Traffic Light Control
INFO-F-410 – Embedded Systems Design

Jamal BEN AZOUZE, Marien BOURGUIGNON, Nicolas DE GROOTE,
Simon PICARD, Arnaud ROSETTE, Gabriel EKANGA

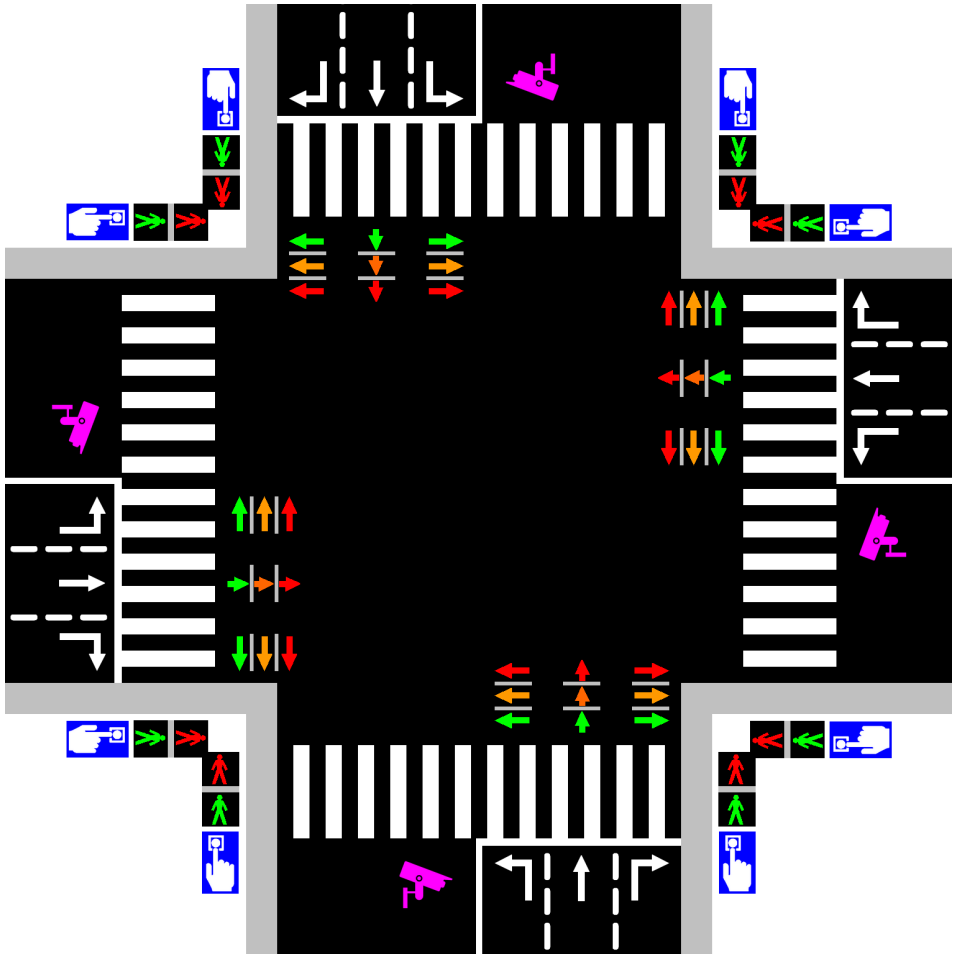
Université Libre de Bruxelles

Département d'Informatique

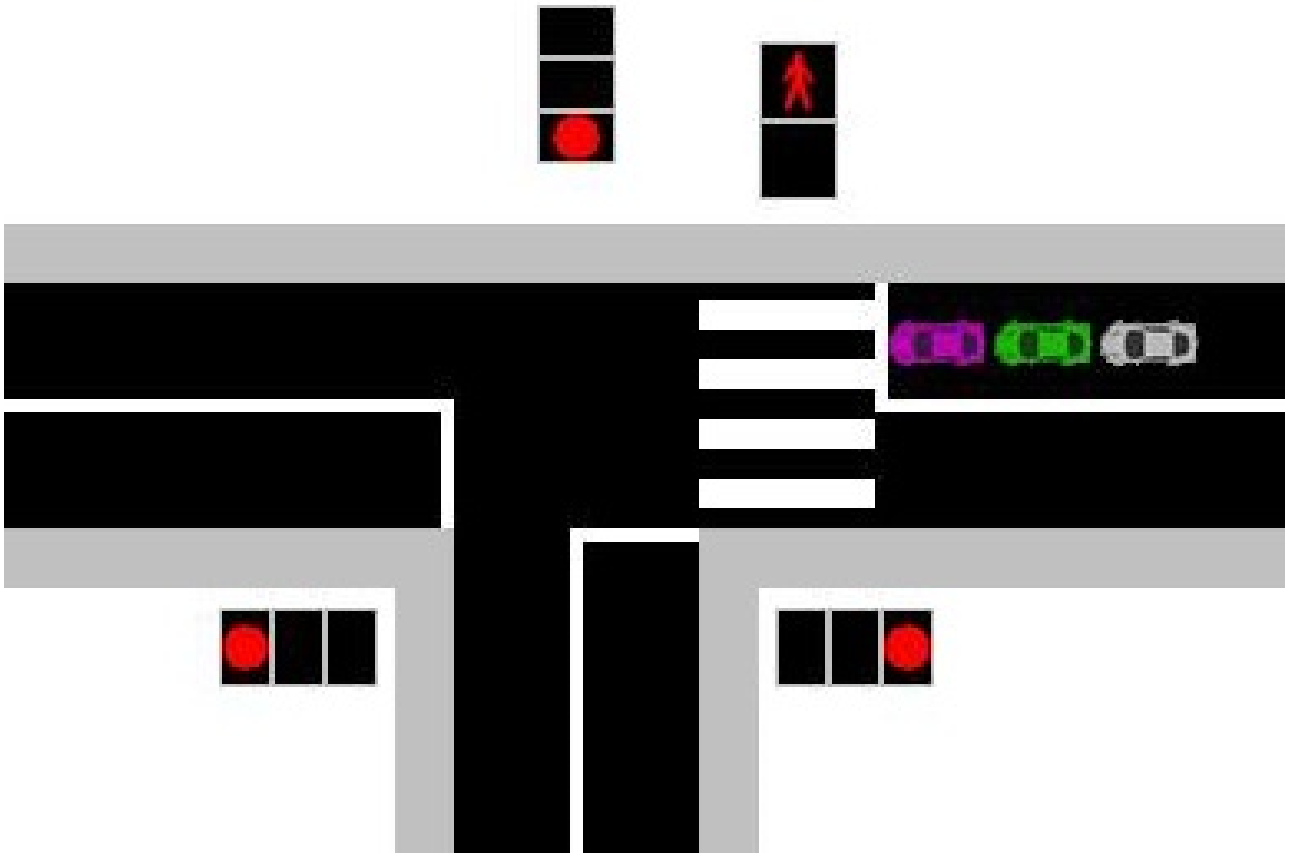
4 Juin 2015

Sommaire

First idea



Refined idea

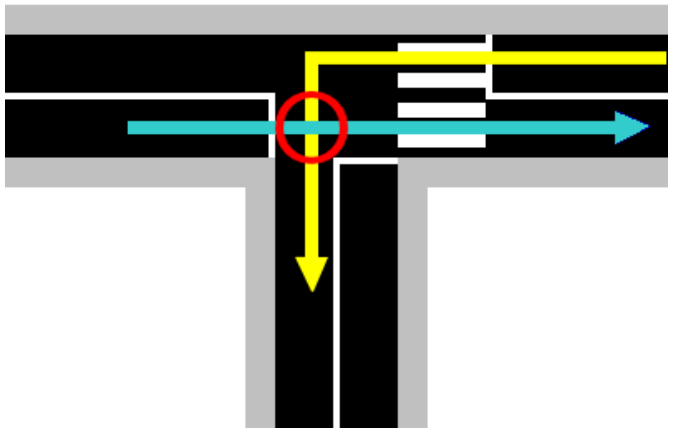


Assumptions

- The traffic lights do not have orange lights -> cars stop instantaneously
- Every entity respect the highway code
- At any time a car can arrive and join a queue of cars
- At any time a pedestrian can push the button to cross the street
- A pedestrian will always cross the street in less than a fixed time
- A car will always cross the crossroads in less than a fixed time

