Runtime Verification Using a Temporal Description Logic

Franz Baader[1] Andreas Bauer[2] Marcel Lippmann[1]

[1]Technische Universität Dresden,
{baader,lippmann}@tcs.inf.tu-dresden.de

[2]The Australian National University,
baueran@rsise.anu.edu.au

7th International Symposium on Frontiers of Combining Systems, FroCoS 2009

Outline

1. Introduction and motivation
2. Runtime verification for $\mathcal{ALC}$-LTL with rigid names with respect to incomplete knowledge
   - The temporal description logic $\mathcal{ALC}$-LTL
   - Generalised Büchi automata for $\mathcal{ALC}$-LTL formulae
   - The monitor construction
   - The complexity of the monitor construction
3. Conclusion

Outline

The motivation for runtime verification

Problems

- Daily life depends on complex and dynamical hardware and software systems.
- Question: Does a system have the desired properties?
- Safety-critical systems (aviation systems, power plants, . . .) correct?
- Commercially used systems correct?
- Testing and simulation are not sufficient.

The motivation for runtime verification

Problems

- Daily life depends on complex and dynamical hardware and software systems.
- Question: Does a system have the desired properties?
- Safety-critical systems (aviation systems, power plants, . . .) correct?
- Commercially used systems correct?
- Testing and simulation are not sufficient.

Solutions

- Model checking (Complete system is known.)
- Runtime verification (Aspects of the system behaviour can be observed.)

The motivation for runtime verification

Problems

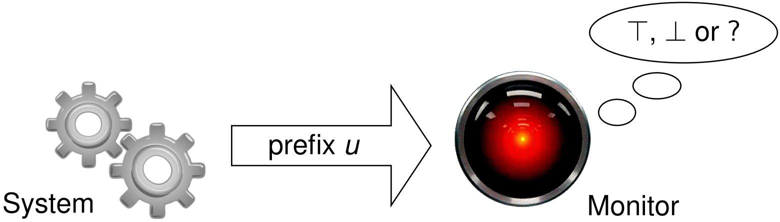
- Daily life depends on complex and dynamical hardware and software systems.
- Question: Does a system have the desired properties?
- Safety-critical systems (aviation systems, power plants, ...) correct?
- Commercially used systems correct?
- Testing and simulation are not sufficient.

Solutions

- Model checking (Complete system is known.)
- Runtime verification (Aspects of the system behaviour can be observed.)

The runtime-verification problem

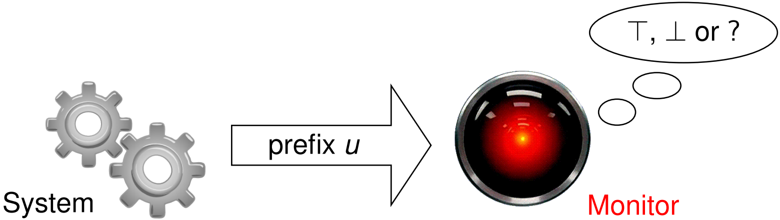
- Some observable system with the desired property ϕ yields a finite prefix u of a trace at each point in time.
- The three possible answers to the runtime-verification problem (u, ϕ):
  - $\top$, if all continuations of u to an infinite trace satisfy ϕ;
  - $\bot$, if all continuations of u to an infinite trace do not satisfy ϕ;
  - ?, if none of the above holds, i. e. there is a continuation that satisfies ϕ, and one that does not.



The basic idea of runtime verification

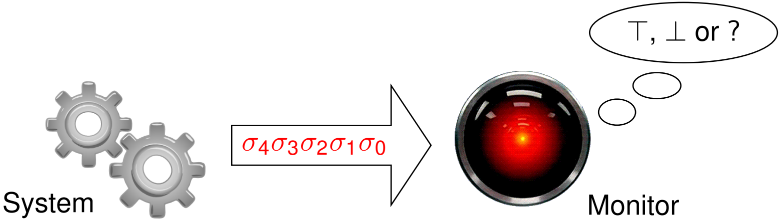
Necessary preparations

- Formalise the desired properties (or parts of it) as logical formula (LTL, . . .).
- Construct a *monitor* out of the formalisation of the desired properties.
  (*Note:* The monitor does not depend on the system.)



