

Concurrency Theory

HML and strong bisimilarity

Joost-Pieter Katoen and Thomas Noll

Lehrstuhl für Informatik 2
Software Modeling and Verification Group

<http://moves.rwth-aachen.de/teaching/ws-1516/ct>

January 18, 2016

Overview

- 1 Aims of this lecture
- 2 Introduction
- 3 Hennessy-Milner logic
- 4 Correspondence HML and strong bisimilarity
- 5 Characteristic properties
- 6 Summary

Overview

- 1 Aims of this lecture
- 2 Introduction
- 3 Hennessy-Milner logic
- 4 Correspondence HML and strong bisimilarity
- 5 Characteristic properties
- 6 Summary

Summary so far

- ▶ Weak and strong bisimilarity are based on mutually mimicking of processes.
- ▶ They possess the required properties of behavioural equivalences.¹
- ▶ In particular, \sim and \approx^c are deadlock sensitive.
- ▶ Hennessy-Milner logic is a logic for expressing properties of processes.

¹For weak bisimilarity the notion of observation congruence was needed.

Summary so far

- ▶ Weak and strong bisimilarity are based on mutually mimicking of processes.
- ▶ They possess the required properties of behavioural equivalences.¹
- ▶ In particular, \sim and \approx^c are deadlock sensitive.
- ▶ Hennessy-Milner logic is a logic for expressing properties of processes.

Aim of this lecture

1. Study the connection between strong bisimilar processes and HML.

¹For weak bisimilarity the notion of observation congruence was needed.

Overview

- 1 Aims of this lecture
- 2 Introduction**
- 3 Hennessy-Milner logic
- 4 Correspondence HML and strong bisimilarity
- 5 Characteristic properties
- 6 Summary

Verifying correctness of reactive systems

Verifying correctness of reactive systems

Equivalence checking approach

Verifying correctness of reactive systems

Equivalence checking approach

$$Impl \equiv Spec$$

- ▶ \equiv is an abstract equivalence, e.g. \sim or \approx^c
- ▶ $Spec$ is often expressed in the same language as $Impl$, e.g., CCS
- ▶ $Spec$ provides the **full** specification of the intended behaviour.

Verifying correctness of reactive systems

Equivalence checking approach

$$Impl \equiv Spec$$

- ▶ \equiv is an abstract equivalence, e.g. \sim or \approx^c
- ▶ $Spec$ is often expressed in the same language as $Impl$, e.g., CCS
- ▶ $Spec$ provides the **full** specification of the intended behaviour.

Model checking approach

Verifying correctness of reactive systems

Equivalence checking approach

$$Impl \equiv Spec$$

- ▶ \equiv is an abstract equivalence, e.g. \sim or \approx^c
- ▶ $Spec$ is often expressed in the same language as $Impl$, e.g., CCS
- ▶ $Spec$ 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, e.g., HML
- ▶ $Property$ is a **partial** specification of the intended behaviour

Overview

- 1 Aims of this lecture
- 2 Introduction
- 3 Hennessy-Milner logic**
- 4 Correspondence HML and strong bisimilarity
- 5 Characteristic properties
- 6 Summary

HML syntax

HML syntax

Syntax of HML formulae

[Hennessy & Milner, 1985]

$$F, G ::= \text{true} \mid \text{false} \mid F \wedge G \mid F \vee G \mid \langle a \rangle F \mid [a] F$$

where a is an action.

HML syntax

Syntax of HML formulae

[Hennessy & Milner, 1985]

$$F, G ::= \text{true} \mid \text{false} \mid F \wedge G \mid F \vee G \mid \langle a \rangle F \mid [a] F$$

where a is an action.

Intuitive interpretation

- ▶ true, all processes satisfy this property
- ▶ false, no process satisfies this property
- ▶ \wedge , \vee logical conjunction and disjunction
- ▶ $\langle a \rangle F$, there is at least one a -successor that satisfies F
- ▶ $[a] F$, all a -successors have to satisfy F .

