Formale Spezifikation und Verifikation

Mirco Tribastone

Institut für Informatik
Ludwig-Maximilians-Universität München
tribastone@pst.ifi.lmu.de

Hennessy-Milner Logic

Verifying Correctness of Reactive Systems

Let *Impl* be an implementation of a system (e.g., in CCS syntax).

Equivalence Checking Approach

$$\mathit{Impl} \equiv \mathit{Spec}$$

- $\equiv$ 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
- Implementation verification requires the full description of both models…
- …and the derivation of the respective state spaces.
- Some specifications may seem unnatural…

Model Checking Approach

Model Checking Approach

$$\mathit{Impl} \models \mathit{Property}$$

- $\models$ 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?

Hennessy-Milner Logic — Syntax

Syntax ($a \in Act$)

$$F, G ::= tt \mid ff \mid F \wedge G \mid F \vee G \mid \langle a \rangle F \mid [a]F$$

- tt all processes satisfy this property
- ff no process satisfies this property
- $\wedge$, $\vee$ usual logical AND and OR connectives
- $\langle a \rangle F$ (possibility) asserts (of a given P): It is possible for P to perform an action a and evolve into a Q that satisfies F — there is at least one a-successor that satisfies F
- $[a]F$ (necessity) asserts (of a given P): If P can perform an action a then it must evolve into a Q that satisfies F — all a-successors have to satisfy F

Hennessy-Milner Logic — Semantics

Let $(Proc, Act, \{\xrightarrow{a} \mid a \in Act\})$ be an LTS.

Satisfaction relation $p \models F$ ($p \in Proc$, F a HM formula)

$$
\begin{aligned}
p &\models tt \quad \text{for each } p \in Proc \\
p &\models ff \quad \text{for no } p \text{ (we also write } p \not\models ff\text{)} \\
p &\models F \wedge G \quad \text{iff } p \models F \text{ and } p \models G \\
p &\models F \vee G \quad \text{iff } p \models F \text{ or } p \models G \\
p &\models \langle a \rangle F \quad \text{iff } p \xrightarrow{a} p' \text{ for some } p' \in Proc \text{ such that } p' \models F \\
p &\models [a]F \quad \text{iff } p' \models F, \text{ for all } p' \in Proc \text{ such that } p \xrightarrow{a} p'
\end{aligned}
$$

We write:

- $p \not\models F$ if p does not satisfy F
- $\langle \{a_1, a_2, \ldots a_n\} \rangle F$ for $\langle a_1 \rangle F \vee \langle a_2 \rangle F \cdots \vee \langle a_n \rangle F$
- $[\{a_1, a_2, \ldots a_n\}]F$ for $[a_1]F \wedge [a_2]F \cdots \wedge [a_n]F$

An Alternative (and Equivalent) Characterisation

Let $(Proc, Act, \{\xrightarrow{a} \mid a \in Act\})$ be an LTS.

Denotational Semantics

Let $[\![F]\!] \in Proc$, with F an HM formula, be defined by

$$[\![tt]\!] = Proc \;, \qquad [\![F \vee G]\!] = [\![F]\!] \cup [\![G]\!] \;,$$

$$[\![ff]\!] = \emptyset \;, \qquad [\![\langle a \rangle F]\!] = \langle \cdot a \cdot \rangle [\![F]\!] \;,$$

$$[\![F \wedge G]\!] = [\![F]\!] \cap [\![G]\!] \;, \qquad [\![[a]F]\!] = [\cdot a \cdot][\![F]\!] \;,$$

where the operators $\langle \cdot a \cdot \rangle, [\cdot a \cdot] : 2^{Proc} \longrightarrow 2^{Proc}$ are defined by

$$\langle \cdot a \cdot \rangle S = \{p \in Proc \mid p \xrightarrow{a} p' \text{ and } p' \in S \text{ for some } p'\},$$

$$[\cdot a \cdot] S = \{p \in Proc \mid p \xrightarrow{a} p' \text{ implies } p' \in S \text{ for each } p'\}.$$

We write $p \models F$ iff $p \in [\![F]\!]$.

Examples

- $E \models \langle \mathit{tick} \rangle \mathit{tt}$
  E can do a tick
- $E \models \langle \mathit{tick} \rangle \langle \mathit{tock} \rangle \mathit{tt}$
  E can do a tick and then a tock
- $E \models \langle \{\mathit{tick}, \mathit{tock}\} \rangle \mathit{tt}$
  E can do a tick or a tock
- $E \models [\mathit{tick}] \mathit{ff}$
  E cannot do a tick
- $E \models \langle \mathit{tick} \rangle \mathit{ff}$
  This is equivalent to false!
- $E \models [\mathit{tick}] \mathit{tt}$
  This is equivalent to true!

Checking Satisfaction

$$C \triangleq tick.C$$

Does C satisfy property $[tick](\langle tick\rangle tt \wedge [tock]ff)$?

