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

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

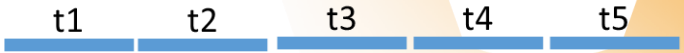
≡ Model the system, verify that you can (easily) express « properties », ensure that the model is provable

- 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

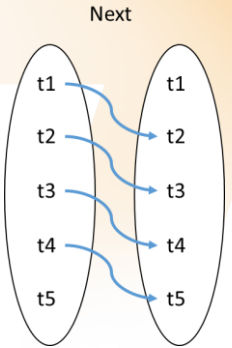
Let the set TrackCircuit = {t1, t2, t3, t4, t5}



Let the function Next $\in$ TrackCircuit $\nrightarrow$ TrackCircuit

Example: Next(t1) = t2, Next(t2) = t3, Next(t3) = t4, Next(t4) = t5

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



Let the function KpAbs : TrackCircuit $\rightarrow$ N

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

Objectives

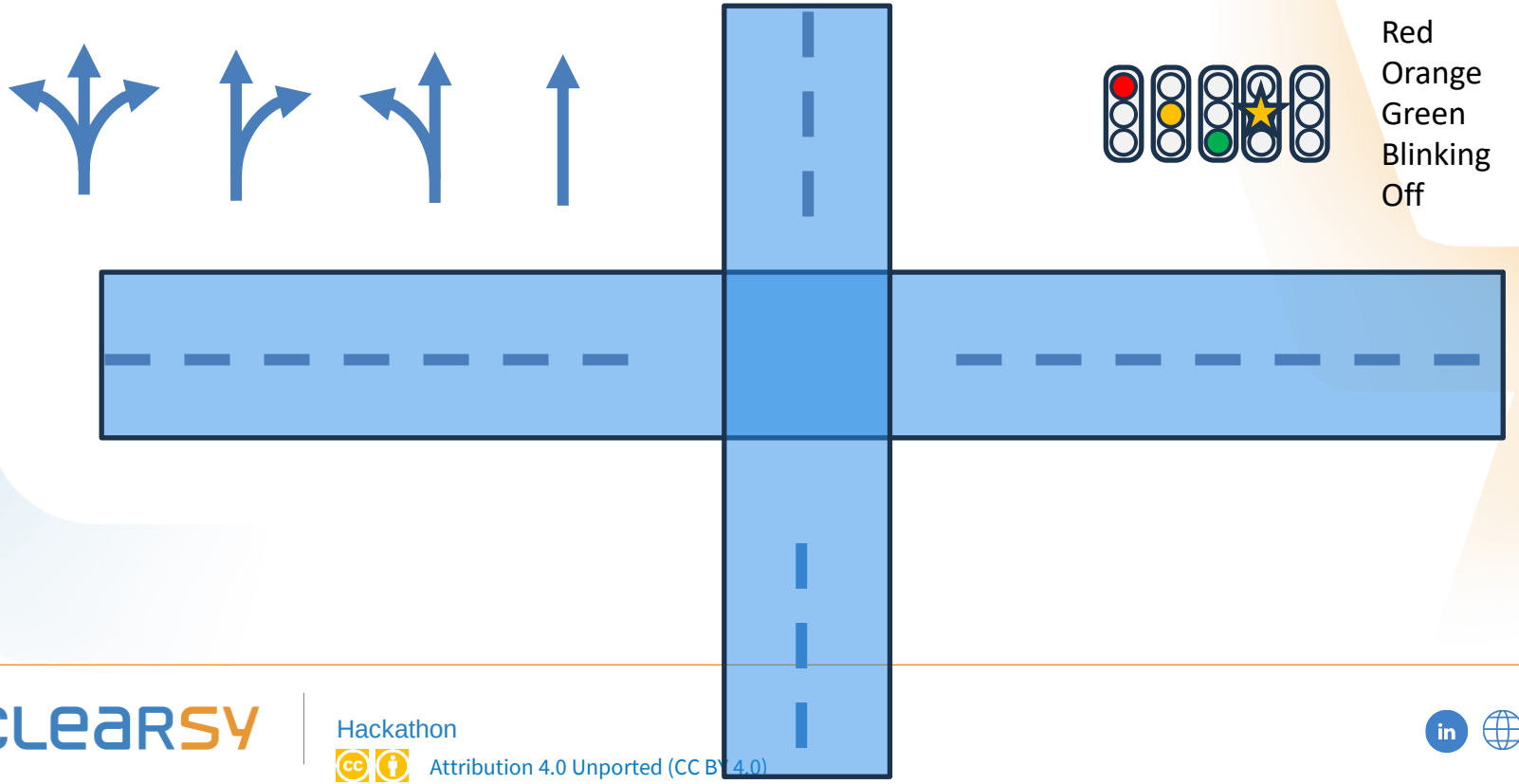
- Model a traffic light controller (1/2)
  - Animate the lights (a bit simplified),
  - no temporal behaviour,
  - Only specification (no implementation)
- CTX machine given
- M0 machine to complete
  - some invariant, most operations
  - Check with proof and ProB
- Questions to answer

