Modal transition systems: a foundation for three-valued program analysis

Michael Huth and David Schmidt
Computing and Information Sciences
Kansas State University

Radha Jagadeesan
Department of Mathematical and Computer Sciences
Loyola University of Chicago

Outline

1. Partial systems

Outline

1. Partial systems
2. Modal transition systems

Outline

1. Partial systems

2. Modal transition systems

3. Property refutation and verification

Outline

1. Partial systems
2. Modal transition systems
3. Property refutation and verification
4. Semantic foundations

Outline

1. Partial systems
2. Modal transition systems
3. Property refutation and verification
4. Semantic foundations
5. Three instances of framework

Outline

1. Partial systems
2. Modal transition systems
3. Property refutation and verification
4. Semantic foundations
5. Three instances of framework
    (a) extended transition systems

Outline

1. Partial systems
2. Modal transition systems
3. Property refutation and verification
4. Semantic foundations
5. Three instances of framework
   (a) extended transition systems
   (b) partial Kripke structures

Outline

1. Partial systems
2. Modal transition systems
3. Property refutation and verification
4. Semantic foundations
5. Three instances of framework
   (a) extended transition systems
   (b) partial Kripke structures
   (c) modal shape graphs

Outline

1. Partial systems
2. Modal transition systems
3. Property refutation and verification
4. Semantic foundations
5. Three instances of framework
   (a) extended transition systems
   (b) partial Kripke structures
   (c) modal shape graphs
6. Conclusions and future work

1 Partial systems

Specification & analysis models are inherently *uncertain*. E.g.,

1 Partial systems

Specification & analysis models are inherently *uncertain*. E.g.,

- non-determinism present in concurrent programs;

1 Partial systems

Specification & analysis models are inherently *uncertain*. E.g.,

- non-determinism present in concurrent programs;
- "loose" system specifications; and

1 Partial systems

Specification & analysis models are inherently *uncertain*. E.g.,

- non-determinism present in concurrent programs;
- "loose" system specifications; and
- imprecise control-flow graphs for inter-procedural flow analysis.

1 Partial systems

Specification & analysis models are inherently *uncertain*. E.g.,

- non-determinism present in concurrent programs;
- "loose" system specifications; and
- imprecise control-flow graphs for inter-procedural flow analysis.

Specification formalisms (Z, UML, OCL, CFGs, etc) require "explicit" uncertainty to allow *partial* specification & analysis.

1 Partial systems

Specification & analysis models are inherently *uncertain*. E.g.,

- non-determinism present in concurrent programs;
- "loose" system specifications; and
- imprecise control-flow graphs for inter-procedural flow analysis.

Specification formalisms (Z, UML, OCL, CFGs, etc) require "explicit" uncertainty to allow *partial* specification & analysis.

Modal transition systems express such aspects and generalize Dams's work (Dams 1996) to *partial* systems.

2 Modal transition systems

(Larsen & Thomsen 1988):

- note that state transition diagrams and bisimulation are too restrictive for refinement and implementation work;

2 Modal transition systems

(Larsen & Thomsen 1988):

- note that state transition diagrams and bisimulation are too restrictive for refinement and implementation work;
- propose systems whose state-transitions have *modalities*:

2 Modal transition systems

(Larsen & Thomsen 1988):

- note that state transition diagrams and bisimulation are too restrictive for refinement and implementation work;
- propose systems whose state-transitions have *modalities*:
  - an a-transition from state s to s' must occur (has to be realized in *all* refinements): $s \rightarrow^{a}_{\Box} s'$.

2 Modal transition systems

(Larsen & Thomsen 1988):

- note that state transition diagrams and bisimulation are too restrictive for refinement and implementation work;
- propose systems whose state-transitions have *modalities*:
  - an a-transition from state s to s' must occur (has to be realized in *all* refinements): $s \rightarrow^{a}_{\Box} s'$.
  - an a-transition from state s to s' may occur (has to be realized in *some* refinement; *uncertainty*): $s \rightarrow^{a}_{\Diamond} s'$;

2 Modal transition systems

(Larsen & Thomsen 1988):

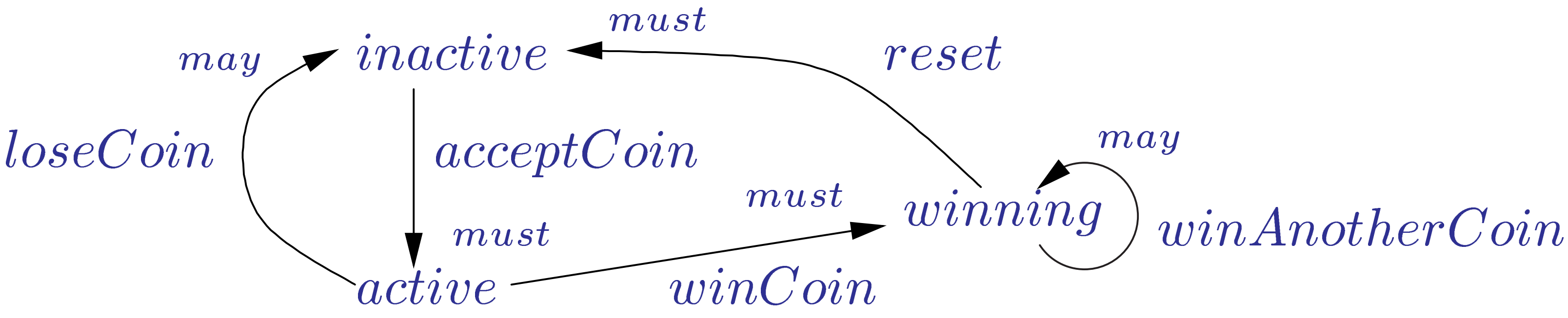
- note that state transition diagrams and bisimulation are too restrictive for refinement and implementation work;
- propose systems whose state-transitions have *modalities*:
  - an a-transition from state s to s' must occur (has to be realized in *all* refinements): $s \rightarrow^{a}_{\Box} s'$.
  - an a-transition from state s to s' may occur (has to be realized in *some* refinement; *uncertainty*): $s \rightarrow^{a}_{\Diamond} s'$;
  - an a-transition from state s to s' must not occur (has to be absent from all refinements): not $s \rightarrow^{a}_{\Diamond} s'$;

2 Modal transition systems

(Larsen & Thomsen 1988):

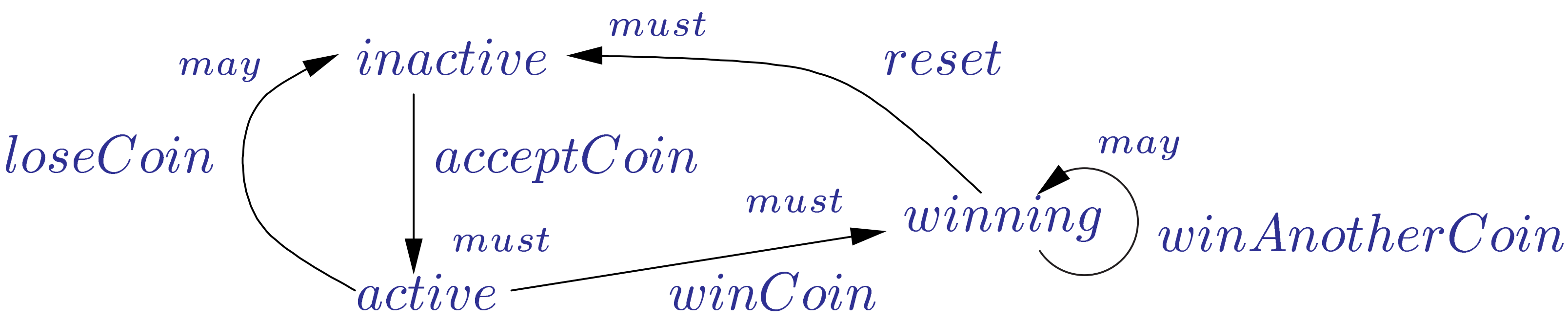
- note that state transition diagrams and bisimulation are too restrictive for refinement and implementation work;
- propose systems whose state-transitions have *modalities*:
  - an a-transition from state s to s' must occur (has to be realized in *all* refinements): $s \to^{a}_{\Box} s'$.
  - an a-transition from state s to s' may occur (has to be realized in *some* refinement; *uncertainty*): $s \to^{a}_{\Diamond} s'$;
  - an a-transition from state s to s' must not occur (has to be absent from all refinements): not $s \to^{a}_{\Diamond} s'$;
- impose a consistency condition: $\to_{\Box} \subseteq \to_{\Diamond}$ (all required behavior is also possible).