Embedded System

- Lives are at stake -> Critical system
- Impossible to test manually every combination -> Use of winning strategies



Actors

1. Pedestrians

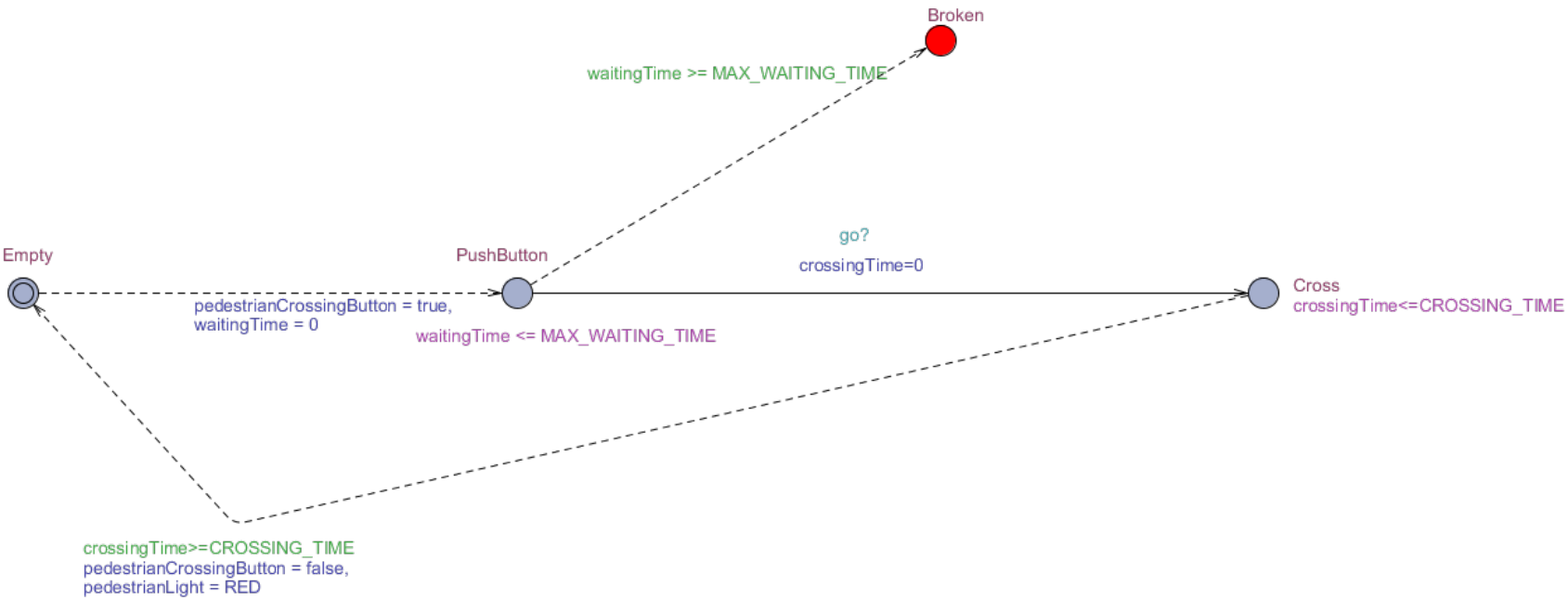
Actors

