ProMoBox: Domain-Specific Modelling Languages for Verification and Testing

Bart Meyers

Hans Vangheluwe

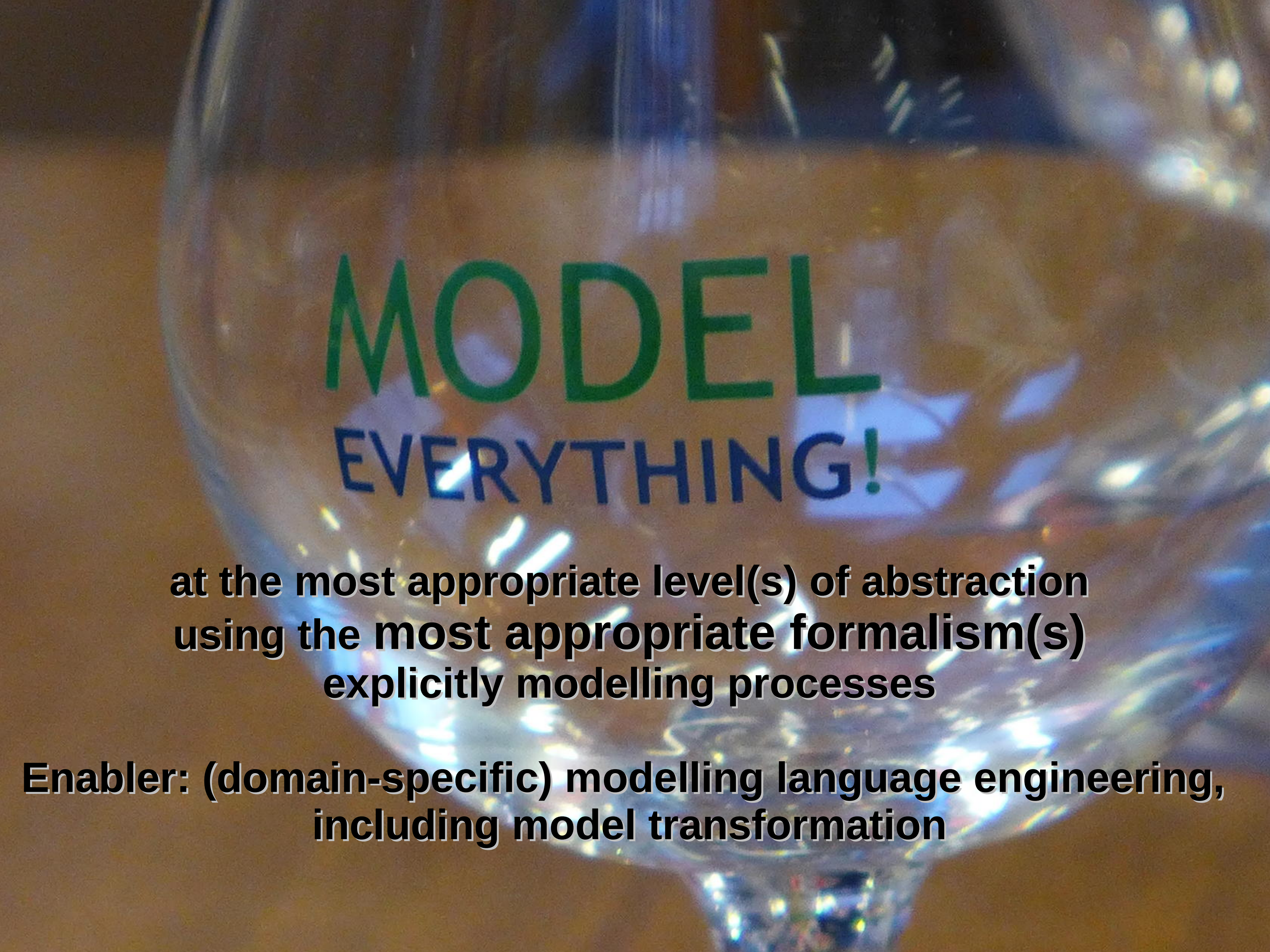
Universiteit Antwerpen

Ansymo
Antwerp Systems & Software Modelling
University of Antwerp

Nexor
Cyber-Physical Systems
University of Antwerp

FLANDERS MAKE
MANUFACTURING INNOVATION NETWORK

McGill



at the most appropriate level(s) of abstraction
using the most appropriate formalism(s)
explicitly modelling processes

Enabler: (domain-specific) modelling language engineering,
including model transformation

UNIVERSITÉ DE GENÈVE
Centre Universitaire d'Informatique

FACULTÉ DES SCIENCES
Professeur D. Buchs, directeur
Professeur G. Falquet, codirecteur

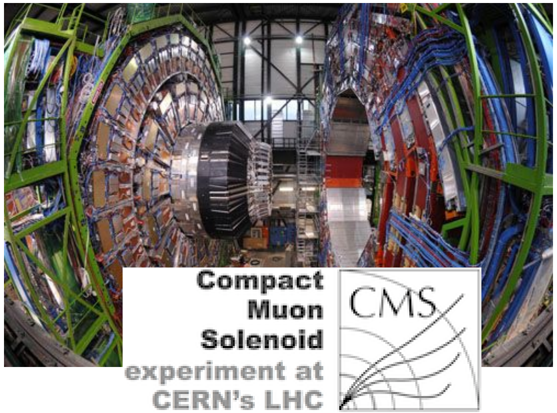
A Methodology For The Development Of Complex Domain Specific Languages

THÈSE

présentée à la Faculté des sciences de l'Université de Genève
pour obtenir le grade de Docteur ès sciences, mention informatique

2010

Matteo Risoldi



CMS Tracker Cosmic Rack

Design

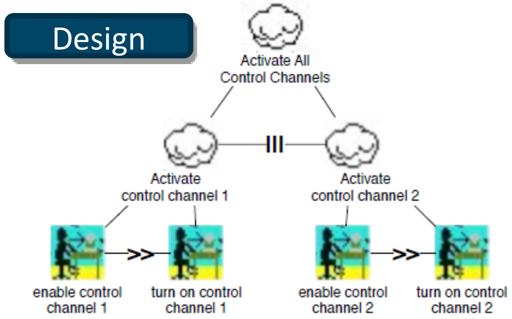
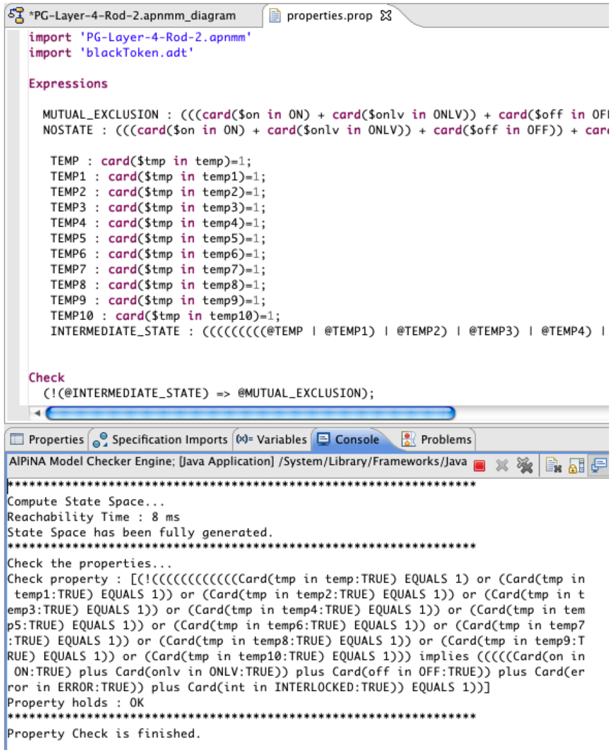


Figure 4.8. CTT for the turn on control channels task

Property



```
import 'PG-Layer-4-Rod-2.apnmm'
import 'blackToken.adt'

Expressions

  MUTUAL_EXCLUSION : (((card($on in ON) + card($onlv in ONLV)) + card($off in OFI
  NOSTATE : (((card($on in ON) + card($onlv in ONLV)) + card($off in OFF)) + card

   TEMP : card($tmp in temp)=1;
   TEMP1 : card($tmp in temp1)=1;
   TEMP2 : card($tmp in temp2)=1;
   TEMP3 : card($tmp in temp3)=1;
   TEMP4 : card($tmp in temp4)=1;
   TEMP5 : card($tmp in temp5)=1;
   TEMP6 : card($tmp in temp6)=1;
   TEMP7 : card($tmp in temp7)=1;
   TEMP8 : card($tmp in temp8)=1;
   TEMP9 : card($tmp in temp9)=1;
   TEMP10 : card($tmp in temp10)=1;
   INTERMEDIATE_STATE : ((((((((@TEMP | @TEMP1) | @TEMP2) | @TEMP3) | @TEMP4) |


Check
  (!(@INTERMEDIATE_STATE) => @MUTUAL_EXCLUSION);
```

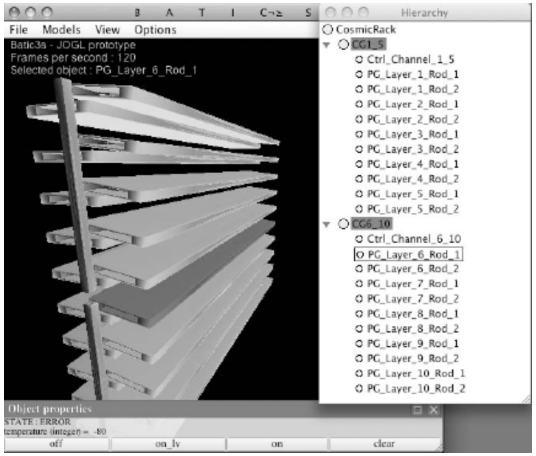
AlPiNA Model Checker Engine; [Java Application] /System/Library/Frameworks/Java

```
*****************************************************************
Compute State Space...
Reachability Time : 8 ms
State Space has been fully generated.
*****************************************************************
Check the properties...
Check property : [(!((((((((((((Card(tmp in temp:TRUE) EQUALS 1) or (Card(tmp in
 temp1:TRUE) EQUALS 1)) or (Card(tmp in temp2:TRUE) EQUALS 1)) or (Card(tmp in t
emp3:TRUE) EQUALS 1)) or (Card(tmp in temp4:TRUE) EQUALS 1)) or (Card(tmp in tem
p5:TRUE) EQUALS 1)) or (Card(tmp in temp6:TRUE) EQUALS 1)) or (Card(tmp in temp7
:TRUE) EQUALS 1)) or (Card(tmp in temp8:TRUE) EQUALS 1)) or (Card(tmp in temp9:T
RUE) EQUALS 1)) or (Card(tmp in temp10:TRUE) EQUALS 1))) implies (((((Card(on in
 ON:TRUE) plus Card(onlv in ONLV:TRUE)) plus Card(off in OFF:TRUE)) plus Card(er
ror in ERROR:TRUE)) plus Card(int in INTERLOCKED:TRUE)) EQUALS 1))]
Property holds : OK
*****************************************************************
Property Check is finished.
```

Alpina

Application

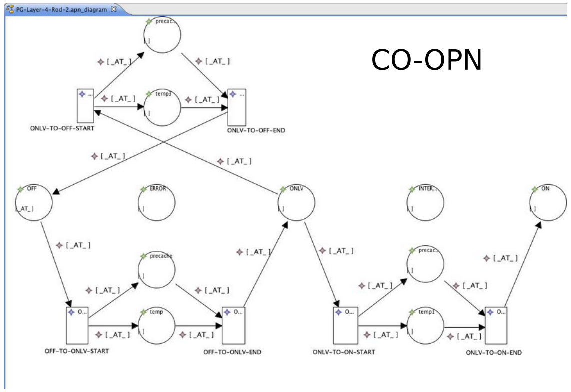
UI prototype



Batic3s - JOGL prototype
Frames per second : 120
Selected object : PG_Layer_6_Rod_1

- CosmicRack
  - CG1_5
    - Ctrl_Channel_1_5
    - PG_Layer_1_Rod_1
    - PG_Layer_1_Rod_2
    - PG_Layer_2_Rod_1
    - PG_Layer_2_Rod_2
    - PG_Layer_3_Rod_1
    - PG_Layer_3_Rod_2
    - PG_Layer_4_Rod_1
    - PG_Layer_4_Rod_2
    - PG_Layer_5_Rod_1
    - PG_Layer_5_Rod_2
  - CG6_10
    - Ctrl_Channel_6_10
    - PG_Layer_6_Rod_1
    - PG_Layer_6_Rod_2
    - PG_Layer_7_Rod_1
    - PG_Layer_7_Rod_2
    - PG_Layer_8_Rod_1
    - PG_Layer_8_Rod_2
    - PG_Layer_9_Rod_1
    - PG_Layer_9_Rod_2
    - PG_Layer_10_Rod_1
    - PG_Layer_10_Rod_2