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

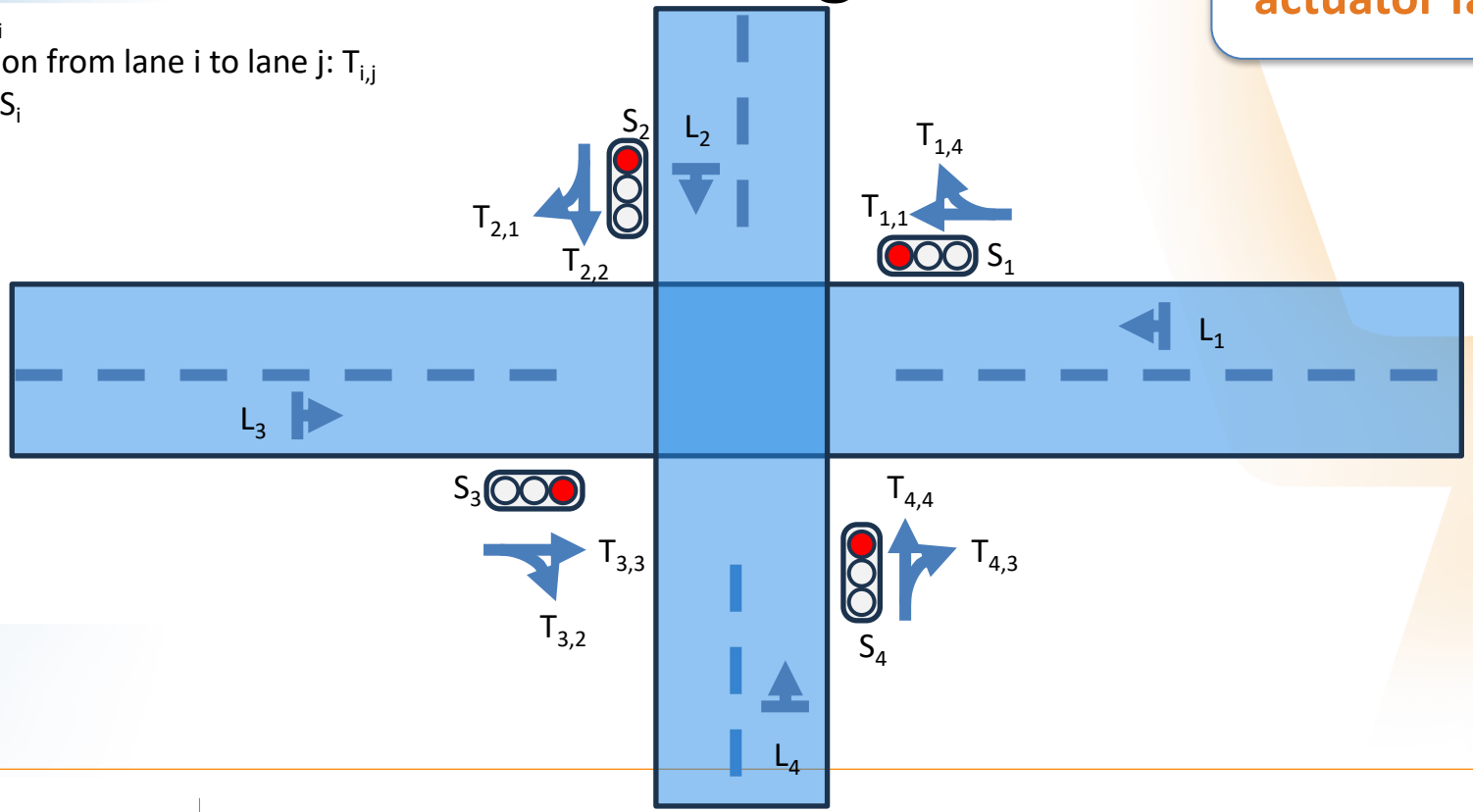
Traffic Lights Controller 1

Simple configuration



Naming

Legend:
Lane: L_i
Transition from lane i to lane j: $T_{i,j}$
Signal: S_i

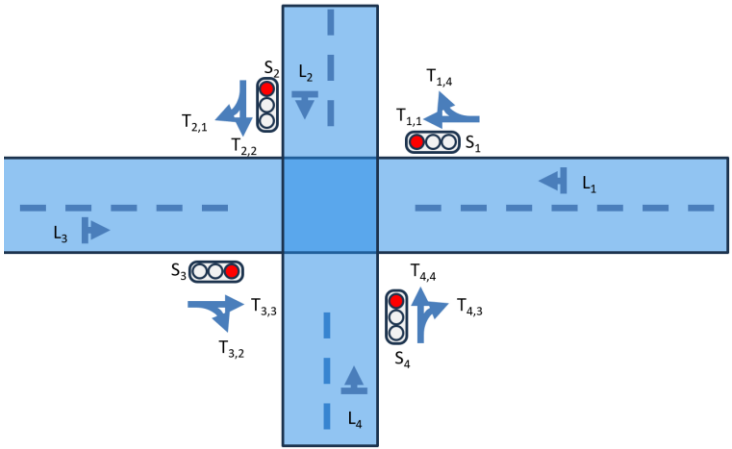


No sensor or actuator failure

