Towards Domain-Specific Property Languages: The ProMoBox Approach

Bart Meyers, Manuel Wimmer, Hans Vangheluwe, Joachim Denil

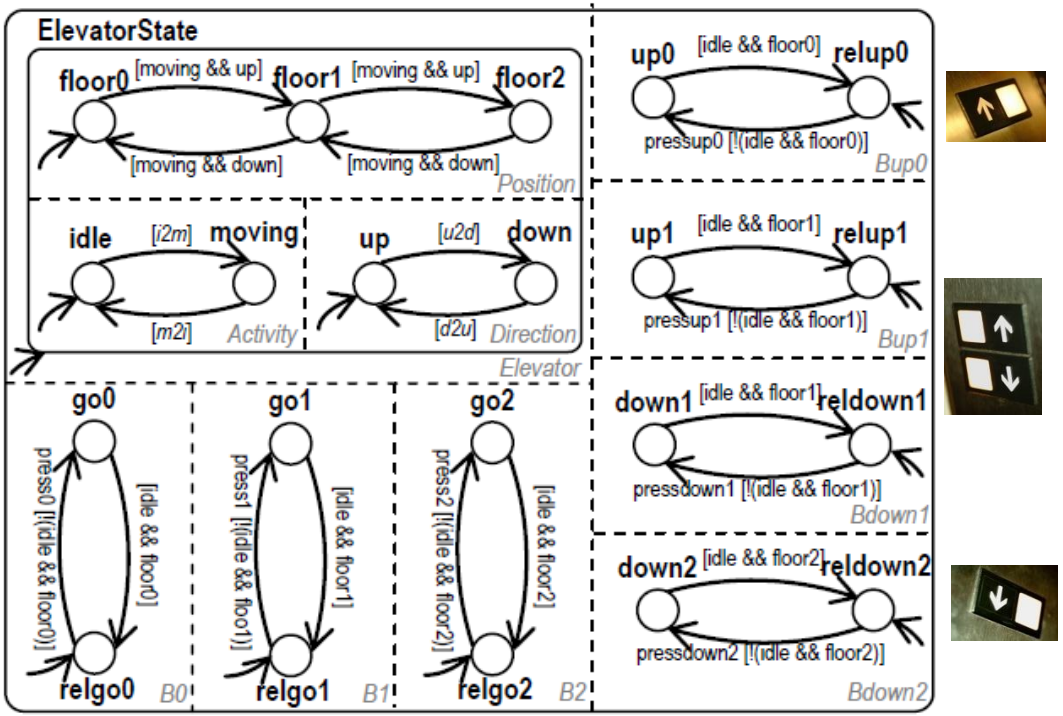
The 13th Workshop on Domain-Specific Modeling @ SPLASH 2013

Motivation

- **DSM is a major building block of MDE**
  - Transition from solution space to design space
  - Domain experts are able to design systems
- **DSM is not unique to the design of systems**
  - Other tasks may benefit as well
  - However, for specifying and verifying properties, domain experts are forced to switch to solution space of model checkers
- **Contribution of ProMoBox**
  - DSM support for specifying and verifying desired properties
    - Properties and verification results are formulated within the base DSML
    - Transparent translation to SPIN for efficiently performing the verification
  - Annotation-based and automated process for enriching design languages with property languages
  - Application of the approach for Statecharts

Running Example: Elevator Statechart Model

- Elevator with three floors
- Transitions are triggered by events/guards



i2m = (go0 or up0) and not floor0
or (go1 or up1 or down1) and not floor1
or (go2 or down2) and not floor2