Object properties
STATE : ERROR
temperature (integer) = -80



CO-OPN

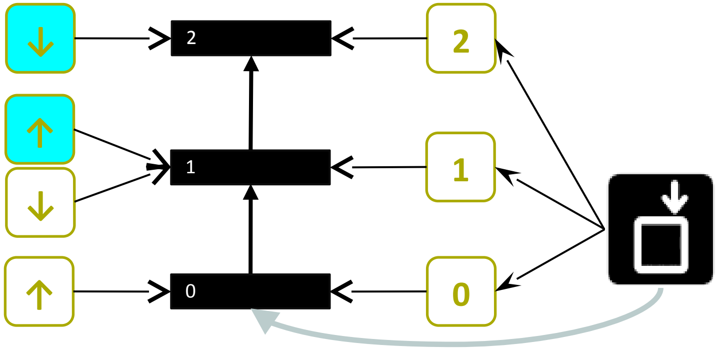
Domain-Specific Modelling

Modelling of complex systems for **domain users**

- Familiar domain **concepts** (reduces cognitive gap)
- Incorporate domain **constraints** (maximally constrain)

Precisely defined (semantics) models

Example: elevator system model in an Elevator DSL



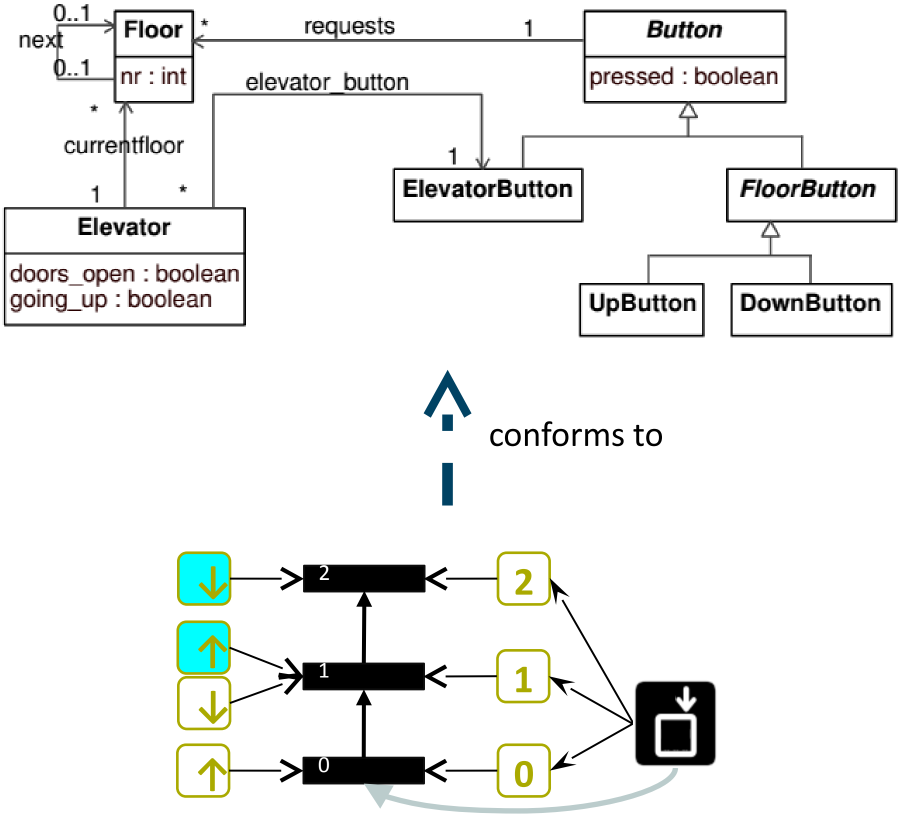
Modelling Language Engineering

Meta-modeling

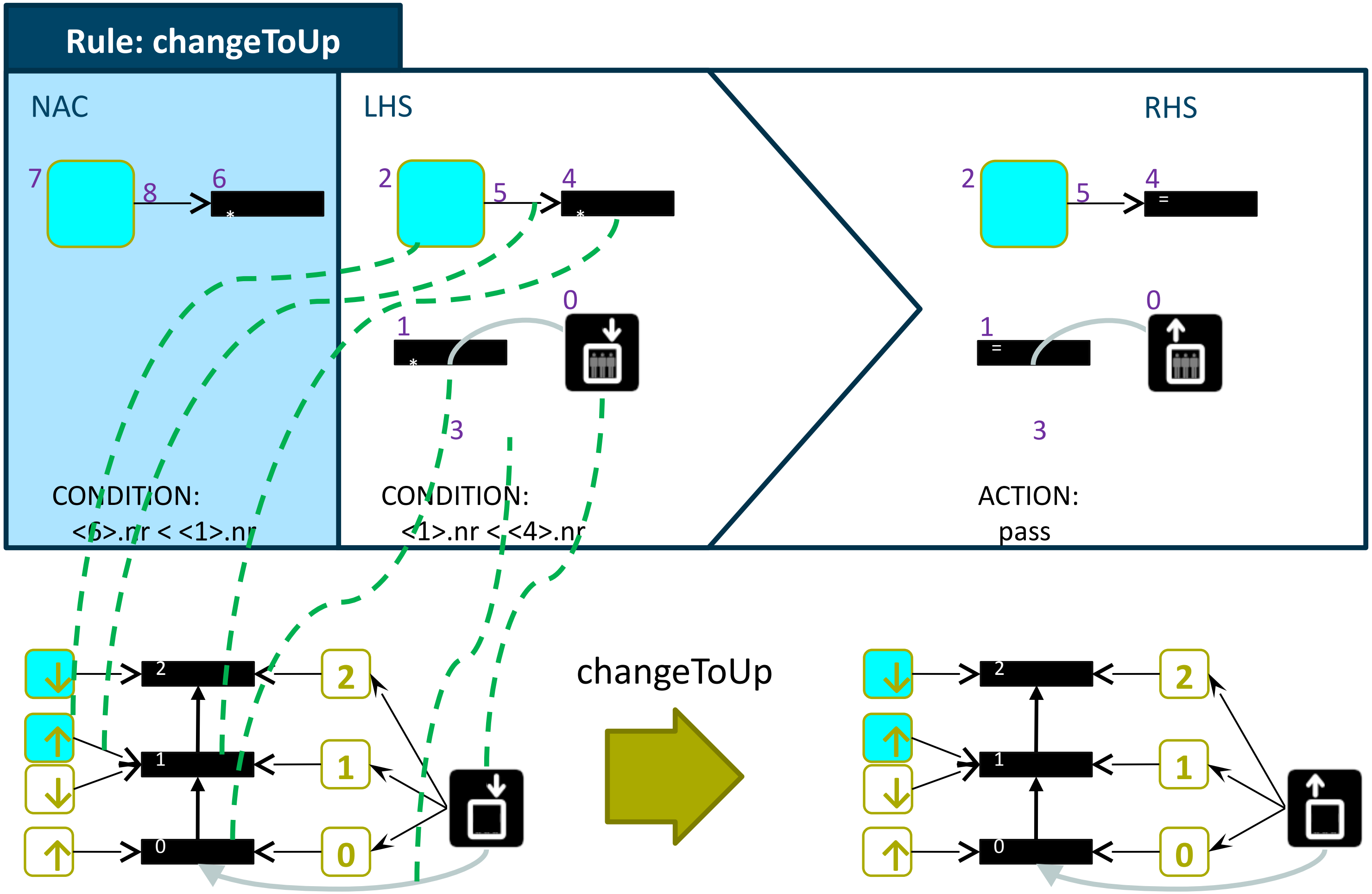
(to specify language abstract syntax)

Model Transformation

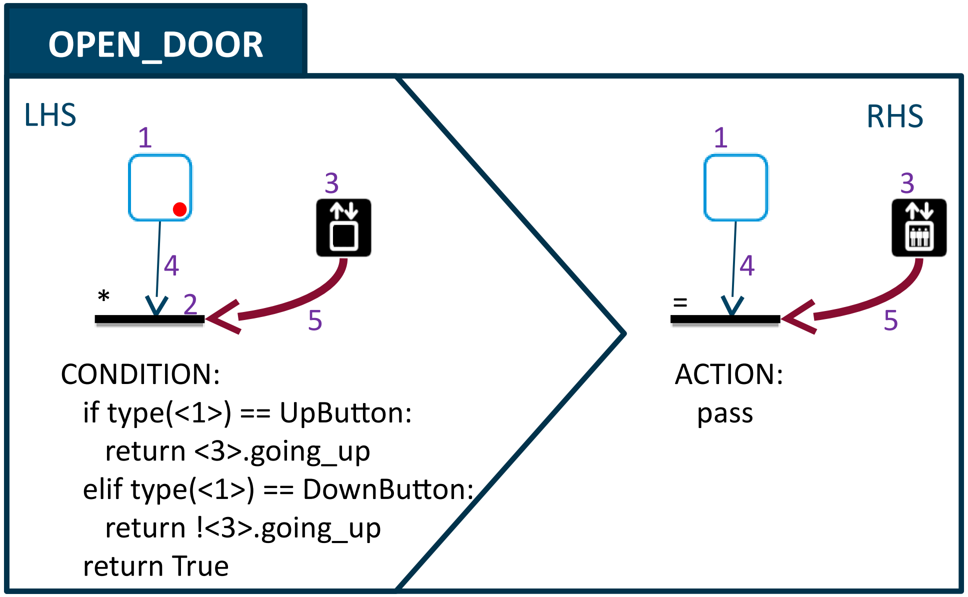
(to specify language semantics)



Rule-Based Transformation (rule)



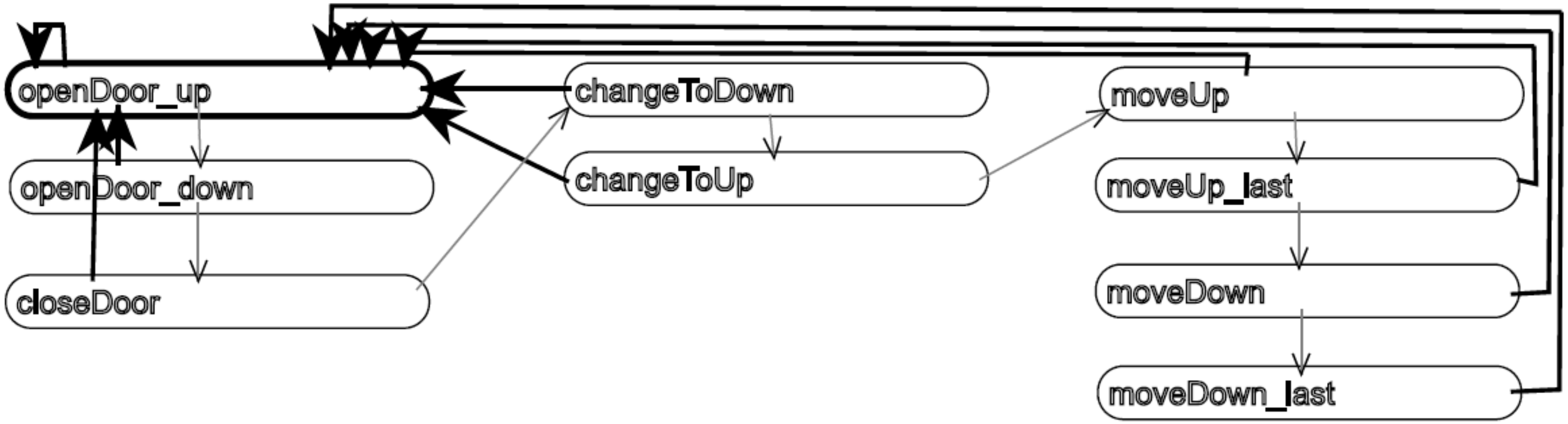
Rule-Based Transformation (rule)



```
CONDITION:
  if type(<1>) == UpButton:
    return <3>.going_up
  elif type(<1>) == DownButton:
    return !<3>.going_up
  return True
```

```
ACTION:
  pass
```