Properties of the monitor

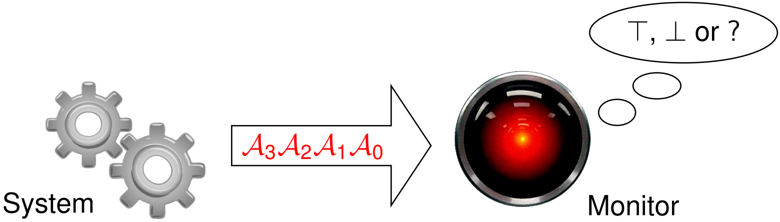
- It solves not a single problem (u, ϕ) $\rightsquigarrow$ prefix u is continuously extended by observing the system behaviour over time.
- The delay between answering (u, ϕ) and $(u\sigma, \phi)$ is constant (if ϕ is assumed to be constant). Thus, the computation of the answer to the next problem does not depend on the length of the already processed prefix.



Why an extension of prop. LTL runtime verification?

Limitations of propositional LTL runtime verification

- If the observations of the system have a complex internal structure
  $\rightsquigarrow$ Extension of the approach to $\mathcal{ALC}$-LTL.
- If one can observe the the system behaviour only restricted / (possibly) incomplete knowledge
  $\rightsquigarrow$ Input of the monitor: $\mathcal{ALC}$-ABoxes.



Emergency ward

- The vital parameters of the patient are measured in short intervals and additional information is available from the patient record and added by medical staff.
- Using a medical ontology, a high-level view of the patient's medical status can be given by ABoxes.
- Critical situations requiring the intervention by a doctor can then be described by an $\mathcal{ALC}$-LTL formula.

- As long as the monitor outputs $?$, it continues monitoring. If it outputs $\top$, we raise an alarm and if it outputs $\bot$, we shut it off.

Outline

1 Introduction and motivation

2 Runtime verification for $\mathcal{ALC}$-LTL with rigid names with respect to incomplete knowledge

- The temporal description logic $\mathcal{ALC}$-LTL
- Generalised Büchi automata for $\mathcal{ALC}$-LTL formulae
- The monitor construction
- The complexity of the monitor construction

3 Conclusion

The monitor construction in a nutshell

$$\begin{tikzcd}
 & \mathcal{G}_\phi \arrow[r] & \widehat{\mathcal{G}}_\phi \arrow[dr] & & \\
\phi \arrow[ur] \arrow[dr] & & & \mathcal{M}_\phi \arrow[r, dashed] & \mathcal{M}_\phi^{\mathit{inc}} \\
 & \mathcal{G}_{\neg\phi} \arrow[r] & \widehat{\mathcal{G}}_{\neg\phi} \arrow[ur] & &
\end{tikzcd}$$

- ϕ ... $\mathcal{ALC}$-LTL formula
- $\mathcal{G}_\phi$, $\mathcal{G}_{\neg\phi}$ ... generalised Büchi automata (GBA) for ϕ and $\neg\phi$
- $\widehat{\mathcal{G}}_\phi$, $\widehat{\mathcal{G}}_{\neg\phi}$ ... GBA for ϕ and $\neg\phi$ respecting rigid names
- $\mathcal{M}_\phi$ ... monitor for ϕ
- $\mathcal{M}_\phi^{\mathit{inc}}$ ... monitor for ϕ working with incomplete knowledge