A slot machine: specification

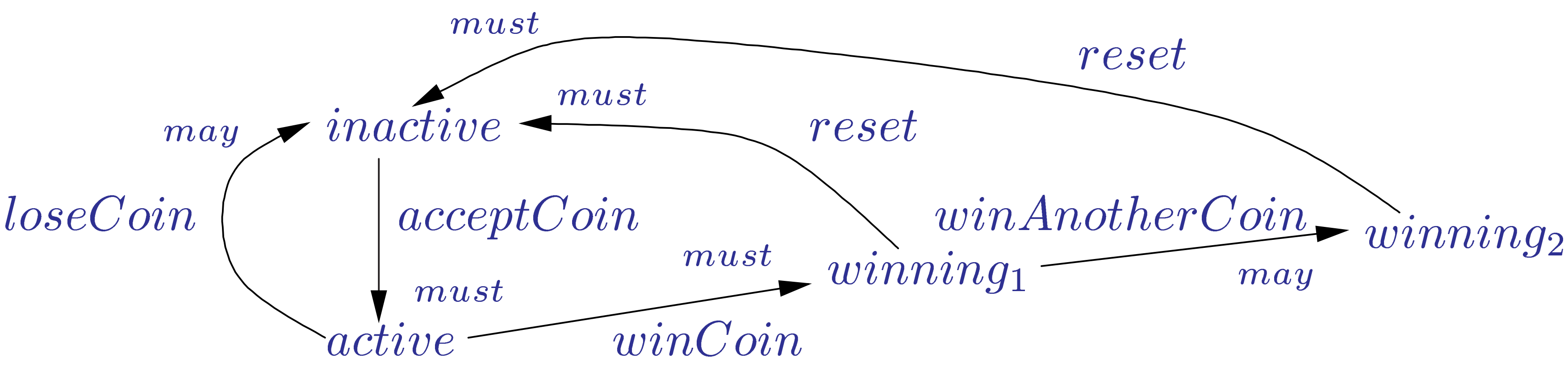


Consistency condition: *must*-transitions comprise $\rightarrow_\Box$, *may*- and *must*-transitions comprise $\rightarrow_\Diamond$.

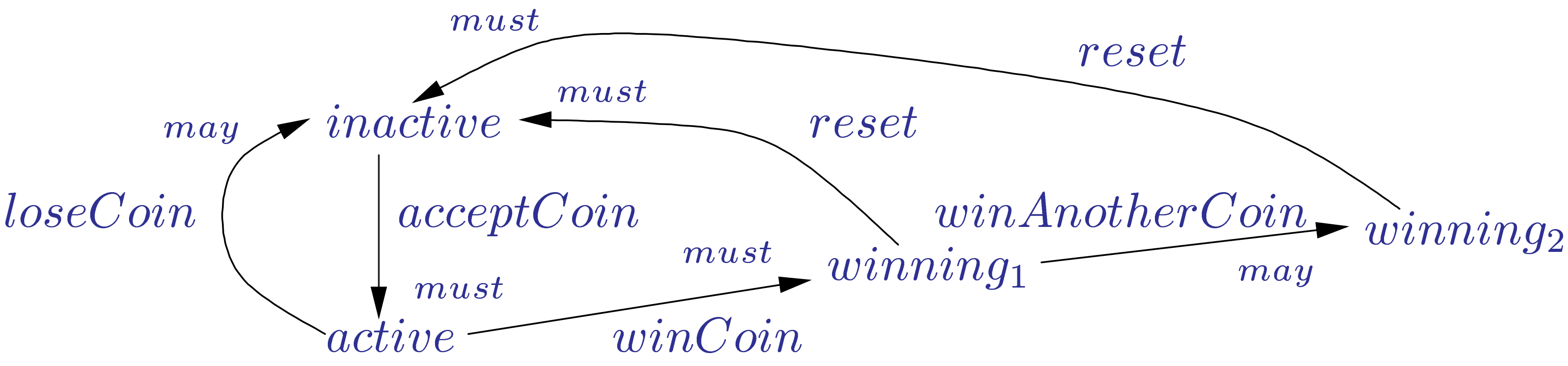
A slot machine: specification and refinement



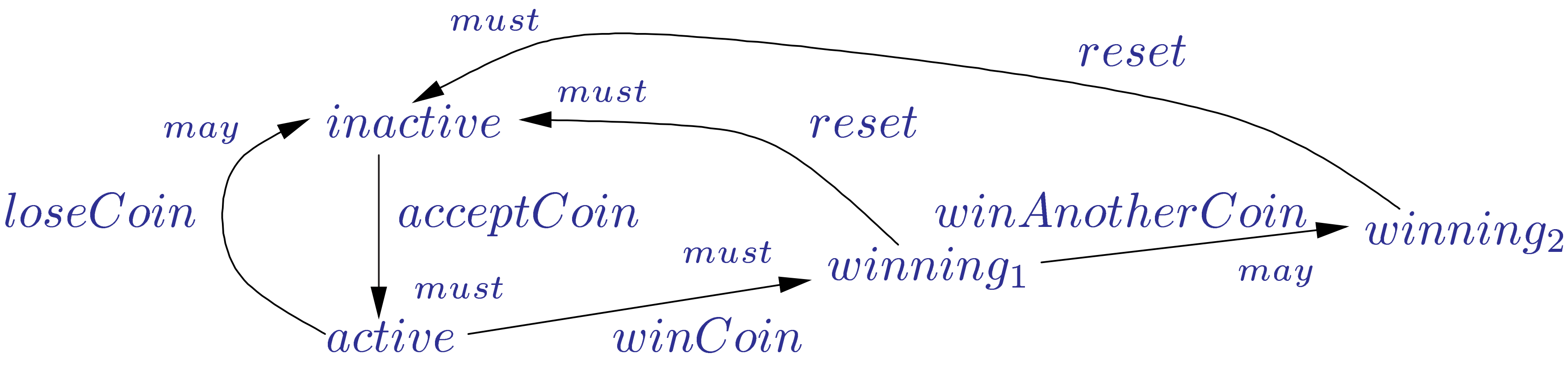
Consistency condition: *must*-transitions comprise $\rightarrow_{\Box}$, *may*- and *must*-transitions comprise $\rightarrow_{\Diamond}$.



