

Ontology-Driven Natural Language Requirement Templates for Model Checking I&C Functions

EHPG 2013

Teemu Tommila, Antti Pakonen, Janne Valkonen
VTT Technical Research Centre of Finland

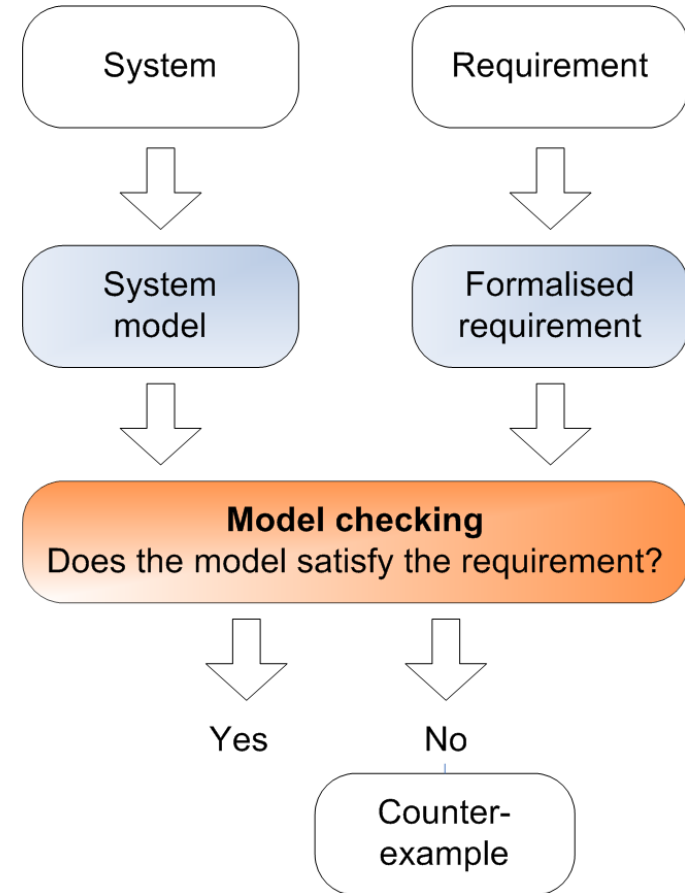
Controlled Natural Language (CNL) in requirements definition

- Motivation
 - Poor quality of textual requirements (vagueness, complexity of sentences, inexact terminology)
 - Need for formalised requirements in advanced tools, e.g. model checking
 - Existing research and technologies in the fields of CNL and semantic web

- The approach
 - Develop recommended natural language templates together with corresponding formal representations
 - Define a vocabulary (ontology) for the domain
 - Develop a concept of a “syntax and ontology controlled requirements editor”

