Modeling & Verification

Hennessy-Milner Logic with Recursion

Max Tschaikowski (tschaikowski@cs.aau.dk)
Slides Courtesy of Giorgio Bacci

in the last Lecture

- Limit of expressibility of Hennessy-Milner logic
- Tarski's Fixed Point Theorem (+ a bit of lattice theory)
- Computing fixed points on finite lattices

in this Lecture

- Fixed point Characterisation of Strong Bisimilarity
- Hennessy-Milner Logic with Recursively defined Variables
- Game characterisation of HML-satisfiability
- Mutually Recursive definitions (+ Characteristic Formula)

Strong Bisimilarity as a Fixed Point

$$\sim = \mathcal{B}(\sim)$$

Tarski's Fixed Point Theorem

Theorem (Tarski)

Let $(D,\sqsubseteq)$ be a complete lattice and $f : D \longrightarrow D$ a monotonic function on D. Then

1. The set of fixed points of f is a complete lattice
2. The least and greatest fixed point of f exists and are given as follows

(least fixed point) $lfp(f) = \sqcap \{d \in D \mid f(d) \sqsubseteq d\}$

(greatest fixed point) $gfp(f) = \sqcup \{d \in D \mid d \sqsubseteq f(d)\}$

Strong Bisimilarity

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

Definition (Strong Bisimulation)

A binary relation $R \subseteq Proc \times Proc$ is a *strong bisimulation* iff whenever s R t, for each $\alpha \in Act$

- if $s \xrightarrow{\alpha} s'$, then $t \xrightarrow{\alpha} t'$, for some t' such that s'R t'
- if $t \xrightarrow{\alpha} t'$, then $s \xrightarrow{\alpha} s'$, for some s' such that s'R t'

Definition (Strong Bisimilarity)

Two states $s,t \in Proc$ are *strongly bisimilar* ($s \sim t$) iff there exists a strong bisimulation R such that s R t.

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

Fixed Point Operator

We would like to define an operator on relations:

$$\mathcal{B} : 2^{\mathrm{Proc}\times\mathrm{Proc}} \longrightarrow 2^{\mathrm{Proc}\times\mathrm{Proc}}$$

Definition (Bisimulation Operator)

Let $R \subseteq \mathrm{Proc}\times\mathrm{Proc}$. Then define $\mathcal{B}(R)$ as follows:

$(s,t) \in \mathcal{B}(R)$ iff for each $\alpha \in \mathrm{Act}$

- if $s \xrightarrow{\alpha} s'$, then $t \xrightarrow{\alpha} t'$, for some t' such that $(s',t') \in R$
- if $t \xrightarrow{\alpha} t'$, then $s \xrightarrow{\alpha} s'$, for some s' such that $(s',t') \in R$.

Bisimilarity is a Fixed Point

Observations

- $(2^{\mathrm{Proc}\times\mathrm{Proc}}, \subseteq)$ is a complete lattice
- the function $\mathcal{B}$ is monotonic
- R is a strong bisimulation iff $R \subseteq \mathcal{B}(R)$

Theorem

Strong bisimilarity is the greatest fixed point of $\mathcal{B}$

$$\sim \; = \bigcup\{R \mid R \subseteq \mathcal{B}(R)\} = \mathrm{gfp}(\mathcal{B})$$

HML with Recursively defined Variables

HML with Recursion

Formulae of HML with recursively defined variables are defined by means of the following the grammar

$$\phi ::= X \mid tt \mid ff \mid \phi \wedge \phi \mid \phi \vee \phi \mid \langle a \rangle \phi \mid [a]\phi$$

recursive variable $X \in \mathbb{X}$

where each variable X is associated with a *unique* recursive equation of either one of the two following form:

$$X \overset{min}{=} \phi_X \quad \text{or} \quad X \overset{max}{=} \phi_X$$

such that ϕ_X is a formula of the logic (possibly containing X).

