Hennessy-Milner logic
syntax and semantics
correspondence with strong bisimilarity
examples in CWB

MH 2008: one slide changed
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
Verifying Correctness of Reactive Systems

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.
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 ($a \in Act$)

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

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 ($a \in \text{Act}$)

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

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 \((a \in \text{Act})\)

\[ F, G ::= \top \mid \bot \mid F \land G \mid F \lor G \mid \langle a \rangle F \mid [a]F \]

Intuition:
- \(\top\): all processes satisfy this property
- \(\bot\): 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)

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

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

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

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

**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 $\llbracket F \rrbracket \subseteq \text{Proc}$ contain all states that satisfy $F$.

Denotational Semantics: $\llbracket - \rrbracket : \text{Formulae} \rightarrow 2^{\text{Proc}}$

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

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

\[
\langle \cdot a \cdot \rangle S = \{ p \in \text{Proc} \mid \exists p'. \ p \xrightarrow{a} p' \text{ and } p' \in S \}
\]

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

**Theorem**

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

\[ p \models F \quad \text{if and only if} \quad p \in [F]. \]

Proof: by structural induction on the structure of the formula \(F\).
The Correspondence Theorem

**Theorem**

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

\[ p \models F \text{ if and only if } p \in \llbracket F \rrbracket. \]

**Proof:** by structural induction on the structure of the formula \(F\).
Image-Finite System

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

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

is finite.
Theorem (Hennessy-Milner)

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

\[ p \sim q \]

if and only if

for every HM formula \( F \): \( (p \models F \iff q \models F) \).
Hennessy-Milner Logic

Correspondence between HM Logic and Strong Bisimilarity

Image-Finite Labelled Transition Systems

Hennessy-Milner Theorem

Example Sessions in CWB

CWB Session

```plaintext
proc S = a.S1
proc S1 = b.nil + c.nil

proc T = a.T1 + a.T2
proc T1 = b.nil
proc T2 = c.nil

prop both = <a>(<b>tt \ /
\ <c>tt)```

```
cwb-nc> chk S both
...TRUE ...
cwb-nc>

cwb-nc> chk T both
...FALSE ...
cwb-nc>

cwb-nc> chk -S bisim S T
false

cwb-nc> S satisfies [a]<b>tt

cwb-nc> T does not

cwb-nc>
```

Lecture 5  Semantics and Verification 2007