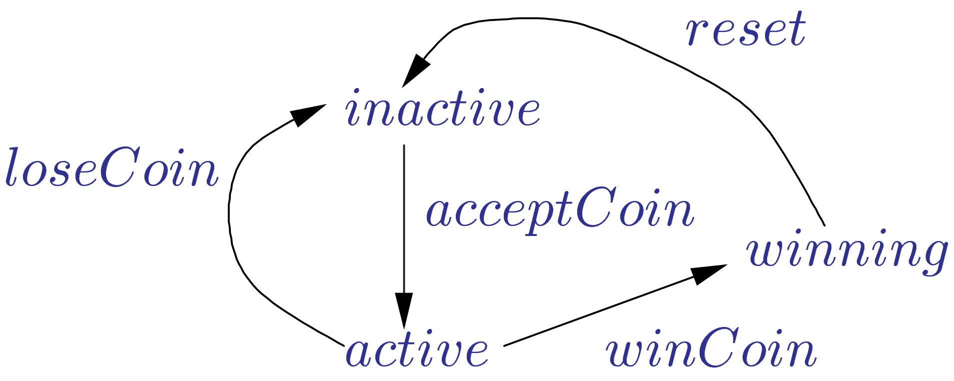
A slot machine: refinement



A slot machine: refinement and implementation



In implementations, all transitions are *must*-transitions:



Formal refinement and abstraction

State s *refines* state t (and t *abstracts* s) iff

Formal refinement and abstraction

State s *refines* state t (and t *abstracts* s) iff

1. (Larsen 1989) whenever $t \rightarrow^{a}_{\Box} t'$, there exists some s' that refines t' such that $s \rightarrow^{a}_{\Box} s'$;

Formal refinement and abstraction

State s *refines* state t (and t *abstracts* s) iff

1. (Larsen 1989) whenever $t \rightarrow^{a}_{\Box} t'$, there exists some s' that refines t' such that $s \rightarrow^{a}_{\Box} s'$;

2. (Larsen 1989) whenever $s \rightarrow^{a}_{\Diamond} s'$, there exists some t' that is refined by s' such that $t \rightarrow^{a}_{\Diamond} t'$;

Formal refinement and abstraction

State s *refines* state t (and t *abstracts* s) iff

1. (Larsen 1989) whenever $t \to_{\Box}^{a} t'$, there exists some s' that refines t' such that $s \to_{\Box}^{a} s'$;

2. (Larsen 1989) whenever $s \to_{\Diamond}^{a} s'$, there exists some t' that is refined by s' such that $t \to_{\Diamond}^{a} t'$;

3. $L_{\Box}(t) \subseteq L_{\Box}(s)$; and

4. $L_{\Diamond}(s) \subseteq L_{\Diamond}(t)$;

Formal refinement and abstraction

State s *refines* state t (and t *abstracts* s) iff

1. (Larsen 1989) whenever $t \rightarrow^{a}_{\Box} t'$, there exists some s' that refines t' such that $s \rightarrow^{a}_{\Box} s'$;

2. (Larsen 1989) whenever $s \rightarrow^{a}_{\Diamond} s'$, there exists some t' that is refined by s' such that $t \rightarrow^{a}_{\Diamond} t'$;

3. $L_{\Box}(t) \subseteq L_{\Box}(s)$; and

4. $L_{\Diamond}(s) \subseteq L_{\Diamond}(t)$;

where $L_{\Box}(t)$ ($L_{\Diamond}(t)$) are those state-propositions that must (may) be true at t.

Formal refinement and abstraction

State s *refines* state t (and t *abstracts* s) iff

1. (Larsen 1989) whenever $t \rightarrow^a_\Box t'$, there exists some s' that refines t' such that $s \rightarrow^a_\Box s'$;
2. (Larsen 1989) whenever $s \rightarrow^a_\Diamond s'$, there exists some t' that is refined by s' such that $t \rightarrow^a_\Diamond t'$;
3. $L_\Box(t) \subseteq L_\Box(s)$; and
4. $L_\Diamond(s) \subseteq L_\Diamond(t)$;

where $L_\Box(t)$ ($L_\Diamond(t)$) are those state-propositions that must (may) be true at t.

We write $\prec$ for the greatest such relation.

3 Property refutation and verification

Two satisfaction relations:

3 Property refutation and verification

Two satisfaction relations:

- $t \models^{must} \phi$: under-approximates "*all* refinements of t satisfy ϕ";

3 Property refutation and verification

Two satisfaction relations:

- $t \models^{must} \phi$: under-approximates "*all* refinements of t satisfy ϕ";

- $t \models^{may} \phi$: over-approximates "*some* refinement of t satisfies ϕ".

3 Property refutation and verification

Two satisfaction relations:

- $t \models^{must} \phi$: under-approximates "*all* refinements of t satisfy ϕ";

- $t \models^{may} \phi$: over-approximates "*some* refinement of t satisfies ϕ".

Sound framework for refutation and verification:

3 Property refutation and verification

Two satisfaction relations:

- $t \models^{must} \phi$: under-approximates "*all* refinements of t satisfy ϕ";

- $t \models^{may} \phi$: over-approximates "*some* refinement of t satisfies ϕ".

Sound framework for refutation and verification:

- To *verify* ϕ at state t, establish $t \models^{must} \phi$.

3 Property refutation and verification

Two satisfaction relations:

- $t \models^{must} \phi$: under-approximates "*all* refinements of t satisfy ϕ";

- $t \models^{may} \phi$: over-approximates "*some* refinement of t satisfies ϕ".

Sound framework for refutation and verification:

- To *verify* ϕ at state t, establish $t \models^{must} \phi$.
- To *refute* ϕ at state t, establish $t \models^{must} \neg\phi$.

3 Property refutation and verification

Two satisfaction relations:

- $t \models^{must} \phi$: under-approximates "*all* refinements of t satisfy ϕ";

- $t \models^{may} \phi$: over-approximates "*some* refinement of t satisfies ϕ".

Sound framework for refutation and verification:

- To *verify* ϕ at state t, establish $t \models^{must} \phi$.
- To *refute* ϕ at state t, establish $t \models^{must} \neg\phi$.

Related work: Cousot & Cousot (2000) define a *universal* and an *existential* abstraction; the latter one, used *in isolation* restricts sound reasoning to *safety* properties.

Temporal logic

Kozen's modal mu-calculus (Kozen 1983) (subsumes tool logics):

$$\phi ::= \bot \mid p \mid Z \mid \neg\phi \mid \phi \wedge \phi \mid \langle a \rangle\, \phi \mid \mu Z.\phi$$

Temporal logic

Kozen's modal mu-calculus (Kozen 1983) (subsumes tool logics):

$$\phi ::= \bot \mid p \mid Z \mid \neg\phi \mid \phi \wedge \phi \mid \langle a \rangle\, \phi \mid \mu Z.\phi$$

Semantic highlights — negation and $\models^{may}$ *not* considered in (Larsen 1989):

Temporal logic

Kozen's modal mu-calculus (Kozen 1983) (subsumes tool logics):

$$\phi ::= \bot \mid p \mid Z \mid \neg\phi \mid \phi \wedge \phi \mid \langle a \rangle\, \phi \mid \mu Z.\phi$$

Semantic highlights — negation and $\models^{may}$ *not* considered in (Larsen 1989):

- $s \models^{must} p$ iff $p \in L_{\Box}(s)$;

Temporal logic

Kozen's modal mu-calculus (Kozen 1983) (subsumes tool logics):

$$\phi ::= \bot \mid p \mid Z \mid \neg\phi \mid \phi \wedge \phi \mid \langle a \rangle\, \phi \mid \mu Z.\phi$$

Semantic highlights — negation and $\models^{may}$ *not* considered in (Larsen 1989):

- $s \models^{must} p$ iff $p \in L_{\Box}(s)$;
- $s \models^{must} \neg\phi$ iff not $s \models^{may} \phi$ (mutual induction: Kelb 1994);

Temporal logic

Kozen's modal mu-calculus (Kozen 1983) (subsumes tool logics):

$$\phi ::= \bot \mid p \mid Z \mid \neg\phi \mid \phi \wedge \phi \mid \langle a \rangle\, \phi \mid \mu Z.\phi$$

Semantic highlights — negation and $\models^{may}$ *not* considered in (Larsen 1989):

- $s \models^{must} p$ iff $p \in L_{\Box}(s)$;
- $s \models^{must} \neg\phi$ iff not $s \models^{may} \phi$ (mutual induction: Kelb 1994);
- $s \models^{must} \langle a \rangle\, \phi$ iff for some s', $s \rightarrow^{a}_{\Box} s'$ and $s' \models^{must} \phi$.

