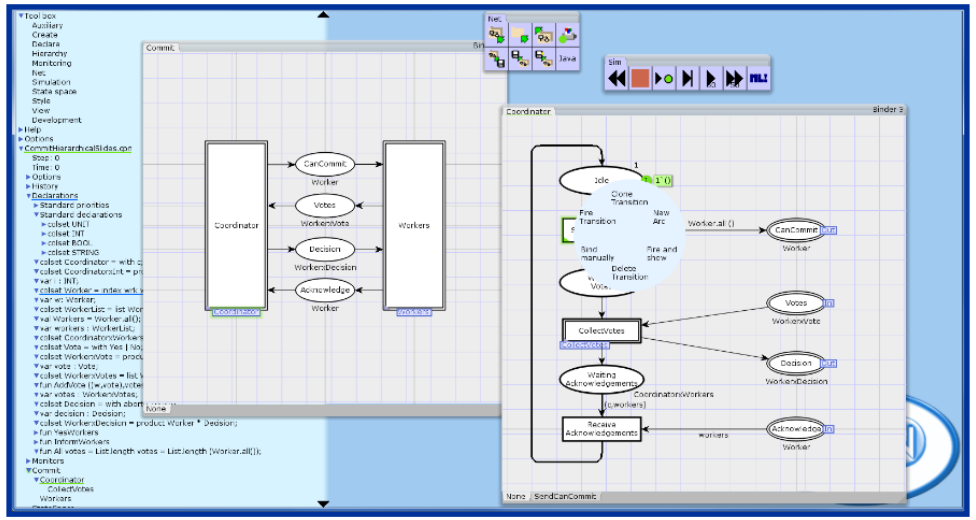
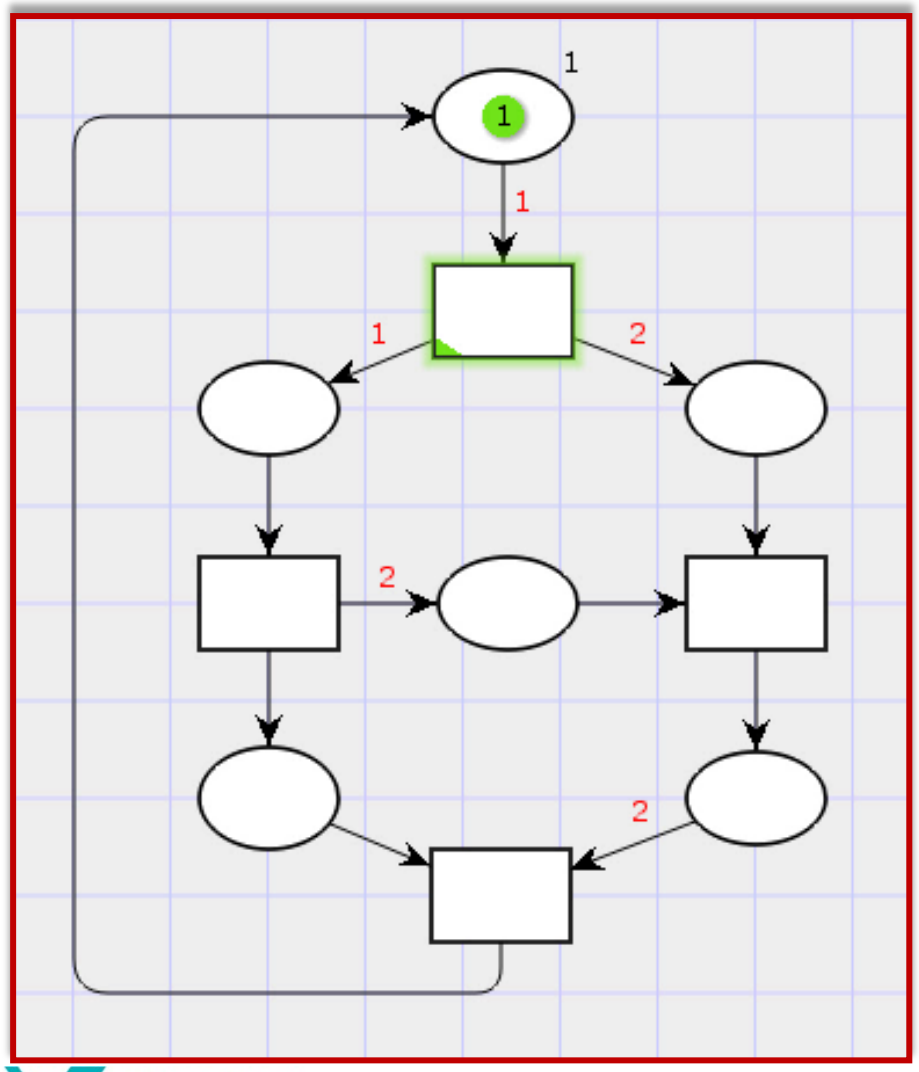
Theory and Tool | Part 1b

