Bisimilarity and Hennessy-Milner Logic

Luca Aceto
ICE-TCS, School of Computer Science, Reykjavik University
Tentative Plan

1. An introduction to Hennessy-Milner logic (HML)
2. Syntax and semantics of HML
3. Correspondence with bisimilarity
4. Hennessy-Milner logic and temporal properties
5. Hennessy-Milner logic with recursion
6. ...?
Let $Impl$ be an implementation of a system.

**Equivalence Checking Approach**

$$Impl \equiv Spec$$

- $\equiv$ is a behavioural equivalence, e.g. $\sim$ or $\approx$
- $Spec$ is 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
Verifying Correctness of Reactive Systems

Let $Impl$ be an implementation of a system.

**Equivalence Checking Approach**

$Impl \equiv Spec$

- $\equiv$ is a behavioural equivalence, e.g. $\sim$ or $\approx$
- $Spec$ is 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.
### Logical 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?
Logical 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?
Logical 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?
Hennessy-Milner Logic – Syntax

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 \text{always/never in the future} or \text{eventually} are not included.
Hennessy-Milner Logic – Syntax

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.
Hennessy-Milner Logic – Syntax

Syntax of the Formulae \((a \in \text{Act})\)

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

Intuition:
- \(\texttt{tt}\) all processes satisfy this property
- \(\texttt{ff}\) 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.
Hennessy-Milner Logic – Semantics

Let \((\text{Proc}, \text{Act}, \{\xrightarrow{a} \mid a \in \text{Act}\})\) 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 $p \in Proc$ and any HM formula $F$

1. $p \models F \quad \rightarrow \quad p \not\models F^c$
2. $p \not\models F \quad \rightarrow \quad p \models F^c$
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 $p \in Proc$ and any HM formula $F$

1. $p \models F \implies p \not\models F^c$
2. $p \not\models F \implies p \models F^c$
Hennessy-Milner Logic – Denotational Semantics

For a formula $F$ let $[F] \subseteq \text{Proc}$ contain all states that satisfy $F$.

**Denotational Semantics: $[\cdot ] : \text{Formulae} \rightarrow 2^{\text{Proc}}$**

- $[tt] = \text{Proc}$ and $[ff] = \emptyset$
- $[F \lor G] = [F] \cup [G]$
- $[F \land G] = [F] \cap [G]$
- $[\langle a \rangle F] = \langle \cdot a \cdot \rangle [F]$
- $[[a]F] = [\cdot a \cdot ][F]$

where $\langle \cdot a \cdot \rangle$, $[\cdot a \cdot ] : 2^{(\text{Proc})} \rightarrow 2^{(\text{Proc})}$ are defined by

$\langle \cdot a \cdot \rangle S = \{ p \in \text{Proc} \mid \exists p'. \ a \rightarrow p' \text{ and } p' \in S \}$

$[\cdot a \cdot ] S = \{ p \in \text{Proc} \mid \forall p'. \ a \rightarrow p' \implies p' \in S \}$.
The Correspondence Theorem

**Theorem**

Let \((Proc, \text{Act}, \{\xrightarrow{a} \mid a \in \text{Act}\})\) be an LTS, \(p \in Proc\) and \(F\) a formula of Hennessy-Milner logic. Then

\[ p \models F \iff p \in \llbracket F \rrbracket. \]

**Proof:** By induction on the structure of the formula \(F\). How?
The Correspondence Theorem

**Theorem**

Let \((\text{Proc}, \text{Act}, \{ \overset{a}{\rightarrow} \mid a \in \text{Act} \})\) be an LTS, \(p \in \text{Proc}\) and \(F\) a formula of Hennessy-Milner logic. Then

\[ p \models F \iff p \in \llbracket F \rrbracket. \]

**Proof:** By induction on the structure of the formula \(F\). How?
Image-Finite System

Let \((\text{Proc}, \text{Act}, \{ \overset{a}{\rightarrow} | a \in \text{Act}\})\) be an LTS. We call it image-finite iff for every \(p \in \text{Proc}\) and every \(a \in \text{Act}\) the set

\[\{ p' \in \text{Proc} \mid p \overset{a}{\rightarrow} p' \}\]

is finite.

**Question:** Are there any connections between image finiteness and finite branching?
Theorem (Hennessy-Milner)

Let \((\text{Proc}, \text{Act}, \{ \xrightarrow{a} \mid a \in \text{Act} \})\) be an image-finite LTS and \(p, q \in St\). Then

\[ p \sim q \]

if and only if

for every HM formula \(F\): \((p \models F \iff q \models F)\).