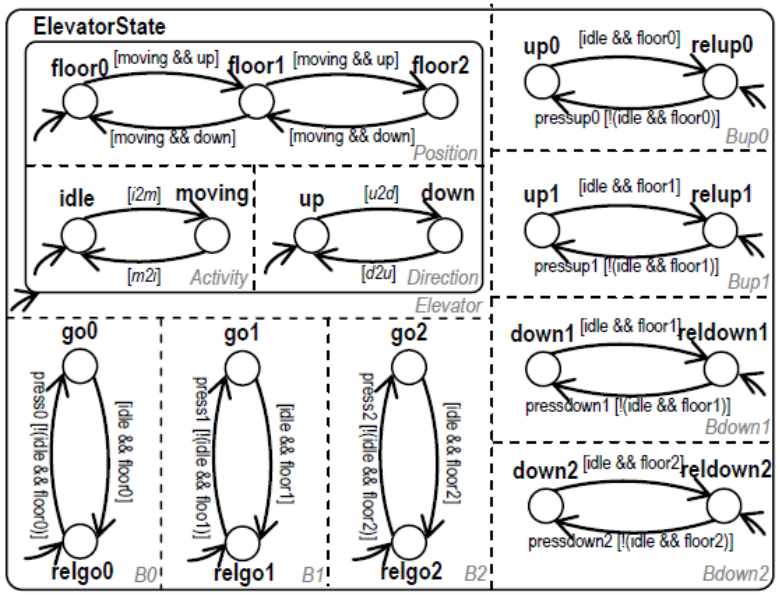
u2d = floor2 and (go1 or up1 or down1 or go0 or up0)
or floor1 and not (go2 or down2) and (go0 or up0)

d2u = floor0 and (go1 or up1 or down1 or go2 or down2)
or floor1 and not (go0 or up0) and (go2 or down2)

m2i = floor0 and go0
or floor0 and up0 and up
or floor1 and go1
or floor1 and up1 and up
or floor1 and down1 and down
or floor2 and go2
or floor2 and down2 and down

Running Example: Elevator Statechart Model

- **Desired properties for the elevator system**
  - **ReachesFloor**: when a request for any floor is made, the elevator eventually opens its doors at that floor;
  - **SkipFloorOnce**: when a request for any floor is made, the elevator opens its doors at the latest the second time it passes that floor.



i2m = (go0 or up0) and not floor0
or (go1 or up1 or down1) and not floor1
or (go2 or down2) and not floor2

u2d = floor2 and (go1 or up1 or down1 or go0 or up0)
or floor1 and not (go2 or down2) and (go0 or up0)

d2u = floor0 and (go1 or up1 or down1 or go2 or down2)
or floor1 and not (go0 or up0) and (go2 or down2)

m2i = floor0 and go0
or floor0 and up0 and up
or floor1 and go1
or floor1 and up1 and up
or floor1 and down1 and down
or floor2 and go2
or floor2 and down2 and down

