UNIVERSITÀ DEGLI STUDI DI GENOVA

MANCHESTER 1824
The University of Manchester

Towards Partial Monitoring: It is Always too Soon to Give Up

Angelo Ferrando[1] and Rafael C. Cardoso[2]
[1] *University of Genova*
[2] The University of Manchester

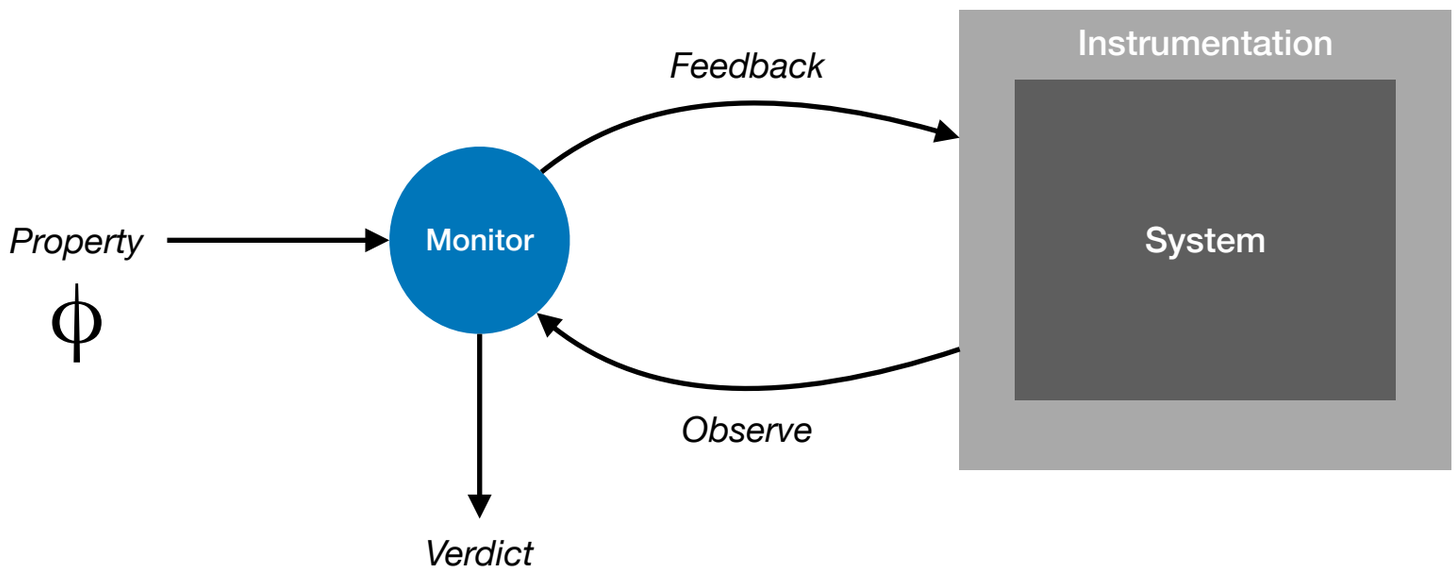
angelo.ferrando@unige.it
Research Fellow

Runtime Verification [in a nutshell]

Complement of

- formal static verification (such as *Model Checking*)
  pros: formal, exhaustive **cons: suffers from scalability**
- testing
  pros: scales well **cons: not formal, not exhaustive**

Dynamic checking of system behaviour using one (or multiple) **Monitor(s)**
pros: formal, scales well, can be done after deployment
cons: not exhaustive



But… Not all properties can be monitored

- A formal property is considered **monitorable**, when it is possible to synthesise a monitor which verifies it. In turn, a formal property is denoted as **non-monitorable**, when it is not possible to synthesise such a monitor.
- There exists different definitions on the requirements for a property to be considered monitorable.
- Nonetheless, the most common requirement for a property to be monitorable is that **it should always be possible to conclude the satisfaction or violation of the property**.

However, some of these non-monitorable properties may still be worth to be analysed at runtime, even though if only **partially**.

Therefore, we pose the following research question:

Is it possible for monitors that are synthesised by non-monitorable properties to be used in practice?

Our proposal

- We suggest that a viable answer to this question is using **partial monitoring**.

- Since a non-monitorable property **does not give any assurance** on satisfaction or violation, we need to synthesise monitors which are **capable of giving up** on the verification process when it is clear that it will never arrive at any conclusion.

- This is especially relevant in the context of autonomous systems, where the availability of computational **power and memory might be limited** (e.g., in embedded systems).

- As a proof of concept, in this work we exemplify our approach in a robotic application where an autonomous rover is deployed into a nuclear facility to perform remote inspection.

Preliminaries

The standard formalism to specify formal properties in RV is propositional Linear Temporal Logic (LTL). The relevant parts of the syntax of LTL are as follows:

$$\varphi = true \mid false \mid ev \mid (\varphi \wedge \varphi') \mid (\varphi \vee \varphi') \mid \neg\varphi \mid (\varphi \ \mathbf{U} \ \varphi') \mid \bigcirc\varphi$$

Let $\sigma \in \Sigma^\omega$ be an infinite sequence of events over Σ, the semantics of LTL is as follows:

$$\begin{aligned}
\sigma &\models ev \text{ if } ev \in \sigma(0) \\
\sigma &\models \neg\varphi \text{ if } \sigma \not\models \varphi \\
\sigma &\models \varphi \wedge \varphi' \text{ if } \sigma \models \varphi \text{ and } \sigma \models \varphi' \\
\sigma &\models \varphi \vee \varphi' \text{ if } \sigma \models \varphi \text{ or } \sigma \models \varphi' \\
\sigma &\models \bigcirc\varphi \text{ if } \sigma^1 \models \varphi \\
\sigma &\models \varphi \,\mathbf{U}\, \varphi' \text{ if } \exists_{i \geq 0}.\sigma^i \models \varphi' \text{ and } \forall_{0 \leq j < i}.\sigma^j \models \varphi
\end{aligned}$$

Thus, given an LTL property φ, we denote $[\![\varphi]\!]$ the language of the property, *i.e.*, the set of traces which satisfy φ; namely $[\![\varphi]\!] = \{\sigma \mid \sigma \models \varphi\}$.

