Model Cheching and HML

Rocco De Nicola

Dipartimento di Sistemi ed Informatica
Università di Firenze

Process Algebras and Concurrent Systems

Verifying Correctness of Reactive Systems

Let *Impl* be an implementation of a system (e.g. in CCS syntax).

Equivalence Checking Approach

$$\mathit{Impl} \equiv \mathit{Spec}$$

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

Verifying Correctness of Reactive Systems

Let *Impl* be an implementation of a system (e.g. in CCS syntax).

Equivalence Checking Approach

$$Impl \equiv Spec$$

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

Our Aim

Develop a logic in which we can express interesting properties of reactive systems.

Modal Properties – what can happen now (possibility, necessity)

- drink a coffee (can drink a coffee now)
- does not drink tea
- drinks both tea and coffee
- drinks tea after coffee

Logical Properties of Reactive Systems

Modal Properties – what can happen now (possibility, necessity)

- drink a coffee (can drink a coffee now)
- does not drink tea
- drinks both tea and coffee
- drinks tea after coffee

Temporal Properties – behaviour in time

- never drinks any alcohol
  (safety property: nothing bad can happen)
- eventually will have a glass of wine
  (liveness property: something good will happen)

Logical Properties of Reactive Systems

Modal Properties – what can happen now (possibility, necessity)

- drink a coffee (can drink a coffee now)
- does not drink tea
- drinks both tea and coffee
- drinks tea after coffee

Temporal Properties – behaviour in time

- never drinks any alcohol
  (safety property: nothing bad can happen)
- eventually will have a glass of wine
  (liveness property: something good will happen)

Can these properties be expressed using equivalence checking?

Hennessy-Milner Logic – Syntax

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

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

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 – 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

Remark

Temporal properties like *always/never in the future* or *eventually* are not included.

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)

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

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:

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

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 HML formula F

1. $p \models F \Longrightarrow p \not\models F^c$
2. $p \not\models F \Longrightarrow p \models F^c$

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 upto depth $md(F)$.

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 upto 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 upto k rounds then $s \models F$ if and only if $t \models F$.

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 upto 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 upto 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

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.

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

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

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

Infinite Conjunctions and Disjunctions vs. Recursion

Problems

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

Infinite Conjunctions and Disjunctions vs. Recursion

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 $X \stackrel{\mathrm{def}}{=} F \vee \langle Act \rangle X$

Infinite Conjunctions and Disjunctions vs. Recursion

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?

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)

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)

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?

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

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 $\langle \cdot a \cdot \rangle, [\cdot a \cdot] : 2^{(Proc)} \rightarrow 2^{(Proc)}$ 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*

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

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.

Image-Finite Labelled Transition System

Image-Finite System

Let $(Proc, Act, \{\xrightarrow{a} | \ a \in Act\})$ 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: $(p \models F \iff q \models F)$.

General Approach – Lattice Theory

Problem

For a set D and a function $f : D \longrightarrow D$, for which elements $x \in D$ we have

$$x = f(x) \text{ ?}$$

Such x's are called fixed points.

General Approach – Lattice Theory

Problem

For a set D and a function $f : D \to D$, for which elements $x \in D$ we have

$$x = f(x) \ ?$$

Such x's are called fixed points.

Partially Ordered Set

Partially ordered set (or simply a partial order) is a pair $(D, \sqsubseteq)$ s.t.

- D is a set
- $\sqsubseteq \subseteq D \times D$ is a binary relation on D which is
  - reflexive: $\forall d \in D.\ d \sqsubseteq d$
  - antisymmetric: $\forall d, e \in D.\ d \sqsubseteq e \ \wedge \ e \sqsubseteq d \ \Rightarrow \ d = e$
  - transitive: $\forall d, e, f \in D.\ d \sqsubseteq e \ \wedge \ e \sqsubseteq f \ \Rightarrow \ d \sqsubseteq f$

Supremum and Infimum

Upper/Lower Bounds (Let $X \subseteq D$)

- $d \in D$ is an upper bound for X (written $X \sqsubseteq d$) iff $x \sqsubseteq d$ for all $x \in X$
- $d \in D$ is a lower bound for X (written $d \sqsubseteq X$) iff $d \sqsubseteq x$ for all $x \in X$

