Using Petri nets within Cyber-Physical System's development

IOPT-Tools - Web-based platform for embedded controllers development based on IOPT Petri nets

Anikó Costa – akc@fct.unl.pt

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

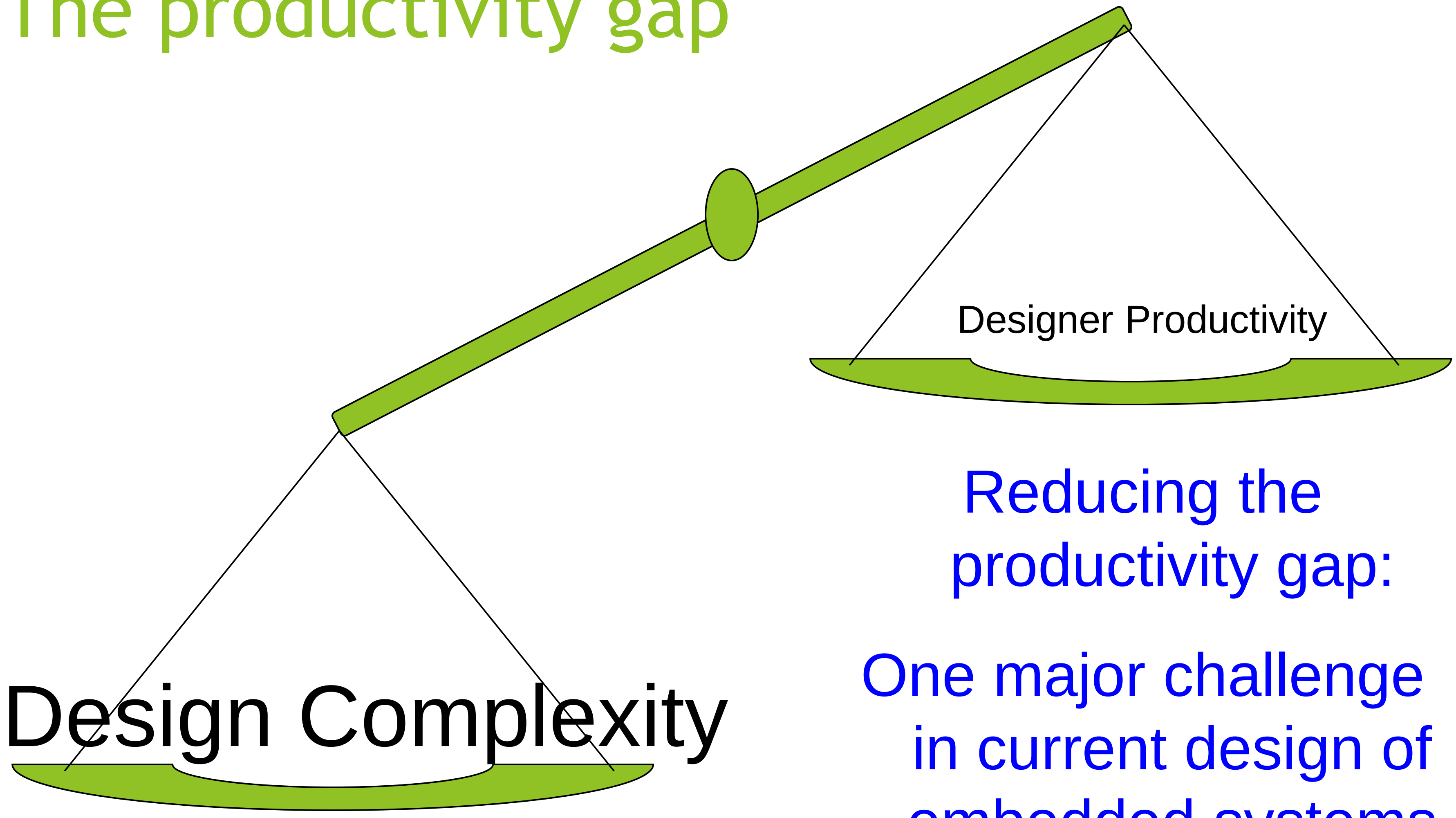
CTS - Centre of Technology and Systems

GRES R&D Group on Reconfigurable and Embedded Systems

Outline

- Motivation to move towards model-based development
- Petri nets - a brief overview
- IOPT-Tools framework
- Conclusions

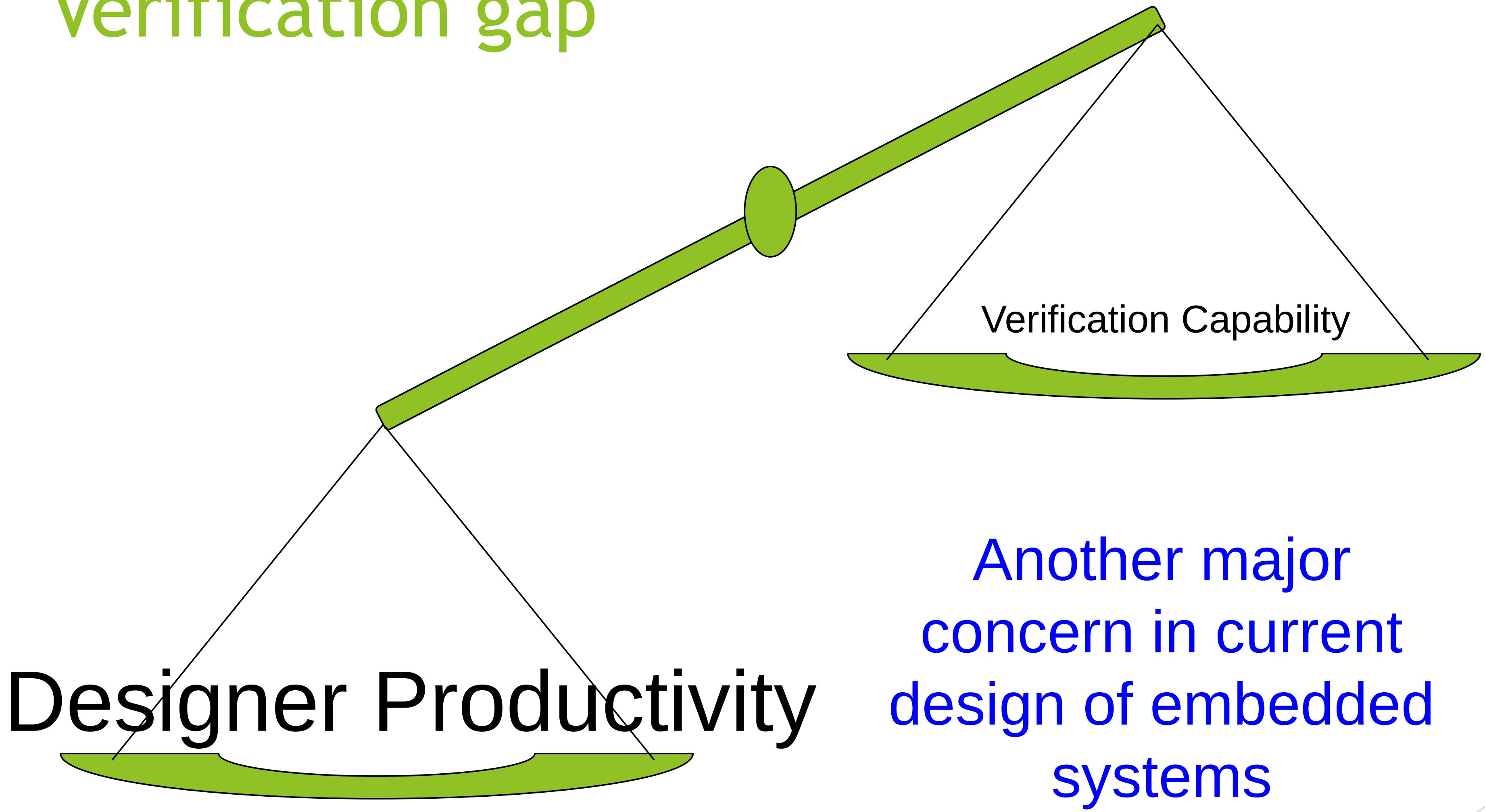
The productivity gap



Reducing the productivity gap:

One major challenge in current design of embedded systems

Verification gap



Another major concern in current design of embedded systems

The performance gap

- More performance always needed (at least wanted)
- Increasing clock frequency is not enough

- Exploiting concurrency and distributed computing and control is one major option to support improvements

- Although, we need mechanisms to allow robust synchronization, sharing of resources, mutual exclusion, and so on …

Open issues and challenges

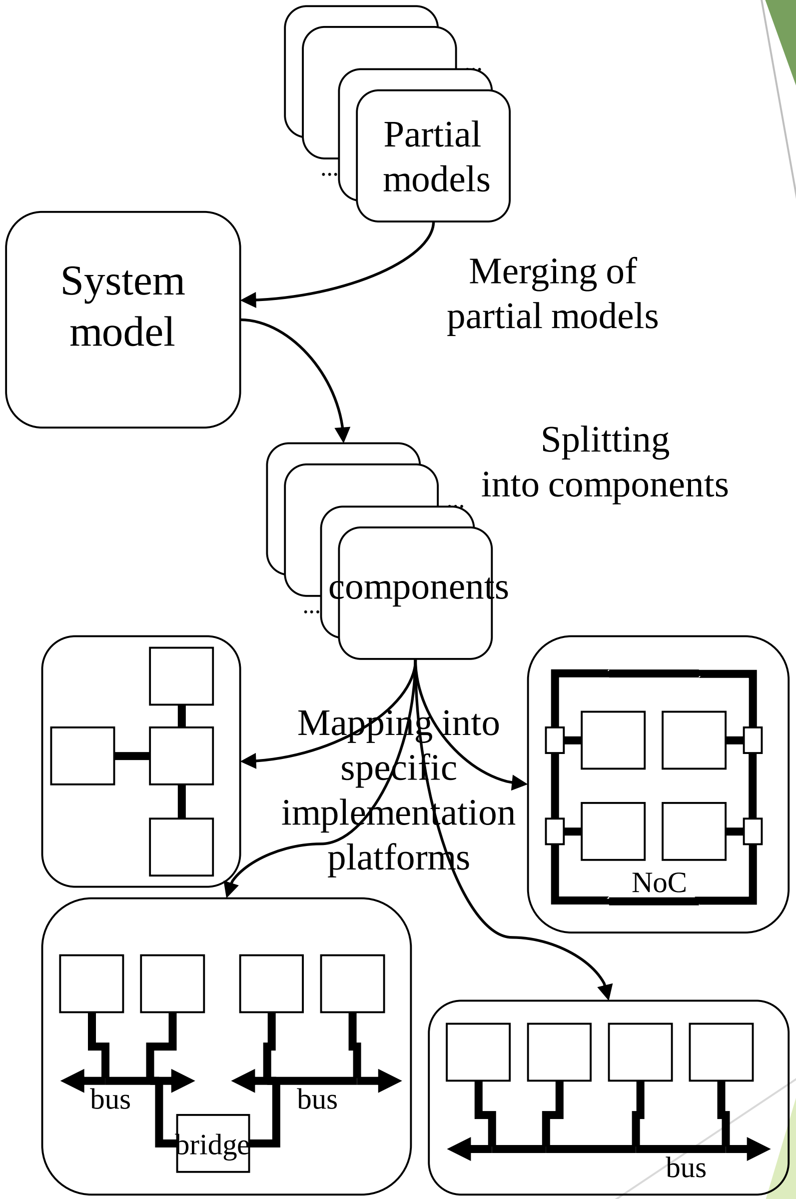
- How to handle design complexity?
- How to reduce the productivity gap?
- How to reduce the verification gap?
- How to cope with the performance gap?
- How to support reliable distributed execution?
- Contribution to the answers:
  - Relying more and more on Model-based Development
  - Increasing usage of design automation tools (including specification, simulation/validation, verification, code generation, and test)

Moving to model-based development

- Models are used not only for describing specifications of the system at earlier phase of development, but also intended to be used along the whole development process, including automatic code generation (verification and implementation)
- Start with platform independent specification, “easily” supporting porting/implementation into specific platforms.
- For that end, an operational model having a precise execution semantics needs to be selected, allowing usage of the model at the different stages od the development process.

Underlying development methodology

- Starting from partial models
- System model by merging partial models
- System components using model splitting
- Mapping into specific implementations platforms



Selection of modeling formalism

- Among those eligible most common formalisms, it is worth to mention state diagrams, hierarchical and concurrent state diagrams, statecharts, and Petri nets.
- All of them can have:
  - Rigorous computational model
  - Precise execution semantics
  - Graphical representation
  - Formal representation

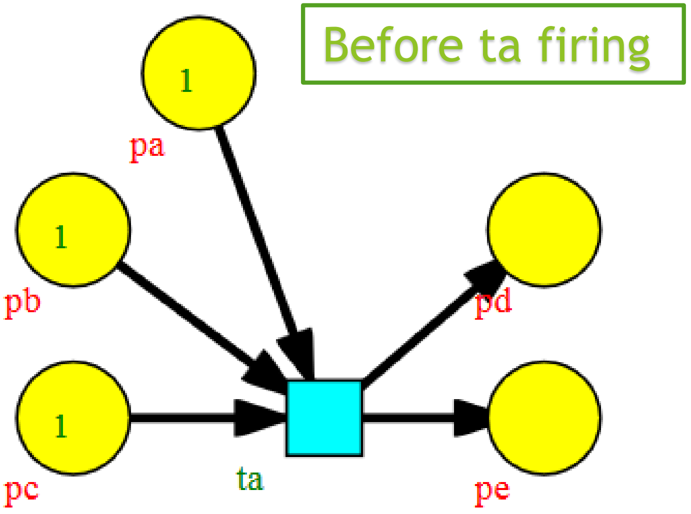
Petri nets : what ? (I)

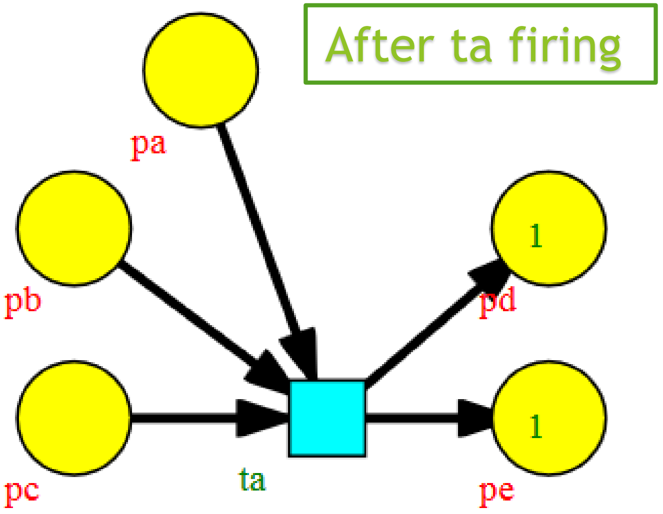
- Petri nets allow the modelling of system’s behavior, starting form the concept of event and condition (close to the state concept).
- A first characterization can seen Petri nets as a generalization of state diagrams.
- Graphical formalism, allowing an easy understanding of system’s behavior (strong point for designers), with formal representation capabilities (strong point for tool developers).

Petri nets: what ? (II)

- Bipartite graph, composed by two types of nodes:
  - Conditions or places, represented as circles or eclipses;
  - Events or transitions, represented by bars, squares or rectangles;
- Directed arcs that can interconnect nodes of different types;
- Model dynamics is associated with transitions, while the places represents the static component.