The temporal description logic $\mathcal{ALC}$-LTL (1)

$$
\begin{aligned}
\phi \quad := \quad & \Box(\mathsf{GermanCitizen} \sqsubseteq \exists \mathsf{insured\_by}.\mathsf{HealthInsurer}) \wedge \\
& \Box(\mathsf{BOB} : \mathsf{Male} \sqcap \mathsf{GermanCitizen}) \wedge \\
& \Diamond\Box((\mathsf{BOB}, \mathsf{TK}) : \mathsf{insured\_by}) \wedge \\
& \Diamond((\mathsf{BOB} : \exists \mathsf{finding}.\mathsf{Concussion}) \wedge \\
& (\mathsf{BOB} : \mathsf{Conscious})\ \mathsf{U}\ (\mathsf{BOB} : \exists \mathsf{procedure}.\mathsf{Examination}))
\end{aligned}
$$

The temporal description logic $\mathcal{ALC}$-LTL (1)

$$\begin{aligned}\phi \quad := \quad & \Box(\text{GermanCitizen} \sqsubseteq \exists \text{insured\_by.HealthInsurer}) \wedge \\ & \Box(\text{BOB} : \text{Male} \sqcap \text{GermanCitizen}) \wedge \\ & \Diamond\Box((\text{BOB}, \text{TK}) : \text{insured\_by}) \wedge \\ & \Diamond((\text{BOB} : \exists \text{finding.Concussion}) \wedge \\ & (\text{BOB} : \text{Conscious}) \ \mathsf{U} \ (\text{BOB} : \exists \text{procedure.Examination}))\end{aligned}$$

- general concept inclusion axiom

The temporal description logic $\mathcal{ALC}$-LTL (1)

$$
\begin{aligned}
\phi \quad := \quad & \Box(\mathsf{GermanCitizen} \sqsubseteq \exists \mathsf{insured\_by}.\mathsf{HealthInsurer}) \wedge \\
& \Box(\mathsf{BOB} : \mathsf{Male} \sqcap \mathsf{GermanCitizen}) \wedge \\
& \Diamond\Box((\mathsf{BOB}, \mathsf{TK}) : \mathsf{insured\_by}) \wedge \\
& \Diamond((\mathsf{BOB} : \exists \mathsf{finding}.\mathsf{Concussion}) \wedge \\
& (\mathsf{BOB} : \mathsf{Conscious})\ \mathsf{U}\ (\mathsf{BOB} : \exists \mathsf{procedure}.\mathsf{Examination}))
\end{aligned}
$$

- general concept inclusion axiom
- concept assertion

The temporal description logic $\mathcal{ALC}$-LTL (1)

$$
\begin{aligned}
\phi \quad := \quad & \Box(\text{GermanCitizen} \sqsubseteq \exists\text{insured\_by}.\text{HealthInsurer}) \wedge \\
& \Box(\text{BOB} : \text{Male} \sqcap \text{GermanCitizen}) \wedge \\
& \Diamond\Box((\text{BOB}, \text{TK}) : \text{insured\_by}) \wedge \\
& \Diamond((\text{BOB} : \exists\text{finding}.\text{Concussion}) \wedge \\
& (\text{BOB} : \text{Conscious}) \, \mathsf{U} \, (\text{BOB} : \exists\text{procedure}.\text{Examination}))
\end{aligned}
$$

- general concept inclusion axiom
- concept assertion
- role assertion

The temporal description logic $\mathcal{ALC}$-LTL (1)

$$
\begin{aligned}
\phi \quad := \quad & \Box(\mathsf{GermanCitizen} \sqsubseteq \exists \mathsf{insured\_by.HealthInsurer}) \wedge \\
& \Box(\mathsf{BOB} : \mathsf{Male} \sqcap \mathsf{GermanCitizen}) \wedge \\
& \Diamond\Box((\mathsf{BOB}, \mathsf{TK}) : \mathsf{insured\_by}) \wedge \\
& \Diamond((\mathsf{BOB} : \exists \mathsf{finding.Concussion}) \wedge \\
& (\mathsf{BOB} : \mathsf{Conscious})\ \mathsf{U}\ (\mathsf{BOB} : \exists \mathsf{procedure.Examination}))
\end{aligned}
$$