Supremum and Infimum

Upper/Lower Bounds (Let $X \subseteq D$)

- $d \in D$ is an upper bound for X (written $X \sqsubseteq d$) iff $x \sqsubseteq d$ for all $x \in X$
- $d \in D$ is a lower bound for X (written $d \sqsubseteq X$) iff $d \sqsubseteq x$ for all $x \in X$

Least Upper Bound and Greatest Lower Bound (Let $X \subseteq D$)

- $d \in D$ is the least upper bound (supremum) for X ($\sqcup X$) iff
  1. $X \sqsubseteq d$
  2. $\forall d' \in D.\ X \sqsubseteq d' \;\Rightarrow\; d \sqsubseteq d'$
- $d \in D$ is the greatest lower bound (infimum) for X ($\sqcap X$) iff
  1. $d \sqsubseteq X$
  2. $\forall d' \in D.\ d' \sqsubseteq X \;\Rightarrow\; d' \sqsubseteq d$

Complete Lattices and Monotonic Functions

Complete Lattice

A partially ordered set $(D, \sqsubseteq)$ is called complete lattice iff $\sqcup X$ and $\sqcap X$ exist for any $X \subseteq D$.

We define the top and bottom by $\top \stackrel{\text{def}}{=} \sqcup D$ and $\bot \stackrel{\text{def}}{=} \sqcap D$.

Complete Lattices and Monotonic Functions

Complete Lattice

A partially ordered set $(D, \sqsubseteq)$ is called complete lattice iff $\sqcup X$ and $\sqcap X$ exist for any $X \subseteq D$.

We define the top and bottom by $\top \stackrel{\text{def}}{=} \sqcup D$ and $\bot \stackrel{\text{def}}{=} \sqcap D$.

Monotonic Function and Fixed Points

A function $f : D \to D$ is called monotonic iff

$$d \sqsubseteq e \quad \Rightarrow \quad f(d) \sqsubseteq f(e)$$

for all $d, e \in D$.

Element $d \in D$ is called fixed point iff $d = f(d)$.

Tarski's Fixed Point Theorem

Theorem (Tarski)

Let $(D, \sqsubseteq)$ be a complete lattice and let $f : D \rightarrow D$ be a monotonic function.

Then f has a unique largest fixed point z_{max} and a unique least fixed point z_{min} given by:

$$z_{max} \stackrel{\text{def}}{=} \sqcup\{x \in D \mid x \sqsubseteq f(x)\}$$

$$z_{min} \stackrel{\text{def}}{=} \sqcap\{x \in D \mid f(x) \sqsubseteq x\}$$

Computing Min and Max Fixed Points on Finite Lattices

Let $(D, \sqsubseteq)$ be a complete lattice and $f : D \to D$ monotonic.
Let $f^1(x) \stackrel{\text{def}}{=} f(x)$ and $f^n(x) \stackrel{\text{def}}{=} f(f^{n-1}(x))$ for $n > 1$, i.e.,

$$f^n(x) = \underbrace{f(f(\ldots f}_{n \text{ times}}(x)\ldots)).$$

Computing Min and Max Fixed Points on Finite Lattices

Let $(D, \sqsubseteq)$ be a complete lattice and $f : D \to D$ monotonic.
Let $f^1(x) \stackrel{\text{def}}{=} f(x)$ and $f^n(x) \stackrel{\text{def}}{=} f(f^{n-1}(x))$ for $n > 1$, i.e.,

$$f^n(x) = \underbrace{f(f(\ldots f}_{n \text{ times}}(x)\ldots)).$$

Theorem

If D is a finite set then there exist integers $M, m > 0$ such that

- $z_{max} = f^M(\top)$
- $z_{min} = f^m(\bot)$

Idea (for z_{min}): The following sequence stabilizes for any finite D

$$\bot \sqsubseteq f(\bot) \sqsubseteq f(f(\bot)) \sqsubseteq f(f(f(\bot))) \sqsubseteq \cdots$$

Monotonic Functions

Monotonic Function and Fixed Points

A function $f : 2^{Proc} \to 2^{Proc}$ is called monotonic iff

$$X \subseteq Y \quad \Rightarrow \quad f(X) \subseteq f(Y)$$

for all $X, Y \in 2^{Proc}$.