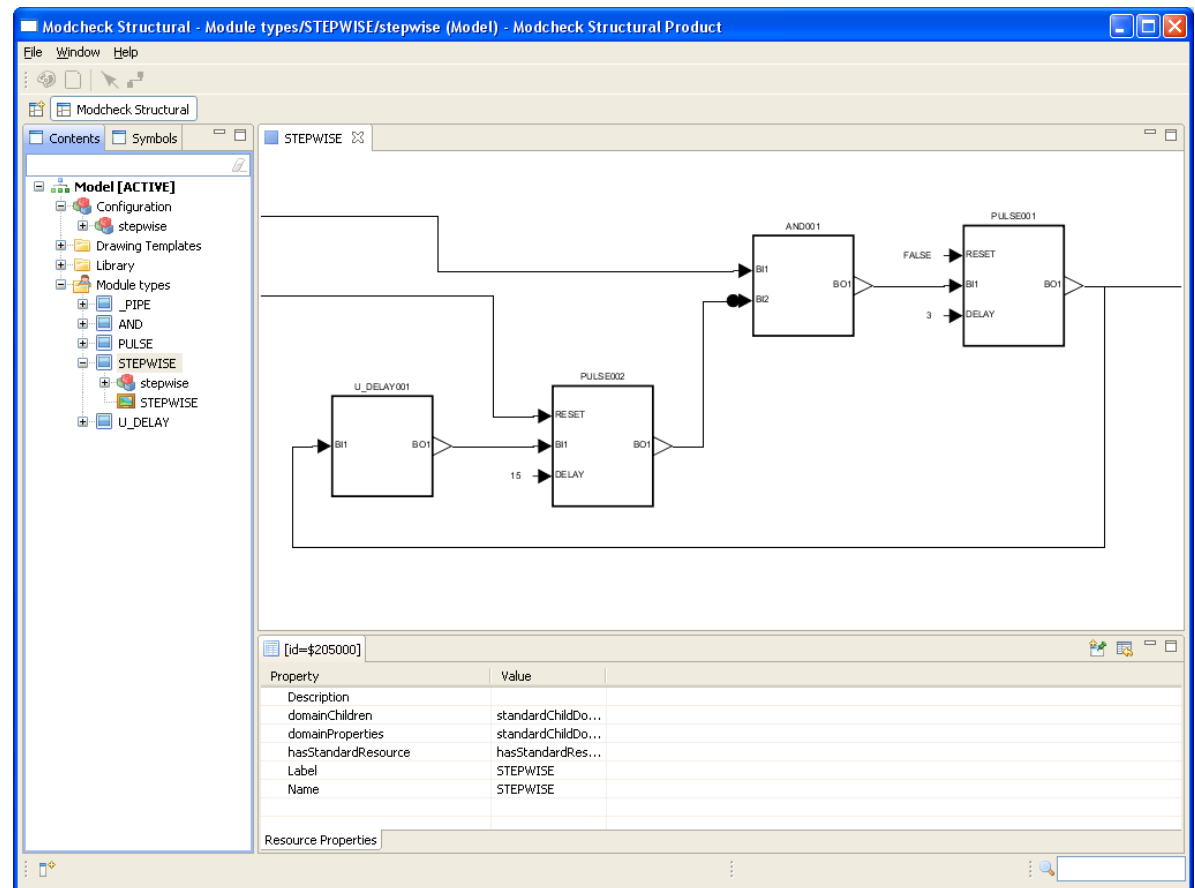
Model checking

- Model checking is an efficient formal method for the verification of critical systems.
- Model checker is a software tool that exhaustively checks that a model of a system satisfies given requirements.
 - The answer is either **YES**...
 - ...or **NO**, in which case the model checker will output a **counter-example**, demonstrating an execution path that does not meet the requirement.



