Pushing runtime verification to the limit

May process semantics be with us

Dario Della Monica[1] Adrian Francalanza[2]

[1]**University of Udine, Italy**
dario.dellamonica@uniud.it

[2]**University of Malta, Malta**
adrian.francalanza@um.edu.mt

OVERLAY 2019

Rende, November 19, 2019

Outline

A quick introduction to runtime verification (monitoring)

Monitoring *HML*

Extending runtime verification applicability
- A failed attempt
- A promising road using process semantics

Conclusions

Outline

A quick introduction to runtime verification (monitoring)

Monitoring *HML*

Extending runtime verification applicability
- A failed attempt
- A promising road using process semantics

Conclusions

Runtime verification

monitoring a single partial execution and try to give a verdict

eventually p

Runtime verification

monitoring a single partial execution and try to give a verdict

eventually p $\neg p$

Runtime verification

monitoring a single partial execution and try to give a verdict

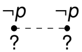
eventually p

$\neg p$
$\bullet$
?

Runtime verification

monitoring a single partial execution and try to give a verdict

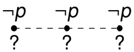
eventually p



Runtime verification

monitoring a single partial execution and try to give a verdict

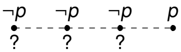
eventually p



Runtime verification

monitoring a single partial execution and try to give a verdict

eventually p



Runtime verification

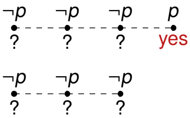
monitoring a single partial execution and try to give a verdict

	$\neg p$	$\neg p$	$\neg p$	p
eventually p	?	?	?	yes

Runtime verification

monitoring a single partial execution and try to give a verdict

eventually p



Runtime verification

monitoring a single partial execution and try to give a verdict

eventually p

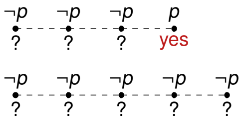
$\neg p$	$\neg p$	$\neg p$	p
?	?	?	yes

$\neg p$	$\neg p$	$\neg p$	$\neg p$
?	?	?	?

Runtime verification

monitoring a single partial execution and try to give a verdict

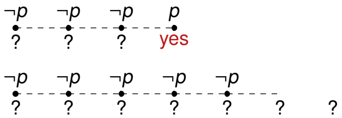
eventually p



Runtime verification

monitoring a single partial execution and try to give a verdict

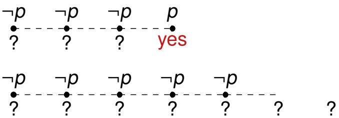
eventually p



Runtime verification

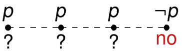
monitoring a single partial execution and try to give a verdict

eventually p



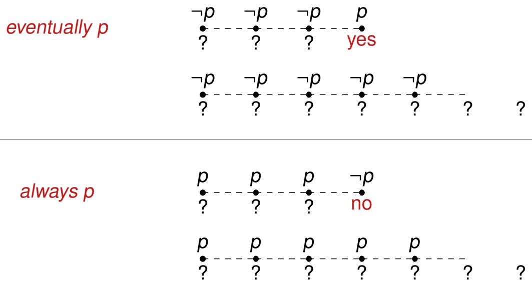
---

always p



Runtime verification

monitoring a single partial execution and try to give a verdict



Runtime verification

monitoring a single partial execution and try to give a verdict

eventually p

$\neg p$	$\neg p$	$\neg p$	p			
?	?	?	yes			

$\neg p$	$\neg p$	$\neg p$	$\neg p$	$\neg p$		
?	?	?	?	?	?	?

---

always p

p	p	p	$\neg p$			
?	?	?	no			

p	p	p	p	p		
?	?	?	?	?	?	?

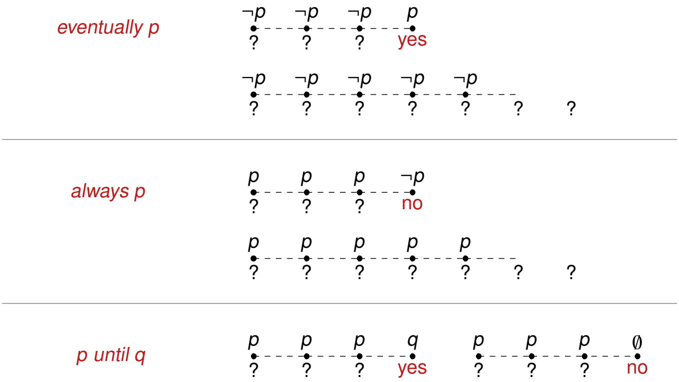
---

p until q

p	p	p	q
?	?	?	yes

Runtime verification

monitoring a single partial execution and try to give a verdict



Runtime verification

monitoring a single partial execution and try to give a verdict

Monitorability

Definition (monitorability)

φ is monitorable $:=$ φ is suitable to be runtime verified

$:=$ either

there exists a **witness for φ-satisfaction** whenever φ is true

or

there exists a **witness for φ-violation** whenever φ is false

witness for φ-satisfaction: finite trace s.t. every system featuring it satisfies φ

witness for φ-violation: finite trace s.t. every system featuring it violates φ

Outline

A quick introduction to runtime verification (monitoring)

