Runtime Verification using a Temporal Description Logic

Franz Baader
Marcel Lippmann

TU Dresden
Germany

Andreas Bauer

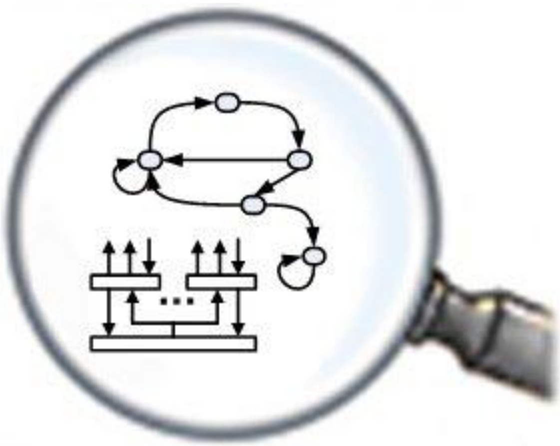
NICTA Canberra
Australia

Runtime Verification

What it is *not*.

Verification:

- system whose behaviour is formally specified
- verify that the system does satisfies a certain (temporal) property before actually running the system



$\models \phi$

Dresden

Runtime Verification

What it is.

Runtime Verification:

- system whose behaviour can only be observed
- monitor the system behaviour during runtime and raise an alarm if a certain (temporal) property is violated

$$\begin{array}{cccc} P & P & \neg P & \neg P \\ \neg Q & Q & \neg Q & Q \end{array}$$

Runtime Verification

What it *really* is.

We need to explain in more detail:

- how properties can be specified;

  linear temporal logic LTL

Dresden

Linear Temporal Logic LTL

syntax

LTL-formulae are built from propositional variables and the constants true and false using

- Boolean operators $\phi \wedge \psi, \phi \vee \psi, \neg\phi, \phi \Rightarrow \psi, \ldots$
- the Next operator $\mathsf{X}\phi$
- the Until operator $\phi \,\mathsf{U}\, \psi$

Abbreviations: $\Diamond\phi := \mathsf{true}\,\mathsf{U}\,\phi$ (eventually ϕ)

$\Box\phi := \neg\Diamond\neg\phi$ (always ϕ)

Dresden

Linear Temporal Logic LTL

example

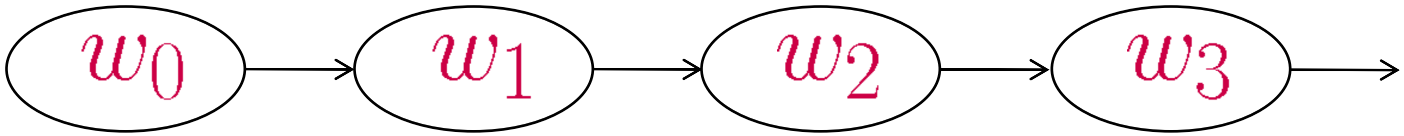
> If the resource is granted to process a,
> then it cannot be granted to process b
> until process a has released the resource.

$$\Box(\mathsf{grant}(a) \Rightarrow (\neg \mathsf{grant}(b)\ \mathsf{U}\ \mathsf{release}(a)))$$

Linear Temporal Logic LTL

semantics

LTL structure: sequence $\mathfrak{W} = (w_i)_{i=0,1,\ldots}$ of propositional interpretations



Validity of ϕ in $\mathfrak{W}$ at time i (written $\mathfrak{W}, i \models \phi$) is defined inductively:

Linear Temporal Logic LTL

semantics

The LTL formula ϕ is satisfiable iff it has a model

i.e., there is an LTL structure $\mathfrak{W}$ such that $\mathfrak{W}, 0 \models \phi$.

