

CLEARSY

Safety Solutions Designer

AIX
LYON
PARIS
STRASBOURG

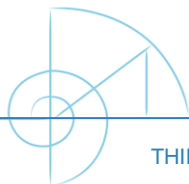
WWW.CLEARSY.COM

Hackathon
JUL2023

Subject #1

Traffic Lights Control

Thierry Lecomte
R&D Director



THIERRY.LECOMTE@CLEARSY.COM



Attribution 4.0 Unported (CC BY 4.0)

Modelling Systems

≡ Read carefully, understand the subject, define « properties » in natural language

≡ Define your modelling approach (how do you represent concepts and links)

- Constants, variables
- Operations

≡ Model the system, verify that you can (easily) express « properties », ensure that the model is provable

- Prove automatically (force 0 and 1)
- Animate with ProB and check that for some scenarios the invariant is verified

Properties with the B Mathematical Language

≡ Modelling language based on set theory and first order predicates logic

Let the set $\text{TrackCircuit} = \{t1, t2, t3, t4, t5\}$

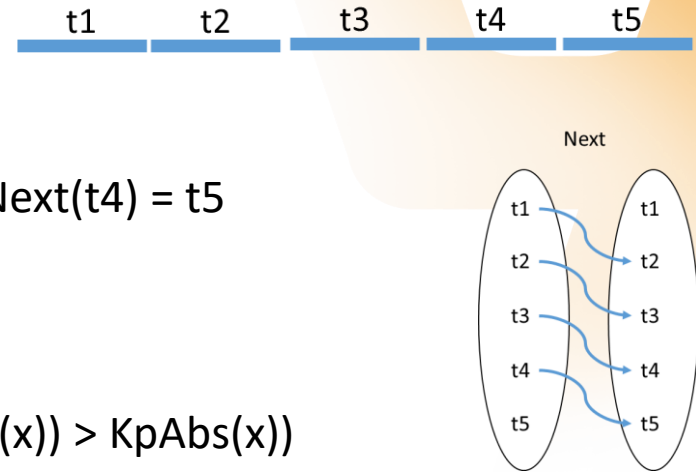
Let the function $\text{Next} \in \text{TrackCircuit} \rightarrow \text{TrackCircuit}$

Example: $\text{Next}(t1) = t2, \text{Next}(t2) = t3, \text{Next}(t3) = t4, \text{Next}(t4) = t5$

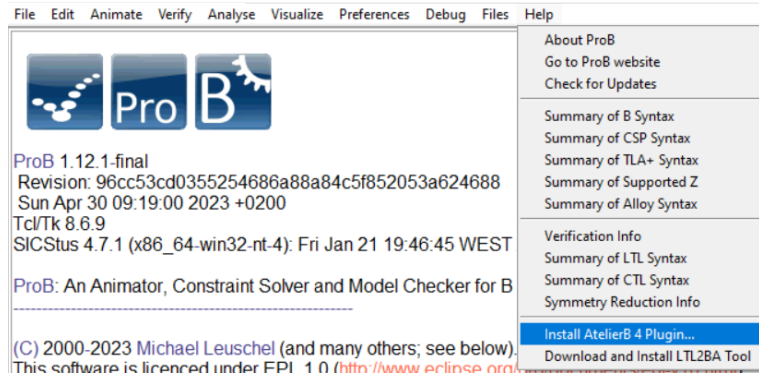
$\text{Next} = \{t1 \mapsto t2, t2 \mapsto t3, t3 \mapsto t4, t4 \mapsto t5\}$