Semantics of HML

For each formula ϕ, we want to define $\langle\!\langle \phi \rangle\!\rangle \subseteq Proc$ as the set of all states that satisfy the formula ϕ

$$\langle\!\langle X \rangle\!\rangle = ??$$

it should be the solution of the recursive equation it is associated with

$$\langle\!\langle tt \rangle\!\rangle = Proc$$

$$\langle\!\langle ff \rangle\!\rangle = \emptyset$$

$$\langle\!\langle \phi \wedge \psi \rangle\!\rangle = \langle\!\langle \phi \rangle\!\rangle \cap \langle\!\langle \psi \rangle\!\rangle$$

$$\langle\!\langle \phi \vee \psi \rangle\!\rangle = \langle\!\langle \phi \rangle\!\rangle \cup \langle\!\langle \psi \rangle\!\rangle$$

$$\langle\!\langle \langle a \rangle \phi \rangle\!\rangle = \langle \cdot a \cdot \rangle \langle\!\langle \phi \rangle\!\rangle$$

$$\langle\!\langle [a] \phi \rangle\!\rangle = [\cdot a \cdot] \langle\!\langle \phi \rangle\!\rangle$$

Semantics of Variables

Assume for the moment that we have *only one* recursively defined variable, say X, with associated recursive definition

$$X \equiv \phi_X$$

Recall that this means
$\langle\!\langle X \rangle\!\rangle = \langle\!\langle \phi_X \rangle\!\rangle$

Example

Assume that $\phi_X = [a]ff \vee \langle a \rangle X$. Then,

$$X \equiv \phi_X \quad \text{iff} \quad \langle\!\langle X \rangle\!\rangle = [\cdot a \cdot]\langle\!\langle ff \rangle\!\rangle \cup \langle \cdot a \cdot \rangle \langle\!\langle X \rangle\!\rangle$$

It has the form of a fixed point!
$\langle\!\langle X \rangle\!\rangle = \mathcal{O}(\langle\!\langle X \rangle\!\rangle)$

Hence we can use the denotational semantics of formulae to induce an operator $\mathcal{O}_\phi : 2^{Proc} \longrightarrow 2^{Proc}$ on the lattice $(2^{Proc}, \subseteq)$

Definition of $\mathcal{O}_\phi$

Formally, the operator $\mathcal{O}_\phi$: $2^{Proc} \longrightarrow 2^{Proc}$ is defined as follows, for arbitrary $S \subseteq Proc$

$$\mathcal{O}_X(S) = S$$

$$\mathcal{O}_{tt}(S) = Proc$$

$$\mathcal{O}_{ff}(S) = \varnothing$$

$$\mathcal{O}_{\phi \wedge \psi}(S) = \mathcal{O}_\phi(S) \cap \mathcal{O}_\psi(S)$$

$$\mathcal{O}_{\phi \vee \psi}(S) = \mathcal{O}_\phi(S) \cup \mathcal{O}_\psi(S)$$

$$\mathcal{O}_{\langle a \rangle \phi}(S) = \langle \cdot a \cdot \rangle \mathcal{O}_\phi(S)$$

$$\mathcal{O}_{[a]\phi}(S) = [\cdot a \cdot]\mathcal{O}_\phi(S)$$

For each formula ϕ the operator $\mathcal{O}_\phi$ is *monotonic*:

$S \subseteq S'$ implies $\mathcal{O}_\phi(S) \subseteq \mathcal{O}_\phi(S')$

Semantics of Variables

Observation

$(2^{Proc}, \subseteq)$ is a complete lattice and $\mathcal{O}_\phi : 2^{Proc} \longrightarrow 2^{Proc}$ is monotonic. So, by Tarski's fixed point theorem, it has *unique* greatest and least fixed points!