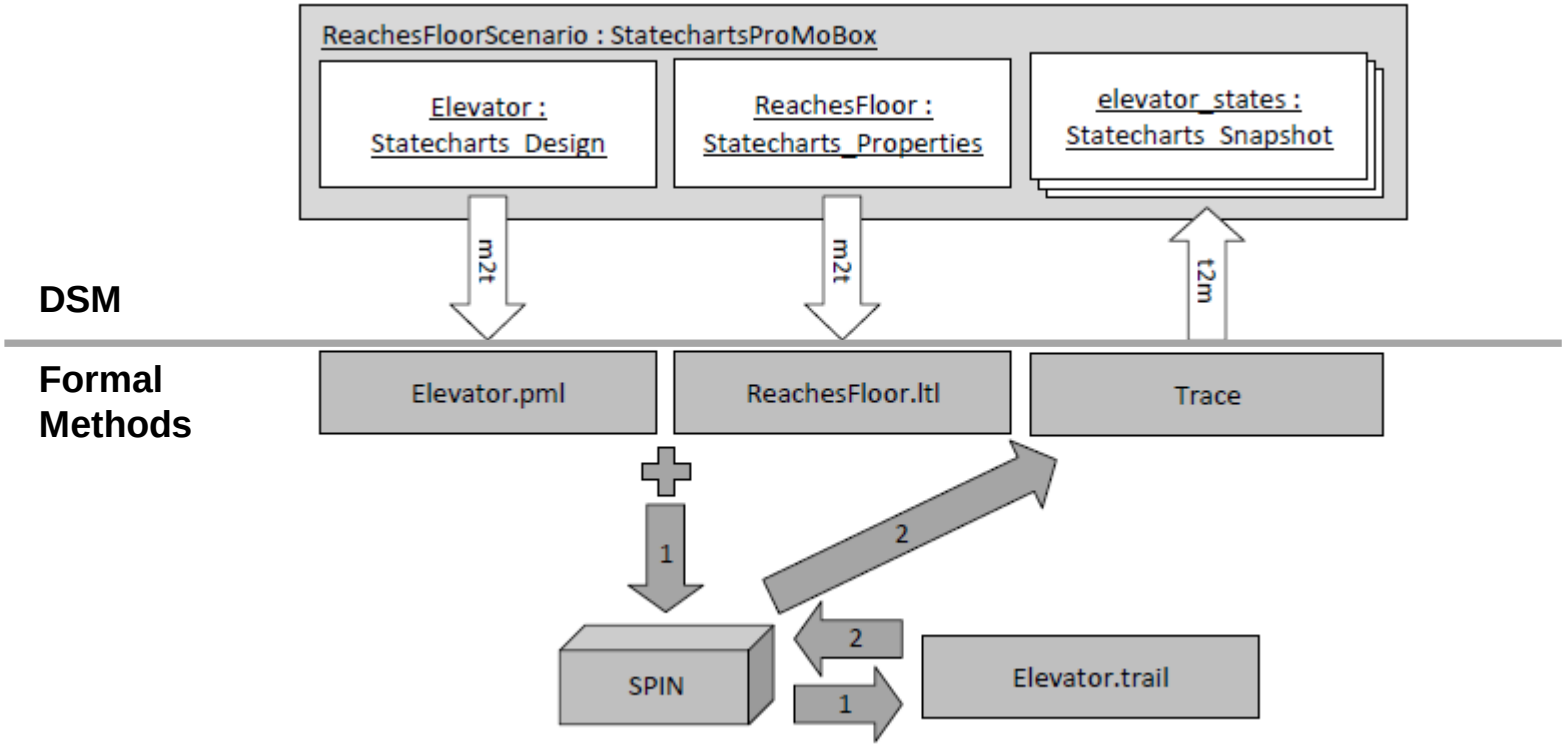
SPIN/PROMELA

SkipFloorOnce in LTL:

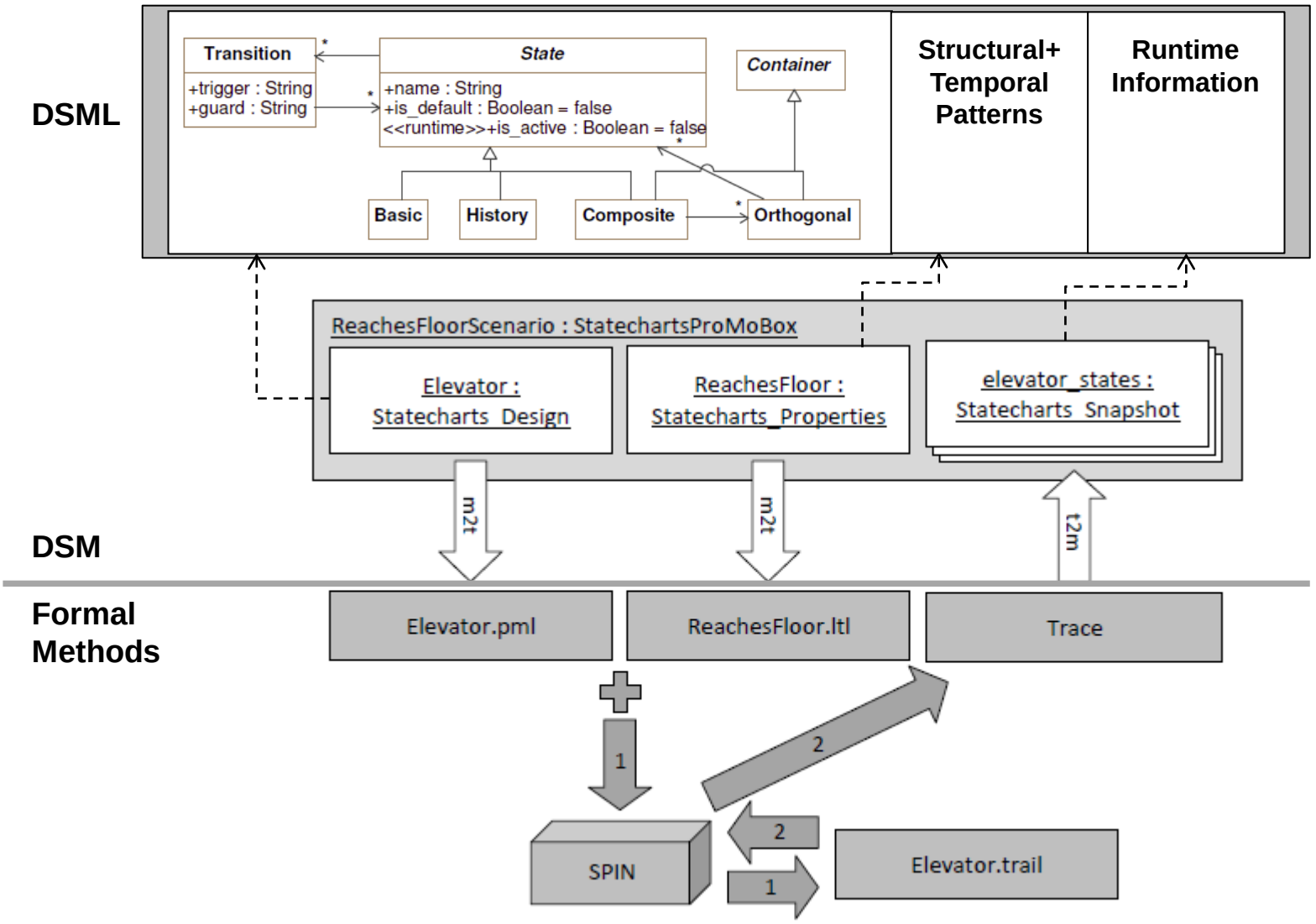
$$\Box(((go0 \wedge up0) \vee \Diamond(floor0 \vee idle)) \rightarrow ((\neg(floor0) \vee \neg(floor0 \vee idle)) \mathcal{U} ((floor0 \vee idle) \wedge (((floor0) \vee \neg(floor0 \vee idle)) \mathcal{U} ((floor0 \vee idle) \wedge ((\neg(floor0) \vee \neg(floor0 \vee idle)) \mathcal{U} ((floor0 \vee idle) \wedge (((floor0) \vee \neg(floor0 \vee idle)) \mathcal{U} ((floor0 \vee idle) \wedge (\neg(floor0) \mathcal{U} (floor0 \vee idle))))))))))) \vee \Box(((go1 \wedge up1 \wedge down1) \vee \Diamond(floor1 \vee idle)) \rightarrow ((\neg(floor1) \vee \neg(floor1 \vee idle)) \mathcal{U} ((floor1 \vee idle) \wedge (((floor1) \vee \neg(floor1 \vee idle)) \mathcal{U} ((floor1 \vee idle) \wedge ((\neg(floor1) \vee \neg(floor1 \vee idle)) \mathcal{U} ((floor1 \vee idle) \wedge (((floor1) \vee \neg(floor1 \vee idle)) \mathcal{U} ((floor1 \vee idle) \wedge (\neg(floor1) \mathcal{U} (floor1 \vee idle))))))))))) \vee \Box(((go2 \wedge down2) \vee \Diamond(floor2 \vee idle)) \rightarrow ((\neg(floor2) \vee \neg(floor2 \vee idle)) \mathcal{U} ((floor2 \vee idle) \wedge (((floor2) \vee \neg(floor2 \vee idle)) \mathcal{U} ((floor2 \vee idle) \wedge ((\neg(floor2) \vee \neg(floor2 \vee idle)) \mathcal{U} ((floor2 \vee idle) \wedge (\neg(floor2) \mathcal{U} (floor2 \vee idle))))))))))$$

The ProMoBox Approach at a Glance

- **Goals**
  1) **Specification** of **temporal properties** on DSML level
  2) **Reduce** the **complexity** of LTL formulas by using specification patterns
  3) **Accessible verification results** on DSML level by using snapshots
  4) **Language generation** support for design-oriented DSMLs



What is needed for a ProMoBox?



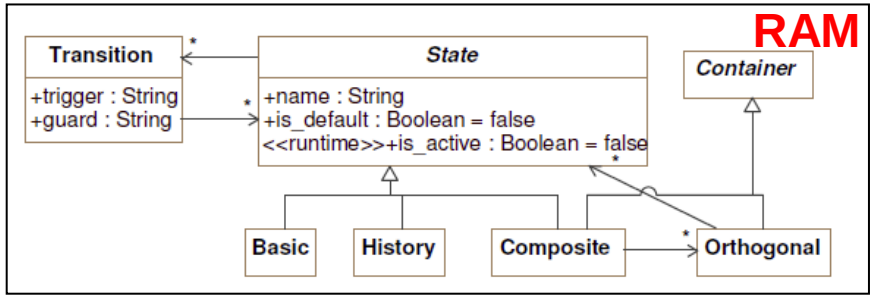
Generating a ProMoBox from a base DSML

- **Automated process to complete DSMLs based on**
  - **Metamodel modules**
  - **Model transformations**