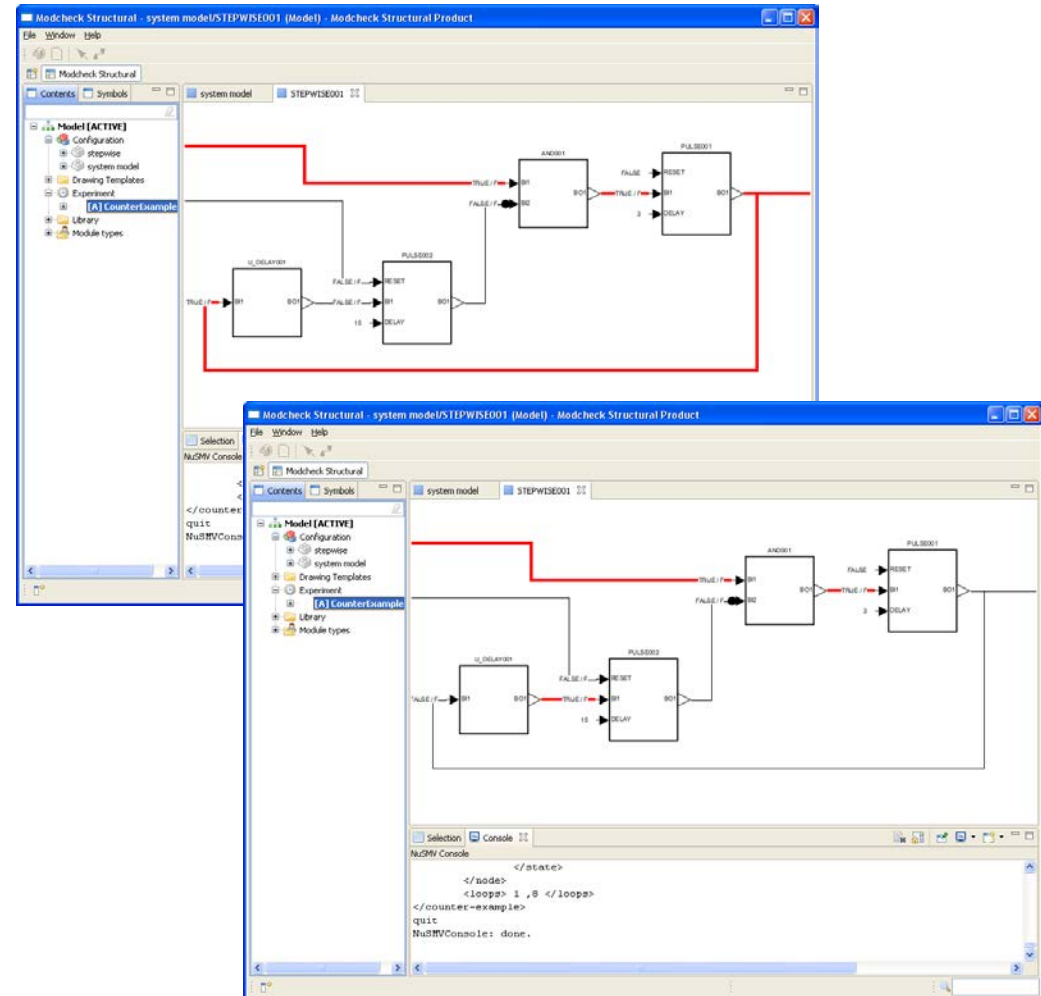
2D modelling with Simantics

- Simantics – open-source platform for modelling and simulation
- <https://www.simantics.org/>
- Specify the function block diagram in a 2D view.



Browsing counter-examples with Simantics

- Review of counter-examples with "living" function block diagrams
- In the future: automatic transformation of SMV model from design data



Requirement templates

- Mapping of natural language description with the associated temporal logic
- "Fill in the blanks" – refer to domain concepts (i.e. devices, functions, I&C system inputs and outputs)

Natural language template		Temporal logic representation									
<table border="1"> <tr> <td>P</td> <td colspan="3">occurs at most 2 times</td> </tr> <tr> <td>between</td> <td>Q</td> <td>and</td> <td>R</td> </tr> </table>	P	occurs at most 2 times			between	Q	and	R	$G((Q \ \& \ F(R)) \rightarrow ((!P \ \& \ !R) \cup (R \mid ((P \ \& \ !R) \cup (R \mid ((!P \ \& \ !R) \cup (R \mid ((P \ \& \ !R) \cup (R \mid (!P \cup R))))))))))$		
P	occurs at most 2 times										
between	Q	and	R								