Monitoring *HML*

Extending runtime verification applicability

- A failed attempt
- A promising road using process semantics

Conclusions

The branching time logic *HML*

$\varphi, \phi \in \mathit{HML} ::=$

tt	(truth)	$\mid \mathsf{ff}$	(falsehood)
$\mid \varphi \vee \phi$	(disjunction)	$\mid \varphi \wedge \phi$	(conjunction)
$\mid \langle\alpha\rangle\varphi$	(possibility)	$\mid [\alpha]\varphi$	(necessity)

The maximal monitorable subset [†]

$$\pi, \varpi \in \text{cHML} ::= \mathsf{tt} \quad \mid \mathsf{ff} \quad \mid \pi \vee \varpi \quad \mid \langle\alpha\rangle\pi$$

$$\theta, \vartheta \in \text{sHML} ::= \mathsf{tt} \quad \mid \mathsf{ff} \quad \mid \theta \wedge \vartheta \quad \mid [\alpha]\theta$$

[†] Francalanza, Aceto, Ingólfsdóttir, *On verifying Hennessy-Milner logic with recursion at runtime*. In **Runtime Verification**, 2015.

The branching time logic *HML*

$\varphi, \phi \in \mathit{HML} ::=$

tt	(truth)	$\mid$ ff	(falsehood)
$\mid\ \varphi \vee \phi$	(disjunction)	$\mid\ \varphi \wedge \phi$	(conjunction)
$\mid\ \langle\alpha\rangle\varphi$	(possibility)	$\mid\ [\alpha]\varphi$	(necessity)

The maximal monitorable subset [†]

$\pi, \varpi \in$ cHML::= tt $\mid$ ff $\mid \pi \vee \varpi$ $\mid \langle\alpha\rangle\pi$

$\theta, \vartheta \in$ sHML::= tt $\mid$ ff $\mid \theta \wedge \vartheta$ $\mid [\alpha]\theta$

[†] Francalanza, Aceto, Ingólfsdóttir, *On verifying Hennessy-Milner logic with recursion at runtime*. In **Runtime Verification**, 2015.

Monitorability: examples

The maximal monitorable subset

$$\pi, \varpi \in \text{cHML} ::= \text{tt} \quad | \; \text{ff} \quad | \; \pi \vee \varpi \quad | \; \langle \alpha \rangle \pi$$

$$\theta, \vartheta \in \text{sHML} ::= \text{tt} \quad | \; \text{ff} \quad | \; \theta \wedge \vartheta \quad | \; [\alpha] \theta$$

Examples

$\langle a \rangle \text{tt}$

positively monitorable

Monitorability: examples

The maximal monitorable subset

$$\pi, \varpi \in \mathsf{cHML} ::= \mathsf{tt} \quad | \; \mathsf{ff} \quad | \; \pi \vee \varpi \quad | \; \langle \alpha \rangle \pi$$
$$\theta, \vartheta \in \mathsf{sHML} ::= \mathsf{tt} \quad | \; \mathsf{ff} \quad | \; \theta \wedge \vartheta \quad | \; [\alpha] \theta$$

Examples

$\langle a \rangle \mathsf{tt}$ — positively monitorable

$[a]\mathsf{ff}$ — negatively monitorable

Monitorability: examples

The maximal monitorable subset

$$\pi, \varpi \in \mathsf{cHML} ::= \mathsf{tt} \quad | \ \mathsf{ff} \quad | \ \pi \vee \varpi \quad | \ \langle \alpha \rangle \pi$$

$$\theta, \vartheta \in \mathsf{sHML} ::= \mathsf{tt} \quad | \ \mathsf{ff} \quad | \ \theta \wedge \vartheta \quad | \ [\alpha] \theta$$

Examples

$\langle a \rangle \mathsf{tt}$ — positively monitorable

$[a]\mathsf{ff}$ — negatively monitorable

$\langle a \rangle \mathsf{tt} \vee \langle b \rangle \mathsf{tt}$ — positively monitorable

Monitorability: examples

The maximal monitorable subset

$$\pi, \varpi \in \mathsf{cHML} ::= \mathsf{tt} \quad | \ \mathsf{ff} \quad | \ \pi \vee \varpi \quad | \ \langle \alpha \rangle \pi$$

$$\theta, \vartheta \in \mathsf{sHML} ::= \mathsf{tt} \quad | \ \mathsf{ff} \quad | \ \theta \wedge \vartheta \quad | \ [\alpha] \theta$$

Examples

$\langle a \rangle \mathsf{tt}$ — positively monitorable

$[a]\mathsf{ff}$ — negatively monitorable

$\langle a \rangle \mathsf{tt} \vee \langle b \rangle \mathsf{tt}$ — positively monitorable