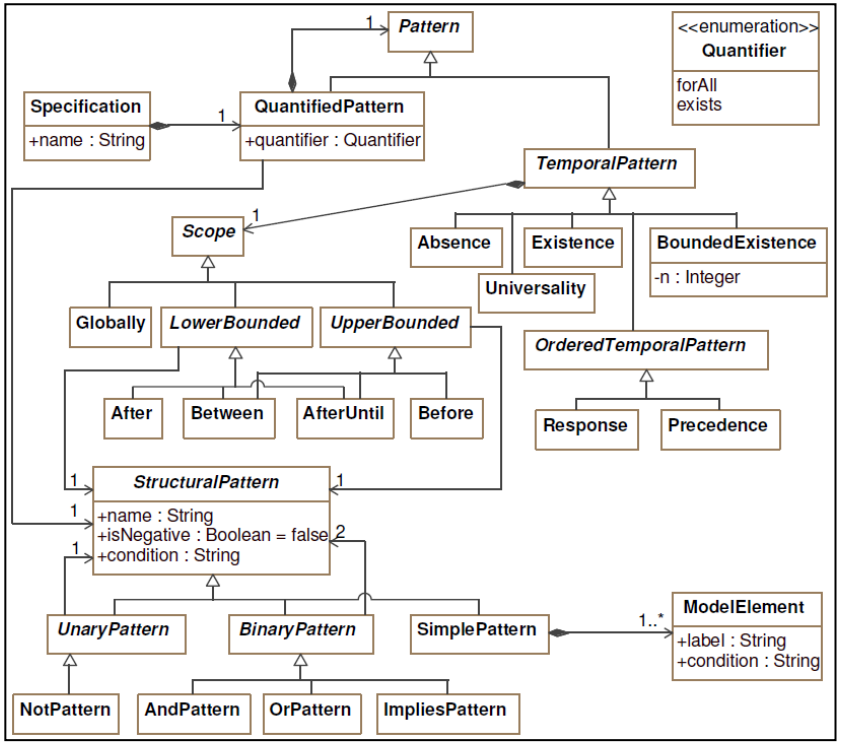
- **Reusable Metamodel Module**
  1) **Quantification**
     - ForAll, Exists
  2) **Structural Patterns**
     - Based on PaMoMo
  3) **Temporal Patterns**
     - Based on Dwyer‘s specification patterns
  4) **Pattern Elements**
     - Based on the host DSML and RAMification
     - Introduction of runtime elements