Requirement templates – an example

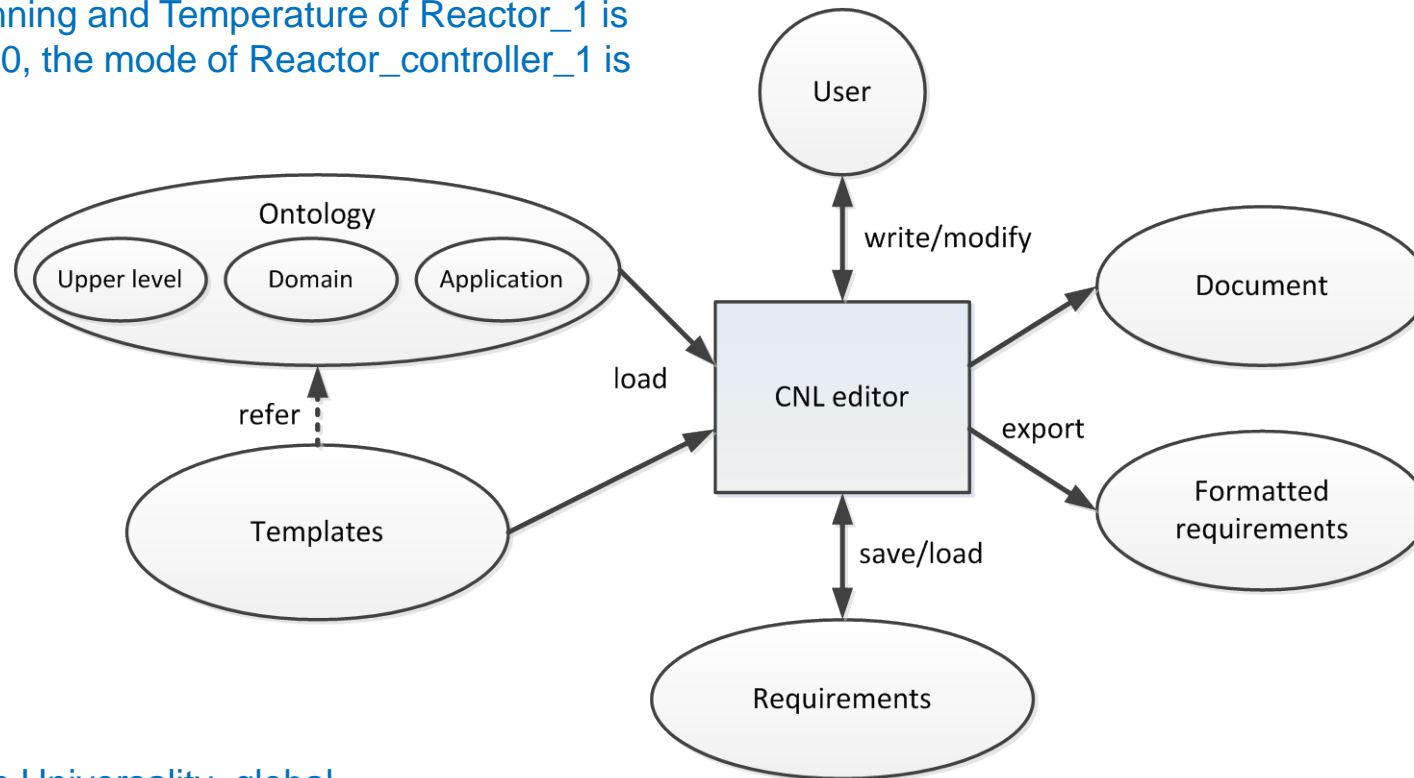
Requirement:

After the shutdown signal, valve V15 must be opened and remain open until the level in tank T4 is below 230 cm.

Natural language template		Temporal logic representation
V15_OPEN	must hold	$G((\text{SHUTDOWN_SIGNAL} \ \& \ !((\text{T4_LEVEL_M} < 230)) \ -> \ (G(\text{V15_OPEN}) \ \ ! \ \text{V15_OPEN} \ U \ (\text{T4_LEVEL_M} < 230))))$
after	SHUTDOWN_SIGNAL	
until	(T4_LEVEL_M < 230)	

