

# Monitoring

(In practice)



**UNIVERSITÀ  
DEGLI STUDI  
DI TRIESTE**

# Outline


- Fast Recap
- Tools and Setup
- How to code a monitor (ANTLR)
- Moonlight
  - Getting started
  - Simple example
  - Bike example
  - Pancreas
  - Gillespy2

# Reason for Monitoring?

- Model Checking suffers from state-space explosion

Monitoring is easier than model checking and it allows to react to observed behaviors satisfying or violating certain properties.

# Monitoring

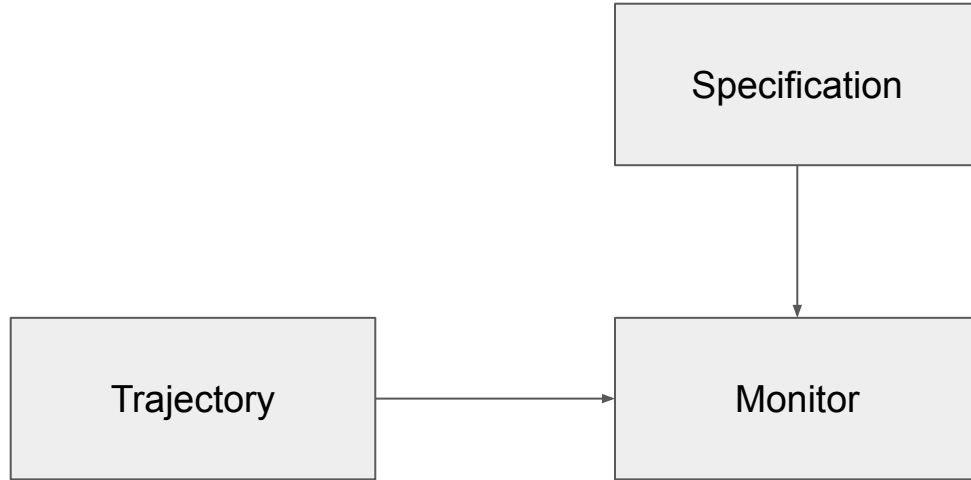


Specification

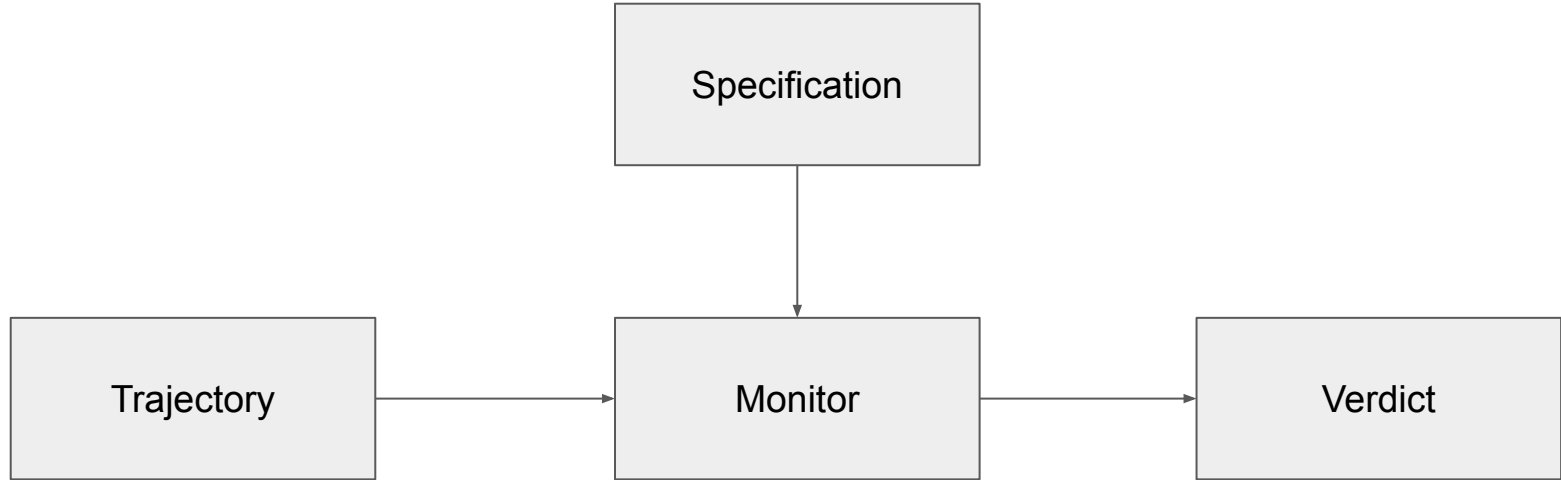
A diagram consisting of two light gray rectangular boxes with black borders. The box labeled 'Specification' is positioned in the upper right area, and the box labeled 'Trajectory' is positioned in the lower left area. There are no lines or arrows connecting the two boxes.

