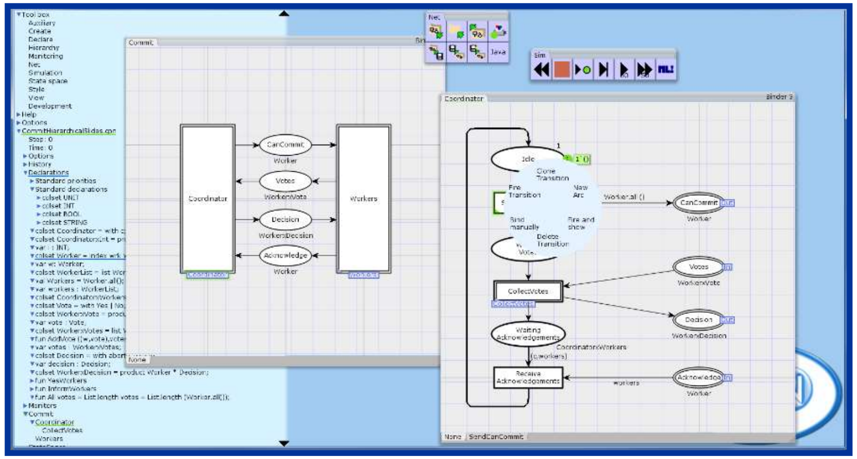
Lecture 5

Hands-on with CPN Tools

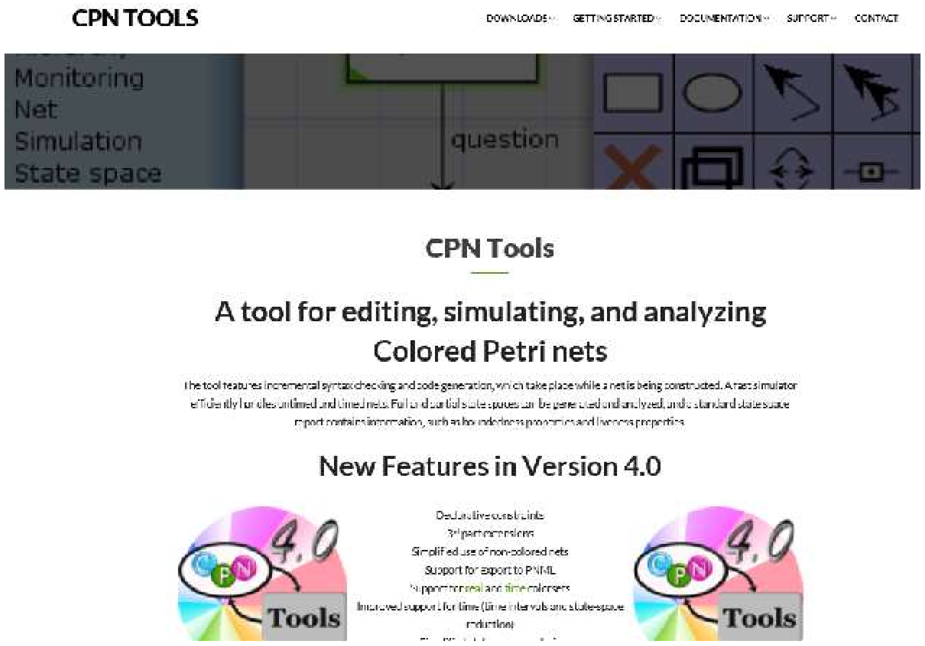


Lars M. Kristensen
Department of Computing, Mathematics, and Physics
Western Norway University of Applied Sciences
Email: lmkr@hvl.no / WWW: home.hib.no/ansatte/lmkr

Western Norway University of Applied Sciences

Installation

- CPN Tools can be downloaded and installed via www.cpntools.org



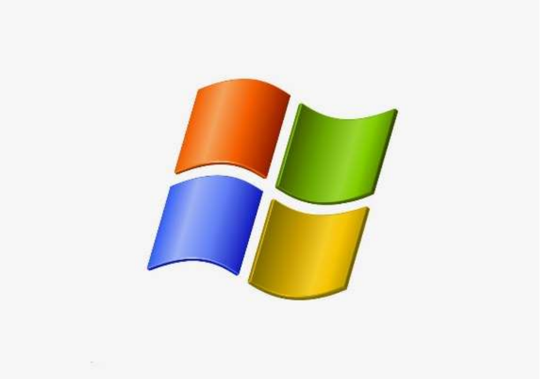
CPN Tools

A tool for editing, simulating, and analyzing Colored Petri nets

The tool features incremental syntax checking and code generation, which take place while a net is being constructed. A fast simulator efficiently handles untimed and timed nets. Full and partial state spaces can be generated and analyzed, and a standard state space report contains information, such as boundedness properties and liveness properties.

New Features in Version 4.0

Declarative constraints
3rd part extensions
Simplified use of non-colored nets
Support for export to PNML
Support for real and time colorsets
Improved support for time (time intervals and state-space reduction)