CNL editor, the overall concept

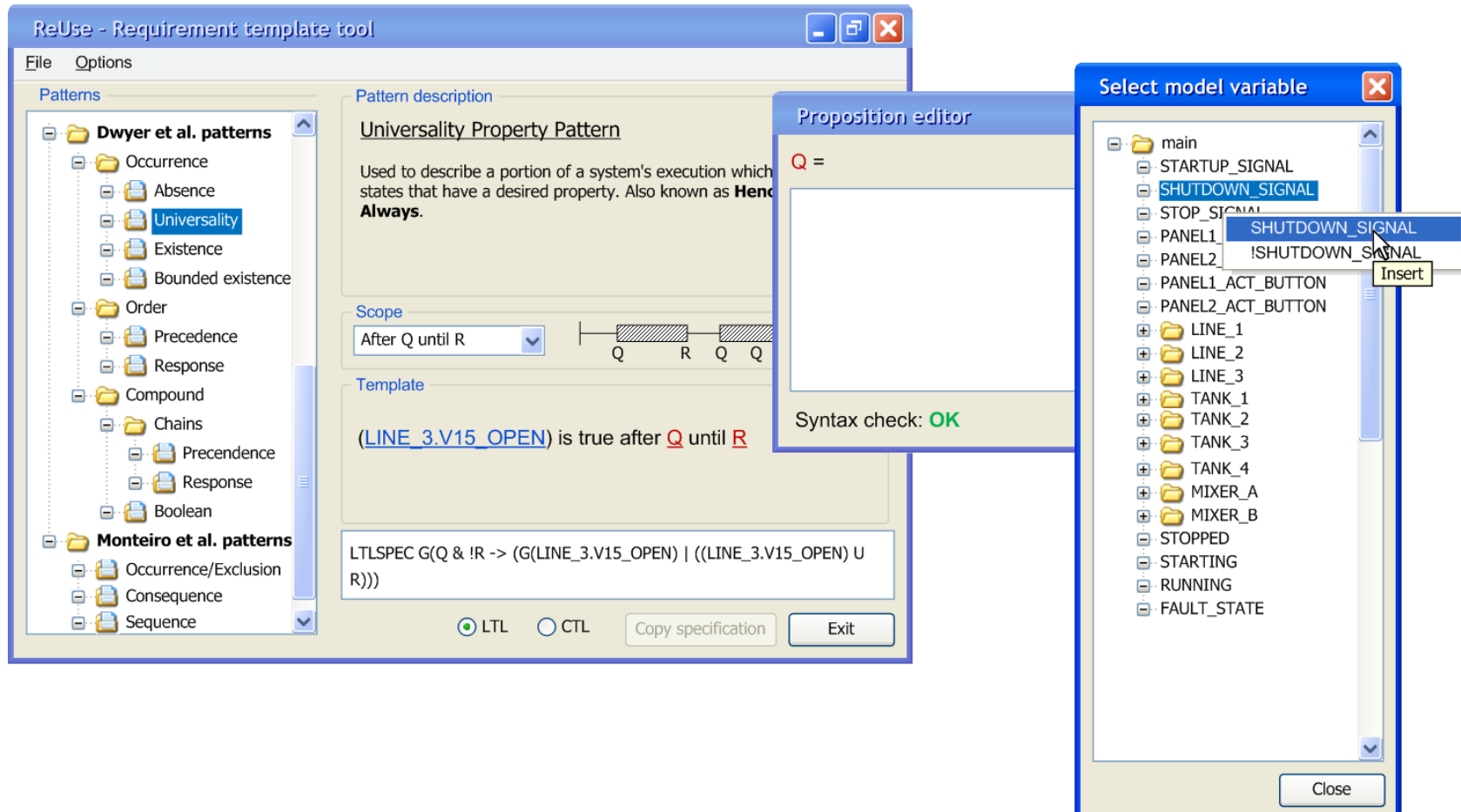
While the Reactor_controller_1 is in operational state Running and Temperature of Reactor_1 is above 100, the mode of Reactor_controller_1 is Auto.



template Universality_global
 sentence Condition <_proposition::P> holds globally.
 formula LTLSPEC G(<P>)

LTLSPEC G ((Reactor1.Temperature > 100) &
 (Reactor_controller_1.state = Running)) ->
 (Reactor_controller_1.mode = Auto)

User interface option 1 – Fill in the blanks



The screenshot displays the ReUse - Requirement template tool interface. The main window shows the 'Universality Property Pattern' configuration. The 'Patterns' list on the left includes 'Dwyer et al. patterns' and 'Monteiro et al. patterns'. The 'Universality' pattern is selected. The 'Pattern description' section provides a brief explanation of the pattern. The 'Scope' section shows a dropdown menu set to 'After Q until R' and a corresponding timeline diagram. The 'Template' section displays the LTL specification: (LINE_3.V15_OPEN) is true after Q until R. The 'Proposition editor' dialog is open, showing the text 'Q =' and a syntax check result of 'OK'. The 'Select model variable' dialog is also open, showing a list of model variables with 'SHUTDOWN_SIGNAL' selected and an 'Insert' button.

ReUse - Requirement template tool

File Options

Patterns

- Dwyer et al. patterns
 - Occurrence
 - Absence
 - Universality
 - Existence
 - Bounded existence
- Order
 - Precedence
 - Response
- Compound
 - Chains
 - Precedence
 - Response
 - Boolean
- Monteiro et al. patterns
 - Occurrence/Exclusion
 - Consequence
 - Sequence

Pattern description

Universality Property Pattern

Used to describe a portion of a system's execution which states that have a desired property. Also known as **Hence Always**.

