Predictive Runtime Verification of Skill-based Robotic Systems using Petri Nets

Validation & Verification, Model-checking,
Software Architecture, Petri net, Robotics
Baptiste Pelletier, Charles Lesire, Christophe Grand, David Doose, Mathieu Rognant

ICRA 2023

Seeking robustness in autonomous robots

Increasing need of **trust** in systems

Robots are getting **more expensive**, tasks and environments are **more complex**, with **increasing autonomy**.

→ Space, underwater, military environments...

Use of **formal methods** to develop the system

- Better **frame** specifications at various levels
- Facilitate **verification** of the behavior
- **Simplify** and make more accessible the designing process

Skillset

```
skill go_to {
    precondition {
        canmove  : lease_status == AutoMode and control_mode == Idle
        ispowered: power_status == PowerOn
    }
    start {control_mode -> Busy}
    invariant {
        is_powered {
            guard power_status == PowerOn
            effect control_mode -> Idle
        }
        is_ok {
            guard lease_status == AutoMode
    }
    }
    interrupt {
        interrupting true
        effect control_mode -> Idle
    }
    success is_arrived {effect control_mode -> Idle}
    failure not_arrived {effect control_mode -> Idle}
}
```

Abstract of a skillset: skill `go_to` and its semantic.

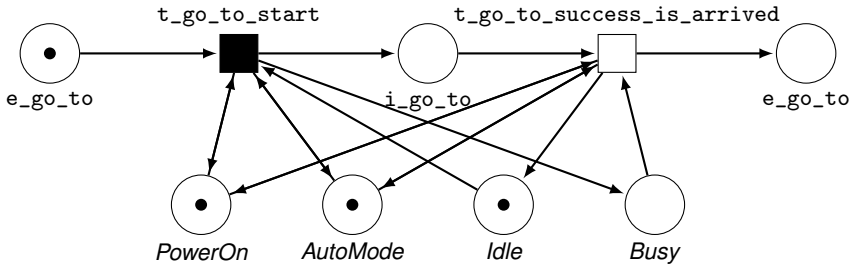
A Skillset model is presented as a **.rl** textfile, with the specification for the model execution, including:

- **Resources**: represent internal or external elements of the system
- **Events**: uncontrollable elements that can change the value of resources
- **Skills**: elementary actions of the system, constrained by resource states and capable of changing them

SkiNet Tool: Offline Verification using Petri nets

Translation of skillset model into Petri net

- Petri nets = set of places, transitions and arcs
- Ideal to model state-machines with concurrency and resource sharing.
- Numerous tools available online for analysis (model-checking)



B. Pelletier, C. Lesire, D. Doose, K. Godary-Dejean, and C. Dramé-Maigné, "Skinet a petri net generation tool for the verification of skillset-based autonomous systems," in Fourth Workshop on Formal Methods for Autonomous Systems. EPTCS, 2022, pp. 1–19.

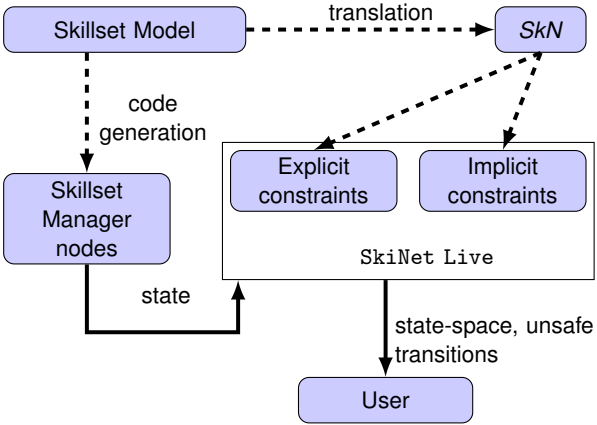
Limits of offline verification

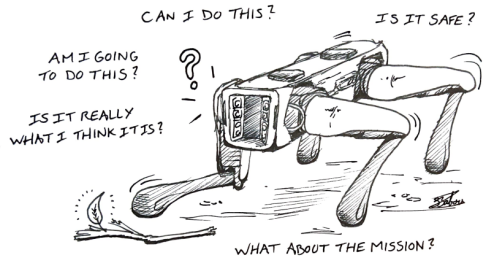
Offline verification is limited if the system is too complex (state-space explosion) or cannot guarantee properties safeness (states that violate properties exist and cannot be removed from the model).
An extension of the tool was developped to **monitor** skillsets execution and **predict the violation of properties**. This verification can be made on a **finite horizon**, thus solving the state-space explosion problem. A **partial state-space** is used and **regurlarly updated** during runtime.