Temporal logic

Kozen's modal mu-calculus (Kozen 1983) (subsumes tool logics):

$$\phi ::= \bot \mid p \mid Z \mid \neg\phi \mid \phi \wedge \phi \mid \langle a \rangle\, \phi \mid \mu Z.\phi$$

Semantic highlights — negation and $\models^{may}$ *not* considered in (Larsen 1989):

- $s \models^{must} p$ iff $p \in L_{\Box}(s)$;
- $s \models^{must} \neg\phi$ iff not $s \models^{may} \phi$ (mutual induction: Kelb 1994);
- $s \models^{must} \langle a \rangle\, \phi$ iff for some s', $s \rightarrow^{a}_{\Box} s'$ and $s' \models^{must} \phi$.

The definition of $\models^{may}$ is as for $\models^{must}$, but we swap $must \leftrightarrow may$ and $\Box \leftrightarrow \Diamond$.

E.g., $s \models^{may} \langle a \rangle\, \phi$ iff for some s', $s \rightarrow^{a}_{\Diamond} s'$ and $s' \models^{may} \phi$.

Sound refutation and verification

For modal transition systems:

Sound refutation and verification

For modal transition systems:

- $t \models^{must} \phi$ and $s \prec t$ imply $s \models^{must} \phi$ (sound verification);

Sound refutation and verification

For modal transition systems:

- $t \models^{must} \phi$ and $s \prec t$ imply $s \models^{must} \phi$ (sound verification);
- $s \models^{may} \phi$ and $s \prec t$ imply $t \models^{may} \phi$ (sound refutation);

Sound refutation and verification

For modal transition systems:

- $t \models^{must} \phi$ and $s \prec t$ imply $s \models^{must} \phi$ (sound verification);
- $s \models^{may} \phi$ and $s \prec t$ imply $t \models^{may} \phi$ (sound refutation);
- not $t \models^{must} \phi \wedge \neg\phi$ (consistency); and

Sound refutation and verification

For modal transition systems:

- $t \models^{must} \phi$ and $s \prec t$ imply $s \models^{must} \phi$ (sound verification);
- $s \models^{may} \phi$ and $s \prec t$ imply $t \models^{may} \phi$ (sound refutation);
- not $t \models^{must} \phi \wedge \neg\phi$ (consistency); and
- $t \models^{may} \phi \vee \neg\phi$ ("completeness").

Sound refutation and verification

For modal transition systems:

- $t \models^{must} \phi$ and $s \prec t$ imply $s \models^{must} \phi$ (sound verification);
- $s \models^{may} \phi$ and $s \prec t$ imply $t \models^{may} \phi$ (sound refutation);
- not $t \models^{must} \phi \wedge \neg\phi$ (consistency); and
- $t \models^{may} \phi \vee \neg\phi$ ("completeness").

Important application: verifying

$$t \models^{must} \phi \rightarrow \psi$$

Sound refutation and verification

For modal transition systems:

- $t \models^{must} \phi$ and $s \prec t$ imply $s \models^{must} \phi$ (sound verification);
- $s \models^{may} \phi$ and $s \prec t$ imply $t \models^{may} \phi$ (sound refutation);
- not $t \models^{must} \phi \wedge \neg\phi$ (consistency); and
- $t \models^{may} \phi \vee \neg\phi$ ("completeness").

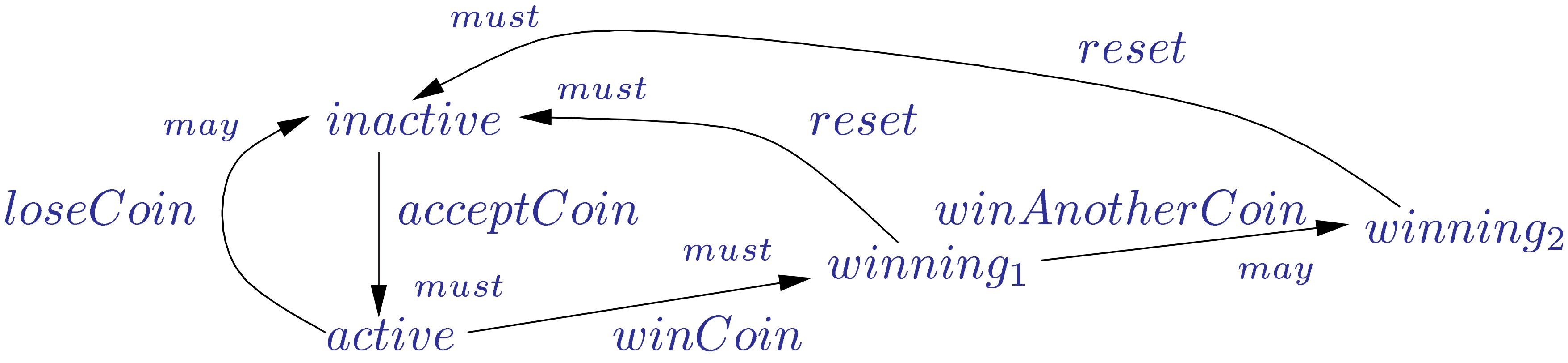
Important application: verifying

$$t \models^{must} \phi \to \psi$$

means either refuting $t \models^{may} \phi$ or verifying $t \models^{must} \psi$.

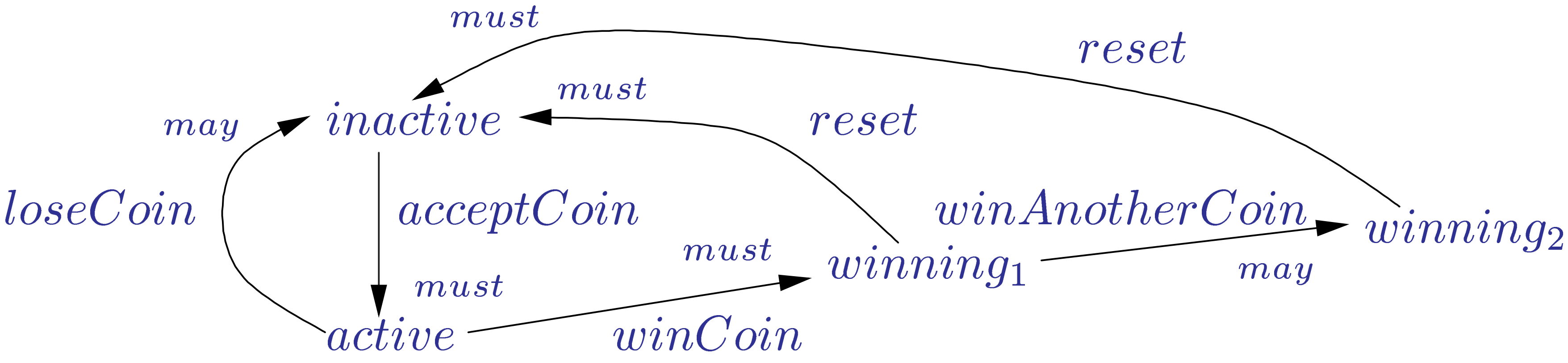
Example refutation

We don't have $\text{inactive} \models^{must} \mathtt{EF}\,(\text{winning} \wedge \mathtt{EX}\,\text{winning})$:

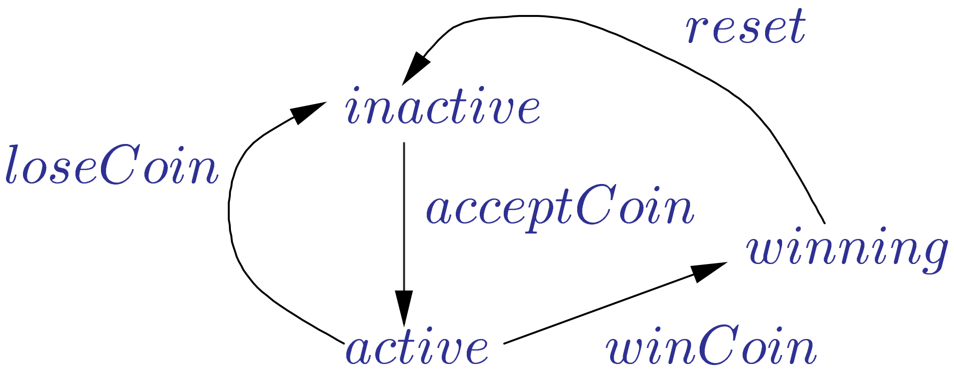


Example refutation

We don't have $\text{inactive} \models^{must} \mathtt{EF}\,(\text{winning} \wedge \mathtt{EX}\,\text{winning})$:

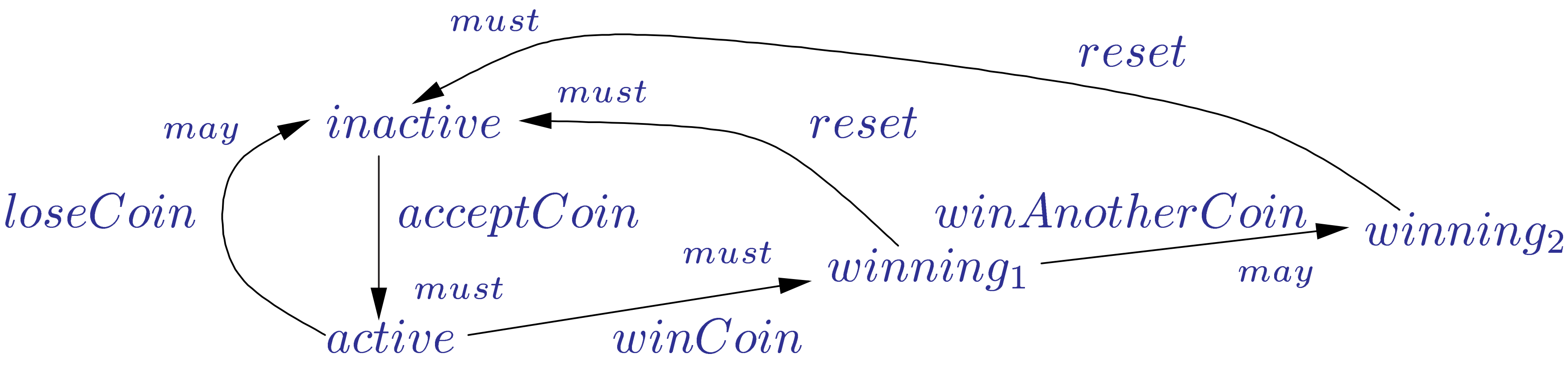


This makes sense as we have
$\text{inactive} \models^{must} \neg\mathtt{EF}\,(\text{winning} \wedge \mathtt{EX}\,\text{winning})$:



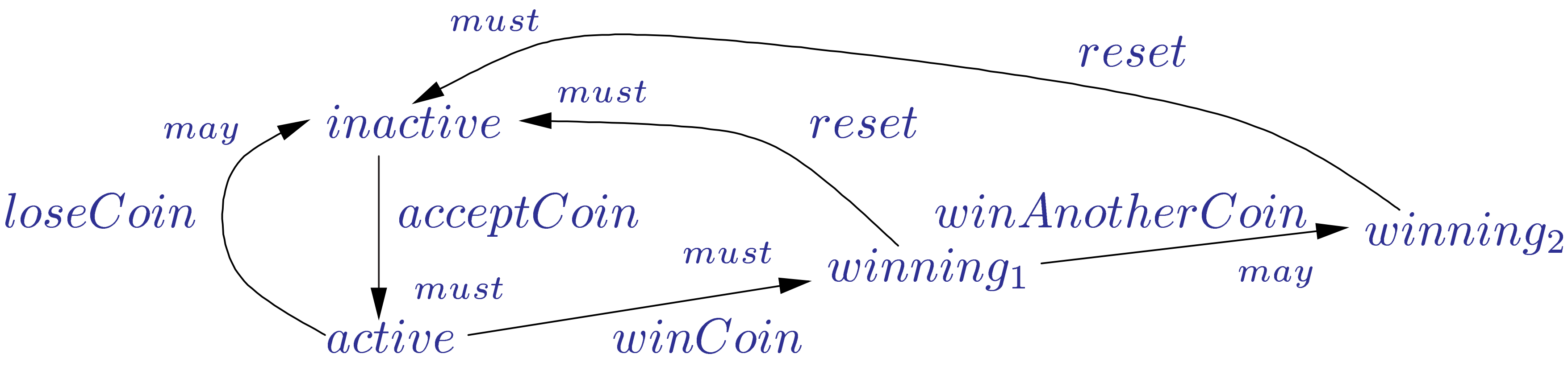
Example verification

We have inactive $\models^{must}$ $\mathtt{EF}$ (winning):

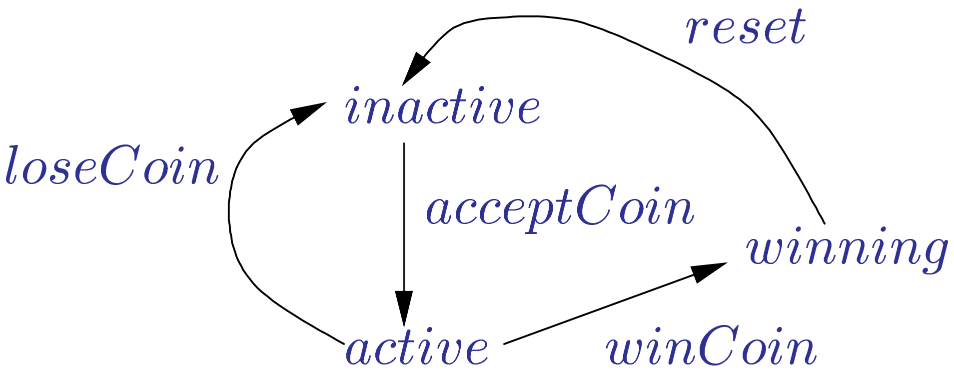


Example verification

We have inactive $\models^{must}$ $\mathtt{EF}$ (winning):



So this property holds for all implementations, in particular for



4 Semantic foundations

Mixed power domain (Gunter 1992 & Heckmann 1993):

$$\mathbf{M}[D] \stackrel{\text{def}}{=} \{(L, U) \in \mathcal{L}[D] \times \mathcal{U}[D] \mid L = {\downarrow}(L \cap U)\}.$$

4 Semantic foundations

Mixed power domain (Gunter 1992 & Heckmann 1993):

$$\mathbf{M}[D] \stackrel{\text{def}}{=} \{(L, U) \in \mathcal{L}[D] \times \mathcal{U}[D] \mid L = \,\downarrow\!(L \cap U)\}.$$

