Hennessy-Milner logic
Syntax and semantics
Correspondence with strong bisimilarity
Examples in CWB
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.
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 \((a \in \text{Act})\)

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

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

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

Intuitions:

- \text{tt} all processes satisfy this property
- \text{ff} no process satisfies this property
- \land, \lor \text{ usual logical AND and OR}
- \langle a \rangle F \text{ there is at least one } a\text{-successor that satisfies } F
- [a]F \text{ all } a\text{-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 \((\mathit{Proc}, \mathit{Act}, \{\overset{a}{\rightarrow} | a \in \mathit{Act}\})\) be an LTS.

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

- \(p \models \mathit{tt}\) for each \(p \in \mathit{Proc}\)
- \(p \models \mathit{ff}\) for no \(p\) (we also write \(p \not\models \mathit{ff}\))
- \(p \models F \land G\) 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 \(p \overset{a}{\rightarrow} p'\) for some \(p' \in \mathit{Proc}\) such that \(p' \models F\)
- \(p \models [a]F\) iff \(p' \models F\), for all \(p' \in \mathit{Proc}\) such that \(p \overset{a}{\rightarrow} p'\)

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. $p \not\models F \iff p \models F^c$
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. $p \not\models F \implies p \models F^c$
For a formula $F$ let $\llbracket F \rrbracket \subseteq \mathit{Proc}$ contain all states that satisfy $F$.

### Denotational Semantics: $\llbracket \cdot \rrbracket : \mathit{Formulae} \rightarrow 2^{\mathit{Proc}}$

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

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

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

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

**Theorem**

Let \((\text{Proc}, \text{Act}, \{\xrightarrow{a} | a \in \text{Act}\})\) 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 \(F\).
The Correspondence Theorem

Theorem

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

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

Proof: by structural induction on the structure of the formula \(F\).
Let \((Proc, Act, \{\xrightarrow{a} | a \in Act\})\) be an LTS. We call it image-finite iff for every \(p \in Proc\) and every \(a \in Act\) the set

\[\{p' \in Proc | p \xrightarrow{a} p'\}\]

is finite.
Theorem (Hennessy-Milner)

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

\[ p \sim q \]

if and only if

for every HM formula \(F\):

\[ p \models F \iff q \models F. \]
CWB Session

hm.cwb

agent S = a.S1;
agent S1 = b.0 + c.0;
agent T = a.T1 + a.T2;
agent T1 = b.0;
agent T2 = c.0;

[luca@vel5638 CWB]$
./xccscwb.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;

EWSCS'07–Lecture 6
Reactive Systems: Modelling, Specification and Verification