HML syntax

Syntax of HML formulae

[Hennessy & Milner, 1985]

$$F, G ::= \text{true} \mid \text{false} \mid F \wedge G \mid F \vee G \mid \langle a \rangle F \mid [a] F$$

where a is an action.

Intuitive interpretation

- ▶ true, all processes satisfy this property
- ▶ false, no process satisfies this property
- ▶ \wedge , \vee logical conjunction and disjunction
- ▶ $\langle a \rangle F$, there is at least one a -successor that satisfies F
- ▶ $[a] F$, all a -successors have to satisfy F .

Note that negation is not an elementary operation in HML.

HML semantics

HML semantics

Let $P \in \text{Prc}$, F, G HML-formulae, and a an action. Then:

$P \models \text{true}$ for each $P \in \text{Prc}$

$P \models \text{false}$ for no $P \in \text{Prc}$

$P \models F \wedge G$ iff $P \models F$ and $P \models G$

$P \models F \vee G$ iff $P \models F$ or $P \models G$

$P \models \langle a \rangle F$ iff $P \xrightarrow{a} P'$ for some $P' \in \text{Prc}$ with $P' \models F$

$P \models [a] F$ iff $P' \models F$ for all $P' \in \text{Prc}$ with $P \xrightarrow{a} P'$.

We write $P \not\models F$ whenever P does not satisfy F , i.e., not $(P \models F)$.

Examples

Negation

Negation

Complement of an HML-formula

$$\text{true}^c = \text{false}$$

$$\text{false}^c = \text{true}$$

$$(F \wedge G)^c = F^c \vee G^c$$

$$(F \vee G)^c = F^c \wedge G^c$$

$$(\langle a \rangle F)^c = [a] F^c$$

$$([a] F)^c = \langle a \rangle F^c$$

Negation

Complement of an HML-formula

$$\text{true}^c = \text{false}$$

$$\text{false}^c = \text{true}$$

$$(F \wedge G)^c = F^c \vee G^c$$

$$(F \vee G)^c = F^c \wedge G^c$$

$$(\langle a \rangle F)^c = [a] F^c$$

$$([a] F)^c = \langle a \rangle F^c$$

Theorem

For any $P \in \text{Prc}$ and HML-formula F :

1. $P \models F$ implies $P \not\models F^c$
2. $P \models F^c$ implies $P \not\models F$.

Negation

Complement of an HML-formula

$$\text{true}^c = \text{false}$$

$$\text{false}^c = \text{true}$$

$$(F \wedge G)^c = F^c \vee G^c$$

$$(F \vee G)^c = F^c \wedge G^c$$

$$(\langle a \rangle F)^c = [a] F^c$$

$$([a] F)^c = \langle a \rangle F^c$$

Theorem

For any $P \in \text{Prc}$ and HML-formula F :

1. $P \models F$ implies $P \not\models F^c$
2. $P \models F^c$ implies $P \not\models F$.

Proof.

By structural induction on F . Rather straightforward. □

Overview

- 1 Aims of this lecture
- 2 Introduction
- 3 Hennessy-Milner logic
- 4 Correspondence HML and strong bisimilarity**
- 5 Characteristic properties
- 6 Summary

Strong bisimulation

Strong bisimulation

[Park, 1981, Milner, 1989]

A binary relation $\mathcal{R} \subseteq Prc \times Prc$ is a **strong bisimulation** whenever for every $(P, Q) \in \mathcal{R}$, and $\alpha \in Act$:

1. if $P \xrightarrow{\alpha} P'$ then there exists $Q' \in Prc$ s.t. $Q \xrightarrow{\alpha} Q'$ and $(P', Q') \in \mathcal{R}$
2. if $Q \xrightarrow{\alpha} Q'$ then there exists $P' \in Prc$ s.t. $P \xrightarrow{\alpha} P'$ and $(P', Q') \in \mathcal{R}$.