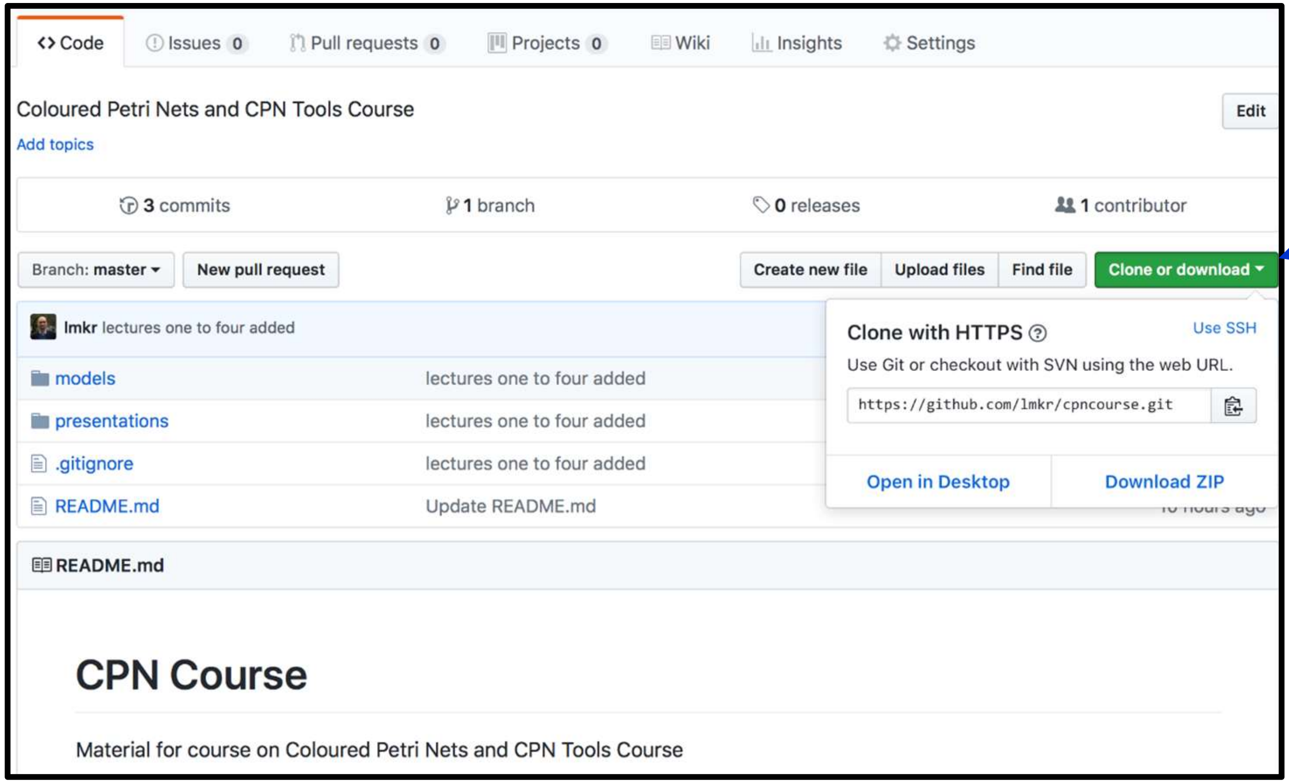


Running on Mac OS / Linux via a virtual machine or emulator.

- Some installations of windows required the application to be run as administrator.

Material

- **Models and presentations are available via the github repository at** https://github.com/lmkr/cpncourse



Coloured Petri Nets and CPN Tools Course

3 commits | 1 branch | 0 releases | 1 contributor

lmkr lectures one to four added

models	lectures one to four added	
presentations	lectures one to four added	
.gitignore	lectures one to four added	
README.md	Update README.md	10 hours ago

Clone with HTTPS

Use Git or checkout with SVN using the web URL.

```
https://github.com/lmkr/cpncourse.git
```

README.md

CPN Course

Material for course on Coloured Petri Nets and CPN Tools Course

Clone the git-repository or download as a zip-file

Hands-on Overview

- **Hands-on 1: Navigate and simulate CPN models**
- **Hands-on 2: Editing of CPN models**
- **Hands-on 3: Building a controller PT-net model**
- **Hands-on 4: Building a controller CPN model**

Hands-on 1: Simulation

- The main aim is to become familiar with how to navigate and simulate CPN models
- Open the two-phase-commit protocol CPN model located in the hands-on folder.
- Use the tools in the Simulation tool palette to conduct interactive and automatic simulations.
- Investigate scenarios where some workers vote No and a scenario where all workers vote Yes.

Hands-on 2: Editing

- **The main aim is to become familiar with how to edit and modify CPN models**
- **Modify the two-phase commit protocol CPN model from hands-on 1 such that**
  1. The coordinator and the workers no longer return to their initial idle state upon completion of the protocol.
  2. The coordinator terminates in a place CoordinatorCompleted and the workers terminate in a place WorkerCompleted
  3. The coordinator puts an Abort-token or a Commit-token on a new place Result to indicate the result of the transaction
  4. Each worker records its vote locally by putting a tuple-token consisting of the worker and the vote in a new place Votes
- **Use simulation to validate the revised model.**

Case Study: Controller

- **Consider a concurrent system consisting of a controller communicating over a bus with**
  - An actuator – a motor for moving a piston and which can be moving up, moving down or be stopped.
  - Two sensors – sends a signal if an UPPER limit / LOWER limit for the piston has been reached.
- **Model a controller that will move the piston up and down between the two limits.**
- **The system should start when it receives a START command and stop on a STOP command.**

Hands-on 3: PT-net Controller

- **Use Place/Transition nets to model (parts) of the controller**
- **The model should consists of**
  - An environment / plant part modelling the motor, the piston, and the two sensors
  - One part modelling the application logic of the controller
  - One part (places) for communication between the plant and the controller
  - One part for receiving the start and stop commands
- **For now – assume that the motor can be in three states STOPPED, MOVINGUP, MOVINGDOWN (we ignore the exact position)**

Hands-on 4: CPN Controller

- **Use CPNs to model the controller**
  - Model the communication across the bus as a single place with an enumeration colour set.
  - Model the state of the motor using an enumeration colour set consisting of the values MOVINGUP, MOVINGDOWN, STOPPED.
- **Split the model into modules**
  - One module for the motor and sensors
  - One module for the system control (start and stop)
  - One module for the controller logic