Proof?
CWB Session

```bash
$ ./xccscwbb.x86-linux
> input "hm.cwb"
> print;
> help logic;
> checkprop(S,<a>(<b>T & <c>T));
  true
> checkprop(T,<a>(<b>T & <c>T));
  false
> help dfstrong;
> dfstrong(S,T);
  [a]<b>T
> exit;
```
Is Hennessy-Milner Logic Powerful Enough?

Modal depth (nesting degree) for Hennessy-Milner formulae:

- $md(tt) = md(ff) = 0$
- $md(F \land G) = md(F \lor G) = \max\{md(F), md(G)\}$
- $md([a]F) = md(\langle a \rangle F) = md(F) + 1$

Idea: a formula $F$ can “see” only up to depth $md(F)$.

Theorem (let $F$ be a HM formula and $k = md(F)$)

If the defender has a defending strategy in the strong bisimulation game from $s$ and $t$ up to $k$ rounds then $s \models F$ if and only if $t \models F$.

Conclusion

There is no Hennessy-Milner formula $F$ that can detect a deadlock in an arbitrary LTS.
Is Hennessy-Milner Logic Powerful Enough?

Modal depth (nesting degree) for Hennessy-Milner formulae:
- \( md(tt) = md(\neg \neg F) = 0 \)
- \( md(F \land G) = md(F \lor G) = \max\{md(F), md(G)\} \)
- \( md([a]F) = md(\langle a \rangle F) = md(F) + 1 \)

Idea: a formula \( F \) can “see” only up to depth \( md(F) \).

Theorem (let \( F \) be a HM formula and \( k = md(F) \))

If the defender has a defending strategy in the strong bisimulation game from \( s \) and \( t \) up to \( k \) rounds then \( s \models F \) if and only if \( t \models F \).

Conclusion

There is no Hennessy-Milner formula \( F \) that can detect a deadlock in an arbitrary LTS.
Is Hennessy-Milner Logic Powerful Enough?

Modal depth (nesting degree) for Hennessy-Milner formulae:

- \( md(tt) = md(\neg\neg) = 0 \)
- \( md(F \land G) = md(F \lor G) = \max\{md(F), md(G)\} \)
- \( md([a]F) = md(\langle a \rangle F) = md(F) + 1 \)

Idea: a formula \( F \) can “see” only up to depth \( md(F) \).

**Theorem** (let \( F \) be a HM formula and \( k = md(F) \))

If the defender has a defending strategy in the strong bisimulation game from \( s \) and \( t \) up to \( k \) rounds then \( s \models F \) if and only if \( t \models F \).

**Conclusion**

There is no Hennessy-Milner formula \( F \) that can detect a deadlock in an arbitrary LTS.
Temporal Properties not Expressible in HM Logic

\[ s \models Inv(F) \text{ iff all states reachable from } s \text{ satisfy } F \]

\[ s \models Pos(F) \text{ iff there is a reachable state which satisfies } F \]

**Fact**

Properties \( Inv(F) \) and \( Pos(F) \) are not expressible in HM logic.

Let \( Act = \{a_1, a_2, \ldots, a_n\} \) be a finite set of actions. We define

\[ \langle Act \rangle F \overset{\text{def}}{=} \langle a_1 \rangle F \lor \langle a_2 \rangle F \lor \ldots \lor \langle a_n \rangle F \]

\[ [Act]F \overset{\text{def}}{=} [a_1]F \land [a_2]F \land \ldots \land [a_n]F \]

\[ Inv(F) \equiv F \land [Act]F \land [Act][Act]F \land [Act][Act][Act]F \land \ldots \]

\[ Pos(F) \equiv F \lor \langle Act \rangle F \lor \langle Act \rangle \langle Act \rangle F \lor \langle Act \rangle \langle Act \rangle \langle Act \rangle F \lor \ldots \]
Temporal Properties not Expressible in HM Logic

\[ s \models Inv(F) \text{ iff all states reachable from } s \text{ satisfy } F \]
\[ s \models Pos(F) \text{ iff there is a reachable state which satisfies } F \]

**Fact**

Properties \( Inv(F) \) and \( Pos(F) \) are not expressible in HM logic.

