \rangle F | [a]F$$

Intuition:
- $\text{tt}$ all processes satisfy this property
- $\text{ff}$ no process satisfies this property
- $\land, \lor$ usual logical AND and OR
- $\langle a \rangle F$ there is at least one $a$-successor that satisfies $F$
- $[a]F$ all $a$-successors have to satisfy $F$

Remark
Temporal properties like *always/never in the future* or *eventually* are not included.
Syntax of the Formulae ($a \in \text{Act}$)

\[ F, G ::= \text{tt} \mid \text{ff} \mid F \land G \mid F \lor G \mid \langle a \rangle F \mid [a]F \]

Intuition:
- \text{tt} all processes satisfy this property
- \text{ff} no process satisfies this property
- $\land$, $\lor$ usual logical AND and OR
- $\langle a \rangle F$ there is at least one $a$-successor that satisfies $F$
- $[a]F$ all $a$-successors have to satisfy $F$

Remark
Temporal properties like *always/never in the future* or *eventually* are not included.
Let \((\text{Proc}, \text{Act}, \{ \xrightarrow{a} | a \in \text{Act}\})\) be an LTS.

### Validity of the logical triple \(p \models F\) (\(p \in \text{Proc}, F\) a HM formula)

- \(p \models \text{tt}\) for each \(p \in \text{Proc}\)
- \(p \models \text{ff}\) for no \(p\) (we also write \(p \not\models \text{ff}\))
- \(p \models F \land G\) iff \(p \models F\) and \(p \models G\)
- \(p \models F \lor G\) iff \(p \models F\) or \(p \models G\)
- \(p \models \langle a \rangle F\) iff \(p \xrightarrow{a} p'\) for some \(p' \in \text{Proc}\) such that \(p' \models F\)
- \(p \models [a]F\) iff \(p' \models F\), for all \(p' \in \text{Proc}\) such that \(p \xrightarrow{a} p'\)

We write \(p \not\models F\) whenever \(p\) does not satisfy \(F\).
What about Negation?

For every formula $F$ we define the formula $F^c$ as follows:

- $tt^c = ff$
- $ff^c = tt$
- $(F \land G)^c = F^c \lor G^c$
- $(F \lor G)^c = F^c \land G^c$
- $(\langle a \rangle F)^c = [a]F^c$
- $([a]F)^c = \langle a \rangle F^c$

**Theorem** ($F^c$ is equivalent to the negation of $F$)

For any $p \in \text{Proc}$ and any HM formula $F$

1. $p \models F \iff p \not\models F^c$
2. $p \not\models F \iff p \models F^c$
What about Negation?

For every formula $F$ we define the formula $F^c$ as follows:

- $tt^c = ff$
- $ff^c = tt$
- $(F \land G)^c = F^c \lor G^c$
- $(F \lor G)^c = F^c \land G^c$
- $(\langle a \rangle F)^c = [a]F^c$
- $([a]F)^c = \langle a \rangle F^c$

**Theorem** ($F^c$ is equivalent to the negation of $F$)

For any $p \in \text{Proc}$ and any HM formula $F$

1. $p \models F \implies p \not\models F^c$
2. $p \not\models F \implies p \models F^c$