A set $X \in 2^{Proc}$ is called a fixed point of f iff $X = f(X)$.

Questions

Is the function $f(X) = X \cup \{s, t\}$ monotonic? What about $g(X) = Proc \setminus X$? Do these functions have fixed points?

Tarski's Fixed Point Theorem

Theorem (Tarski)

Let $f : 2^{Proc} \rightarrow 2^{Proc}$ be a monotonic function.
Then f has a unique largest fixed point z_{max} and a unique least fixed point z_{min} given by:

$$z_{max} \stackrel{\mathrm{def}}{=} \bigcup \{X \in 2^{Proc} \mid X \subseteq f(X)\}$$

$$z_{min} \stackrel{\mathrm{def}}{=} \bigcap \{X \in 2^{Proc} \mid f(X) \subseteq X\}$$

Computing Min and Max Fixed Points on Finite Sets

Let $f : 2^{Proc} \longrightarrow 2^{Proc}$ be monotonic.
Let $f^1(X) \stackrel{\mathrm{def}}{=} f(X)$ and $f^n(X) \stackrel{\mathrm{def}}{=} f(f^{n-1}(X))$ for $n > 1$, i.e.,

$$f^n(X) = \underbrace{f(f(\ldots f}_{n \text{ times}}(X)\ldots)).$$

Computing Min and Max Fixed Points on Finite Sets

Let $f : 2^{Proc} \longrightarrow 2^{Proc}$ be monotonic.
Let $f^1(X) \stackrel{\text{def}}{=} f(X)$ and $f^n(X) \stackrel{\text{def}}{=} f(f^{n-1}(X))$ for $n > 1$, i.e.,

$$f^n(X) = \underbrace{f(f(\ldots f}_{n \text{ times}}(X)\ldots)).$$

Theorem

If 2^{Proc} is a finite set then there exist integers $M, m > 0$ such that

- $z_{max} = f^M(Proc)$
- $z_{min} = f^m(\emptyset)$

Idea (for z_{min}): The following sequence stabilizes for any finite 2^{Proc}

$$\emptyset \subseteq f(\emptyset) \subseteq f(f(\emptyset)) \subseteq f(f(f(\emptyset))) \subseteq \cdots$$

HML with One Recursively Defined Variable

Syntax of Formulae

Formulae are given by the following abstract syntax

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

where $a \in Act$ and X is a distinguished variable with a definition

- $X \stackrel{\min}{=} F_X$, or $X \stackrel{\max}{=} F_X$

such that F_X is a formula of the logic (can contain X).

HML with One Recursively Defined Variable

Syntax of Formulae

Formulae are given by the following abstract syntax

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

where $a \in Act$ and X is a distinguished variable with a definition

- $X \stackrel{\min}{=} F_X$, or $X \stackrel{\max}{=} F_X$

such that F_X is a formula of the logic (can contain X).

How to Define Semantics?

For every formula F we define a function $O_F : 2^{Proc} \rightarrow 2^{Proc}$ s.t.

- if S is the set of processes that satisfy X then
- $O_F(S)$ is the set of processes that satisfy F.

Definition of $O_F : 2^{Proc} \longrightarrow 2^{Proc}$ (let $S \subseteq 2^{Proc}$)

$$
\begin{aligned}
O_X(S) &= S \\
O_{tt}(S) &= Proc \\
O_{ff}(S) &= \emptyset \\
O_{F_1 \wedge F_2}(S) &= O_{F_1}(S) \cap O_{F_2}(S) \\
O_{F_1 \vee F_2}(S) &= O_{F_1}(S) \cup O_{F_2}(S) \\
O_{\langle a \rangle F}(S) &= \langle \cdot a \cdot \rangle O_F(S) \\
O_{[a] F}(S) &= [\cdot a \cdot] O_F(S)
\end{aligned}
$$

Definition of $O_F : 2^{Proc} \longrightarrow 2^{Proc}$ (let $S \subseteq 2^{Proc}$)

$$
\begin{array}{rcl}
O_X(S) & = & S \\
O_{tt}(S) & = & Proc \\
O_{ff}(S) & = & \emptyset \\
O_{F_1 \wedge F_2}(S) & = & O_{F_1}(S) \cap O_{F_2}(S) \\
O_{F_1 \vee F_2}(S) & = & O_{F_1}(S) \cup O_{F_2}(S) \\
O_{\langle a \rangle F}(S) & = & \langle \cdot a \cdot \rangle O_F(S) \\
O_{[a]F}(S) & = & [\cdot a \cdot] O_F(S)
\end{array}
$$

O_F is monotonic for every formula F

$$S_1 \subseteq S_2 \ \Rightarrow \ O_F(S_1) \subseteq O_F(S_2)$$

Proof: easy (structural induction on the structure of F).