Runtime Monitor

Definition 1 (Monitor) *Let S be a system with alphabet Σ, and φ be an LTL property. Then, a monitor for φ is a function $Mon_\varphi : \Sigma^* \to \mathbb{B}_3$, where $\mathbb{B}_3 = \{\top, \bot, ?\}$:*

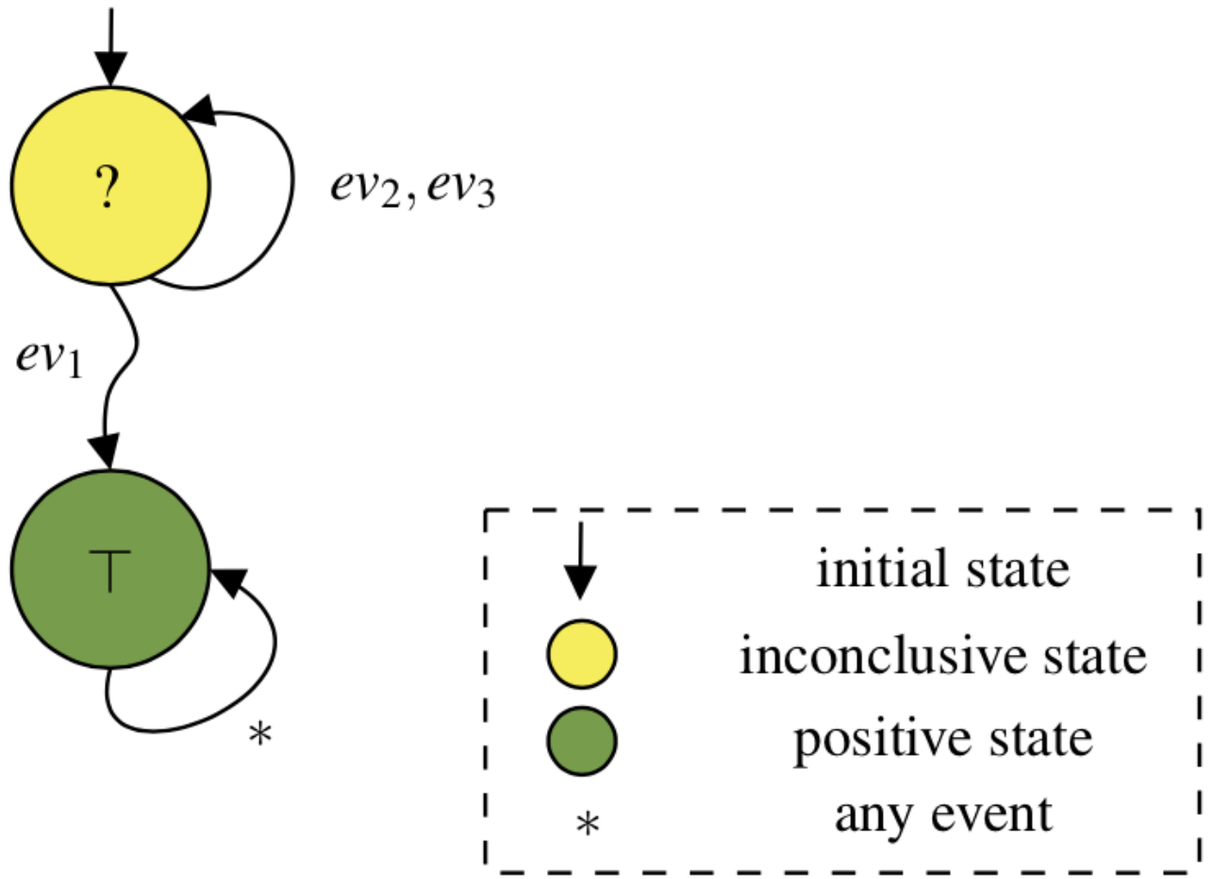
$$Mon_\varphi(\sigma) = \begin{cases} \top & \forall_{u \in \Sigma^\omega}.\sigma \bullet u \in [\![\varphi]\!] \\ \bot & \forall_{u \in \Sigma^\omega}.\sigma \bullet u \notin [\![\varphi]\!] \\ ? & otherwise \end{cases}$$

where $\bullet$ is the standard trace concatenation operator.

A monitor function is usually implemented as a **Finite State Machine (FSM)**, specifically a **Moore machine** (FSM where the output value of a state is only determined by the state).

Example of LTL monitor specified as Moore machine

Example 1 *Let S be a system with alphabet $\Sigma = \{ev_1, ev_2, ev_3\}$, and $\varphi = \Diamond ev_1$ be an LTL property to verify. In natural language, φ reads as: "eventually event ev_1 is going to be observed".*



Monitorability

We consider the definitions of monitorability introduced by Pnueli and Zaks [1], one of the most cited by the community.

Definition 2 *A property* φ *is* σ*-monitorable, where* $\sigma \in \Sigma^*$*, if there is some* $u \in \Sigma^*$ *such that* φ *is positively or negatively determined by* $\sigma \bullet u$*.*

Definition 2 states that a property is *monitorable* w.r.t. a trace of events, if we can find at least one continuation for which the property is satisfied or violated.

