

Concurrency Theory

Lecture 4: Hennessy-Milner Logic with Recursion

Joost-Pieter Katoen Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)



{katoen,noll}@cs.rwth-aachen.de

<http://www-i2.informatik.rwth-aachen.de/i2/ct13/>

Winter Semester 2013/14

Next week:

Tue 5 Nov 12:15-13:45 AH 6 ("Fachschafts-VV") →

Thu 7 Nov 14:15-15:45 AH 1

- 1 Recap: Hennessy-Milner Logic
- 2 HML and Process Traces
- 3 Adding Recursion to HML
- 4 HML with One Recursive Variable

Definition (Syntax of HML)

The set *HMF* of **H**ennessy-**M**ilner **f**ormulae over a set of actions *Act* is defined by the following syntax:

$F ::=$	tt	(true)
	ff	(false)
	$F_1 \wedge F_2$	(conjunction)
	$F_1 \vee F_2$	(disjunction)
	$\langle \alpha \rangle F$	(diamond)
	$[\alpha] F$	(box)

where $\alpha \in \text{Act}$.

Abbreviations for $L = \{\alpha_1, \dots, \alpha_n\}$ ($n \in \mathbb{N}$):

- $\langle L \rangle F := \langle \alpha_1 \rangle F \vee \dots \vee \langle \alpha_n \rangle F$
- $[L] F := [\alpha_1] F \wedge \dots \wedge [\alpha_n] F$
- In particular, $\langle \emptyset \rangle F := \text{ff}$ and $[\emptyset] F := \text{tt}$

Definition (Semantics of HML)

Let $(S, Act, \longrightarrow)$ be an LTS and $F \in HMF$. The set of processes in S that **satisfy** F , $\llbracket F \rrbracket \subseteq S$, is defined by

$$\begin{array}{ll} \llbracket \text{tt} \rrbracket := S & \llbracket \text{ff} \rrbracket := \emptyset \\ \llbracket F_1 \wedge F_2 \rrbracket := \llbracket F_1 \rrbracket \cap \llbracket F_2 \rrbracket & \llbracket F_1 \vee F_2 \rrbracket := \llbracket F_1 \rrbracket \cup \llbracket F_2 \rrbracket \\ \llbracket \langle \alpha \rangle F \rrbracket := \langle \cdot \alpha \cdot \rangle (\llbracket F \rrbracket) & \llbracket [\alpha] F \rrbracket := [\cdot \alpha \cdot] (\llbracket F \rrbracket) \end{array}$$

where $\langle \cdot \alpha \cdot \rangle, [\cdot \alpha \cdot] : 2^S \rightarrow 2^S$ are given by

$$\begin{array}{l} \langle \cdot \alpha \cdot \rangle (T) := \{s \in S \mid \exists s' \in T : s \xrightarrow{\alpha} s'\} \\ [\cdot \alpha \cdot] (T) := \{s \in S \mid \forall s' \in S : s \xrightarrow{\alpha} s' \implies s' \in T\} \end{array}$$

We write $s \models F$ iff $s \in \llbracket F \rrbracket$. Two HML formulae are **equivalent** (written $F \equiv G$) iff they are satisfied by the same processes in every LTS.

Goal: reduce processes to the action sequences they can perform

Definition (Trace language)

For every $P \in \text{Prc}$, let

$$\text{Tr}(P) := \{w \in \text{Act}^* \mid \text{ex. } P' \in \text{Prc} \text{ such that } P \xrightarrow{w} P'\}$$

be the **trace language** of P

(where $\xrightarrow{w} := \xrightarrow{a_1} \circ \dots \circ \xrightarrow{a_n}$ for $w = a_1 \dots a_n$).

$P, Q \in \text{Prc}$ are called **trace equivalent** if $\text{Tr}(P) = \text{Tr}(Q)$.

Example (One-place buffer)

$$B = in.\overline{out}.B$$

$$\implies \text{Tr}(B) = (in \cdot \overline{out})^* \cdot (in + \epsilon)$$

- 1 Recap: Hennessy-Milner Logic
- 2 HML and Process Traces
- 3 Adding Recursion to HML
- 4 HML with One Recursive Variable

Lemma 4.1

Let $(Prc, Act, \longrightarrow)$ be an LTS, and let $P, Q \in Prc$ satisfy the same HMF (i.e., $\forall F \in HMF : P \models F \iff Q \models F$). Then $Tr(P) = Tr(Q)$.

Proof.

on the board □

Remark: the converse does *not* hold.