1. Pedestrians
2. Cars

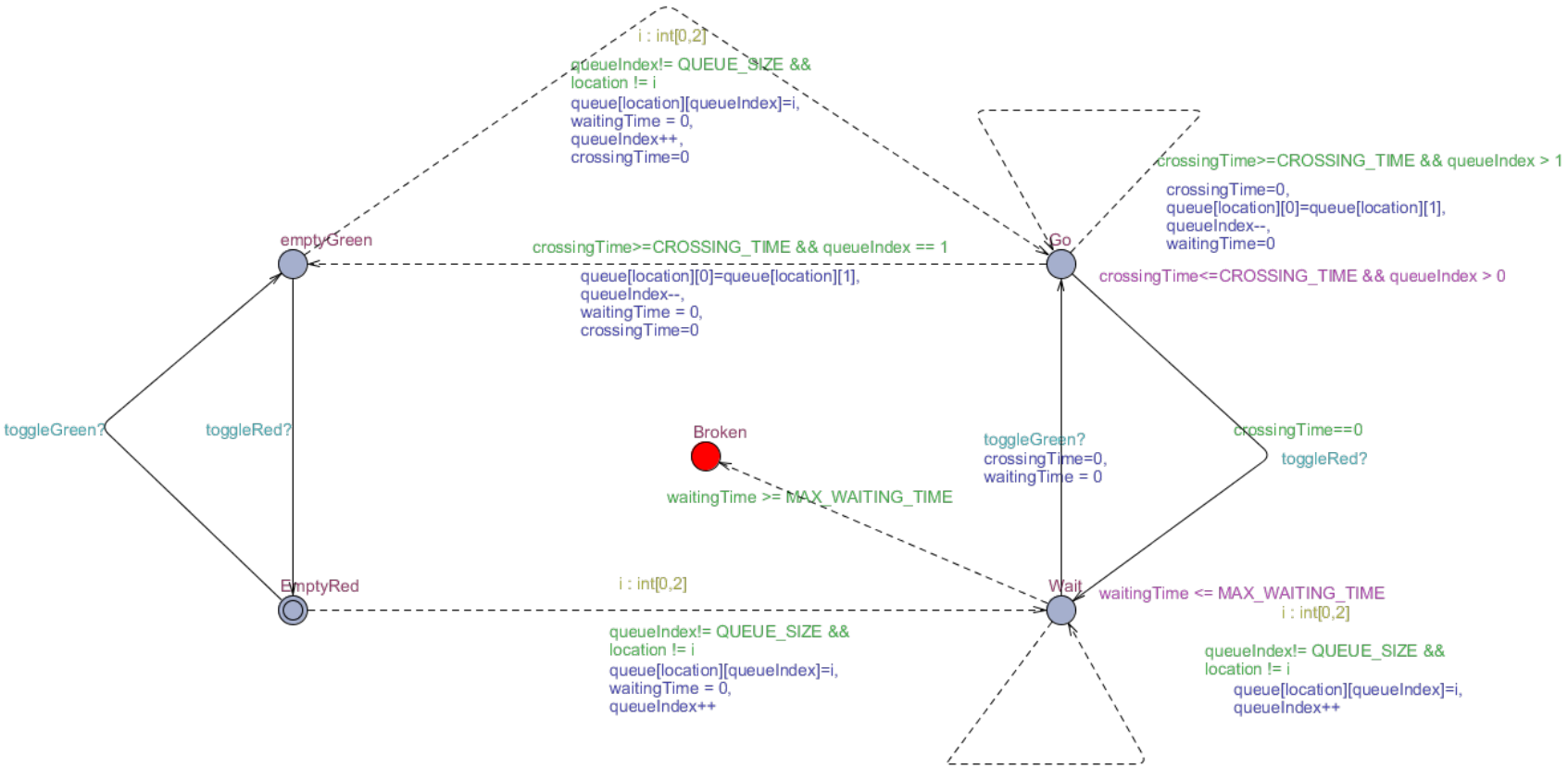
Actors

1. Pedestrians
2. Cars
3. Traffic lights

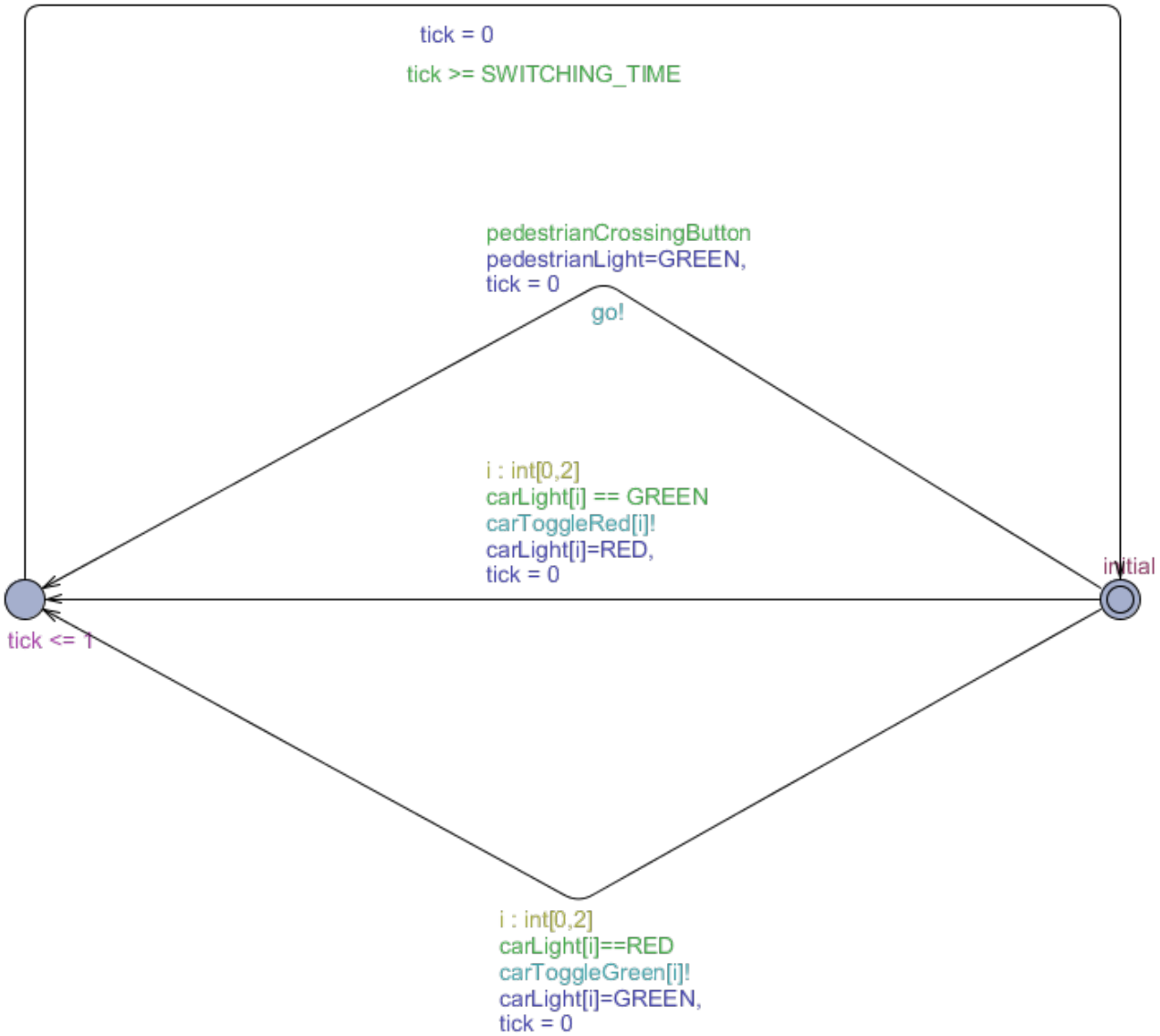
Pedestrian generator automaton



Car generator automaton



