TU Clausthal

Institute for Software and Systems Engineering

Requirement Engineering

Lecture 9: Requirements Documentation

Formal Requirements Specification

Prof. Dr. Benjamin Leiding
M.Sc. Anant Sujatanagarjuna

Reminder

Bonus Task

- Submission Deadline: 25.01.2023 – 1:59 PM
- Submission Location: Moodle

General Requirements Engineering Process

Overview

<table>
<tr><td colspan="6">Requirements Engineering</td></tr>
<tr><td colspan="4">Requirements Analysis</td><td colspan="2">Requirements Management</td></tr>
<tr><td>Elicitation</td><td>Negotiation</td><td>Documentation</td><td>Validation</td><td>Change Management</td><td>Tracing</td></tr>
</table>

Lecture 9: Requirements Documentation

Content

1. Formal Specification Techniques

FORMAL REQUIREMENTS SPECIFICATION

Lecture 4: Requirements Documentation

Content

1. Model-based Requirements Documentation Techniques
2. Formal Specification Techniques
    1. **Coloured Petri Nets (CPNs)**
    2. AOM → CPN Mapping
    3. Four Variable Model
    4. NRL / SCR

Formal Specification Techniques

Coloured Petri Nets – Motivation: Why CPNs?

- They can model concurrency and communication in complex systems very well.
- They combine concepts from Petri Nets and programming languages (CPN ML).
- They are Formal (syntatically and mathematically defined)
- They are *executable*.
- They can be derived from other models, such as AOM.
- **Allows for easy manual or automatic system verification and evaluation.**

Formal Specification Techniques

Coloured Petri Nets – Motivation: Why CPNs?

Activity	Trigger(s)	Precondition(s)	Postcondition(s)
Reserve Charging Port(CP)	Driver wants to charge EV	Driver has bought energy, CP is free, EV is ready to charge	Driver has reserved CP
Charge EV	〃	Driver has reserved CP	Driver has charged EV
Agree on charging speed	〃	Driver has reserved CP, Max CP speed ≥ Min EV speed, Max EV speed ≥ Min CP speed	Charging speed is agreed upon
Begin Charging	〃	Charging speed is agreed upon	EV has begun charging
End Charging	None	EV has competed charging	Driver has charged EV, CP is free

Formal Specification Techniques

Coloured Petri Nets – Motivation: Why CPNs?

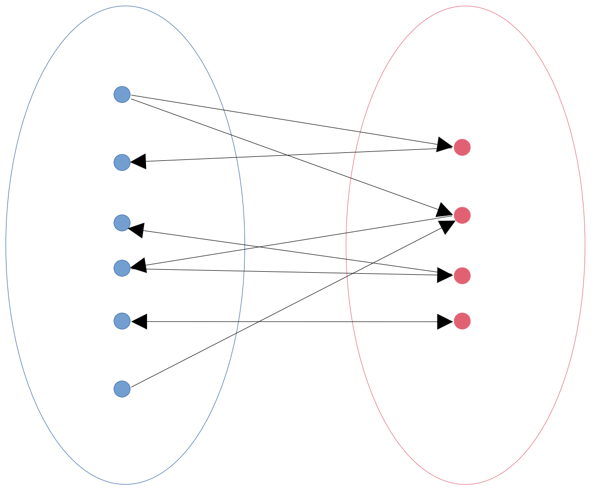
Activity	Trigger(s)	Precondition(s)	Postcondition(s)
Reserve Charging Port(CP)	Driver wants to charge EV	Driver has bought energy, CP is free, EV is ready to charge	Driver has reserved CP
Charge EV	〃	Driver has reserved CP	Driver has charged EV
Agree on charging speed	〃	Driver has reserved CP, Max CP speed ≥ Min EV speed, Max EV speed ≥ Min CP speed	Charging speed is agreed upon
Begin Charging	〃	Charging speed is agreed upon	EV has begun charging
End Charging	None	EV has competed charging	Driver has charged EV, CP is free

- How do we know that these conditions are enough?
- Can we model these conditions and simulate, for example; can the system handle 10 EVs and 3 charging stations, or does the system ever get into an infititnte blocking state?

Formal Specification Techniques

Coloured Petri Nets – What are CPNs?

- Directed bipartite graphs:
- Bipartite graphs divide the vertices of the graph into:
- two disjoint and independent sets.
- Edges in the graph ONLY connect vertices from one set to the other.



Formal Specification Techniques

Coloured Petri Nets – What are CPNs?

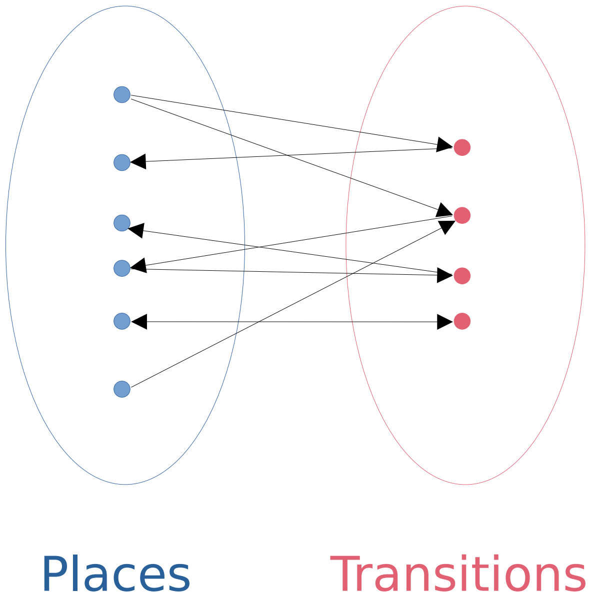
Directed bipartite graphs:

- CPNs divide the vertices of the graph into:

Places

Transitions

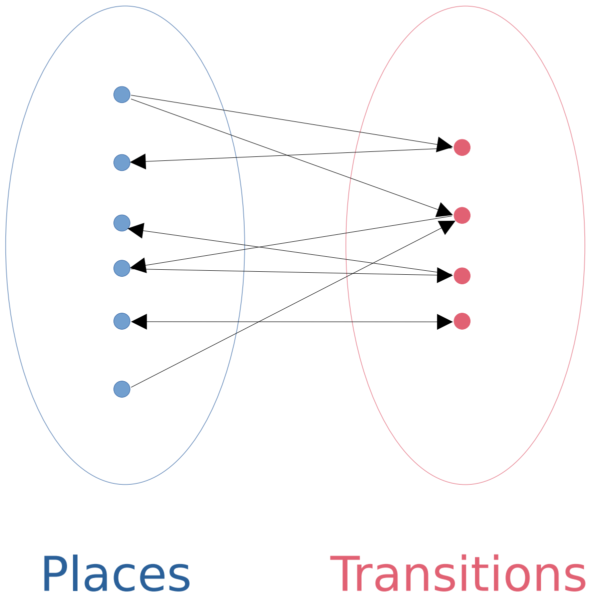
- *Arcs* in the graph **ONLY** connect Places with Transitions.



Formal Specification Techniques

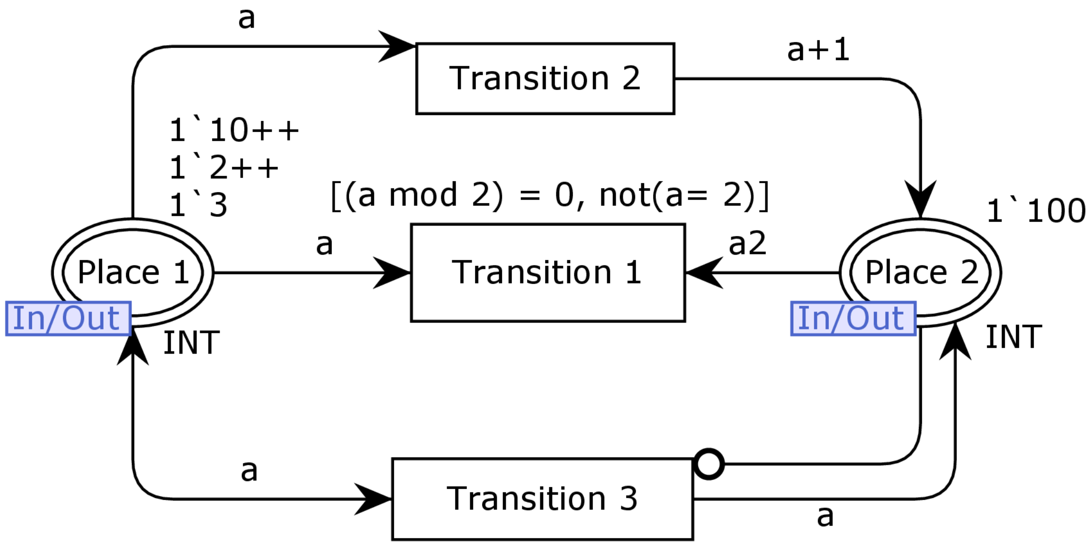
Coloured Petri Nets – What are CPNs?

- CPNs simulate the *change of state* in the system with the exchange of Tokens.
- Tokens reside in, and flow between Places though transitions.
- Tokens never *stay* in Transitions.



Formal Specification Techniques
Coloured Petri Nets – Core Concepts : ColorSets, Tokens and Variables

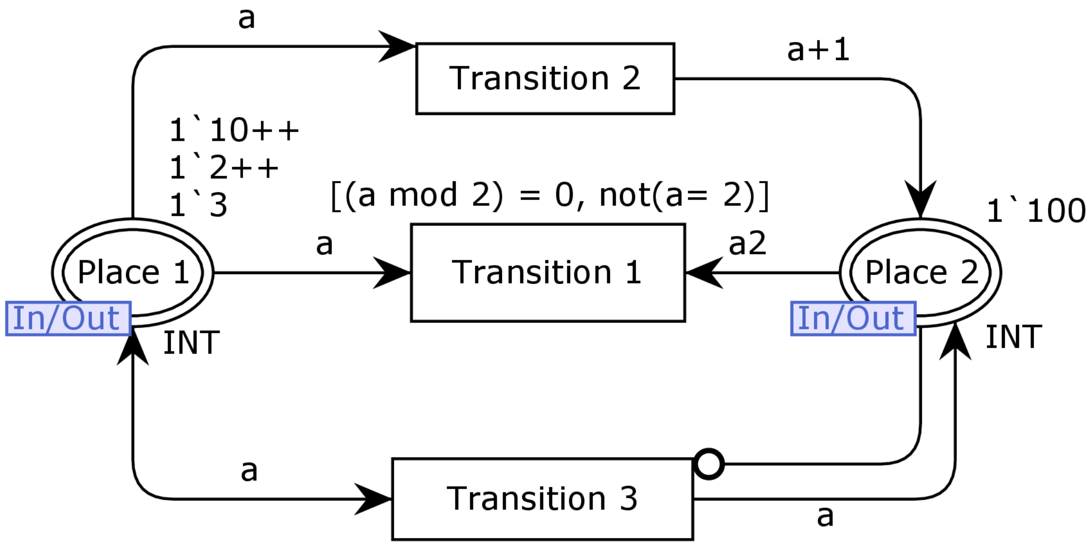
- Places can hold tokens of only one *type*(= ColorSet).
- Tokens are *instances* of ColorSets.
- Variables are used to distinguish tokens while in transit.




```
var a: INT;
var a2:
INT;
```

Formal Specification Techniques
Coloured Petri Nets – Core Concepts : Markings, Arc Inscriptions and Guards

- Markings specify the tokens held by a Place.
- Arc Inscriptions are expressions that can *modify* the tokens when the transition occurs.
- Guards are a comma-separated list of conditions that must be satisfied for a Transition.