Example 4.2

- Let $P := a.(b.nil + c.nil) \in Prc$, $Q := a.b.nil + a.c.nil \in Prc$
- Then $Tr(P) = Tr(Q) = \{\varepsilon, a, ab, ac\}$
- Let $F := [a](\langle b \rangle tt \wedge \langle c \rangle tt) \in HMF$
- Then $P \models F$ but $Q \not\models F$
- [later: $P, Q \in Prc$ HML-equivalent iff bismilar]

- 1 Recap: Hennessy-Milner Logic
- 2 HML and Process Traces
- 3 Adding Recursion to HML
- 4 HML with One Recursive Variable

Observation: HML formulae only describe **finite** part of process behaviour

- each modal operator ($[.]$, $\langle . \rangle$) talks about *one* step
- only finite nesting of operators (**modal depth**)

Example 4.3

- $F := (\langle a \rangle [a] \text{ff}) \vee \langle b \rangle \text{tt} \in \text{HMF}$ has modal depth 2
- Checking F involves analysis of all traces of length ≤ 2

But: sometimes necessary to refer to **arbitrarily long computations** (e.g., “no deadlock state reachable”)

- possible solution: support **infinite conjunctions and disjunctions**

Example 4.4

- Let $C = a.C$, $D = a.D + a.nil$
- Then $C \models [a]\langle a \rangle tt$ but $D \not\models [a]\langle a \rangle tt$
- Now redefine D as $D_n = a.D_n + a.E_n$ where $n \in \mathbb{N}$, $E_k = a.E_{k-1}$ ($1 \leq k \leq n$), $E_0 = nil$
- Then (for $[\alpha]^k F := \underbrace{[\alpha] \dots [\alpha]}_{k \text{ times}} F$):
 - $C \models [a]^k \langle a \rangle tt$ for all $k \in \mathbb{N}$
 - $D_n \models [a]^k \langle a \rangle tt$ for all $0 \leq k \leq n$
 - $D_n \not\models [a]^k \langle a \rangle tt$ for all $k > n$
- Conclusion: no HML formula can distinguish C and all D_n
- Generally: **invariance** property “always $\langle a \rangle tt$ ” not expressible
- Requires **infinite conjunction**:

$$Inv(\langle a \rangle tt) = \langle a \rangle tt \wedge [a]\langle a \rangle tt \wedge [a][a]\langle a \rangle tt \wedge \dots = \bigwedge_{k \in \mathbb{N}} [a]^k \langle a \rangle tt$$

Dually: **possibility** properties expressible by infinite disjunctions

Example 4.5

- Let $C = a.C$, $D = a.D + a.nil$ as before
- C has no possibility to terminate
- D has the option to terminate (i.e., to eventually satisfy $[a]ff$) at any time by choosing the $a.nil$ branch
- Representable by **infinite disjunction**:

$$Pos([a]ff) = [a]ff \vee \langle a \rangle [a]ff \vee \langle a \rangle \langle a \rangle [a]ff \vee \dots = \bigvee_{k \in \mathbb{N}} \langle a \rangle^k [a]ff$$

Problem: infinite formulae not easy to handle

Solution: employ **recursion**!

- $Inv(\langle a \rangle tt) \equiv \langle a \rangle tt \wedge [a] \text{Inv}(\langle a \rangle tt)$
- $Pos([a] ff) \equiv [a] ff \vee \langle a \rangle Pos([a] ff)$

Interpretation: the sets of states $X, Y \subseteq S$ satisfying the respective formula should solve the corresponding equation, i.e.,

- $X = \langle \cdot a \cdot \rangle(S) \cap [\cdot a \cdot](X)$
- $Y = [\cdot a \cdot](\emptyset) \cup \langle \cdot a \cdot \rangle(Y)$

Open questions

- Do such recursive equations (always) have solutions?
- If so, are they unique?
- How can we compute whether a process satisfies a recursive formula?

Example 4.6

- Consider again $C = a.C$, $D = a.D + a.nil$
- Invariant: $X \equiv \langle a \rangle tt \wedge [a]X$
 - $X = \emptyset$ is a solution (as no process can satisfy both $\langle a \rangle tt$ and $[a]ff$)
 - but we expect $C \in X$ (as C can perform a invariantly)
 - in fact, $X = \{C\}$ also solves the equation (and is the **greatest solution** w.r.t. $\subseteq$) $\implies$ write $X \stackrel{max}{=} \langle a \rangle tt \wedge [a]X$
- Possibility: $Y \equiv [a]ff \vee \langle a \rangle Y$
 - greatest solution: $Y = \{C, D, nil\}$
 - but we expect $C \notin Y$ (as C cannot terminate at all)
 - here: **least solution** w.r.t. $\subseteq$: $Y = \{D, nil\}$ $\implies$ write $Y \stackrel{min}{=} [a]ff \vee \langle a \rangle Y$

Uniqueness of Solutions

Uniqueness of solutions

- Use **greatest solutions** for properties that hold unless the process has a finite computation that **disproves** it.
- Use **least solutions** for properties that hold if the process has a finite computation that **proves** it.