Firing rules



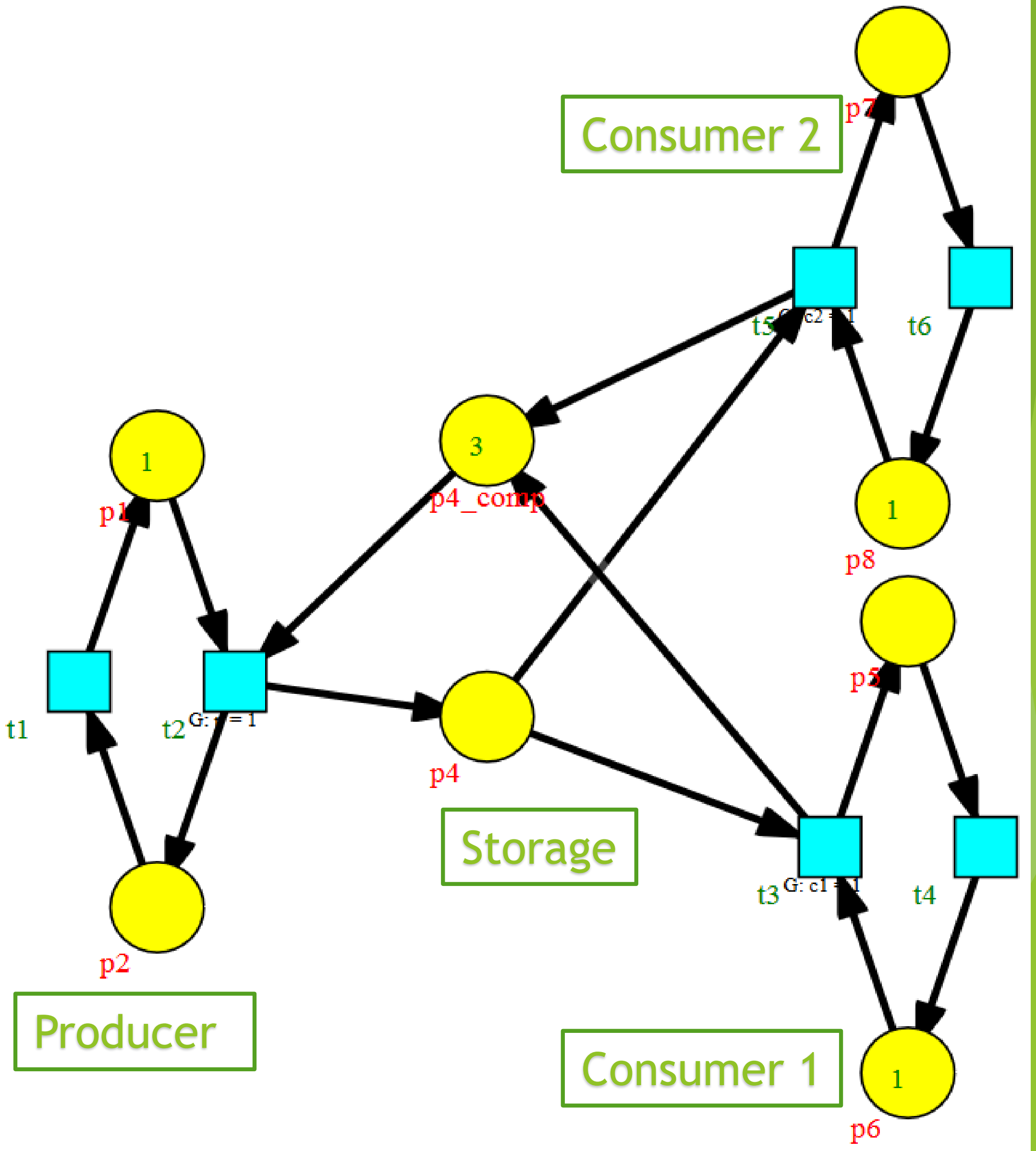


- Enabled transitions can be fired
- To be enabled, necessary to comply with enabling pre-conditions (input places marked).
- For some types of Petri nets, also necessary to comply with post-conditions (output places unmarked).
- Destruction of input tokens and creation of output tokens -> Atomic action

Example:
Producer-consumer system

Typical modeling situations:

- Concurrency
- Local evaluation
- Synchronization
- Conflict



Petri net classes

- Low-level Petri nets versus High-level Petri nets
  - Place-Transition nets vs Coloured Petri nets
  - Safe, bound vs unbound nets
- Autonomous Petri net vs Non-autonomous Petri nets

Operational semantics / implementation issues:

- Synchronous execution
- Asynchronous execution
- Globally asynchronous locally synchronous (GALS)

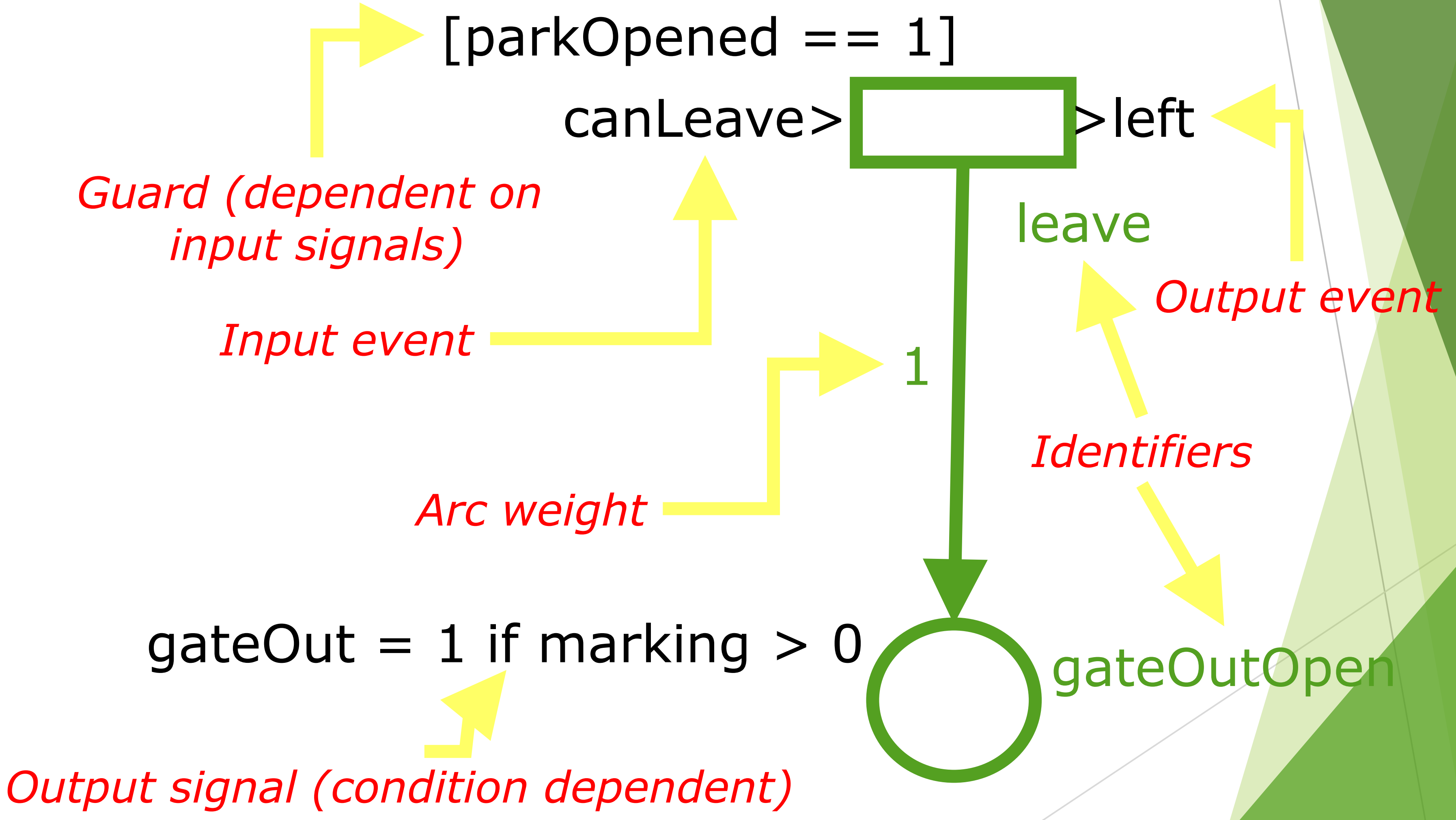
Petri nets for controller modeling

- Starting with autonomous classes of Petri nets…
- Extremely important to have the possibility to add dependencies to the environment under control, namely input and output signals and events.
- In those cases, Petri nets classes become non-autonomous.
- Several classes of non-autonomous Petri nets have been referred in the literature (some having strong links with automation systems ex. Silva 1985, Frey & Wagner 2000)

The Input-Output Place-Transition Petri net class (IOPT nets)

- Extended from the Place/Transition net class with non-autonomous dependencies:
  - Input and output signals, Input and output events
  - Transition firing conditioned by input events and guard function constrained by input signals
  - Transition firing can generate output event and/or update output signals
  - Output signals can also be associated with places
  - Introduction of time domains and communication channels
  - Includes transition priorities and Test arcs

IOPT – Input-Output Place-Transition nets (Syntax)

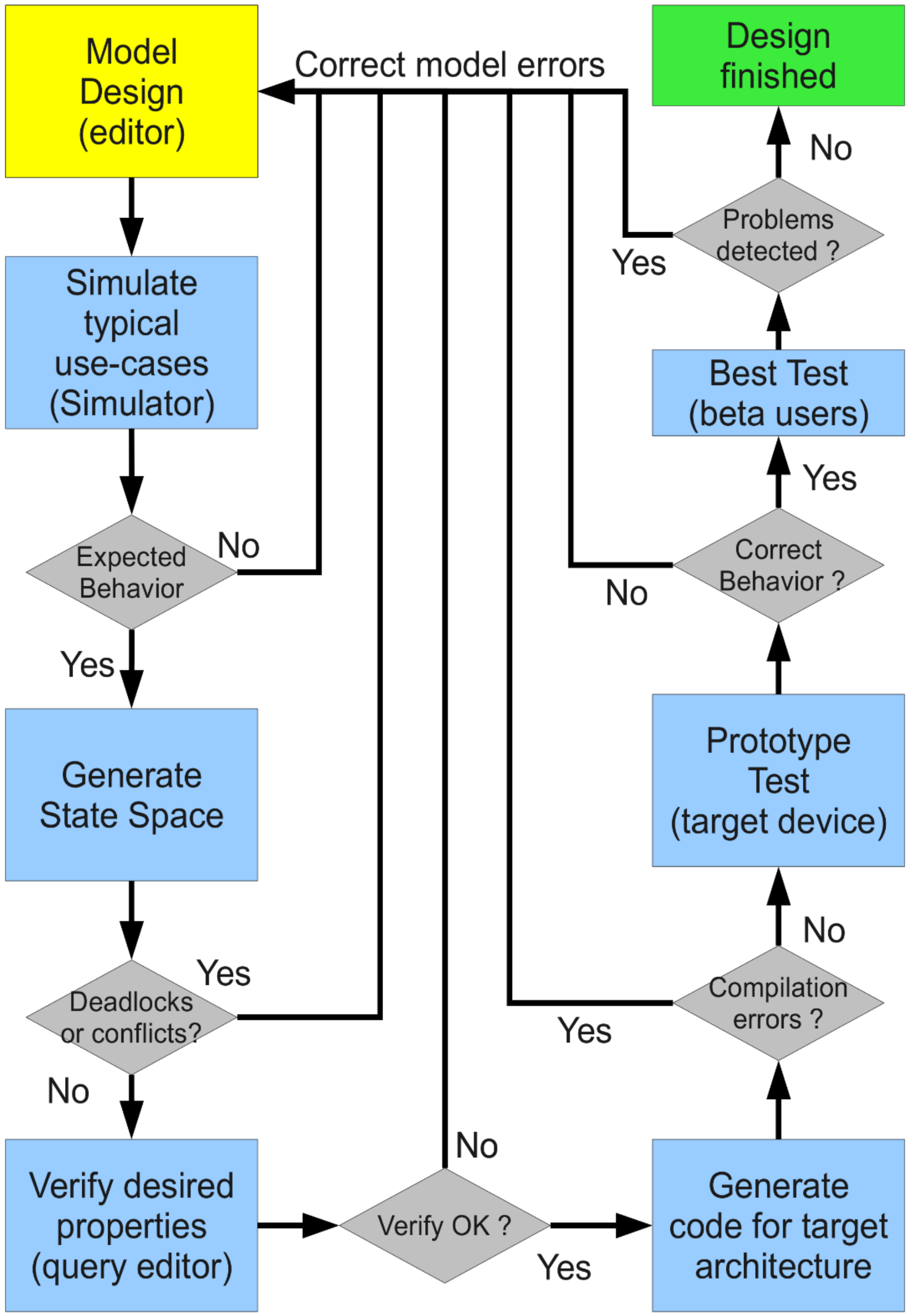


The IOPT-Tools – a cloud-based framework

- Petri nets already have a set of supporting tools mostly covering specification and verification.
- However, Petri nets need additional tools, mostly covering automatic code generation, to be fully integrated in engineering development flows.
- A contribution using IOPT nets is available at **http://gres.uninova.pt/IOPT-Tools**
- IOPT-Tools is an integrated development tool framework covering the whole phase of embedded controllers development (including automatic code generation),testing (including simulation and verification) and maintenance
- IOPT-Tools have been extensively validated within engineering courses at NOVA University of Lisbon (and used by others)

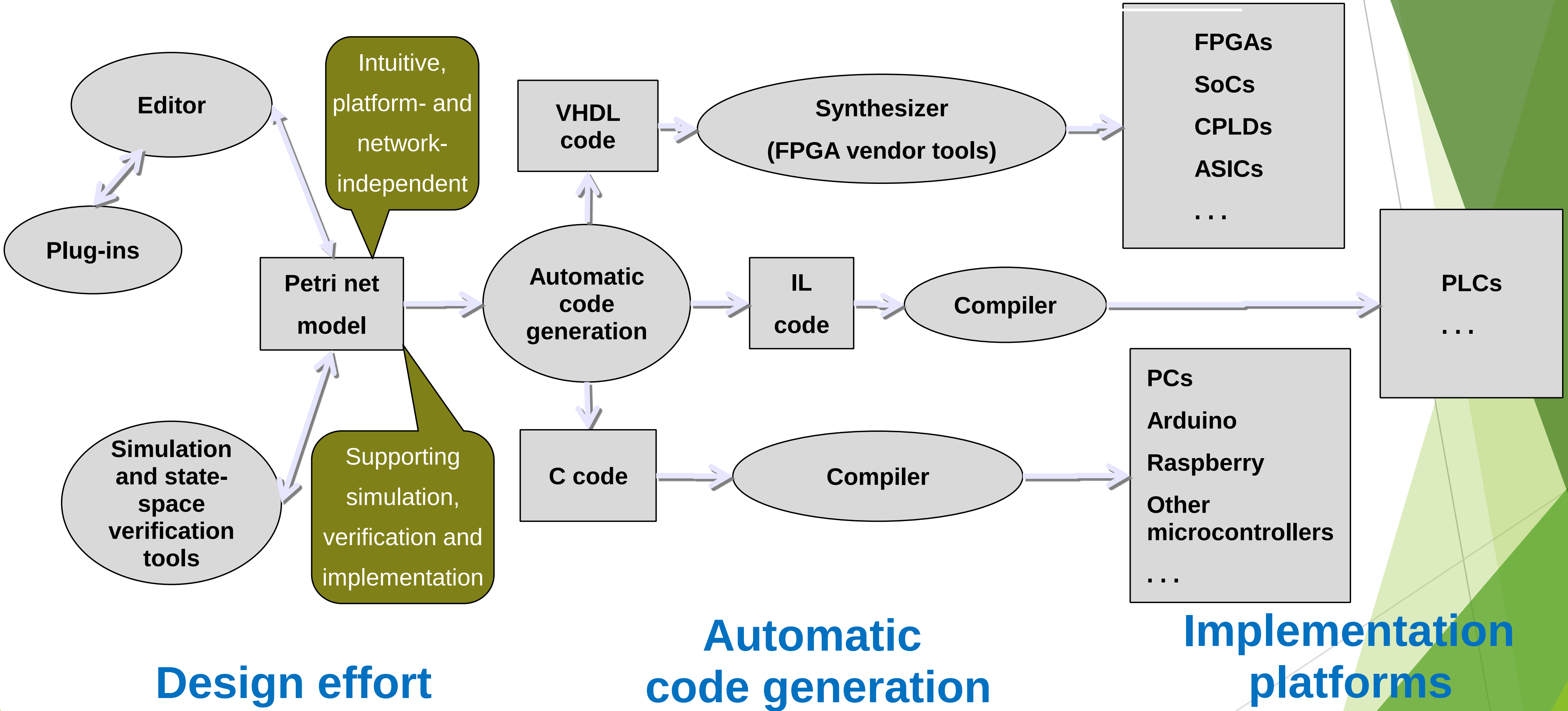
IOPT-Tools cloud-based framework

- Tools are offered under a cloud-based user interface
- Web User Interface (http://gres.uninova.pt/IOPT-Tools/)
- AJAX Based IOPT Petri Net Editor
- Simulation
- Remote Debugger
- State Space Generation Tool
- Model-checking using a Query System
- Automatic controller C code generator
- Automatic controller VHDL hardware synthesis
- Automatic IL code generator for PLCs



Development flow

Demo vídeo available at https://goo.gl/MxFHti



IOPT-Tools entry page

IOPT Tools

gres.uninova.pt/IOPT-Tools-V1.2/login.php

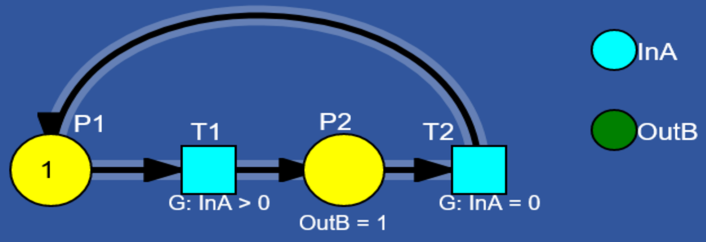
IOPT Tools

User Login:

Username:

Password:

Login | New User | Cancel



NOTES: Example models available under username "**models**" and password "**models**".