$\langle a \rangle [b] \mathsf{ff}$

Monitorability: examples

The maximal monitorable subset

$$\pi, \varpi \in \mathsf{cHML} ::= \mathsf{tt} \quad | \ \mathsf{ff} \quad | \ \pi \vee \varpi \quad | \ \langle \alpha \rangle \pi$$

$$\theta, \vartheta \in \mathsf{sHML} ::= \mathsf{tt} \quad | \ \mathsf{ff} \quad | \ \theta \wedge \vartheta \quad | \ [\alpha]\theta$$

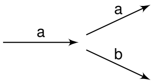
Examples

$\langle a \rangle \mathsf{tt}$ — positively monitorable

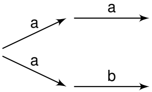
$[a]\mathsf{ff}$ — negatively monitorable

$\langle a \rangle \mathsf{tt} \vee \langle b \rangle \mathsf{tt}$ — positively monitorable

$\langle a \rangle [b]\mathsf{ff}$



$\langle a \rangle [b]\mathsf{ff}$ is false



$\langle a \rangle [b]\mathsf{ff}$ is true

Monitorability: examples

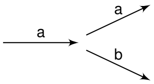
The maximal monitorable subset

$$\pi, \varpi \in \mathrm{cHML} ::= \mathrm{tt} \quad | \ \mathrm{ff} \quad | \ \pi \vee \varpi \quad | \ \langle \alpha \rangle \pi$$

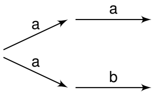
$$\theta, \vartheta \in \mathrm{sHML} ::= \mathrm{tt} \quad | \ \mathrm{ff} \quad | \ \theta \wedge \vartheta \quad | \ [\alpha] \theta$$

Examples

$\langle a \rangle \mathrm{tt}$	$[a]\mathrm{ff}$	$\langle a \rangle \mathrm{tt} \vee \langle b \rangle \mathrm{tt}$	$\cancel{\langle a \rangle [b] \mathrm{ff}}$
positively monitorable	negatively monitorable	positively monitorable	not monitorable



$\langle a \rangle [b]\mathrm{ff}$ is false



$\langle a \rangle [b]\mathrm{ff}$ is true

Limitations of monitoring

$\langle a \rangle(\langle b \rangle \mathrm{tt} \vee [c]\mathrm{ff})$: not monitorable, do model checking

Limitations of monitoring

$\langle a\rangle(\langle b\rangle \mathrm{tt} \vee [c]\mathrm{ff})$: not monitorable, do model checking

$$\langle a\rangle(\langle b\rangle \mathrm{tt} \vee [c]\mathrm{ff}) \;\equiv\; \langle a\rangle\langle b\rangle \mathrm{tt} \;\vee\; \langle a\rangle[c]\mathrm{ff}$$

Limitations of monitoring

$\langle a \rangle (\langle b \rangle \mathrm{tt} \vee [c]\mathrm{ff})$: not monitorable, do model checking

$$\langle a \rangle (\langle b \rangle \mathrm{tt} \vee [c]\mathrm{ff}) \equiv \underbrace{\langle a \rangle \langle b \rangle \mathrm{tt}}_{\text{monitorable: do runtime verification}} \vee \underbrace{\langle a \rangle [c]\mathrm{ff}}_{\text{not monitorable: do model checking}}$$

Outline

A quick introduction to runtime verification (monitoring)

Monitoring *HML*

Extending runtime verification applicability
- A failed attempt
- A promising road using process semantics

Conclusions

The problem: Maximal monitorable semantic sub-formula

Skip

Input: a formula φ of HML

$\psi ::= \mathsf{tt} \mid \mathsf{ff} \mid \psi \vee \psi \mid \psi \wedge \psi \mid \langle\alpha\rangle\psi \mid [\alpha]\psi$

Output: φ^{MON} (a maximal monitorable semantic sub-formula of φ)

The problem: Maximal monitorable semantic sub-formula

Skip

$\psi ::= \mathsf{tt} \mid \mathsf{ff} \mid \psi \vee \psi \mid \psi \wedge \psi \mid \langle \alpha \rangle \psi \mid [\alpha] \psi$

Input: a formula φ of HML

Output: φ^{MON} (a maximal monitorable semantic sub-formula of φ)

- monitorable: $\varphi^{MON} \in cHML$

$\psi ::= \mathsf{tt} \mid \mathsf{ff} \mid \psi \vee \psi \mid \langle \alpha \rangle \psi$

The problem: Maximal monitorable semantic sub-formula

Skip

$\psi ::= \mathsf{tt} \mid \mathsf{ff} \mid \psi\vee\psi \mid \psi\wedge\psi \mid \langle\alpha\rangle\psi \mid [\alpha]\psi$

Input: a formula φ of HML

Output: φ^{MON} (a maximal monitorable semantic sub-formula of φ)

$\psi ::= \mathsf{tt} \mid \mathsf{ff} \mid \psi\vee\psi \mid \langle\alpha\rangle\psi$

- monitorable: $\varphi^{MON} \in cHML$
- semantic fragment (every process that satisfies φ^{MON}, also satisfies φ): $[\![\varphi^{MON}]\!] \subseteq [\![\varphi]\!]$