Let the function $\text{KpAbs} : \text{TrackCircuit} \rightarrow \mathbb{N}$

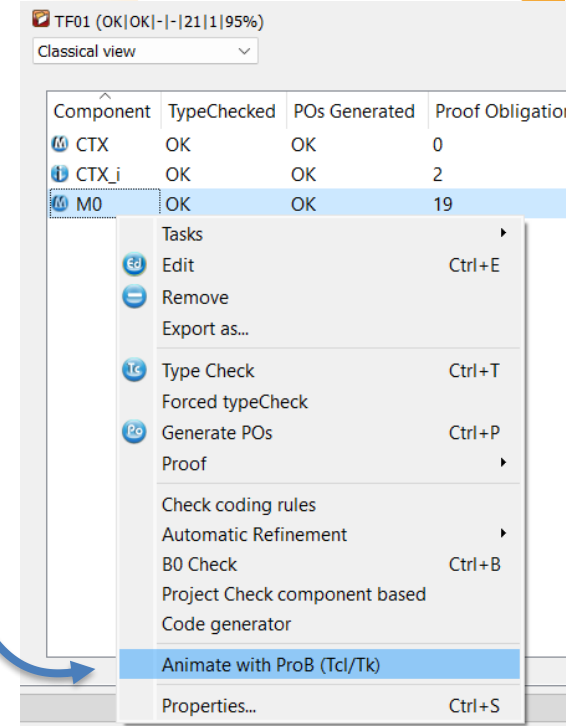
$\forall x. (x \in \text{TrackCircuit} \wedge x \in \text{dom}(\text{Next}) \Rightarrow \text{KpAbs}(\text{Next}(x)) > \text{KpAbs}(x))$



Add Prob tool to Atelier B menu



- ▶ Open ProB Help menu
- ▶ Install Atelier B 4 plugin
- ▶ Select the Atelier B file of your install directory
- ▶ probtcl.etoole file created in extensions subdirectory



Objectives

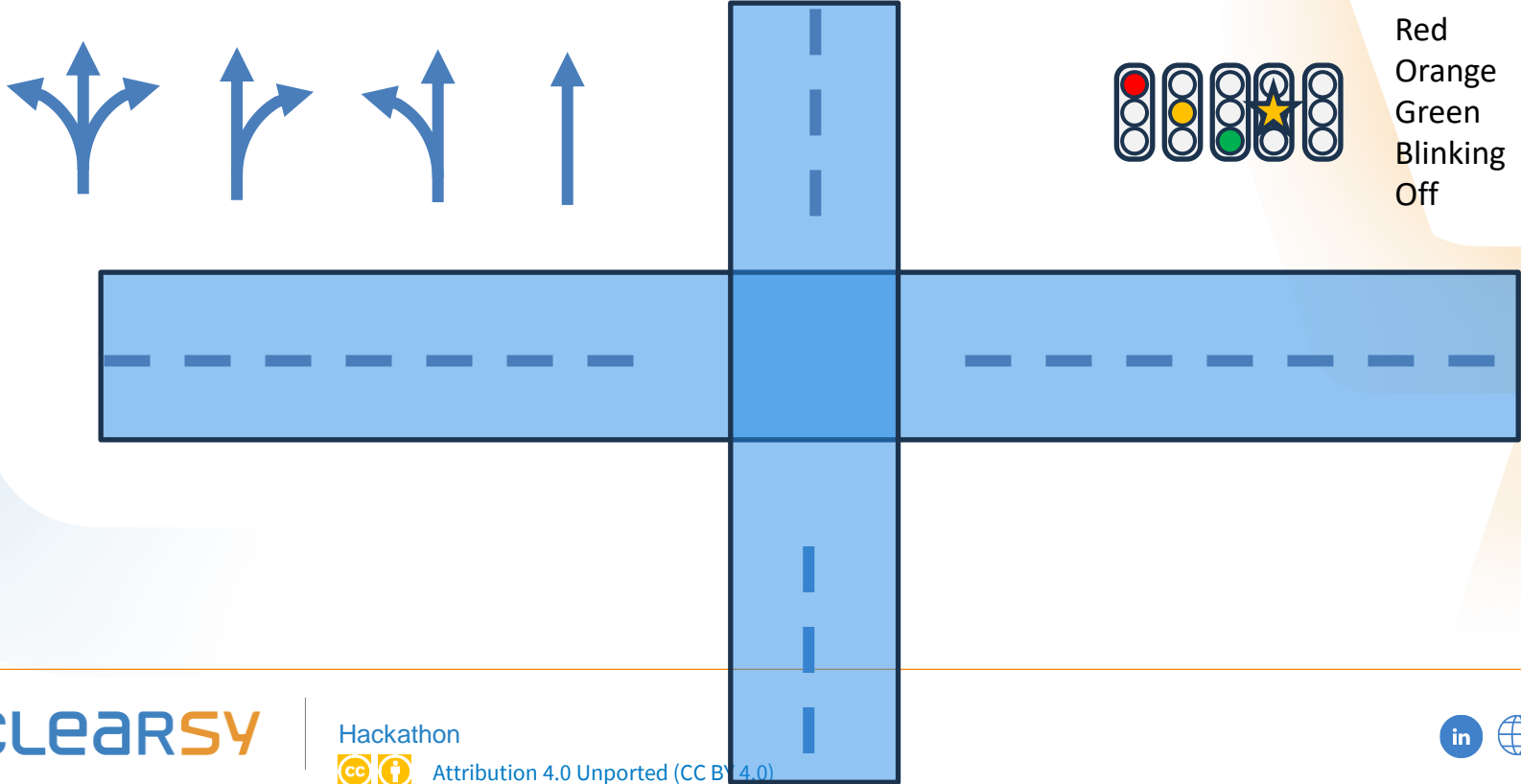
- ▶ Model a traffic light controller (1/2)
 - ▷ Animate the lights (a bit simplified),
 - ▷ no temporal behaviour,
 - ▷ Only specification (no implementation)
- ▶ CTX machine given
- ▶ M0 machine to complete
 - ▷ some invariant, most operations
 - ▷ Check with proof and ProB
- ▶ Questions to answer