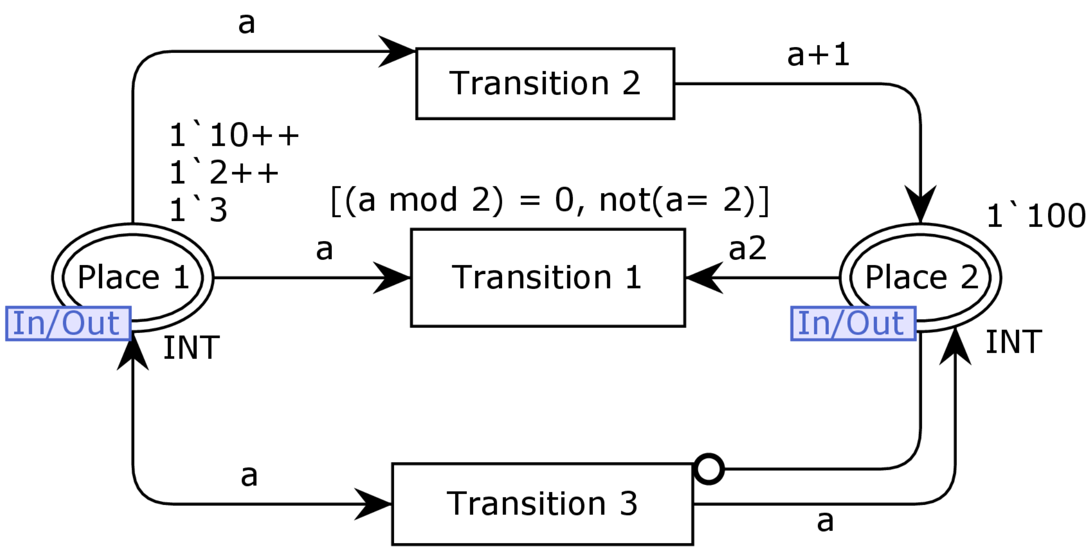



```
var a: INT;
var a2:
INT;
```

Formal Specification Techniques

Coloured Petri Nets – Core Concepts : Flow of Tokens

- A Transition is enabled if and only if:
  - All it’s incoming arcs have at least one token.
  - All guard conditions are satisfied.
  - No place that is connected with an inhibitor arc has any tokens.

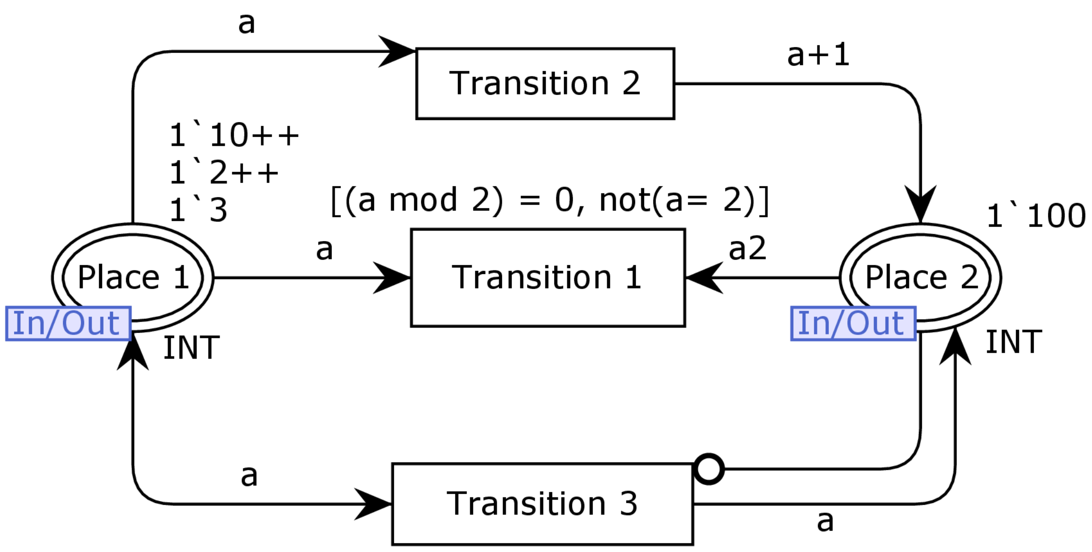



```
var a: INT;
var a2:
INT;
```

Formal Specification Techniques

Coloured Petri Nets – Core Concepts : Flow of Tokens

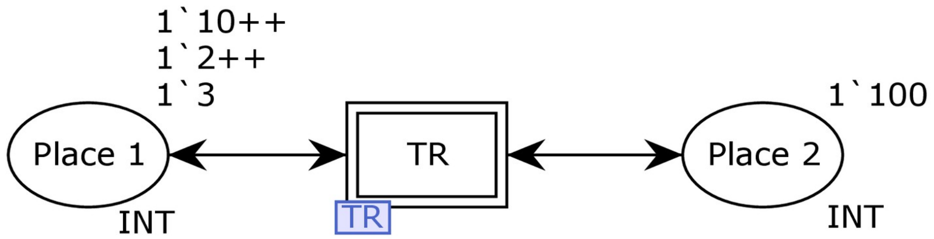
- When a Transition is fired:
  - Any one suitable token is consumed from each input place, according to outgoing arc inscriptions.
  - The tokens are transferred to the output places according to outgoing arc inscriptions.
- **IMP:** Only one transition can be fired at a time.