Semantics of the variable X

- If $X \stackrel{min}{=} \phi$, then $\langle\!\langle X \rangle\!\rangle = \bigcap \{\, S \subseteq Proc \mid \mathcal{O}_\phi(S) \subseteq S \,\}$ — least fixed point
- If $X \stackrel{max}{=} \phi$, then $\langle\!\langle X \rangle\!\rangle = \bigcup \{\, S \subseteq Proc \mid S \subseteq \mathcal{O}_\phi(S) \,\}$ — greatest fixed point

Recursive Properties

- Pos(ϕ): $X \overset{min}{=} \phi \vee \langle Act \rangle X$

possibly ϕ

- Inv(ϕ): $X \overset{max}{=} \phi \wedge [Act]X$

invariantly ϕ

in all paths, ϕ is *eventually* satisfied

- Even(ϕ): $X \overset{min}{=} \phi \vee (\langle Act \rangle tt \wedge [Act]X)$
- Safe(ϕ): $X \overset{max}{=} \phi \wedge ([Act]ff \vee \langle Act \rangle X)$

exists a path where ϕ is always satisfied

where ϕ^c is the complement of ϕ

Note that ...

$Inv(\phi^c) = Pos(\phi)^c$ and $Safe(\phi^c) = Even(\phi)^c$

Strong & Weak Until

It is also possible to express that ϕ should be satisfied in each transition step until ψ becomes true

Strong Until

$\phi \, \mathcal{U}^s \, \psi$: $\quad X \stackrel{min}{=} \psi \vee (\phi \wedge \langle Act \rangle tt \wedge [Act]X)$

"*all paths reach a state where ψ is satisfied and ϕ must hold until this happens*"

Weak Until

$\phi \, \mathcal{U}^w \, \psi$: $\quad X \stackrel{max}{=} \psi \vee (\phi \wedge [Act]X)$

"*in all paths, ϕ must hold until it is reached a state where ψ is satisfied (but maybe this will never happen!)*"

Examples

Note that ...

Inv(ϕ) = $\phi\ \mathcal{U}^{w}$ ff and Even(ϕ) = tt $\mathcal{U}^{s}\ \phi$

Inv(ϕ): $X \overset{max}{=} \phi \wedge [Act]X$

$\phi\ \mathcal{U}^{w}$ ff: $X \overset{max}{=} ff \vee (\phi \wedge [Act]X)$

Even(ϕ): $X \overset{min}{=} \phi \vee (\langle Act \rangle tt \wedge [Act]X)$

tt $\mathcal{U}^{s}\ \phi$: $X \overset{min}{=} \phi \vee (tt \wedge \langle Act \rangle tt \wedge [Act]X)$

Game Semantics for HML

satisfiability proven or disproven
by playing a game!

Game Characterisation

Intuition: **attacker** is aiming to prove $s \not\models \phi$,
while **defender** is aiming to prove $s \models \phi$.

The *configurations* of the game are pairs of the form (s,ϕ)

Definition (Next Configuration)

- (s,tt) and (s,ff) have no successor
- $(s,\phi\wedge\psi)$ and $(s,\phi\vee\psi)$ have as successors: (s,ϕ) and (s,ψ)
- $(s,\langle a\rangle\phi)$ and $(s,[a]\phi)$ have as successor all configurations of the form (s',ϕ), for all s' such that $s \xrightarrow{a} s'$
- (s,X) has as unique successor (s,ϕ_X), for $X \overset{min}{=} \phi_X$ or $X \overset{max}{=} \phi_X$

The rules of the Game

The game starts from the configuration (s,ϕ) and the next configuration is selected according to the following rules:

Rules of the game

- **Attacker** picks a successor configuration whenever the current configuration is of the form $(s,\phi\wedge\psi)$ or $(s,[a]\phi)$
- **Defender** picks a successor configuration whenever the current configuration is of the form $(s,\phi\vee\psi)$ or $(s,\langle a\rangle\phi)$.
- If the current configuration is of the form (s,X), then the configuration is uniquely determined

A *play* is a maximal sequence of configuration following the rules.

Winning Conditions

Finite play

- **Attacker** wins if **defender** gets stuck (i.e., there are no next configurations) or the configuration (s,ff) is reached.
- **Defender** wins if **attacker** gets stuck (i.e., there are no next configurations) or the configuration (s,tt) is reached.

Infinite play

- **Attacker** wins if X is defined as $X \overset{\min}{=} \phi_X$.
- **Defender** wins if X is defined as $X \overset{\max}{=} \phi_X$.