Dummy controller



Winning condition

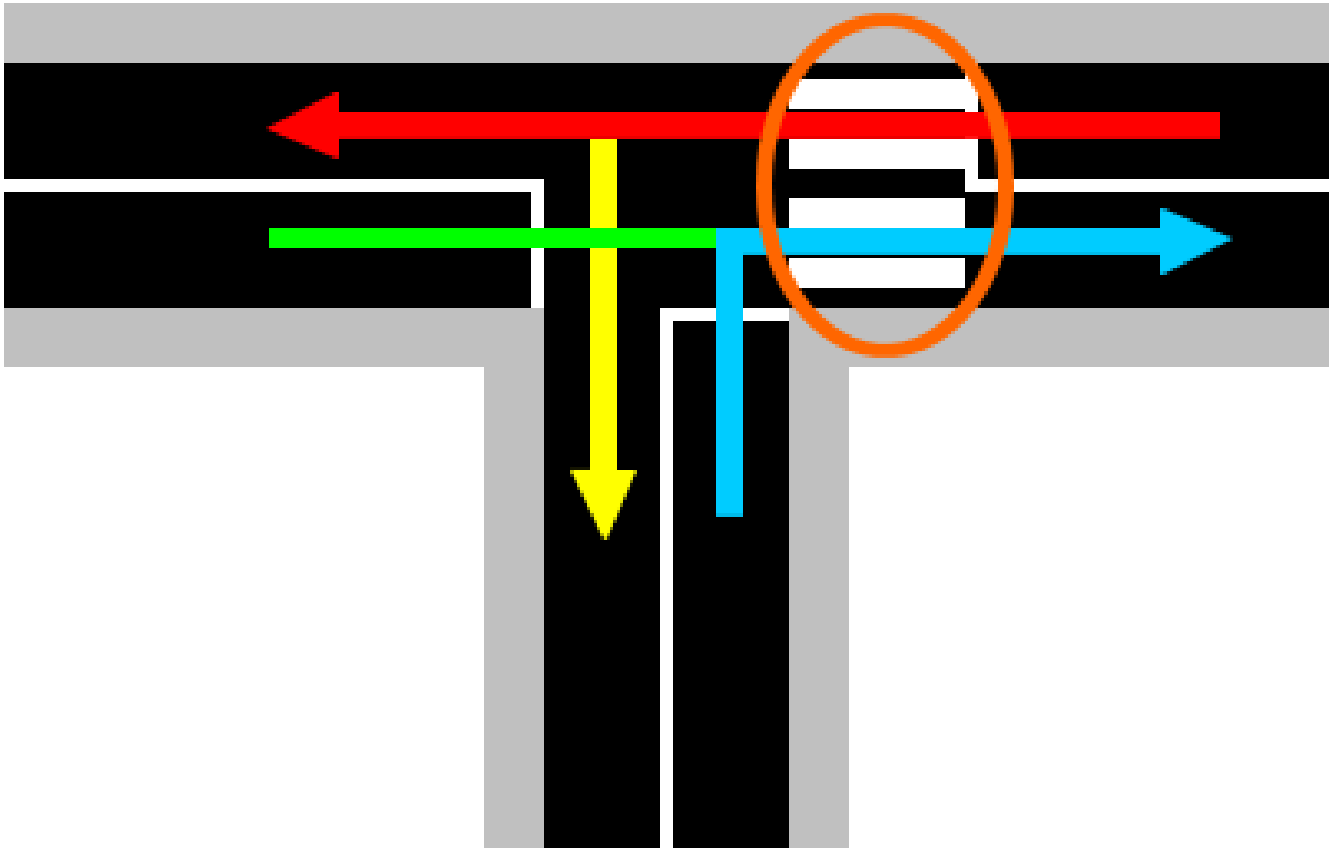
Pure safety : the controler **must** avoid the losing states.

```
control : A[] not(lose)
```

1. A pedestrian is never knocked down

```
form_1 =
control: A[] not(PedestrianGeneratorEast.Cross &&
(
    CarGeneratorEast.Go ||
    (
        CarGeneratorWest.Go && queue[W][0] == U
    ) || (
        CarGeneratorSouth.Go && queue[S][0] == R
    )))
```