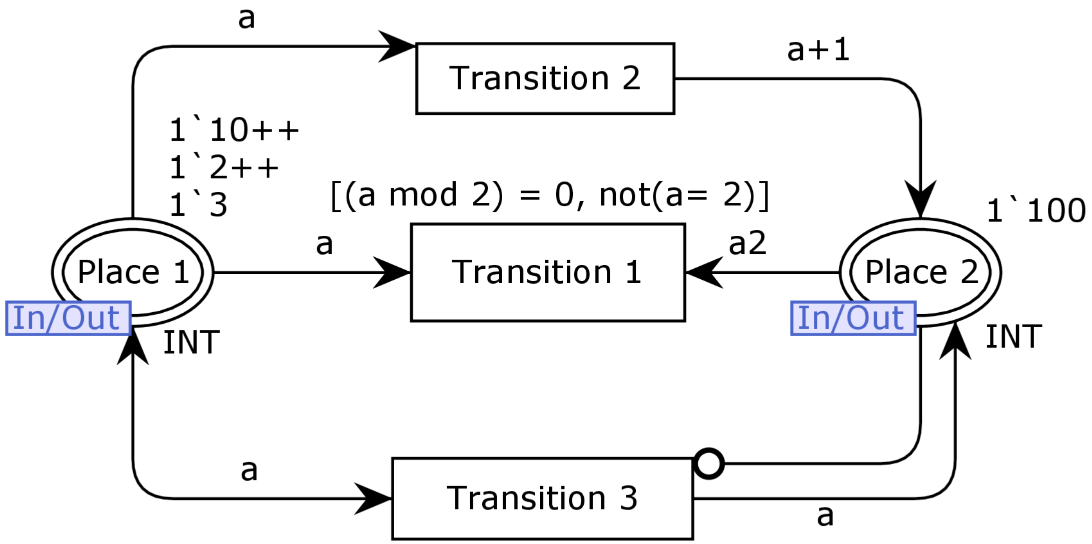



```
var a: INT;
var a2:
INT;
```

Formal Specification Techniques

Coloured Petri Nets – Core Concepts : Hierarchical CPNs






```
var a: INT;
var a2:
INT;
```

Formal Specification Techniques

Coloured Petri Nets – Evaluation using state-space simulations

Basic Idea

- Compute all the reachable states and the state changes of the CPN model
- Represent these as a directed graph where nodes represent states and arcs represent occurring events
- Pros
- From a constructed state–space, it is possible to verify various aspects of the behaviour of the system:
  - Absence of deadlocks
  - Possibility of always being able to reach a given state
  - Guaranteed delivery of a given service
- Cons
  - State-space graph size (and computation time) increases exponentially! → requires independent computation in some cases.

K. Jensen et al. (2007) – Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems. International Journal on Software Tools for Technology Transfer (9): 213-254. DOI:10.1007/s10009-007-0038-x

Formal Specification Techniques

Coloured Petri Nets – CPN Tools

“A tool for editing, simulating, and analysing Coloured Petri Nets”

cpntools.org

Formal Specification Techniques

Coloured Petri Nets – Live Example

“Simulate a program reading a file and then writing to it.”

cpntools.org

Lecture 4: Requirements Documentation

Content

1. Model-based Requirements Documentation Techniques
2. Formal Specification Techniques
    1. Coloured Petri Nets (CPNs)
    2. **AOM → CPN Mapping**
    3. Four Variable Model
    4. NRL / SCR

AOM → CPN Mapping

Motivation and Methodology

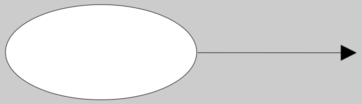
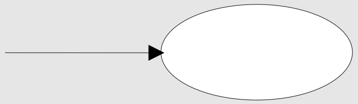
- CPN models are powerful, *BUT* can be arbitrarily complex. Where to start?
- It is important to ensure inter-model consistancy.
- Mapping AOM → CPN models leverages the advantages of both, and also creates a feedback loop using the simulation and evaluation capabilities of CPNs.

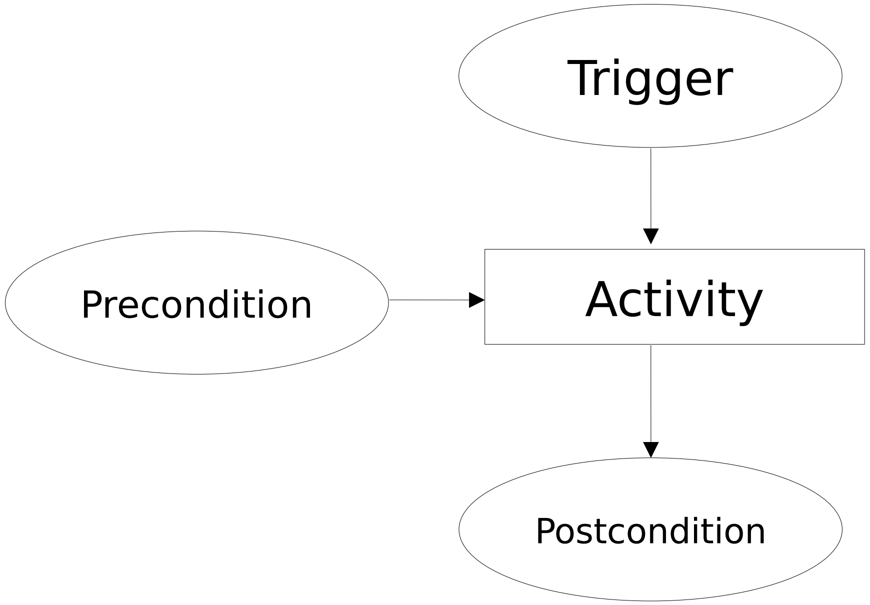
- *Heuristics for Designing and Evaluating Socio-Technical Agent-Oriented Behaviour Models with Coloured Petri Nets (2014).* Msury Mahunnah, Alex Norta, Lixin Ma, Kuldar Taveter

M. Mahunnah et al. (2014) – Heuristics for Designing and Evaluating Socio-technical Agent-Oriented Behaviour Models with Coloured Petri Nets. IEEE 38th International Computer Software and Applications Conference Workshops (2014): 438-443. https://doi.org/10.1109/COMPSACW.2014.74