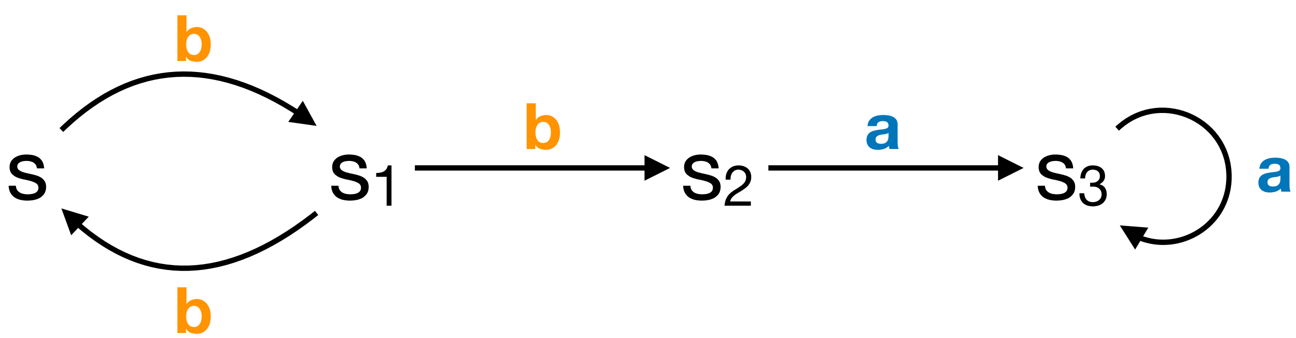
Semantics via a Game

A universal strategy is a "game plot" that takes into consideration all the possible available moves that the opponent player can do.

Theorem

- $s \models \phi$ iff **defender** has a universal winning strategy starting from (s, ϕ).
- $s \not\models \phi$ iff **attacker** has a universal winning strategy starting from (s, ϕ).

Example 1



$$X \overset{\min}{=} \langle a\rangle tt \vee \langle b\rangle X$$

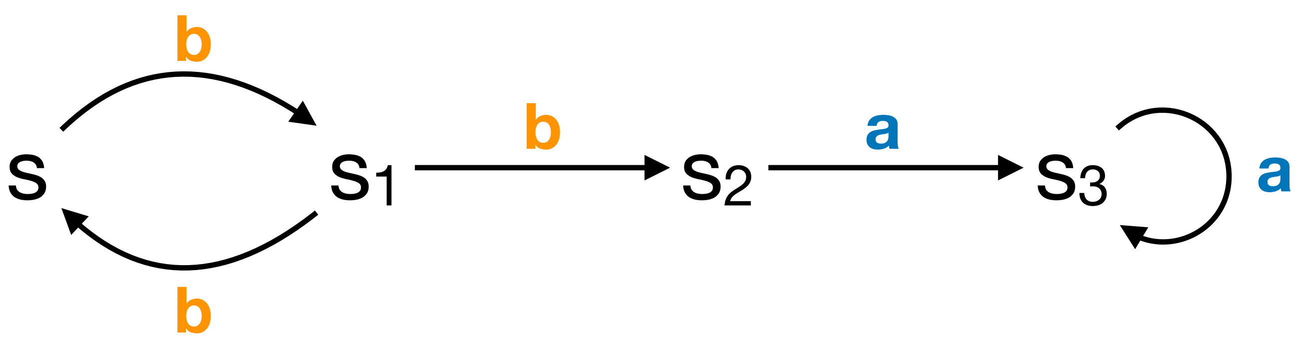
We show that $s \models X$ by defining a universal winning strategy for **defender** starting from (s,X)

$$(s,X) \longrightarrow (s, \langle a\rangle tt \vee \langle b\rangle X) \xrightarrow{D} (s, \langle b\rangle X) \xrightarrow{D} (s_1, X)$$

$$\longrightarrow (s_1, \langle a\rangle tt \vee \langle b\rangle X) \xrightarrow{D} (s_1, \langle b\rangle X) \xrightarrow{D} (s_2, X)$$

$$\longrightarrow (s_2, \langle a\rangle tt \vee \langle b\rangle X) \xrightarrow{D} (s_2, \langle a\rangle tt) \xrightarrow{D} (s_3, tt) \quad \checkmark$$