- general concept inclusion axiom
- concept assertion
- role assertion
- conjunction, negation and temporal modalities
  (Usual abbreviations: $\Box\psi := (\top \sqsubseteq \top)\mathsf{U}\psi$, $\Diamond\psi := \neg\Box\neg\psi, \ldots$)

The temporal description logic $\mathcal{ALC}$-LTL (1)

$$\begin{aligned}\phi \quad := \quad & \Box(\mathsf{GermanCitizen} \sqsubseteq \exists \mathsf{insured\_by.HealthInsurer}) \wedge \\ & \Box(\mathsf{BOB} : \textcolor{red}{\mathsf{Male}} \sqcap \mathsf{GermanCitizen}) \wedge \\ & \Diamond\Box((\mathsf{BOB}, \mathsf{TK}) : \mathsf{insured\_by}) \wedge \\ & \Diamond((\mathsf{BOB} : \exists \mathsf{finding.Concussion}) \wedge \\ & (\mathsf{BOB} : \mathsf{Conscious})\ \mathsf{U}\ (\mathsf{BOB} : \exists \mathsf{procedure.Examination}))\end{aligned}$$

- general concept inclusion axiom
- concept assertion
- role assertion
- conjunction, negation and temporal modalities
  (Usual abbreviations: $\Box\psi := (\top \sqsubseteq \top)\mathsf{U}\psi$, $\Diamond\psi := \neg\Box\neg\psi$, $\ldots$)
- rigid concept or role name

The temporal description logic $\mathcal{ALC}$-LTL (1)

$$
\begin{aligned}
\phi \quad := \quad & \Box(\text{GermanCitizen} \sqsubseteq \exists\text{insured\_by}.\text{HealthInsurer}) \land \\
& \Box(\text{BOB} : \text{Male} \sqcap \text{GermanCitizen}) \land \\
& \Diamond\Box((\text{BOB}, \text{TK}) : \text{insured\_by}) \land \\
& \Diamond((\text{BOB} : \exists\text{finding}.\text{Concussion}) \land \\
& (\text{BOB} : \text{Conscious})\ \mathsf{U}\ (\text{BOB} : \exists\text{procedure}.\text{Examination}))
\end{aligned}
$$

- general concept inclusion axiom
- concept assertion
- role assertion
- conjunction, negation and temporal modalities
  (Usual abbreviations: $\Box\psi := (\top \sqsubseteq \top)\mathsf{U}\psi$, $\Diamond\psi := \neg\Box\neg\psi, \ldots$)
- rigid concept or role name
- flexible concept or role name

The temporal description logic $\mathcal{ALC}$-LTL (1)

$$\begin{aligned}\phi \quad := \quad & \Box(\text{GermanCitizen} \sqsubseteq \exists\text{insured\_by.HealthInsurer}) \land \\ & \Box(\text{BOB} : \text{Male} \sqcap \text{GermanCitizen}) \land \\ & \Diamond\Box((\text{BOB}, \text{TK}) : \text{insured\_by}) \land \\ & \Diamond((\text{BOB} : \exists\text{finding.Concussion}) \land \\ & (\text{BOB} : \text{Conscious})\ \mathsf{U}\ (\text{BOB} : \exists\text{procedure.Examination}))\end{aligned}$$

- general concept inclusion axiom
- concept assertion
- role assertion
- conjunction, negation and temporal modalities
  (Usual abbreviations: $\Box\psi := (\top \sqsubseteq \top)\mathsf{U}\psi$, $\Diamond\psi := \neg\Box\neg\psi, \ldots$)
- rigid concept or role name
- flexible concept or role name

The temporal description logic $\mathcal{ALC}$-LTL (2)

Semantics: $\mathcal{ALC}$-LTL structure $\mathfrak{I} = (\mathcal{I}_i)_{i=0,1,\ldots}$

- sequence of $\mathcal{ALC}$-interpretations $\mathcal{I}_i = (\Delta^{\mathcal{I}_i}, \cdot^{\mathcal{I}_i})$
- straight-forward extension of LTL-structures
- $\mathfrak{I}$ respects rigid names if the $\mathcal{I}_i$ interpret rigid concept and role names always in the same manner.