AOM → CPN Mapping

Mapping Methodology

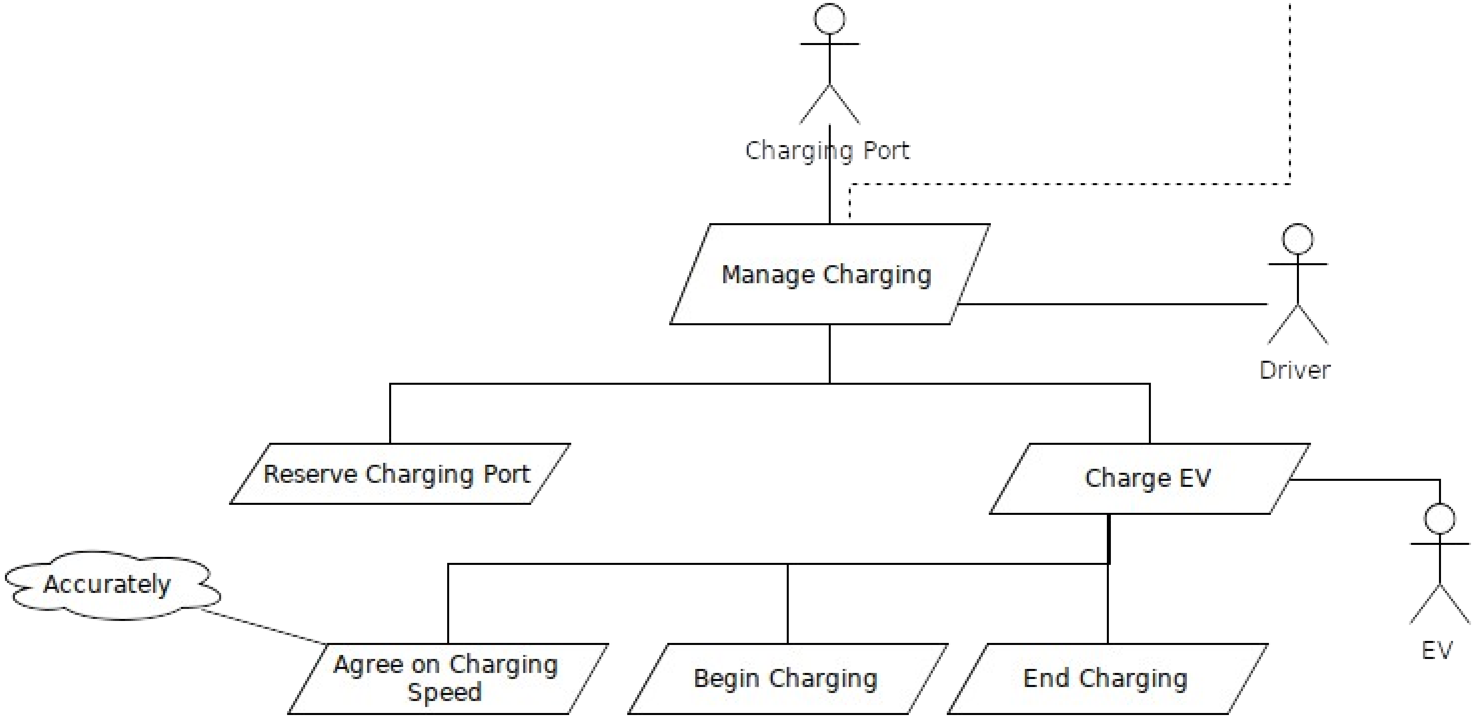
Notation	Name
	Connecting arc
	Sub-goal or activity
	Trigger/ precondition
	Postcondition
	Goal
`[<condition(s)>]`	Precondition(s)



Adapted from M. Mahunnah et al. (2014) – Heuristics for Designing and Evaluating Socio-technical Agent-Oriented Behaviour Models with Coloured Petri Nets. IEEE 38th International Computer Software and Applications Conference Workshops (2014): 438-443. https://doi.org/10.1109/COMPSACW.2014.74

AOM → CPN Mapping

Example: Automated EV Charging Station (Manage Charging)



AOM → CPN Mapping

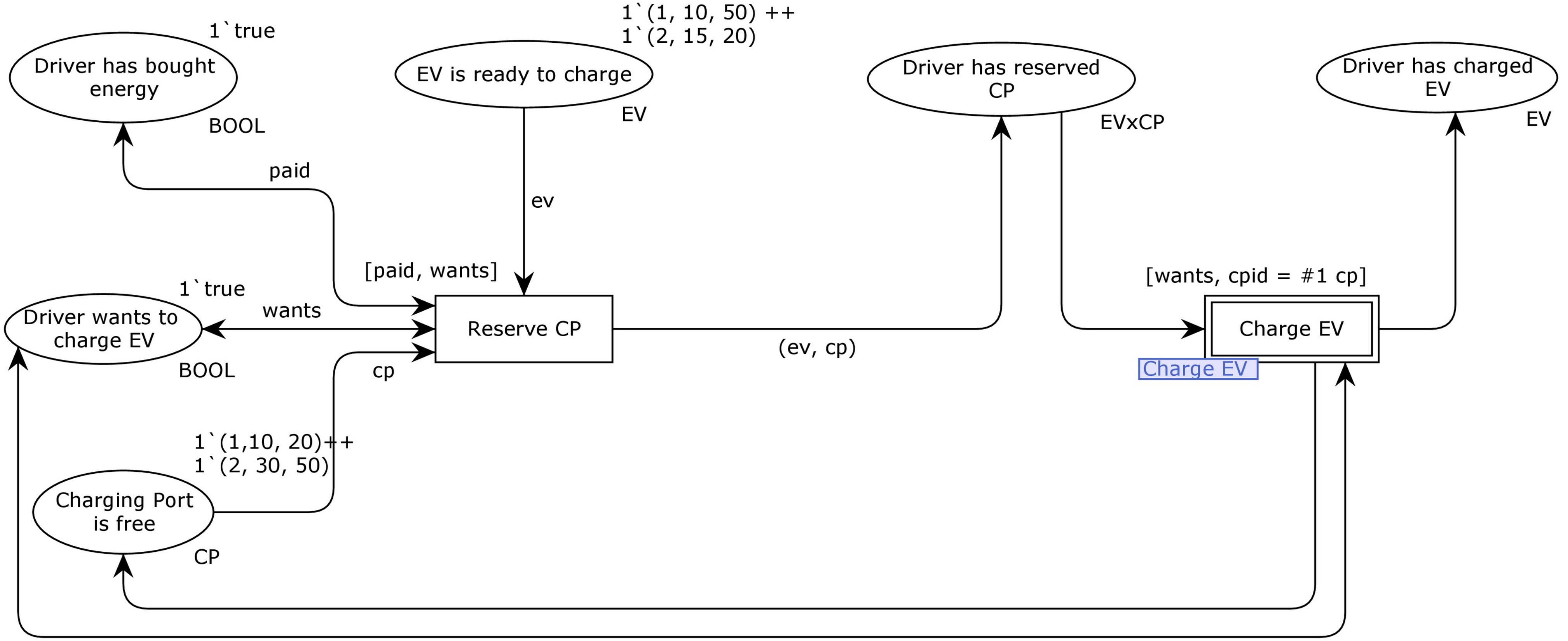
Example: Automated EV Charging Station (Manage Charging)