The problem: Maximal monitorable semantic sub-formula

Skip

Input: a formula φ of HML

$\psi ::= \mathrm{tt} \mid \mathrm{ff} \mid \psi \vee \psi \mid \psi \wedge \psi \mid \langle \alpha \rangle \psi \mid [\alpha] \psi$

Output: φ^{MON} (a maximal monitorable semantic sub-formula of φ)

$\psi ::= \mathrm{tt} \mid \mathrm{ff} \mid \psi \vee \psi \mid \langle \alpha \rangle \psi$

- monitorable: $\varphi^{MON} \in cHML$
- semantic fragment (every process that satisfies φ^{MON}, also satisfies φ): $[\![\varphi^{MON}]\!] \subseteq [\![\varphi]\!]$
- maximal: $\forall \psi. [\![\psi]\!] \subseteq [\![\varphi]\!] \rightarrow [\![\psi]\!] \subseteq [\![\varphi^{MON}]\!]$

The problem: Maximal monitorable semantic sub-formula

Skip

$\psi ::= \mathrm{tt} \mid \mathrm{ff} \mid \psi \vee \psi \mid \psi \wedge \psi \mid \langle \alpha \rangle \psi \mid [\alpha]\psi$

Input: a formula φ of HML

Output: φ^{MON} (a maximal monitorable semantic sub-formula of φ)

$\psi ::= \mathrm{tt} \mid \mathrm{ff} \mid \psi \vee \psi \mid \langle \alpha \rangle \psi$

- monitorable: $\varphi^{MON} \in cHML$
- semantic fragment (every process that satisfies φ^{MON}, also satisfies φ): $[\![\varphi^{MON}]\!] \subseteq [\![\varphi]\!]$
- maximal: $\forall \psi. [\![\psi]\!] \subseteq [\![\varphi]\!] \rightarrow [\![\psi]\!] \subseteq [\![\varphi^{MON}]\!]$

Then $\varphi \equiv \varphi^{MON} \vee \varphi_{|\varphi^{MON}}$ where
- φ^{MON} is monitorable
- $\varphi_{|\varphi^{MON}}$ must be model checked

Outline

A quick introduction to runtime verification (monitoring)

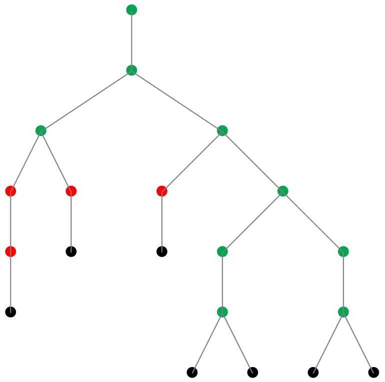
Monitoring *HML*

Extending runtime verification applicability
- A failed attempt
- A promising road using process semantics

Conclusions

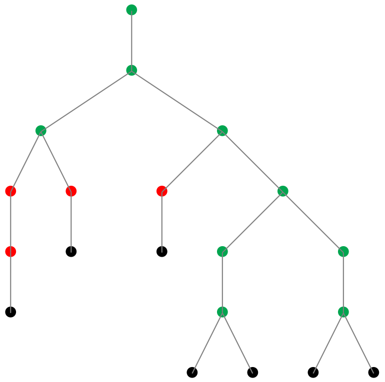
The decomposition procedure

1. Identify highest universal nodes
2. Remove subtrees rooted in highest universal nodes
3. Remove new leaves



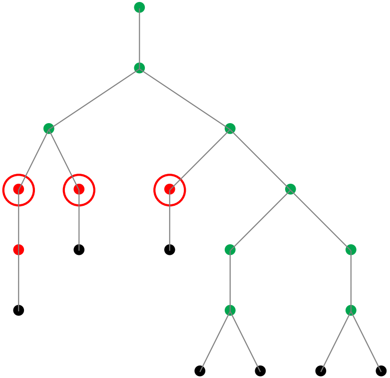
The decomposition procedure

1. **Identify highest universal nodes**
2. Remove subtrees rooted in highest universal nodes
3. Remove new leaves