Generalised Büchi automata for $\mathcal{ALC}$-LTL formulae (1)

For *propositional* LTL-formulae:

- well-known construction (by Vardi, Wolper and Sistla)

For $\mathcal{ALC}$-LTL formulae without rigid names:

- We basically follow the same idea as for propositional LTL.
- *Alphabet* Σ^ϕ*:*
  - Not $\mathcal{ALC}$-interpretations $\rightsquigarrow$ infinite alphabet
  - $\mathcal{ALC}$-types for ϕ (maximal, consistent sets of ϕ-literals)
  - Type for $\mathcal{ALC}$-interpretation: $\tau_\phi(\mathcal{I}) = \{\alpha \in \Sigma^\phi \mid \mathcal{I} \models \alpha\}$
- *Correctness of* $\mathcal{G}_\phi$*:*

  For every $w \in (\Sigma^\phi)^\omega$, we have $w \in L_\omega(\mathcal{G}_\phi)$ iff there exists an $\mathfrak{I}$ such that $\tau_\phi(\mathfrak{I}) = w$ and $\mathfrak{I}, 0 \models \phi$.

Generalised Büchi automata for $\mathcal{ALC}$-LTL formulae (2)

For $\mathcal{ALC}$-LTL formulae with rigid names:

- $\mathcal{G}_\phi$ does not respect rigid names, since there is no guarantee that for $w \in L_\omega(\mathcal{G}_\phi)$ a corresponding $\mathfrak{I}$ respects rigid names.
- $\widehat{\mathcal{G}}_\phi$, which is an extension of $\mathcal{G}_\phi$, enforces this.
  - $\widehat{\mathcal{G}}_\phi$ keeps track of which $\mathcal{ALC}$-types it has already read.
  - $\widehat{\mathcal{G}}_\phi$ allows only transitions if the set of such $\mathcal{ALC}$-types is consistent w. r. t. rigid names.
  - State space of $\widehat{\mathcal{G}}_\phi$:
    - First component: works like $\mathcal{G}_\phi$.
    - Second component: collects all $\mathcal{ALC}$-types which were read.
- *Correctness of* $\widehat{\mathcal{G}}_\phi$*:*

  For every $w \in (\Sigma^\phi)^\omega$, we have $w \in L_\omega(\widehat{\mathcal{G}}_\phi)$ iff there exists an $\mathfrak{I}$ respecting rigid names such that $\tau_\phi(\mathfrak{I}) = w$ and $\mathfrak{I}, 0 \models \phi$.

The monitor construction in the case of complete knowledge

The monitor $\mathcal{M}_\phi$:

- is defined as a deterministic Moore automaton (finite automaton with state output).
- solves the runtime-verification problem: for input *w* it outputs at the reached state the answer to (w, ϕ).

The monitor construction in the case of complete knowledge

The monitor $\mathcal{M}_\phi$:

- is defined as a deterministic Moore automaton (finite automaton with state output).
- solves the runtime-verification problem: for input w it outputs at the reached state the answer to (w, ϕ).

The construction of $\mathcal{M}_\phi$:

- View $\widehat{\mathcal{G}}_\phi$ and $\widehat{\mathcal{G}}_{\neg\phi}$ as automata working on finite words and make them deterministic.
- Build the product automaton of the deterministic automata obtained this way.
- The output is determined through emptiness tests for $\widehat{\mathcal{G}}_\phi$ and $\widehat{\mathcal{G}}_{\neg\phi}$ varying the initial states.