Let \( Act = \{a_1, a_2, \ldots, a_n\} \) be a finite set of actions. We define

- \( \langle Act \rangle F \overset{\text{def}}{=} \langle a_1 \rangle F \lor \langle a_2 \rangle F \lor \ldots \lor \langle a_n \rangle F \)
- \( [Act] F \overset{\text{def}}{=} [a_1] F \land [a_2] F \land \ldots \land [a_n] F \)

\[ Inv(F) \overset{\text{def}}{=} F \land [Act] F \land [Act][Act] F \land [Act][Act][Act] F \land \ldots \]
\[ Pos(F) \overset{\text{def}}{=} F \lor \langle Act \rangle F \lor \langle Act \rangle \langle Act \rangle F \lor \langle Act \rangle \langle Act \rangle \langle Act \rangle F \lor \ldots \]
Temporal Properties not Expressible in HM Logic

\[ s \models Inv(F) \text{ iff all states reachable from } s \text{ satisfy } F \]
\[ s \models Pos(F) \text{ iff there is a reachable state which satisfies } F \]

**Fact**

Properties \( Inv(F) \) and \( Pos(F) \) are not expressible in HM logic.

Let \( Act = \{a_1, a_2, \ldots, a_n\} \) be a finite set of actions. We define

\[ \langle Act \rangle F \overset{\text{def}}{=} \langle a_1 \rangle F \lor \langle a_2 \rangle F \lor \ldots \lor \langle a_n \rangle F \]

\[ [Act]F \overset{\text{def}}{=} [a_1]F \land [a_2]F \land \ldots \land [a_n]F \]

\[ Inv(F) \equiv F \land [Act]F \land [Act][Act]F \land [Act][Act][Act]F \land \ldots \]
\[ Pos(F) \equiv F \lor \langle Act \rangle F \lor \langle Act \rangle \langle Act \rangle F \lor \langle Act \rangle \langle Act \rangle \langle Act \rangle F \lor \ldots \]
Infinite Conjunctions and Disjunctions vs. Recursion

Problems

- infinite formulae are not allowed in HM logic
- infinite formulae are difficult to handle

Why don’t we use recursion?

- $\text{Inv}(F)$ expressed by $X \overset{\text{def}}{=} F \land [\text{Act}]X$
- $\text{Pos}(F)$ expressed by $X \overset{\text{def}}{=} F \lor \langle \text{Act} \rangle X$

Question: How to define the semantics of such equations?
Infinite Conjunctions and Disjunctions vs. Recursion

Problems

- infinite formulae are not allowed in HM logic
- infinite formulae are difficult to handle

Why don’t we use recursion?

- \( Inv(F) \) expressed by \( X \overset{\text{def}}{=} F \land [Act]X \)
- \( Pos(F) \) expressed by \( X \overset{\text{def}}{=} F \lor \langle Act\rangle X \)

Question: How to define the semantics of such equations?
Introduction to Model Checking
Hennessy-Milner Logic
Hennessy-Milner Logic with One Recursive Definition
Selection of Temporal Properties

Temporal Properties – Invariance and Possibility

Infinite Conjunctions and Disjunctions vs. Recursion

Problems
- infinite formulae are not allowed in HM logic
- infinite formulae are difficult to handle

Why don’t we use recursion?
- $Inv(F)$ expressed by $X \triangleq F \land [Act]X$
- $Pos(F)$ expressed by $X \triangleq F \lor \langle Act \rangle X$

Question: How to define the semantics of such equations?
Solving Equations is Tricky

Equations over Natural Numbers ($n \in \mathbb{N}$)

- $n = 2 \times n$ one solution $n = 0$
- $n = n + 1$ no solution
- $n = 1 \times n$ many solutions (every $n \in \text{Nat}$ is a solution)

Equations over Sets of Integers ($M \in 2^\mathbb{N}$)

- $M = (\{7\} \cap M) \cup \{7\}$ one solution $M = \{7\}$
- $M = \mathbb{N} \setminus M$ no solution
- $M = \{3\} \cup M$ each $M \supseteq \{3\}$ is a solution

What about Equations over Processes?