Activity	Trigger(s)	Precondition(s)	Postcondition(s)
Reserve Charging Port(CP)	Driver wants to charge EV	Driver has bought energy, CP is free, EV is ready to charge	Driver has reserved CP
Charge EV	〃	Driver has reserved CP	Driver has charged EV
Agree on charging speed	〃	Driver has reserved CP, Max CP speed ≥ Min EV speed, Max EV speed ≥ Min CP speed	Charging speed is agreed upon
Begin Charging	〃	Charging speed is agreed upon	EV has begun charging
End Charging	None	EV has competed charging	Driver has charged EV, CP is free

AOM → CPN Mapping

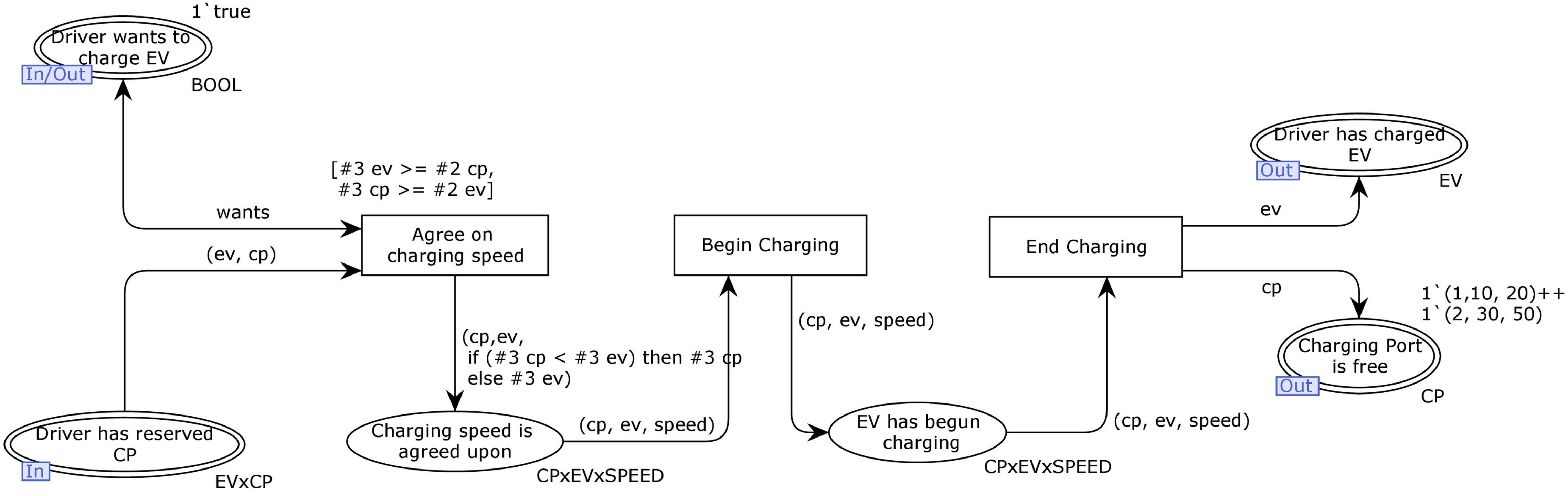
Example: Automated EV Charging Station (Manage Charging)

Here we assume some connection to the Buy Energy Sub-Goal that gives a true/false response indicating whether a driver has paid or not.



AOM → CPN Mapping

Example: Automated EV Charging Station (Charge EV)



AOM → CPN Mapping

State-space Simulations

Basic Idea

- Compute all the reachable states and the state changes of the CPN model
- Represent these as a directed graph where nodes represent states and arcs represent occurring events

Pros

- From a constructed state–space, it is possible to verify various aspects of the behaviour of the system:
  - Absence of deadlocks
  - Possibility of always being able to reach a given state
  - Guaranteed delivery of a given service

Cons

  - State-space graph size (and computation time) increases exponentially! → requires independent computation in some cases.

K. Jensen et al. (2007) – Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems. International Journal on Software Tools for Technology Transfer (9): 213-254. DOI:10.1007/s10009-007-0038-x

Lecture 4: Requirements Documentation

Content

Formal Specification Techniques

Four Variable Model – Basic Elements

- Four Variables
  - Input variables → Physical variables measured by input devices
  - Output variables → Physical variables controlled by output devices
  - Monitored environmental variables
  - Controlled environmental variables

- Relations
  - NAT, REQ, IN/OUT, SOF, SOFREQ
- Are used as basis for requirements documentation

Formal Specification Techniques

Four Variable Model – Documents

- System Requirements Document
  - Models the complete system as a black-box
  - Includes a description of the environment
  - Identifies a set of quantities and associates each one with a mathematical variable
  - Describes constraints of the environment like physical laws
  - Describes constraints related to the new system

Formal Specification Techniques

Four Variable Model – Documents

- System Design Document
  - Describes relevant properties of peripheral devices
  - Identifies input- and output registers, modeled as mathematical variables
  - Describes relation between input registers and associated environmental quantities
  - Describes relation between output registers and associated environmental quantities