Generating a ProMoBox for Statecharts

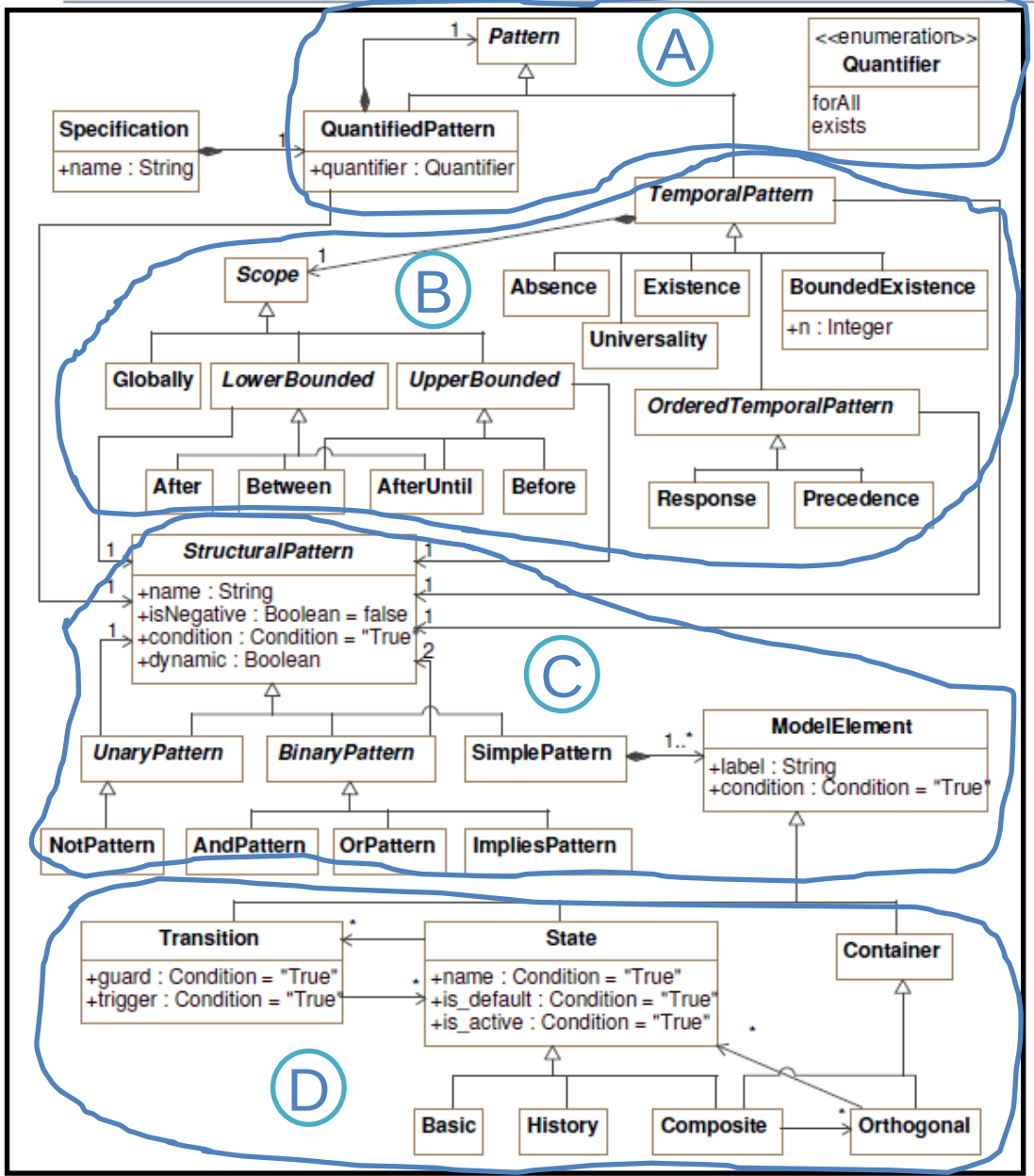
DSML Model



Property Specification Module



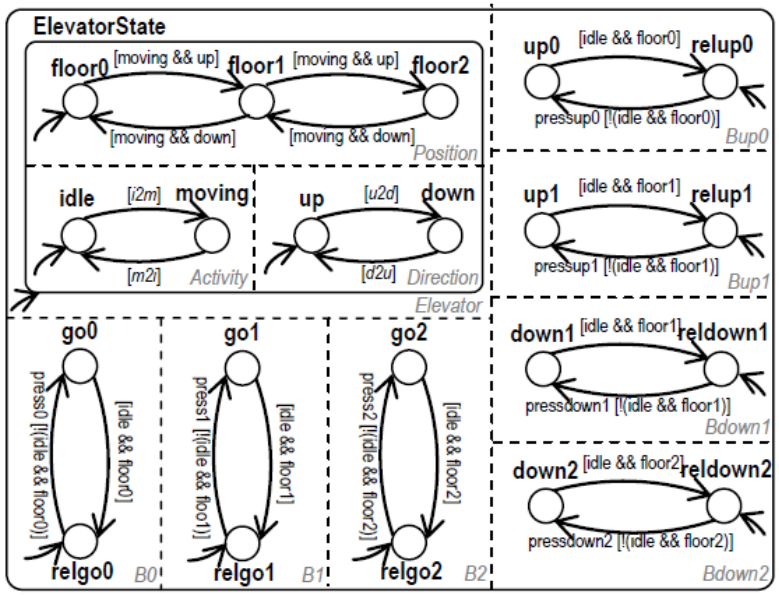
The Resulting ProMoBox for Statecharts



- **A. Quantification**
  - ForAll, Exists
- **B. Temporal Patterns**
  - Based on specification patterns
- **C. Structural Patterns**
  - Based on PaMoMo
- **D. Pattern Elements**
  - Based on RAMification

Running Example: Elevator Statechart Model Revisited

- **Desired properties for the elevator system**
  - **ReachesFloor**: when a request for any floor is made, the elevator eventually opens its doors at that floor