$X \overset{\text{def}}{=} [a]f \lor \langle a \rangle X \Rightarrow \text{find } S \subseteq 2^{\text{Proc}} \text{ s.t. } S = [\cdot a \cdot]\emptyset \cup \langle \cdot a \cdot \rangle S$
Solving Equations is Tricky

Equations over Natural Numbers \((n \in \mathbb{N})\)

\[
\begin{align*}
n &= 2 \times n & \text{one solution } n = 0 \\
n &= n + 1 & \text{no solution} \\
n &= 1 \times n & \text{many solutions (every } n \in \text{Nat} \text{ is a solution)}
\end{align*}
\]

Equations over Sets of Integers \((M \in 2^\mathbb{N})\)

\[
\begin{align*}
M &= (\{7\} \cap M) \cup \{7\} & \text{one solution } M = \{7\} \\
M &= \mathbb{N} \setminus M & \text{no solution} \\
M &= \{3\} \cup M & \text{each } M \supseteq \{3\} \text{ is a solution}
\end{align*}
\]

What about Equations over Processes?

\[
X \overset{\text{def}}{=} [a]f \lor \langle a \rangle X \quad \Rightarrow \quad \text{find } S \subseteq 2^{\text{Proc}} \text{ s.t. } S = [\cdot a \cdot] \emptyset \cup \langle \cdot a \cdot \rangle S
\]
Solving Equations is Tricky

Equations over Natural Numbers ($n \in \mathbb{N}$)

- $n = 2 \cdot n$: one solution $n = 0$
- $n = n + 1$: no solution
- $n = 1 \cdot n$: many solutions (every $n \in \text{Nat}$ is a solution)

Equations over Sets of Integers ($M \in 2^\mathbb{N}$)

- $M = (\{7\} \cap M) \cup \{7\}$: one solution $M = \{7\}$
- $M = \mathbb{N} \setminus M$: no solution
- $M = \{3\} \cup M$: each $M \supseteq \{3\}$ is a solution

What about Equations over Processes?

$X \overset{\text{def}}{=} [a]ff \lor \langle a \rangle X \Rightarrow \text{find } S \subseteq 2^{\text{Proc}} \text{ s.t. } S = [\cdot a \cdot]\emptyset \cup \langle \cdot a \cdot \rangle S$
Monotonic Functions

Monotonic Function and Fixed Points

A function \( f : 2^{Proc} \rightarrow 2^{Proc} \) is called monotonic iff

\[
X \subseteq Y \implies f(X) \subseteq f(Y)
\]

for all \( X, Y \in 2^{Proc} \).

A set \( X \in 2^{Proc} \) is called a fixed point of \( f \) iff \( X = f(X) \).

Questions

Is the function \( f(X) = X \cup \{s, t\} \) monotonic? What about \( g(X) = \text{Proc} \setminus X \)? Do these functions have fixed points?
Tarski’s Fixed Point Theorem

Theorem (Tarski)

Let \( f : 2^{Proc} \rightarrow 2^{Proc} \) be a monotonic function. Then \( f \) has a unique largest fixed point \( z_{\text{max}} \) and a unique least fixed point \( z_{\text{min}} \) given by:

\[
z_{\text{max}} \overset{\text{def}}{=} \bigcup \{ X \in 2^{Proc} \mid X \subseteq f(X) \}
\]

\[
z_{\text{min}} \overset{\text{def}}{=} \bigcap \{ X \in 2^{Proc} \mid f(X) \subseteq X \}
\]
Computing Min and Max Fixed Points on Finite Sets

Let $f : 2^{\text{Proc}} \rightarrow 2^{\text{Proc}}$ be monotonic.
Let $f^1(X) \overset{\text{def}}{=} f(X)$ and $f^n(X) \overset{\text{def}}{=} f(f^{n-1}(X))$ for $n > 1$, i.e.,

$$f^n(X) = f(f(\ldots f(X) \ldots)).$$

$n$ times

Theorem

If $2^{\text{Proc}}$ is a finite set then there exist integers $M, m > 0$ such that

- $z_{\text{max}} = f^M(\text{Proc})$
- $z_{\text{min}} = f^m(\emptyset)$

Idea (for $z_{\text{min}}$): The following sequence stabilizes for any finite $2^{\text{Proc}}$

$$\emptyset \subseteq f(\emptyset) \subseteq f(f(\emptyset)) \subseteq f(f(f(\emptyset))) \subseteq \cdots$$
Computing Min and Max Fixed Points on Finite Sets