Validity of ϕ in $\mathfrak{W}$ at time i (written $\mathfrak{W}, i \models \phi$) is defined inductively:

$\mathfrak{W}, i \models p$	iff	w_i makes p true
$\mathfrak{W}, i \models \mathsf{true}$	iff	$\cdots$
$\mathfrak{W}, i \models \phi \wedge \psi$	iff	$\mathfrak{W}, i \models \phi$ and $\mathfrak{W}, i \models \psi$
$\mathfrak{W}, i \models \phi \vee \psi$	iff	$\cdots$
$\mathfrak{W}, i \models \mathsf{X}\phi$	iff	$\mathfrak{W}, i+1 \models \phi$
$\mathfrak{W}, i \models \phi \,\mathsf{U}\, \psi$	iff	there is $k \geq i$ such that $\mathfrak{W}, k \models \psi$ and $\mathfrak{W}, j \models \phi$ for all $j, i \leq j < k$

Dresden

The LTL formula ϕ is satisfiable iff it has a model
i.e., there is an LTL structure $\mathfrak{W}$ such that $\mathfrak{W}, 0 \models \phi$.

$\mathfrak{W}, 0 \models \phi$ "$\mathfrak{W}$ satisfies ϕ"

$\mathfrak{W}, 0 \not\models \phi$ "$\mathfrak{W}$ violates ϕ"

Deciding satisfiability:

- For every LTL-formula ϕ we can construct a Büchi automaton $\mathcal{A}_\phi$ such that
  - $L(\mathcal{A}_\phi)$ consists of the models of ϕ, viewed as infinite words, and thus ϕ is satisfiable iff $L(\mathcal{A}_\phi) \neq \emptyset$.
  - The size of $\mathcal{A}_\phi$ is exponential in the size of ϕ.

Runtime Verification

What it *really* is.

We need to specify in more detail:

- how properties can be specified;

  linear temporal logic LTL

- how the monitor is supposed to work:
  - What does it receive as input?
  - What should it yield as output?

The runtime verification problem

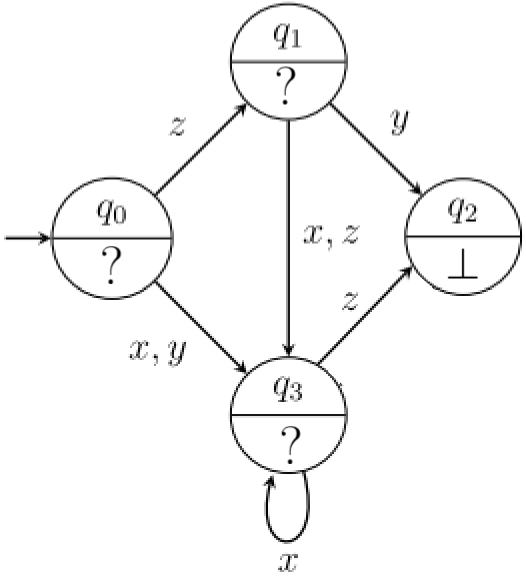
$$\begin{array}{c|c|c|c} P & P & \neg P & \neg P \\ \neg Q & Q & \neg Q & Q \end{array} \quad \overset{?}{\models} \phi$$

At each time point, we have seen a finite initial fragment $\mathfrak{U}$ of an LTL-interpretation.

Dresden

The monitor

$$\mathfrak{U} = \begin{array}{c|c|c} P & P & \neg P \\ \neg Q & Q & \neg Q \\ y & x & z \end{array}$$



A monitor for ϕ is a deterministic finite automaton $\mathcal{M}_\phi$ with state output such that the following holds:

- if state q is reached from the initial state with input $\mathfrak{U}$,
- then the output of state q is $m(\mathfrak{U}, \phi)$.

Dresden

The monitor

A monitor $\mathcal{M}_\phi$ can be constructed from the Büchi automata $\mathcal{A}_\phi$ and $\mathcal{A}_{\neg\phi}$ for ϕ and $\neg\phi$

- apply powerset construction to the Büchi automata
- build the product automaton
- compute the output of each state by reachability analyses in $\mathcal{A}_\phi$ and $\mathcal{A}_{\neg\phi}$

Complexity:

- The size of $\mathcal{M}_\phi$ is doubly exponential in the size of ϕ.
- The time needed to execute a single transition and to output the value does not depend on the length of the initial fragment $\mathfrak{U}$ already read.
- The monitor can compute $m(\mathfrak{U}, \phi)$ in time linear in the length of $\mathfrak{U}$.

Runtime Verification using a Temporal Description Logic

ontology-based runtime verification

Avoid limitations of purely propositional approach:

⇒ Description Logic interpretations can have a complex relational structure.

⇒ Description Logic knowledge bases can describe incomplete knowledge.