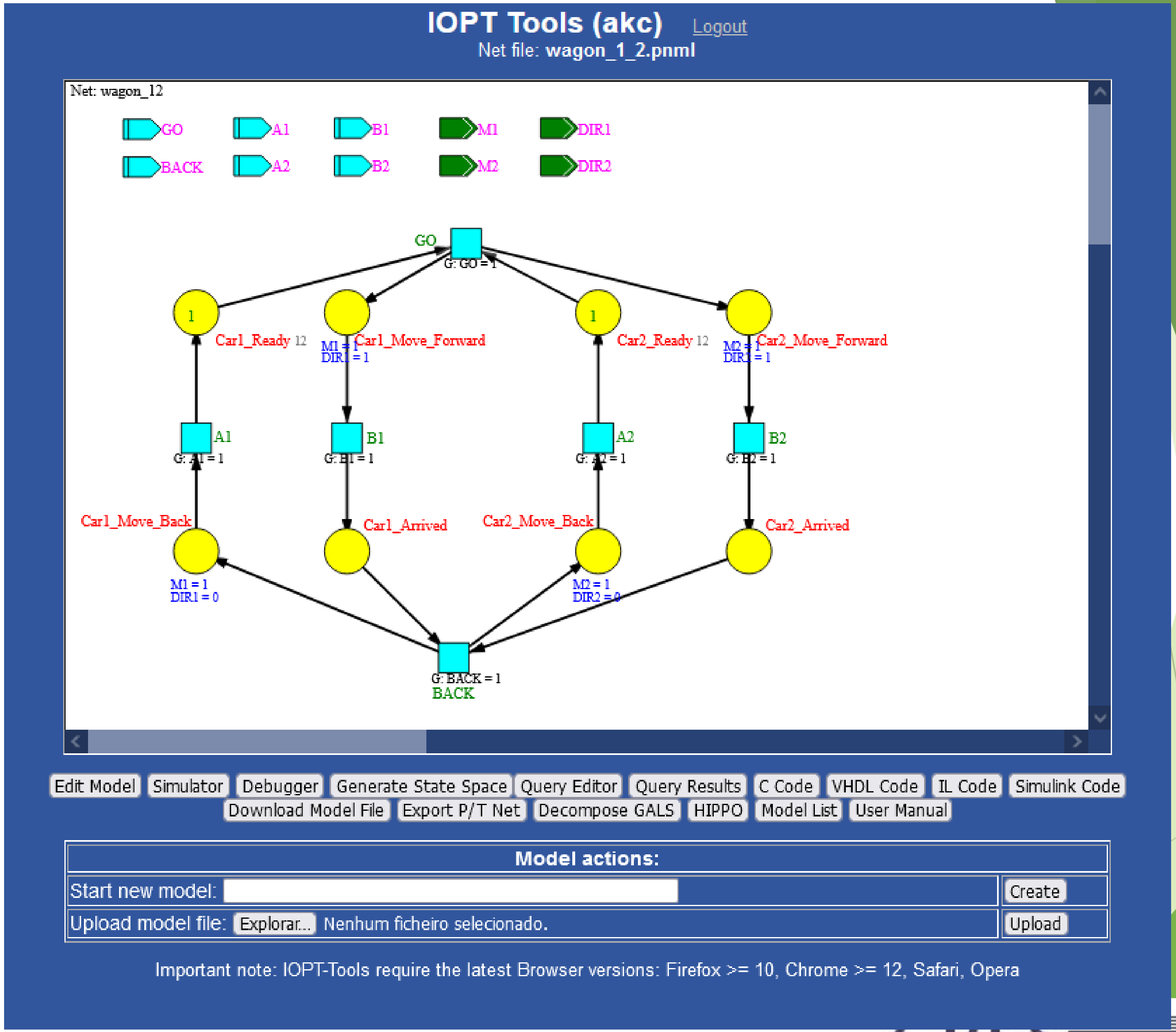
Anonymous users can login into the system with username "**guest**" and password "**guest**", to create new IOPT models and test all system functionalities. However, these models will be openly accessible to all users and may be modified by other users or deleted at any time. Therefore, creating a personal user account [free] is highly recommended.

For information about the tools please download the User Manual .

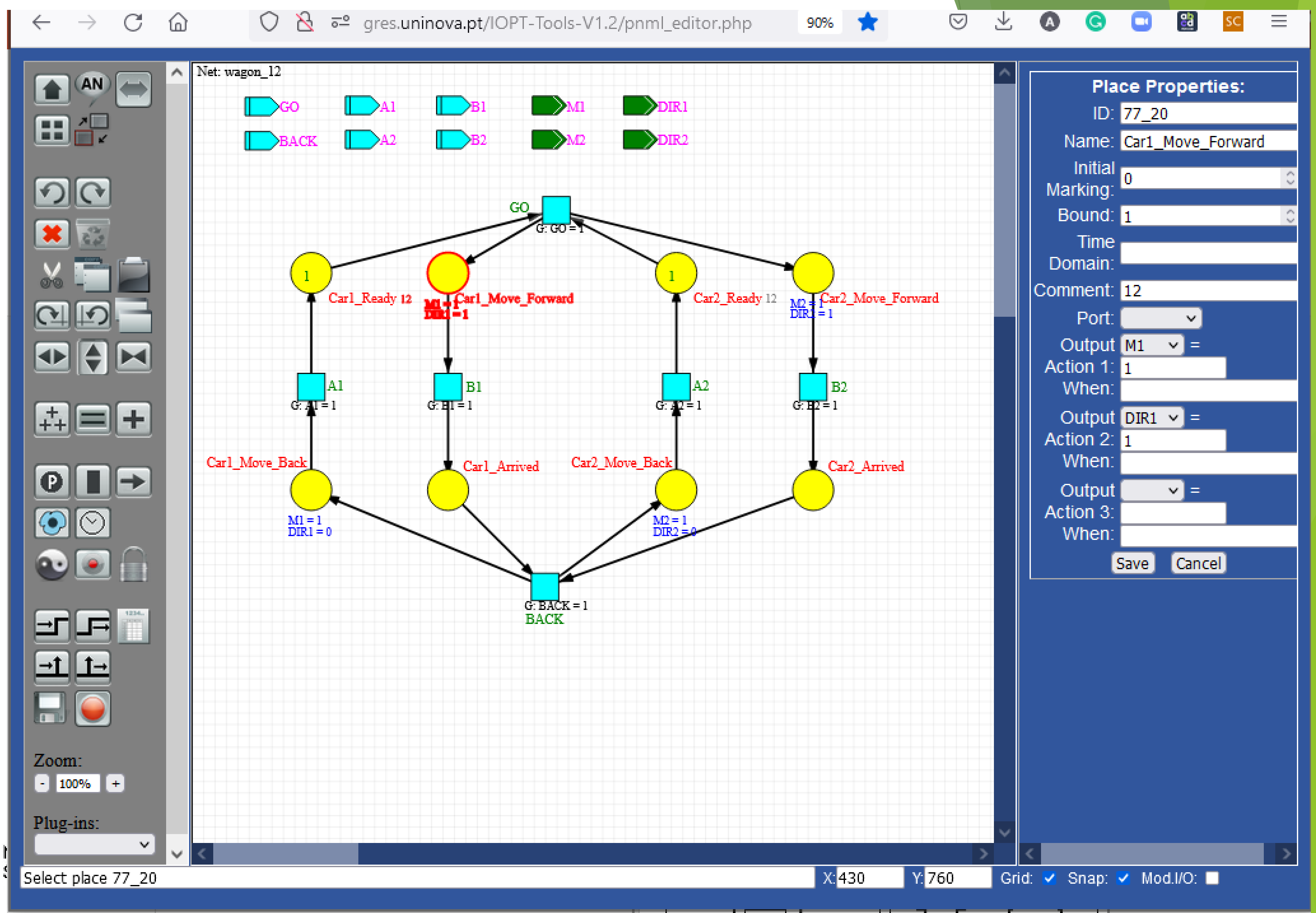
IOPT-Tools have been developed by several members of the R&D Group on Reconfigurable and Embedded Systems (GRES). At current development phase, some changes and improvements will occur in the near future. Comments or requests can be directed to gres@uninova.pt

Important note: IOPT-Tools require the latest Browser versions: Firefox >= 10, Chrome >= 12, Safari, Opera

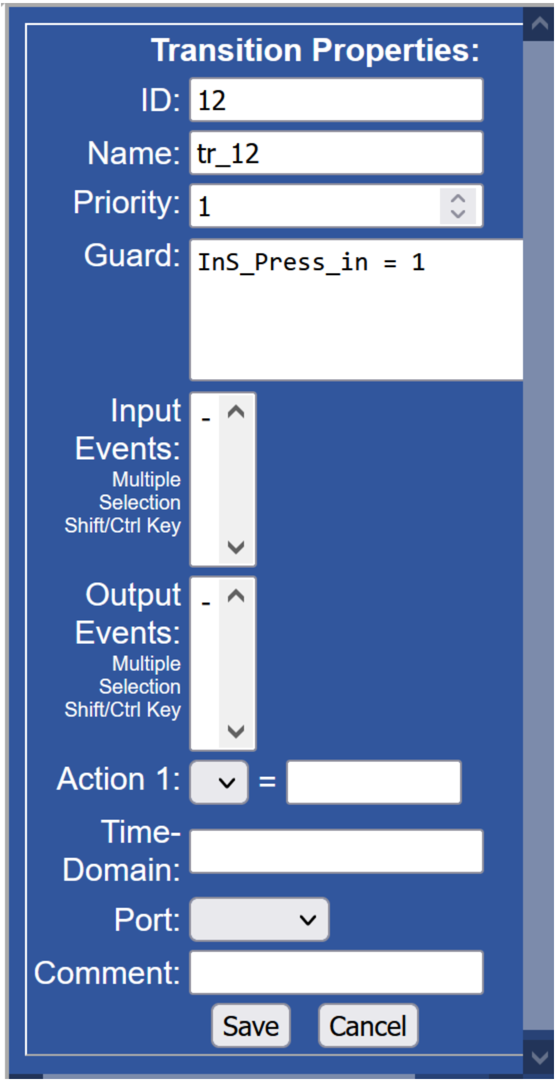
IOPT-Tools Overview

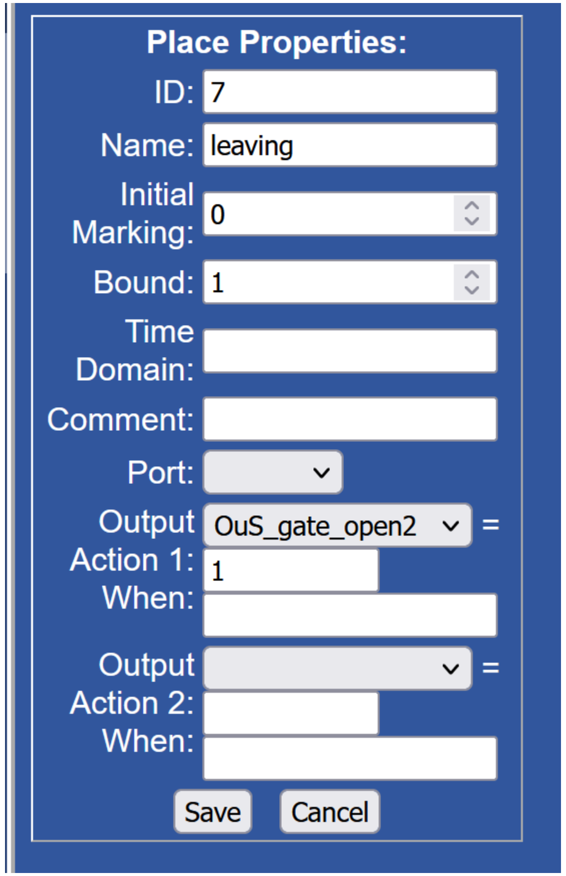


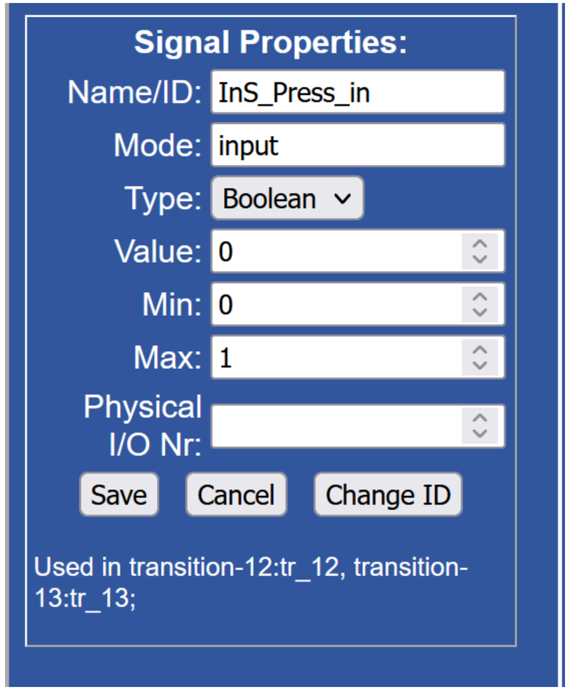
Editor

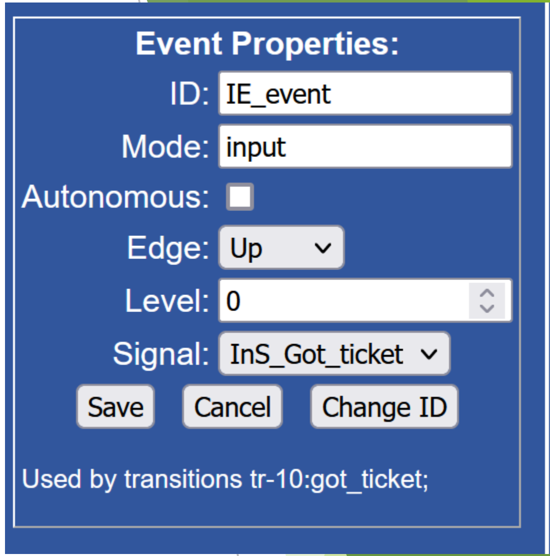


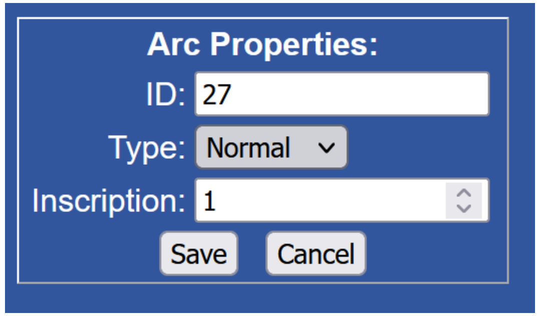
gres.uninova.pt/IOPT-Tools-V1.2/pnml_editor.php
Net: wagon_12
GO
A1
B1
M1
DIR1
BACK
A2
B2
M2
DIR2
Car1_Ready 12
Car1_Move_Forward
Car2_Ready 12
Car2_Move_Forward
Car1_Move_Back
Car1_Arrived
Car2_Move_Back
Car2_Arrived
G: GO = 1
G: BACK = 1
M1 = 1
DIR1 = 0
M2 = 1
Place Properties:
ID: 77_20
Name: Car1_Move_Forward
Initial Marking: 0
Bound: 1
Time Domain:
Comment: 12
Port:
Output Action 1: M1 = 1
When:
Output Action 2: DIR1 = 1
When:
Output Action 3:
When:
Save
Cancel
Zoom:
100%
Plug-ins:
Select place 77_20
X: 430
Y: 760
Grid:
Snap:
Mod.I/O:


Transition Properties:
ID: 12
Name: tr_12
Priority: 1
Guard: InS_Press_in = 1
Input Events:
Multiple Selection Shift/Ctrl Key
Output Events:
Multiple Selection Shift/Ctrl Key
Action 1: =
Time-Domain:
Port:
Comment:
Save
Cancel

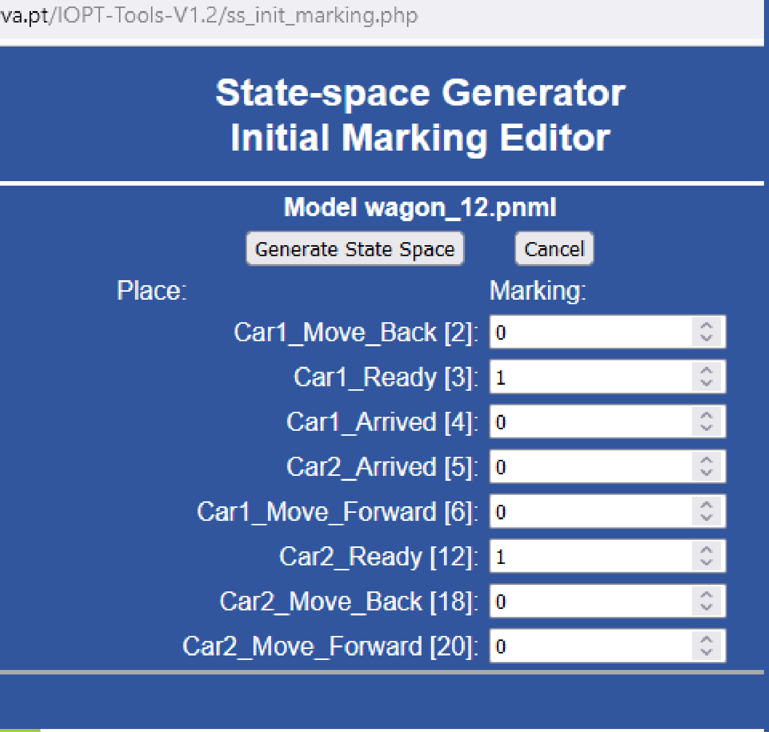

Place Properties:
ID: 7
Name: leaving
Initial Marking: 0
Bound: 1
Time Domain:
Comment:
Port:
Output Action 1: OuS_gate_open2 = 1
When:
Output Action 2: =
When:
Save
Cancel