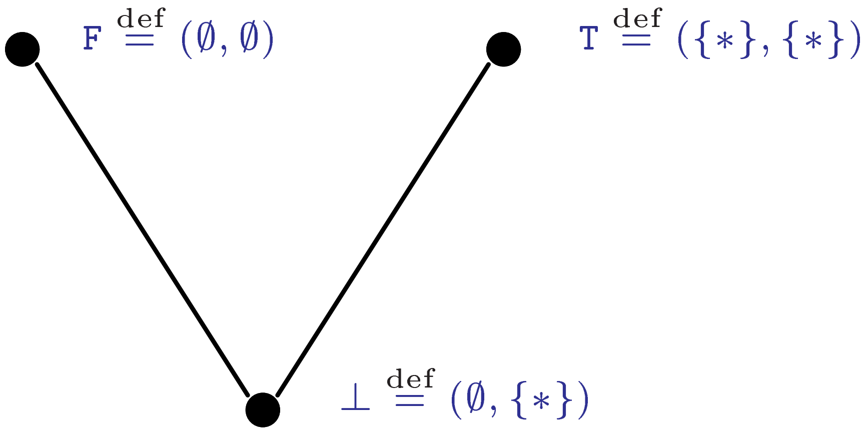
For example, the mixed power domain of $\{*\}$ equals

4 Semantic foundations

Mixed power domain (Gunter 1992 & Heckmann 1993):

$$\mathbf{M}[D] \stackrel{\text{def}}{=} \{(L, U) \in \mathcal{L}[D] \times \mathcal{U}[D] \mid L = {\downarrow}(L \cap U)\}.$$

For example, the mixed power domain of $\{*\}$ equals

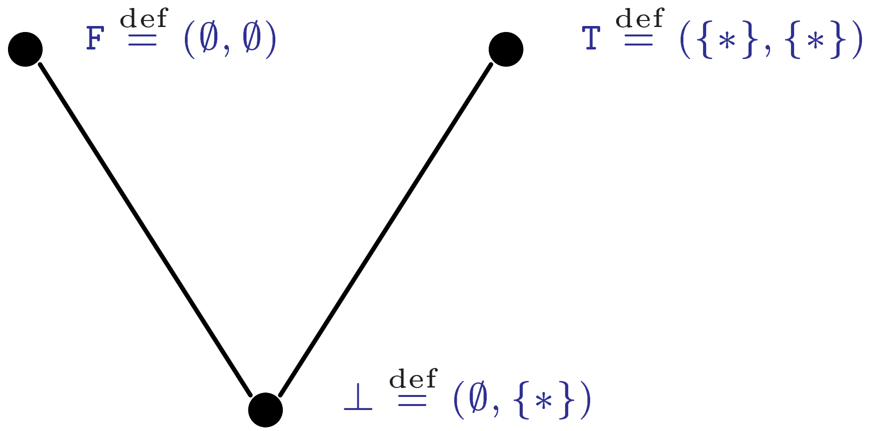


4 Semantic foundations

Mixed power domain (Gunter 1992 & Heckmann 1993):

$$\mathbf{M}[D] \stackrel{\text{def}}{=} \{(L, U) \in \mathcal{L}[D] \times \mathcal{U}[D] \mid L = {\downarrow}(L \cap U)\}.$$

For example, the mixed power domain of $\{*\}$ equals



(The pair $(\{*\}, \emptyset)$ violates the consistency condition.)

A domain equation for modal transition systems

$$\mathcal{D} \cong \prod_{a \in \mathtt{Act}} \mathbf{M}[\mathcal{D}] \qquad \text{(no propositions)}$$

A domain equation for modal transition systems

$$\mathcal{D} \cong \prod_{a \in \mathtt{Act}} \mathbf{M}[\mathcal{D}] \qquad \text{(no propositions)}$$

For $d = (L_a, U_a)_{a \in \mathtt{Act}} \in \mathcal{D}$:

A domain equation for modal transition systems

$$\mathcal{D} \cong \prod_{a \in \mathtt{Act}} \mathbf{M}[\mathcal{D}] \qquad \text{(no propositions)}$$

For $d = (L_a, U_a)_{a \in \mathtt{Act}} \in \mathcal{D}$:

- $d \rightarrow^a_\diamond d'$ iff $d' \in L_a$

A domain equation for modal transition systems

$$\mathcal{D} \cong \prod_{a \in \mathtt{Act}} \mathbf{M}[\mathcal{D}] \qquad \text{(no propositions)}$$

For $d = (L_a, U_a)_{a \in \mathtt{Act}} \in \mathcal{D}$:

- $d \rightarrow^a_\Diamond d'$ iff $d' \in L_a$
- $d \rightarrow^a_\Box d'$ iff $d' \in U_a$.

A domain equation for modal transition systems

$$\mathcal{D} \cong \prod_{a \in \mathtt{Act}} \mathbf{M}[\mathcal{D}] \qquad \text{(no propositions)}$$

For $d = (L_a, U_a)_{a \in \mathtt{Act}} \in \mathcal{D}$:

- $d \rightarrow^a_\Diamond d'$ iff $d' \in L_a$
- $d \rightarrow^a_\Box d'$ iff $d' \in U_a$.

1. Internal full abstraction: $\leq$ is greatest abstraction relation.

A domain equation for modal transition systems

$$\mathcal{D} \cong \prod_{a \in \mathtt{Act}} \mathbf{M}[\mathcal{D}] \qquad \text{(no propositions)}$$

For $d = (L_a, U_a)_{a \in \mathtt{Act}} \in \mathcal{D}$:

- $d \to^a_\diamond d'$ iff $d' \in L_a$
- $d \to^a_\Box d'$ iff $d' \in U_a$.

1. Internal full abstraction: $\leq$ is greatest abstraction relation.
2. Parametric abstraction of non-flat data *and* control possible.

A domain equation for modal transition systems

$$\mathcal{D} \cong \prod_{a \in \mathtt{Act}} \mathbf{M}[\mathcal{D}] \qquad \text{(no propositions)}$$

For $d = (L_a, U_a)_{a \in \mathtt{Act}} \in \mathcal{D}$:

- $d \rightarrow^a_\Diamond d'$ iff $d' \in L_a$
- $d \rightarrow^a_\Box d'$ iff $d' \in U_a$.

1. Internal full abstraction: $\leq$ is greatest abstraction relation.
2. Parametric abstraction of non-flat data *and* control possible.
3. Domain equations may reflect quantitative aspects, such as time, probability, or cost.

A domain equation for modal transition systems

$$\mathcal{D} \cong \prod_{a \in \mathtt{Act}} \mathbf{M}[\mathcal{D}] \qquad \text{(no propositions)}$$

For $d = (L_a, U_a)_{a \in \mathtt{Act}} \in \mathcal{D}$:

- $d \to^a_\Diamond d'$ iff $d' \in L_a$
- $d \to^a_\Box d'$ iff $d' \in U_a$.

1. Internal full abstraction: $\leq$ is greatest abstraction relation.
2. Parametric abstraction of non-flat data *and* control possible.
3. Domain equations may reflect quantitative aspects, such as time, probability, or cost.
4. Domain-theoretic machinery a powerful tool for reasoning about higher-order and intuitionstic aspects.