Background on Coloured Petri Nets: From Place/Transition Nets to Coloured Petri Nets



Lars Michael Kristensen
Department of Computer Science, Electrical Engineering, and Mathematical Sciences
Western Norway University of Applied Sciences
Email: lmkr@hvl.no

Quick Recap: Petri Net Concepts



State modelling

- **Places** (ellipses) that may hold **tokens**
- **Marking (state)**: distribution of **tokens** on the places
- **Initial marking**: initial state

Event (action) modelling

- **Transitions** (rectangles)
- **Directed arcs:** connecting places and transitions
- **Arc weights:** specifying tokens to be added/removed by transitions

Execution (token game)

- **Current marking**
- **Transition enabling**
- **Transition ocurrence**

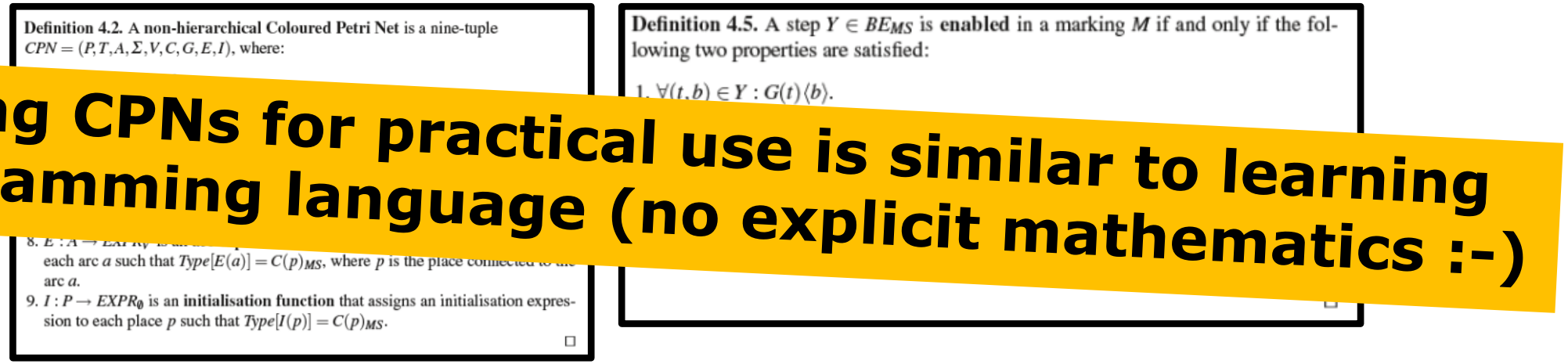
High-level Petri Nets

- Petri Nets are divided into low-level and high-level Petri Nets
  - Low-level Petri Nets (such as Place/Transition Nets) are primarily suited as a theoretical model for concurrency, but are also applied for modelling and verification of hardware systems
  - High-level Petri Nets (such as CP-nets and Predicate/Transition Nets) are aimed at practical use and software systems, in particular because they allow for construction of compact and parameterised models
- High-level Petri Nets is an ISO/IEC standard*
  - The CPN modelling language and the supporting CPN Tools conform to this standard

* https://www.iso.org/standard/38225.html

CPN models are formal

- **The CPN modelling language has a mathematical definition of both its syntax and semantics**
- **The formal representation is important**
  - Would have been impossible to develop a sound and powerful modelling language without it
  - Provides the foundation for the definition of the behavioural properties and for the formal analysis and verification methods



Definition 4.2. A **non-hierarchical Coloured Petri Net** is a nine-tuple $CPN = (P, T, A, \Sigma, V, C, G, E, I)$, where:

each arc a such that $Type[E(a)] = C(p)_{MS}$, where p is the place connected to the arc a.

9. $I : P \rightarrow EXPR_{\emptyset}$ is an **initialisation function** that assigns an initialisation expression to each place p such that $Type[I(p)] = C(p)_{MS}$.