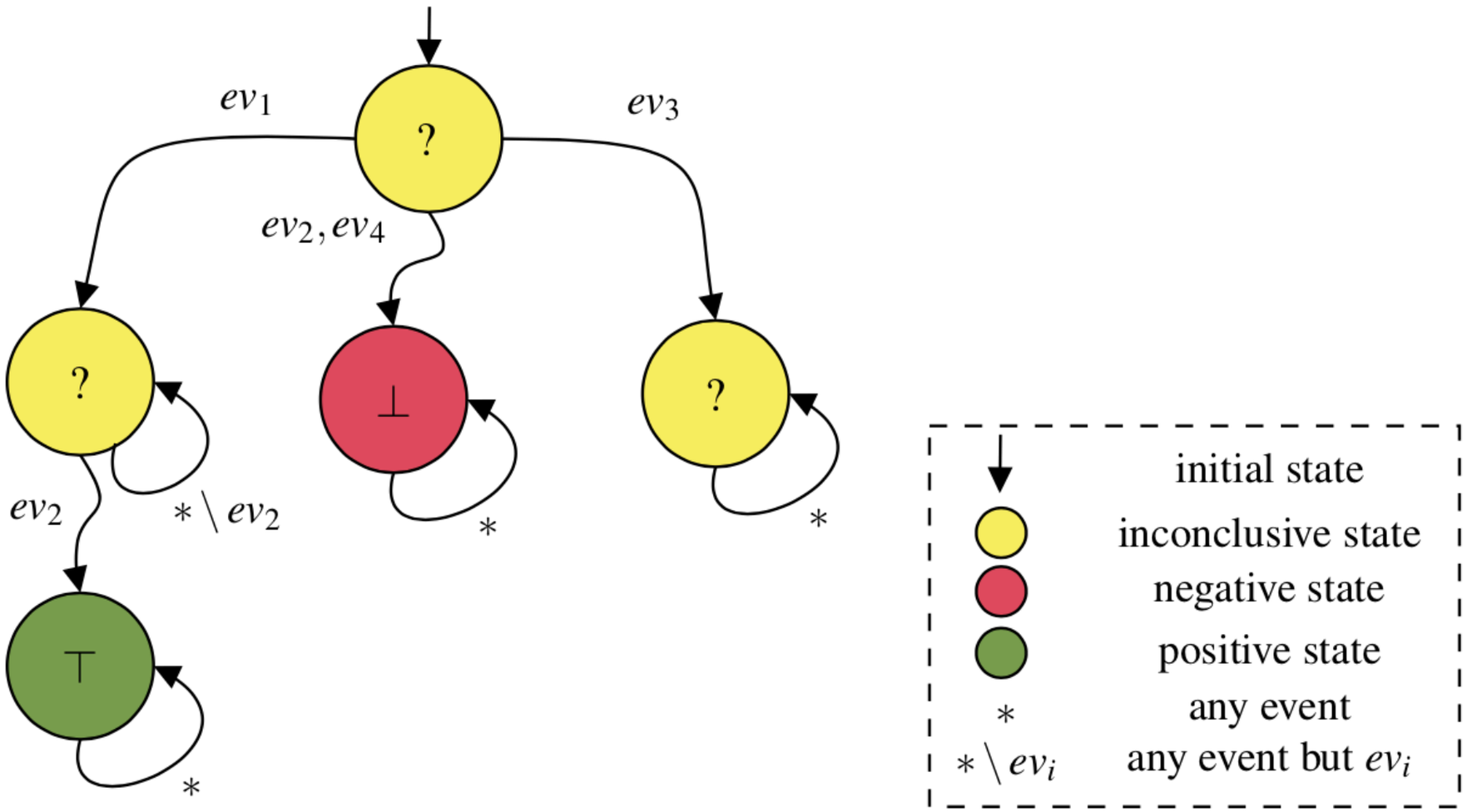
Intuitively, this means that we know that for at least one trace of events the monitor will conclude the verification process.

[1] Amir Pnueli & Aleksandr Zaks (2006): PSL Model Checking and Run-Time Verification Via Testers. In Jayadev Misra, Tobias Nipkow & Emil Sekerinski, editors: FM 2006: Formal Methods, 14th International Symposium on Formal Methods.

Existentially Pnueli-Zaks monitorable

Definition 3 *A property φ is (existentially Pnueli-Zaks) $\exists_{PZ}$-monitorable if it is σ-monitorable for some finite trace $\sigma \in \Sigma^*$. The class of all $\exists_{PZ}$-monitorable properties is denoted as $\exists_{PZ}$.*