Trajectory

# Monitoring

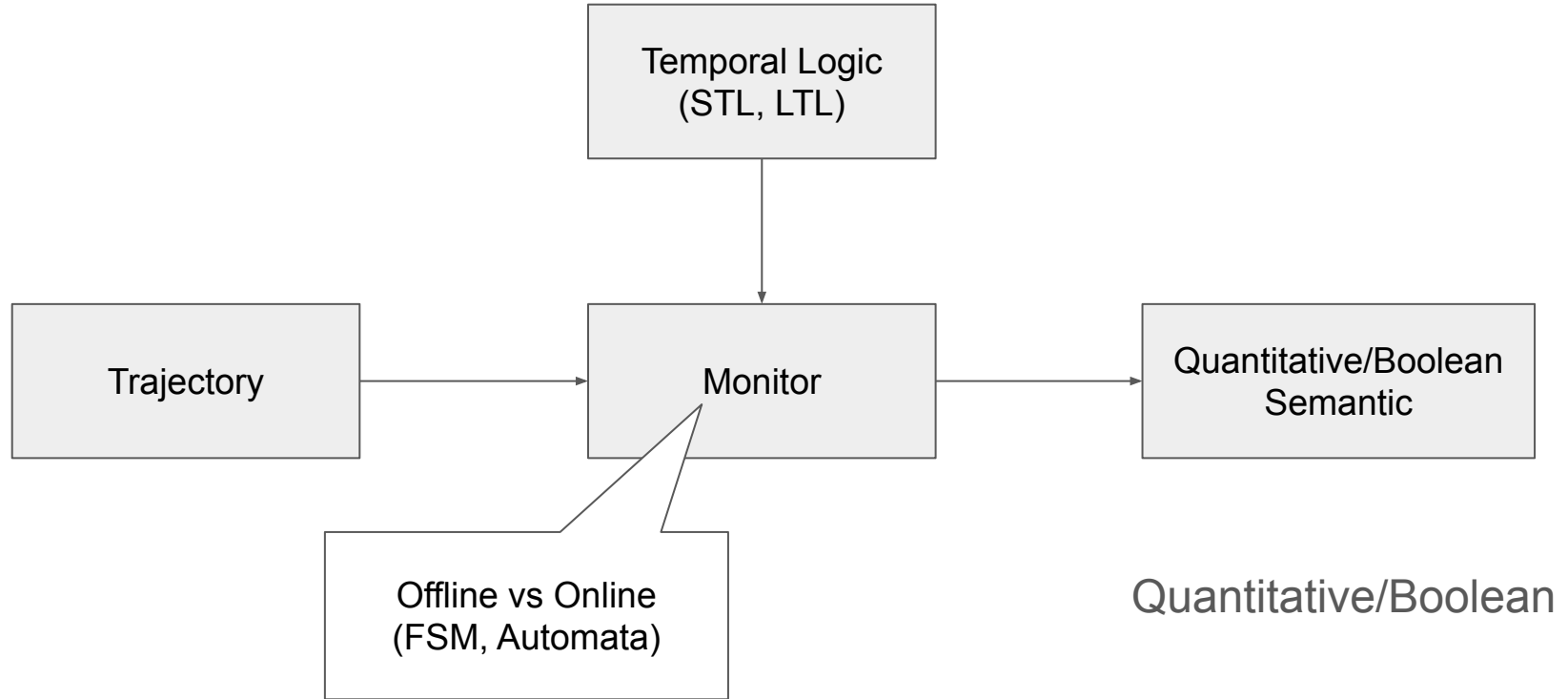


# Monitoring

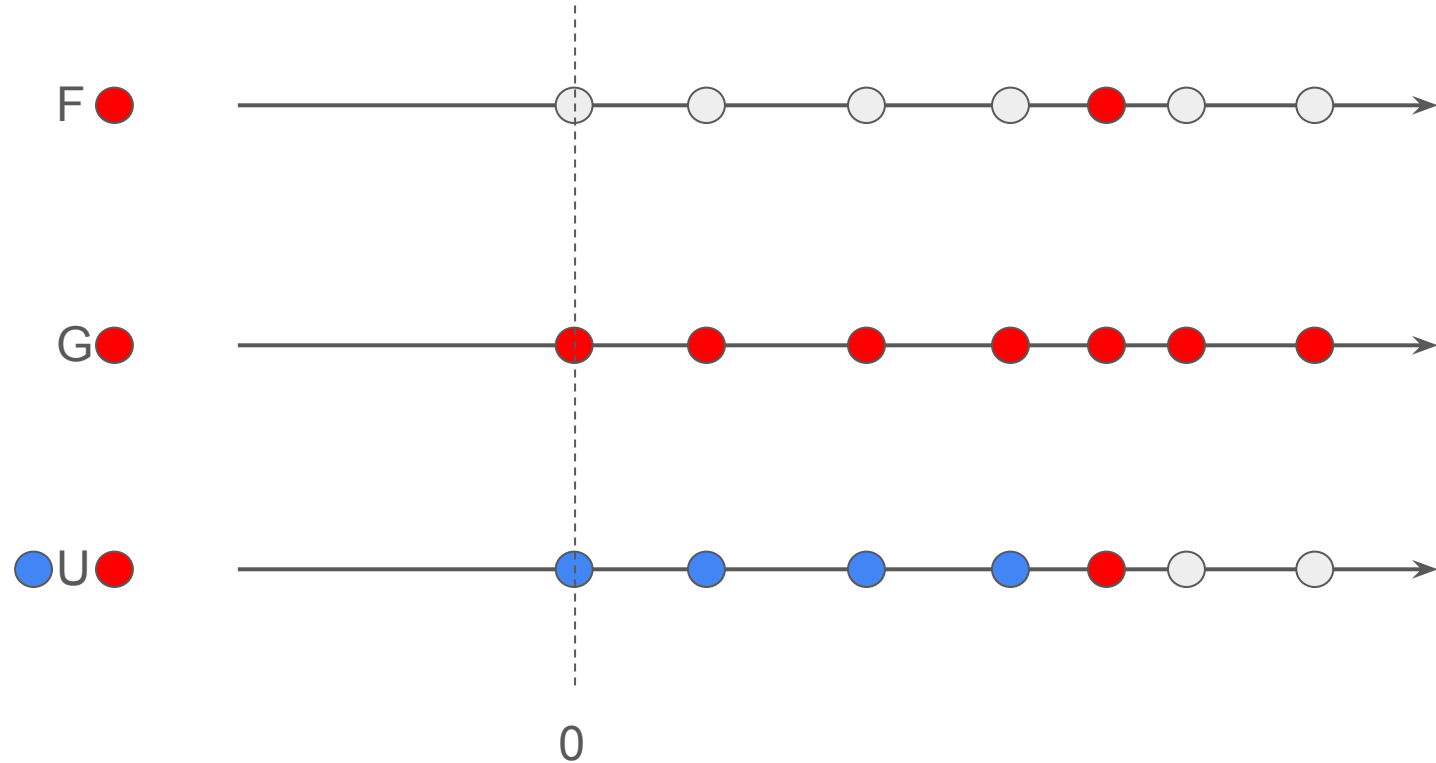


Quantitative/Boolean

# Monitoring



# LTL - Temporal operators





# Specifying via STREL logic

$$\begin{aligned} \varphi := & \top \mid \perp \mid p \circ c \mid \neg \varphi \mid \varphi \vee \varphi \mid \\ & \varphi \mathcal{U}_I \varphi \mid \\ & \varphi \mathcal{R}_{\leq d} \varphi \mid \mathcal{E}_{\geq d} \varphi \end{aligned}$$

Diagram illustrating the structure of STREL logic formulas:

- Signal Temporal Logic (blue bracket) covers the first three lines of the formula definition.
- Signal Temporal Reach & Escape Logic (purple bracket) covers the last two lines of the formula definition.

1. Maler O., Nickovic D. - Monitoring Temporal Properties of Continuous Signals. FTRTFT 2004, FORMATS 2004.

2. Bartocci E., Bortolussi L., Loretto M., and Nenzi L. - Monitoring mobile and spatially distributed cyber-physical systems. MEMOCODE '17