□

Definition 4.5. A step $Y \in BE_{MS}$ is **enabled** in a marking M if and only if the following two properties are satisfied:

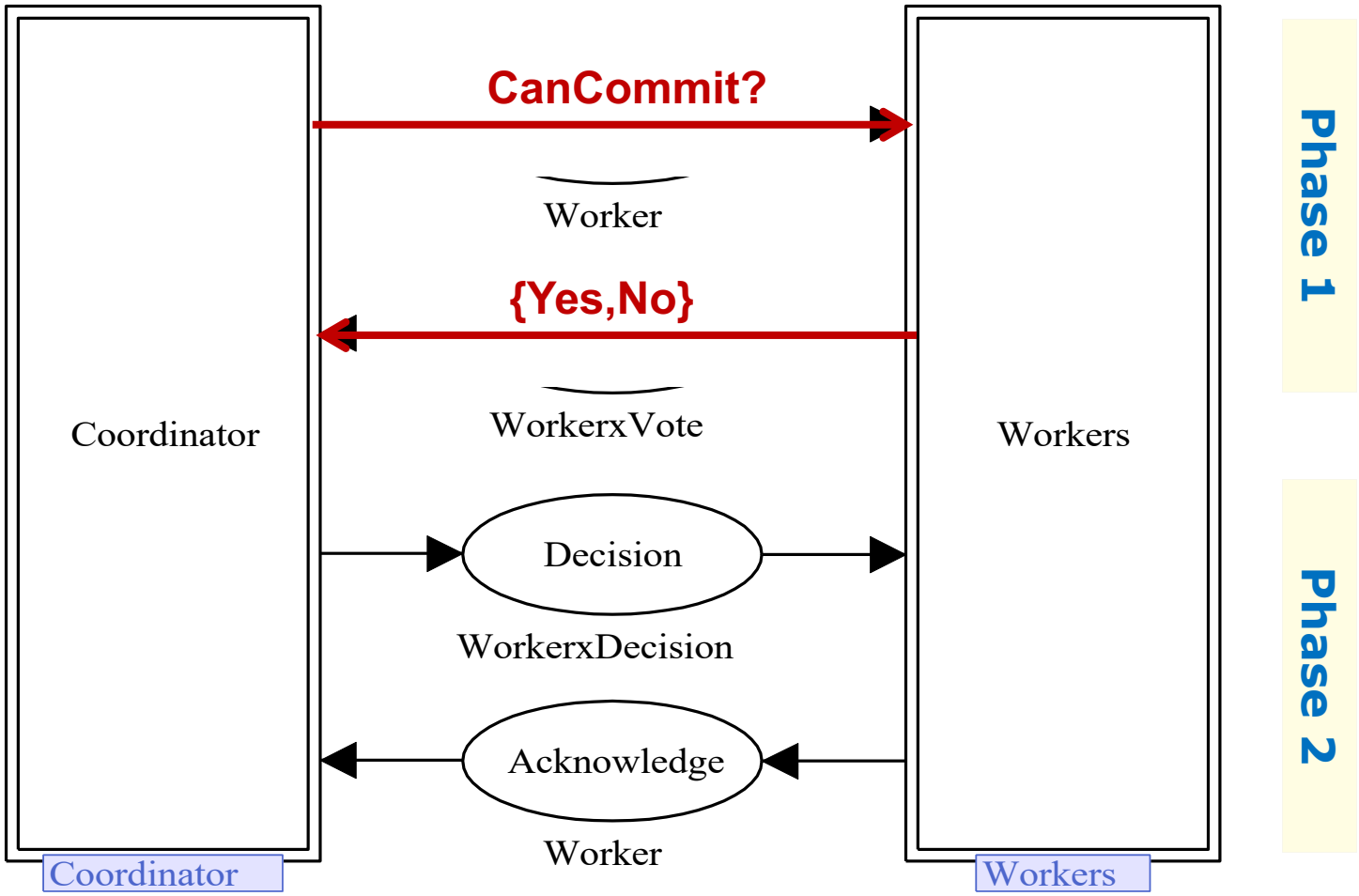
1. $\forall (t, b) \in Y : G(t)\langle b \rangle$.

□

Learning CPNs for practical use is similar to learning a programming language (no explicit mathematics :-)

Two-phase commit protocol

- **How to model phase 1 with PT-nets?**



CPN Tools demo

`part1b-background-tpc-empty.cpn | part1b-background-tpc-ptnet.cpn`

- **Construction, editing and simulation of basic Petri Net models in CPN Tools**
- **First part of the two-phase commit protocol using Place/Transition Nets**
  - How to model send and receive CanCommit with one worker?
  - How to model Yes/No votes?
  - How to model multiple workers?

Why do we need CPNs ?

- **A main limitation of Place/Transition Nets is scalability to large (real) concurrent software systems**
  - Does not support parametric systems in an elegant way
  - Modelling of data is inconvenient
  - Does not allow models to be split into modules
- **CPNs include the basic syntactical and semantical concepts of Place/Transition Nets**
  - The black/anonymous PT-net tokens are represented using the `UNIT` type and the unit value `()`
- **CPNs also provides additional language constructs originating from Place/Transition Nets**
  - Inhibitor arcs and reset arcs
  - Transition priorities