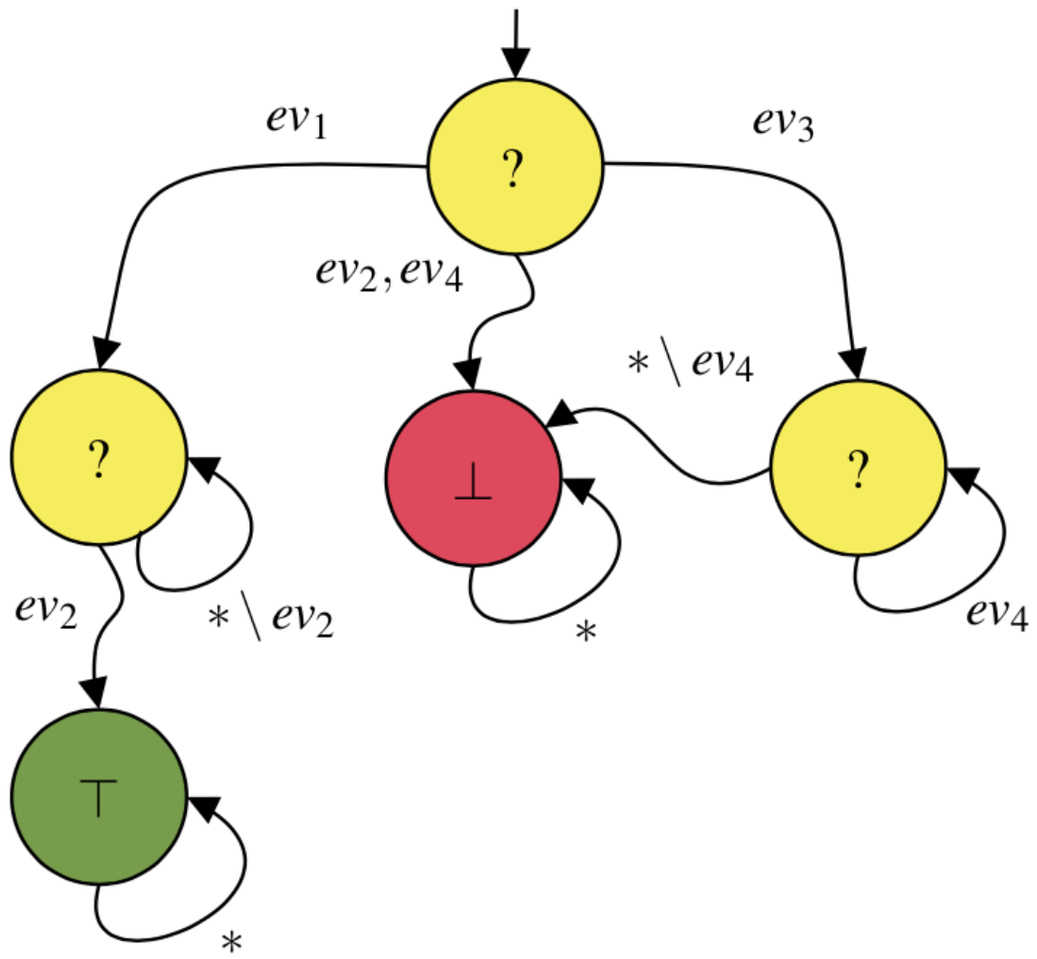
Example 2 *Let us assume $\varphi = (ev_1 \wedge \Diamond ev_2) \vee (ev_3 \wedge \Box\Diamond ev_4)$, and $\Sigma = \{ev_1, ev_2, ev_3, ev_4\}$. This is an example of a $\exists_{PZ}$-monitorable property, since we can find some $\sigma \in \Sigma^*$ for which φ is σ-monitorable; e.g., any trace starting with ev_1 can eventually satisfy φ (by observing ev_2).*



Universally Pnueli-Zaks monitorable

Definition 4 *A property φ is (universally Pnueli-Zaks) $\forall_{PZ}$-monitorable if it is σ-monitorable for all finite trace $\sigma \in \Sigma^*$. The class of all $\forall_{PZ}$-monitorable properties is denoted as $\forall_{PZ}$.*

Example 3 *Let us assume $\varphi = (ev_1 \rightarrow \Diamond ev_2) \vee (ev_3 \rightarrow \Box ev_4)$, and $\Sigma = \{ev_1, ev_2, ev_3, ev_4\}$. This is an example of a $\forall_{PZ}$-monitorable property, since for every $\sigma \in \Sigma^*$, the property is σ-monitorable; we can always find a continuation $u \in \Sigma^*$ s.t. the φ is positively or negatively determined.*



Safety and Co-Safety properties

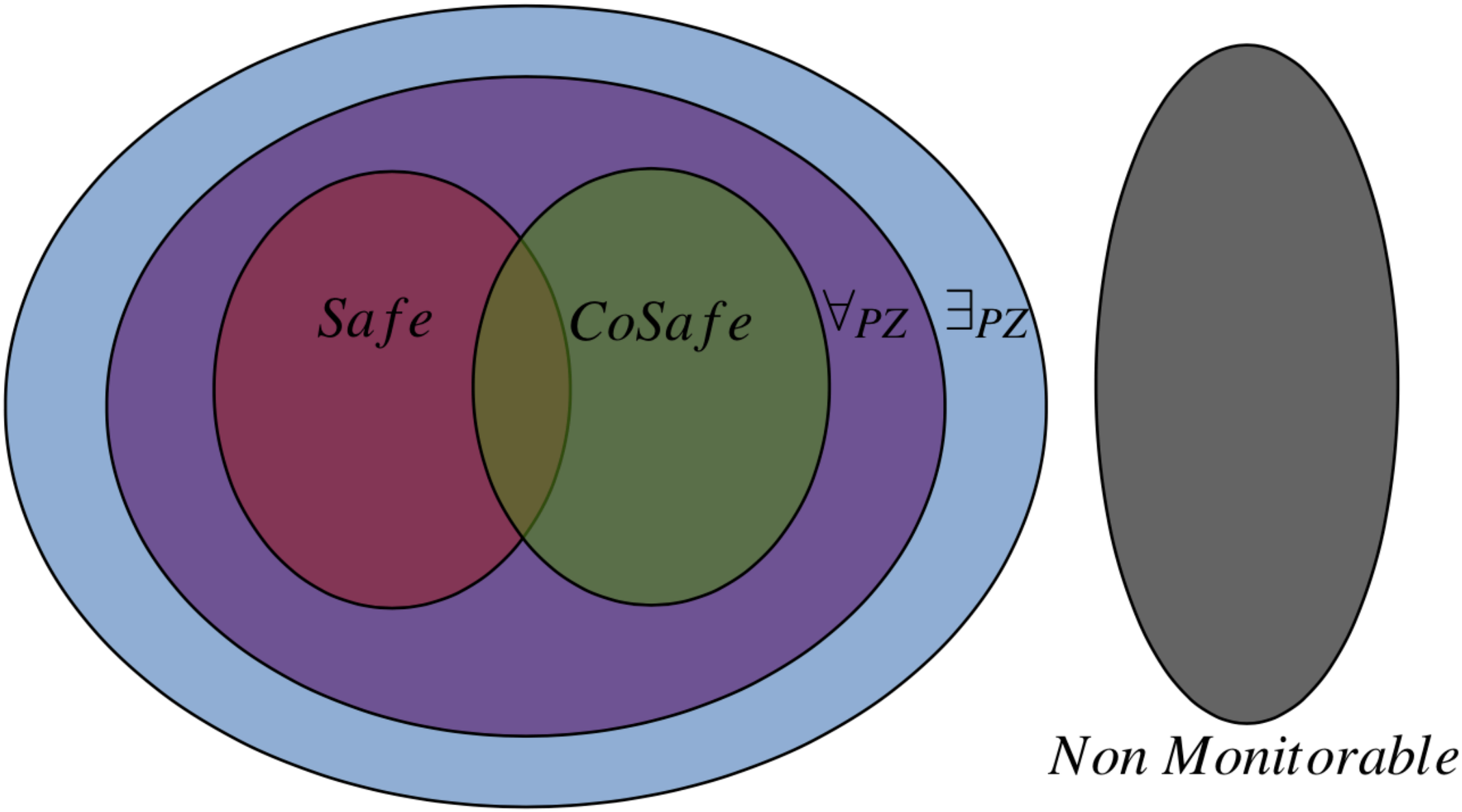
Definition 5 *A property φ is a* safety *property if every $\sigma \notin [\![\varphi]\!]$ has a prefix that determines φ negatively. The class of safety properties is denoted as Safe.*