Strong bisimulation

Strong bisimulation

[Park, 1981, Milner, 1989]

A binary relation $\mathcal{R} \subseteq Prc \times Prc$ is a **strong bisimulation** whenever for every $(P, Q) \in \mathcal{R}$, and $\alpha \in Act$:

1. if $P \xrightarrow{\alpha} P'$ then there exists $Q' \in Prc$ s.t. $Q \xrightarrow{\alpha} Q'$ and $(P', Q') \in \mathcal{R}$
2. if $Q \xrightarrow{\alpha} Q'$ then there exists $P' \in Prc$ s.t. $P \xrightarrow{\alpha} P'$ and $(P', Q') \in \mathcal{R}$.

Strong bisimilarity

The processes P and Q are **strongly bisimilar**, denoted $P \sim Q$, iff there is a strong bisimulation \mathcal{R} with $(P, Q) \in \mathcal{R}$. Thus,

$$\sim = \bigcup \{ \mathcal{R} \mid \mathcal{R} \text{ is a strong bisimulation} \}.$$

Relation \sim is called **strong bisimilarity**.

Image-finite transition system

Image-finite transition system

Image-finite process

A process P is **image-finite** iff the set $\{ P' \in Proc \mid P \xrightarrow{\alpha} P' \}$ is finite for every action α (possibly $\alpha = \tau$). A labeled transition system is **image-finite** if so is each of its states.

Image-finite transition system

Image-finite process

A process P is **image-finite** iff the set $\{P' \in \text{Prc} \mid P \xrightarrow{\alpha} P'\}$ is finite for every action α (possibly $\alpha = \tau$). A labeled transition system is **image-finite** if so is each of its states.

Examples

The process $A_{rep} = a.nil \parallel A_{rep}$ is not image-finite. By induction on n , one can prove that for each $n \in \mathbb{N}$:

$$A_{rep} \xrightarrow{a} \underbrace{a.nil \parallel \dots \parallel a.nil}_{n \text{ times}} \parallel nil \parallel A_{rep}$$

Also the process $A^\omega = \sum_{i \geq 0} a^i$ with $a^0 = nil$ and $a^{i+1} = a.a^i$ is not image-finite.

Relationship HML and trace equivalence

Recall from Lecture 4:

HML and trace equivalence

If $P, Q \in \text{Prc}$ satisfy the same HML-formulae, i.e., for every HML-formula F it holds $P \models F$ iff $Q \models F$, then $\text{Tr}(P) = \text{Tr}(Q)$. The converse does not hold.

Relationship HML and strong bisimilarity

Relationship HML and strong bisimilarity

Hennessy-Milner theorem

Let $(Proc, Act, \{ \xrightarrow{a} \mid a \in Act \})$ be an image-finite LTS and $P, Q \in Proc$.
Then:

$$P \sim Q$$

if and only if

for every HML-formula $F : (P \models F \text{ iff } Q \models F)$.

Relationship HML and strong bisimilarity

Hennessy-Milner theorem

Let $(Prc, Act, \{ \xrightarrow{a} \mid a \in Act \})$ be an image-finite LTS and $P, Q \in Prc$.
Then:

$$P \sim Q$$

if and only if

for every HML-formula $F : (P \models F \text{ iff } Q \models F)$.

Proof.

On the board. □

Relationship HML and strong bisimilarity

Hennessy-Milner theorem

Let $(Proc, Act, \{ \xrightarrow{a} \mid a \in Act \})$ be an image-finite LTS and $P, Q \in Proc$.
Then:

$$P \sim Q$$

if and only if

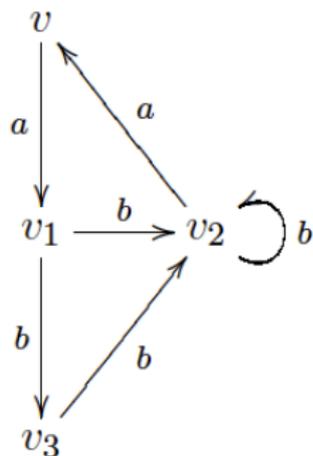
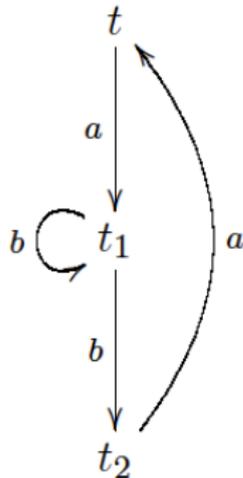
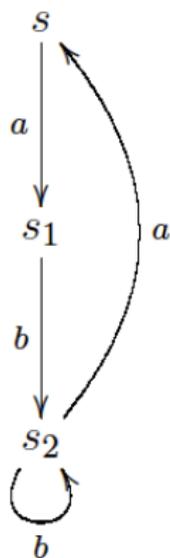
for every HML-formula $F : (P \models F \text{ iff } Q \models F)$.

Proof.

On the board. □

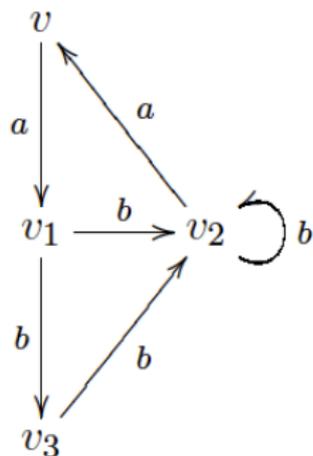
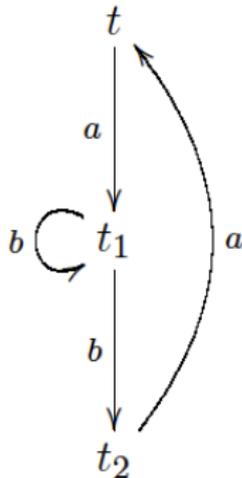
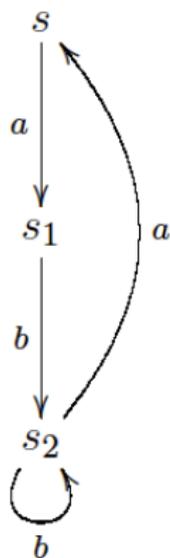
Showing $P \not\sim Q$ thus amounts to finding a single HML-formula F with
 $P \models F$ and $Q \not\models F$.

Example



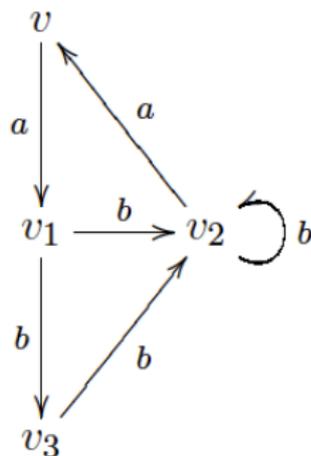
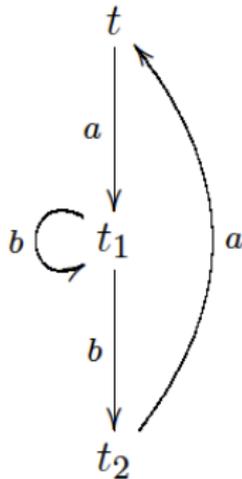
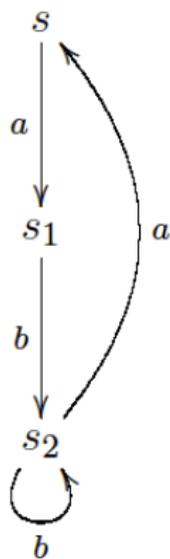
It follows $s \not\sim t$ and $s \not\sim v$ and $t \not\sim v$.