Definitions, Constraints & Properties

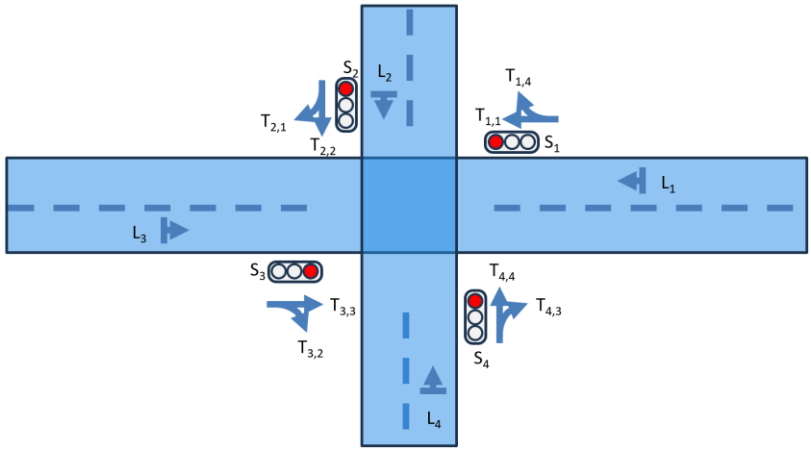
```
SETS
    LANES = {L1, L2, L3, L4};
    SIGNALS = {S1, S2, S3, S4}
CONSTANTS
    INTERSECT_LANES,
    TRANSITIONS

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



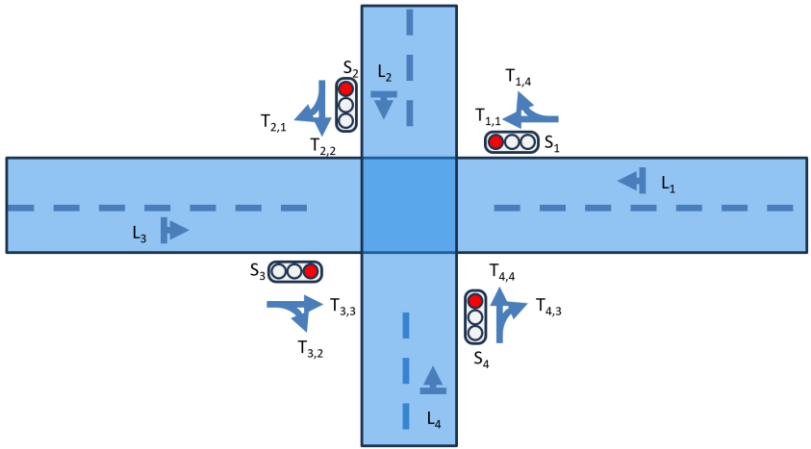
Definitions, Constraints & Properties