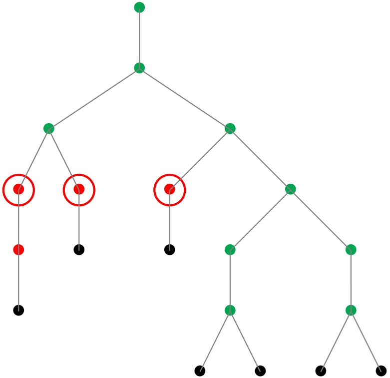
The decomposition procedure

1. Identify highest universal nodes
2. Remove subtrees rooted in highest universal nodes
3. Remove new leaves



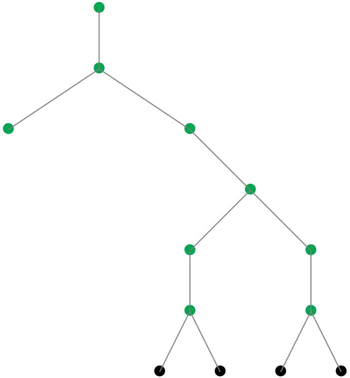
The decomposition procedure

1. Identify highest universal nodes
2. **Remove subtrees rooted in highest universal nodes**
3. Remove new leaves



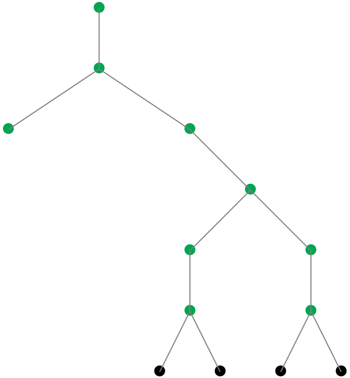
The decomposition procedure

1. Identify highest universal nodes
2. Remove subtrees rooted in highest universal nodes
3. Remove new leaves



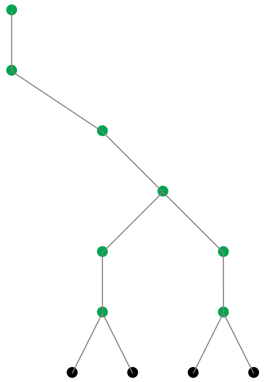
The decomposition procedure

1. Identify highest universal nodes
2. Remove subtrees rooted in highest universal nodes
3. **Remove new leaves**



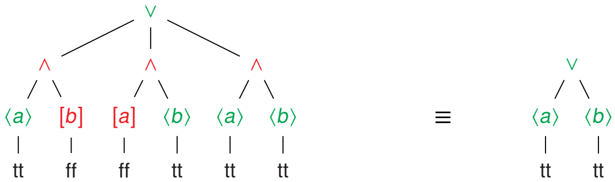
The decomposition procedure

1. Identify highest universal nodes
2. Remove subtrees rooted in highest universal nodes
3. Remove new leaves



A counterexample

$$(\langle a\rangle tt \wedge [b]ff) \vee ([a]ff \wedge \langle b\rangle tt) \vee (\langle a\rangle tt \wedge \langle b\rangle tt) \quad \equiv \quad \langle a\rangle tt \vee \langle b\rangle tt$$



A counterexample

$$(\langle a\rangle \mathrm{tt} \wedge [b]\mathrm{ff}) \vee ([a]\mathrm{ff} \wedge \langle b\rangle \mathrm{tt}) \vee (\langle a\rangle \mathrm{tt} \wedge \langle b\rangle \mathrm{tt}) \quad \equiv \quad \langle a\rangle \mathrm{tt} \vee \langle b\rangle \mathrm{tt}$$

$\vee$

$$\equiv \quad \begin{array}{ccc} & \vee & \\ / & & \backslash \\ \langle a\rangle & & \langle b\rangle \\ | & & | \\ \mathrm{tt} & & \mathrm{tt} \end{array}$$

A counterexample

$$(\langle a\rangle\mathrm{tt}\wedge[b]\mathrm{ff})\vee([a]\mathrm{ff}\wedge\langle b\rangle\mathrm{tt})\vee(\langle a\rangle\mathrm{tt}\wedge\langle b\rangle\mathrm{tt}) \quad \equiv \quad \langle a\rangle\mathrm{tt}\vee\langle b\rangle\mathrm{tt}$$

$$\equiv \quad \begin{array}{ccc} & \vee & \\ / & & \backslash \\ \langle a\rangle & & \langle b\rangle \\ | & & | \\ \mathrm{tt} & & \mathrm{tt} \end{array}$$

Outline

A quick introduction to runtime verification (monitoring)

Monitoring *HML*

Extending runtime verification applicability

- A failed attempt
- A promising road using process semantics

Conclusions

Modal transition systems might hold the answer

- Our approach was purely syntactic
- Formulas are semantics object but difficult to manipulate
- Syntactic trees of formulas can be manipulated but they are... well... too syntactic
- We needed a representation of formulas that
  - is easy to manipulate
  - carries semantics information about formulas and their relationship

Modal transition systems might hold the answer

- Our approach was purely syntactic
- Formulas are semantics object but difficult to manipulate
- Syntactic trees of formulas can be manipulated but they are... well... too syntactic
- We needed a representation of formulas that
  - is easy to manipulate
  - carries semantics information about formulas and their relationship

Then... process semantics

... but which one?

Modal transition systems might hold the answer

- Our approach was purely syntactic
- Formulas are semantics object but difficult to manipulate
- Syntactic trees of formulas can be manipulated but they are... well... too syntactic
- We needed a representation of formulas that
  - is easy to manipulate
  - carries semantics information about formulas and their relationship

Then... process semantics

... but which one?

Modal transition systems (MTS)

Why MTS?

- Several process semantics (simulation, trace, bisimulation, ...)
- Branching-time... still many of them (from simulation up to bisimulation)
- Complete for *HML*: bisimulation is a possible candidate

Why MTS?

