Reactive Systems: Modelling, Specification and Verification

EWSCS'07–Lecture 7

- Syntax and semantics of Hennessy-Milner logic (reprise)
- Correspondence with strong bisimilarity
- Examples in CWB
- Hennessy-Milner logic and temporal properties

Hennessy-Milner Logic – Syntax

Syntax of the Formulae ($a \in Act$)

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

Intuition:

- tt all processes satisfy this property
- ff no process satisfies this property
- $\wedge$, $\vee$ 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

Hennessy-Milner Logic – Semantics

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

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

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

We write $p \not\models F$ whenever p does not satisfy F.

What about Negation?

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

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

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:

- $tt^c = f\!f$
- $f\!f^c = tt$
- $(F \wedge G)^c = F^c \vee G^c$
- $(F \vee G)^c = F^c \wedge G^c$
- $(\langle a \rangle F)^c = [a]F^c$
- $([a]F)^c = \langle a \rangle F^c$

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

Idea: $[\![F]\!]$ is the set of all states that satisfy F

- $[\![tt]\!] = Proc$
- $[\![ff]\!] = \emptyset$
- $[\![F \vee G]\!] = [\![F]\!] \cup [\![G]\!]$
- $[\![F \wedge G]\!] = [\![F]\!] \cap [\![G]\!]$
- $[\![\langle a \rangle F]\!] = \langle \cdot a \cdot \rangle [\![F]\!]$
- $[\![[a]F]\!] = [\cdot a \cdot][\![F]\!]$

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

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

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

The Correspondence Theorem

Theorem

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

$$p \models F \quad \textit{if and only if} \quad p \in [\![F]\!].$$

Proof: by structural induction on the structure of the formula F.

The Correspondence Theorem

Theorem

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

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

Proof: by structural induction on the structure of the formula F.

Image-Finite Labelled Transition System

Image-Finite System

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

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

is finite.

Relationship between HM Logic and Strong Bisimilarity

Theorem (Hennessy-Milner)

Let $(Proc, Act, \{\xrightarrow{a} \mid a \in Act\})$ 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;
```

Is Hennessy-Milner Logic Powerful Enough?

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

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

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:

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

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:

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

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

$s \models Inv(F)$ iff all states reachable from s satisfy F
$s \models Pos(F)$ 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

- $\langle Act\rangle F \stackrel{\mathrm{def}}{=} \langle a_1\rangle F \vee \langle a_2\rangle F \vee \ldots \vee \langle a_n\rangle F$
- $[Act]F \stackrel{\mathrm{def}}{=} [a_1]F \wedge [a_2]F \wedge \ldots \wedge [a_n]F$

$Inv(F) \equiv F \wedge [Act]F \wedge [Act][Act]F \wedge [Act][Act][Act]F \wedge \ldots$
$Pos(F) \equiv F \vee \langle Act\rangle F \vee \langle Act\rangle\langle Act\rangle F \vee \langle Act\rangle\langle Act\rangle\langle Act\rangle F \vee \ldots$

Temporal Properties not Expressible in HM Logic

$s \models Inv(F)$ iff all states reachable from s satisfy F
$s \models Pos(F)$ 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

- $\langle Act \rangle F \stackrel{\text{def}}{=} \langle a_1 \rangle F \vee \langle a_2 \rangle F \vee \ldots \vee \langle a_n \rangle F$
- $[Act]F \stackrel{\text{def}}{=} [a_1]F \wedge [a_2]F \wedge \ldots \wedge [a_n]F$

$Inv(F) \equiv F \wedge [Act]F \wedge [Act][Act]F \wedge [Act][Act][Act]F \wedge \ldots$
$Pos(F) \equiv F \vee \langle Act \rangle F \vee \langle Act \rangle \langle Act \rangle F \vee \langle Act \rangle \langle Act \rangle \langle Act \rangle F \vee \ldots$

Temporal Properties not Expressible in HM Logic

$s \models Inv(F)$ iff all states reachable from s satisfy F
$s \models Pos(F)$ 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

- $\langle Act \rangle F \stackrel{\text{def}}{=} \langle a_1 \rangle F \vee \langle a_2 \rangle F \vee \ldots \vee \langle a_n \rangle F$
- $[Act]F \stackrel{\text{def}}{=} [a_1]F \wedge [a_2]F \wedge \ldots \wedge [a_n]F$

$Inv(F) \equiv F \wedge [Act]F \wedge [Act][Act]F \wedge [Act][Act][Act]F \wedge \ldots$
$Pos(F) \equiv F \vee \langle Act \rangle F \vee \langle Act \rangle\langle Act \rangle F \vee \langle Act \rangle\langle Act \rangle\langle Act \rangle F \vee \ldots$

Problems

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

Why not to use recursion?

- $Inv(F)$ expressed by $X \stackrel{\mathrm{def}}{=} F \wedge [Act]X$
- $Pos(F)$ expressed by $X \stackrel{\mathrm{def}}{=} F \vee \langle Act \rangle X$

Question: How to define the semantics of such equations?

Problems

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

Why not to use recursion?

- $Inv(F)$ expressed by $X \stackrel{\text{def}}{=} F \wedge [Act]X$
- $Pos(F)$ expressed by $X \stackrel{\text{def}}{=} F \vee \langle Act\rangle X$

Question: How to define the semantics of such equations?

Problems

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

Why not to use recursion?

- $Inv(F)$ expressed by $X \stackrel{\text{def}}{=} F \wedge [Act]X$
- $Pos(F)$ expressed by $X \stackrel{\text{def}}{=} F \vee \langle Act \rangle X$

Question: How to define the semantics of such equations?

Solving Equations is Tricky

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

$n = 2 * n$	one solution $n = 0$
$n = n + 1$	no solution
$n = 1 * n$	many solutions (every $n \in Nat$ is a solution)

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

$M = \{7\} \cap M$	one solution $M = \{7\}$
$M = \mathbb{N} \smallsetminus M$	no solution
$M = \{3\} \cup M$	many solutions (every $M \supseteq \{3\}$ is a solution)

What about Equations over Processes?

$X \stackrel{\text{def}}{=} [a]ff \vee \langle a \rangle X \quad \Rightarrow \quad$ find $S \subseteq 2^{Proc}$ s.t. $S = [\cdot a \cdot]\emptyset \cup \langle \cdot a \cdot \rangle S$

Solving Equations is Tricky

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

$n = 2 * n$	one solution $n = 0$
$n = n + 1$	no solution
$n = 1 * n$	many solutions (every $n \in Nat$ is a solution)

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

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

What about Equations over Processes?

$$X \stackrel{\text{def}}{=} [a]f\!f \vee \langle a \rangle X \quad \Rightarrow \quad \text{find } S \subseteq 2^{Proc} \text{ s.t. } S = [\cdot a \cdot]\emptyset \cup \langle \cdot a \cdot \rangle S$$

Solving Equations is Tricky

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

$n = 2 * n$ one solution $n = 0$

$n = n + 1$ no solution

$n = 1 * n$ many solutions (every $n \in Nat$ is a solution)

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

$M = \{7\} \cap M$ one solution $M = \{7\}$

$M = \mathbb{N} \setminus M$ no solution

$M = \{3\} \cup M$ many solutions (every $M \supseteq \{3\}$ is a solution)

What about Equations over Processes?

$$X \stackrel{\text{def}}{=} [a]ff \vee \langle a \rangle X \quad \Rightarrow \quad \text{find } S \subseteq 2^{Proc} \text{ s.t. } S = [\cdot a \cdot]\emptyset \cup \langle \cdot a \cdot \rangle S$$