Example 2



$$X \overset{max}{=} \langle b\rangle tt \wedge [b]X$$

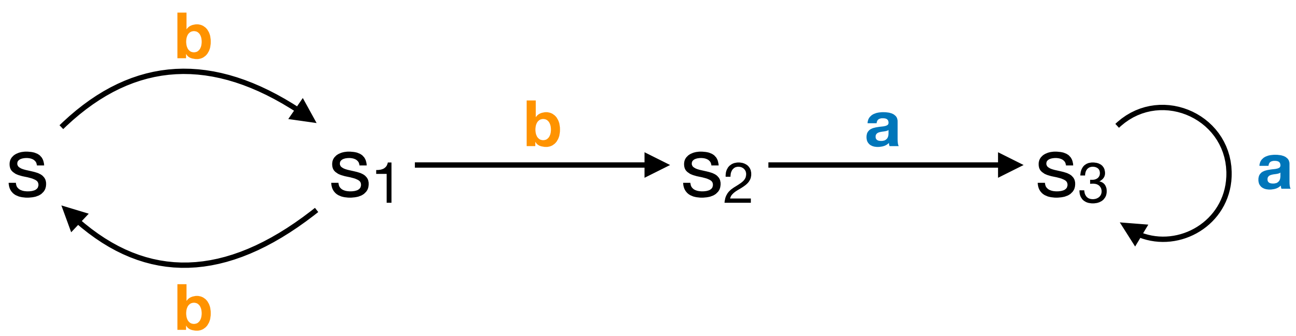
We show that $s \not\models X$ by defining a universal winning strategy for **attacker** starting from (s,X)

$$(s,X) \longrightarrow (s, \langle b\rangle tt \wedge [b]X) \xrightarrow{A} (s, [b]X) \xrightarrow{A} (s_1, X)$$

$$\longrightarrow (s_1, \langle b\rangle tt \wedge [b]X) \xrightarrow{A} (s_1, [b]X) \xrightarrow{A} (s_2, X)$$

$$\longrightarrow (s_2, \langle b\rangle tt \wedge [b]X) \xrightarrow{A} (s_2, \langle b\rangle tt) \not\longrightarrow$$

Example 3



$$X \overset{max}{=} \langle a \rangle tt \wedge [a]X$$

We show that $s_2 \vDash X$ by defining a universal winning strategy for **defender** starting from (s_2, X)

From (s_2,X): $(s_2, X) \longrightarrow (s_2, \langle a \rangle tt \wedge [a]X)$

- if $(s_2, \langle a \rangle tt \wedge [a]X) \xrightarrow{A} (s_2, \langle a \rangle tt)$, then $(s_2, \langle a \rangle tt) \xrightarrow{D} (s_3, tt)$ $\checkmark$
- if $(s_2, \langle a \rangle tt \wedge [a]X) \xrightarrow{A} (s_2, [a]X)$, then $(s_2, [a]X) \xrightarrow{A} (s_3, X)$

From (s_3,X): $(s_3, X) \longrightarrow (s_3, \langle a \rangle tt \wedge [a]X)$

- if $(s_3, \langle a \rangle tt \wedge [a]X) \xrightarrow{A} (s_3, \langle a \rangle tt)$, then $(s_3, \langle a \rangle tt) \xrightarrow{D} (s_3, tt)$ $\checkmark$
- if $(s_3, \langle a \rangle tt \wedge [a]X) \xrightarrow{A} (s_3, [a]X)$, then $(s_3, [a]X) \xrightarrow{A} (s_3, X)$ ... $\checkmark$

Mutually Recursive Equational Systems

More than one variable!

More than One Variable

Nested Definitions of Recursive Variables

$$X \overset{\min}{=} Y \vee \langle Act \rangle X \qquad Y \overset{\max}{=} \langle a \rangle tt \wedge \langle Act \rangle Y$$

In this case, we can compute the solution of X and Y (i.e., their semantics) by first compute 《Y》 and then 《X》.