Objectives

- ▶ Model a traffic light controller (2/2)
 - ▷ With CLEARSY Safety Platform as a safeguard to control the behaviour
 - ▷ Specification and implementation
 - ▷ Compile with IDE, play with simulator
 - ▷ Add temporal behaviour
- ▶ Project skeleton created – user_logic OPERATION to specify and implement
- ▶ Questions to answer

Traffic Lights Controller 1

Simple configuration



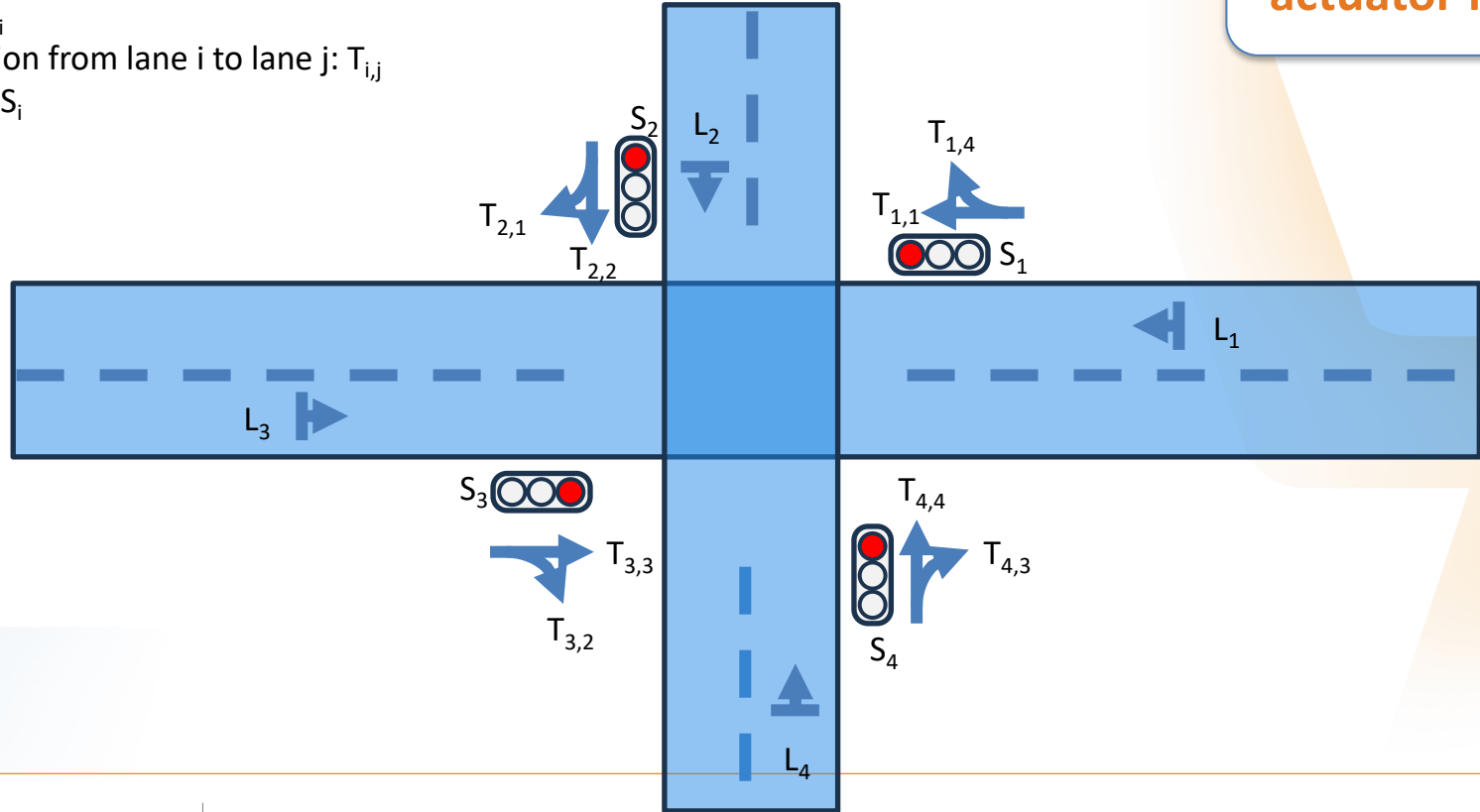
Legend:

Lane: L_i

Transition from lane i to lane j : $T_{i,j}$

Signal: S_i

Naming



No sensor or
actuator failure

Definitions, Constraints & Properties

SETS

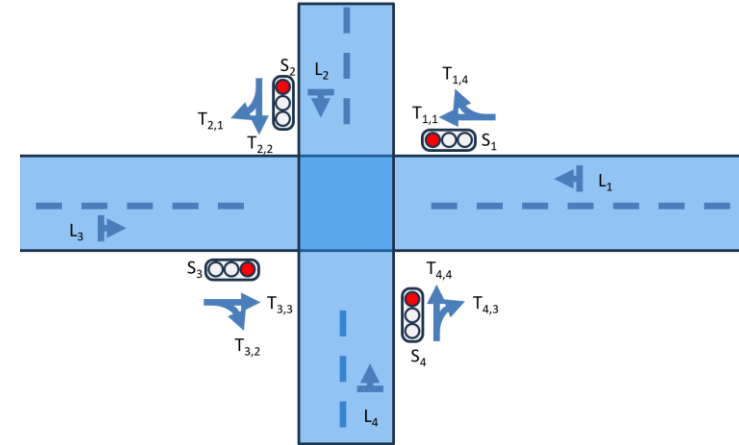
```
LANES = {L1, L2, L3, L4};  
SIGNALS = {S1, S2, S3, S4}
```

CONSTANTS

```
INTERSECT_LANES,  
TRANSITIONS
```

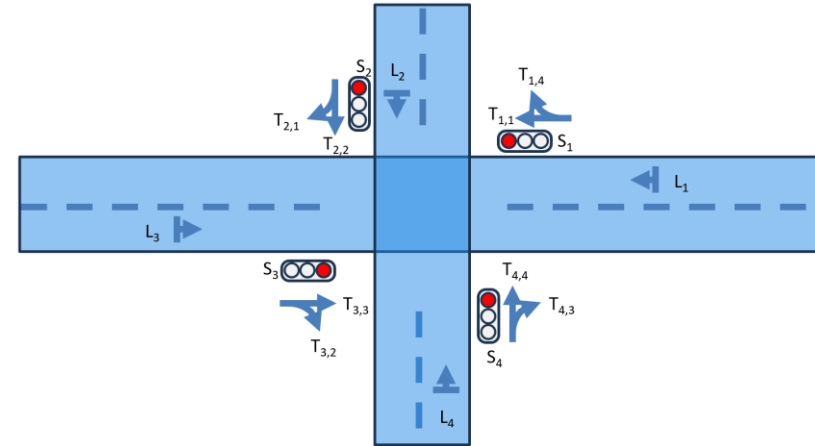
PROPERTIES

```
INTERSECT_LANES <: LANES * LANES &  
TRANSITIONS : SIGNALS <-> ( LANES * LANES ) &
```