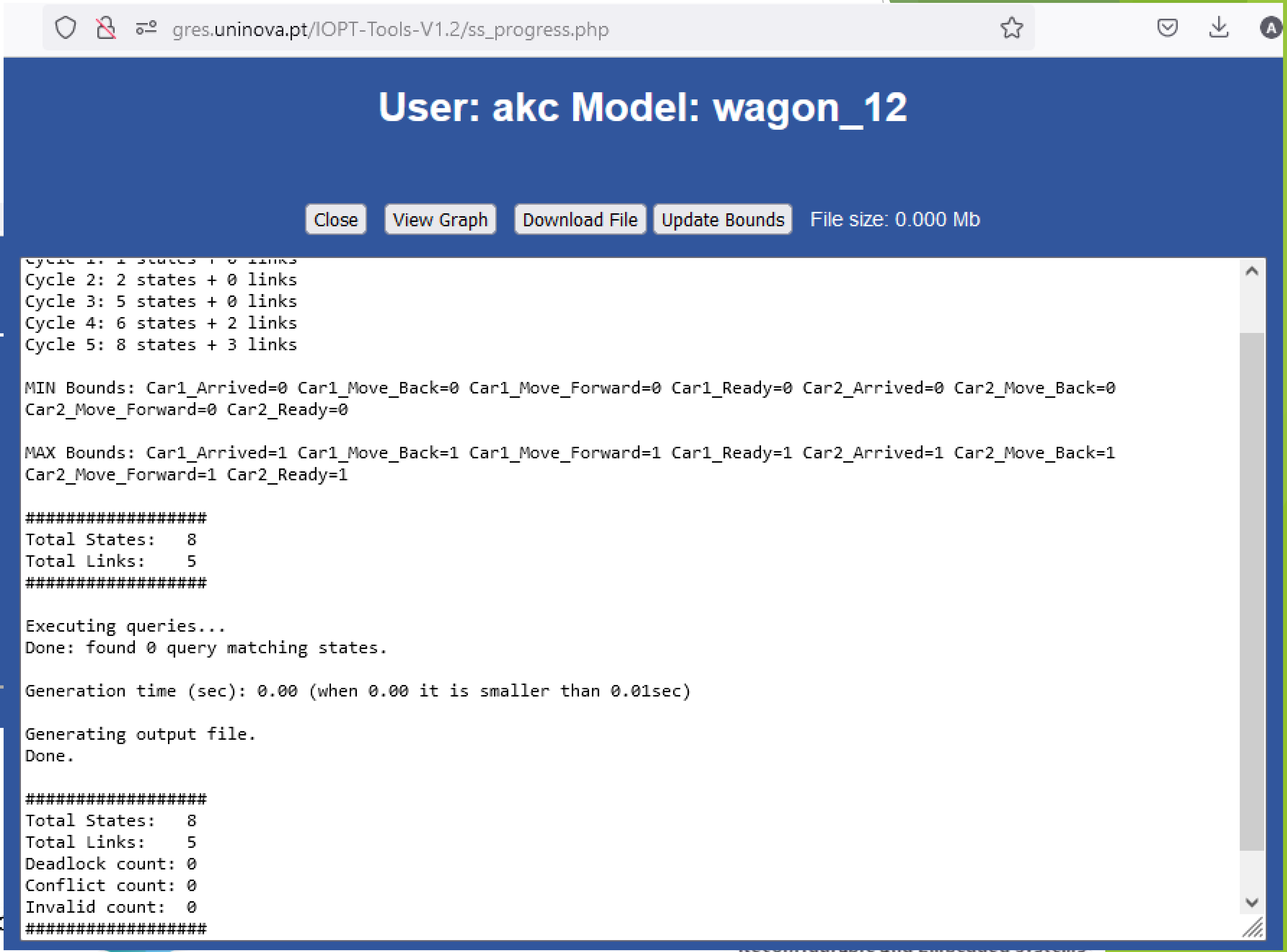

Signal Properties:
Name/ID: InS_Press_in
Mode: input
Type: Boolean
Value: 0
Min: 0
Max: 1
Physical I/O Nr:
Save
Cancel
Change ID
Used in transition-12:tr_12, transition-13:tr_13;


Event Properties:
ID: IE_event
Mode: input
Autonomous:
Edge: Up
Level: 0
Signal: InS_Got_ticket
Save
Cancel
Change ID
Used by transitions tr-10:got_ticket;


Arc Properties:
ID: 27
Type: Normal
Inscription: 1
Save
Cancel

Space state generator





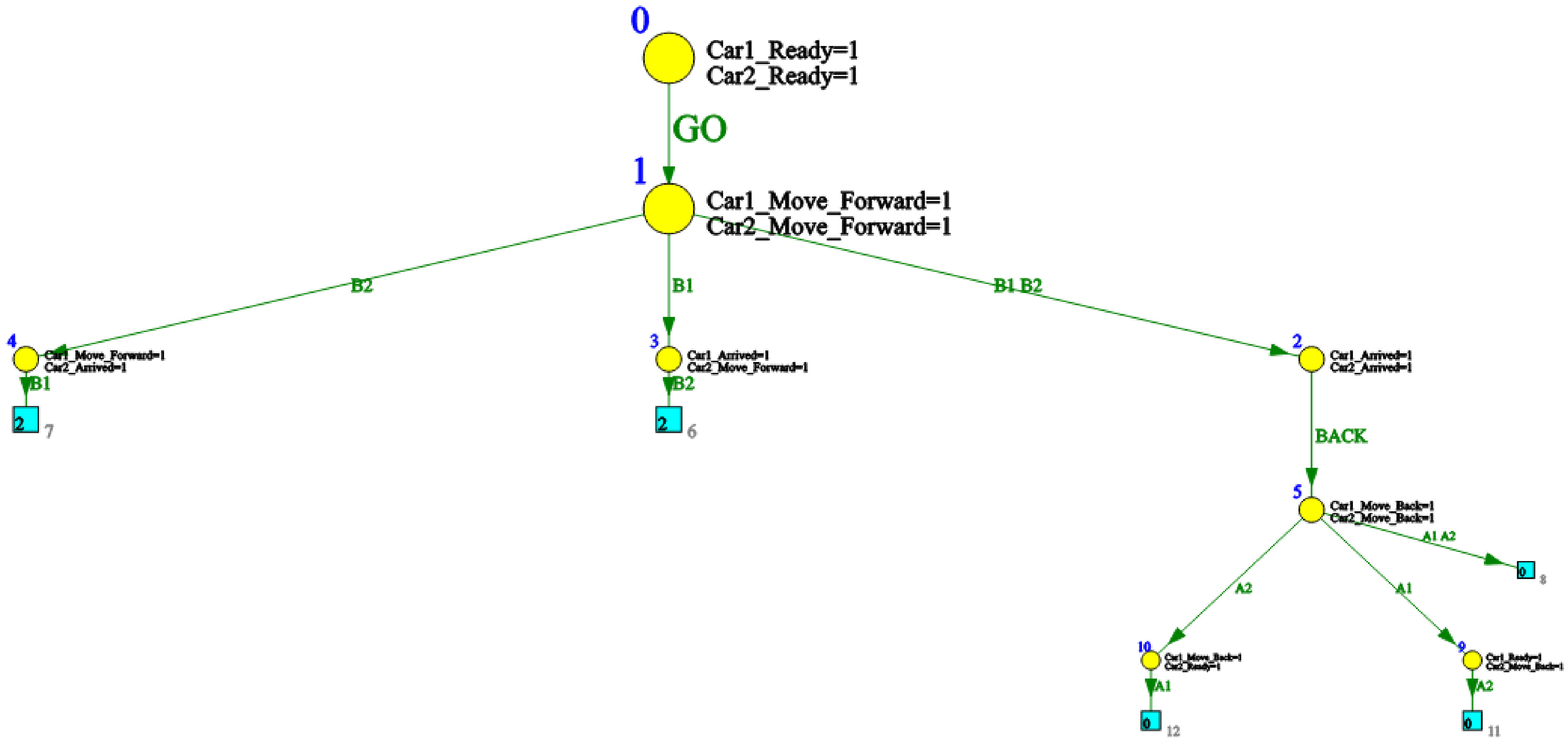
gres.uninova.pt/IOPT-Tools-V1.2/ss_operation.php?View=View+Graph

Net wagon_12

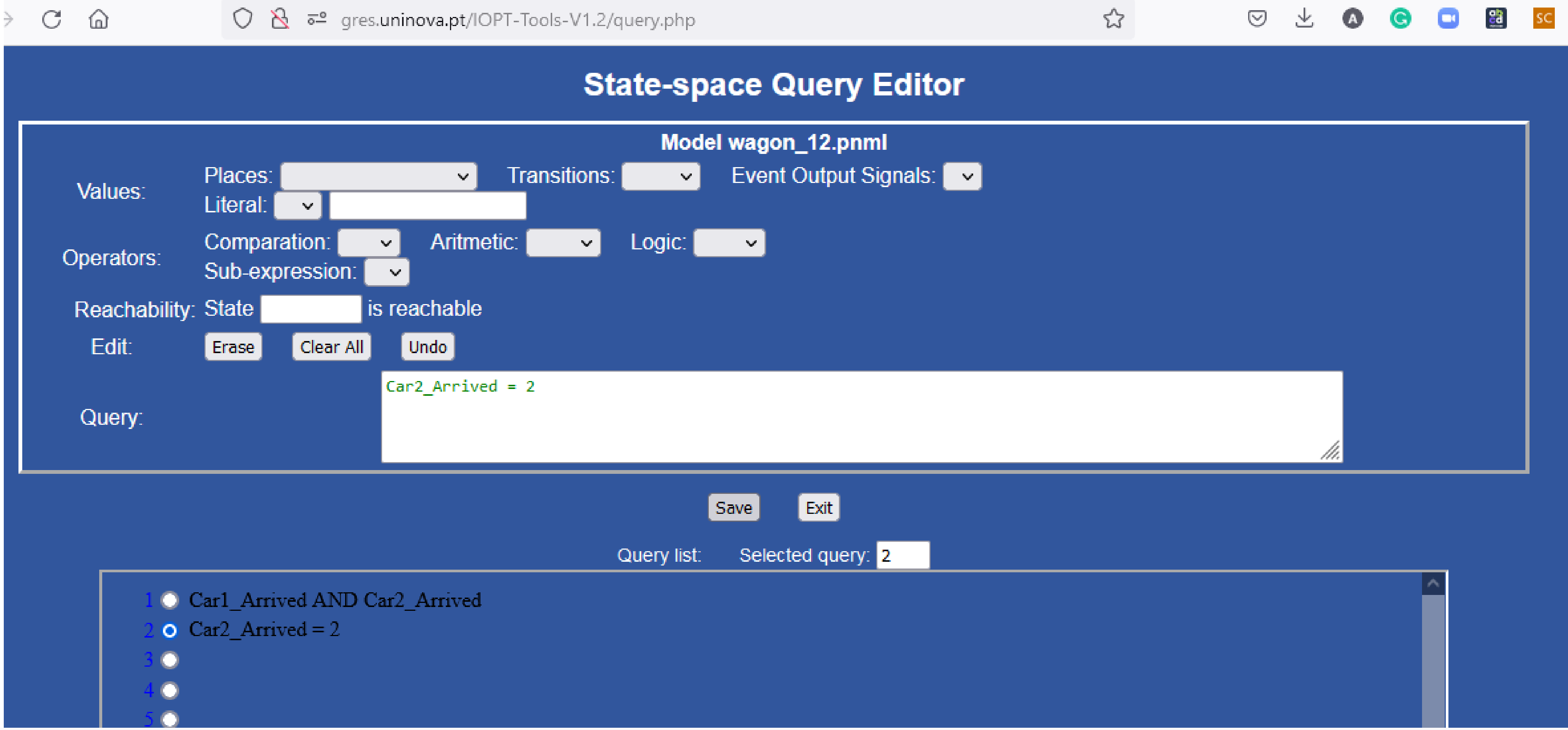
8 (from 8) Nodes, 5 Loops, 0 Deadlocks, 0 Conflicts, Max. Depth = 6, 0 Invalid

Min Bound = [Car1_Arrived=0 Car1_Move_Back=0 Car1_Move_Forward=0 Car1_Ready=0 Car2_Arrived=0 Car2_Move_Back=0 Car2_Move_Forward=0 Car2_Ready=0]

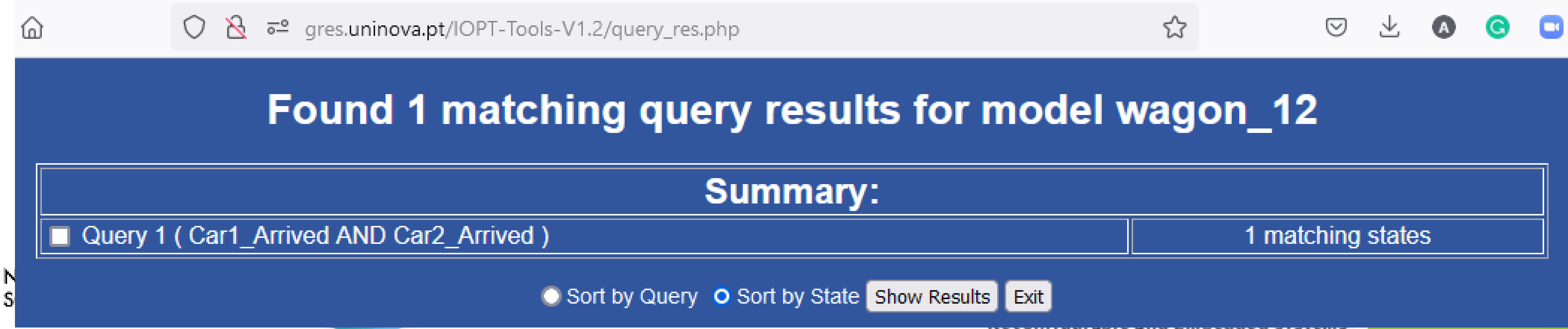
Max Bound = [Car1_Arrived=1 Car1_Move_Back=1 Car1_Move_Forward=1 Car1_Ready=1 Car2_Arrived=1 Car2_Move_Back=1 Car2_Move_Forward=1 Car2_Ready=1]