- Several process semantics (simulation, trace, bisimulation, ...)
- Branching-time... still many of them (from simulation up to bisimulation)
- Complete for *HML*: bisimulation is a possible candidate
  - problem: **all** characteristic formulas are out of the monitorable fragment
  - problem: bisimulation induces equivalence (rather than preorder) relations over processes (LTS's)
  - you can only talk of single processes (up to bisimilarity)

Why MTS?

- Several process semantics (simulation, trace, bisimulation, ...)
- Branching-time... still many of them (from simulation up to bisimulation)
- Complete for *HML*: bisimulation is a possible candidate
    - problem: **all** characteristic formulas are out of the monitorable fragment
    - problem: bisimulation induces equivalence (rather than preorder) relations over processes (LTS's)
    - you can only talk of single processes (up to bisimilarity)
- A more suitable graphical representation of formulas of *HML* is given by modal transition systems (MTS's) and modal refinement defined over them

Why MTS?

- Several process semantics (simulation, trace, bisimulation, ...)
- Branching-time... still many of them (from simulation up to bisimulation)
- Complete for *HML*: bisimulation is a possible candidate
  - problem: **all** characteristic formulas are out of the monitorable fragment
  - problem: bisimulation induces equivalence (rather than preorder) relations over processes (LTS's)
  - you can only talk of single processes (up to bisimilarity)
- A more suitable graphical representation of formulas of *HML* is given by modal transition systems (MTS's) and modal refinement defined over them
  - every formula of *HML* is representable as a (finite set of) MTS
  - a translation back from MTS's into formulas also exists
  - modal refinement over MTS's is a preorder that carries the semantics information about formulas and their relationship

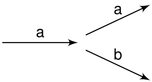
What are MTS's?

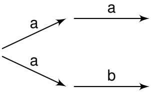
- Fix alphabet Σ
- An LTS is a pair $(P, \rightarrow)$, where
  - P is a finite set of processes
  - $\rightarrow \subseteq P \times \Sigma \times P$

What are MTS's?

- Fix alphabet Σ
- An LTS is a pair $(P, \rightarrow)$, where
  - P is a finite set of processes
  - $\rightarrow \subseteq P \times \Sigma \times P$

we focus on acyclic LTS
(as we consider *HML* rather than μ*HML*)

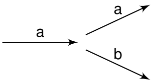


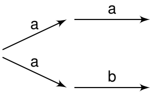


What are MTS's?

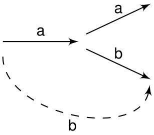
- Fix alphabet Σ
- An LTS is a pair $(P, \rightarrow)$, where
  - P is a finite set of processes
  - $\rightarrow \subseteq P \times \Sigma \times P$

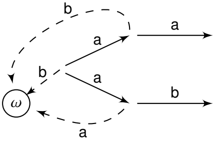
we focus on acyclic LTS
(as we consider *HML* rather than μ*HML*)

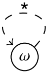




- An MTS is a triple $(P, \rightarrow, \dashrightarrow)$, where
  - P is a finite set of processes
  - $\rightarrow \subseteq P \times \Sigma \times P$
  - $\dashrightarrow \subseteq P \times \Sigma \times P$ and $\rightarrow \subseteq \dashrightarrow$







What is a modal refinement preorder relation?

Let M, M' be MTS's (processes)

M' is a refinement of M (denoted $M \sqsubseteq M'$) iff

- M' must do everything M must do ($\rightarrow$)
- M may do everything M' may do ($\dashrightarrow$)

What is a modal refinement preorder relation?

Let M, M' be MTS's (processes)

M' is a refinement of M (denoted $M \sqsubseteq M'$) iff

- M' must do everything M must do ($\longrightarrow$)
- M may do everything M' may do ($\dashrightarrow$)

is the weakest process ($\omega \sqsubseteq M'$ for every MTS M'), called the universal process and denoted by ω

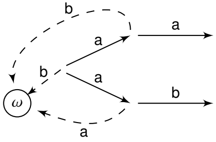
What is a modal refinement preorder relation?

Let M, M' be MTS's (processes)

M' is a refinement of M (denoted $M \sqsubseteq M'$) iff

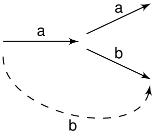
- M' must do everything M must do ($\rightarrow$)
- M may do everything M' may do ($\dashrightarrow$)

is the weakest process ($\omega \sqsubseteq M'$ for every MTS M'), called the universal process and denoted by ω



$\sqsubseteq$

$\not\sqsupseteq$

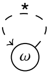


What is a modal refinement preorder relation?