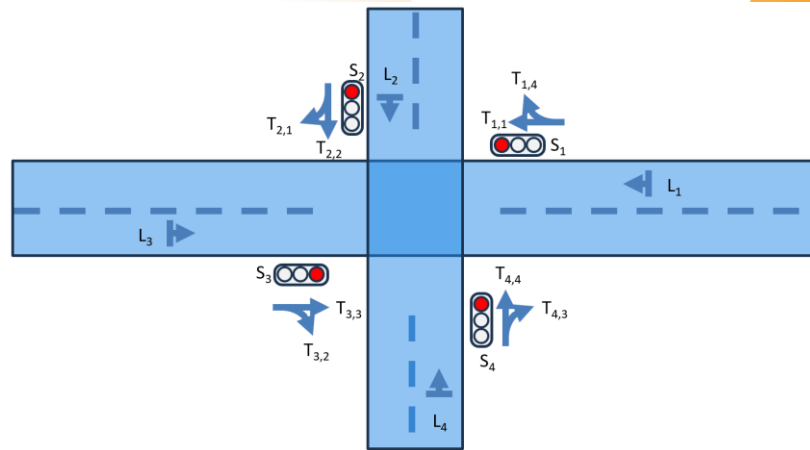
Definitions, Constraints & Properties

```
INTERSECT_LANES = {  
    L1 |-> L2, L1 |-> L4,  
    L2 |-> L1, L2 |-> L3,  
    L3 |-> L2, L3 |-> L4,  
    L4 |-> L1, L4 |-> L3}
```



Definitions, Constraints & Properties

```
TRANSITIONS = {  
    S1 |-> (L1 |-> L1) ,  
    S1 |-> (L1 |-> L4) ,  
    S2 |-> (L2 |-> L2) ,  
    S2 |-> (L2 |-> L1) ,  
    S3 |-> (L3 |-> L3) ,  
    S3 |-> (L3 |-> L2) ,  
    S4 |-> (L4 |-> L4) ,  
    S4 |-> (L4 |-> L3)  
}
```



Definitions, Constraints & Properties

VARIABLES

```
rs, /* red signals */  
os, /* orange signals */  
gs, /* green signals */  
bs /* blinking (orange signals) */
```

INVARIANT

```
rs <: SIGNALS &  
os <: SIGNALS &  
gs <: SIGNALS &  
bs <: SIGNALS &  
(bs = SIGNALS or bs = {}) & /* all blinking or none blinking */  
rs \/ os \/ gs \/ bs = SIGNALS & /* coherency */  
rs /\ os /\ gs /\ bs = {} /* coherency */  
.. ..  
.. ..
```

Definitions, Constraints & Properties

INITIALISATION

```
rs := {} ||  
os := {} ||  
gs := {} ||  
bs := SIGNALS
```

OPERATIONS

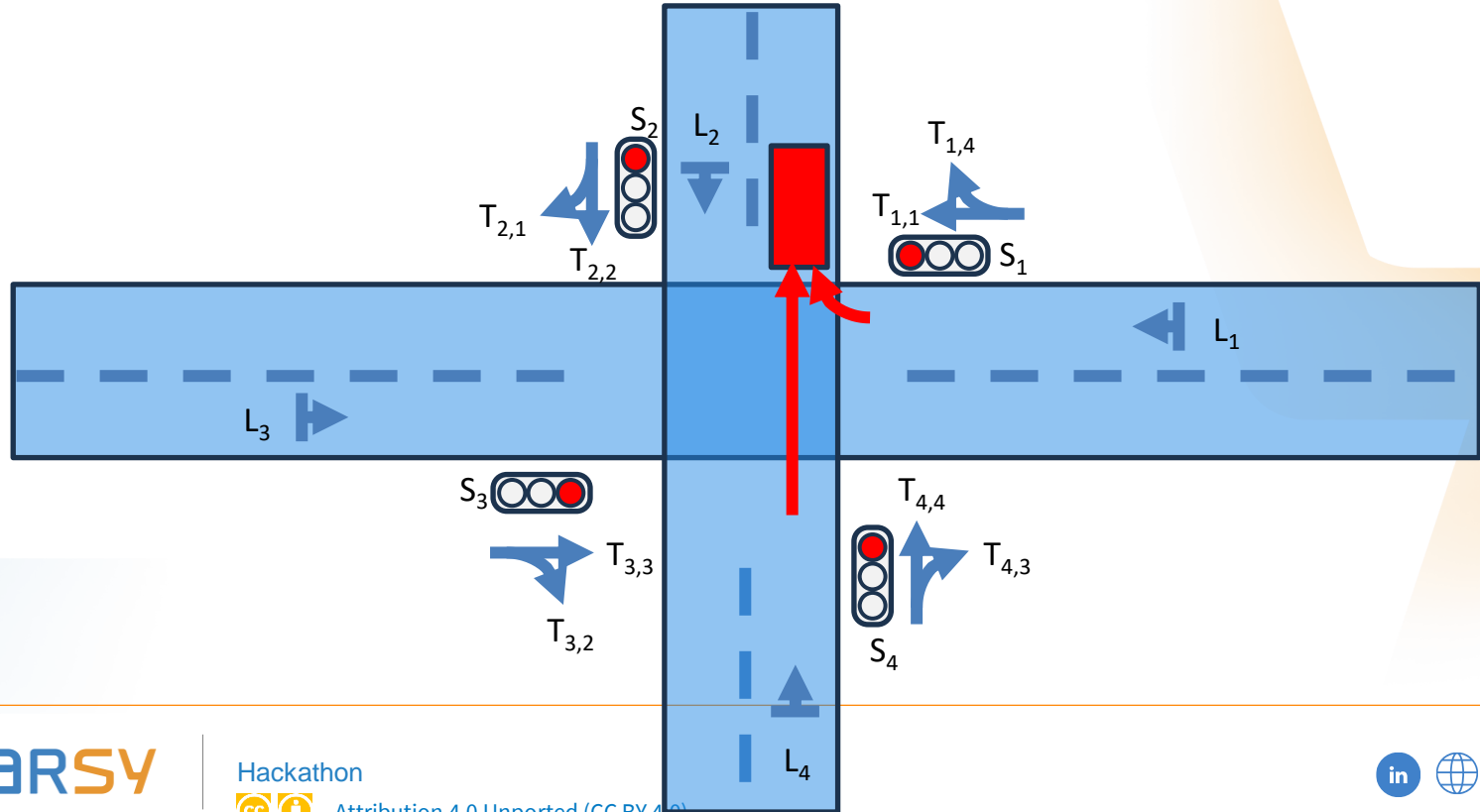
```
reset_turn_on = /* no precondition, all signals on orange blinking */
```