These properties can only be violated at runtime, which means the resulting monitor can only report negative and inconclusive verdicts. This is due to the fact that safety properties are satisfied only by infinite traces of events, and at runtime we only have access to finite traces.

Definition 6 *A property φ is a* co-safety *property if every $\sigma \in [\![\varphi]\!]$ has a prefix that determines φ positively. The class of co-safety properties is denoted as CoSafe.*

These properties can only be satisfied at runtime, which means the resulting monitor can only report positive and inconclusive verdicts. This is due to the fact that co-safety properties are violated only by infinite traces of events.

Hierarchy of classes of monitorable properties



Partial Monitoring

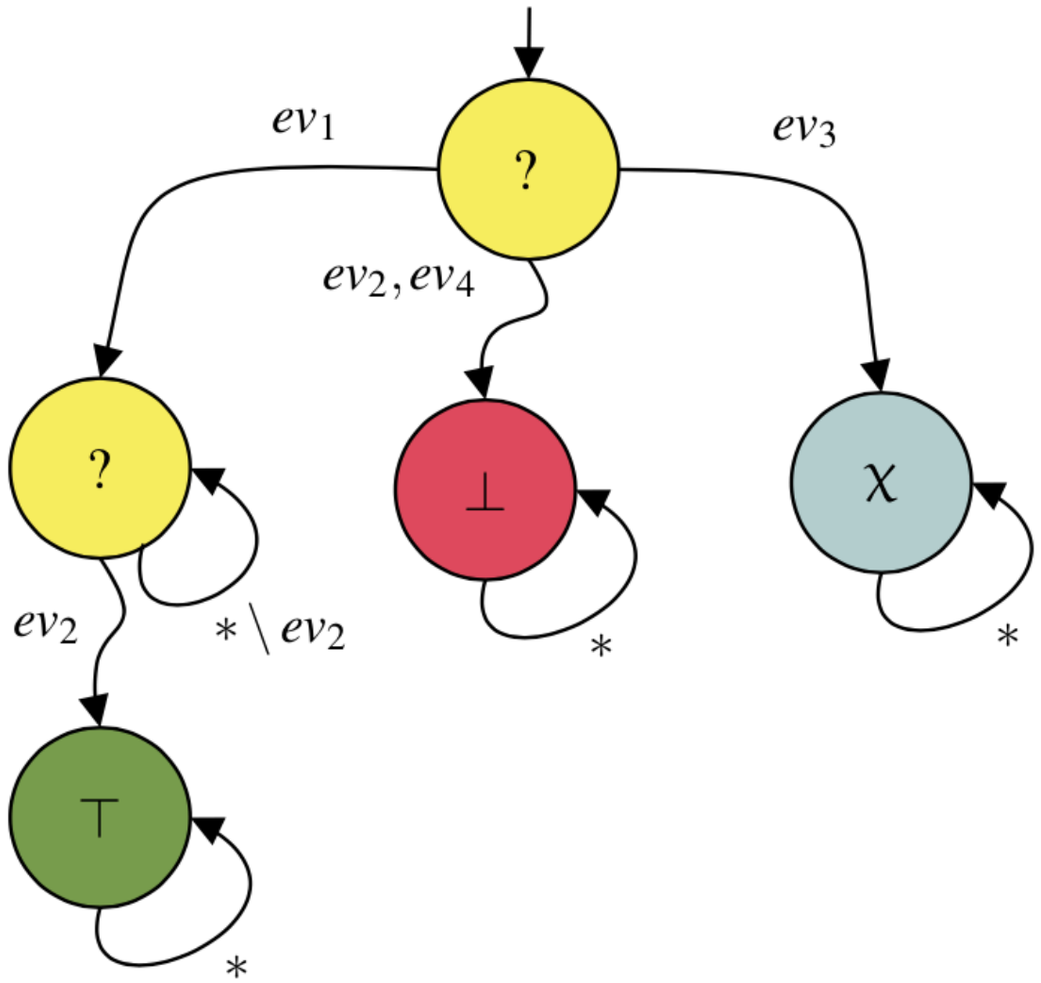
Definition 7 (Partial Monitor) *Let S be a system with alphabet Σ, and φ be an LTL property. Then, a* partial monitor *for φ is a function $Mon_\varphi : \Sigma^* \to \mathbb{B}_4$, where $\mathbb{B}_4 = \{\top, \bot, ?, \chi\}$:*