The monitor construction in the case of incomplete knowledge (1)

The representation of incomplete knowledge

- A consistent $\mathcal{ALC}$-ABox $\mathcal{A}$ represents incomplete knowledge (OWA). We have these three possiblities:
  - $\mathcal{A} \models \alpha$
  - $\mathcal{A} \models \neg\alpha$
  - $\mathcal{A} \not\models \alpha$ and $\mathcal{A} \not\models \neg\alpha$

The monitor construction in the case of incomplete knowledge (1)

The representation of incomplete knowledge

- A consistent $\mathcal{ALC}$-ABox $\mathcal{A}$ represents incomplete knowledge (OWA). We have these three possiblities:
  - $\mathcal{A} \models \alpha$
  - $\mathcal{A} \models \neg\alpha$
  - $\mathcal{A} \not\models \alpha$ and $\mathcal{A} \not\models \neg\alpha$

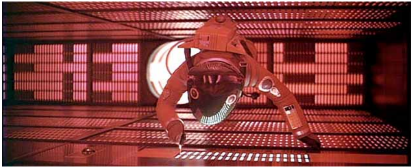
The monitor $\mathcal{M}_\phi^{inc}$:

- The construction is almost identical to the one of $\mathcal{M}_\phi$.
- *Alphabet:* consistent $\mathcal{ALC}$-ABoxes over the vocabulary occurring in ϕ.
- *Transitions:*
  - All we know: the observations of the system are a model of $\mathcal{A}$.
  - $\mathcal{M}_\phi^{inc}$ must consider all the transitions in $\widehat{\mathcal{G}}_\phi$ and $\widehat{\mathcal{G}}_{\neg\phi}$ that can be induced by such models.

The monitor construction in the case of incomplete knowledge (2)

Properties of $\mathcal{M}_\phi^{inc}$:

- No real Moore machine due to the infinite alphabet.
  $\rightsquigarrow$ The definition of a deterministic Moore automaton can easily be extended.
- It is not possible to precompute such an infinite monitor.
  $\rightsquigarrow$ Given some state and an input ABox, one then needs to compute the transition on-the-fly.
- The state space of the monitor, however, is finite.



The complexity of the monitor construction w. r. t. the number of states

Complexity of constructing the GBA:

- $\mathcal{G}_\phi$ and $\mathcal{G}_{\neg\phi}$ can be constructed in *exponential* time.
- $\widehat{\mathcal{G}}_\phi$ and $\widehat{\mathcal{G}}_{\neg\phi}$ can be constructed in *double exponential* time.

The overall complexity for constructing the state space of the monitor:

- The state space of $\mathcal{M}_\phi$ ($\mathcal{M}_\phi^{inc}$) can be constructed in *triple exponential* time.
- The state space of $\mathcal{M}_\phi$ ($\mathcal{M}_\phi^{inc}$) can be constructed in *double exponential* time, if we do not allow rigid names.

Outline

1. Introduction and motivation
2. Runtime verification for $\mathcal{ALC}$-LTL with rigid names with respect to incomplete knowledge
3. Conclusion

Conclusion

- We extended the three-valued approach to runtime verification in propositional LTL to $\mathcal{ALC}$-LTL and the case where the observed system behaviour is described (incompletely) by $\mathcal{ALC}$-ABoxes.
- The complexity of the monitor construction is quite high, ...
  - ...but it should be noted that this is worse-case complexity. One could use minimisation techniques for the GBA and the monitor.
  - ...but the size of the formula is usually quite small, whereas the system is monitored over a long period of time.
  - ...the large size of the monitor can probably not be avoided: The construction of $\mathcal{G}_\phi$ and $\widehat{\mathcal{G}}_\phi$ is optimal.
    Also for runtime verification in propositional LTL, the constructed monitors have actually a size that is one exponential higher than the size of the GBA.

Literature

Recommendations for further reading

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.

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 (FSTTCS'06)*, volume 4337 of *Lecture Notes in Computer Science*, Kolkata, India, December 2006. Springer-Verlag.