Let $f : 2^{Proc} \rightarrow 2^{Proc}$ be monotonic.
Let $f^1(X) \overset{\text{def}}{=} f(X)$ and $f^n(X) \overset{\text{def}}{=} f(f^{n-1}(X))$ for $n > 1$, i.e.,

$$f^n(X) = f(f(\ldots f(X)\ldots)).$$


**Theorem**

If $2^{Proc}$ is a finite set then there exist integers $M, m > 0$ such that

- $z_{\max} = f^M(Proc)$
- $z_{\min} = f^m(\emptyset)$

Idea (for $z_{\min}$): The following sequence stabilizes for any finite $2^{Proc}$

$$\emptyset \subseteq f(\emptyset) \subseteq f(f(\emptyset)) \subseteq f(f(f(\emptyset))) \subseteq \cdots$$
HML with One Recursively Defined Variable

Syntax of Formulae

Formulae are given by the following abstract syntax

\[ F ::= X \mid tt \mid ff \mid F_1 \land F_2 \mid F_1 \lor F_2 \mid \langle a \rangle F \mid [a]F \]

where \( a \in Act \) and \( X \) is a distinguished variable with a definition

\[ X_{\text{min}} \equiv F_X, \text{ or } X_{\text{max}} \equiv F_X \]

such that \( F_X \) is a formula of the logic (can contain \( X \)).

Semantics?

For every formula \( F \) we define a function \( O_F : 2^{Proc} \rightarrow 2^{Proc} \) s.t.

- if \( S \) is the set of processes that satisfy \( X \) then
- \( O_F(S) \) is the set of processes that satisfy \( F \).
Syntax of Formulae

Formulae are given by the following abstract syntax

\[ F ::= X \mid tt \mid ff \mid F_1 \land F_2 \mid F_1 \lor F_2 \mid \langle a \rangle F \mid [a]F \]

where \( a \in Act \) and \( X \) is a distinguished variable with a definition

- \( X \overset{\text{min}}{=} F_X \), or \( X \overset{\text{max}}{=} F_X \)

such that \( F_X \) is a formula of the logic (can contain \( X \)).

Semantics?

For every formula \( F \) we define a function \( O_F : 2^{Proc} \to 2^{Proc} \) s.t.

- if \( S \) is the set of processes that satisfy \( X \) then
- \( O_F(S) \) is the set of processes that satisfy \( F \).
Definition of $O_F : 2^{Proc} \rightarrow 2^{Proc}$ (let $S \subseteq 2^{Proc}$)

\[ O_X(S) = S \]
\[ O_{tt}(S) = Proc \]
\[ O_{\#}(S) = \emptyset \]
\[ O_{F_1 \land F_2}(S) = O_{F_1}(S) \cap O_{F_2}(S) \]
\[ O_{F_1 \lor F_2}(S) = O_{F_1}(S) \cup O_{F_2}(S) \]
\[ O_{\langle a \rangle F}(S) = \langle \cdot a \cdot \rangle O_F(S) \]
\[ O_{[a]F}(S) = [\cdot a \cdot] O_F(S) \]

$O_F$ is monotonic for every formula $F$

\[ S_1 \subseteq S_2 \Rightarrow O_F(S_1) \subseteq O_F(S_2) \]

Proof: By structural induction on $F$. 

Luca Aceto 
Bisimilarity and HML
Definition of $O_F : 2^{Proc} \rightarrow 2^{Proc} \ (let \ S \subseteq 2^{Proc})$

\[
\begin{align*}
O_X(S) &= S \\
O_{tt}(S) &= Proc \\
O_{\#}(S) &= \emptyset \\
O_{F_1 \land F_2}(S) &= O_{F_1}(S) \cap O_{F_2}(S) \\
O_{F_1 \lor F_2}(S) &= O_{F_1}(S) \cup O_{F_2}(S) \\
O_{\langle a \rangle F}(S) &= \langle \cdot a \cdot \rangle O_F(S) \\
O_{[a] F}(S) &= [\cdot a \cdot] O_F(S)
\end{align*}
\]

$O_F$ is monotonic for every formula $F$

\[
S_1 \subseteq S_2 \Rightarrow O_F(S_1) \subseteq O_F(S_2)
\]

Proof: By structural induction on $F$. 

Luca Aceto  
Bisimilarity and HML
Observation

We know $O_F$ is monotonic, so $O_F$ has a unique greatest and least fixed point.