$$Mon_\varphi(\sigma) = \begin{cases} \top & \forall_{u \in \Sigma^\omega}.\, \sigma \bullet u \in [\![\varphi]\!] \\ \bot & \forall_{u \in \Sigma^\omega}.\, \sigma \bullet u \notin [\![\varphi]\!] \\ ? & \exists_{u \in \Sigma^*}.((\forall_{u' \in \Sigma^\omega}.\, \sigma \bullet u \bullet u' \in [\![\varphi]\!]) \vee (\forall_{u' \in \Sigma^\omega}.\, \sigma \bullet u \bullet u' \notin [\![\varphi]\!])) \\ \chi & \textit{otherwise} \end{cases}$$

where $\bullet$ is the standard trace concatenation operator.

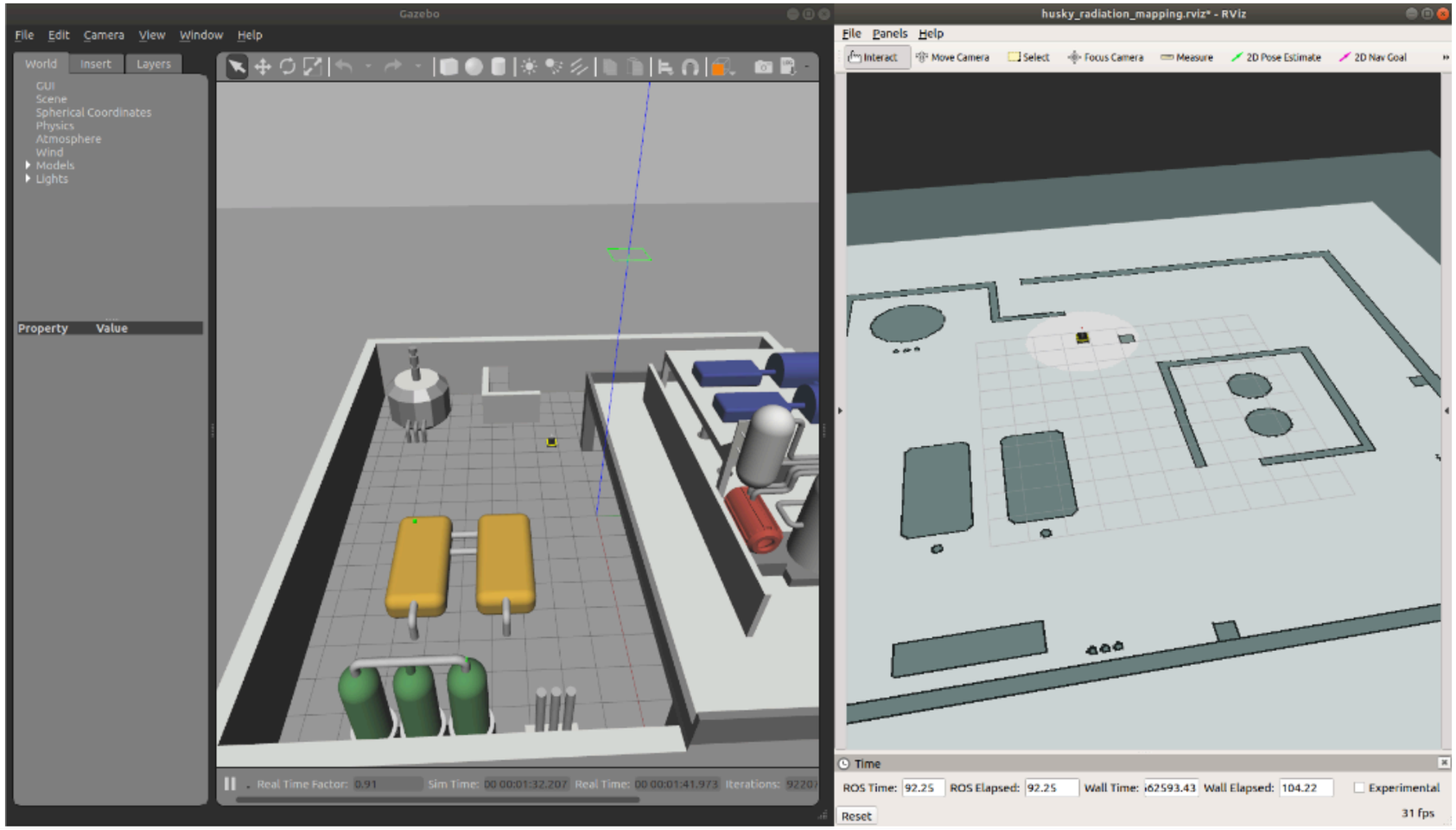
Partial Monitoring

Example 4 *Considering once again the property of Example 2,* $\varphi = (ev_1 \wedge \Diamond ev_2) \vee (ev_3 \wedge \Box\Diamond ev_4)$*, with* $\Sigma = \{ev_1, ev_2, ev_3, ev_4\}$*. This property is* $\exists_{PZ}$*-monitorable, since not all* $\sigma \in \Sigma^*$ *are* σ*-monitorable. For this reason this property would usually be discarded, since no guarantees can be given that the resulting monitor will be able to conclude anything. In this case, by following Definition 7, we can update the monitor with the additional outcome to represent the cases where it should give up.*



Remote inspection case study

- We demonstrate the usefulness of our approach by applying it to a remote inspection example.
- This example is based on a simulation of an autonomous rover deployed to perform remote inspection of nuclear facilities.
- The rover has access to sensors, which are used to detect the level of radiation, and a camera, which is used to acquire images of tanks (containing radioactive material) to perform an integrity analysis (e.g., deterioration of the container).
- The objective of the rover is to patrol and inspect important locations (i.e., waypoints) around the facility.