Properties need to be formulated in a temporal Description Logic.

Dresden

Combining DLs with TLs

two-dimensional semantics

DL interpreta

DL interpretation

DL interpretation

DL interpretation

$\neg$Conscious(Bob)

$\neg$holds(Bob, Bottle)

Broken(Bottle)

Temporal Structure

Conscious(Bob)

holds(Bob, Bottle)

Dresden

Degrees of freedom

which DL and which TL?

DL dimension: $\mathcal{ALC}$

$C \sqcap D, C \sqcup D, \neg C, \forall r.C, \exists r.C$

prototypical

TL dimension: LTL

$\phi \wedge \psi, \phi \vee \psi, \neg \phi, \mathsf{X}\phi, \phi \,\mathsf{U}\, \psi$

Concept description language

Constructors of the DL $\mathcal{ALC}$:

$C \sqcap D, C \sqcup D, \neg C, \forall r.C, \exists r.C$

A man

that has a rich or beautiful wife,

a son and a daughter,

and only nice friends.

TBox

definition of concepts

$Happy\_man \equiv Human \sqcap \ldots$

ABox

properties of individuals

$Happy\_man(Franz)$

$married\_to(Franz, Inge)$

$child(Franz, Luisa)$

Degrees of freedom

Which pieces of DL syntax can temporal operators be applied to?

~~Temporal operators used as concept constructors:~~

~~$\exists \mathsf{finding.Concussion} \;\sqcap\; \mathsf{Conscious} \; \mathcal{U} \; \exists \mathsf{procedure.Examination}$~~

Temporal operators applied to TBox axioms:

$\Diamond\Box(\mathsf{UScitizen} \sqsubseteq \exists \mathsf{insured\_by.HealthInsurer})$

Temporal operators applied to ABox assertions:

$\Diamond((\exists \mathsf{finding.Concussion})(\mathsf{BOB}) \;\wedge$
$\mathsf{Conscious}(\mathsf{BOB}) \; \mathcal{U} \; (\exists \mathsf{procedure.Examination})(\mathsf{BOB}))$

Our choice

Dresden

Degrees of freedom

Are there rigid concepts/roles whose interpretation does not vary over time?

Rigid concepts/roles:

have the same extension in every world of the temporal structure

Human, has_father

have both

Our choice

Flexible concepts/roles:

extension may change when going to another world of the temporal structure

Conscious, insured_by

Dresden

$\mathcal{ALC}$-LTL syntax

$\mathcal{ALC}$-LTL formulae are defined by induction:

- if α is an $\mathcal{ALC}$-TBox axiom or an $\mathcal{ALC}$-ABox assertion, then α is an $\mathcal{ALC}$-LTL formula;
- if ϕ, ψ are $\mathcal{ALC}$-LTL formulae, then so are
  $\phi \wedge \psi$, $\phi \vee \psi$, $\neg\phi$,
  $\phi \mathsf{U} \psi$, and $\mathsf{X}\phi$.

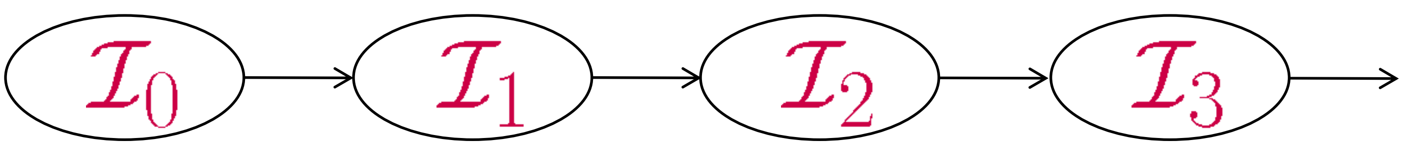
The set of concept (role) names is partitioned into sets of rigid and flexible (concept) role names.