BEGIN

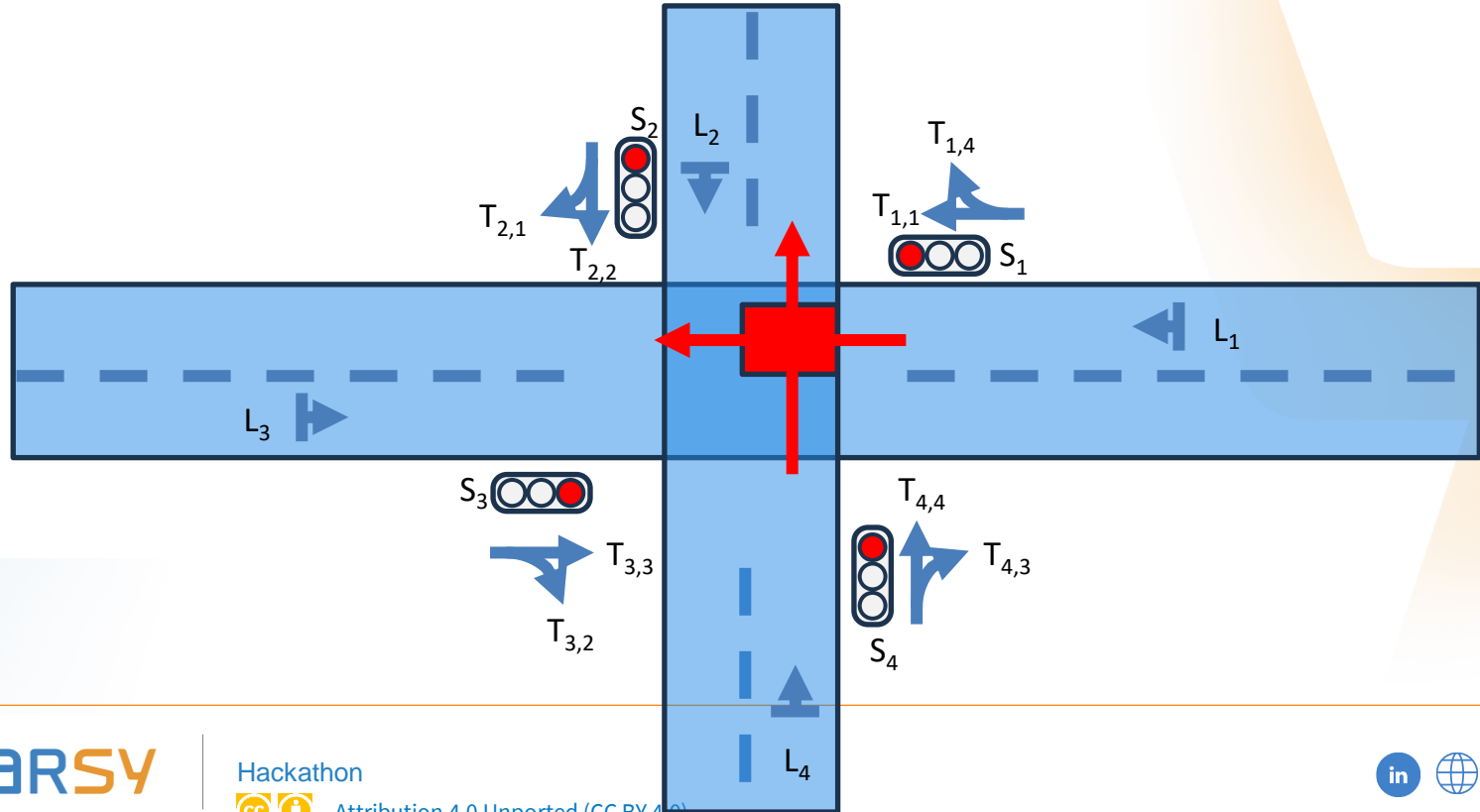
```
rs := {} ||  
os := {} ||  
gs := {} ||  
bs := SIGNALS
```