# Atomic propositions

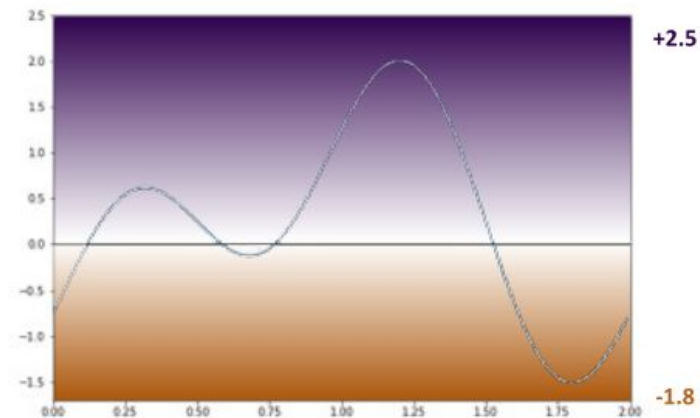
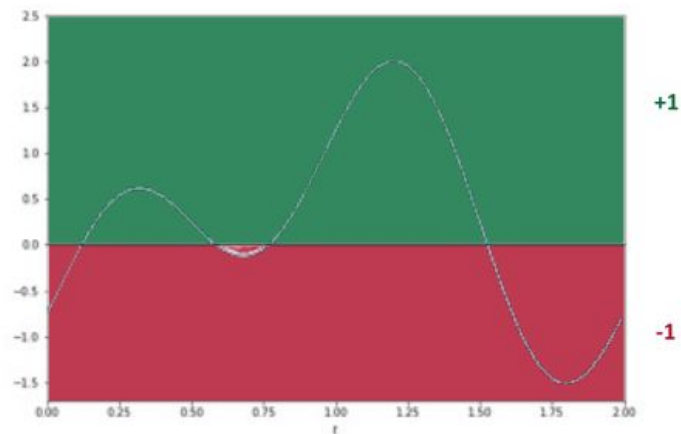
*Inequalities over signals*