$\mathcal{ALC}$-LTL semantics

$\mathcal{ALC}$-LTL structure

sequence $\mathfrak{I} = (\mathcal{I}_i)_{i=0,1,\ldots}$ of $\mathcal{ALC}$-interpretations $\mathcal{I}_i = (\Delta, \cdot^{\mathcal{I}_i})$

- over the same domain
- obeying the unique name assumption for individuals (UNA)
- interpreting all individuals as well as the rigid concept and role names in the same way



$\mathcal{ALC}$-LTL semantics

Given an $\mathcal{ALC}$-LTL formula ϕ, an $\mathcal{ALC}$-LTL structure $\mathfrak{I} = (\mathcal{I}_i)_{i=0,1,\ldots}$, and a time point $i \in \{0, 1, 2, \ldots\}$, validity of ϕ in $\mathfrak{I}$ at time i (written $\mathfrak{I}, i \models \phi$) is defined inductively:

The $\mathcal{ALC}$-LTL formula ϕ is satisfiable iff there is an $\mathcal{ALC}$-LTL structure $\mathfrak{I}$ such that $\mathfrak{I}, 0 \models \phi$.

Dresden

Satisfiability in $\mathcal{ALC}$-LTL

	With rigid names	Without rigid names
$\mathcal{ALC}$-LTL	2-ExpTime-complete	ExpTime-complete

With rigid names: both rigid and flexible concepts and roles

Without rigid names: all concepts and roles are flexible

For every $\mathcal{ALC}$-LTL-formula ϕ we can construct a Büchi automaton $\mathcal{A}_\phi$ such that:

- $L(\mathcal{A}_\phi)$ consists of (abstractions of) the models of ϕ.
- Without rigid names, the size of $\mathcal{A}_\phi$ is exponential in the size of ϕ.
- With rigid names, the size of $\mathcal{A}_\phi$ is doubly exponential in the size of ϕ.

Runtime verification in $\mathcal{ALC}$-LTL

The case of complete observations:

monitor "sees" (abstractions of) $\mathcal{ALC}$-interpretations.

For every $\mathcal{ALC}$-LTL-formula ϕ we can construct a correct monitor $\mathcal{M}_\phi$ of size

- doubly exponential in the size of ϕ for formulae without rigid names.

!!!!!!!

Dresden

Runtime verification in $\mathcal{ALC}$-LTL

The case of incomplete observations:

monitor "sees" $\mathcal{ALC}$-ABoxes.

The sequence of $\mathcal{ALC}$-interpretations $\mathcal{I}_1\mathcal{I}_2\ldots\mathcal{I}_k$ is a precification of the sequence of ABoxes $\mathcal{A}_1\mathcal{A}_2\ldots\mathcal{A}_k$ if

$$\mathcal{I}_i \text{ is a model of } \mathcal{A}_i \quad (i = 1, \ldots, k)$$

$$m(\mathcal{A}_1\ldots\mathcal{A}_k,\phi) = \begin{cases} \top & \text{if } m(\mathcal{I}_1\ldots\mathcal{I}_k,\phi) = \top \\ & \text{for all precifications } \mathcal{I}_1\ldots\mathcal{I}_k \text{ of } \mathcal{A}_1\ldots\mathcal{A}_k \\ \bot & \text{if } m(\mathcal{I}_1\ldots\mathcal{I}_k,\phi) = \bot \\ & \text{for all precifications } \mathcal{I}_1\ldots\mathcal{I}_k \text{ of } \mathcal{A}_1\ldots\mathcal{A}_k \\ ? & \text{otherwise} \end{cases}$$

Dresden

Runtime verification in $\mathcal{ALC}$-LTL

The case of incomplete observations:

monitor "sees" $\mathcal{ALC}$-ABoxes.

For every $\mathcal{ALC}$-LTL-formula ϕ we can construct a correct monitor $\mathcal{M}_\phi$ of size

- doubly exponential in the size of ϕ for formulae without rigid names.

!!!!!!!

References

- Andreas Bauer, Martin Leucker, and Christian Schallhart.
  Monitoring of real-time properties.
  In Proceedings of the 26th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS06), volume 4337 of Lecture Notes in Computer Science, Springer-Verlag, 2006.

- Franz Baader, Silvio Ghilardi, and Carsten Lutz.
  LTL over Description Logic Axioms.
  In Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR2008), 2008.

- Franz Baader, Andreas Bauer, and Marcel Lippmann.
  Runtime Verification Using a Temporal Description Logic.
  In Proceedings of the 7th International Symposium on Frontiers of Combining Systems (FroCoS 2009), volume 5749 of Lecture Notes in Computer Science, Springer-Verlag, 2009.