Lezione 22 MSC
HML with recursive definitions

Roberto Gorrieri

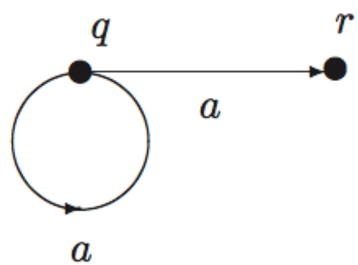
Motivation

- An HML formula can describe only a finite part of the overall behaviour (modal depth)
- [a]<a>tt ∨ <b>tt can be checked by looking at the first two performable actions only.
- We desire to express properties that may occur in arbitrarily long computations:

Safety properties: “for all the reachable states action a cannot be performed”

Liveness properties: “eventually, a state will be reached where action b can be performed”

Motivation (2)



$$p \models [a]\langle a\rangle tt$$

$$q \not\models [a]\langle a\rangle tt.$$

$$r = r_0 \xrightarrow{a} r_1 \xrightarrow{a} r_2 \xrightarrow{a} \cdots \xrightarrow{a} r_{n-1} \xrightarrow{a} r_n \quad (n \geq 0)$$

$$p \models [a]^{n+1}\langle a\rangle tt \qquad q \not\models [a]^{n+1}\langle a\rangle tt$$

- There is no HML formula distinguishing the two for any n:

$$Inv(\langle a\rangle tt) = \langle a\rangle tt \wedge [a]\langle a\rangle tt \wedge [a][a]\langle a\rangle tt \wedge \cdots = \bigwedge_{i\geq 0} [a]^i\langle a\rangle tt.$$

$$Pos([a]ff) = [a]ff \vee \langle a\rangle[a]ff \vee \langle a\rangle\langle a\rangle[a]ff \vee \cdots = \bigvee_{i\geq 0} \langle a\rangle^i[a]ff.$$

How to express finitely such formulae?

$$Inv(\langle a\rangle tt) = \langle a\rangle tt \wedge [a]\langle a\rangle tt \wedge [a][a]\langle a\rangle tt \wedge \cdots = \bigwedge_{i\geq 0} [a]^i \langle a\rangle tt.$$

- By means of a recursive equation

$$X \equiv \langle a\rangle tt \wedge [a]X$$

where F ≡ G means that F and G are equivalent. So we are looking for a solution of this recursive equation:

$$S = \langle \cdot a \cdot \rangle \mathbf{Proc} \cap [\cdot a \cdot] S.$$

F(S) = <.a.>Proc ∩ [.a.]S is monotone? 2^{Proc} is a complete lattice? Do we look for least or largest fixpoints? Over the lts of the previous slide, the least solution is the emptyset, while the largest solution is {p}. ➔ largest solution!

How to express finitely? (2)

$$Pos([a]\mathit{ff}) = [a]\mathit{ff} \vee \langle a\rangle[a]\mathit{ff} \vee \langle a\rangle\langle a\rangle[a]\mathit{ff} \vee \cdots = \bigvee_{i\geq 0} \langle a\rangle^i[a]\mathit{ff}.$$

- By means of a recursive equation

$$Y \equiv [a]\mathit{ff} \vee \langle a\rangle Y.$$

where F $\equiv$ G means that F and G are equivalent. So we are looking for a fixpoint solution of this function: G(S) = [.a.]∅ ∪ <.a.>S

Is G monotone? 2^{Proc} is a complete lattice? Do we look for least or largest fixpoints? Over the lts of the previous slide, the least solution is {q, r}, while the largest solution is {p, q, r}. ➔ least solution!

When min, when max?

Intuitively, we use largest solutions for those properties of a process that hold unless it has a finite computation that disproves the property. For instance, process q does *not* have property $Inv(\langle a \rangle tt)$ because it can reach a state in which no a-labelled transition is possible. Conversely, we use least solutions for those properties of a process that hold if it has a finite computation sequence which 'witnesses' the property. For instance, a process has the property $Pos(\langle a \rangle tt)$ if it has a computation leading to a state that can perform an a-labelled transition. This computation witnesses to the fact that the process can perform an a-labelled transition at some point in its behaviour.

Recursive formulae (1)

- Inv(<a>tt) $X \stackrel{\max}{=} \langle a \rangle tt \wedge [a] X$
- The LTS has one action only (namely “a”): in this state, “a” can be done, and, whatever transition is performed, “a” is still executable.