$$p > c$$

$$\begin{array}{c} \chi \\ 1 \mid -1 \end{array}$$

$$\begin{array}{c} \rho \\ x \in \mathbb{R} \end{array}$$

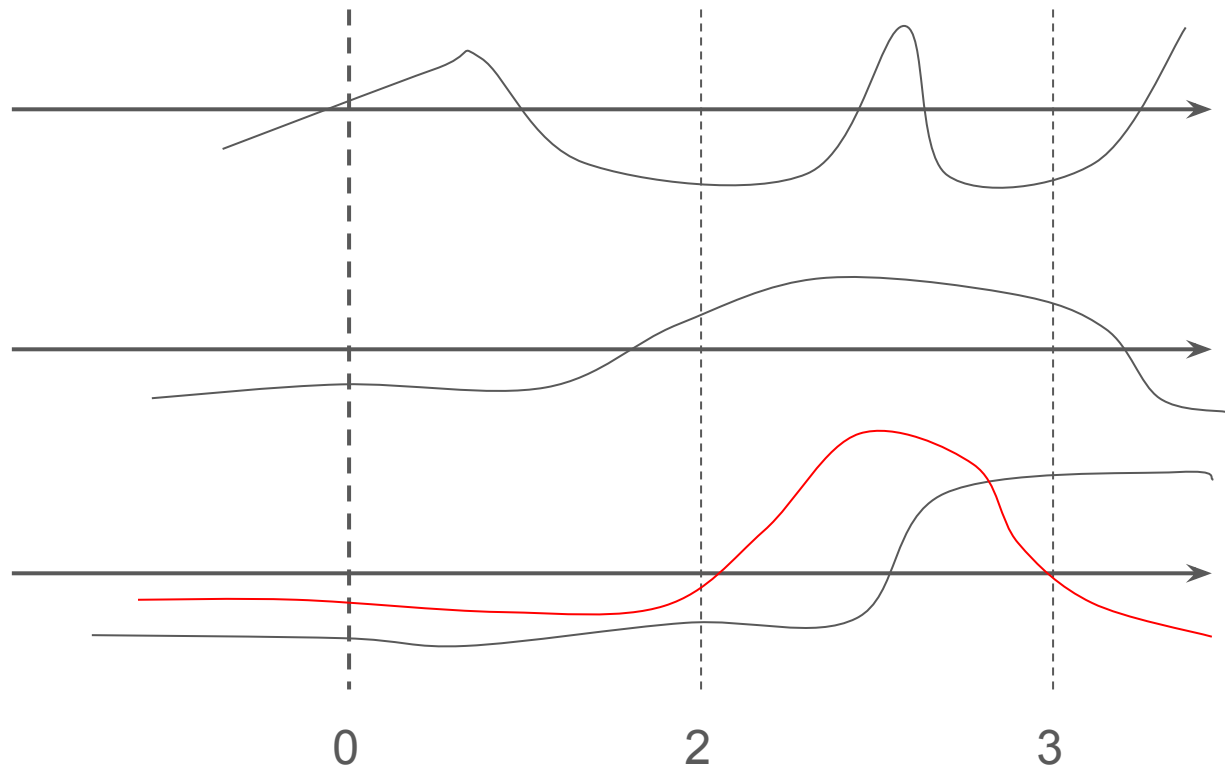
*how much above/below*

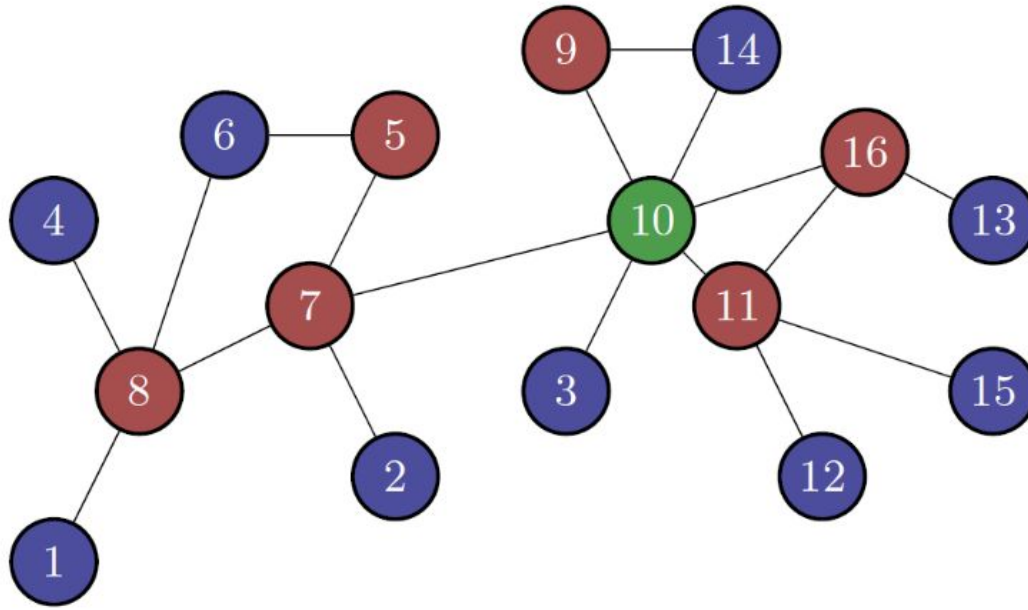


$F[2,3] \ X > 0$

$G[2,3] \ X > 0$