Example 4.7

Let $(S, Act, \longrightarrow)$ be an LTS, $s \in S$, and $F \in HMF$.

- **Invariant:** $Inv(F) \equiv X$ for $X \stackrel{max}{=} F \wedge [Act]X$
 - $s \models Inv(F)$ if all states reachable from s satisfy F
- **Possibility:** $Pos(F) \equiv Y$ for $Y \stackrel{min}{=} F \vee \langle Act \rangle Y$
 - $s \models Pos(F)$ if a state satisfying F is reachable from s
- **Safety:** $Safe(F) \equiv X$ for $X \stackrel{max}{=} F \wedge ([Act]ff \vee \langle Act \rangle X)$
 - $s \models Safe(F)$ if s has a complete (i.e., infinite or terminating) transition sequence where each state satisfies F
- **Eventuality:** $Evt(F) \equiv Y$ for $Y \stackrel{min}{=} F \vee (\langle Act \rangle tt \wedge [Act] Y)$
 - $s \models Evt(F)$ if each complete transition sequence starting in s contains a state satisfying F

- 1 Recap: Hennessy-Milner Logic
- 2 HML and Process Traces
- 3 Adding Recursion to HML
- 4 HML with One Recursive Variable

Syntax of HML with One Recursive Variable

Initially: only one variable

Later: mutual recursion

Definition 4.8 (Syntax of HML with one variable)

The set HMF_X of Hennessy-Milner formulae with one variable X over a set of actions Act is defined by the following syntax:

$F ::= X$	(variable)
tt	(true)
ff	(false)
$F_1 \wedge F_2$	(conjunction)
$F_1 \vee F_2$	(disjunction)
$\langle \alpha \rangle F$	(diamond)
$[\alpha] F$	(box)

where $\alpha \in Act$.

Semantics of HML with One Recursive Variable I

So far: $\llbracket F \rrbracket \subseteq S$ for $F \in \text{HMF}$ and LTS $(S, \text{Act}, \longrightarrow)$

Now: semantics of formula depends on states that (are assumed to) satisfy X

Definition 4.9 (Semantics of HML)

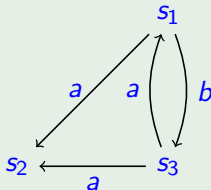
Let $(S, \text{Act}, \longrightarrow)$ be an LTS and $F \in \text{HMF}_X$. The semantics of F ,

$$\llbracket F \rrbracket : 2^S \rightarrow 2^S,$$

is defined by

$$\begin{aligned}\llbracket X \rrbracket(T) &:= T \\ \llbracket \text{tt} \rrbracket(T) &:= S \\ \llbracket \text{ff} \rrbracket(T) &:= \emptyset \\ \llbracket F_1 \wedge F_2 \rrbracket(T) &:= \llbracket F_1 \rrbracket(T) \cap \llbracket F_2 \rrbracket(T) \\ \llbracket F_1 \vee F_2 \rrbracket(T) &:= \llbracket F_1 \rrbracket(T) \cup \llbracket F_2 \rrbracket(T) \\ \llbracket \langle \alpha \rangle F \rrbracket(T) &:= \langle \cdot \alpha \cdot \rangle(\llbracket F \rrbracket(T)) \\ \llbracket [\alpha] F \rrbracket(T) &:= [\cdot \alpha \cdot](\llbracket F \rrbracket(T))\end{aligned}$$

Example 4.10



Let $S := \{s_1, s_2, s_3\}$.

- $\llbracket \langle a \rangle X \rrbracket (\{s_1\}) = \{s_3\}$
- $\llbracket \langle a \rangle X \rrbracket (\{s_1, s_2\}) = \{s_1, s_3\}$
- $\llbracket [b] X \rrbracket (\{s_2\}) = \{s_2, s_3\}$

Semantics of HML with One Recursive Variable III

- Idea underlying the definition of

$$\llbracket . \rrbracket : HMF_X \rightarrow (2^S \rightarrow 2^S) :$$

if $T \subseteq S$ gives the set of states that satisfy X , then $\llbracket F \rrbracket(T)$ will be the set of states that satisfy F

- How to determine this T ?
- According to previous discussion: as solution of **recursive equation** of the form $X = F_X$ where $F_X \in HMF_X$
- But: solution **not unique**; therefore write:

$$X \stackrel{\min}{=} F_X \quad \text{or} \quad X \stackrel{\max}{=} F_X$$

- In the following we will see:
 - Equation $X = F_X$ always **solvable**
 - Least and greatest solutions are **unique** and can be obtained by **fixed-point iteration**