```
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

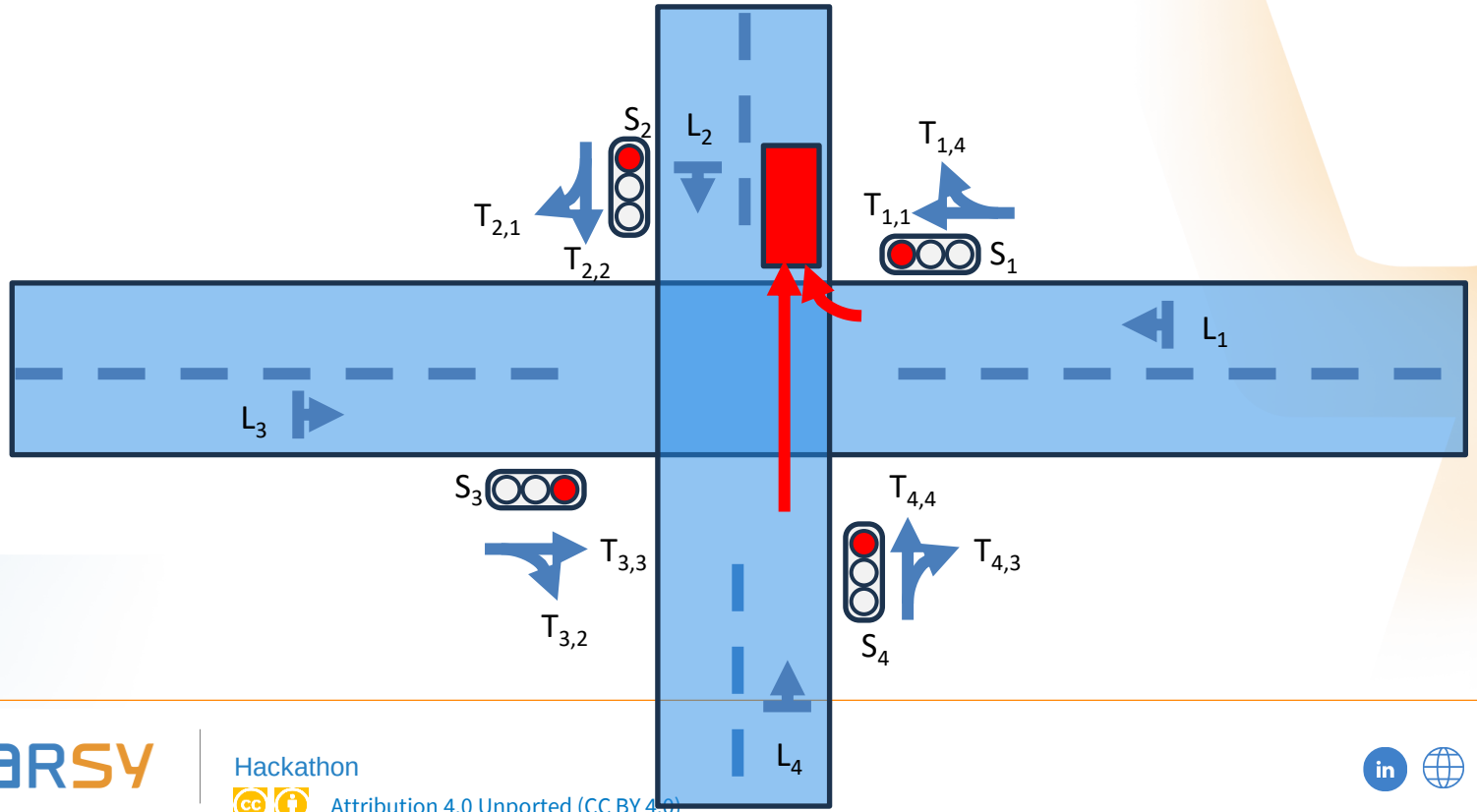
```
VARIABLES
    rs, /* red signals */
    os, /* orange signals */
    gs, /* green signals */
    bs /* blinking (orange signals) */
INVARIANT
    rs <: SIGNALS &
    os <: SIGNALS &
    gs <: SIGNALS &
    bs <: SIGNALS &
    (bs = SIGNALS or bs ={}) & /* all blinking or none blinking */
    rs \/ os \/ gs \/ bs = SIGNALS & /* coherency */
    rs /\ os /\ gs /\ bs = {}  /* coherency */
```