Rule-Based Transformation (schedule)



openDoor_up

openDoor_down

closeDoor

changeToDown

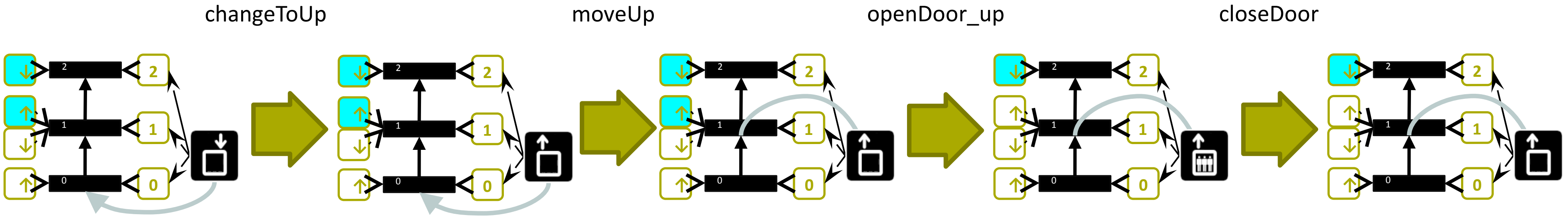
changeToUp

moveUp

moveUp_last

moveDown

moveDown_last



changeToUp

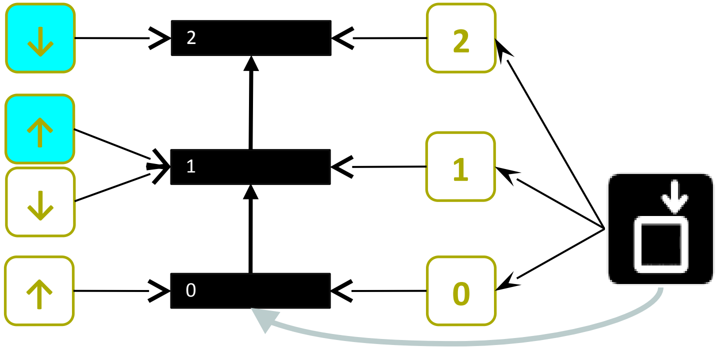
moveUp

openDoor_up

closeDoor

Properties for DSMLs: State of the Art

Design ⊨ Property



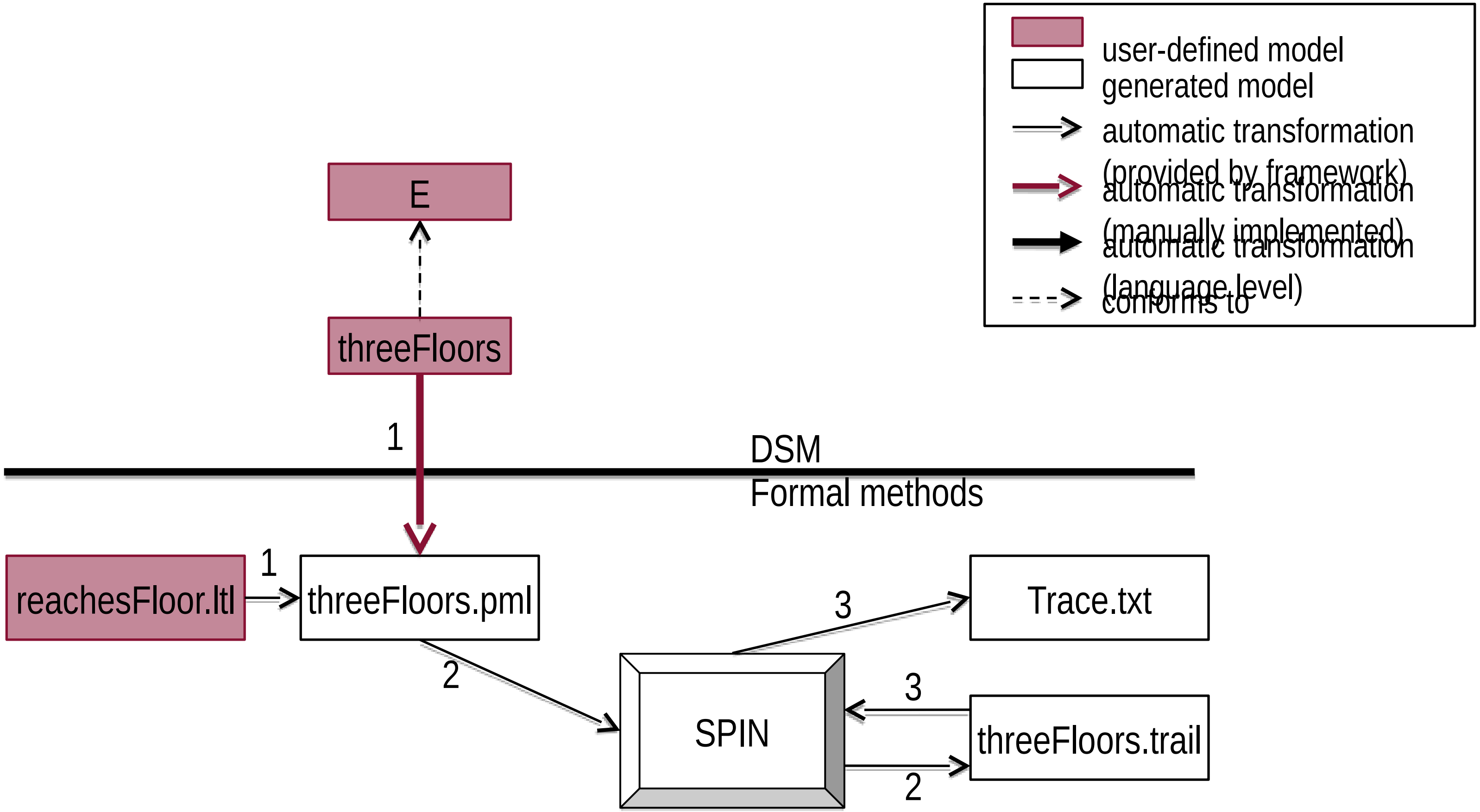
M2T

.pml ⊨

LTL formula

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

State of the Art: Architecture



▨ user-defined model

□ generated model

automatic transformation (provided by framework)

automatic transformation (manually implemented)

automatic transformation (language level)

conforms to

E

threeFloors

1

DSM

Formal methods

reachesFloor.ltl

1

threeFloors.pml

2

3

Trace.txt

SPIN

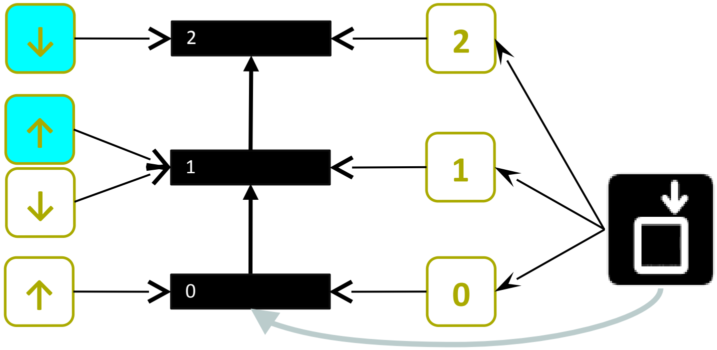
3

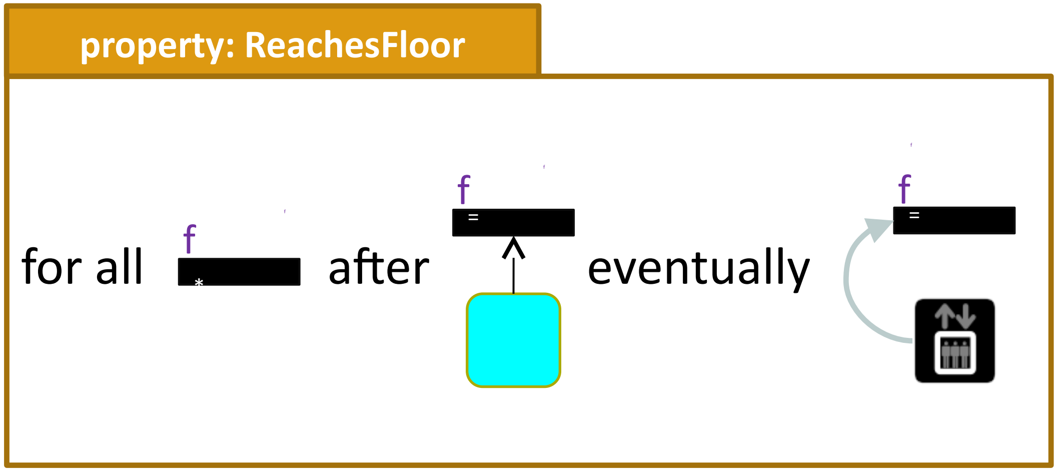
threeFloors.trail

2

Properties for DSMLs: Property DSML

Design ⊨ Property

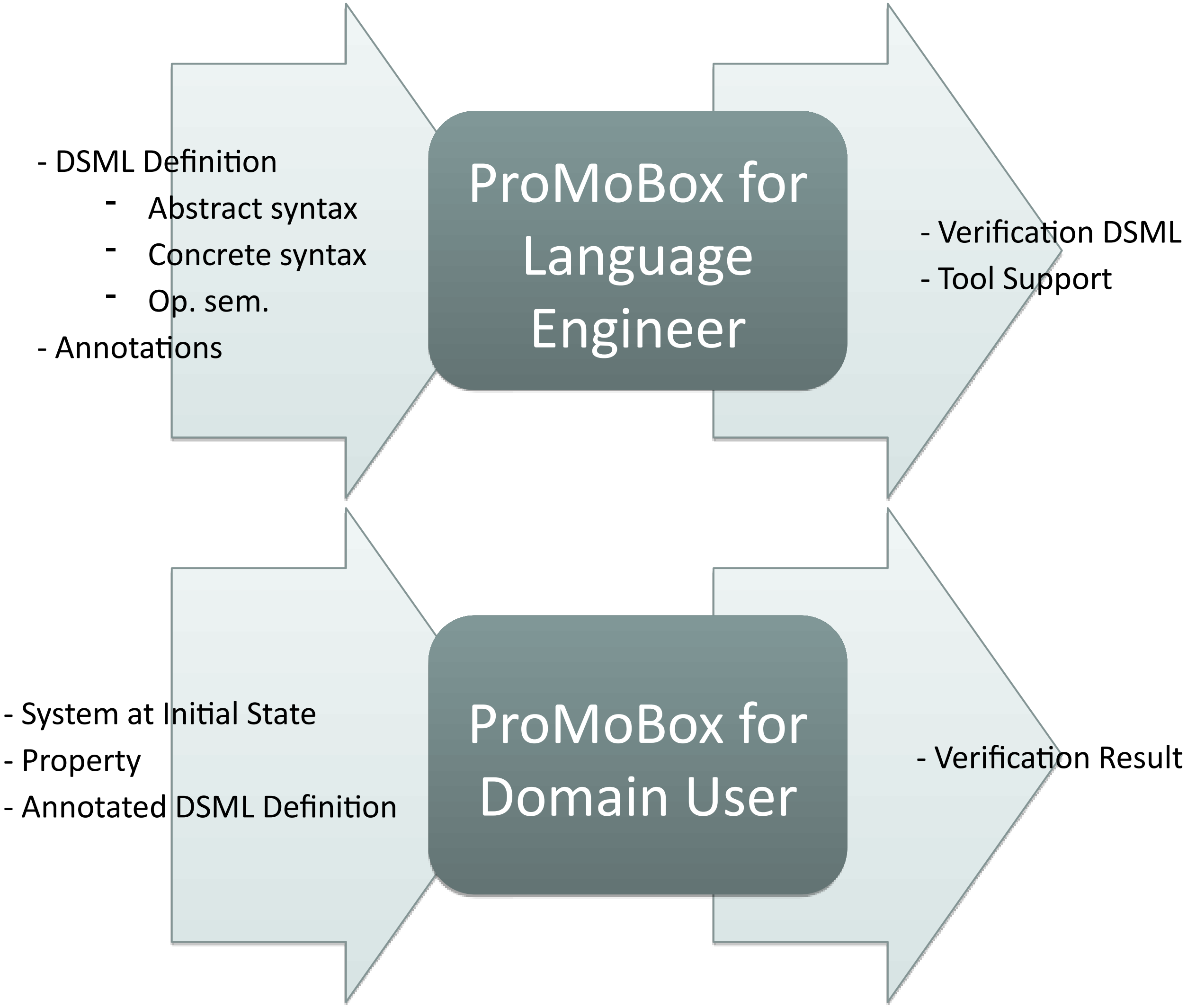




property: ReachesFloor

for all f [redacted] after f = [redacted] eventually f = [redacted]

Verification Support for DSMLs



- DSML Definition
  - Abstract syntax
  - Concrete syntax
  - Op. sem.
- Annotations

