 Hennessy-Milner logic
syntax and semantics
 correspondence with strong bisimilarity
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?
Syntax of the Formulae \((a \in Act)\)

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

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 \underline{always}/

\underline{never in the future} or \underline{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:

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

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

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

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

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

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

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 if for every $p \in Proc$ and every $a \in Act$ the set 

$$\{p' \in Proc \mid p \xrightarrow{a} p'\}$$

is finite.
Theorem (Hennessy-Milner)

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

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