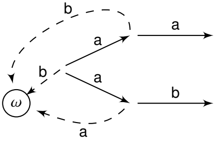
Let M, M' be MTS's (processes)

M' is a refinement of M (denoted $M \sqsubseteq M'$) iff

- M' must do everything M must do ($\rightarrow$)
- M may do everything M' may do ($\dashrightarrow$)

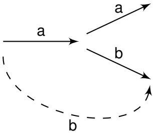


is the weakest process ($\omega \sqsubseteq M'$ for every MTS M'), called the universal process and denoted by ω



$\sqsubseteq$

$\not\sqsupseteq$



(we focus on acyclic MTS)

The problem: Maximal monitorable semantic sub-formula

$\psi ::= \mathrm{tt} \mid \mathrm{ff} \mid \psi \vee \psi \mid \psi \wedge \psi \mid \langle \alpha \rangle \psi \mid [\alpha] \psi$

Input: a formula φ of HML

Output: φ^{MON} (a maximal monitorable semantic sub-formula of φ)

The problem: Maximal monitorable semantic sub-formula

Input: a formula φ of HML

$\psi ::= \mathrm{tt} \mid \mathrm{ff} \mid \psi \vee \psi \mid \psi \wedge \psi \mid \langle\alpha\rangle\psi \mid [\alpha]\psi$

Output: φ^{MON} (a maximal monitorable semantic sub-formula of φ)

- monitorable: $\varphi^{MON} \in cHML$

$\psi ::= \mathrm{tt} \mid \mathrm{ff} \mid \psi \vee \psi \mid \langle\alpha\rangle\psi$

The problem: Maximal monitorable semantic sub-formula

Input: a formula φ of HML $\quad \psi ::= \mathsf{tt} \mid \mathsf{ff} \mid \psi \vee \psi \mid \psi \wedge \psi \mid \langle \alpha \rangle \psi \mid [\alpha]\psi$

Output: φ^{MON} (a maximal monitorable semantic sub-formula of φ)

- monitorable: $\varphi^{MON} \in cHML \quad \psi ::= \mathsf{tt} \mid \mathsf{ff} \mid \psi \vee \psi \mid \langle \alpha \rangle \psi$
- semantic fragment (every process that satisfies φ^{MON}, also satisfies φ): $[\![\varphi^{MON}]\!] \subseteq [\![\varphi]\!]$

The problem: Maximal monitorable semantic sub-formula

$\psi ::= \mathsf{tt} \mid \mathsf{ff} \mid \psi \vee \psi \mid \psi \wedge \psi \mid \langle \alpha \rangle \psi \mid [\alpha]\psi$

Input: a formula φ of HML

Output: φ^{MON} (a maximal monitorable semantic sub-formula of φ)

$\psi ::= \mathsf{tt} \mid \mathsf{ff} \mid \psi \vee \psi \mid \langle \alpha \rangle \psi$

- monitorable: $\varphi^{MON} \in cHML$
- semantic fragment (every process that satisfies φ^{MON}, also satisfies φ): $[\![\varphi^{MON}]\!] \subseteq [\![\varphi]\!]$
- maximal: $\forall \psi. [\![\psi]\!] \subseteq [\![\varphi]\!] \rightarrow [\![\psi]\!] \subseteq [\![\varphi^{MON}]\!]$

The problem: Maximal monitorable semantic sub-formula

$\psi ::= \mathsf{tt} \mid \mathsf{ff} \mid \psi \vee \psi \mid \psi \wedge \psi \mid \langle \alpha \rangle \psi \mid [\alpha]\psi$

Input: a formula φ of HML

Output: φ^{MON} (a maximal monitorable semantic sub-formula of φ)

$\psi ::= \mathsf{tt} \mid \mathsf{ff} \mid \psi \vee \psi \mid \langle \alpha \rangle \psi$

- monitorable: $\varphi^{MON} \in cHML$
- semantic fragment (every process that satisfies φ^{MON}, also satisfies φ): $[\![\varphi^{MON}]\!] \subseteq [\![\varphi]\!]$
- maximal: $\forall \psi . [\![\psi]\!] \subseteq [\![\varphi]\!] \rightarrow [\![\psi]\!] \subseteq [\![\varphi^{MON}]\!]$

Then $\varphi \equiv \varphi^{MON} \vee \varphi_{|\varphi^{MON}}$ where
- φ^{MON} is monitorable
- $\varphi_{|\varphi^{MON}}$ must be model checked

The solution idea

- $\varphi \mapsto \mathrm{MTS}(\varphi)$ [†] *(transform φ into a set of MTS's)*
- $\mathrm{MTS}(\varphi) = \{M_1, \ldots, M_n\}$ is a finite set of MTS s.t.

  $$M \text{ satisfies } \varphi \quad \text{iff} \quad M_i \sqsubseteq M \text{ for some } i$$

  for all MTS M

  (wlog. we can assume $\mathrm{MTS}(\varphi) = \{M_\varphi\}$ to be a singleton)

[†] Boudol, Larsen, *Graphical Versus Logical Specifications*. **Theor. Comput. Sci.** 106(1): 3-20 (1992).

The solution idea

- $\varphi \mapsto \mathrm{MTS}(\varphi)$ *(transform φ into a set of MTS's)*
- $\mathrm{MTS}(\varphi) = \{M_1, \ldots, M_n\}$ is a finite set of MTS s.t.

  $$M \text{ satisfies } \varphi \quad \text{iff} \quad M_i \sqsubseteq M \text{ for some } i$$

  for all MTS M

  (wlog. we can assume $\mathrm{MTS}(\varphi) = \{M_\varphi\}$ to be a singleton)
- PROBLEM: the logical representation of an MTS M (characteristic formula of M, denoted $\chi(M)$) is not guaranteed to be in cHML

The solution idea

- $\varphi \mapsto \text{MTS}(\varphi)$ *(transform φ into a set of MTS's)*
- $\text{MTS}(\varphi) = \{M_1, \ldots, M_n\}$ is a finite set of MTS s.t.