ProMoBox for Language Engineer

- Verification DSML
- Tool Support

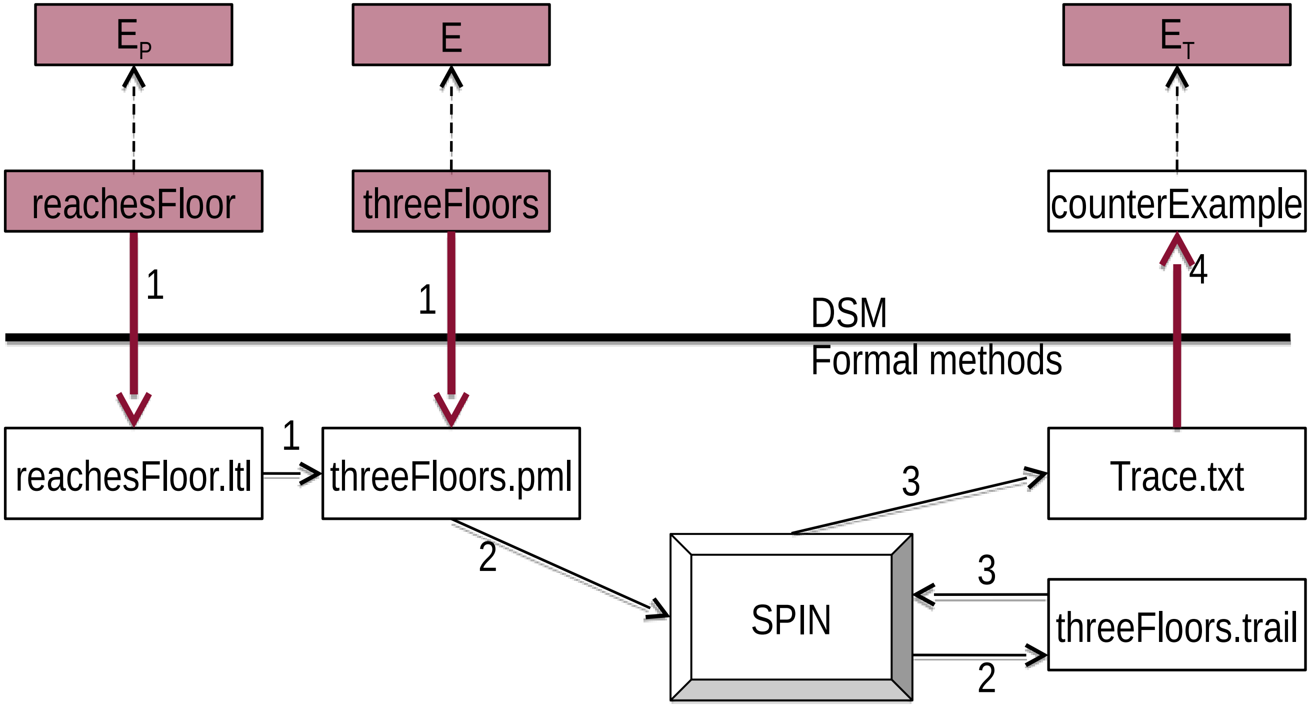
- System at Initial State
- Property
- Annotated DSML Definition

ProMoBox for Domain User

- Verification Result

Pull up to DSL Level

- 🟥 user-defined model
- □ generated model
- → automatic transformation (provided by framework)
- → automatic transformation (manually implemented)
- ➡ automatic transformation (language level)
- - - -> conforms to



modular
modeling
language
engineering
with ProMoBox

5 related languages

🟥 user-defined model

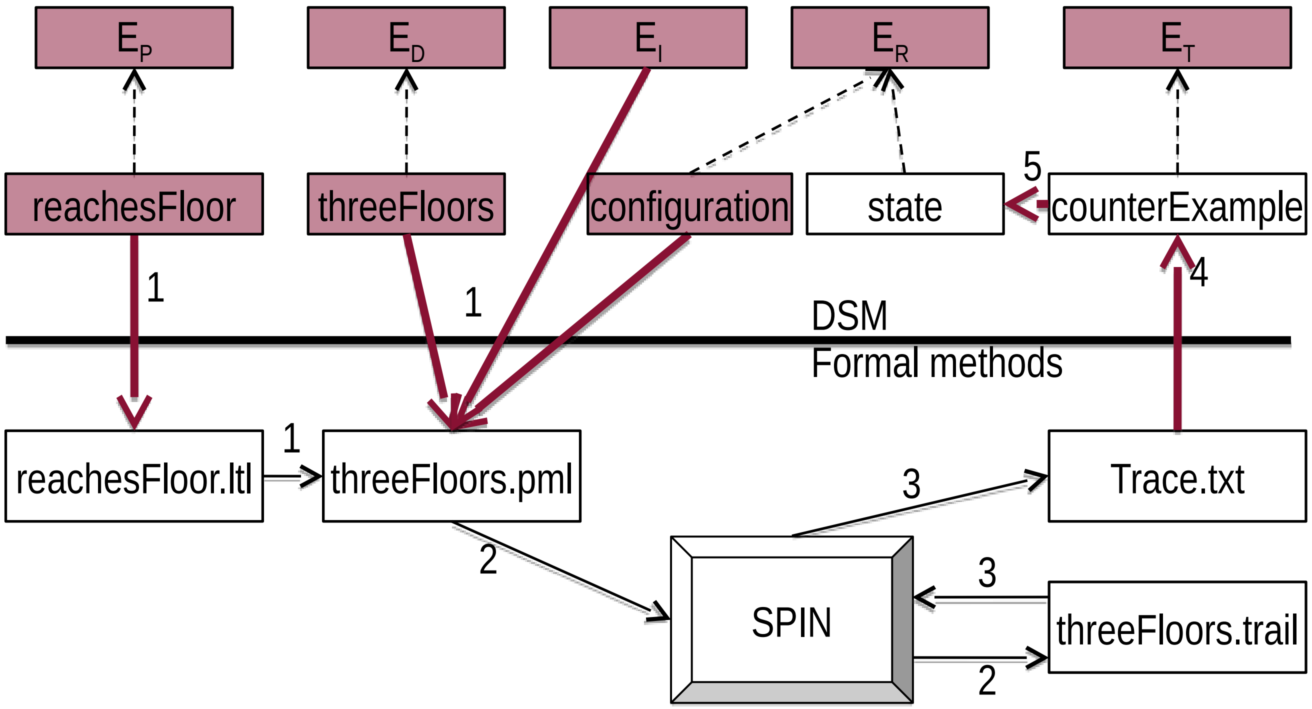
□ generated model

→ automatic transformation (provided by framework)

→ automatic transformation (manually implemented)

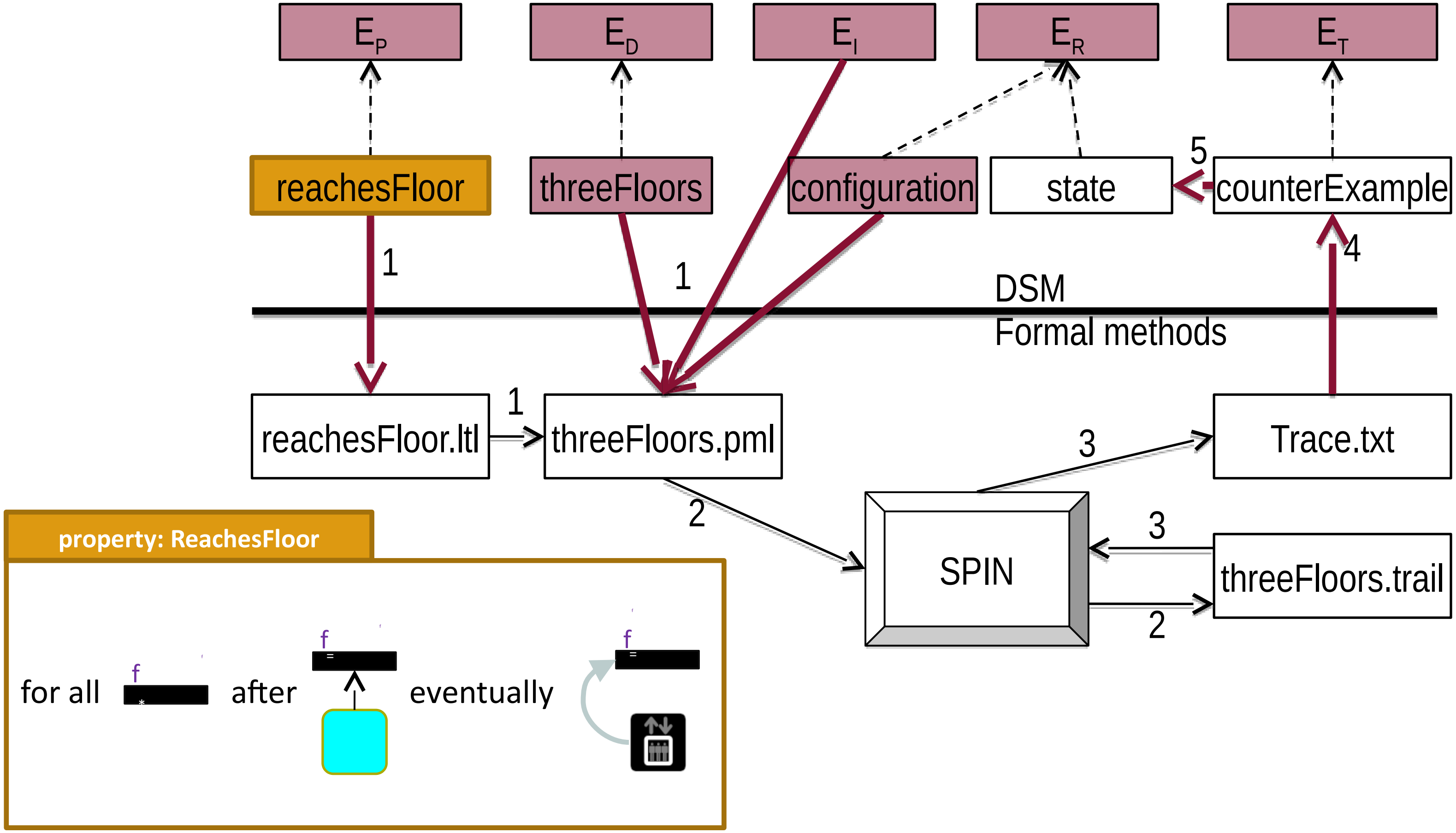
➡ automatic transformation (language level)

⇢ conforms to



Property language

- 🟥 user-defined model
- □ generated model
- → automatic transformation (provided by framework)
- → automatic transformation (manually implemented)
- ➡ automatic transformation (language level)
- - - -> conforms to



Design language

🟥 user-defined model

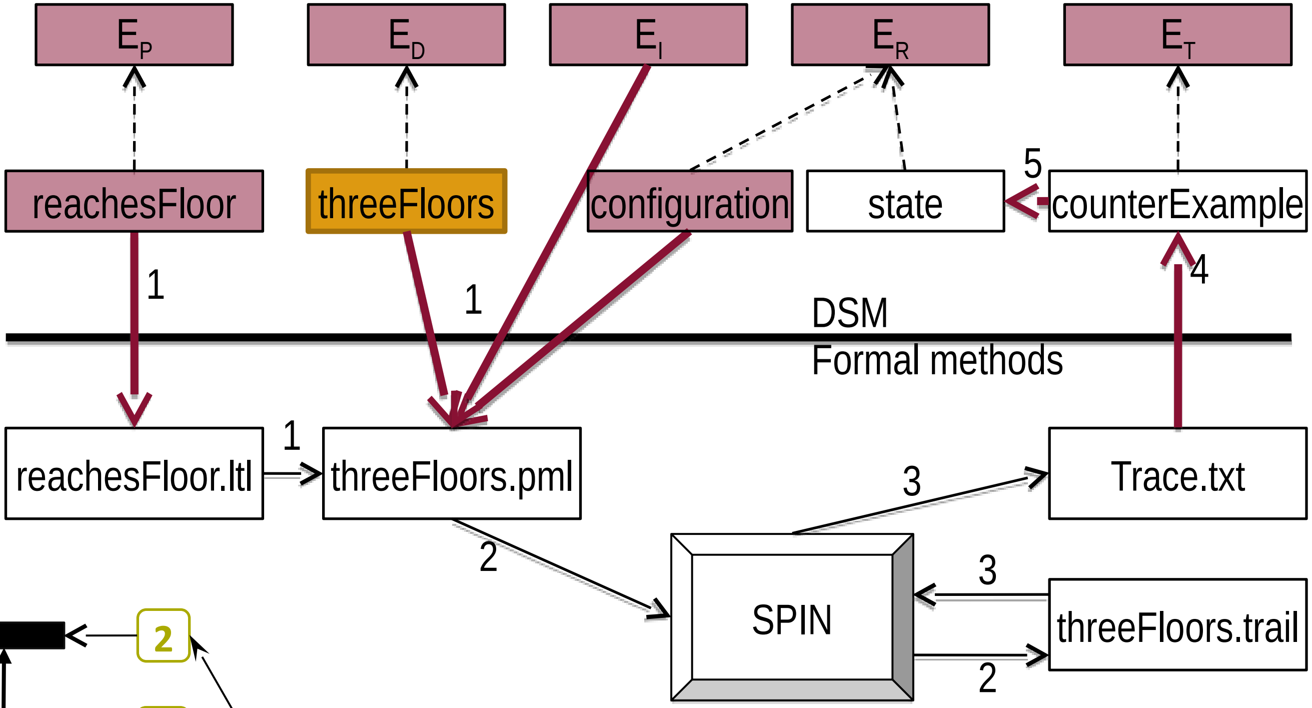
□ generated model

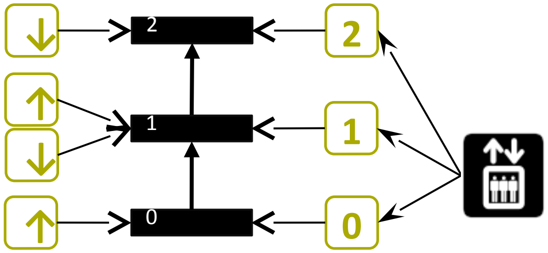
→ automatic transformation (provided by framework)