Winning condition



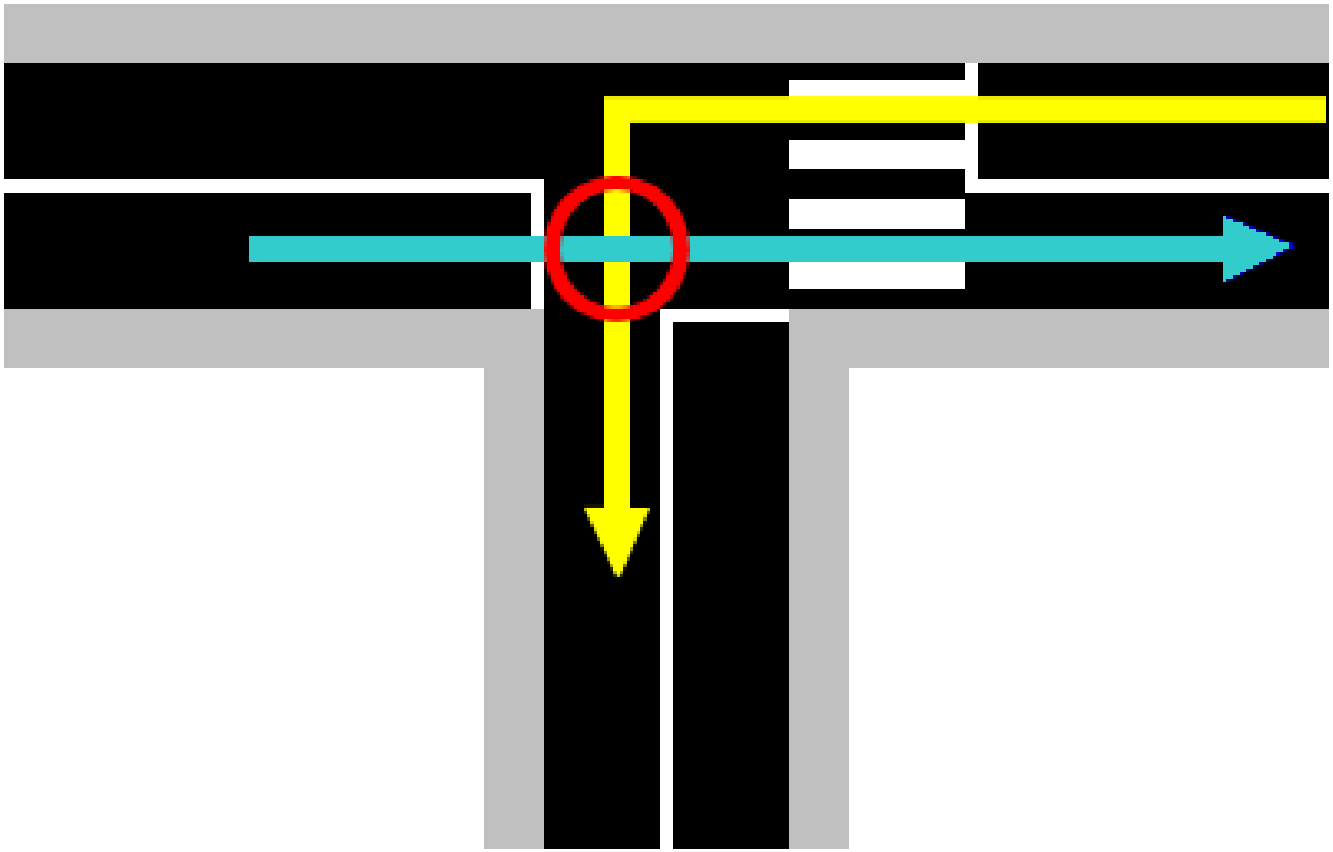
Winning condition

2. Cars should never collide

```
form_2 =
Control: A[] not
(
    CarGeneratorWest.Go && queue[W][0] == U &&
    ((
        CarGeneratorEast.Go && queue[E][0] == L
    ) || (
        CarGeneratorSouth.Go &&
        ( queue[S][0] == L || queue[S][0] == R )
    ))
) || (
    CarGeneratorWest.Go && queue[W][0] == R
    && CarGeneratorEast.Go && queue[E][0] == L
)
```

Same idea for the two other orientations.

Winning condition



Winning condition

3. Pedestrians do not wait infinitely

```
form_3 =
control: A[] not (PedestrianGeneratorEast.Broken)
```

4. Cars do not wait infinitely

```
form_4 =
control: A[] not (CarGeneratorEast.Broken ||
                  CarGeneratorSouth.Broken ||
                  CarGeneratorWest.Broken)
```

Full condition

```
control: A[] not (
    form_1 || form_2 || form_3 ||form_4
)
```

Simulator

Base idea

- Illustrate controller-environment interactions
- Environment behaves randomly
- Controller avoids the bad states by implementing a winning strategy

Problems

- The winning strategy is huge !
- It specifies a transition of the controller from each system state.
- Generate an execution trace from Uppaal tiga does not work (software problem)