$X < 0 \ U[2,3] \ Y > 0$



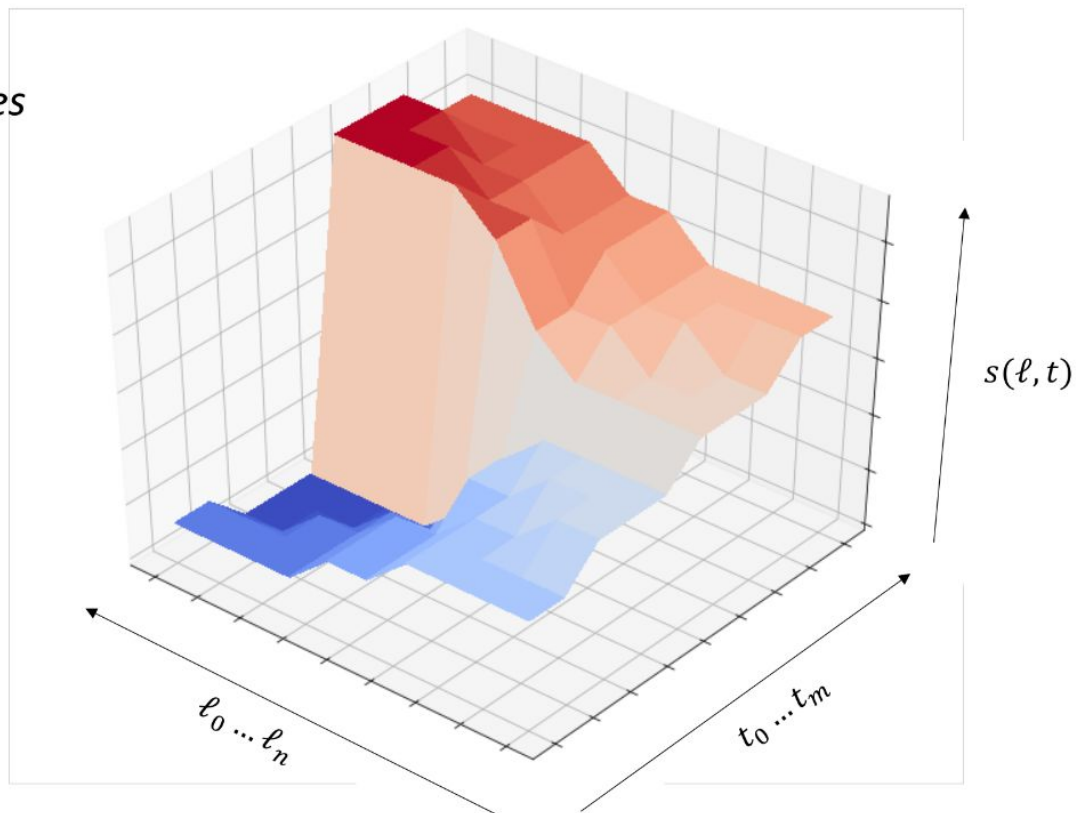


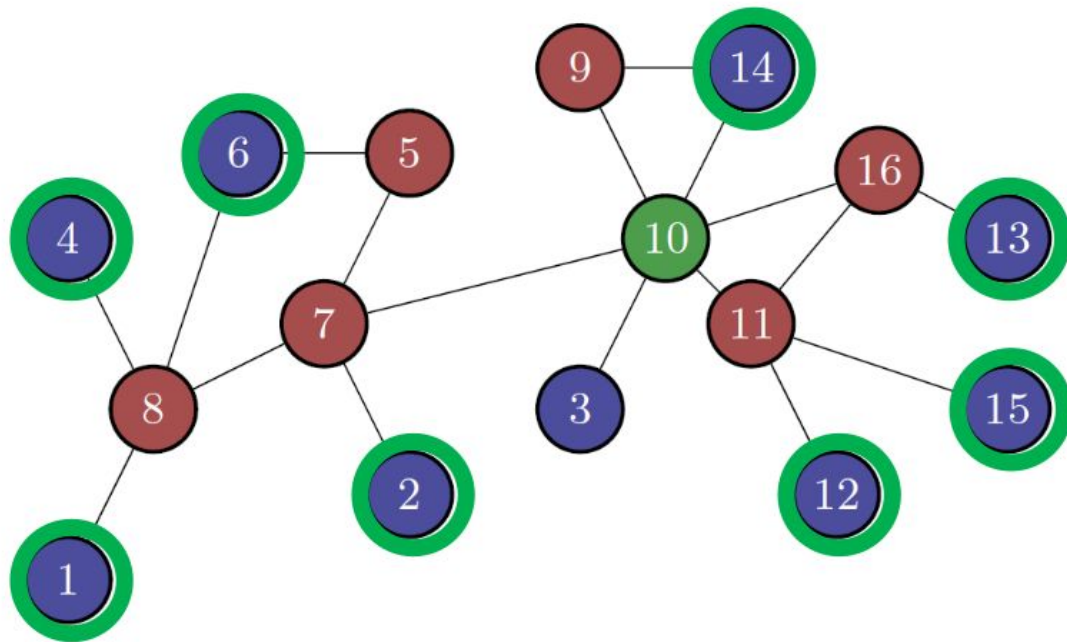
*From L. Nenzi, E. Bartocci, L. Bortolussi, and M. Loreti.  
A logic for monitoring dynamic networks of spatially-distributed cyber-physical systems, 2021.*