→ automatic transformation (manually implemented)

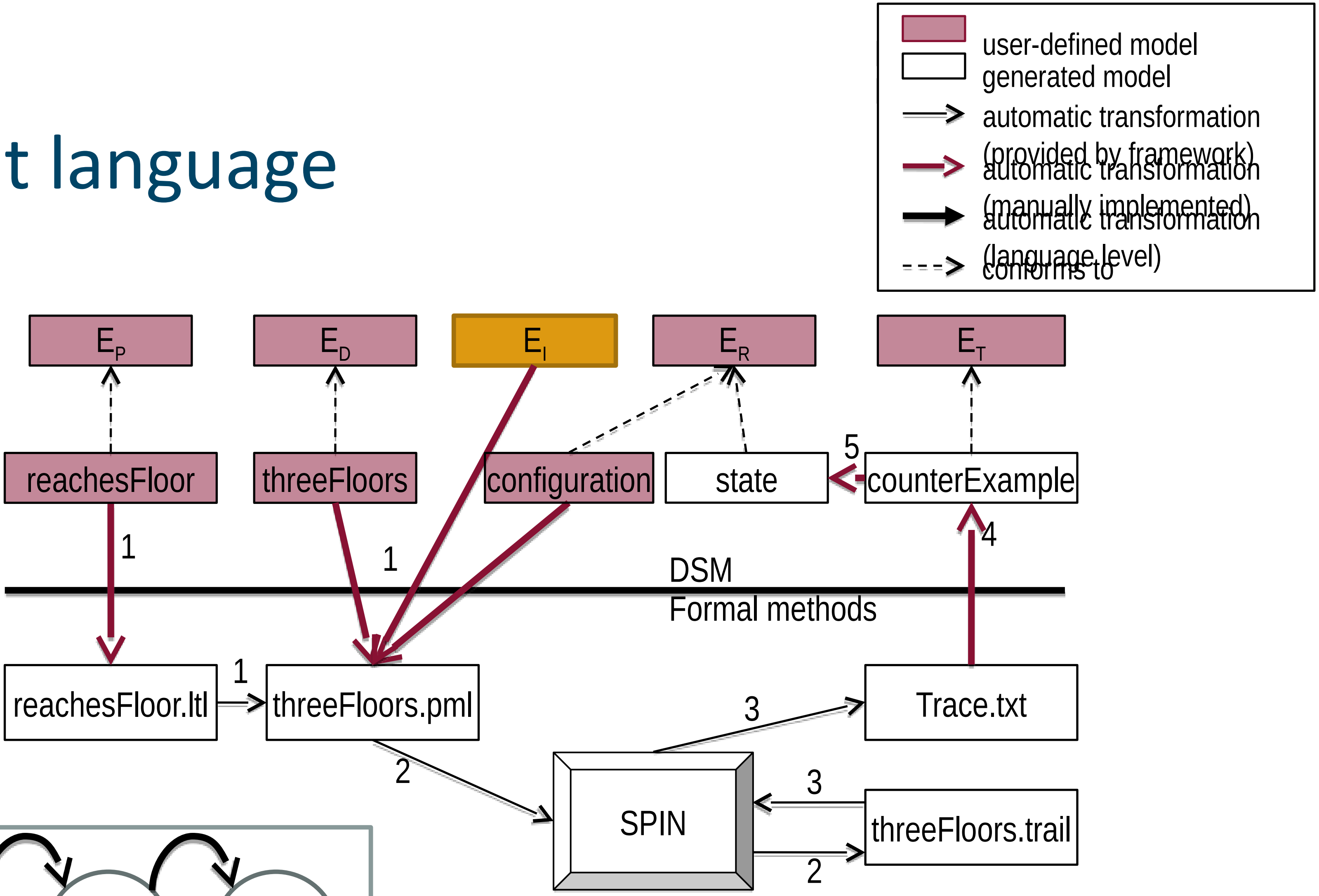
➡ automatic transformation (language level)

--> conforms to





Input language



🟥 user-defined model

□ generated model

automatic transformation (provided by framework)

automatic transformation (manually implemented)

automatic transformation (language level)

conforms to

E_P E_D E_I E_R E_T

reachesFloor threeFloors configuration state counterExample

1 1 5 4

DSM

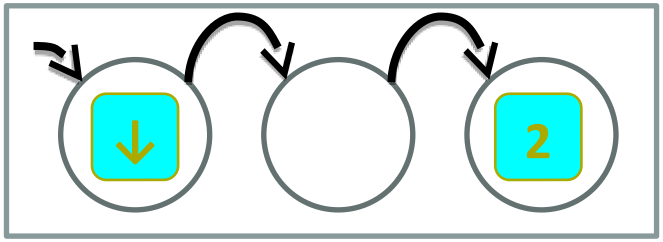
Formal methods

reachesFloor.ltl 1 threeFloors.pml Trace.txt

3 2

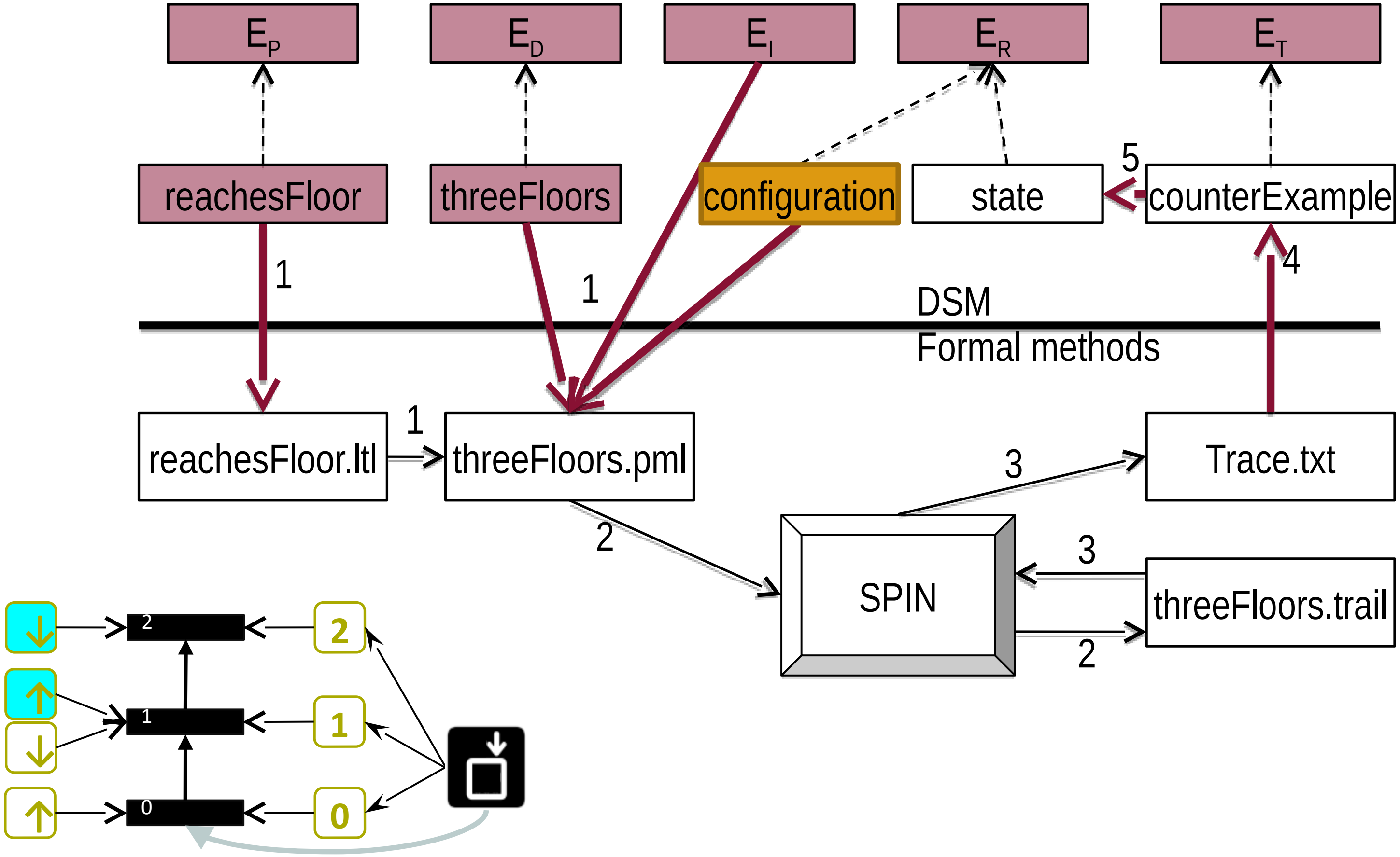
SPIN

3 threeFloors.trail 2



Runtime language

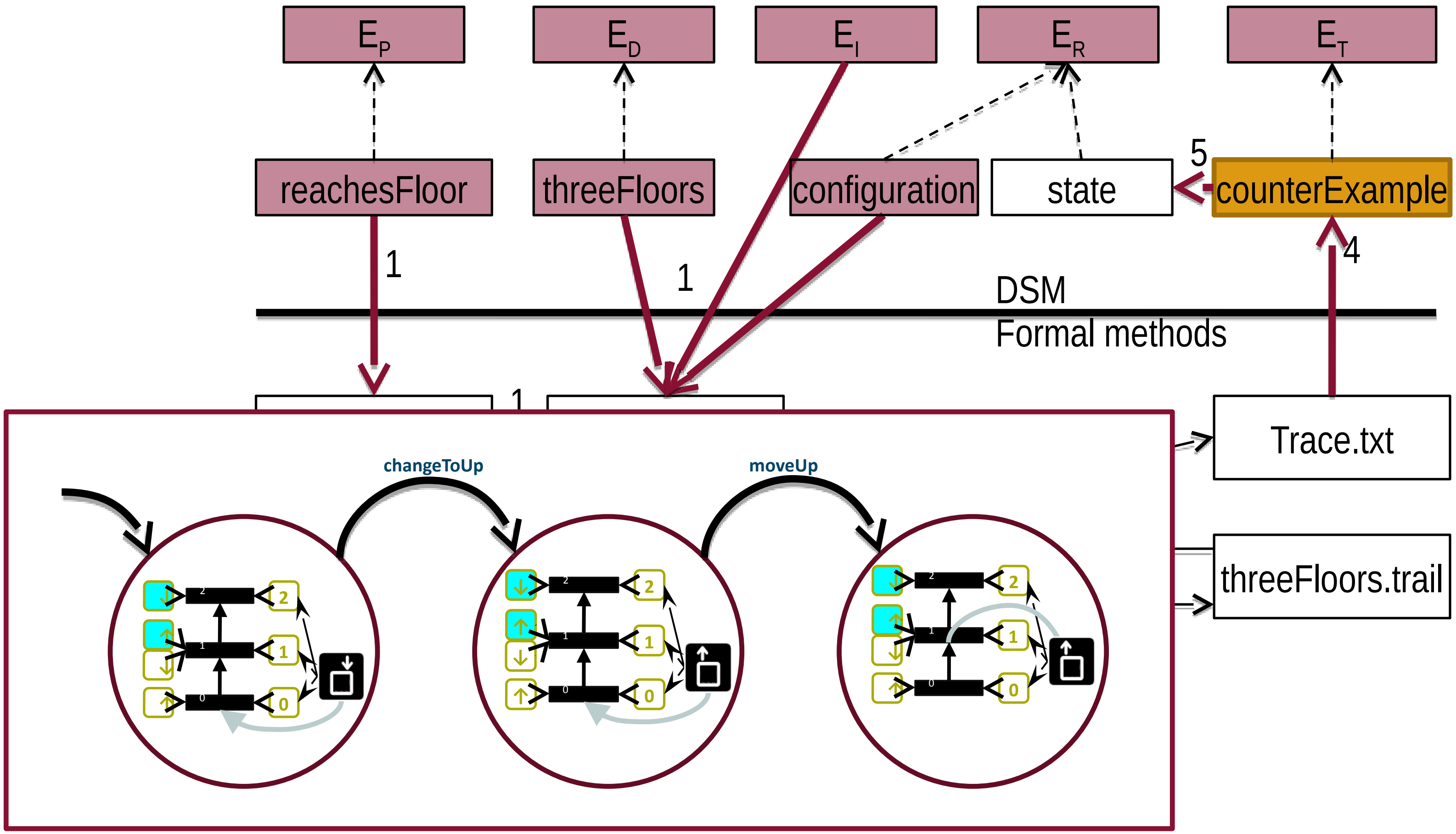
🟥 user-defined model
□ generated model
→ automatic transformation (provided by framework)
→ automatic transformation (manually implemented)
➡ automatic transformation (language level)
- - -> conforms to