Scope

After Q until R

Template

(LINE_3.V15_OPEN) is true after Q until R

LTLSPEC $G(Q \ \& \ !R \ \rightarrow \ (G(\text{LINE_3.V15_OPEN}) \ | \ ((\text{LINE_3.V15_OPEN}) \ U \ R)))$

LTL CTL Copy specification Exit

Proposition editor

Q =

Syntax check: OK

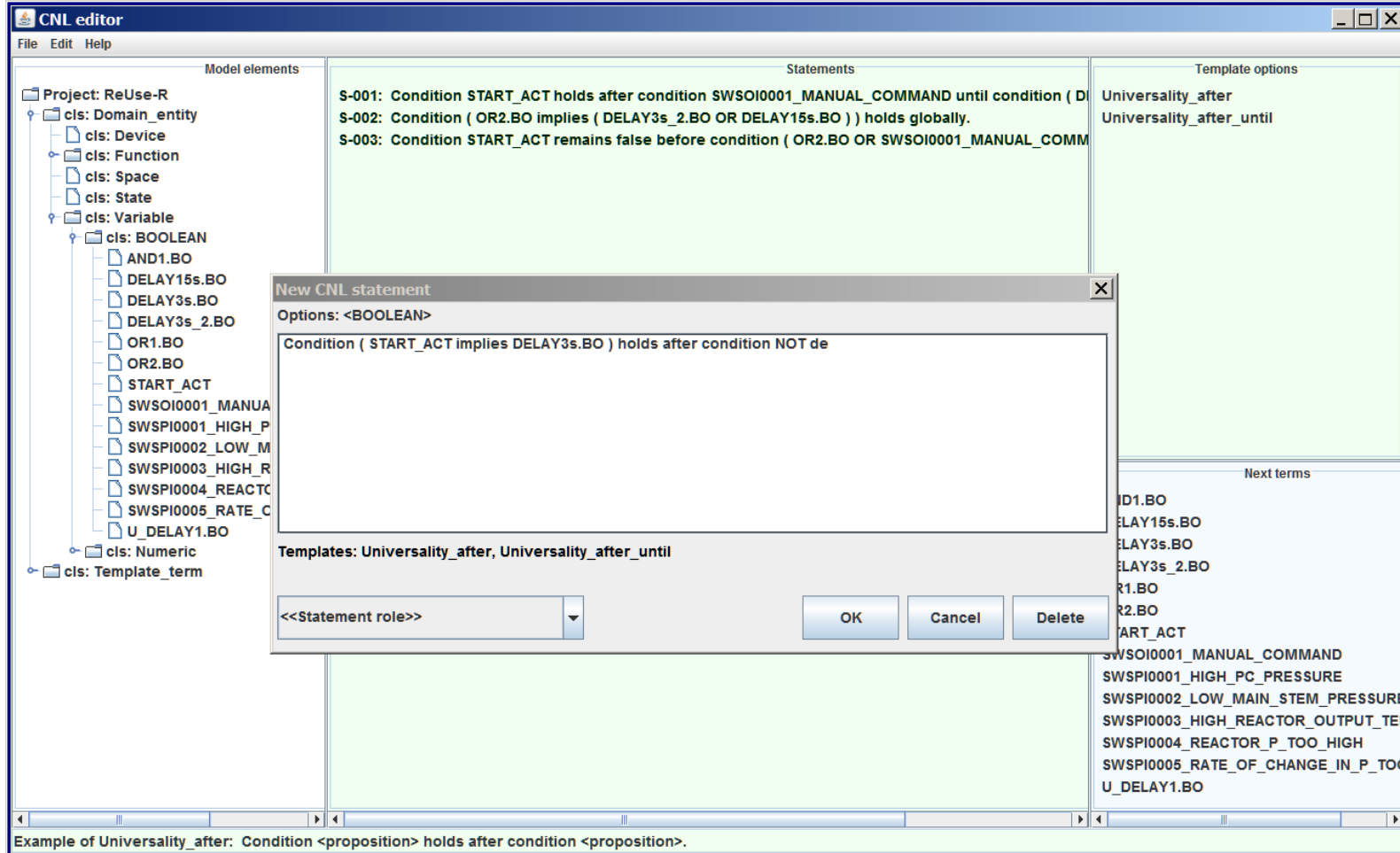
Select model variable

- main
 - STARTUP_SIGNAL
 - SHUTDOWN_SIGNAL
 - STOP_SIGNAL
 - PANEL1
 - SHUTDOWN_SIGNAL
 - ISHUTDOWN_SIGNAL
 - PANEL2
 - PANEL1_ACT_BUTTON
 - PANEL2_ACT_BUTTON
 - LINE_1
 - LINE_2
 - LINE_3
 - TANK_1
 - TANK_2
 - TANK_3
 - TANK_4
 - MIXER_A
 - MIXER_B
 - STOPPED
 - STARTING
 - RUNNING
 - FAULT_STATE

Insert

Close

Option 2 - Editing guided by templates with autocomplete



The screenshot displays the CNL editor interface. On the left, a tree view shows the project structure under 'Project: ReUse-R', including classes like 'Domain_entity', 'Device', 'Function', 'Space', 'State', 'Variable', 'BOOLEAN', and 'Numeric'. The main area is divided into three panes: 'Model elements', 'Statements', and 'Template options'. The 'Statements' pane contains three conditions (S-001, S-002, S-003). The 'Template options' pane lists 'Universality_after' and 'Universality_after_until'. A 'New CNL statement' dialog box is open, showing the current options as '<BOOLEAN>' and a template suggestion: 'Condition (START_ACT implies DELAY3s.BO) holds after condition NOT de'. The dialog also shows a list of 'Next terms' and buttons for 'OK', 'Cancel', and 'Delete'. At the bottom, an example of the 'Universality_after' template is provided: 'Example of Universality_after: Condition <proposition> holds after condition <proposition>.'