Definitions, Constraints & Properties

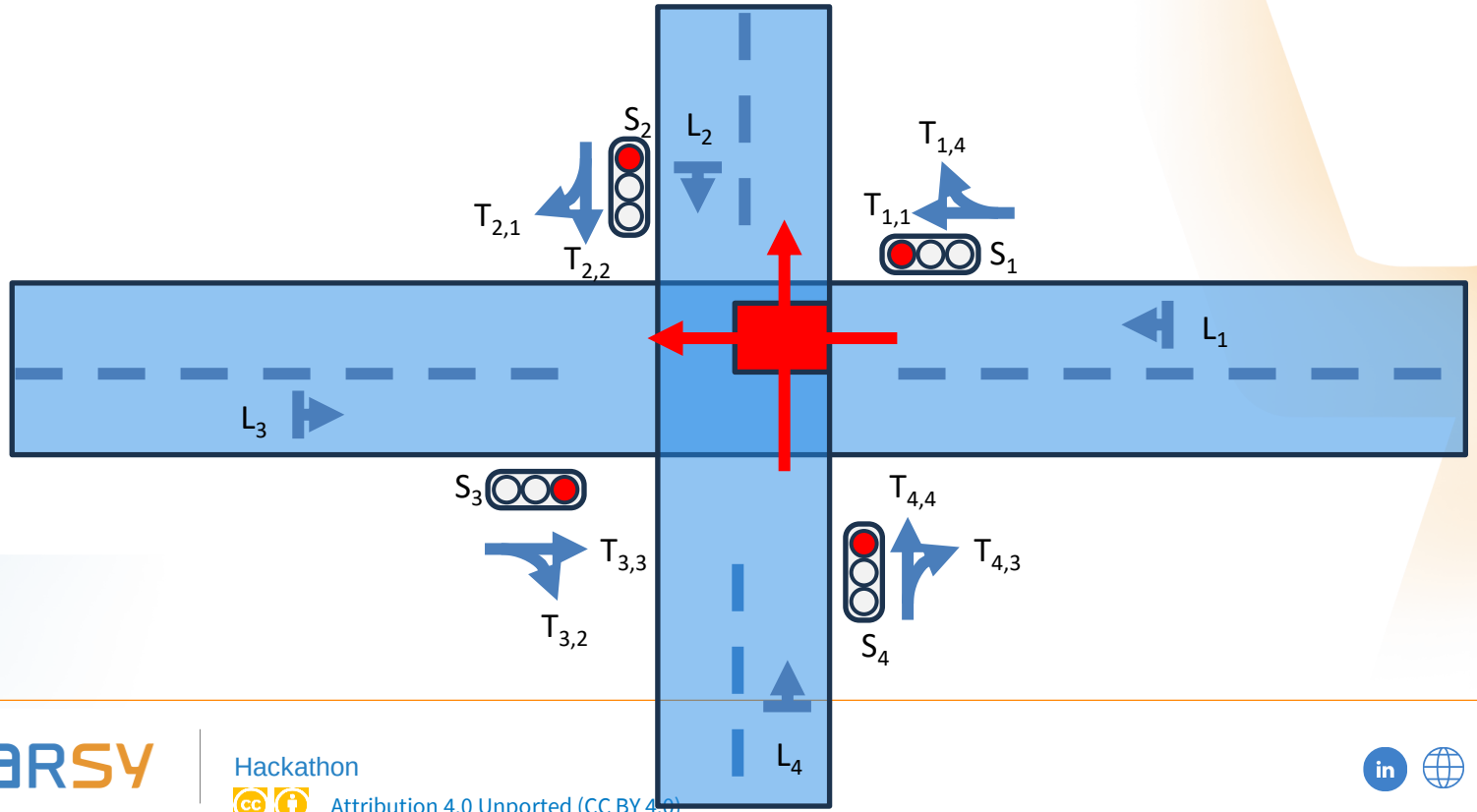
```
INITIALISATION
    rs := {} ||
    os := {} ||
    gs := {} ||
    bs := SIGNALS

OPERATIONS
    reset_turn_on = /* no precondition, all signals on orange blinking */
    BEGIN
        rs := {} ||
        os := {} ||
        gs := {} ||
        bs := SIGNALS
    END;
```