- Pos([a]ff) $Y \stackrel{\min}{=} [a] ff \vee \langle a \rangle Y.$
- The LTS has one action only (namely “a”): in this state, either “a” cannot be done, or, there is an “a”-labeled transition that leads to a state where this property holds.

- Inv(F) $X \stackrel{\max}{=} F \wedge [\mathsf{Act}] X$
- F holds for all reachable states (LTS over Act)

- Pos(F) $Y \stackrel{\min}{=} F \vee \langle \mathsf{Act} \rangle Y.$
- there is a reachable state where F holds (LTS over Act)

Recursive formulae (1)

- What is the meaning of Inv(<Act>tt)? No reachable deadlock
- What is the meaning of Pos([Act]ff)? May reach a deadlock
- Note that $\mathrm{Inv}(F)^c = \mathrm{Pos}(F^c)$ (c for complement/ negation)

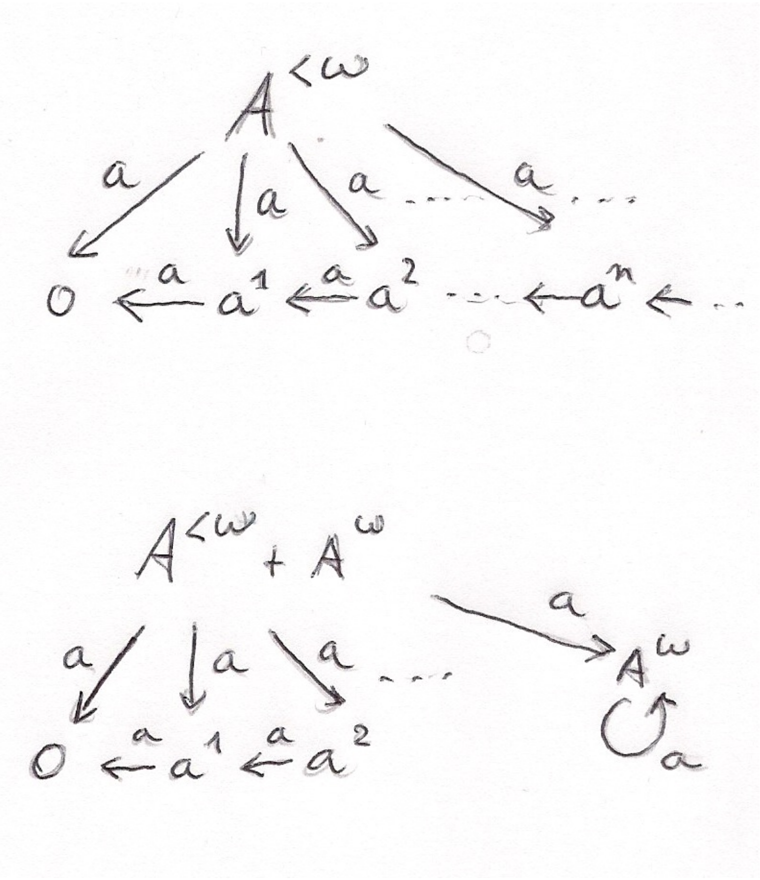
Example

- $A^{<w} + A^{w} \models \langle a \rangle \mathrm{Inv}(\langle a \rangle tt)$

while $A^{<w}$ does not

- $A^{<w} \models [a]\mathrm{Pos}([a]ff)$

while $A^{<w} + A^{w}$ does not



Other recursive properties

- Safe(F) holds if there is a complete (finite or infinite) computation where each traversed state satisfies F
$$X \stackrel{\max}{=} F \wedge ([\mathrm{Act}]ff \vee \langle \mathrm{Act} \rangle X)$$

- Even(F) holds if each of its complete computation will contain at least one state satisfying F
$$Y \stackrel{\min}{=} F \vee (\langle \mathrm{Act} \rangle tt \wedge [\mathrm{Act}]Y)$$
- Note that Safe(F)^c = Even(F^c) (c for complement/ negation)

Other recursive properties (2)