# Signals

*Signals defined over real values*

$$s: L \times T \rightarrow \mathbb{R}^n$$

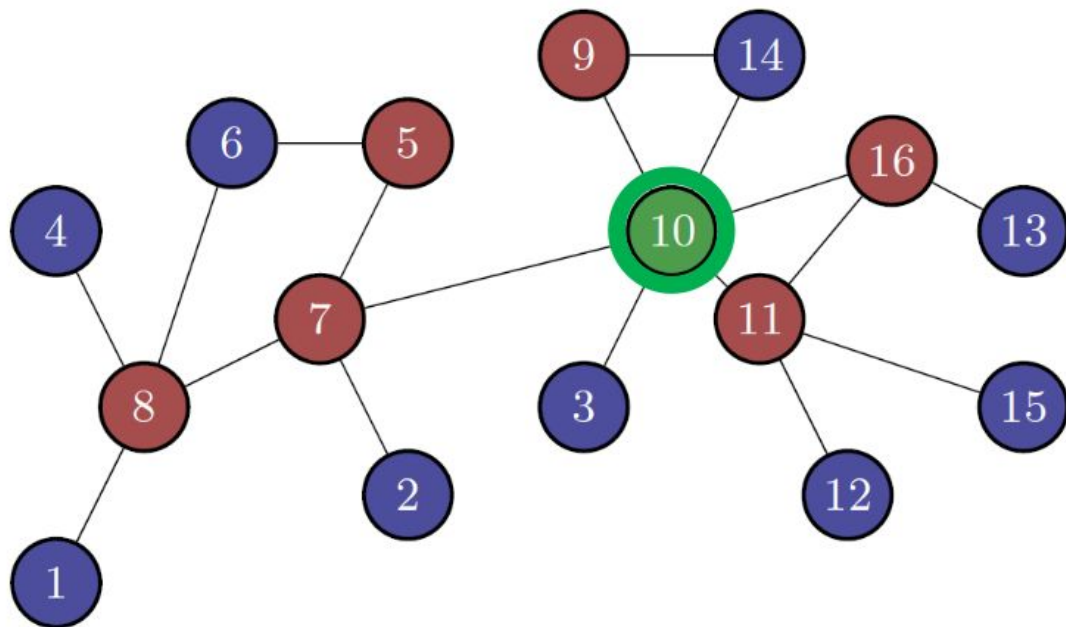




Reachability: *blue*  $\mathcal{R}_{\leq 1}$  *red*.

From L. Nenzi, E. Bartocci, L. Bortolussi, and M. Loreti.

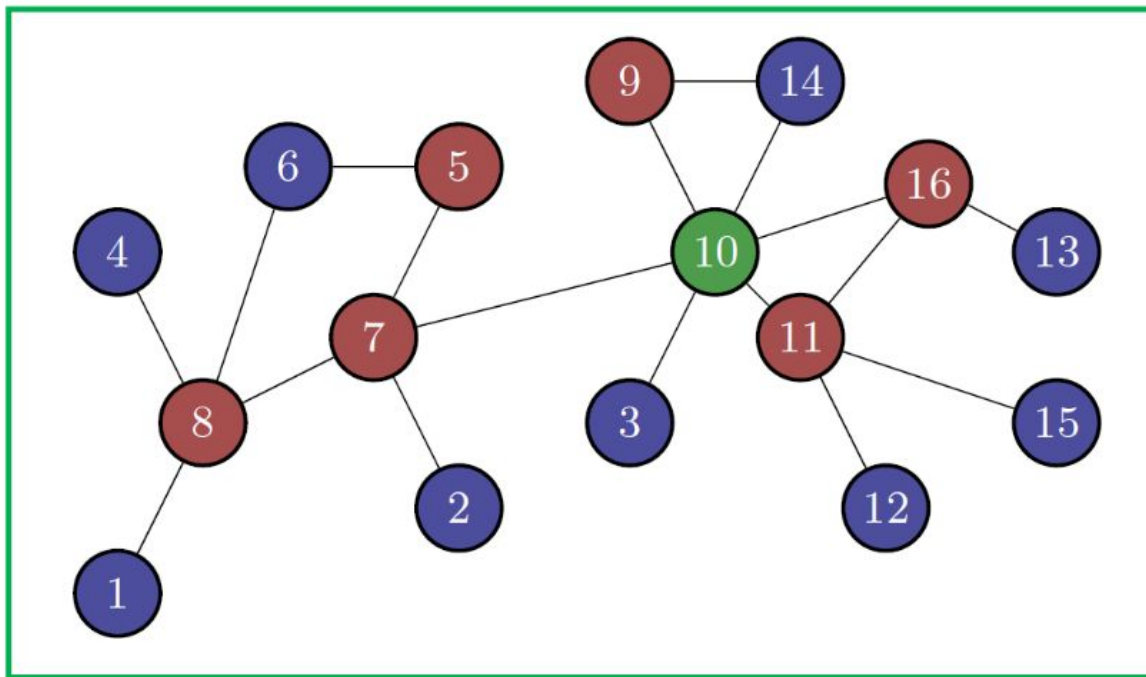
*A logic for monitoring dynamic networks of spatially-distributed cyber-physical systems, 2021.*



**Escape:**  $\mathcal{E}_{\geq 2} \neg \text{blue}$ .

From L. Nenzi, E. Bartocci, L. Bortolussi, and M. Loreti.