Example



It follows $s \not\sim t$ and $s \not\sim v$ and $t \not\sim v$. Distinguishing HML-formulas for:

Example

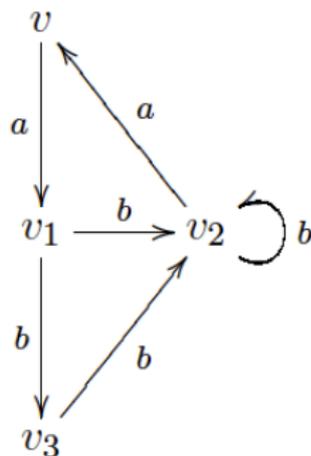
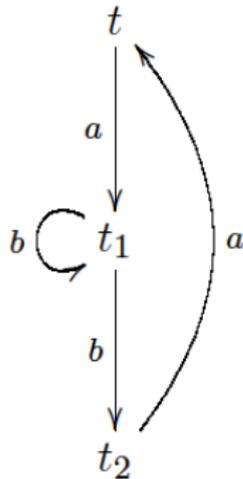
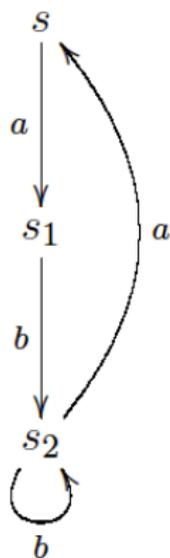


It follows $s \not\sim t$ and $s \not\sim v$ and $t \not\sim v$. Distinguishing HML-formulas for:

▶ s and t is: $F = \langle a \rangle \langle b \rangle [b]$ false

as $t \models F$ and $s \not\models F$

Example



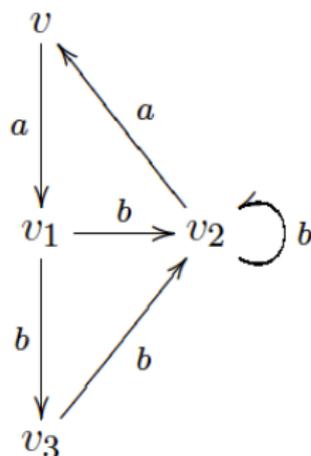
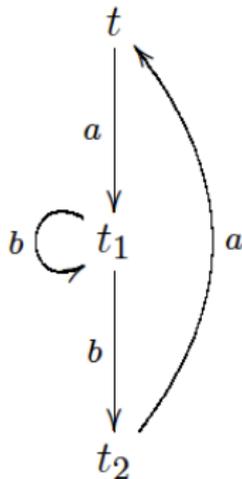
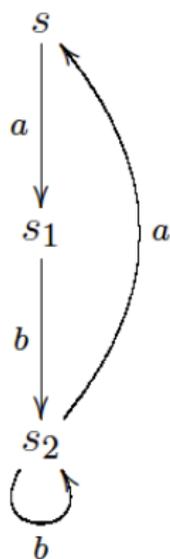
It follows $s \not\sim t$ and $s \not\sim v$ and $t \not\sim v$. Distinguishing HML-formulas for:

- ▶ s and t is: $F = \langle a \rangle \langle b \rangle [b]$ false
- ▶ s and v is: $F = \langle a \rangle \langle b \rangle [a]$ false

as $t \models F$ and $s \not\models F$

as $v \models F$ and $s \not\models F$

Example



It follows $s \not\sim t$ and $s \not\sim v$ and $t \not\sim v$. Distinguishing HML-formulas for:

- ▶ s and t is: $F = \langle a \rangle \langle b \rangle [b]$ false as $t \models F$ and $s \not\models F$
- ▶ s and v is: $F = \langle a \rangle \langle b \rangle [a]$ false as $v \models F$ and $s \not\models F$
- ▶ t and v is: $F = \langle a \rangle \langle b \rangle (\langle a \rangle \text{ true} \wedge \langle b \rangle \text{ true})$ as $v \models F$ and $t \not\models F$.

Counterexample for non image-finite processes

Overview

- 1 Aims of this lecture
- 2 Introduction
- 3 Hennessy-Milner logic
- 4 Correspondence HML and strong bisimilarity
- 5 Characteristic properties**
- 6 Summary

Characteristic properties

The Hennessy-Milner theorem asserts that for image-finite processes, strong bisimilarity and HML-equivalence coincide.

Characteristic properties

The Hennessy-Milner theorem asserts that for image-finite processes, strong bisimilarity and HML-equivalence coincide.