$F\,\mathcal{U}^s\,G$, the so-called *strong until*, which says that sooner or later p reaches a state where G is true and in all the states it traverses before this happens F must hold;

$$F\,\mathcal{U}^s\,G \overset{\min}{=} G \vee (F \wedge \langle \mathsf{Act} \rangle tt \wedge [\mathsf{Act}](F\,\mathcal{U}^s\,G))$$

$F\mathcal{U}^w G$, the so-called *weak until*, which says that F must hold in all the states p traverses until it reaches a state where G holds (but maybe this will never happen!).

$$F\,\mathcal{U}^w\,G \overset{\max}{=} G \vee (F \wedge [\mathsf{Act}](F\,\mathcal{U}^w\,G))$$

In fact, $Even(G) \equiv tt\,\mathcal{U}^s\,G$ and $Inv(F) \equiv F\,\mathcal{U}^w\,ff$.

Syntax and Semantics of HML with recursion

- Formulae over a single variable X

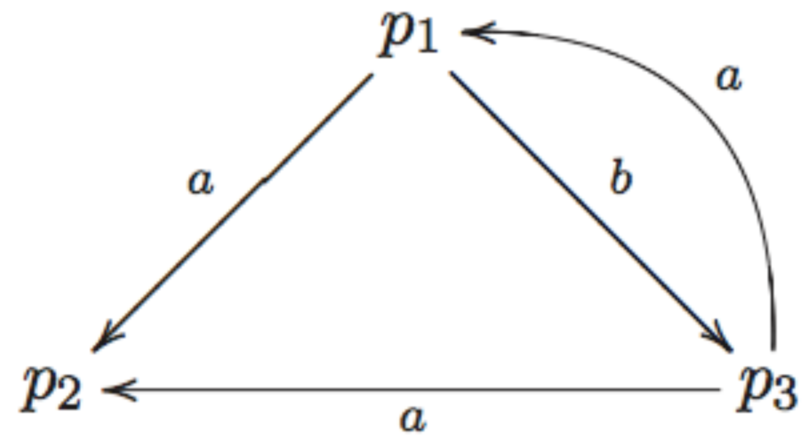
The syntax for Hennessy–Milner logic with one variable X, denoted by $\mathcal{M}_{\{X\}}$, is given by the following grammar:

$$F ::= X \mid tt \mid ff \mid F_1 \wedge F_2 \mid F_1 \vee F_2 \mid \langle a \rangle F \mid [a]F.$$

- Semantics of an "open" formula

Semantically a formula F (which may contain a variable X) is interpreted as a function $\mathcal{O}_F : 2^{\mathsf{Proc}} \to 2^{\mathsf{Proc}}$ that, given a set of processes that are assumed to satisfy X, gives us the set of processes that satisfy F.

Example



Example 6.2 Consider the formula $F = \langle a \rangle X$ and let Proc be the set of states in the transition graph in Figure 6.2. If X is satisfied by p_1 then $\langle a \rangle X$ will be satisfied by p_3, i.e. we expect that

$$\mathcal{O}_{\langle a \rangle X}(\{p_1\}) \;=\; \{p_3\}.$$

If the set of states satisfying X is $\{p_1, p_2\}$ then $\langle a \rangle X$ will be satisfied by $\{p_1, p_3\}$. Therefore we expect that

$$\mathcal{O}_{\langle a \rangle X}(\{p_1, p_2\}) \;=\; \{p_1, p_3\}.$$

What is the set $\mathcal{O}_{[b]X}(\{p_2\})$? ♦

Semantics for "open" formulae

Definition 6.1 Let $(\mathsf{Proc}, \mathsf{Act}, \{\xrightarrow{a} \mid a \in \mathsf{Act}\})$ be a labelled transition system. For each $S \subseteq \mathsf{Proc}$ and formula F, we define $\mathcal{O}_F(S)$ inductively as follows:

$$\begin{aligned}
\mathcal{O}_X(S) &= S, \\
\mathcal{O}_{tt}(S) &= \mathsf{Proc}, \\
\mathcal{O}_{ff}(S) &= \emptyset, \\
\mathcal{O}_{F_1 \wedge F_2}(S) &= \mathcal{O}_{F_1}(S) \cap \mathcal{O}_{F_2}(S), \\
\mathcal{O}_{F_1 \vee F_2}(S) &= \mathcal{O}_{F_1}(S) \cup \mathcal{O}_{F_2}(S), \\
\mathcal{O}_{\langle a \rangle F}(S) &= \langle \cdot a \cdot \rangle \mathcal{O}_F(S), \\
\mathcal{O}_{[a]F}(S) &= [\cdot a \cdot] \mathcal{O}_F(S).
\end{aligned}$$

♦

- Exercise: Show that O_F is monotone for any F.