Collision #1



Collision #2



Activities

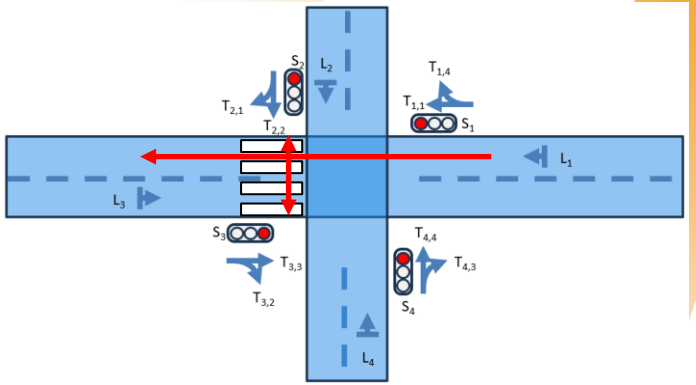
- Add invariant to avoid collision#1 and #2
- Specify the operations
  - `start_exploitation`: moving from all blinking to all orange
  - `orange_to_red`: moving from all orange to all red
  - `red_to_green1`: set S1 and S3 to green
  - `red_to_green2`: set S2 and S4 to green
  - `green_to_orange`: set 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.

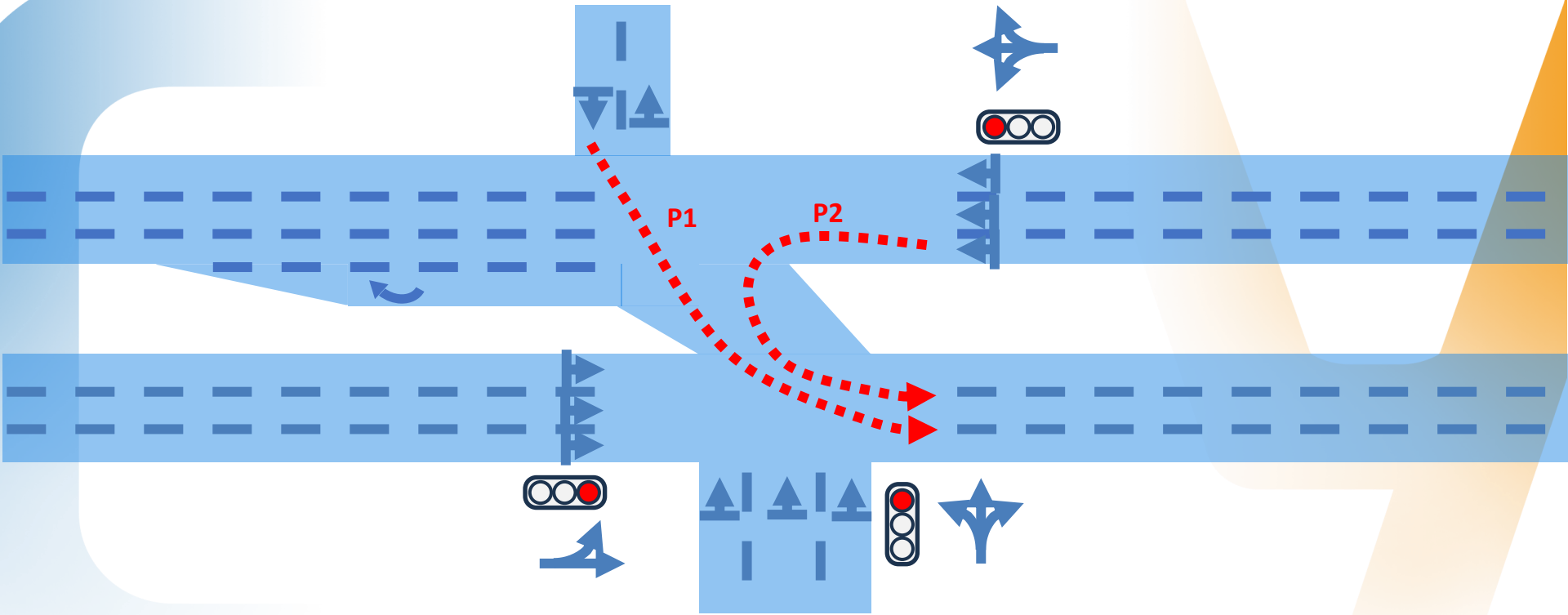


Activities

- Question (no modelling required):
  - Our system, including pedestrian crossings, 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 equipement (please provide a picture) ?

Traffic Lights Controller 2


Condomínio Duo
Capim Macio
Rua Jornalista Alexis Gurgel
Cirne Pneus -
Roberto Freire
Magasin de pneus
Drogaria Globo -
Roberto Freire I
Silva e Macedo
Assessoria Condominial
Crine Pneus
Magasin de pneus



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 trafic 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: the CLEARSY Safety Platform. 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 ?

Activities

- So 3 traffic lights represent 3x3 bulbs (green, orange, and red). It is not possible to track the 9 of them with only 3 inputs. Considering the main safety property of traffic lights (that is related to green lights), we are going to supervise these 3 bulbs.
- Define a supervision function to monitor in realtime their state.
  - One output has to be used to report an abnormal safety-related behaviour. Abnormal safety behaviour detected is kept on while the board is on.
  - The other output has to be used to report abnormal non-safety related behaviour (what if no green bulb is never turned on, if they switch too frequently). Abnormal non-safety behaviour is kept transient.

Activities

- Specify and implement the supervision function.
  - The combinatorial behaviour is defined in the specification
  - The temporal behaviour has to be defined in the implementation.
- Program delays to detect abnormal behaviour (switch more frequent than every 5s)
- Test your function in emulation mode.
- Embed your function on the board (using CSSP runner), and test it against scenarios (Arduino programmed to stimulate inputs)