Statements and corresponding temporal logic formulas

Statement S-001

Sentence Condition START_ACT holds after condition SWSOI0001_MANUAL_COMMAND until condition (DELAY15s.clock is equal_to 0).

Statement S-002

Sentence Condition (OR2.BO implies (DELAY3s_2.BO OR DELAY15s.BO)) holds globally.

Statement S-003

Sentence Condition START_ACT remains false before condition (OR2.BO OR SWSOI0001_MANUAL_COMMAND).

Statement S-004

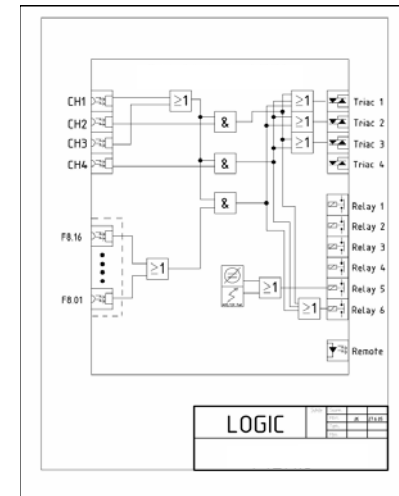
Sentence Condition (START_ACT implies DELAY3s.BO) holds after condition NOT DELAY3s_2.BO until condition (DELAY15s.clock is equal_to 0).

```
LTLSPEC G(SWSOI0001_MANUAL_COMMAND & !(DELAY15s.clock = 0) -> (G(START_ACT) | (START_ACT U (DELAY15s.clock = 0))))
LTLSPEC G((OR2.BO -> (DELAY3s_2.BO | DELAY15s.BO)))
LTLSPEC F(OR2.BO | SWSOI0001_MANUAL_COMMAND) -> (!START_ACT U (OR2.BO | SWSOI0001_MANUAL_COMMAND))
LTLSPEC G(!DELAY3s_2.BO & !(DELAY15s.clock = 0) -> (G((START_ACT -> DELAY3s.BO)) | ((START_ACT -> DELAY3s.BO) U (DELAY15s.clock = 0))))
```

Conclusions / further research

- What kind of templates would be needed for real-world I&C systems?
- How should the CNL editor be implemented and integrated with other design / analysis tools?
- How should the user interface look like?
- Practical evaluation is needed!

- <http://www.vtt.fi/modelchecking/>



- STUK - Radiation and Nuclear Safety Authority, Finland
- Evaluation of NPP I&C system designs 2008-2011
- Fortum, Power company, Finland
- Verification of nuclear I&C by model checking 2012-