Query editor

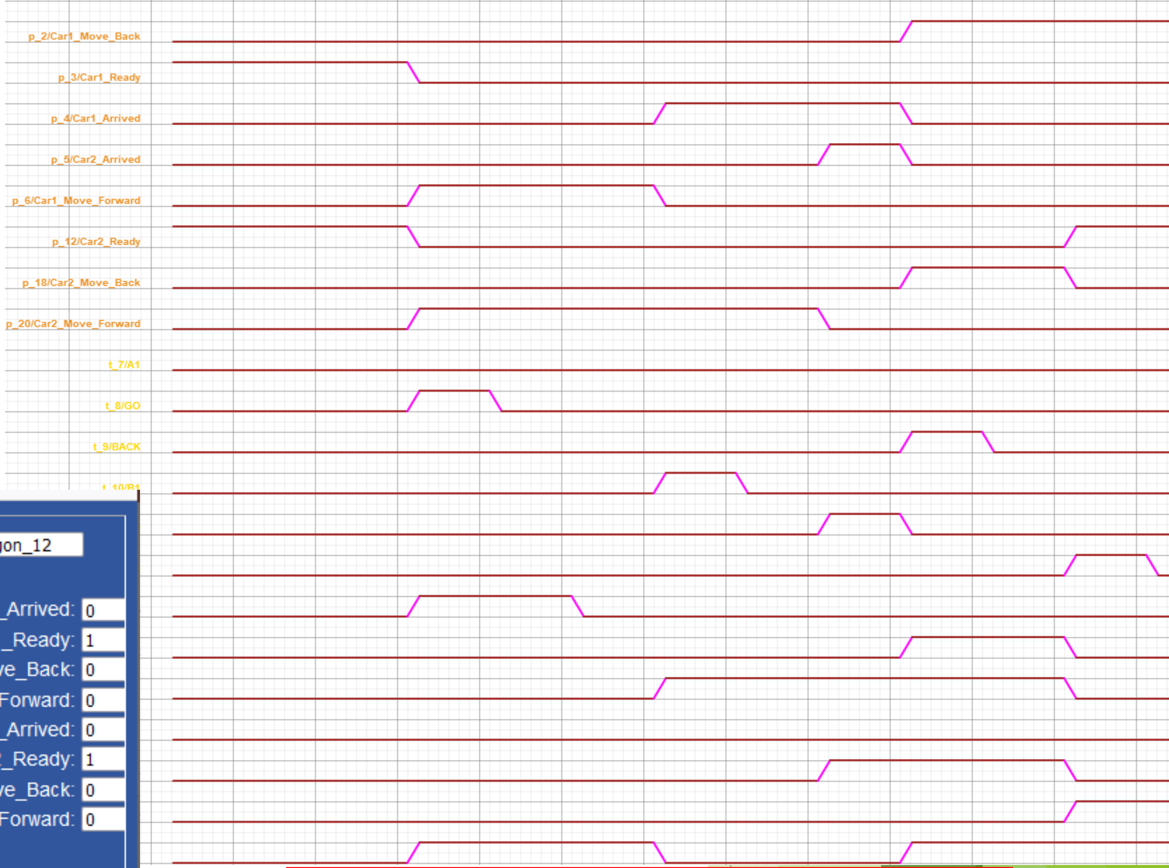


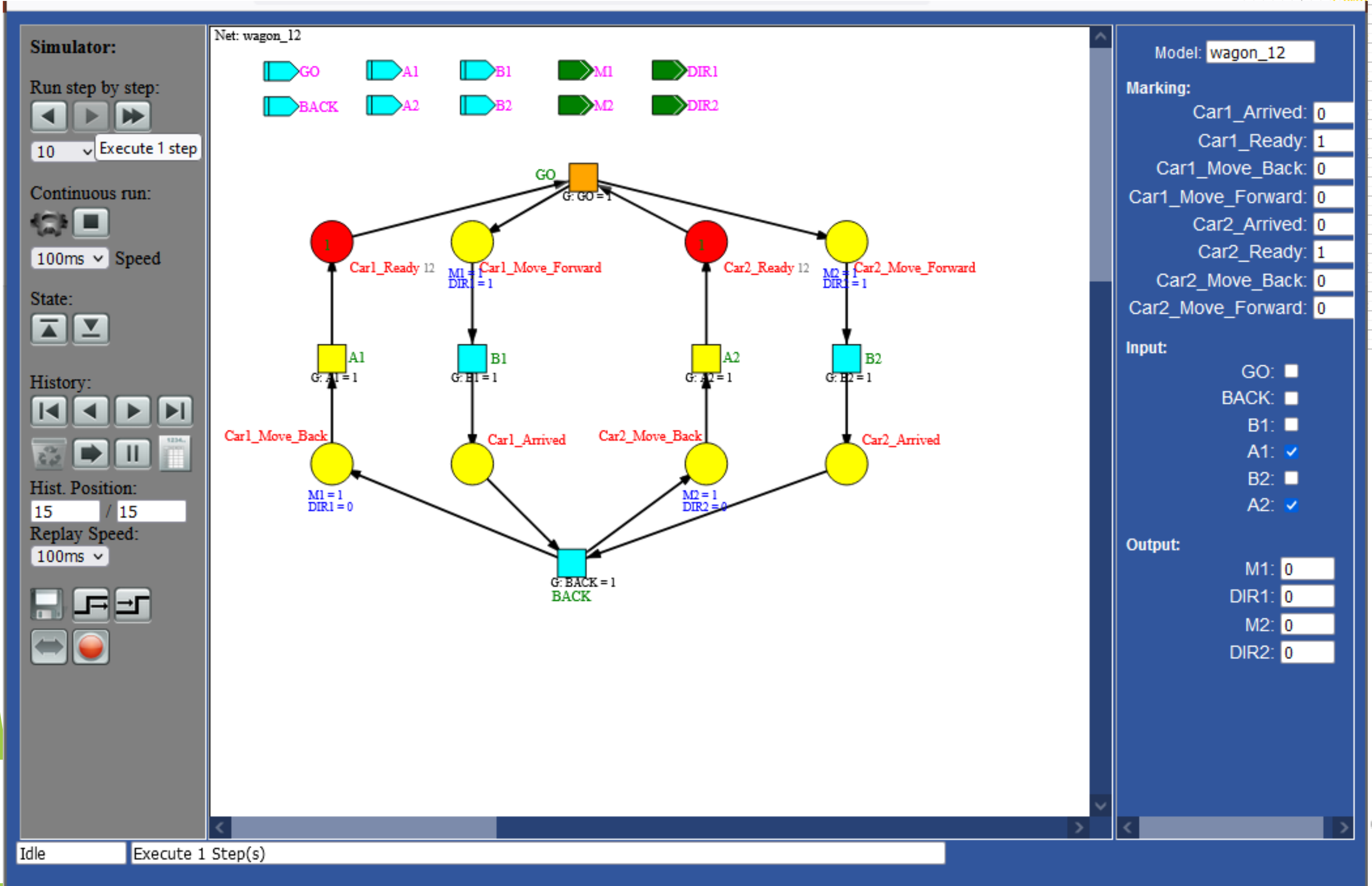
Query results



IOPT-Tools - Simulator

Token player



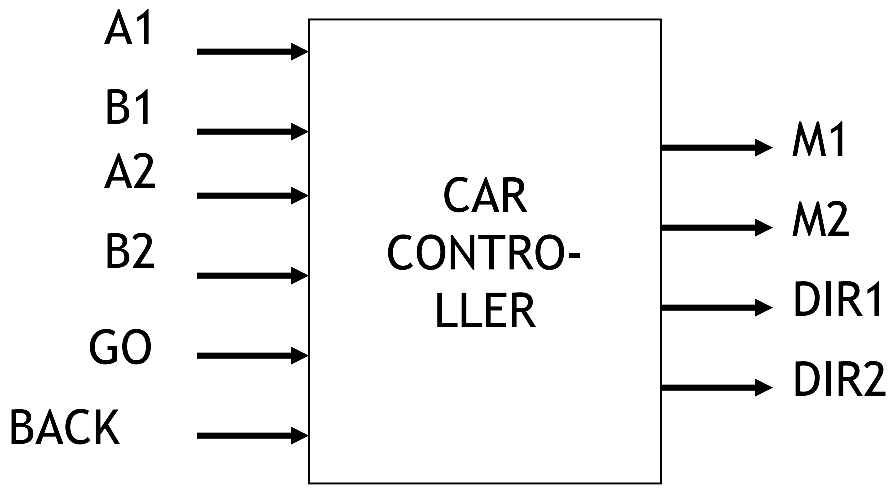


Timing diagram

d Systems

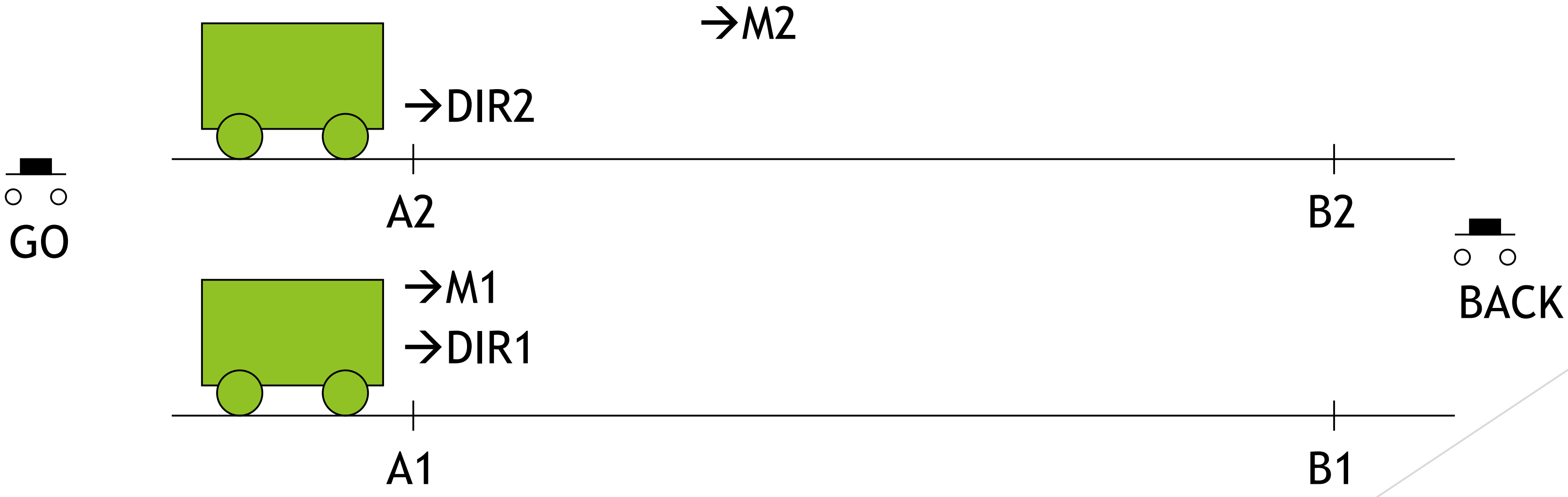
GRES
R&D Group on
Reconfigurable and Embedded Systems

An example

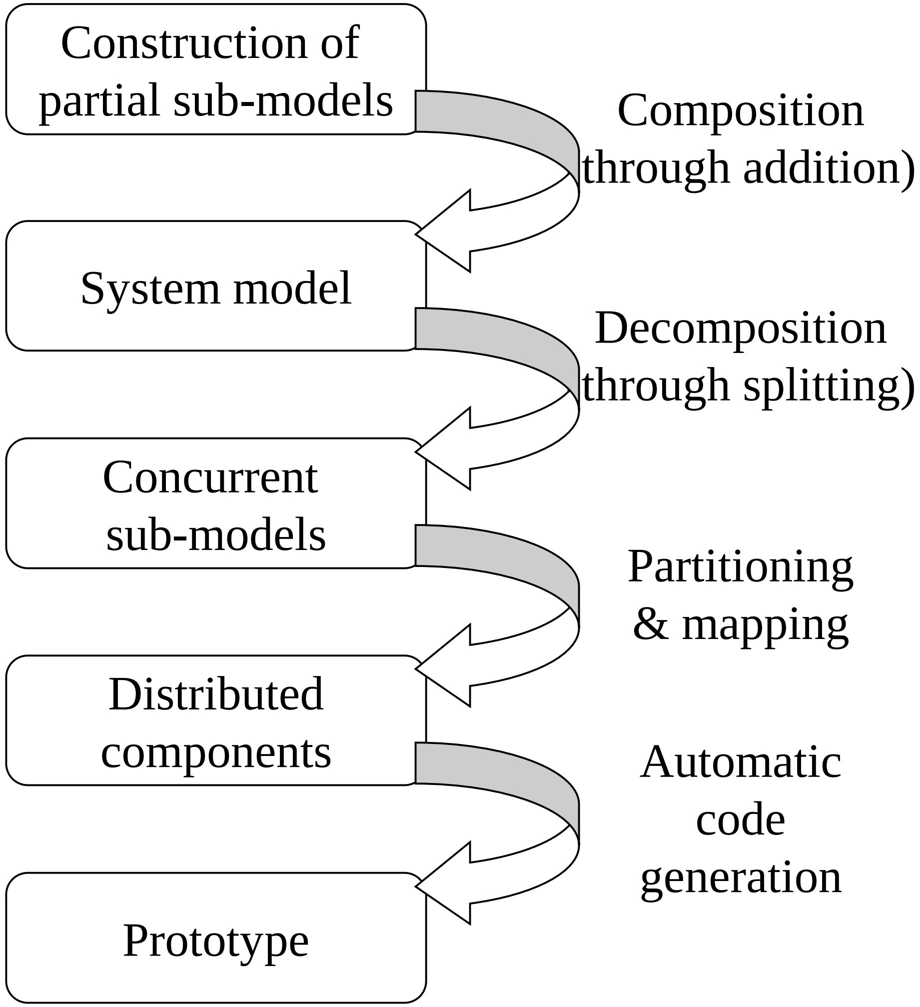


Goal:

To model the behaviour of a two car transportation system; Cars are synchronized at the beginning and at the end.