Semantics for "closed" formulae

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

As shown in the previous section, such an equation can be interpreted as the set equation

$$[\![X]\!] = \mathcal{O}_{F_X}([\![X]\!]). \tag{6.6}$$

As $\mathcal{O}_{F_X}$ is a monotonic function over a complete lattice we know that (6.6) has solutions, i.e. that $\mathcal{O}_{F_X}$ has fixed points. In particular Tarski's fixed point theorem (Theorem 4.1) gives us that there is a unique *largest* fixed point, which we now denote FIX $\mathcal{O}_{F_X}$, and also a unique *least* one, which we denote fix $\mathcal{O}_{F_X}$. These are given respectively by

$$\text{FIX}\ \mathcal{O}_{F_X} = \bigcup \{S \subseteq \mathsf{Proc} \mid S \subseteq \mathcal{O}_{F_X}(S)\},$$

$$\text{fix}\ \mathcal{O}_{F_X} = \bigcap \{S \subseteq \mathsf{Proc} \mid \mathcal{O}_{F_X}(S) \subseteq S\}.$$

Semantics for “closed” formulae (2)

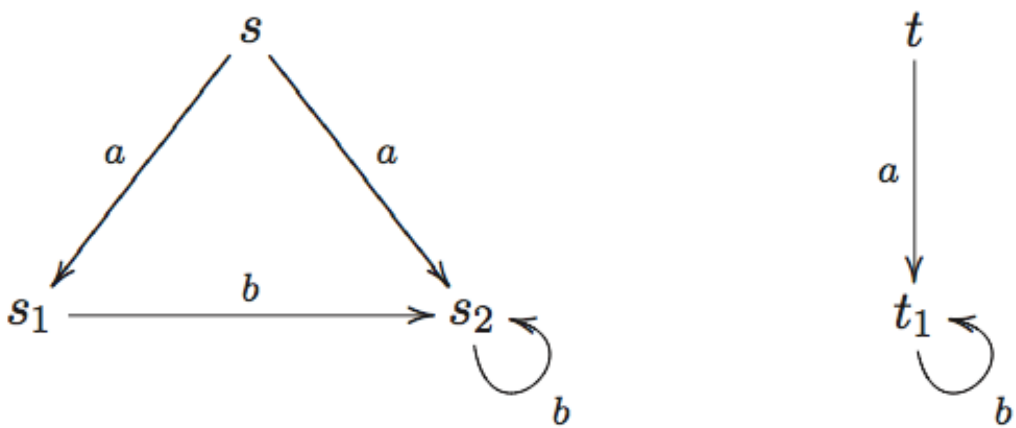
When Proc is finite we have the following characterization of the largest and least fixed points.

Theorem 6.1 If Proc is finite then FIX $\mathcal{O}_{F_X} = (\mathcal{O}_{F_X})^M(\mathsf{Proc})$ for some M and fix $\mathcal{O}_{F_X} = (\mathcal{O}_{F_X})^m(\emptyset)$ for some m.

Proof. This follows directly from the fixed point theorem for finite complete lattices. See Theorem 4.2 for the details. □

Example

- X =max Fx   with   Fx = <b>tt ∧ [b]X



- We are looking for the largest solution to the equation:

$$[\![X]\!] = (\langle\cdot b\cdot\rangle\{s, s_1, s_2, t, t_1\}) \cap [\cdot b\cdot][\![X]\!]$$

- That is, the largest fixpoint for the function

$$\mathcal{O}_{F_X}(S) = (\langle\cdot b\cdot\rangle\{s, s_1, s_2, t, t_1\}) \cap [\cdot b\cdot]S.$$

Example (2)