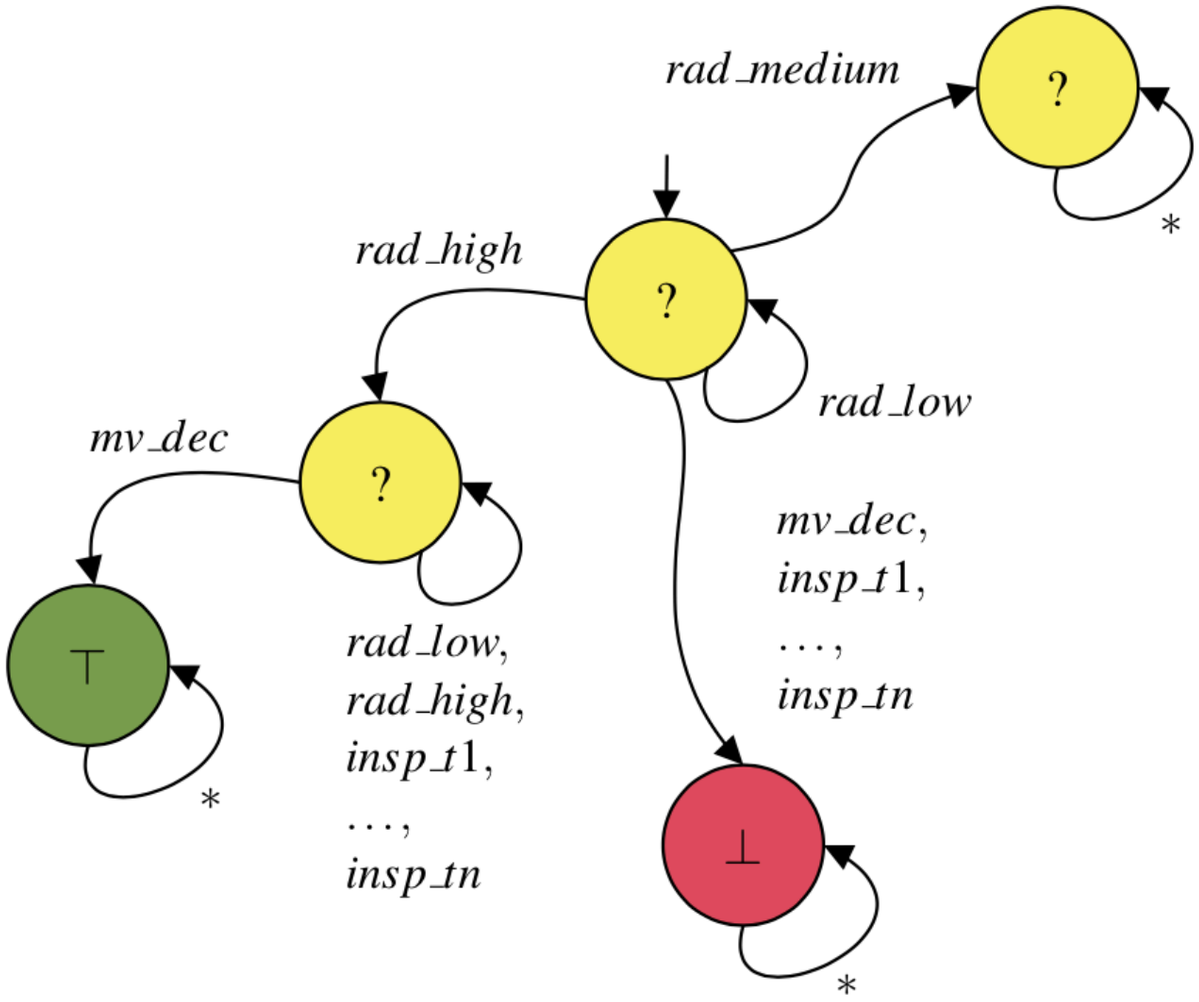
Remote inspection case study

Example 6 *Let us consider an interesting property where we can demonstrate that partial monitoring can be useful:*