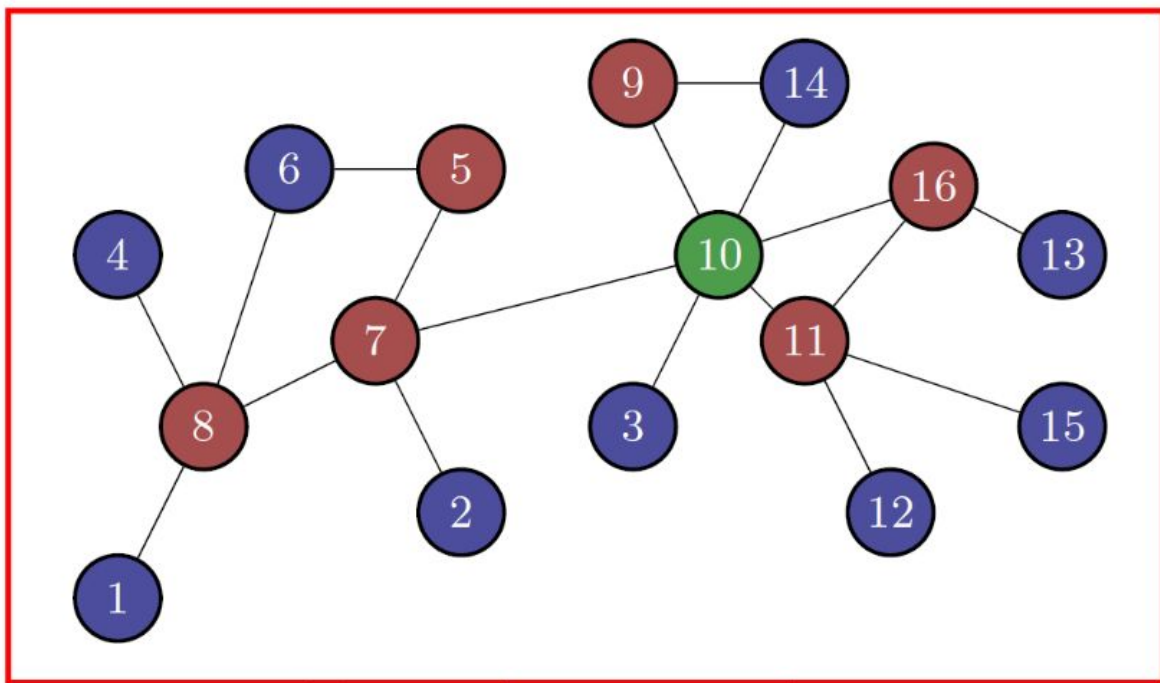
*A logic for monitoring dynamic networks of spatially-distributed cyber-physical systems, 2021.*



Somewhere:  $\Diamond_{\leq 4} \text{green.}$

From L. Nenzi, E. Bartocci, L. Bortolussi, and M. Loreti.  
*A logic for monitoring dynamic networks of spatially-distributed cyber-physical systems*, 2021.





Everywhere:  $\Box_{\geq 2} red$ .

From L. Nenzi, E. Bartocci, L. Bortolussi, and M. Loreti.

*A logic for monitoring dynamic networks of spatially-distributed cyber-physical systems, 2021.*

# Tools



**Python 3.8+**



**JDK 21+**

*Visual Studio Code*



*PyCharm*



*IntelliJ*

**IDEs**

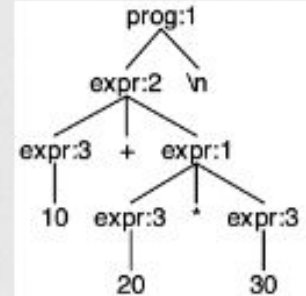


# ANTLR (ANother Tool for Language Recognition)

is a powerful parser generator for reading, processing, executing, or translating structured text or binary files. It's widely used to build languages, tools, and frameworks. From a grammar, ANTLR generates a parser that can build and walk parse trees.

```
grammar Expr;
prog:  (expr NEWLINE)* ;
expr:  expr ('*' | '/' ) expr
      | expr ('+' | '-' ) expr
      | INT
      | '(' expr ')'
      ;
NEWLINE : [\r\n]+ ;
INT      : [0-9]+ ;
```

```
$ antlr4-parse Expr.g4 prog -gui
10+20*30
^D
$ antlr4 Expr.g4 # gen code
$ ls ExprParser.java
ExprParser.java
```





# ANTLR (ANother Tool for Language Recognition)

```
grammar Expr;
prog:  (expr NEWLINE)* ;
expr:  expr ('*' | '/' ) expr
      | expr ('+' | '-' ) expr
      | INT
      | '(' expr ')'
      ;
NEWLINE : [\r\n]+ ;
INT      : [0-9]+ ;
```

ANTLR with visitor  
pattern



Lexer



Parser



Visitor