We therefore have that our first approximation to the largest fixed point is the set

$$
\begin{aligned}
\mathcal{O}_{F_X}(\{s, s_1, s_2, t, t_1\}) &= (\langle \cdot b \cdot \rangle \{s, s_1, s_2, t, t_1\}) \cap [\cdot b \cdot]\{s, s_1, s_2, t, t_1\} \\
&= \{s_1, s_2, t_1\} \cap \{s, s_1, s_2, t, t_1\} \\
&= \{s_1, s_2, t_1\}.
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{O}_{F_X}(\{s_1, s_2, t_1\}) &= (\langle \cdot b \cdot \rangle \{s, s_1, s_2, t, t_1\}) \cap [\cdot b \cdot]\{s_1, s_2, t_1\} \\
&= \{s_1, s_2, t_1\} \cap \{s, s_1, s_2, t, t_1\} \\
&= \{s_1, s_2, t_1\}.
\end{aligned}
$$

- Therefore, {s1, s2, t1} is the largest fixpoint.

Largest fixpoint and invariant properties

As we saw in the previous section, the property $Inv(F)$ is obtained as the largest fixed point to the recursive equation

$$X = F \wedge [\mathsf{Act}]X.$$

We will now show that $Inv(F)$ defined in this way indeed expresses that F *holds at all states in all transition sequences.*

For this purpose we let $\mathcal{I} : 2^{\mathsf{Proc}} \longrightarrow 2^{\mathsf{Proc}}$ be the corresponding semantic function, i.e.

$$\mathcal{I}(S) = [\![F]\!] \cap [\cdot\mathsf{Act}\cdot]S.$$

By Tarski's fixed point theorem this equation has exactly one largest solution, given by

$$\mathrm{FIX}\,\mathcal{I} = \bigcup\{S \mid S \subseteq \mathcal{I}(S)\}.$$

Largest fixpoint and invariant properties (2)

To show that FIX $\mathcal{I}$ indeed characterizes precisely the set of processes for which all states in all computations satisfy the property F, we need a direct (and obviously correct) formulation of this set. This is given by the set Inv, defined as follows:

$$Inv = \{ p \mid p \xrightarrow{\sigma} p' \text{ implies } p' \in [\![F]\!] \text{ for each } \sigma \in \mathsf{Act}^* \text{ and } p' \in \mathsf{Proc} \}.$$

Theorem 6.2 For every LTS $(\mathsf{Proc}, \mathsf{Act}, \{ \xrightarrow{a} \mid a \in \mathsf{Act}\})$, $Inv \; = \;$ FIX $\mathcal{I}$ holds.

Proof. We show the validity of the statement by proving each of the inclusions $Inv \subseteq$ FIX $\mathcal{I}$ and FIX $\mathcal{I} \subseteq Inv$ separately.

$Inv \subseteq$ FIX $\mathcal{I}$. To prove this inclusion it is sufficient to show that $Inv \subseteq \mathcal{I}(Inv)$. (Why?) To this end, let $p \in Inv$. Then, for all $\sigma \in \mathsf{Act}^*$ and $p' \in \mathsf{Proc}$,

$$p \xrightarrow{\sigma} p' \text{ implies } p' \in [\![F]\!]. \tag{6.7}$$

Inv $\subseteq$ Fix(I)

We must establish that $p \in \mathcal{I}(Inv)$ or, equivalently, that $p \in [\![F]\!]$ and $p \in [\cdot\mathsf{Act}\cdot]Inv$. We obtain the first of these two statements by taking $\sigma = \varepsilon$ in (6.7), because $p \xrightarrow{\varepsilon} p$ always holds.

To prove that $p \in [\cdot\mathsf{Act}\cdot]Inv$, we have to show that, for each process p' and action a,

$$p \xrightarrow{a} p' \text{ implies } p' \in Inv.$$

This is equivalent to proving that, for each sequence of actions σ' and process p'',

$$p \xrightarrow{a} p' \text{ and } p' \xrightarrow{\sigma'} p'' \text{ imply } p'' \in [\![F]\!].$$

However, this follows immediately by letting $\sigma = a\sigma'$ in (6.7).

Fix(I) $\subseteq$ Inv

$\mathrm{FIX}\,\mathcal{I} \subseteq Inv$. First we note that, since $\mathrm{FIX}\,\mathcal{I}$ is a fixed point of $\mathcal{I}$, it holds that

$$\mathrm{FIX}\,\mathcal{I} = [\![F]\!] \cap [\cdot\mathsf{Act}\cdot]\mathrm{FIX}\,\mathcal{I}. \tag{6.8}$$

To prove that $\mathrm{FIX}\,\mathcal{I} \subseteq Inv$, assume that $p \in \mathrm{FIX}\,\mathcal{I}$ and that $p \xrightarrow{\sigma} p'$. We shall show that $p' \in [\![F]\!]$ by induction on $|\sigma|$, the length of σ.