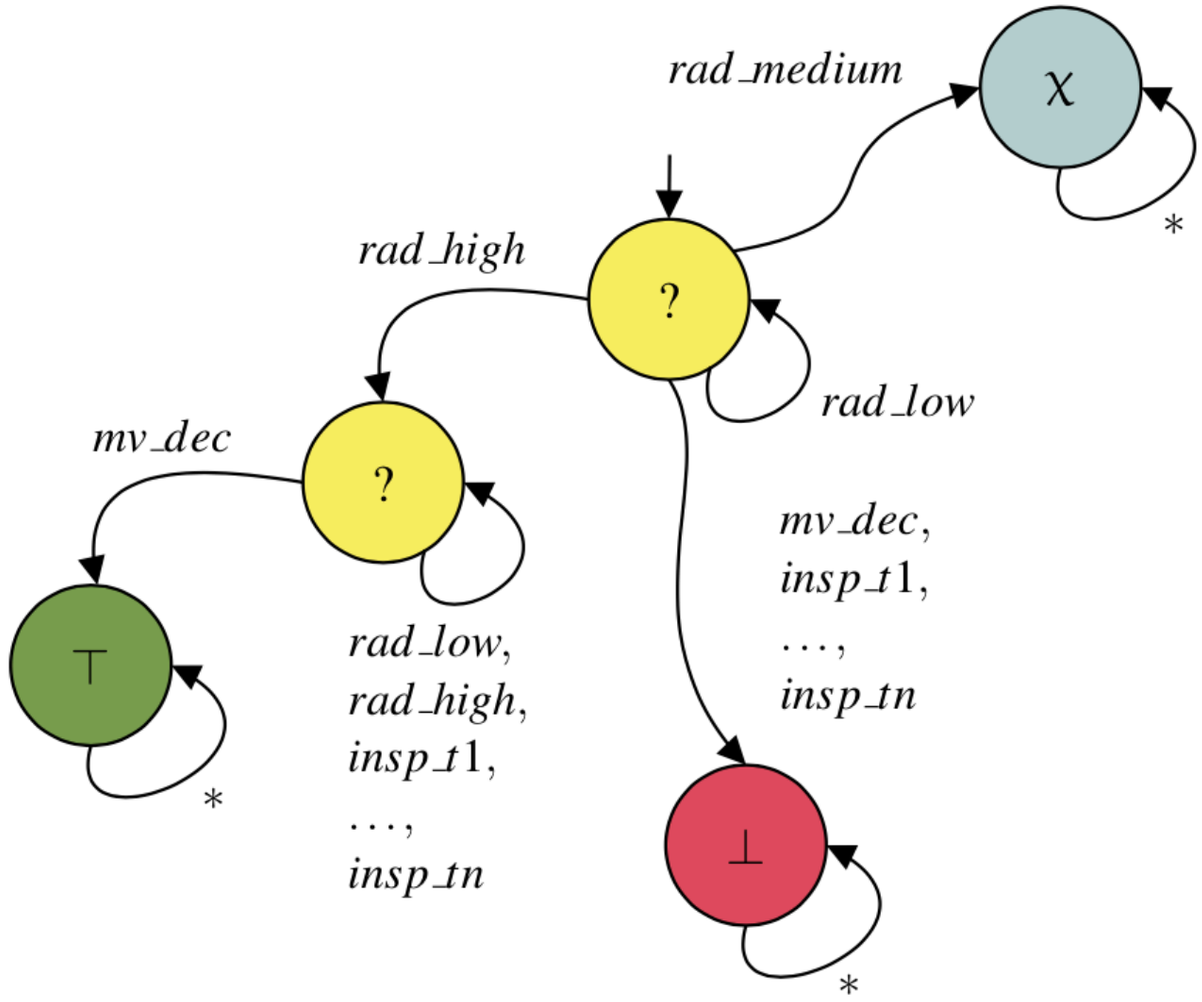
$$\varphi = radiation\_low\ \mathbf{U}((radiation\_high \wedge \Diamond move\_to\_decontamination) \vee$$
$$(radiation\_medium \wedge \Box\Diamond(inspect\_tank\_1 \vee inspect\_tank\_2 \vee \ldots \vee inspect\_tank\_n)))$$

This property says that we can observe the event radiation_low until we observe either radiation_high or radiation_medium. Low, medium, and high radiation refer to the level of radiation that is currently observed by the radiation sensor. If radiation_high is observed, then eventually we have to observe the event move_to_decontamination, which represents the command being sent to move the rover to a decontamination zone (i.e., high level of radiation can be dangerous to the rover). Otherwise, if we observe radiation_medium, then we have to patrol one of the radiation tanks (1...n) to identify if there are any abnormalities.

Remote inspection case study



(a) Monitor (as Moore machine) for the property.



(b) Updated monitor after applying our technique.

Implementation

- We implemented the transformation from standard monitor to partial monitor in Java.
  https://github.com/AngeloFerrando/PartialMonitor
- This tool depends on LamaConv, a Java library that is capable of translating expressions in temporal logics into equivalent automata, and to generate monitors out of these automata.
  https://www.isp.uni-luebeck.de/lamaconv

The process:

1. Our tool calls LamaConv and makes it generate a standard three-valued LTL monitor.
2. For each inconclusive state it performs a reachability analysis.
3. Each inconclusive state that cannot reach any non-inconclusive state is labelled with χ instead of ?

Conclusions and Future Work

Coming back to the research question:

Is it possible for monitors that are synthesised by non-monitorable properties to be used in practice?

Answer: *Yes, and …*

- We discussed the issue of handling monitors generated from non-monitorable properties and how such monitors can be extended to give up when no final verdict can be ever concluded.

- We described a practical technique to perform reachability analysis of LTL monitors obtained using standard synthesis approaches, and how the resulting partial monitors can avoid to get stuck.

- We have demonstrated the use of partial monitoring in an example from the robotics domain.

In Future Work:

- We plan to extend the approach to other formalisms.

- We want to integrate our approach with existing RV tools for autonomous systems, like ROSMonitoring [1]

[1] Angelo Ferrando, Rafael C. Cardoso, Michael Fisher, Davide Ancona, Luca Franceschini, Viviana Mascardi: ROSMonitoring: A Runtime Verification Framework for ROS. TAROS 2020: 387-399

UNIVERSITÀ DEGLI STUDI DI GENOVA

MANCHESTER 1824
The University of Manchester

Towards Partial Monitoring: It is Always too Soon to Give Up

Angelo Ferrando[1] and Rafael C. Cardoso[2]
[1] *University of Genova*
[2] The University of Manchester

angelo.ferrando@unige.it
Research Fellow

Thanks for your attention!