Logical Negation

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

- $tt^c = ff$
- $ff^c = tt$
- $(F \wedge G)^c = F^c \vee G^c$
- $(F \vee G)^c = F^c \wedge G^c$
- $(\langle a \rangle F)^c = [a]F^c$. For instance $(\langle a \rangle tt)^c = [a]ff$
- $([a]F)^c = \langle a \rangle F^c$. For instance $([a]ff)^c = \langle a \rangle tt$

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

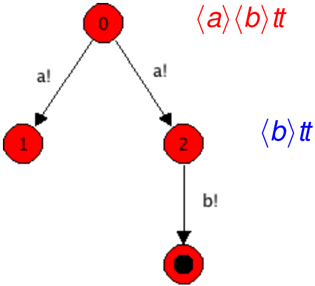
For any $p \in Proc$ and any HML formula F

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

Checking Validity of HML Formulae

1. Decompose the HML formula into all its subformulas
2. Starting with the smallest subformula, label all states of the LTS where it holds
3. Repeat the previous step for the smallest remaining formula
4. If the state is labeled with the formula to be checked the formula is valid that state, otherwise, it is invalid.

Does the transition system corresponding to $a.\mathbf{0} + a.b.\mathbf{0}$ satisfy the formula $\langle a \rangle \langle b \rangle tt$

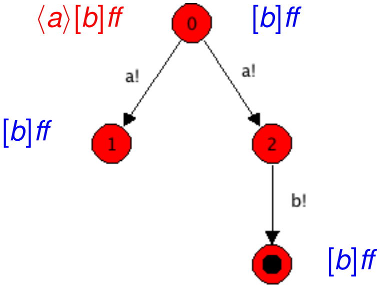


Subformulae of $\langle a \rangle \langle b \rangle tt$:

$tt \qquad \langle b \rangle tt \qquad \langle a \rangle \langle b \rangle tt$

Examples of Model Checking

Does the transition system corresponding to $a.\mathbf{0} + a.b.\mathbf{0}$ satisfy formula $\langle a \rangle [b] \mathit{ff}$

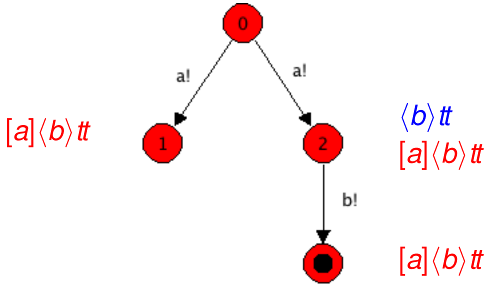


Subformulae of $\langle a \rangle [b] \mathit{ff}$:

$$\mathit{ff} \qquad [b]\mathit{ff} \qquad \langle a \rangle [b] \mathit{ff}$$

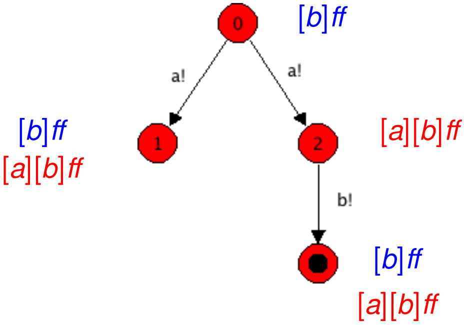
Examples of Model Checking

Does the transition system corresponding to $a.\mathbf{0} + a.b.\mathbf{0}$ satisfy formula $[a]\langle b\rangle tt$



Examples of Model Checking

Does the transition system corresponding to $a.\mathbf{0} + a.b.\mathbf{0}$ satisfy formula $[a][b]ff$



HML and Bisimulation

Examples

- $a.(b.\mathbf{0} + c.\mathbf{0}) \models \langle a \rangle(\langle b \rangle tt \wedge \langle c \rangle tt)$
- $a.b.\mathbf{0} + a.c.\mathbf{0} \not\models \langle a \rangle(\langle b \rangle tt \wedge \langle c \rangle tt)$

- $a.b.\mathbf{0} \models [a]\langle b \rangle tt$
- $a.b.\mathbf{0} + a.\mathbf{0} \not\models [a]\langle b \rangle tt$

- $a.b.(c.\mathbf{0} + d.\mathbf{0}) \models [a]\langle b \rangle \langle c \rangle tt$
- $a.b.c.\mathbf{0} + a.b.d.\mathbf{0} \not\models [a]\langle b \rangle \langle c \rangle tt$

- $a.(b.c.\mathbf{0} + b.d.\mathbf{0}) \models [a](\langle b \rangle \langle c \rangle tt \wedge \langle b \rangle \langle d \rangle tt)$
- $a.b.c.\mathbf{0} + a.b.d.\mathbf{0} \not\models [a](\langle b \rangle \langle c \rangle tt \wedge \langle b \rangle \langle d \rangle tt)$

HML and Bisimulation

Theorem

$P \sim Q$ if and only $P \models F$ if and only if $Q \models F$ for every HML formula F.

Proof

($\Longrightarrow$) Proceeds by induction on F. The interesting case is $[a]F$.
($\Longleftarrow$) We show that the set $\mathcal{S}$ of all pair of processes that satisfy the same HML formulae is a bisimulation. Suppose $\mathcal{S}$ is not a bisimulation. Then, there exists a pair $< P, Q > \in \mathcal{S}$ such that Q cannot match a move $P \xrightarrow{a} P'$. There are two cases.
Case 1: Q does not have a transition $Q \xrightarrow{a} Q'$, but then clearly P and Q do not satisfy the same formulae.
Case 2: for every evolution of $Q \xrightarrow{a} Q'$, Q' and P' do not satisfy the same formulae. Then, it is possible to construct a formula (of the form $\langle a \rangle F$ with $F = F_1 \wedge \ldots \wedge F_n$) that P satisfies but Q does not.