E_P
E_D
E_I
E_R
E_T
reachesFloor
threeFloors
configuration
state
5
counterExample
1
1
4
DSM
Formal methods
reachesFloor.ltl
1
threeFloors.pml
3
Trace.txt
2
SPIN
3
threeFloors.trail
2

Trace language

- 🟥 user-defined model
- □ generated model
- → automatic transformation (provided by framework)
- → automatic transformation (manually implemented)
- ➡ automatic transformation (language level)
- - - → conforms to



E_P E_D E_I E_R E_T

reachesFloor

threeFloors

configuration

state

counterExample

5

1

1

4

DSM

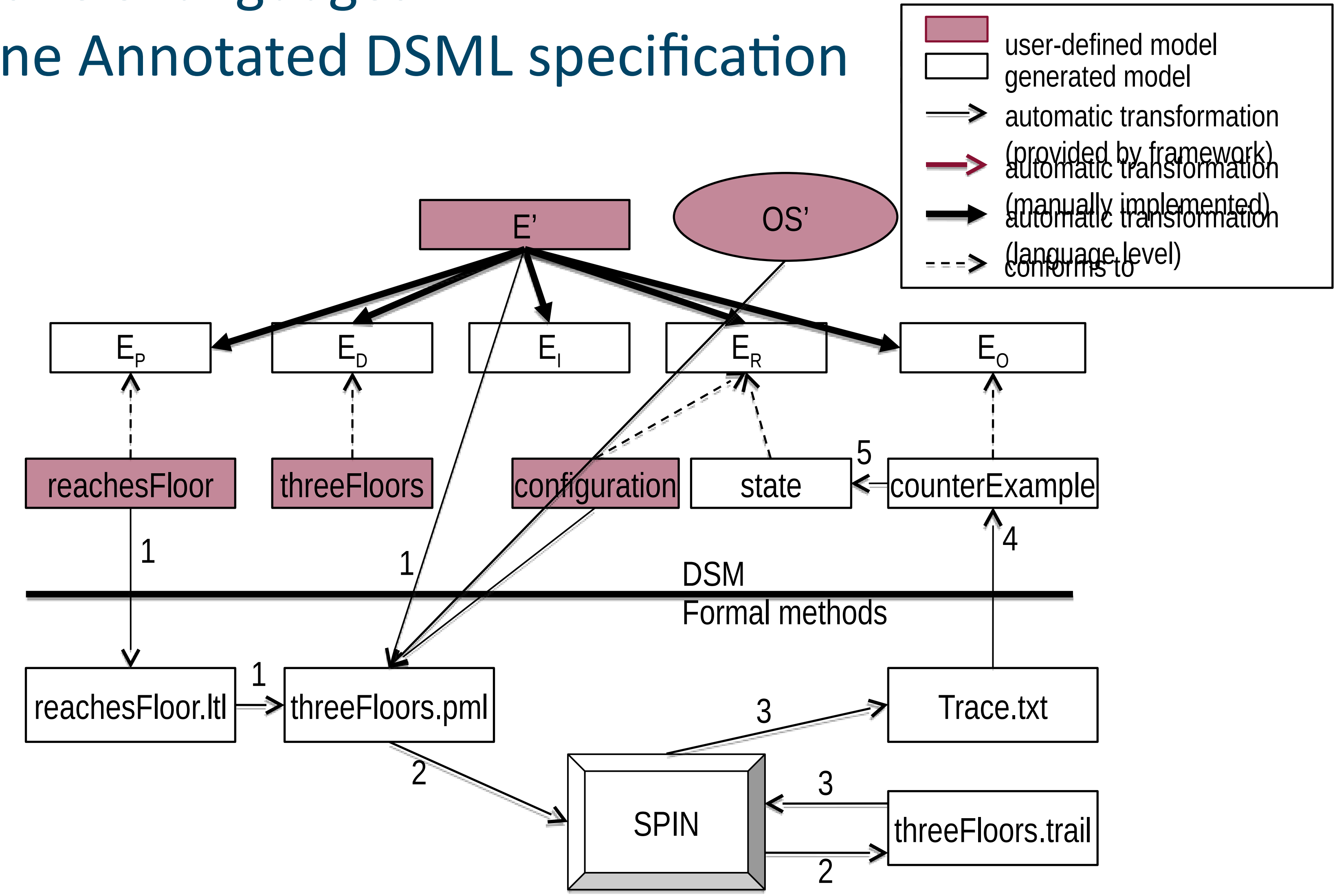
Formal methods

1

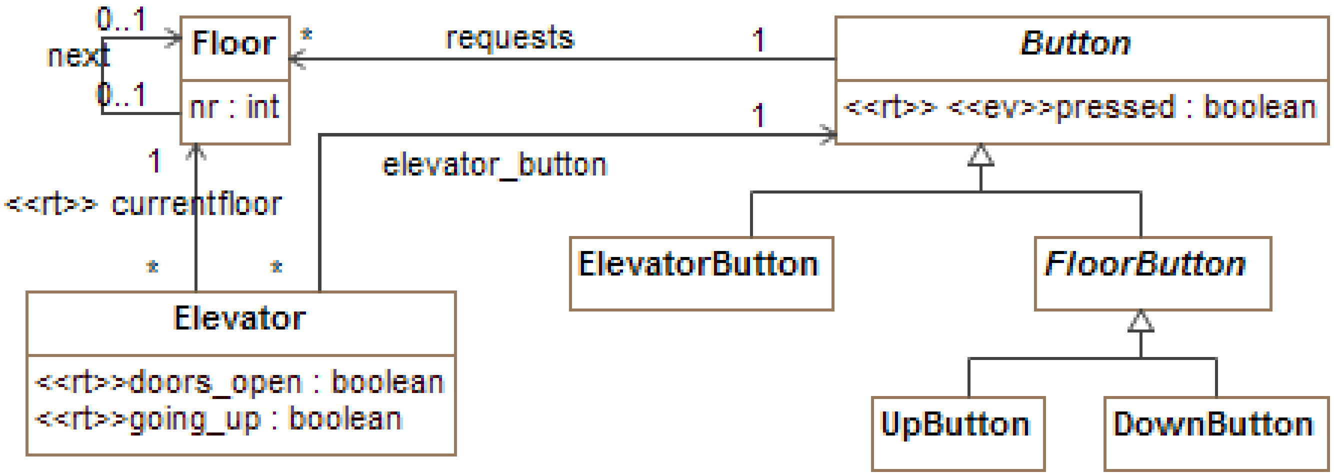
Trace.txt

threeFloors.trail

Synthesize 5 languages from one Annotated DSML specification

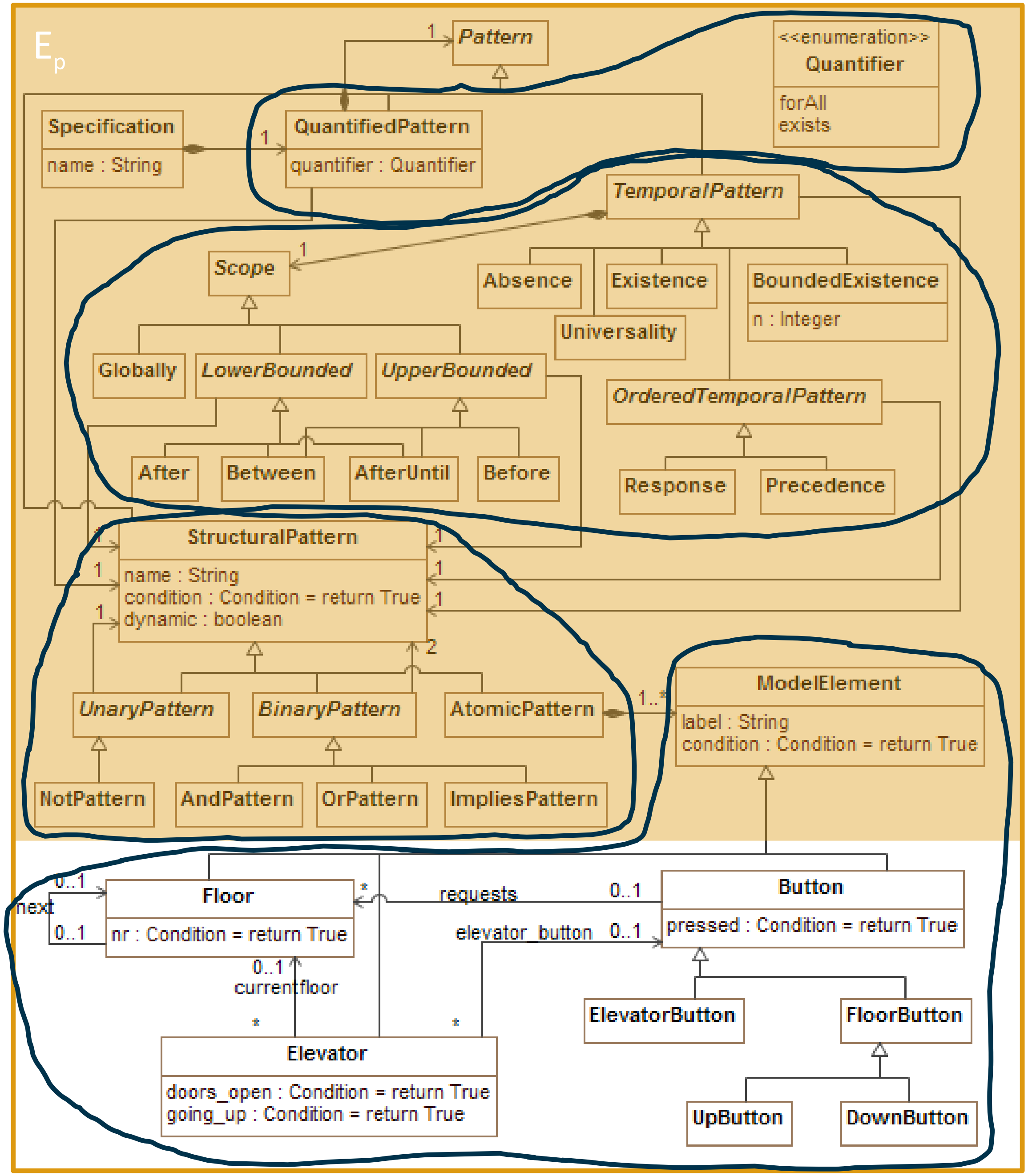


Annotated metamodel



Annotation	Design	Runtime	Input	Output	Property
	x	x		x	x
<<rt>>		x		x	x
<<ev>>			x		
<<tr>>	x	x	x	x	x

E_p



Quantification

Matthew B. Dwyer, George S. Avrunin, James C. Corbett. Patterns in Property Specifications for Finite-State Verification. ICSE 1999: 411-420