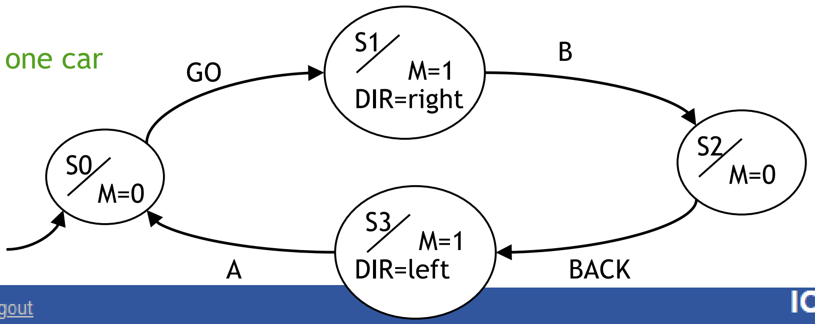


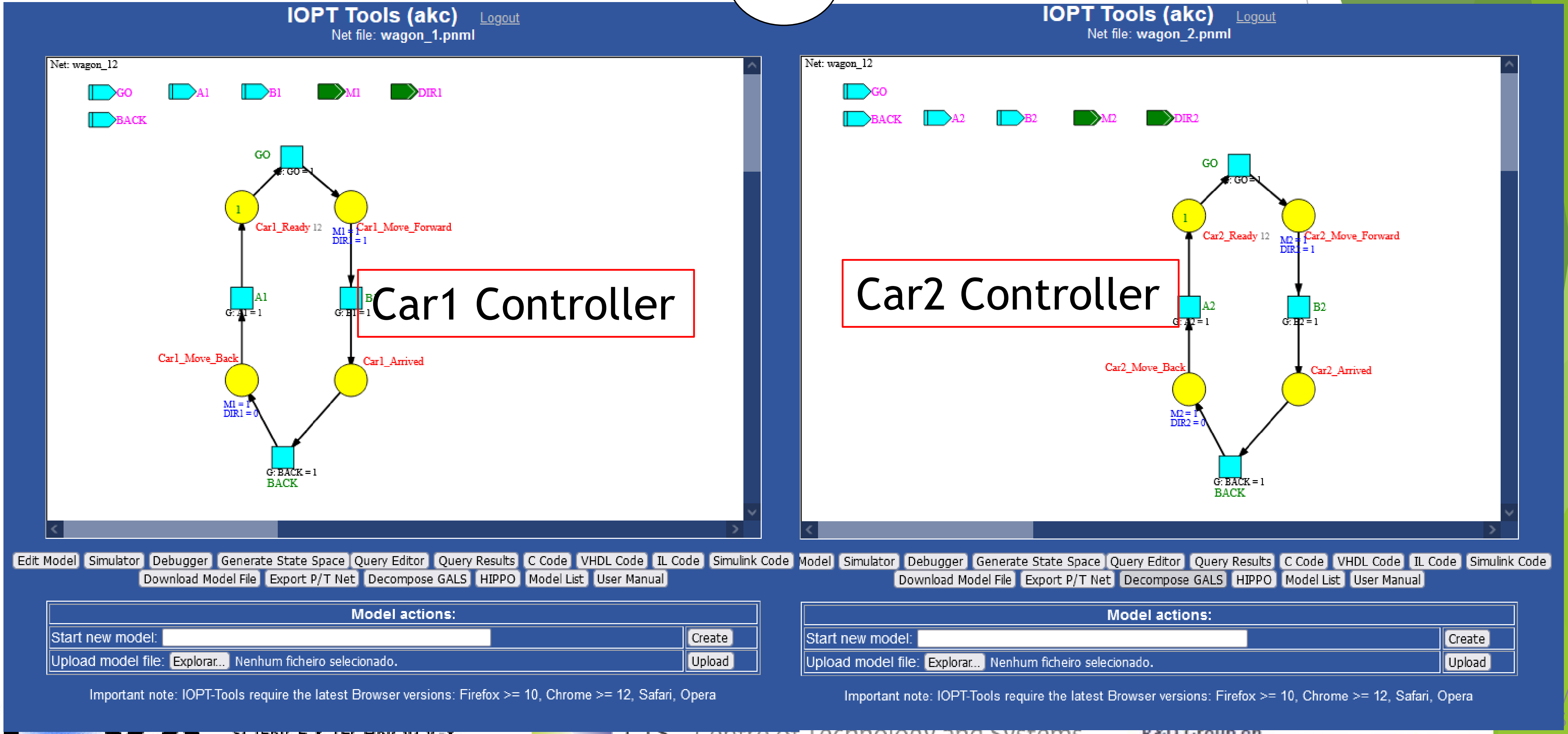
Underlying methodology



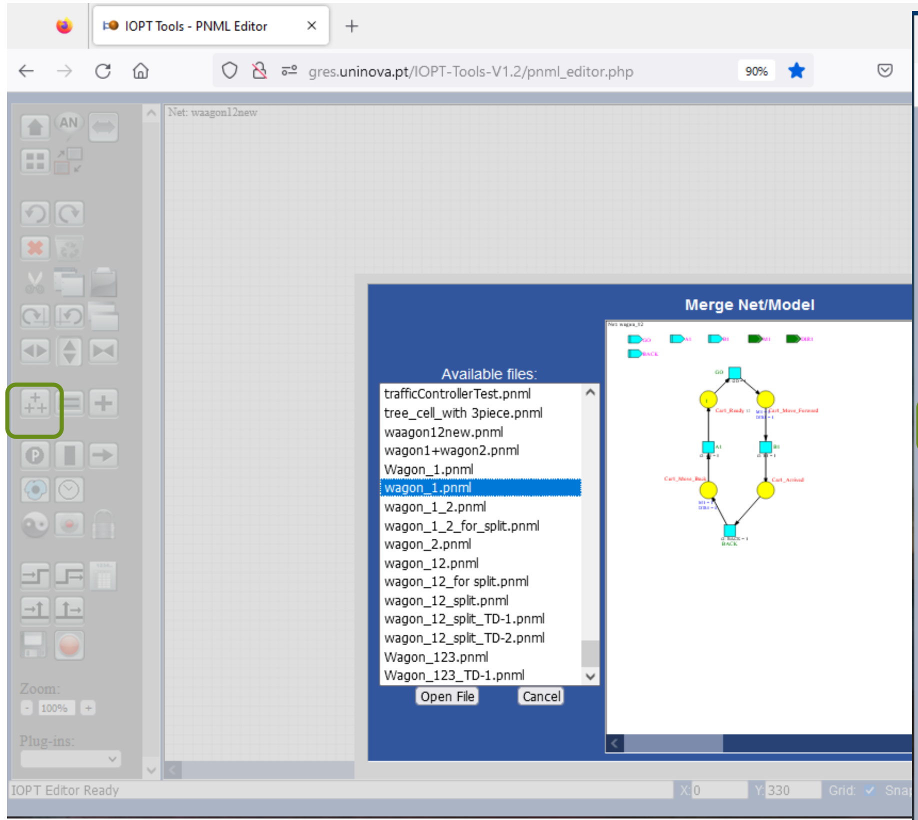
Partial models

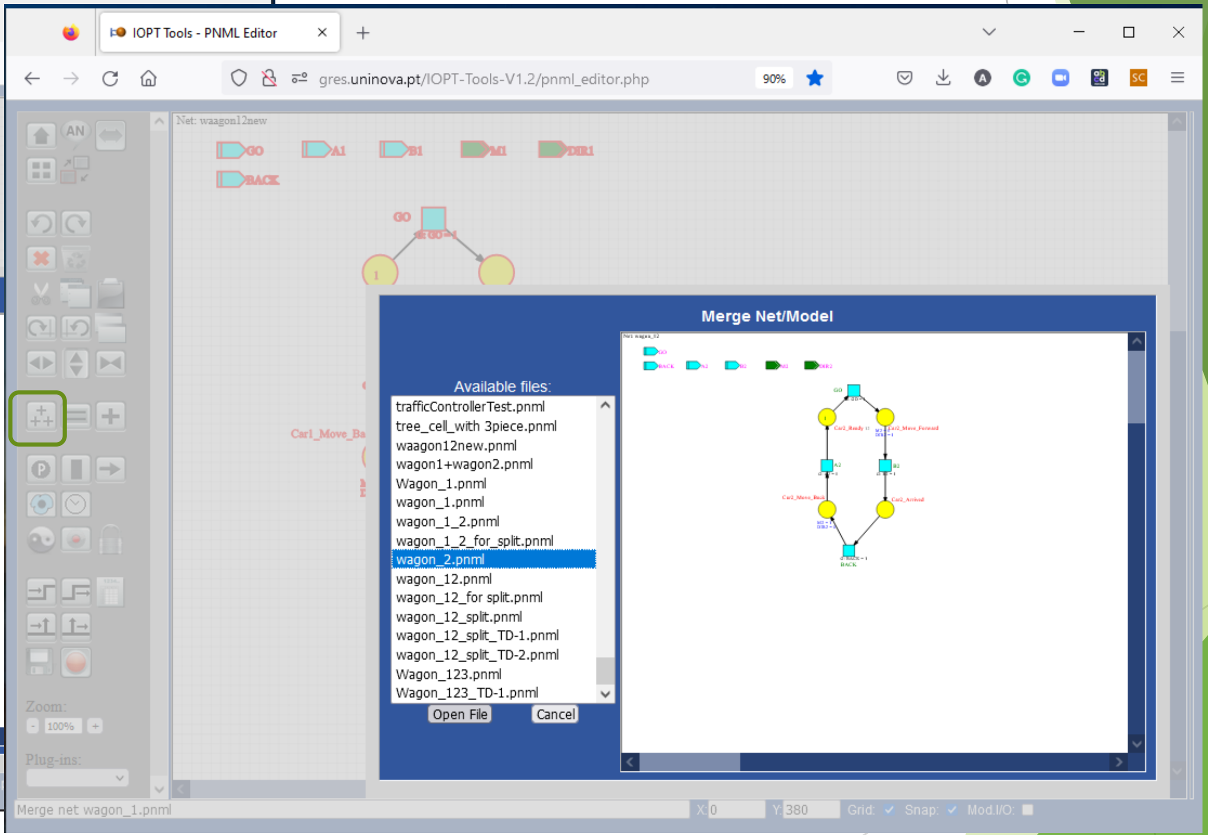
With one car



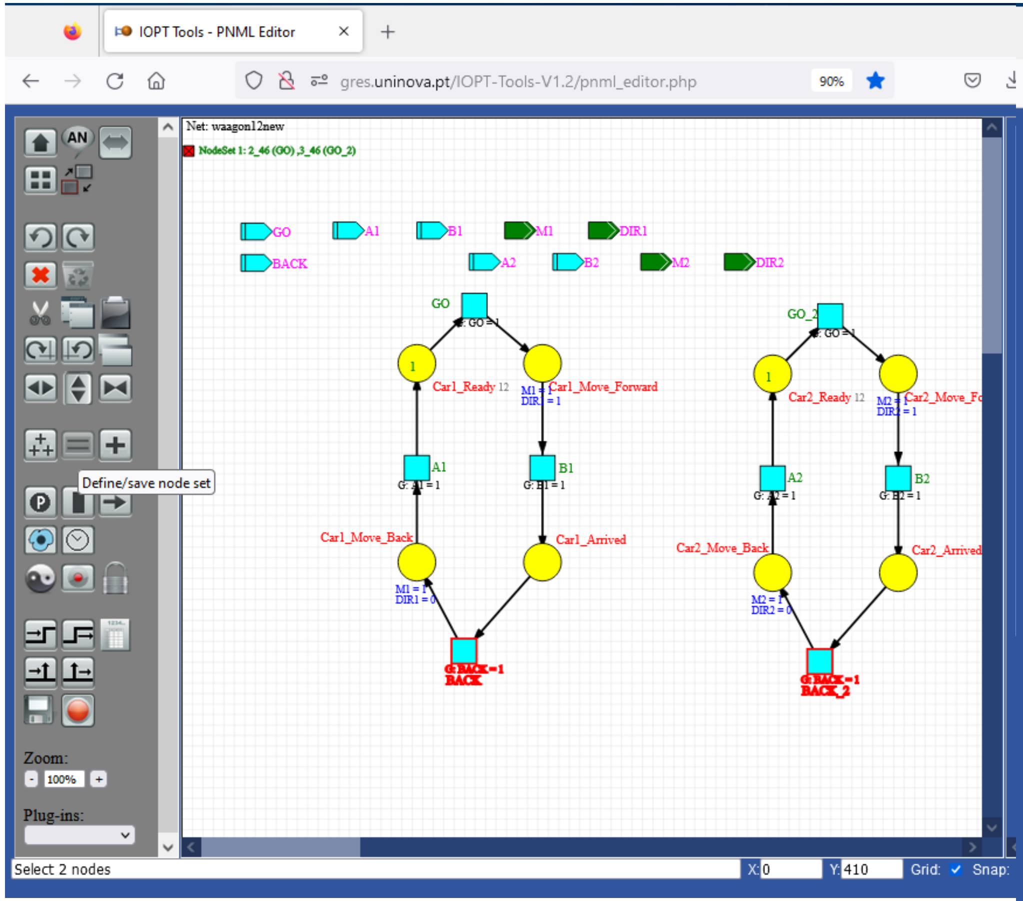


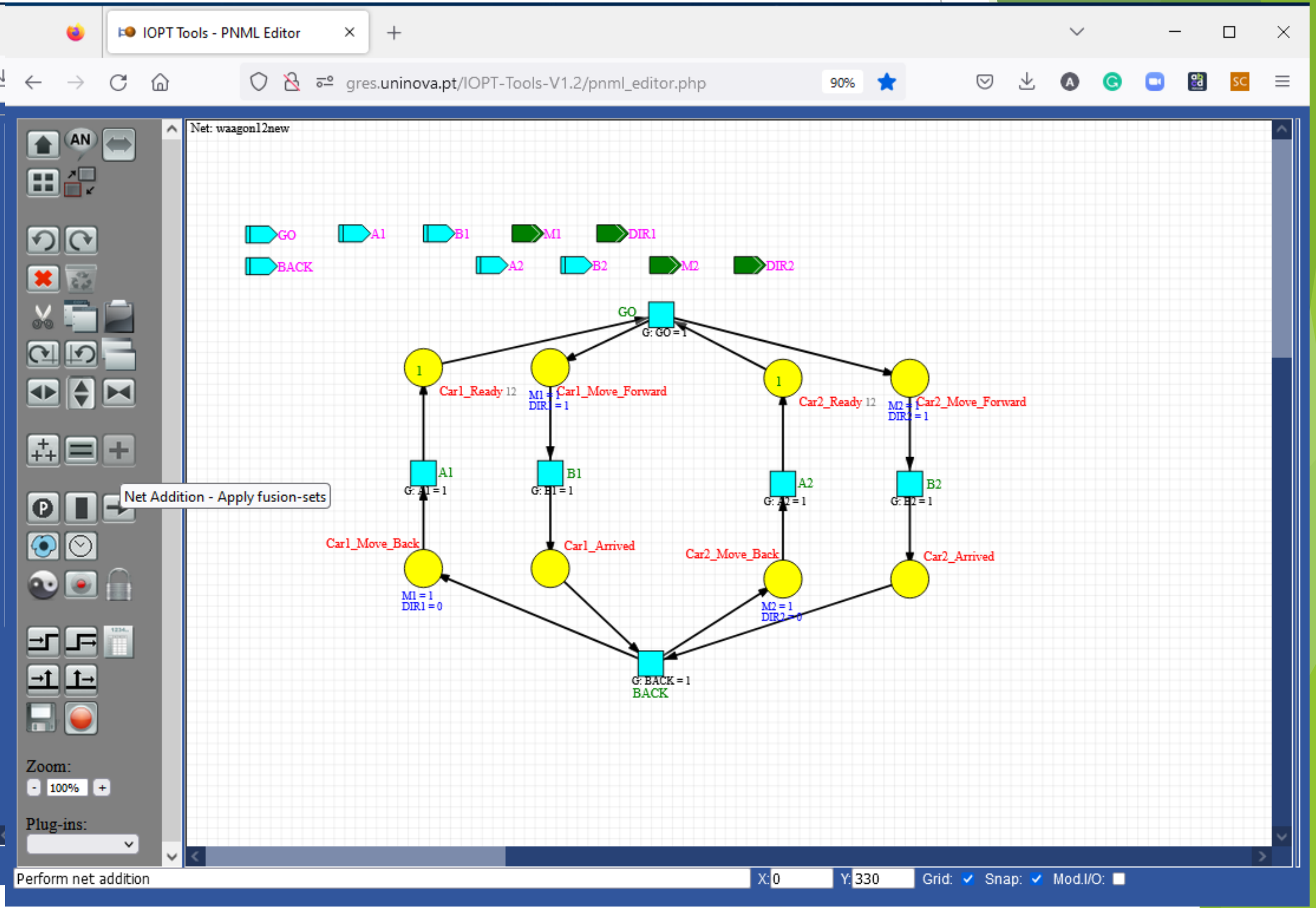
Model composition
Net Merging



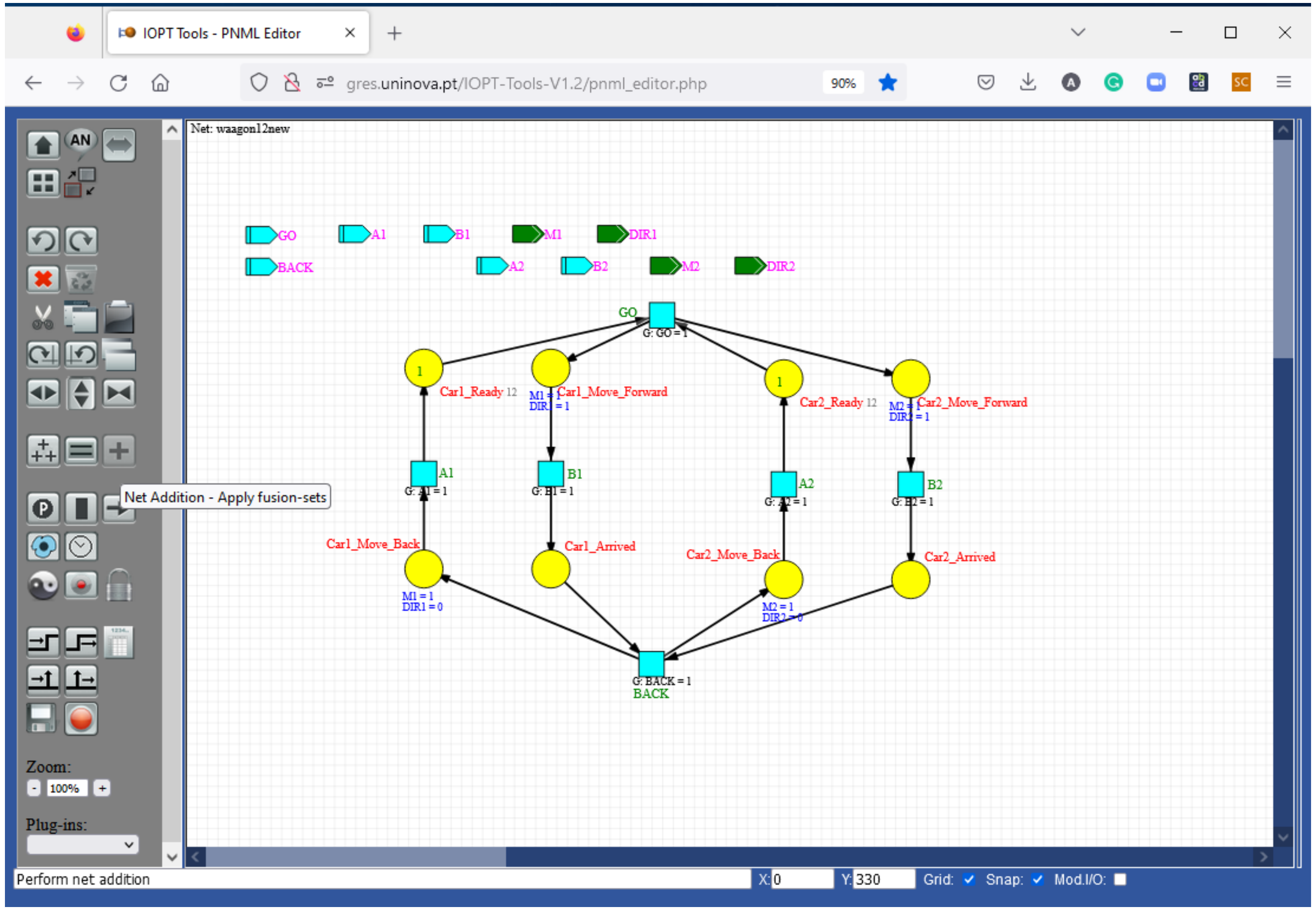


Model composition
Node fusion





System model – results of net addition

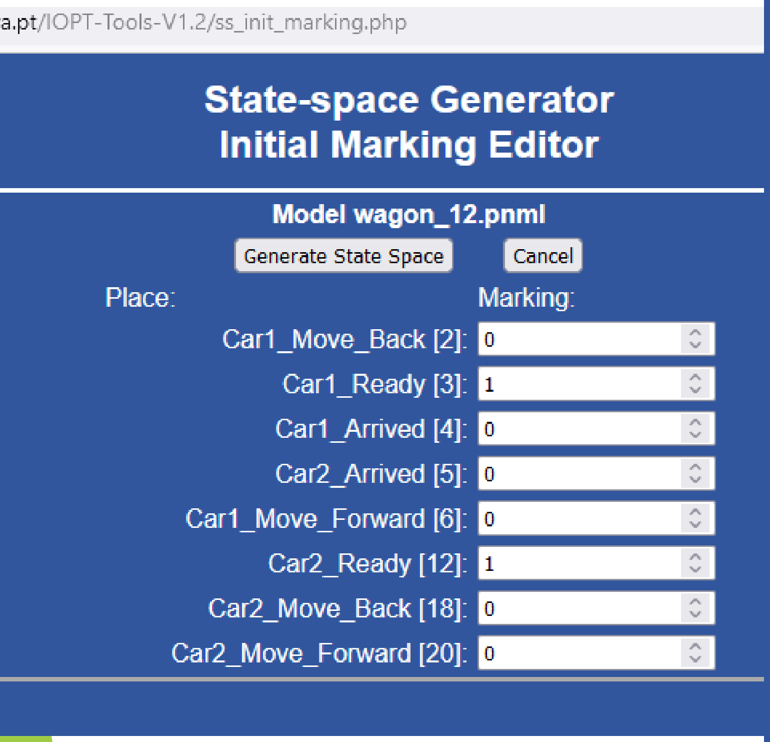


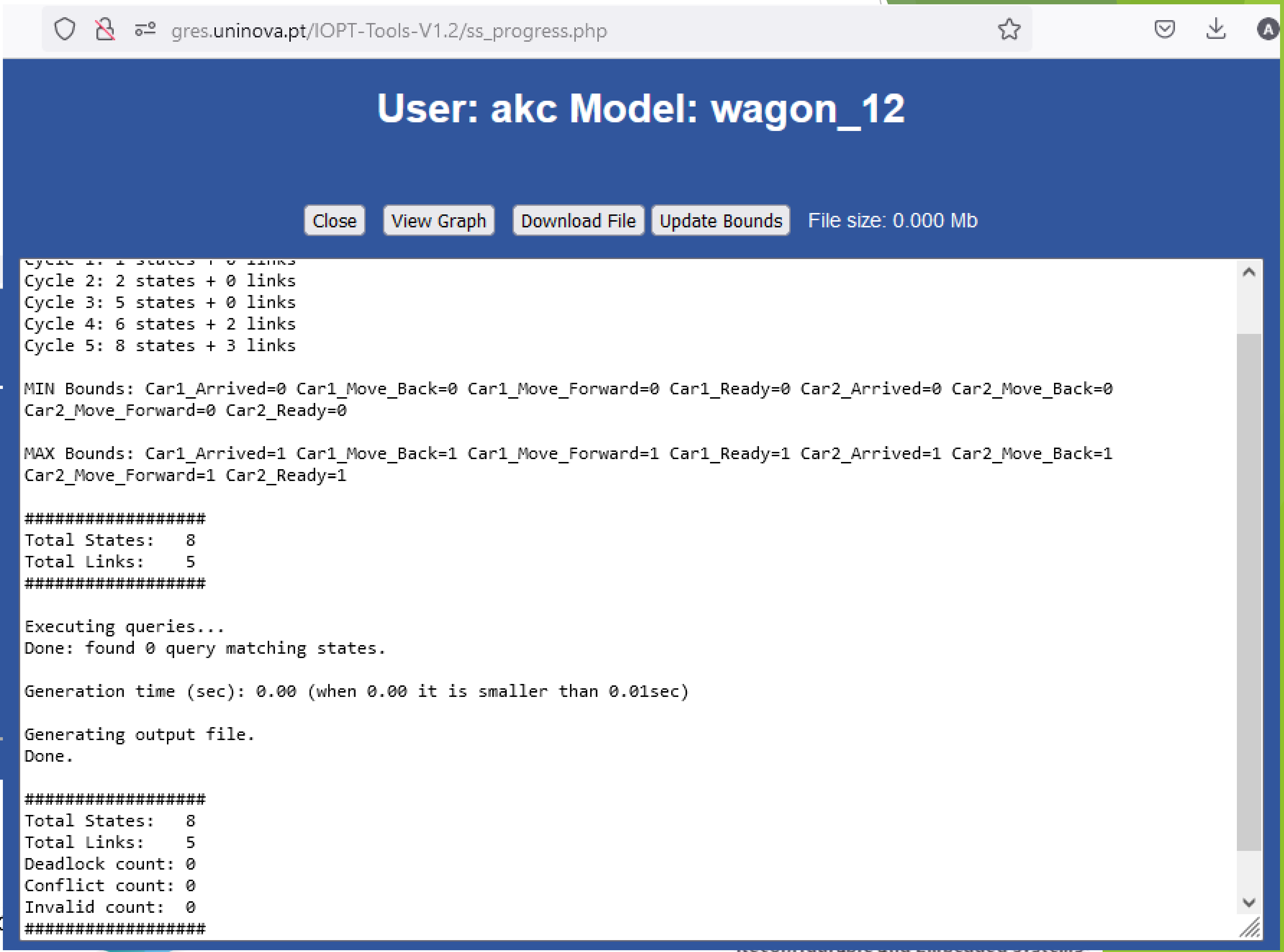
NOVA SCHOOL OF SCIENCE & TECHNOLOGY