END;

Collision #1



Collision #2



Activities

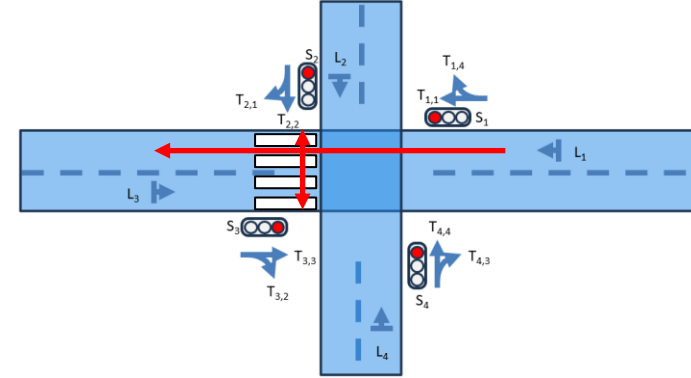
- Add invariant to avoid collision#1 and #2
- Specify the operations
 - `start_exploitation`: moving from all blinking to all orange
 - `orange_to_red`: moving from all orange to all red
 - `red_to_green1`: set S1 and S3 to green
 - `red_to_green2`: set S2 and S4 to green
 - `green_to_orange`: set green signals to orange
- Check the proof,
- Animate with ProB, check that “known scenarios” are playable, check invariants

Activities

- Add a `red_to_green` operation with dangerous signals activated, check that it is detected by ProB (invariant broken).
- It is possible to have signals going from red to green while some signals are orange (going to red). Reinforce the preconditions to avoid it.
- Add another state corresponding to all lights are off, change the precondition of `reset_turn_on`, add an operation `turn_off`.

Activities

- Complete the model with pedestrian crossing signal (red or green)(1 for each lane).
Constants for the crossings PC1, PC2, PC3, PC4 and link with LANES, variables for specific lights (rpc, gpc)
- Complete the invariant (*pedestrians are not allowed to cross if a green lane is going through the crossing*), the pre-and post conditions of the OPERATIONS.