Base case $\sigma = \varepsilon$. For this case $p = p'$ and therefore, by (6.8) and our assumption that $p \in \mathrm{FIX}\,\mathcal{I}$, it holds that $p' \in [\![F]\!]$, which was to be shown.

Inductive step $\sigma = a\sigma'$. Now $p \xrightarrow{a} p'' \xrightarrow{\sigma'} p'$ for some p''. By (6.8) and our assumption that $p \in \mathrm{FIX}\,\mathcal{I}$, it follows that $p'' \in \mathrm{FIX}\,\mathcal{I}$. As $|\sigma'| < |\sigma|$ and $p'' \in \mathrm{FIX}\,\mathcal{I}$, by the induction hypothesis we may conclude that $p' \in [\![F]\!]$, as required.

This completes the proof of the second inclusion.

Mutually recursive equational systems

- So far we have only allowed one equation with one variable. However, it is sometimes useful, or even necessary, to define formulae recursively using two or more variables.
- Property: It is always the case that a process can perform an a-labelled transition leading to a state where b-transitions can be executed forever.

$$Inv(\langle a\rangle \mathrm{Forever}(b)) \stackrel{\max}{=} \langle a\rangle \mathrm{Forever}(b) \wedge [\mathrm{Act}] Inv(\langle a\rangle \mathrm{Forever}(b))$$

$$\mathrm{Forever}(b) \stackrel{\max}{=} \langle b\rangle \mathrm{Forever}(b)$$

Syntax

In general, a *mutually recursive equational system* has the form

$$X_1 = F_{X_1},$$
$$\vdots$$
$$X_n = F_{X_n},$$

where $\mathcal{X} = \{X_1, \ldots, X_n\}$ is a set of variables and, for $1 \leq i \leq n$, the formula F_{X_i} is in $\mathcal{M}_{\mathcal{X}}$ and can therefore contain any variable from $\mathcal{X}$. An example of such an equational system is

$$X = [a]Y,$$
$$Y = \langle a \rangle X.$$

- The key point is that all the equations are to be of the same type: either all max or all min!

Semantics

$$S_1 = \mathcal{O}_{F_{X_1}}(S_1, \ldots, S_n),$$
$$\vdots$$
$$S_n = \mathcal{O}_{F_{X_n}}(S_1, \ldots, S_n).$$

- such a system is interpreted over n-dimensional vectors of sets of processes, where n is the number of variables in X . Thus the new domain is D = $(2^{Proc})^n$ (n times the cross product of 2^{Proc} with itself), with a partial order defined component-wise:

$$(S_1, \ldots, S_n) \sqsubseteq (S'_1, \ldots, S'_n) \text{ if } S_1 \subseteq S'_1 \text{ and } S_2 \subseteq S'_2 \text{ and } \cdots \text{ and } S_n \subseteq S'_n$$

Semantics (2)

$(\mathcal{D}, \sqsubseteq)$ defined in this way yields a complete lattice with the least upper bound and the greatest lower bound also defined component-wise:

$$\bigsqcup\{(A_1^i, \ldots, A_n^i) \mid i \in I\} = (\bigcup\{A_1^i \mid i \in I\}, \ldots, \bigcup\{A_n^i \mid i \in I\}),$$

$$\bigsqcap\{(A_1^i, \ldots, A_n^i) \mid i \in I\} = (\bigcap\{A_1^i \mid i \in I\}, \ldots, \bigcap\{A_n^i \mid i \in I\}),$$

where I is an index set.