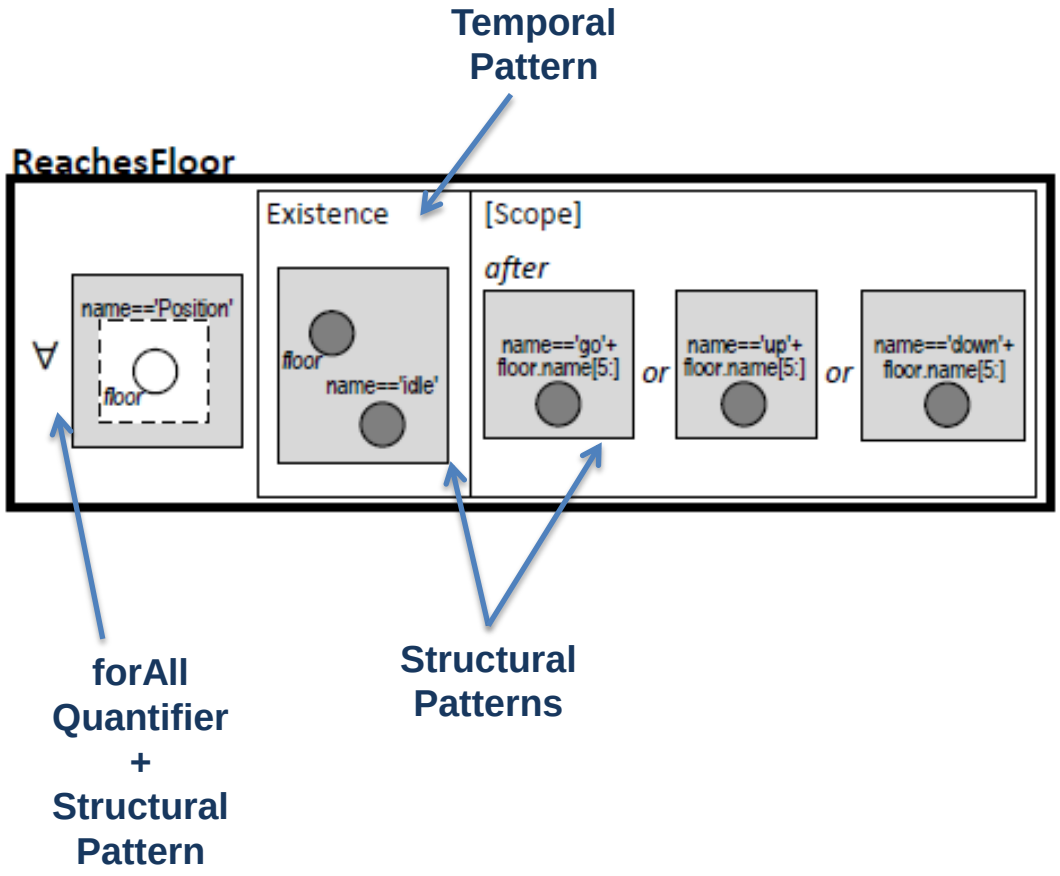


i2m = (go0 or up0) and not floor0
or (go1 or up1 or down1) and not floor1
or (go2 or down2) and not floor2

u2d = floor2 and (go1 or up1 or down1 or go0 or up0)
or floor1 and not (go2 or down2) and (go0 or up0)