Simulator

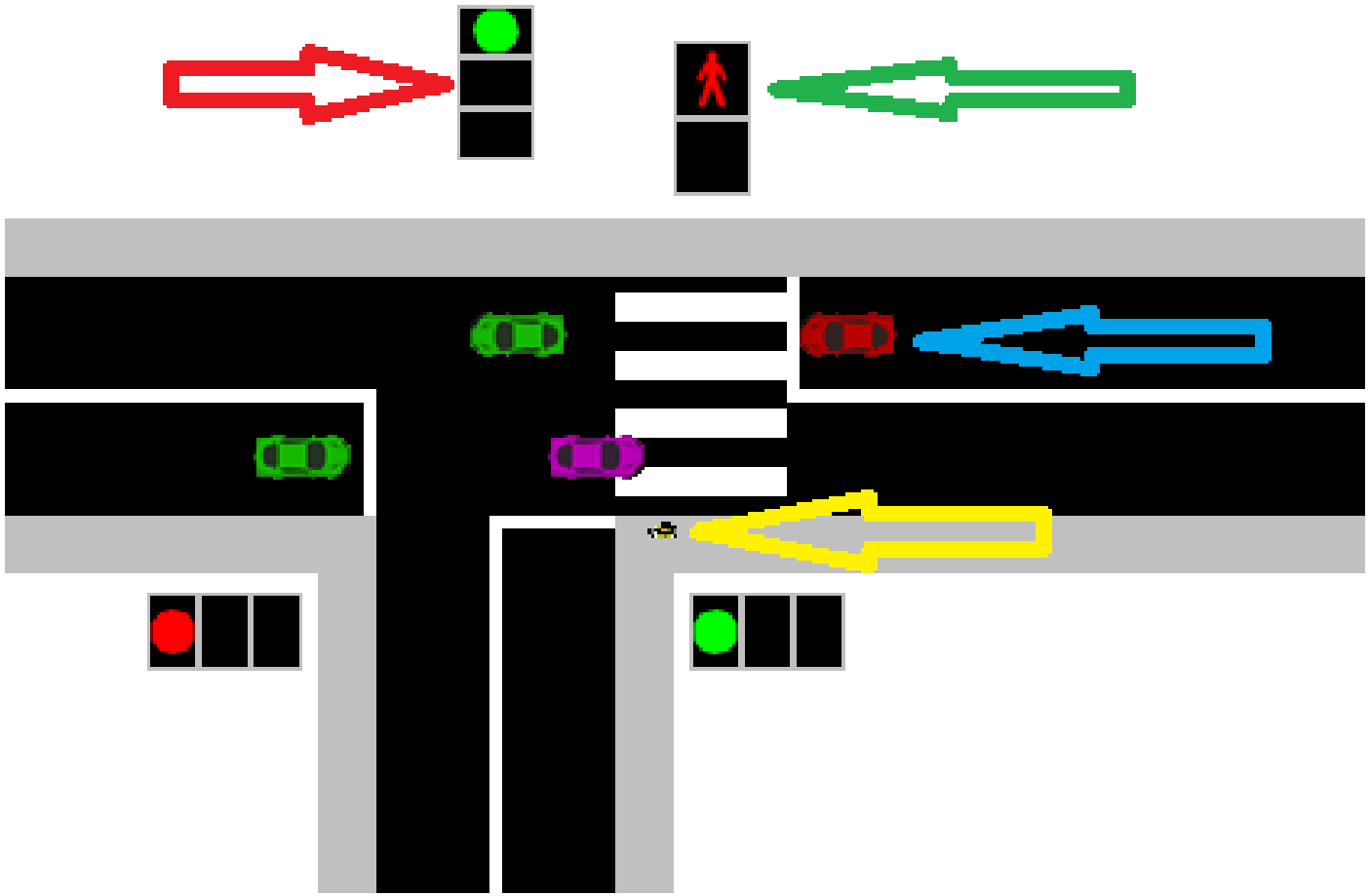
Solution

- Define a controller following a winning strategy in Uppaal (not tiga)
- Then generate a trace of the whole system.
- Finally, convert this trace via the Libutap library (readable)

GUI

- Cars (bleu arrow), car lights (red arrow)
- Pedestrians (yellow arrow), pedestrian light (green arrow)

Simulator



Conclusion

- The difficulty to model a system
- The necessity to simplify the model to generate a winning strategy
- The importance of choosing a good tool for our problem
- The necessity to identify the different properties to satisfy
- The necessity to identify the different actors which intervene in our problem
- The difficulty to generate a winning strategy
- The limit of the tools and how to extract a trace
- This project allowed us to better understand the difficulty and the constraints which weigh on the modeling of a system
- Little changes can increase a lot the number of states
- The necessity to do compromises