As a next step, we show that for **finite** transition systems, the equivalence classes under \sim can be characterised with a single formula in HML extended with recursion.

Characteristic properties

The Hennessy-Milner theorem asserts that for image-finite processes, strong bisimilarity and HML-equivalence coincide.

As a next step, we show that for **finite** transition systems, the equivalence classes under \sim can be characterised with a single formula in HML extended with recursion.

For finite process P , this HML-formula X_P is called P 's **characteristic property**.

The need for recursion

The need for recursion

There is **no recursion-free** HML-formula F_p that can characterize the process P defined by $X = a.X$ up to strong bisimilarity.

²The maximal number of nested occurrences of modal operators in F .

The need for recursion

The need for recursion

There is **no recursion-free** HML-formula F_p that can characterize the process P defined by $X = a.X$ up to strong bisimilarity.

Proof.

By contraposition. Let HML-formula F_p with $\llbracket F_p \rrbracket = \{ Q \mid P \sim Q \}$. In particular, $P \models F_p$ and $Q \models F_p$ implies $P \sim Q$ for each Q . We will show that this cannot hold for any formula F_p . Let P_0, P_1, P_2, \dots be defined by $P_0 = \text{nil}$ and $P_{i+1} = a.P_i$. P_i can execute i a -actions in a row and then terminates. Nothing else. Obviously $P \not\sim P_i$ for every i . Claim: for every HML-formula F it holds $P \models F$ iff $P_k \models F$ where k is the modal depth² of F . This can be proven by induction on the structure of F . Thus, $P \sim P_k$. Contradiction. \square

²The maximal number of nested occurrences of modal operators in F .

HML with recursion

HML with recursion

Syntax of recursive HML formulae

[Hennessy & Milner, 1985]

Let $\mathcal{X} = \{X_1, \dots, X_n\}$ be a set of **variables**. The syntax of HML over \mathcal{X} is defined by the grammar:

$$F, G ::= X_i \mid \text{true} \mid \text{false} \mid F \wedge G \mid F \vee G \mid \langle a \rangle F \mid [a] F$$

where $0 < i \leq n$ and a is an action. A **mutually recursive equation system** has the form

$$(X_i = F_{X_i} \mid 0 < i \leq n)$$

where F_{X_i} is a HML-formula over \mathcal{X} for every $0 < i \leq n$.

We skip the details of the semantics; see Lecture 5 for the details.

Characteristic formula

Consider the finite LTS $(\{P_1, \dots, P_n\}, Act, \rightarrow)$ and let $\mathcal{X} = \{X_{P_1}, \dots, X_{P_n}, \dots\}$ contain (at least) n variables.

Intuitively, X_P is the syntactic symbol for the characteristic formula of process P .

A characteristic formula for P has to describe which actions P can perform, which actions it cannot perform and what happens after performing an action.

Example

A coffee machine (again) on the black board.

Characteristic property

Characteristic property

[Ingolfsdottir et. al, 1987]

For finite process $P \in Proc$, let recursive HML-formula X_P be defined by:

$$X_P \stackrel{\text{max}}{=} \bigwedge_{a, P'. P \xrightarrow{a} P'} \langle a \rangle X_{P'} \wedge \bigwedge_a [a] \left(\bigvee_{a, P'. P \xrightarrow{a} P'} X_{P'} \right)$$

Then: $Q \models X_P$ iff $P \sim Q$ for every $Q \in Proc$.

The formula X_P is called the **characteristic property** of P .

Proof.

Outside the scope of this lecture. □

Overview

- 1 Aims of this lecture
- 2 Introduction
- 3 Hennessy-Milner logic
- 4 Correspondence HML and strong bisimilarity
- 5 Characteristic properties
- 6 Summary**

Summary

1. Strong bisimilarity and HML-equivalence coincide for image-finite processes.
2. This result does not hold for processes that are not image-finite.
3. Any two strong bisimilar processes satisfy the same HML formulas.
4. For finite processes a recursive HML-formula does exist that precisely characterises the strong bisimilar processes.