Semantics

Observation

We know O_F is monotonic, so O_F has a unique greatest and least fixed point.

Semantics of the Variable X

- If $X \overset{\max}{=} F_X$ then

$$[\![X]\!] = \bigcup\{S \subseteq Proc \mid S \subseteq O_{F_X}(S)\}.$$

- If $X \overset{\min}{=} F_X$ then

$$[\![X]\!] = \bigcap\{S \subseteq Proc \mid O_{F_X}(S) \subseteq S\}.$$

Selection of Temporal Properties

- *Inv(F)*: $X \stackrel{\max}{=} F \wedge [Act]X$
- *Pos(F)*: $X \stackrel{\min}{=} F \vee \langle Act \rangle X$

Selection of Temporal Properties

- $Inv(F)$: $X \stackrel{\max}{=} F \wedge [Act]X$
- $Pos(F)$: $X \stackrel{\min}{=} F \vee \langle Act \rangle X$

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

Selection of Temporal Properties

- $Inv(F)$: $X \overset{\max}{=} F \wedge [Act]X$
- $Pos(F)$: $X \overset{\min}{=} F \vee \langle Act \rangle X$
- $Safe(F)$: $X \overset{\max}{=} F \wedge ([Act]ff \vee \langle Act \rangle X)$
- $Even(F)$: $X \overset{\min}{=} F \vee (\langle Act \rangle tt \wedge [Act]X)$
- $F \, \mathcal{U}^w \, G$: $X \overset{\max}{=} G \vee (F \wedge [Act]X)$
- $F \, \mathcal{U}^s \, G$: $X \overset{\min}{=} G \vee (F \wedge \langle Act \rangle tt \wedge [Act]X)$

- $Inv(F)$: $X \overset{\max}{=} F \wedge [Act]X$
- $Pos(F)$: $X \overset{\min}{=} F \vee \langle Act\rangle X$

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

- $F\ \mathcal{U}^w\ G$: $X \overset{\max}{=} G \vee (F \wedge [Act]X)$
- $F\ \mathcal{U}^s\ G$: $X \overset{\min}{=} G \vee (F \wedge \langle Act\rangle \mathit{tt} \wedge [Act]X)$

Using until we can express e.g. $Inv(F)$ and $Even(F)$:

$$Inv(F) \equiv F\ \mathcal{U}^w\ \mathit{ff} \qquad\qquad Even(F) \equiv \mathit{tt}\ \mathcal{U}^s\ F$$

Examples of More Advanced Recursive Formulae

Nested Definitions of Recursive Variables

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

Solution: compute first $[\![Y]\!]$ and then $[\![X]\!]$.

Examples of More Advanced Recursive Formulae

Nested Definitions of Recursive Variables

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

Solution: compute first $[\![Y]\!]$ and then $[\![X]\!]$.

Mutually Recursive Definitions

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

Solution: consider a complete lattice $(2^{Proc} \times 2^{Proc}, \sqsubseteq)$ where $(S_1, S_2) \sqsubseteq (S_1', S_2')$ iff $S_1 \subseteq S_1'$ and $S_2 \subseteq S_2'$.

Examples of More Advanced Recursive Formulae

Nested Definitions of Recursive Variables

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

Solution: compute first $[\![Y]\!]$ and then $[\![X]\!]$.

Mutually Recursive Definitions

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

Solution: consider a complete lattice $(2^{Proc} \times 2^{Proc}, \sqsubseteq)$ where $(S_1, S_2) \sqsubseteq (S_1', S_2')$ iff $S_1 \subseteq S_1'$ and $S_2 \subseteq S_2'$.

Theorem (Characteristic Property for Finite-State Processes)

Let s be a process with finitely many reachable states. There exists a property X_s s.t. for all processes t: $s \sim t$ if and only if $t \in [\![X_s]\!]$.