  M satisfies φ iff $M_i \sqsubseteq M$ for some i

  for all MTS M

  (wlog. we can assume $\text{MTS}(\varphi) = \{M_\varphi\}$ to be a singleton)
- PROBLEM: the logical representation of an MTS M (characteristic formula of M, denoted $\chi(M)$) is not guaranteed to be in cHML
- consider almost-universal MTS's, i.e.,
  - every state has may transitions to ω (1-step-universal):
    characteristic formulas are in $cHML + \wedge$
  - every state (except ω) has exactly one outgoing must transition

  characteristic formulas are in *cHML*

The solution idea (cont'd)

- Let *almost-un* be the set of almost-universal MTS's
- Let $\mathit{refinements}(M_\varphi)$ be the set of refinements of M_φ

The solution idea (cont'd)

- Let *almost-un* be the set of almost-universal MTS's
- Let *refinements*(M_φ) be the set of refinements of M_φ

Theorem (soundness – claim)

Let $M \in$ *almost-un be a refinement of* M_φ. *Then,* $[\![\chi(M)]\!] \subseteq [\![\varphi]\!]$

The solution idea (cont'd)

- Let *almost-un* be the set of almost-universal MTS's
- Let *refinements*(M_φ) be the set of refinements of M_φ

Theorem (soundness – claim)

Let $M \in$ almost-un be a refinement of M_φ. Then, $[\![\chi(M)]\!] \subseteq [\![\varphi]\!]$

Corollary

$$[\![\bigvee_{M \,\in\, \mathit{almost\text{-}un} \,\cap\, \mathit{refinements}(M_\varphi)} \chi(M)]\!] \subseteq [\![\varphi]\!]$$

The solution idea (cont'd)

- Let *almost-un* be the set of almost-universal MTS's
- Let *refinements*(M_φ) be the set of refinements of M_φ

Theorem (soundness – claim)

Let $M \in$ almost-un be a refinement of M_φ. Then, $[\![\chi(M)]\!] \subseteq [\![\varphi]\!]$

Corollary

$$[\![\bigvee_{M \,\in\, \mathit{almost\text{-}un} \,\cap\, \mathit{refinements}(M_\varphi)} \chi(M)]\!] \subseteq [\![\varphi]\!]$$

Theorem (finiteness – claim)

Let M be an MTS. If M is not almost-universal, then none of its refinements is almost-universal either

The solution idea (cont'd)

- Let *almost-un* be the set of almost-universal MTS's
- Let $\mathit{refinements}(M_\varphi)$ be the set of refinements of M_φ

Theorem (soundness – claim)

Let $M \in \mathit{almost\text{-}un}$ be a refinement of M_φ. Then, $[\![\chi(M)]\!] \subseteq [\![\varphi]\!]$

Corollary

$$[\![\bigvee_{M \,\in\, \mathit{almost\text{-}un} \,\cap\, \mathit{refinements}(M_\varphi)} \chi(M)]\!] \subseteq [\![\varphi]\!]$$

Theorem (finiteness – claim)

Let M be an MTS. If M is not almost-universal, then none of its refinements is almost-universal either

Corollary

$$[\![\bigvee_{M \,\in\, \mathit{almost\text{-}un} \,\cap\, \mathit{refinements}(M_\varphi)} \chi(M)]\!] = [\![\bigvee_{M \,\in\, \mathit{almost\text{-}un} \,\cap\, \{M_\varphi\}} \chi(M)]\!] \subseteq [\![\varphi]\!]$$

The solution idea (cont'd)

Maximality follows from a continuity property

Lemma (claim)

Let M be an almost-universal MTS. If all of its ultimate refinements (i.e., $\dashrightarrow = \longrightarrow$) satisfy an HML formula ψ, then M satisfies ψ, too

Corollary (maximality)

$\bigvee_{M \,\in\, \textit{almost-un} \,\cap\, \mathrm{MTS}(\varphi)} \chi(M)$ *is the maximal monitorable semantic sub-formula of a given HML formula φ, i.e.,*

$$\bigvee_{M \,\in\, \textit{almost-un} \,\cap\, \mathrm{MTS}(\varphi)} \chi(M) = \varphi^{MON}$$

Outline

A quick introduction to runtime verification (monitoring)

Monitoring *HML*

Extending runtime verification applicability
- A failed attempt
- A promising road using process semantics

Conclusions

What is missing?

- To extend the approach to full μ*HML*
  - MTS's with cycles
- Even in the context of *HML*, extend monitoring abilities
  - a monitor knows when a process terminates (complete-simulation)
  - a monitor knows which are the next (1-step) possible states (ready-simulation)
- Complexity analysis
  - comparison with a (doubly exponential) recent approach (not published yet)