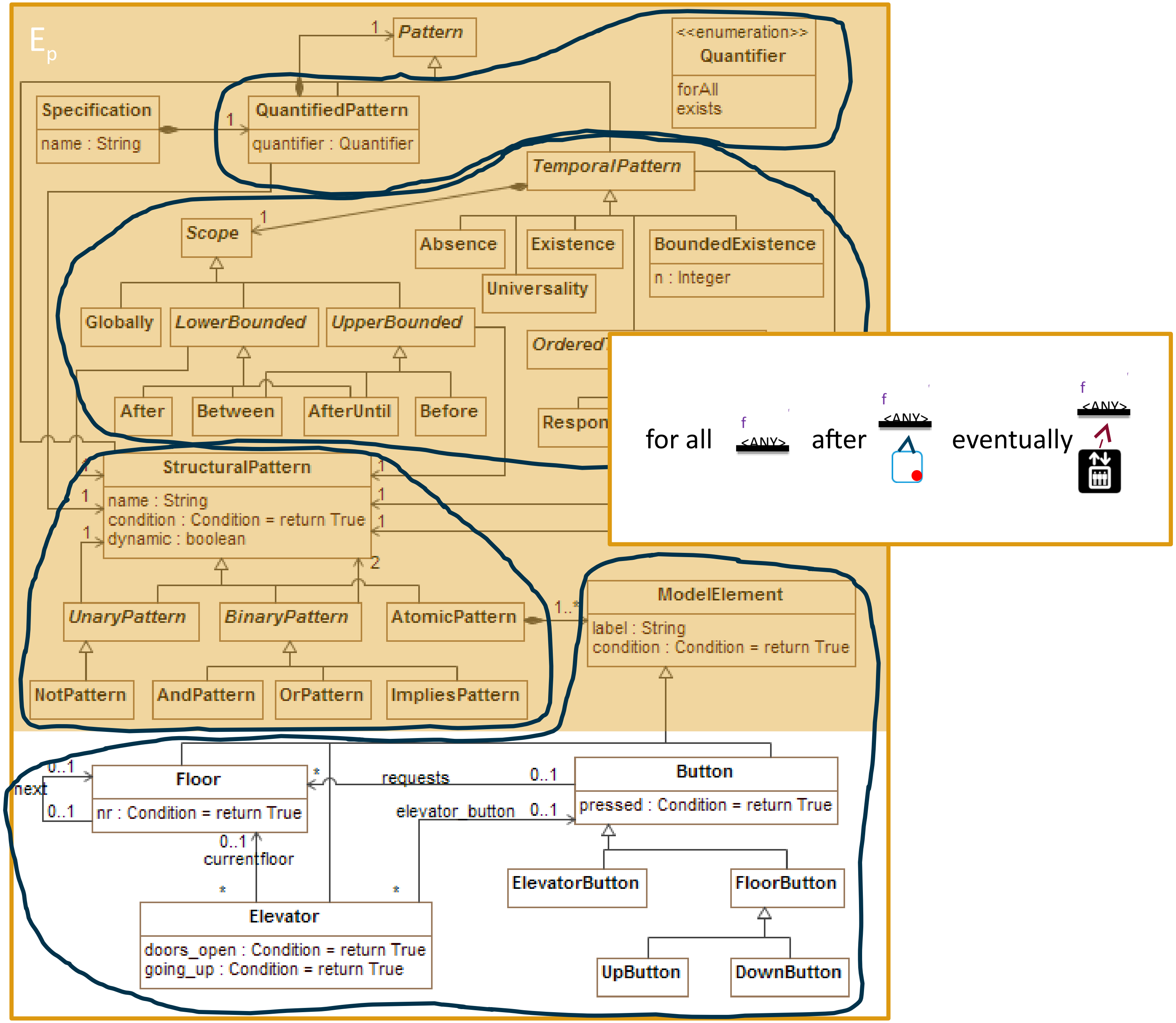
Temporal Patterns

Structural Patterns

DSL-specific Elements

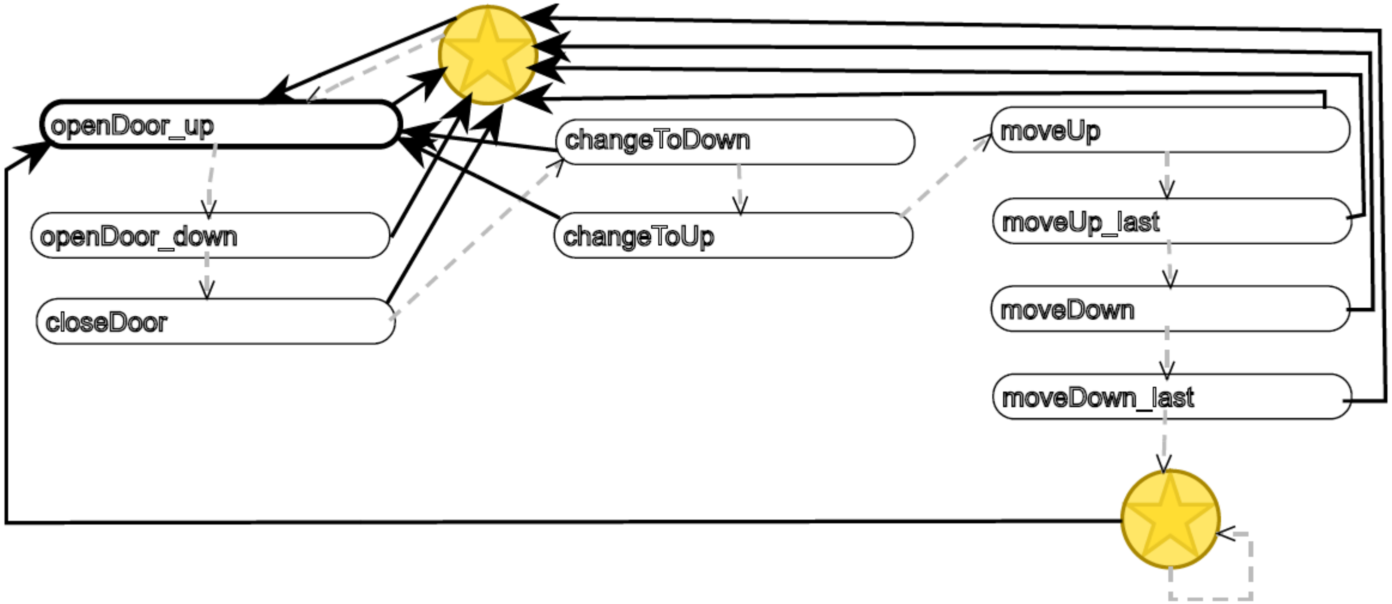
modular
modeling
language
engineering
with ProMoBox

E_p



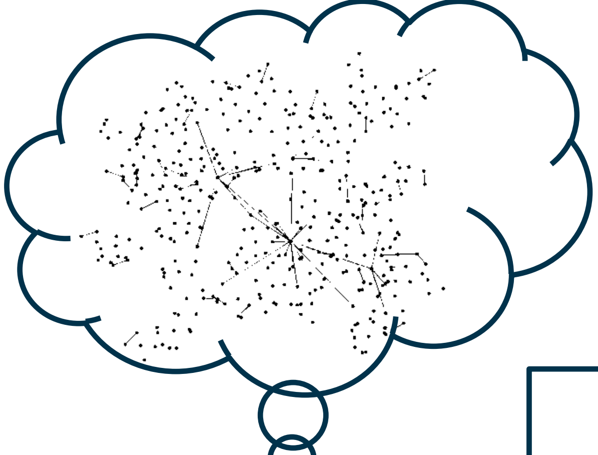
E_p
Pattern
1
<<enumeration>>
Quantifier
forAll
exists
Specification
name : String
1
QuantifiedPattern
quantifier : Quantifier
TemporalPattern
Scope
1
Absence
Existence
BoundedExistence
n : Integer
Universality
Globally
LowerBounded
UpperBounded
Ordered
After
Between
AfterUntil
Before
Respon
StructuralPattern
name : String
condition : Condition = return True
dynamic : boolean
1
1
1
1
1
1
2
UnaryPattern
BinaryPattern
AtomicPattern
1..*
NotPattern
AndPattern
OrPattern
ImpliesPattern
ModelElement
label : String
condition : Condition = return True
Floor
nr : Condition = return True
0..1
next
0..1
*
requests
0..1
Button
pressed : Condition = return True
elevator_button
0..1
0..1
currentfloor
*
*
Elevator
doors_open : Condition = return True
going_up : Condition = return True
ElevatorButton
FloorButton
UpButton
DownButton
for all
f
<ANY>
after
f
<ANY>
eventually
f
<ANY>

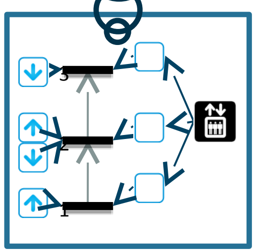
Annotated Operational Semantics

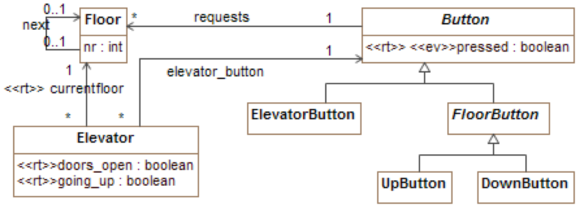


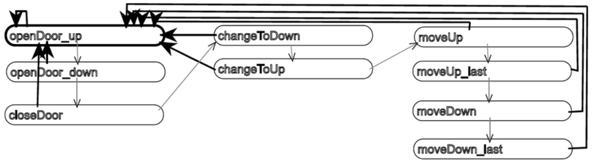
Specify:

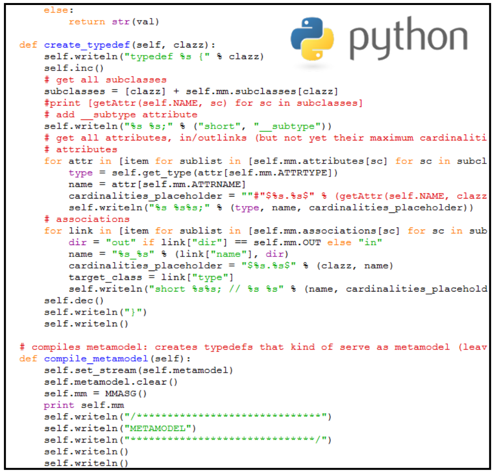
- **Input step**: when can an input event occur?
- **Output step**: what does a trace look like?











```python
    else:
        return str(val)

def create_typedef(self, clazz):
    self.writeln("typedef %s {" % clazz)
    self.inc()
    # get all subclasses
    subclasses = [clazz] + self.mm.subclasses[clazz]
    #print [getAttr(self.NAME, sc) for sc in subclasses]
    # add __subtype attribute
    self.writeln("%s %s;" % ("short", "__subtype"))
    # get all attributes, in/outlinks (but not yet their maximum cardinaliti
    # attributes
    for attr in [item for sublist in [self.mm.attributes[sc] for sc in subcl
        type = self.get_type(attr[self.mm.ATTRTYPE])
        name = attr[self.mm.ATTRNAME]
        cardinalities_placeholder = ""#"$%s.%s$" % (getAttr(self.NAME, clazz
        self.writeln("%s %s%s;" % (type, name, cardinalities_placeholder))
    # associations
    for link in [item for sublist in [self.mm.associations[sc] for sc in sub
        dir = "out" if link["dir"] == self.mm.OUT else "in"
        name = "%s_%s" % (link["name"], dir)
        cardinalities_placeholder = "$%s.%s$" % (clazz, name)
        target_class = link["type"]
        self.writeln("short %s%s; // %s %s" % (name, cardinalities_placehold
    self.dec()
    self.writeln("}")
    self.writeln()

# compiles metamodel: creates typedefs that kind of serve as metamodel (leav
def compile_metamodel(self):
    self.set_stream(self.metamodel)
    self.metamodel.clear()
    self.mm = MMASG()
    print self.mm
    self.writeln("/******************************")
    self.writeln("METAMODEL")
    self.writeln("******************************/")
    self.writeln()
    self.writeln()
```

Generic code generator

Optimizations!