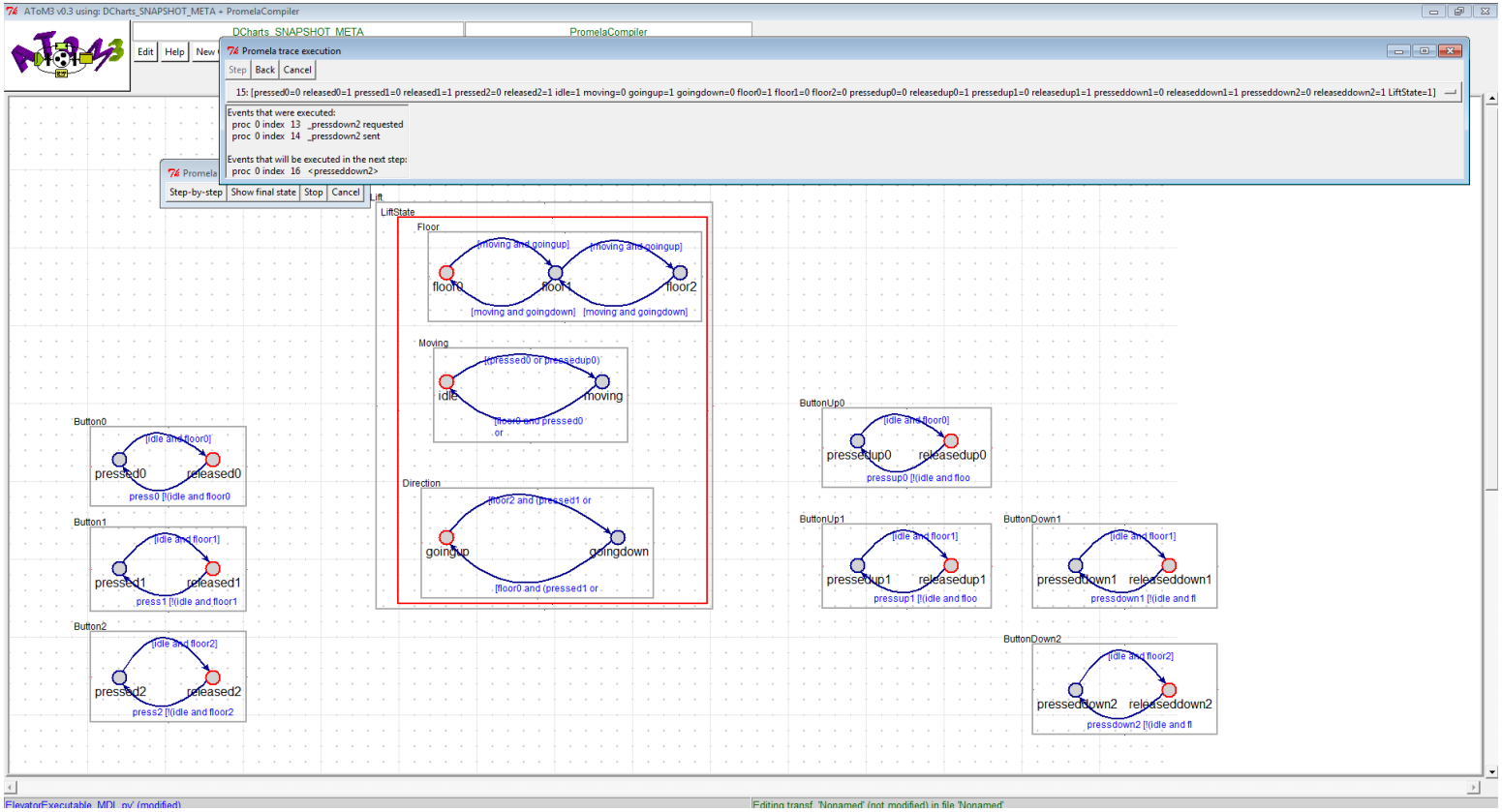
d2u = floor0 and (go1 or up1 or down1 or go2 or down2)
or floor1 and not (go0 or up0) and (go2 or down2)

m2i = floor0 and go0
or floor0 and up0 and up
or floor1 and go1
or floor1 and up1 and up
or floor1 and down1 and down
or floor2 and go2
or floor2 and down2 and down



Running Example: Elevator Statechart Model Revisited

- **Counter Example Representation**
  - Trace expressed in a snapshot language
  - Animation tool exploits this information by replying the trace information on the model level



Experiences

- Properties for the evaluator system are specified and verified with ProMoBox
  - Tool support in ATOM^3
- Complexity of the generated LTL formula for the ReachesFloor property
  - 31 operators
  - 20 proposition variables,
  - brackets depth of 7 levels
- Complexity of the generated LTL formula for the SkipFloorOnce property
  - 107 operators
  - 82 proposition variables,
  - brackets depth of 11 levels
- Verification of the model
  - Maximum search depth of 10.000
  - Sufficient to explore the full state space
  - Done within minutes

What's next?

1) Provide an elevator DSML on top of Statecharts

- Reuse the current framework as an intermediate layer to provide model checking possibilities for several DSML that fit their semantics on state/transition systems.

2) Currently no support for real-time properties

- Make use of documented patterns for the real-time property domain and formulate them as DSML
  - V. Gruhn and R. Laue. Patterns for timed property specifications. ENTCS, 153(2):117-133, 2006.
  - S. Konrad and B.H.C. Cheng. Real-time specification patterns. In ICSE'05, pp. 372-381, 2005.