CTS - Centre of Technology and Systems

GRES R&D Group on Reconfigurable and Embedded Systems

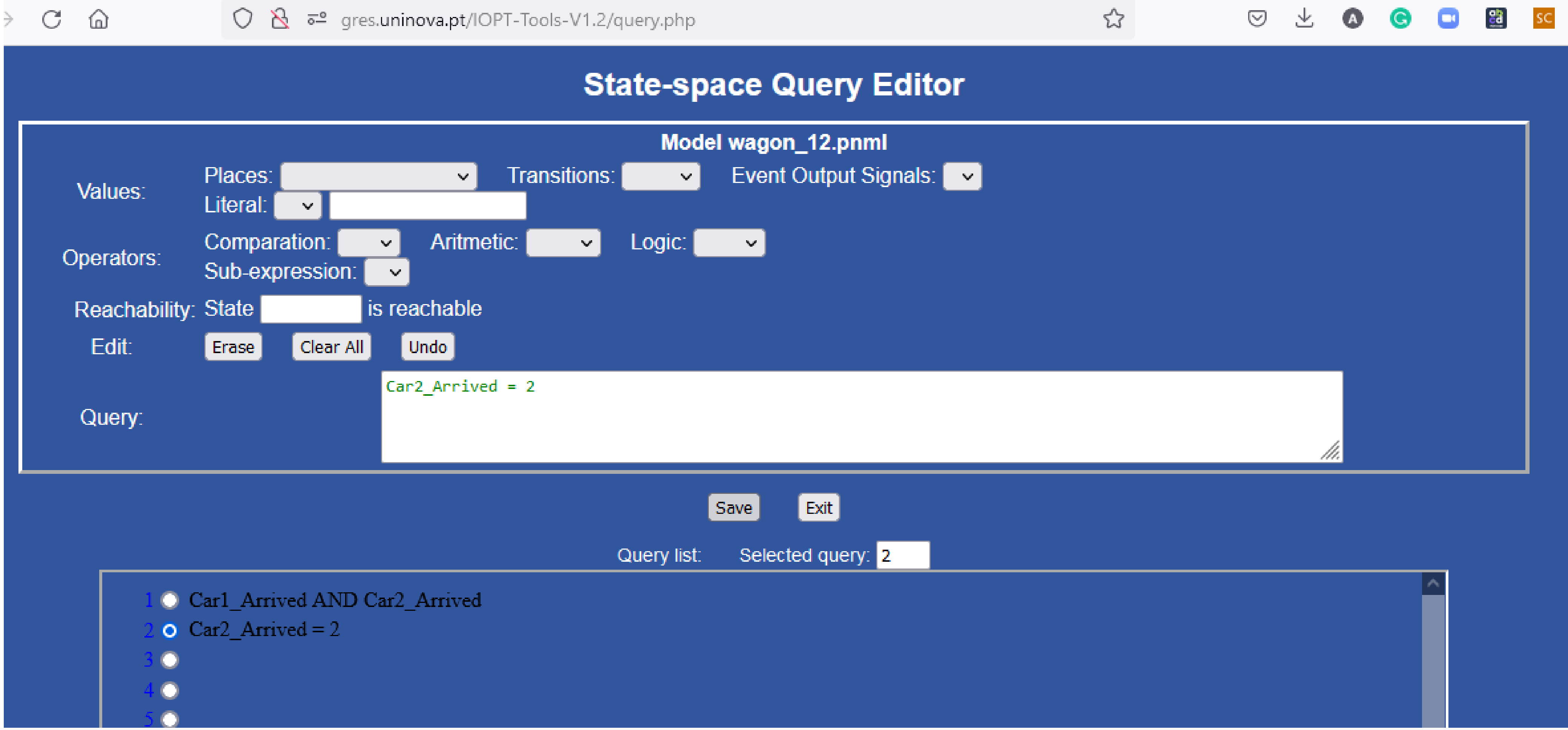
Space state generator



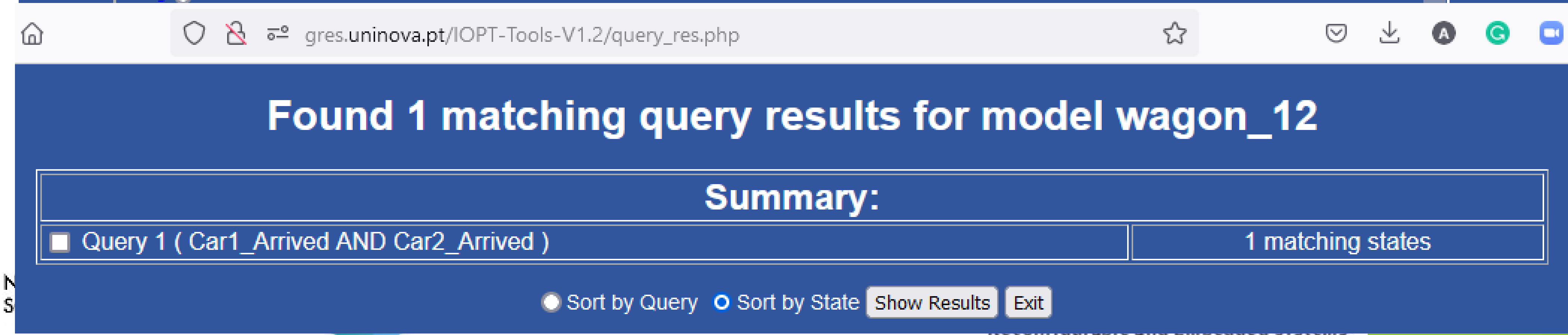


NOVA SCHOOL OF
SCIENCE & TECHNOLOG

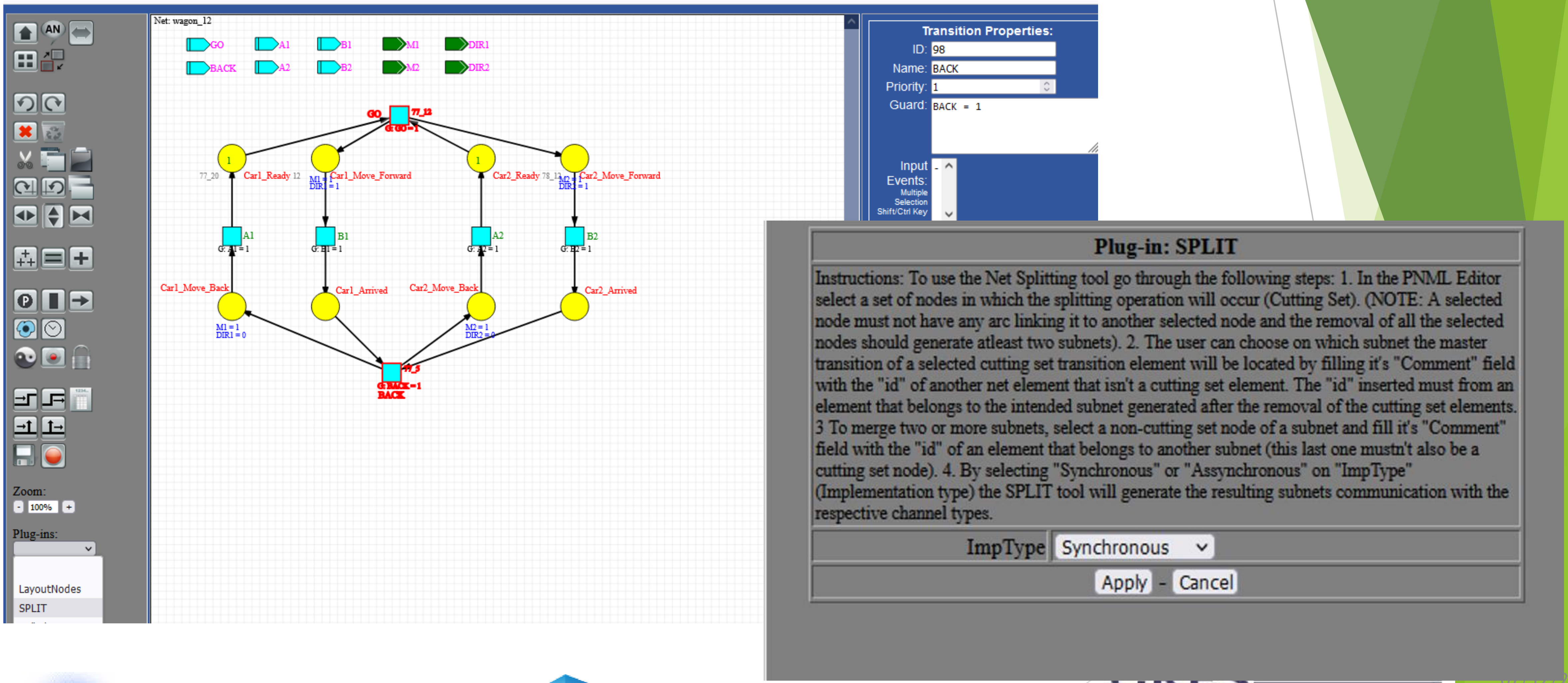
Query editor



Query results



Component decomposition - Net Splitting

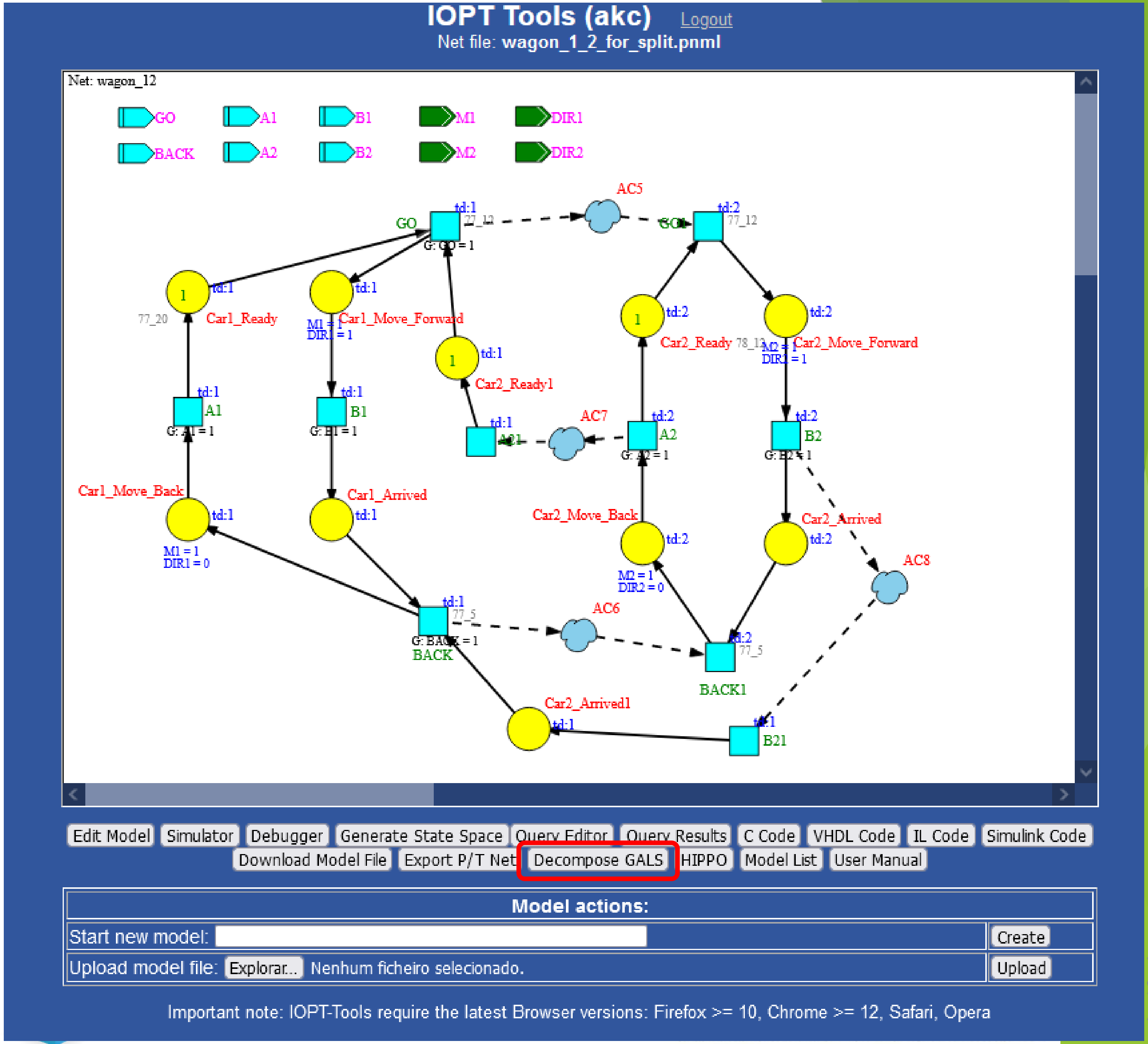


NOVA SCHOOL OF SCIENCE & TECHNOLOGY

CTS - Centre of Technology and Systems

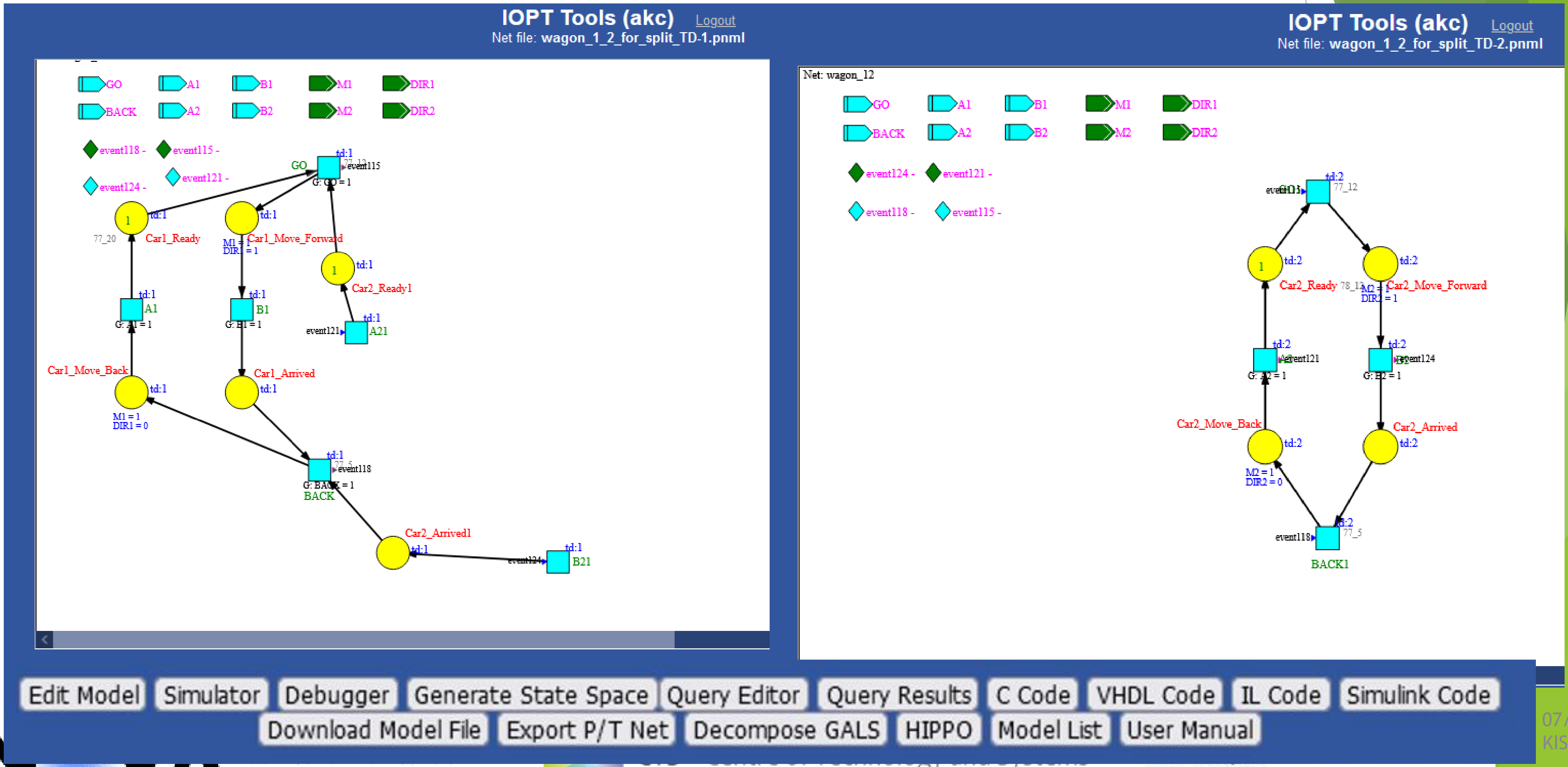
R&D Group on Reconfigurable and Embedded Systems