Formal Specification Techniques

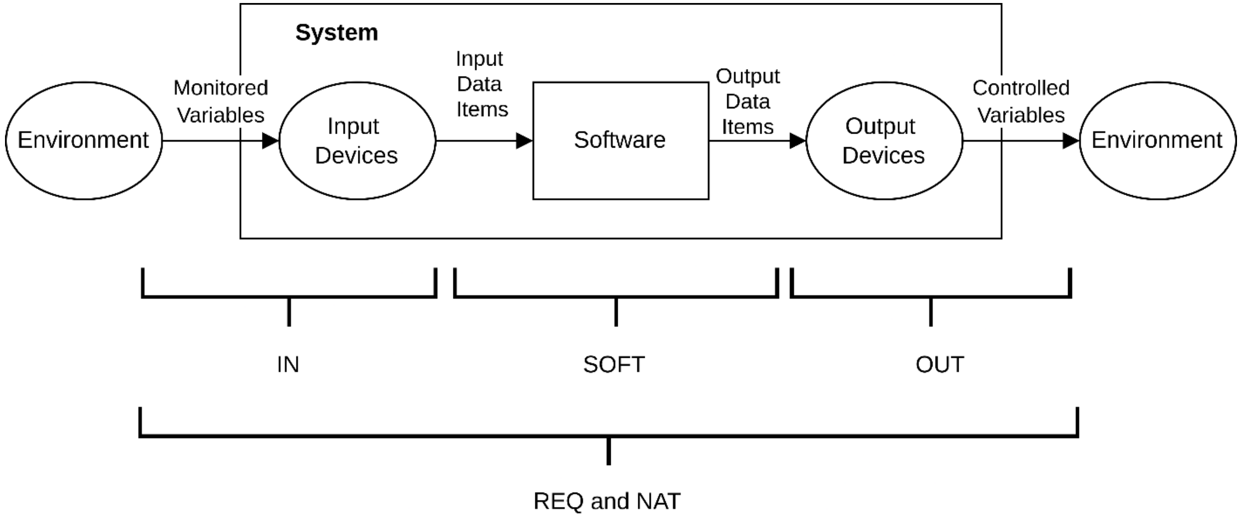
Four Variable Model – Documents

- Software Requirements Document
  - Combination of system requirements document and system design document
- Software Behavior Specification
  - Records additional design decisions
  - Provides a description of the actual software behavior

Formal Specification Techniques

Four Variable Model – Relations

- *NAT*: expresses constraints due to restrictions imposed by nature
- *REQ*: expresses the requirements of the system
- *IN, OUT*: input and output relations
- *SOF*: behavior of a particular software implementation
- *SOFREQ*: software requirements relation, all acceptable software behavior

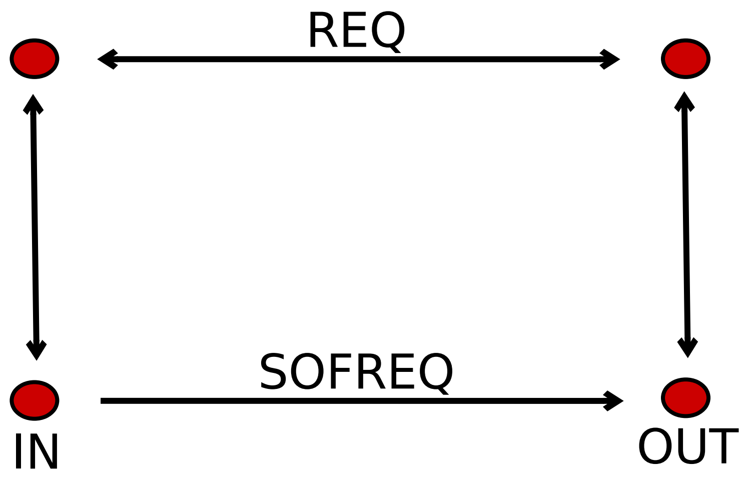


Based on C.L. Heitmeyer, R.D. Jeffords, B.G. Labaw (2013) – Automated Consistency Checking Requirements Specifications

Formal Specification Techniques

Four Variable Model – Basic Relations

- Basic Relations
- $SOFREQ \subseteq SOF$
- $IN, OUT, REQ \xrightarrow[\text{w.r.t. NAT}]{} SOFREQ$
- $OUT \circ SOFREQ \circ IN = REQ$



Formal Specification Techniques

Four Variable Model – Variables

- Monitored variables
  - Mathematical function whose domain consists of real numbers
  - Environmental function $m^t : \mathbb{R} \rightarrow Value$
  - Value at time *s*: $m^t(s)$
  - Vector of monitored variables: $m^t = (m^t_1, m^t_2, \ldots, m^t_p)$

- Controlled variables
  - Mathematical function whose domain consists of real numbers
  - Environmental function $c^t : \mathbb{R} \rightarrow Value$
  - Vector of monitored variables: $c^t = (c^t_1, c^t_2, \ldots, c^t_q)$

Formal Specification Techniques

Four Variable Model – Relations

- REQ
  - Expresses the requirements of the system
  - *domain(REQ)*: set of vectors containing exactly the instances of m^t allowed by the environmental constraints
  - *range(REQ)*: set of vectors containing only those instances of c^t considered permissible
  - $(m^t, c^t) \in REQ$ if environmental constraints allow the controlled variables to take the values given by c^t, if the values of the monitored variables are given by m^t
  - *REQ* may **tolerate ’small’ errors** in the values of controlled variables

Formal Specification Techniques

Four Variable Model – Relations

- IN
  - Describes the behavior of the input devices
  - Is a relation due to imprecision in the measurement

- OUT
  - Describes the behavior of output devices
  - Is a relation due to device imperfections

Formal Specification Techniques

Four Variable Model – Relations

- SOF
  - Describes behavior of a particular software implementation
  - *domain(SOF)*: set of vectors containing all possible instances of i^t
  - *range(SOF)*: set of vectors containing all possible instances of o^t
  - $(i^t, o^t) \in SOF$ iff the software could produce values described by o^t
  - **Verification condition:** *SOF* implements a subset of *REQ*
    - $SOF \subseteq IN^{-1} \bigcirc REQ \bigcirc OUT^{-1}$
    - (where $\bigcirc$ denotes the composition of binary relations)
    - Often: $SOF = IN^{-1} \bigcirc REQ \bigcirc OUT^{-1}$