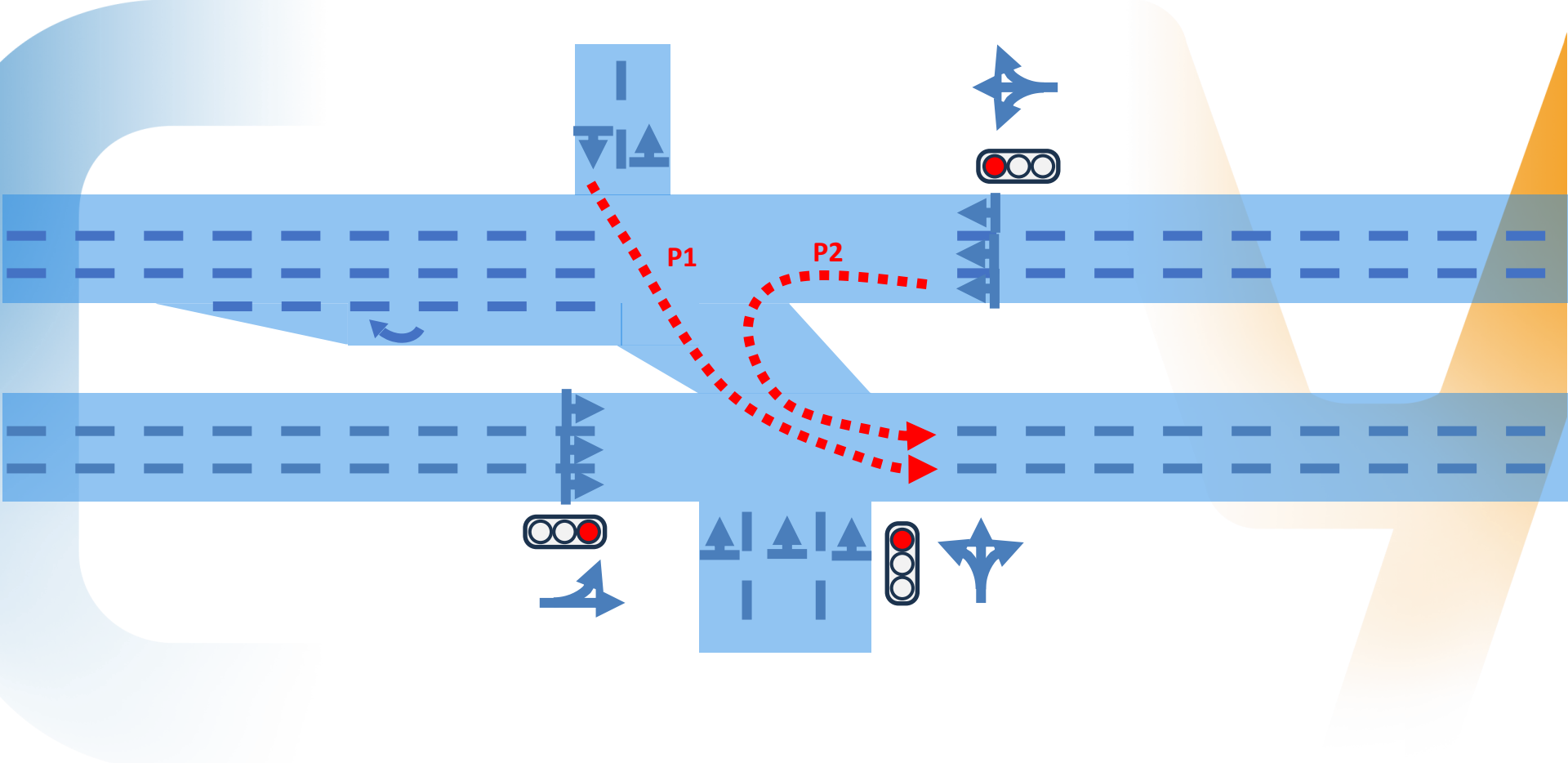


Activities

- Question (no modelling required):
 - Our system is unaware of what is happening in the real world. Ex: a traffic light is turning to green but not car in the lane is waiting to pass. How would you make it more efficient ? Where are you going to install/position extra equipment (please provide a picture) ?

Traffic Lights Controller 2





Activities

- Adapt the modelling from the previous example for this configuration:
 - Only 3 signals,
 - One road (north) is not controlled by signals
 - We do not consider pedestrian crossing
 - What scheduling do you propose for the signals ?
 - Is it possible to handle P1 (cars coming from the not controlled road to the southern road) ? How ?
 - How do you handle P2 ?

Activities

- It has been decided that the traffic lights control is going to be developed without formal methods.
- However, as it is safety related, we need to design a « supervisor » in charge of checking if « everything is going well ». Unfortunately our city does not have a lot of money and only computers with 3 digital (Boolean) inputs and 2 digital outputs are available. And only 1 computer for this configuration.
 - Is it possible to define a safety verification ? If yes, how the inputs are connected to the signals and what is the meaning of the outputs ?