Two types of constraints to observe

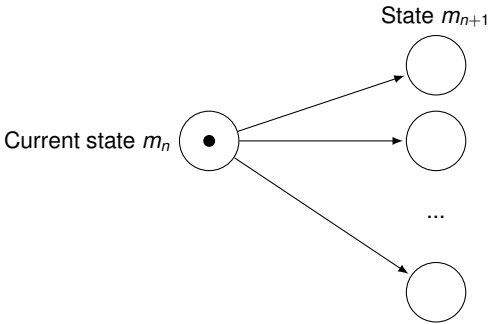
- ***Explicit* constraints**: states that **directly** violate one or more properties.
- ***Implicit* constraints**: states that will **inevitably, in the future**, lead to the violation of one or more properties.

SkiNet Live


Skillset Model
translation
SkN
code generation
Skillset Manager nodes
Explicit constraints
Implicit constraints
SkiNet Live
state
state-space, unsafe transitions
User


CAN I DO THIS?
IS IT SAFE?
AM I GOING TO DO THIS?
IS IT REALLY WHAT I THINK IT IS?
WHAT ABOUT THE MISSION?

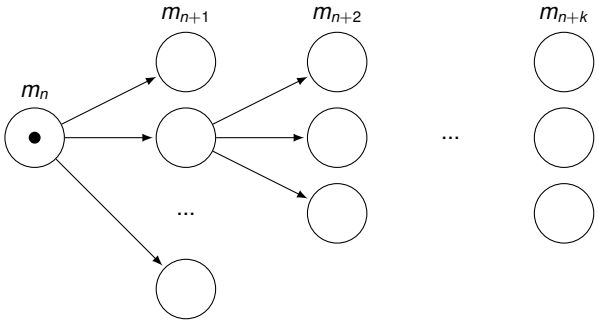
Immediate explicit constraints - Quick search



Quick search of **immediate** explicit constraints is easily implemented:

- Running state m_n
- Fire available transitions one by one
- If state m_{n+1} violates one or more properties $\rightarrow$ **Dangerous state**

Implicit and Explicit constraints - State-space search

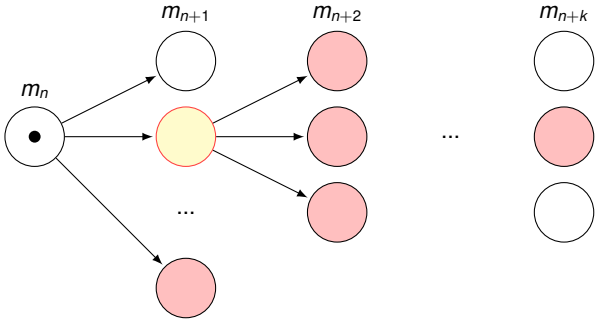


Slower exploration of the next states, jusqu'à une limite de temps ajustable:

- Running state m_n
- Fire transitions one by one
- For **every** state m_{n+1}, fire all transitions,
- ...
- When $t_{explo} = t_{timeout}$: **stop**

Partial state-space μ obtained contains explicits and implicits **until depth** μ_{depth}.

Implicit and Explicit constraints - State-space search



With CTL: **explicit** and **implicit** constrainted are obtained with

$$AF\neg\phi$$

We obtain all states that violate, or will inevitably lead to the violation of the properties.

SkiNet Live: experiment with Spot+Jaco

Mission Protocol

- go_to_waypoint with Spot to reach trap location
- teleoperation of Spot with teleop skill for final approach and positioning
- deploy Jaco robotic arm with arm_to_ready skill
- teleoperation of the arm with arm_joystick skill
- fold arm and move to the next trap location

The following properties must be respected throughout the mission to guarantee system safety:

$$\texttt{action\_guard} = \textit{not}\ (\texttt{spot} : \textit{Busy}\ \textit{and}\ \texttt{manipulator} : \textit{Busy}) \tag{1}$$

$$\texttt{safety\_guard} = \textit{not}\ (\texttt{spot} : \textit{i\_go\_to\_waypoint}\ \textit{and}\ \textit{not}\ \texttt{manipulator} : \textit{HP}) \tag{2}$$

Thank you for listening!!

Please come check out my poster and video presentations for ICRA 2023!

baptiste.pelletier@onera.fr

https://gitlab.com/onera-robot-skills/skinet-release