Formal Specification Techniques

Four Variable Model – Relations

- SOFREQ
  - Characterizes all acceptable software behavior
  - SOFREQ corresponds on the software level to REQ on the system level
  - It consists of all tuples (i^t, o^t) satisfying for all m^t, c^t :
    - $(IN(m^t, i^t) \wedge OUT(o^t, c^t) \wedge NAT(m^t, c^t)) \rightarrow REQ(m^t, c^t)$

Formal Specification Techniques

Four Variable Model – Feasibility and Acceptability

- Feasibility of REQ w.r.t. NAT
  - Requirements should specify behavior for all cases that can arise
  - $domain(NAT) \subseteq domain(REQ)$
  - $domain(NAT \cap REQ) = domain(NAT) \cap domain(REQ) = domain(NAT)$
- Acceptability
  - Describes the behavior that the software must exhibit to be acceptable for use and for the requirements to be satisfied
  - $NAT \cap (IN \circ SOF \circ OUT) \subseteq REQ$

Lecture 4: Requirements Documentation

Content

Formal Specification Techniques

NRL / SCR – Motivation

- Tabular notations → precise and compact notation for requirements
- Applications → avionics systems, controlling nuclear power plants, telephone networks
- Can be used for automatic analysis
- History
  - 1978: flight program of the A-7 aircraft
  - Real-time, embedded system
  - Organizations: Bell Laboratories, Ontario Hydro, Naval Research Laboratory, Lockheed
  - Applications: submarine communication system, shutdown system for the Darlington nuclear power plant, flight program for Lockheed’s C130J aircraft.

Formal Specification Techniques

NRL / SCR – SCR Formal Model

- Behavior
  - Non-deterministic system environment
  - Deterministic system behavior
- 4VM → Monitored and controlled variables, NAT, REQ
- Model: System is represented as labeled transition system (LTS), responds to each monitored event
- Synchronous behavior: The system completely processes one set of inputs before processing the next state
- One input assumption: At most one monitored variable is allowed to change from one state to the next

Formal Specification Techniques / NRL / SCR

NRL / SCR – SCR Formal Model

- Auxiliary variables: (specification of *REQ*)
  - Mode classes: values are called modes
  - Modes: equivalence class of system states
  - Terms: internal variables

- System: labeled transition system $(S, I, E^m, \rightarrow)$ consisting of
  - States S, initial states I
  - Labels E^m (set of monitored events)
  - Transition relation $\rightarrow$ realized as a function that maps a monitored event $e \in E^m$ and the current state $s \in S$ to the next state $s' \in S$

Formal Specification Techniques / NRL / SCR

NRL / SCR – SCR Formal Model

- Condition → Predicate defined on a single system state
- Event → Predicate defined on two system states
- Occurrence → An event occurs if a condition changes
- *@T(c)*: *c* becomes *true*
- *@F(c)*: *c* becomes *false*
- Conditioned event → $@T(c_1)\ WHEN\ c_2$

Special form: $@T(c)\ WHEN\ d$ *iff* $\neg c \wedge c' \wedge d$

Formal Specification Techniques / NRL / SCR

NRL / SCR – SCR Tables

- Mode transition table:
  - Associates a source mode and an event with a destination mode
  - Each table should describe a **total function**
  - Should exhibit disjointness and coverage properties
- Event table:
  - An event table defines how a term or controlled variable changes in response to input events
  - Defines a (partial) function from modes and events to variable values
- Condition table:
  - A condition table defines the value of a term or controlled variable under every possible condition
  - Defines a total function from modes and conditions to variable values

Formal Specification Techniques / NRL / SCR

NRL / SCR – SCR Tables

Current Mode	Powered on	Too Cold	Temp OK	Too Hot	New Mode
Off	@T @T @T	- t -	t - -	- - t	**Inactive** **Heat** **AC**
Inactive	@F - -	- @T -	- - -	- - @	**Off** **Heat** **AC**
Heat	@F -	- -	- @T	- -	**Off** **Inactive**
AC	@F -	- -	- @T	- -	**Off** **Inactive**

Formal Specification Techniques / NRL / SCR

NRL / SCR – SCR Tables

Event table

Modes		
NoFailure	@T(INMODE)	Never
ACFailure, HeatFailure	Never	@T(INMODE)
Warning light =	**Off**	**On**

Condition table

Modes	Events	
NoFailure	true	false
ACFailure	temp > temp0	temp <= temp0
HeatFailure	false	waterlevel = low
Warning light =	**Off**	**On**

Formal Specification Techniques / NRL / SCR

NRL / SCR – Tool Support: SCR Toolset

- Specification editor for creating the tabular specification
- Simulator for validation
- Dependency graph browser for understanding the relationship between different parts of the specification
- Consistency checker for analyzing syntax, type correctness, determinism, case coverage, ...
- Model checker for checking linear temporal properties of finite state systems
- Theorem prover for checking properties deductively, avoiding the state explosion problem, often user interaction necessary

Formal Specification Techniques / NRL / SCR

NRL / SCR – SCR Toolset: Construction of Requirements Specifications

- Specification
  - Specification editor
  - Function tables: define the value of dependent variables
  - Dictionaries: variable declarations, environmental assumptions, type definitions

- Analysis
  - Consistency checker, property checker, dependency graph browser
  - Well-formedness errors
  - Disjointness and coverage: no nondeterminism, no missing cases

Formal Specification Techniques / NRL / SCR

NRL / SCR – SCR Toolset: Construction of Requirements Specifications

- Validation
  - Check inconsistencies between the intended and the specified behavior
  - Simulator: the user can run scenarios (sequences of monitored events)
  - Invariant generator: the user can generate state invariants

- Application analysis
  - Check application properties
  - Model checker
  - Property checker
  - Theorem prover (TAME, PVS)

SUMMARY

Summary

- Formal specification techniques
  - Coloured Petri Nets (incl. mapping from AOM), four variable model, NRL/SCR
  - More exist and might be better suited for your project

Questions?