Mutually Recursive Definitions

$$X \overset{\max}{=} [a]Y \qquad Y \overset{\max}{=} \langle a \rangle X$$

In this case, we need to consider the complete lattice $(2^{Proc} \times 2^{Proc}, \sqsubseteq)$, where $(S,S') \sqsubseteq (Q,Q')$ iff $S \subseteq Q$ and $S' \subseteq Q'$.

Mutual Recursion

In general, a mutually recursive equational system has the form

$$D = \begin{cases} X_1 = \phi_1 \, , \\ \quad \vdots \\ X_n = \phi_n \, . \end{cases}$$

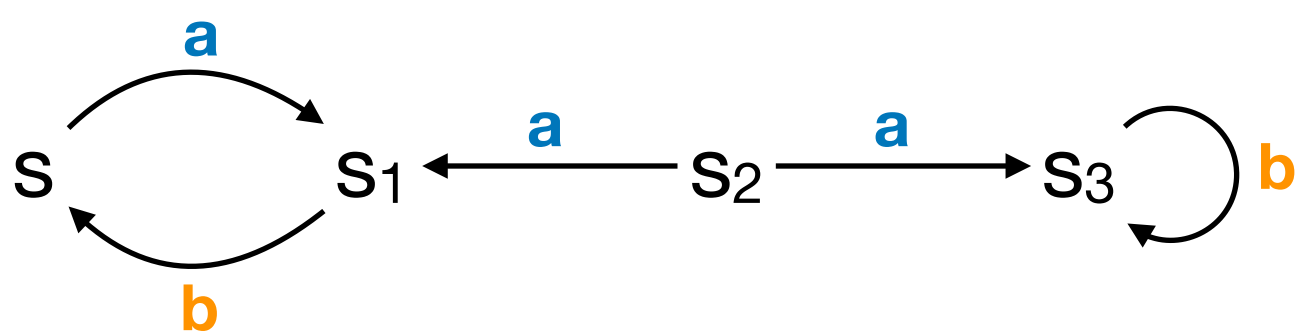
formulas can contain any of the variables X_i

The semantic function that is used to obtain the *largest* or *least* solution

$$\mathcal{O}_D(S_1, \ldots, S_n) = (\, \mathcal{O}_{\phi 1}(S_1, \ldots, S_n), \ldots, \mathcal{O}_{\phi n}(S_1, \ldots, S_n) \,)$$

where $\mathcal{O}_{Xi}(S_1, \ldots, S_n) = S_i$ for $1 \leq i \leq n$

Example



$$D = \begin{cases} X \overset{\max}{=} \langle a \rangle Y \wedge [a]Y \wedge [b]\mathit{ff} \\ Y \overset{\max}{=} \langle b \rangle X \wedge [b]Y \wedge [a]\mathit{ff} \end{cases}$$

$$\mathcal{O}_D(S_1,S_2) =$$

$$(\langle \cdot a \cdot \rangle S_2 \cap [\cdot a \cdot]S_2 \cap \{s,s_2\}, \langle \cdot b \cdot \rangle S_1 \cap [\cdot a \cdot]S_2 \cap \{s_1,s_3\})$$

hence, to compute the *largest solution* of D we can use the iterative algorithm for computing the *largest fixed point*

Characteristic Property

Ones we have recursive variables and the possibility of mutually recursive definitions (hence a system of equations) we obtain

Theorem (Characteristic Formula)

Let (Proc, Act, $\{\xrightarrow{a} \mid a \in \mathrm{Act}\}$) be a *finite* LTS and $s \in \mathrm{Proc}$.
Then, the formula

$$X_s \stackrel{max}{=} \bigwedge_{a,\, s \xrightarrow{a} t} \langle a \rangle X_t \;\wedge\; \bigwedge_{a} \, [a] \Big(\bigvee_{s \xrightarrow{a} t} X_t \Big)$$

is the *characteristic property* for s, i.e., for each $t \in \mathrm{Proc}$,

$$t \models X_s \quad \text{iff} \quad t \sim s$$

next time...

First Mini-Project