5 Three instances of framework

(a) Extended transition systems (Milner 1981)

- are labeled transition systems $\mathcal{E} = (\Sigma_E, \mathtt{Act}, \rightarrow)$ together with a *divergence relation* $\uparrow\ \subseteq \Sigma_E \times \mathtt{Act}$;

5 Three instances of framework

(a) Extended transition systems (Milner 1981)

- are labeled transition systems $\mathcal{E} = (\Sigma_E, \texttt{Act}, \rightarrow)$ together with a *divergence relation* $\uparrow \subseteq \Sigma_E \times \texttt{Act}$;

- we write $s \downarrow a$ (intuition: "all a-transitions out of s are present") when $s \uparrow a$ fails to hold;

5 Three instances of framework

(a) Extended transition systems (Milner 1981)

- are labeled transition systems $\mathcal{E} = (\Sigma_E, \mathtt{Act}, \rightarrow)$ together with a *divergence relation* $\uparrow \subseteq \Sigma_E \times \mathtt{Act}$;
- we write $s \downarrow a$ (intuition: "all a-transitions out of s are present") when $s \uparrow a$ fails to hold;
- $\mathcal{E}$ as a modal transition system $\mathcal{M}[\mathcal{E}]$:

$$
\begin{array}{lll}
s \rightarrow_a^{must} s' & \text{iff} & s \rightarrow_a s' \\
s \rightarrow_a^{may} s' & \text{iff} & s \rightarrow_a s' \text{ or } s \uparrow a.
\end{array}
$$

Partial bisimulation as abstraction

1. Greatest partial bisimulation (Milner 1981) on $\mathcal{E}$ is greatest abstraction on $\mathcal{M}[\mathcal{E}]$.

Partial bisimulation as abstraction

1. Greatest partial bisimulation (Milner 1981) on $\mathcal{E}$ is greatest abstraction on $\mathcal{M}[\mathcal{E}]$.

2. The "intuitionistic" semantics of Hennessy-Milner logic for $s \in \Sigma_E$ matches $\models^{must}$ for $\mathcal{M}[\mathcal{E}]$.

Partial bisimulation as abstraction

1. Greatest partial bisimulation (Milner 1981) on $\mathcal{E}$ is greatest abstraction on $\mathcal{M}[\mathcal{E}]$.

2. The "intuitionistic" semantics of Hennessy-Milner logic for $s \in \Sigma_E$ matches $\models^{must}$ for $\mathcal{M}[\mathcal{E}]$.

Conclusion: The models, abstraction, and temporal logic semantics for partial bisimulation are a special instance of the corresponding notions for modal transition systems.

(b) Partial Kripke structures

- Bruns & Godefroid (1999) abstract propositional information at states ($L(s, p) \in \{\mathtt{F}, \mathtt{T}\}$) with a three-valued logic $\mathbf{3} \stackrel{\text{def}}{=} \{\perp < \mathtt{F}, \mathtt{T}\}$;

(b) Partial Kripke structures

- Bruns & Godefroid (1999) abstract propositional information at states ($L(s,p) \in \{\mathtt{F}, \mathtt{T}\}$) with a three-valued logic $\mathbf{3} \stackrel{\text{def}}{=} \{\perp < \mathtt{F}, \mathtt{T}\}$;

- define an abstraction relation w.r.t. the order on $\mathbf{3}$; and

(b) Partial Kripke structures

- Bruns & Godefroid (1999) abstract propositional information at states ($L(s, p) \in \{\mathtt{F}, \mathtt{T}\}$) with a three-valued logic $\mathbf{3} \stackrel{\text{def}}{=} \{\perp < \mathtt{F}, \mathtt{T}\}$;
- define an abstraction relation w.r.t. the order on $\mathbf{3}$; and
- model check CTL formulas over such partial state spaces.

(b) Partial Kripke structures

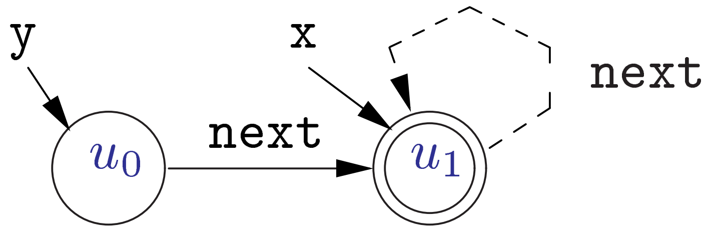
- Bruns & Godefroid (1999) abstract propositional information at states ($L(s,p) \in \{\mathtt{F}, \mathtt{T}\}$) with a three-valued logic $\mathbf{3} \stackrel{\text{def}}{=} \{\bot < \mathtt{F}, \mathtt{T}\}$;
- define an abstraction relation w.r.t. the order on $\mathbf{3}$; and
- model check CTL formulas over such partial state spaces.
- The order-isomorphism

$$
\begin{array}{rcll}
\mathbf{M}[\Sigma] & \cong & \Sigma \to \mathbf{3} & \qquad (\Sigma \text{ discrete}) \\
(f^{-1}\{\text{true}\}, f^{-1}\{\bot, \text{true}\}) & \cong & f &
\end{array}
$$

  recaptures their models, abstraction relation, and CTL semantics as an instance of the corresponding notions of modal transition systems.

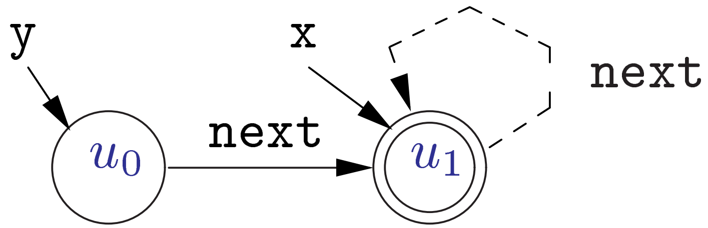
(c) Modal shape graphs

Shape analysis: contents of heap storage is approximated by a graph. The graph



(c) Modal shape graphs

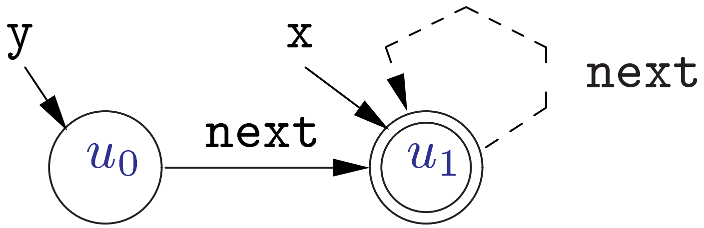
Shape analysis: contents of heap storage is approximated by a graph. The graph



- approximates a singly linked list of length at least two.

(c) Modal shape graphs

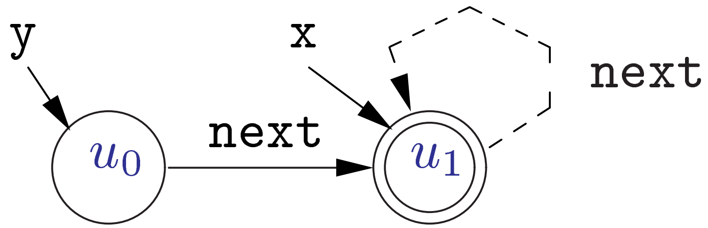
Shape analysis: contents of heap storage is approximated by a graph. The graph



- approximates a singly linked list of length at least two.
- Local variables, `x` and `y`, point to objects.

(c) Modal shape graphs

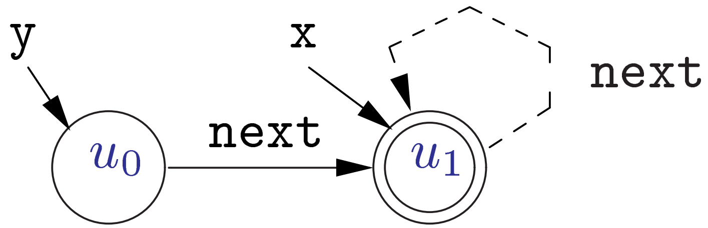
Shape analysis: contents of heap storage is approximated by a graph. The graph



- approximates a singly linked list of length at least two.
- Local variables, `x` and `y`, point to objects.
- Objects have arcs to `next` fields; solid ones are *must*-, dotted ones are *may*-transitions.