Net splitting



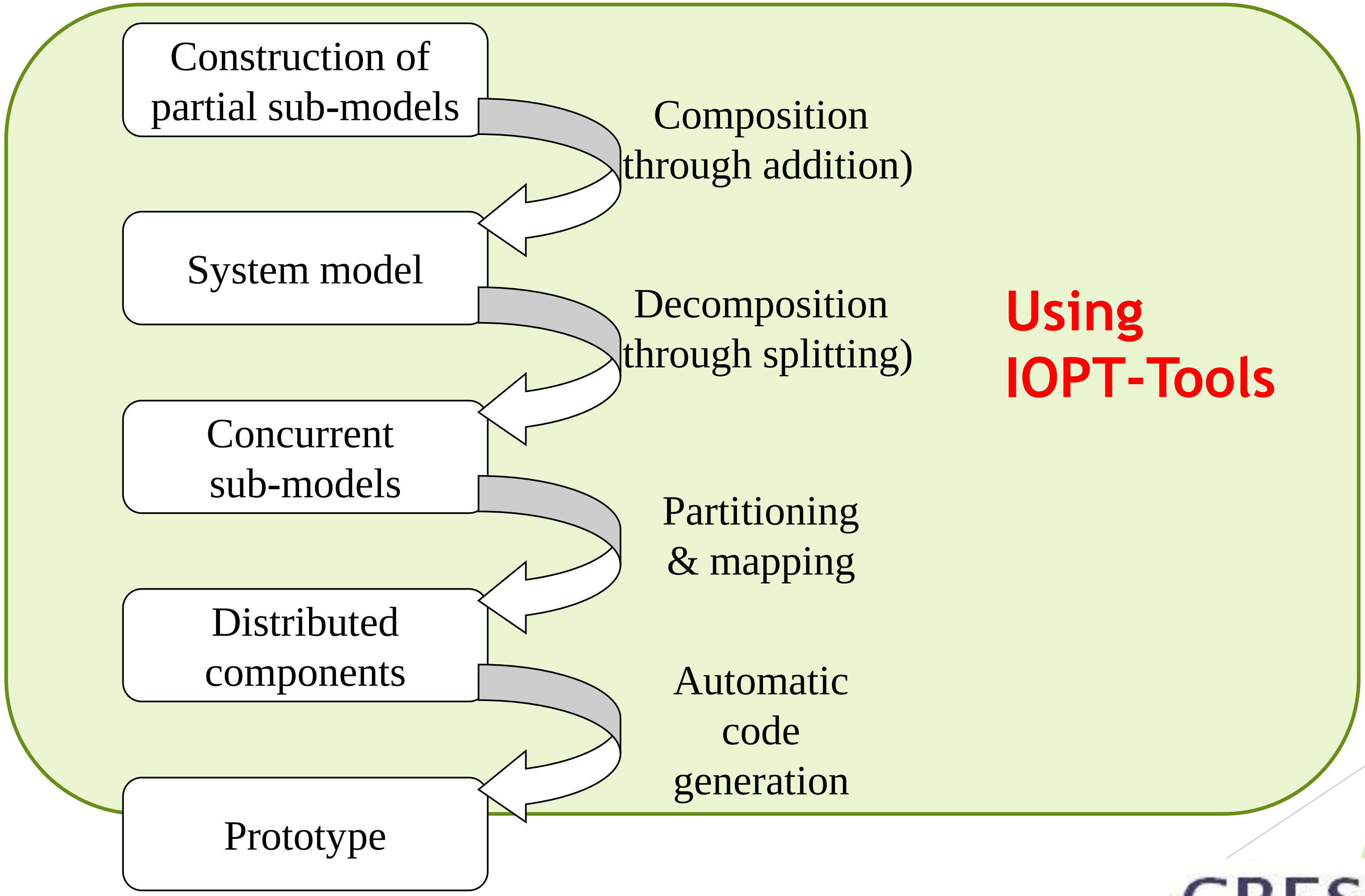
NOVA SCHOOL OF SCIENCE & TECHNOLOGY

Distributed component



Reconfigurable and Embedded Systems

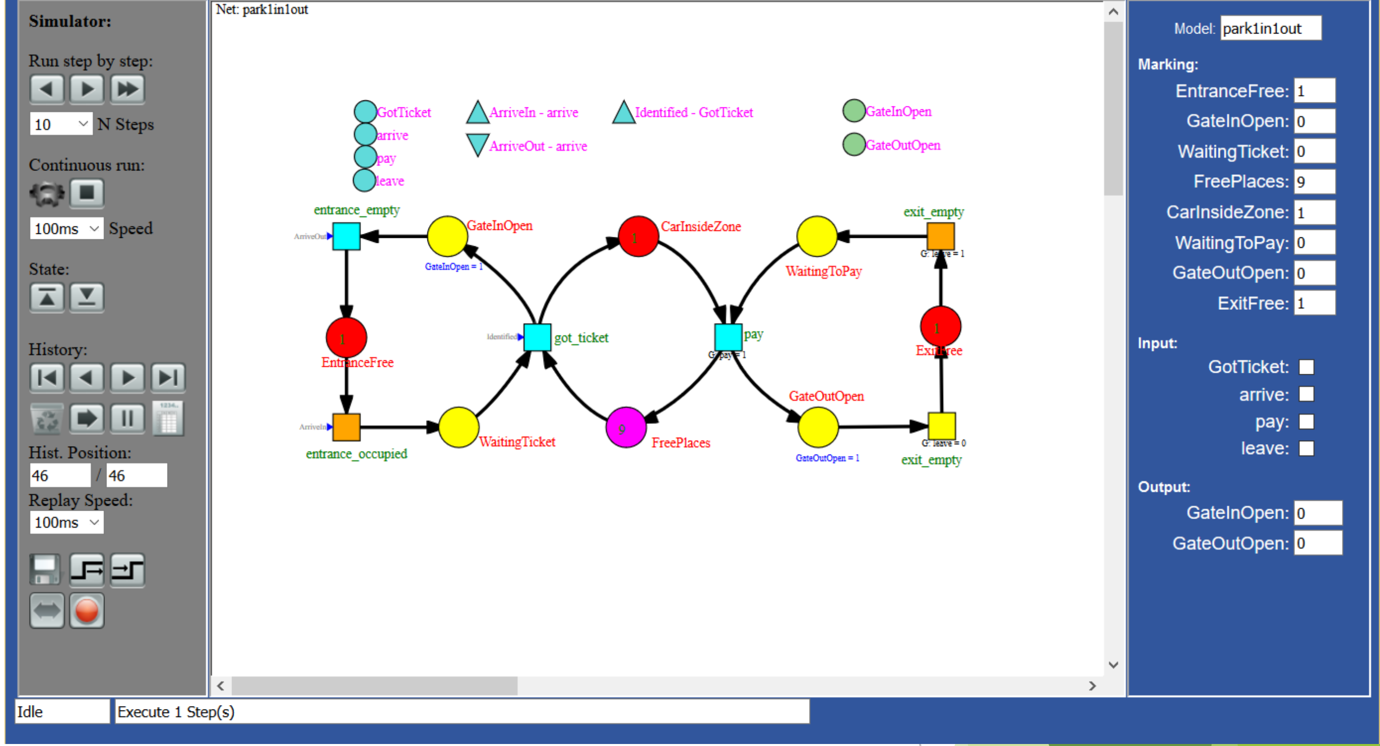
Underlying methodology



IOPT-Tools – Remote Debugger

- Communication architecture to enable the remote control, monitoring and debug of embedded system controllers designed using IOPT Petri nets.
- The architecture adds Internet connectivity capabilities to the controllers produced by the automatic code generators, enabling online remote debugging and monitoring using the IOPT simulator tool.
- Furthermore, it enables the creation of web based graphical user interfaces for remote operation and the development of distributed systems where a Petri net model running on a central system supervises the actions of multiple remote subsystems.

IOPT-Tools – Remote Debugger



Usage of a minimalist HTTP server, Implemented directly on the controller code.

Supporting:

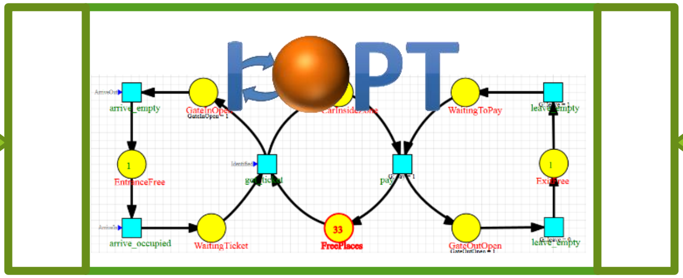
- Status monitoring functions,
- A tracing mechanism with step-by-step execution and breakpoints definition,
- The capability to remotely force the value of input and output signals, used to implement hardware-in-the-loop solution where the simulator takes full control of the physical embedded devices.

Application example

- An IOPT-based controller receiving signals from an EEG signals acquisition system and actuating movements of a quadcopter.

EPOC (by Emotiv)

Bluetooth



Controller

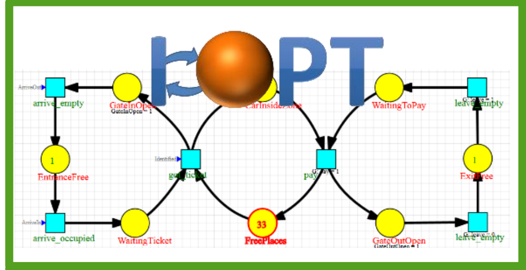
Wi-Fi

AR.Drone 2.0 (by Parrot)

Application example

Adding remote monitoring and control.

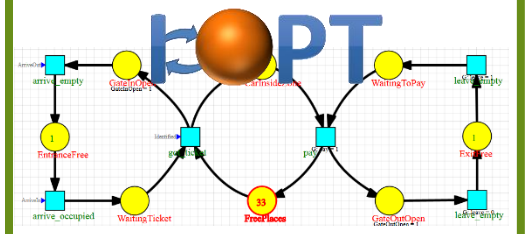
Remote debugger



web browser client

internet

dedicated http server



Controller

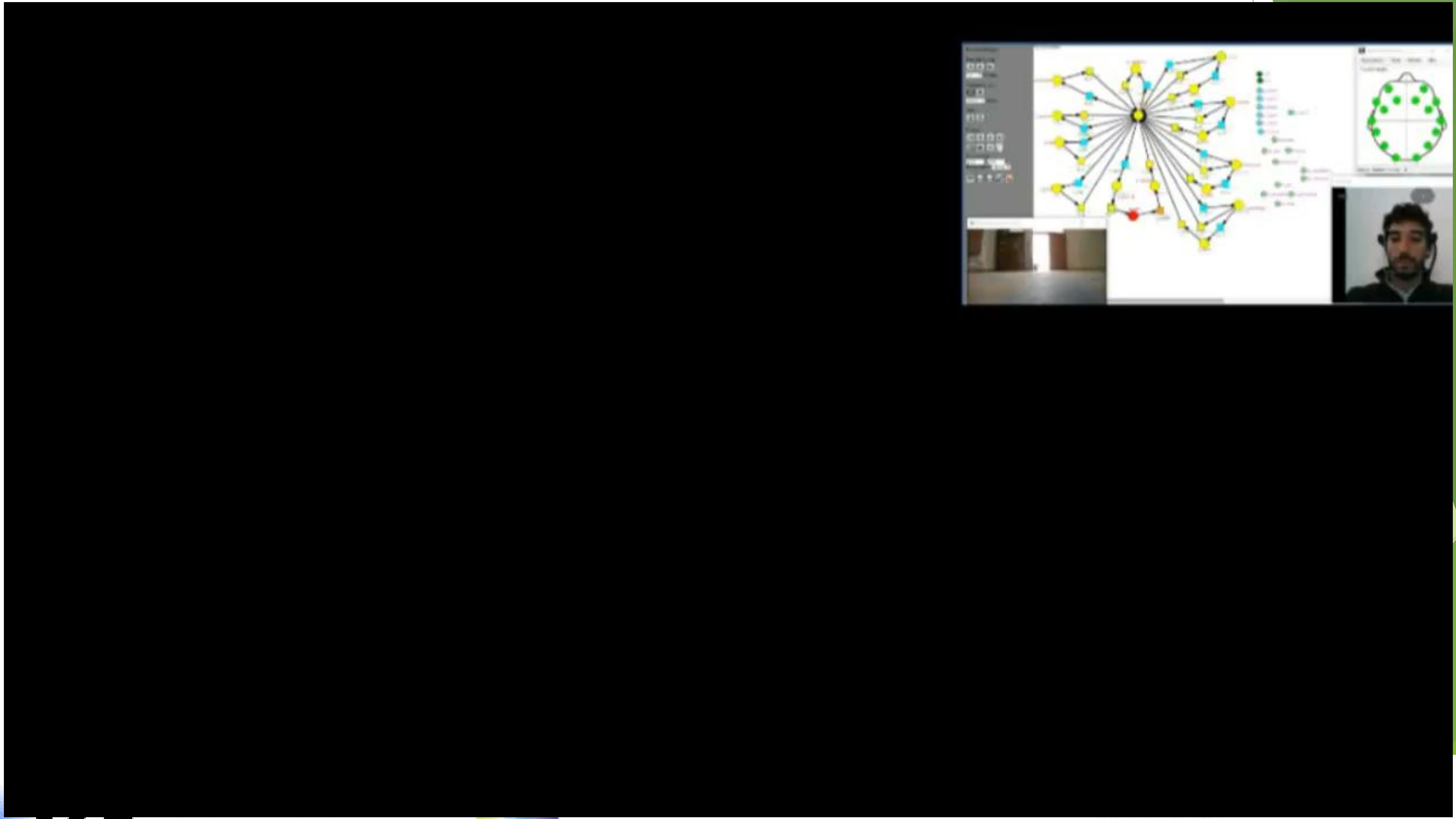
EPOC (by Emotiv)

Bluetooth

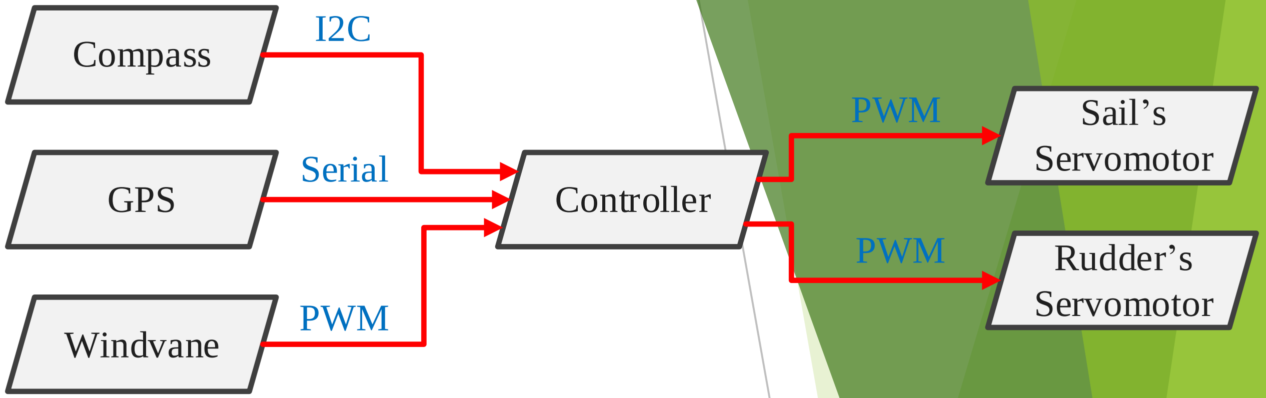
Wi-Fi

AR.Drone 2.0 (by Parrot)

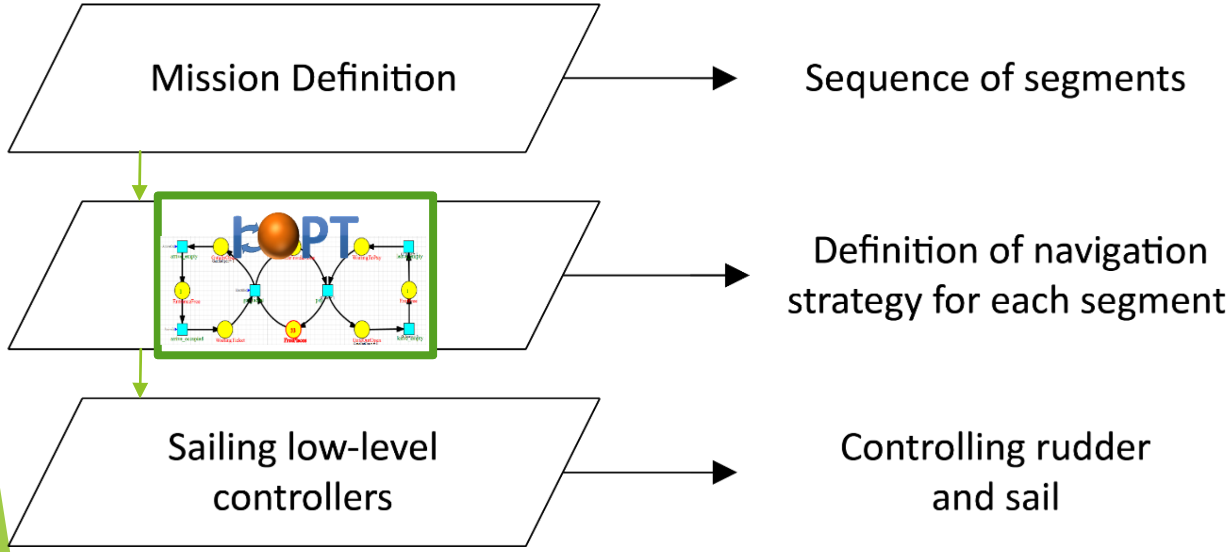
On-line control with remote monitoring



Other application example



- Autonomous navigation of a sailboat



Conclusions

- In the years ahead it is expected the appearance of millions of new Internet aware embedded devices, both for existing applications and for applications yet to be discovered.
- This way, the development tools for embedded and cyber-physical systems will need to offer rapid prototyping as well as the support for remote operation, monitoring, debug, troubleshoot and diagnose problems on malfunction devices.
- Model-based development and Petri nets have an important role to play.
- IOPT-Tools have been successful used to developed embedded controllers, targeting both software and hardware platforms.