- Let D be a declaration over the set of variables X = {X1, . . . , Xn} that associates a formula F_{Xi} with each variable X_i, $1 \leq i \leq n$. (That is a system of equations.)
- We are looking for the largest or least solution of the equation:

$$[\![D]\!](S_1, \ldots, S_n) = (\mathcal{O}_{F_{X_1}}(S_1, \ldots, S_n), \ldots, \mathcal{O}_{F_{X_n}}(S_1, \ldots, S_n)).$$

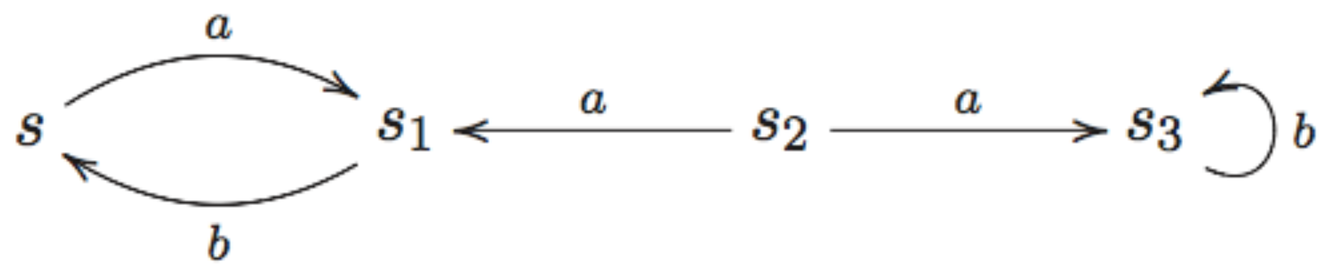
- where

$$\mathcal{O}_{X_i}(S_1, \ldots, S_n) = S_i \quad (1 \leq i \leq n).$$

Example

- Consider the system:
- Consider the lts

$$X \stackrel{\max}{=} \langle a \rangle Y \wedge [a]Y \wedge [b]\mathit{ff},$$
$$Y \stackrel{\max}{=} \langle b \rangle X \wedge [b]X \wedge [a]\mathit{ff}$$



- We have to compute the largest fixpoint of the function that maps (S_1, S_2) to

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

Example (2)

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

- S_1 stands for the set of states that are assumed to satisfy X,
- S_2 stands for the set of states that are assumed to satisfy Y,
- $\langle\cdot a\cdot\rangle S_2 \cap [\cdot a\cdot] S_2 \cap \{s, s_2\}$ is the set of states that satisfy the right-hand side of the defining equation for X under these assumptions, and
- $\langle\cdot b\cdot\rangle S_1 \cap [\cdot b\cdot] S_1 \cap \{s_1, s_3\}$ is the set of states that satisfy the right-hand side of the defining equation for Y under these assumptions.

- The starting pair is the top elment (S1, S2) =

$$(\{s, s_1, s_2, s_3\}, \{s, s_1, s_2, s_3\})$$

- By applying the function, the resulting pair is

$$(\{s, s_2\}, \{s_1, s_3\}).$$

Example (3)

- Iterating the procedure starting from $(\{s, s_2\}, \{s_1, s_3\})$.

we get $(\{s, s_2\}, \{s_1\})$

- Iterating the procedure starting from $(\{s, s_2\}, \{s_1\})$

we get $(\{s\}, \{s_1\})$.

- Iterating the procedure starting from $(\{s\}, \{s_1\})$.

we get again $(\{s\}, \{s_1\})$.

So this is the largest solution we were looking for!

Mixing largest and least fixed points

- For some properties, it is necessary to use both max and min recursive equations!
- Property: It is possible for the system to reach a state which may diverge.

$$Pos(F) \overset{\min}{=} F \vee \langle \mathbf{Act} \rangle Pos(F).$$

F =max <tau>F

- How to compute the semantics for such compound systems of equations? First compute the semantics for F (it is not based on anything else); then use such a soution to compute the semantics of Pos.

Nested mutually recursive equations

Definition 6.2 A n-nested mutually recursive equational system E is an n-tuple

$$\langle (D_1, \mathcal{X}_1, m_1), (D_2, \mathcal{X}_2, m_2), \ldots, (D_n, \mathcal{X}_n, m_n) \rangle,$$

where the $\mathcal{X}_i$ are pairwise-disjoint finite sets of variables and, for each $1 \leq i \leq n$,

- D_i is a declaration mapping the variables in the set $\mathcal{X}_i$ to formulae in HML with recursion that may use variables in the set $\bigcup_{1 \leq j \leq i} \mathcal{X}_j$,
- $m_i = \max$ or $m_i = \min$, and
- $m_i \neq m_{i+1}$.
    - such systems of equations have a unique solution, obtained by solving the first block and then proceeding with the others using the solutions already obtained for the previous blocks.