Semantics of the Variable $X$

- If $X^{\text{max}} = F_X$ then
  \[
  \llbracket X \rrbracket = \bigcup \{ S \subseteq \text{Proc} \mid S \subseteq O_{F_X}(S) \}.
  \]

- If $X^{\text{min}} = F_X$ then
  \[
  \llbracket X \rrbracket = \bigcap \{ S \subseteq \text{Proc} \mid O_{F_X}(S) \subseteq S \}.
  \]
Selection of Temporal Properties

- **Inv**(*F*): \( X^{\text{max}} = F \wedge [\text{Act}]X \)
- **Pos**(*F*): \( X^{\text{min}} = F \vee \langle \text{Act} \rangle X \)
- **Safe**(*F*): \( X^{\text{max}} = F \wedge ([\text{Act}]ff \vee \langle \text{Act} \rangle X) \)
- **Even**(*F*): \( X^{\text{min}} = F \vee ([\text{Act}]tt \wedge [\text{Act}]X) \)
- **F U^w G**: \( X^{\text{max}} = G \vee (F \wedge [\text{Act}]X) \)
- **F U^s G**: \( X^{\text{min}} = G \vee (F \wedge \langle \text{Act} \rangle tt \wedge [\text{Act}]X) \)

Using until we can express e.g. **Inv**(*F*) and **Even**(*F*):

\[
\text{Inv}(F) \equiv F \mathbin{U^w} ff \\
\text{Even}(F) \equiv tt \mathbin{U^s} F
\]
Selection of Temporal Properties

- $\text{Inv}(F)$: \[ X^{\text{max}} = F \land [\text{Act}]X \]
- $\text{Pos}(F)$: \[ X^{\text{min}} = F \lor \langle \text{Act} \rangle X \]
- $\text{Safe}(F)$: \[ X^{\text{max}} = F \land ([\text{Act}]\mathbf{f} \lor \langle \text{Act} \rangle X) \]
- $\text{Even}(F)$: \[ X^{\text{min}} = F \lor (\langle \text{Act} \rangle \mathbf{t} \land [\text{Act}]X) \]

- $F U^w G$: \[ X^{\text{max}} = G \lor (F \land [\text{Act}]X) \]
- $F U^s G$: \[ X^{\text{min}} = G \lor (F \land \langle \text{Act} \rangle \mathbf{t} \land [\text{Act}]X) \]

Using until we can express e.g. $\text{Inv}(F)$ and $\text{Even}(F)$:

$\text{Inv}(F) \equiv F U^w \mathbf{f}$ \hspace{1cm} $\text{Even}(F) \equiv \mathbf{t} U^s F$
Selection of Temporal Properties

- **Inv**($F$): $X^{\text{max}} = F \land [\text{Act}]X$
- **Pos**($F$): $X^{\text{min}} = F \lor \langle \text{Act} \rangle X$

- **Safe**($F$): $X^{\text{max}} = F \land ([\text{Act}]\text{ff} \lor \langle \text{Act} \rangle X)$
- **Even**($F$): $X^{\text{min}} = F \lor (\langle \text{Act} \rangle \text{tt} \land [\text{Act}]X)$

- $F U^w G$: $X^{\text{max}} = G \lor (F \land [\text{Act}]X)$
- $F U^s G$: $X^{\text{min}} = G \lor (F \land \langle \text{Act} \rangle \text{tt} \land [\text{Act}]X)$

Using until we can express e.g. **Inv**($F$) and **Even**($F$):

$$\text{Inv}(F) \equiv F U^w \text{ff} \quad \text{Even}(F) \equiv \text{tt} U^s F$$
Selection of Temporal Properties

- \( \text{Inv}(F) \): \[ X \overset{\text{max}}{=} F \land [\text{Act}]X \]
- \( \text{Pos}(F) \): \[ X \overset{\text{min}}{=} F \lor \langle \text{Act} \rangle X \]
- \( \text{Safe}(F) \): \[ X \overset{\text{max}}{=} F \land ([\text{Act}]\text{ff} \lor \langle \text{Act} \rangle X) \]
- \( \text{Even}(F) \): \[ X \overset{\text{min}}{=} F \lor (\langle \text{Act} \rangle \text{tt} \land [\text{Act}]X) \]
- \( F \bigcup^w G \): \[ X \overset{\text{max}}{=} G \lor (F \land [\text{Act}]X) \]
- \( F \bigcup^s G \): \[ X \overset{\text{min}}{=} G \lor (F \land \langle \text{Act} \rangle \text{tt} \land [\text{Act}]X) \]

Using until we can express e.g. \( \text{Inv}(F) \) and \( \text{Even}(F) \):

\[ \text{Inv}(F) \equiv F \bigcup^w \text{ff} \quad \text{Even}(F) \equiv \text{tt} \bigcup^s F \]