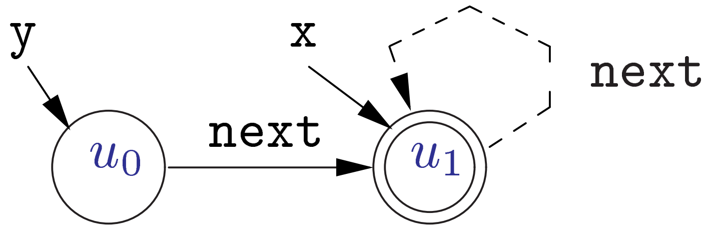
(c) Modal shape graphs

Shape analysis: contents of heap storage is approximated by a graph. The graph

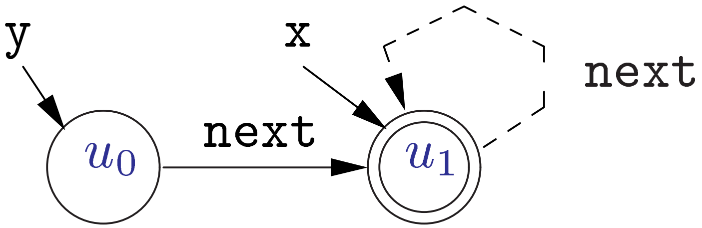


- approximates a singly linked list of length at least two.
- Local variables, `x` and `y`, point to objects.
- Objects have arcs to **`next`** fields; solid ones are *must*-, dotted ones are *may*-transitions.
- Double-circled objects are "summary nodes", possibly representing more than one object.

The shape graph



The shape graph and its coding via predicates

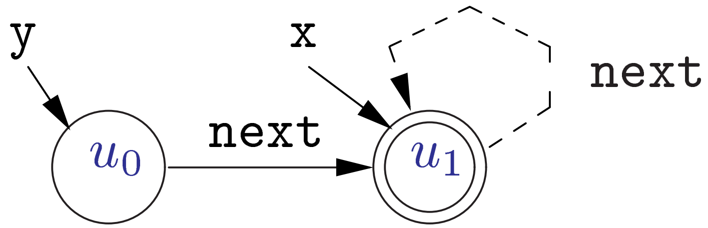


M. Sagiv, T. Reps, and R. Wilhelm (1999):

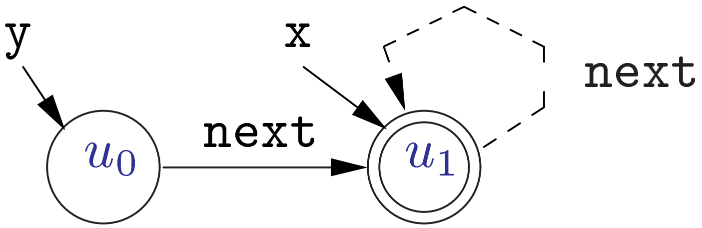
(Kleene ordering $0 < 1/2 < 1$)

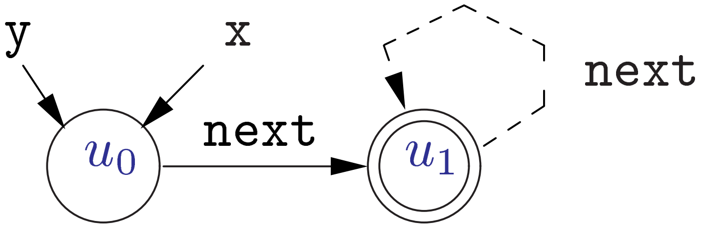
$$
\begin{array}{ll}
x(u_0) = 0 & next(u_0, u_0) = 0 \\
x(u_1) = 1 & next(u_0, u_1) = 1 \\
y(u_0) = 1 & next(u_1, u_0) = 0 \\
y(u_1) = 0 & next(u_1, u_1) = 1/2 \\
sm(u_0) = 0 & \\
sm(u_1) = 1 &
\end{array}
$$

The same shape graph

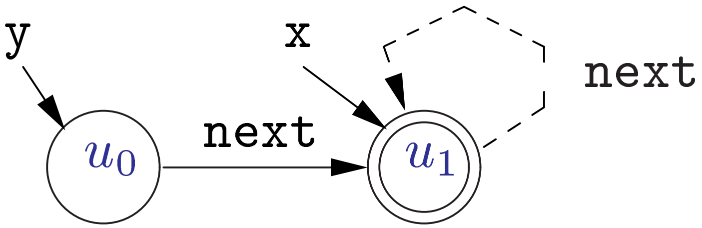


The same shape graph and the effect of `x = y`

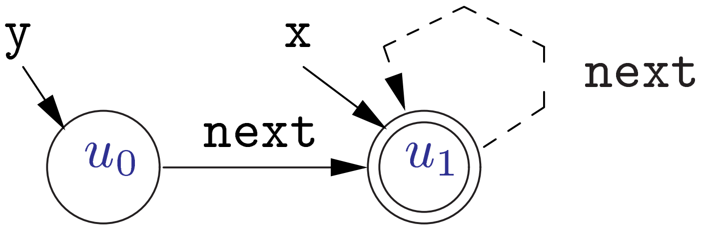


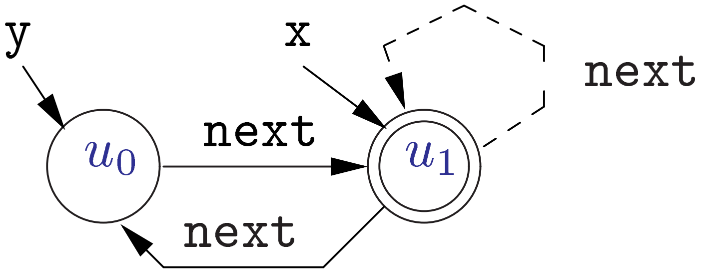


The shape graph



The shape graph and the effect of `x.next = y`





Notice: The *may*-transition out of the summary node persists.

Recasting parts of three-valued shape analysis

(M. Sagiv, T. Reps, and R. Wilhelm 1999): in the MTS framework

- their embedding theorem expresses soundness of verification;

Recasting parts of three-valued shape analysis

(M. Sagiv, T. Reps, and R. Wilhelm 1999): in the MTS framework

- their embedding theorem expresses soundness of verification;
- their `focus` operation, `instrumentation predicates`, and `coerce` operation have connections to assume-guarantee reasoning and generalized model checking (Bruns & Godefroid 2000);

Recasting parts of three-valued shape analysis

(M. Sagiv, T. Reps, and R. Wilhelm 1999): in the MTS framework

- their embedding theorem expresses soundness of verification;
- their `focus` operation, `instrumentation predicates`, and `coerce` operation have connections to assume-guarantee reasoning and generalized model checking (Bruns & Godefroid 2000);
- plenty of practically important properties — such as shape invariants, direction relationship, interference relationship, aliasing, cycles, etc — have simple encodings in CTL or LTL.

Conclusions and future work

Modal transition systems

1. their abstraction notion, and temporal logic semantics are a robust and executable framework for a range of applications in the design and analysis of software.

Conclusions and future work

Modal transition systems

1. their abstraction notion, and temporal logic semantics are a robust and executable framework for a range of applications in the design and analysis of software.

2. allow for sound reasoning about partial or incomplete designs and programs, thereby recovering bisimulation techniques on total or complete designs.

Conclusions and future work

Modal transition systems

1. their abstraction notion, and temporal logic semantics are a robust and executable framework for a range of applications in the design and analysis of software.

2. allow for sound reasoning about partial or incomplete designs and programs, thereby recovering bisimulation techniques on total or complete designs.

3. exploit all four possible interactions of path quantifiers and state-transition modalities.

Conclusions and future work

Modal transition systems

1. their abstraction notion, and temporal logic semantics are a robust and executable framework for a range of applications in the design and analysis of software.

2. allow for sound reasoning about partial or incomplete designs and programs, thereby recovering bisimulation techniques on total or complete designs.

3. exploit all four possible interactions of path quantifiers and state-transition modalities.

4. await novel applications, e.g., compositional pointer analysis (Whaley & Rinard 1999) and computing abstract models (Godefroid, Huth, and Jagadeesan 2001).