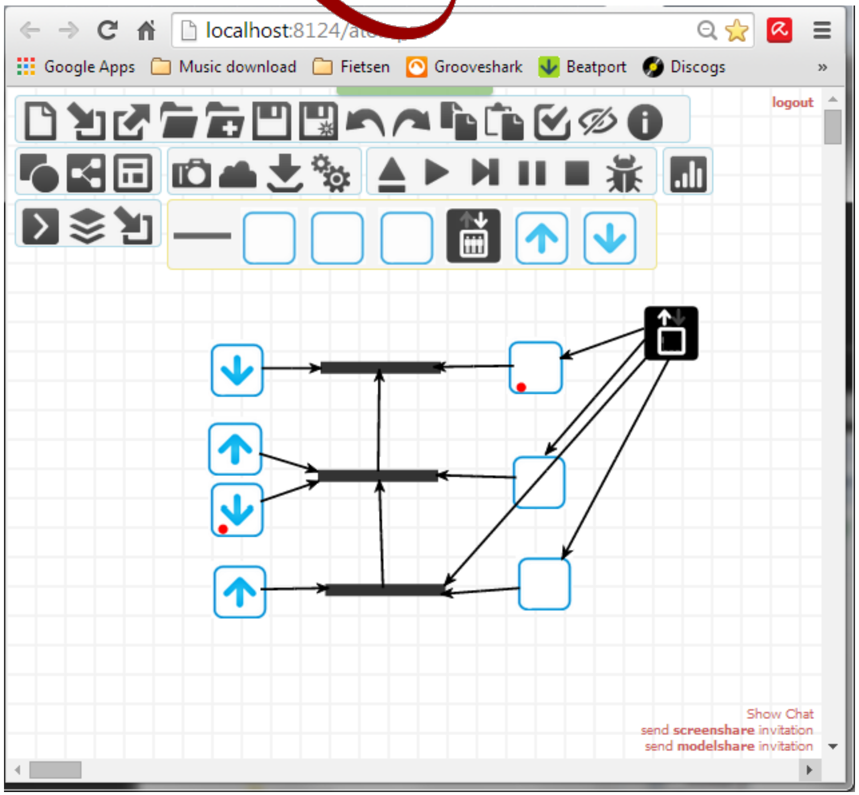
Can use valuable system- and property-specific information

```
    MOVEUP_schedule:
    if
      :: (success == 1) -> // when successful
        rule = moveup_;
        nextrule = environmentstep1_;
        goto OUTPUT;
      :: else -> // when not applicable
        nextrule = moveup_last_;
        goto NEXTSTEP;
    fi;
    CLOSEDOOR_schedule:
    if
      :: (success == 1) -> // when successful
        rule = closedoor_;
        nextrule = environmentstep1_;
        goto OUTPUT;
      :: else -> // when not applicable
        nextrule = moveup_;
        goto NEXTSTEP;
    fi;
    ENVIRONMENTSTEP2_schedule:
    if
      :: (success == 1) -> // when successful
        rule = environmentstep2_;
        nextrule = opendoor_up_;
        goto OUTPUT;
      :: else -> // when not applicable
        nextrule = environmentstep2_;
        goto NEXTSTEP;
    fi;
    ENVIRONMENTSTEP1_schedule:
    if
      :: (success == 1) -> // when successful
        rule = environmentstep1_;
        nextrule = opendoor_up_;
        goto OUTPUT;
      :: else -> // when not applicable
        nextrule = opendoor_up_;
        goto NEXTSTEP;
    fi;
 MOVEUP:
    success = 0;
    // looking for node elevator5 type Elevator, Elevator from None by 
    // looking for node floor1 type Floor, Floor from elevator5 by foll
    // looking for node floor2 type Floor, Floor from floor1 by followi
    // looking for node button3 type Button, Button from None by follow
    // looking for node floor0 type Floor, Floor from button3 by follow
    elevator5 = 0; // this is the only Elevator so index must be 0
    button3_max = 7;
    button3_index_map[0] = 0;
    button3_index_map[1] = 1;
    button3_index_map[2] = 2;
    button3_index_map[3] = 3;
    button3_index_map[4] = 4;
    button3_index_map[5] = 5;
    button3_index_map[6] = 6;
    do
    :: ((elevator5 >= 0) && (s.elevator_[elevator5].__subtype == Elevat
(r.elevator_[elevator5].going_up == 1) && r.elevator_[elevator5].curren
(s.floor_[r.elevator_[elevator5].currentfloor_out].__subtype == FloorTy
(s.floor_[r.elevator_[elevator5].currentfloor_out].next_out != r.elevat
(s.floor_[s.floor_[r.elevator_[elevator5].currentfloor_out].next_out]._
      if
      :: ((button3_max > 0) && (button3_index_map[0] >= 0) && (s.button
s.button_[button3_index_map[0]].__subtype == DownButtonType || s.button
(r.button_[button3_index_map[0]].pressed == 1) && (elevator5 >= 0) && (
(r.elevator_[elevator5].doors_open == 0) && (r.elevator_[elevator5].goi
(r.elevator_[elevator5].currentfloor_out >= 0) && (s.floor_[r.elevator_
(s.floor_[r.elevator_[elevator5].currentfloor_out].next_out >= 0) && (s
r.elevator_[elevator5].currentfloor_out) && (s.floor_[s.floor_[r.elevat
(s.button_[button3_index_map[0]].requests_out >= 0) && (s.button_[butto
s.floor_[r.elevator_[elevator5].currentfloor_out].next_out) && (s.butto
r.elevator_[elevator5].currentfloor_out) && (s.floor_[s.button_[button3
(s.floor_[s.button_[button3_index_map[0]].requests_out].nr > s.floor_[r
      :: ((button3_max > 1) && (button3_index_map[1] >= 0) && (s.button
```

DSM implementation

- "A Tool for Multi-Paradigm Modeling"
- Successor to AToM^3
- Cloud- and browser based
- Model everything!
  - At the most appropriate level(s) of abstraction
  - Using the most appropriate formalism(s)

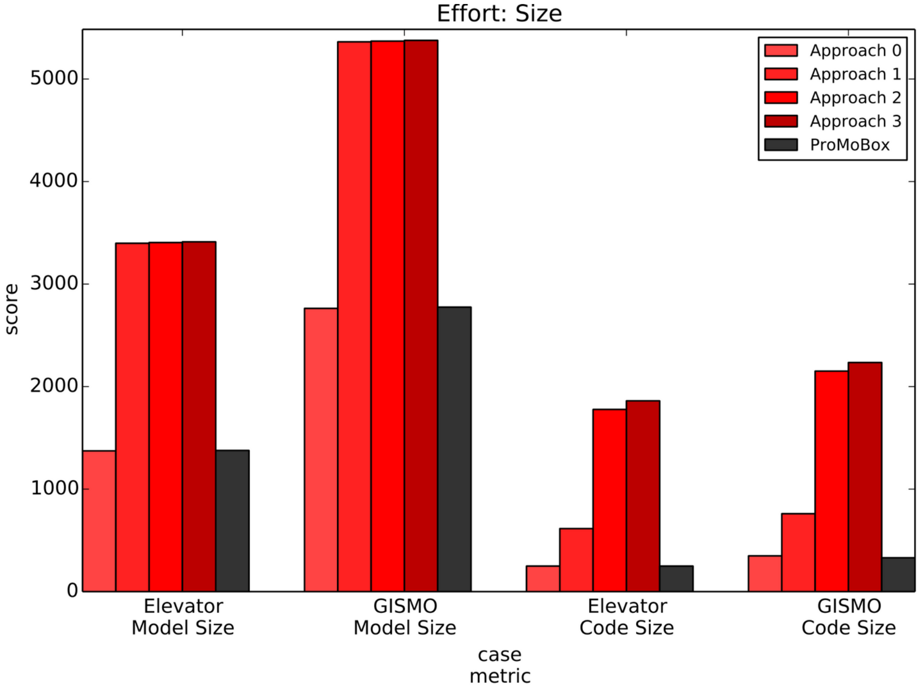
AToMPM

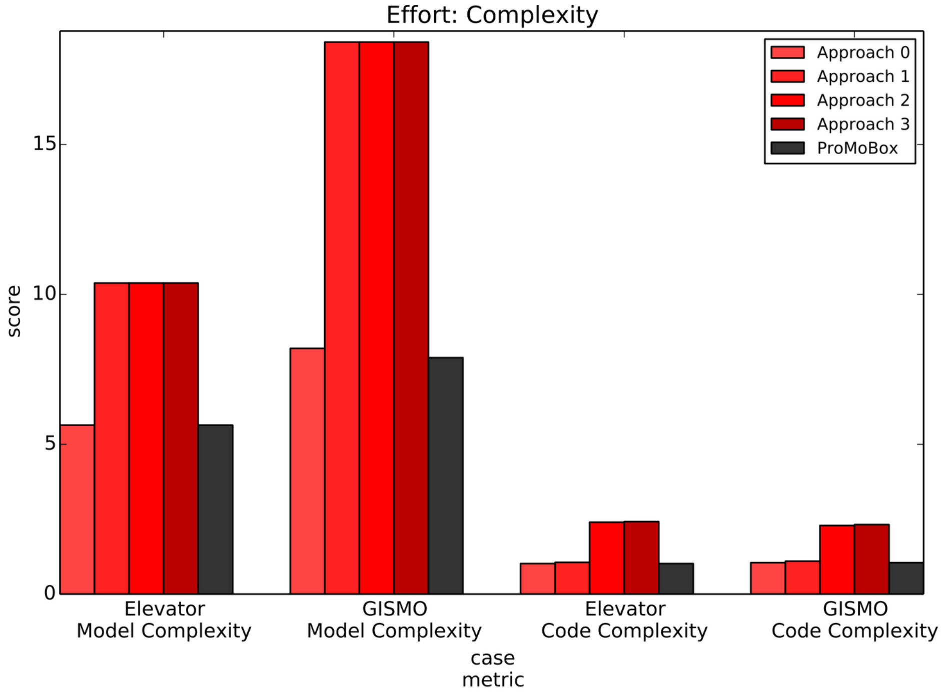


Evaluation (modelling effort)

In comparison with existing approaches:

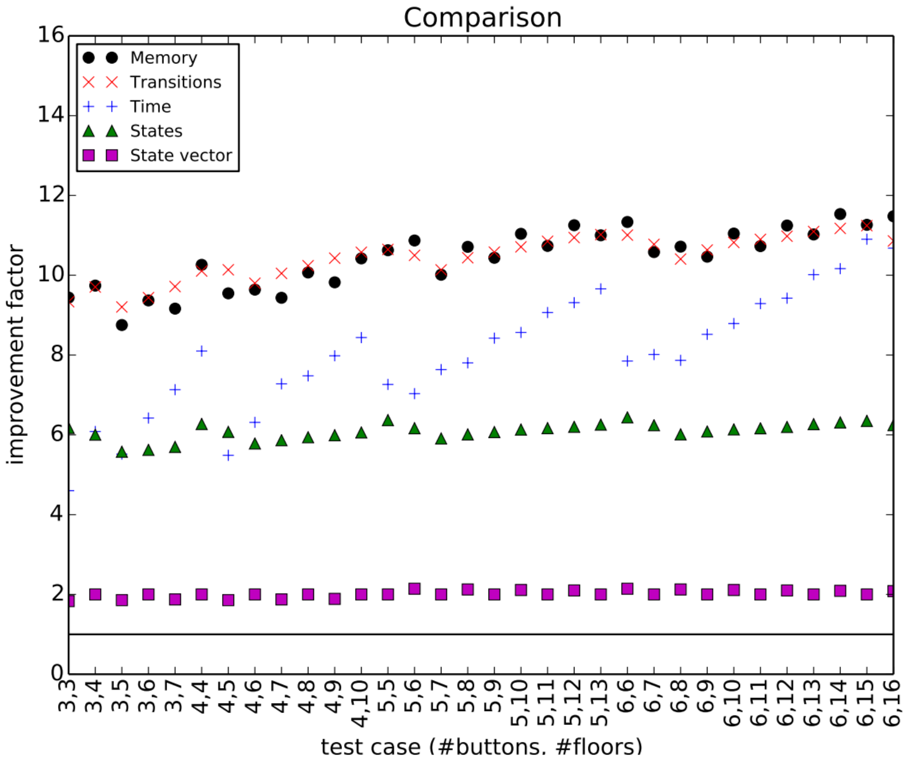
- ***Approach 1***: no DSML for properties is available. Instead, properties are directly modelled in logic, but there is a mapping to a formal language
- ***Approach 2***: a DSML for properties is created including a mapping to a verification backbone, but no counterexample parsing is supported
- ***Approach 3***: a DSML for properties is created including a mapping to and counterexample parsing from a verification backbone. This is in fact the only approach that offers the same functionality as *ProMoBox*





Evaluation (model checking performance)

Elevator case study, in comparison with an adapted elevator implementation from the literature [merz08]



[merz08] Stephan Merz. An introduction to model checking. In Stephan Merz and Nicolas Navet, editors, Modeling and Verification of Real-Time Systems - Formalisms and Software Tools, pages 81–116. ISTE Publishing, 2008.

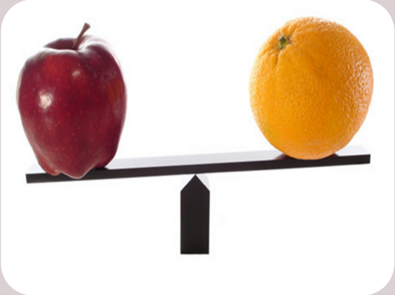
State Space Explosion Problem

Testing in DSM?

Specify initial state → Run operational semantics → Inspect result

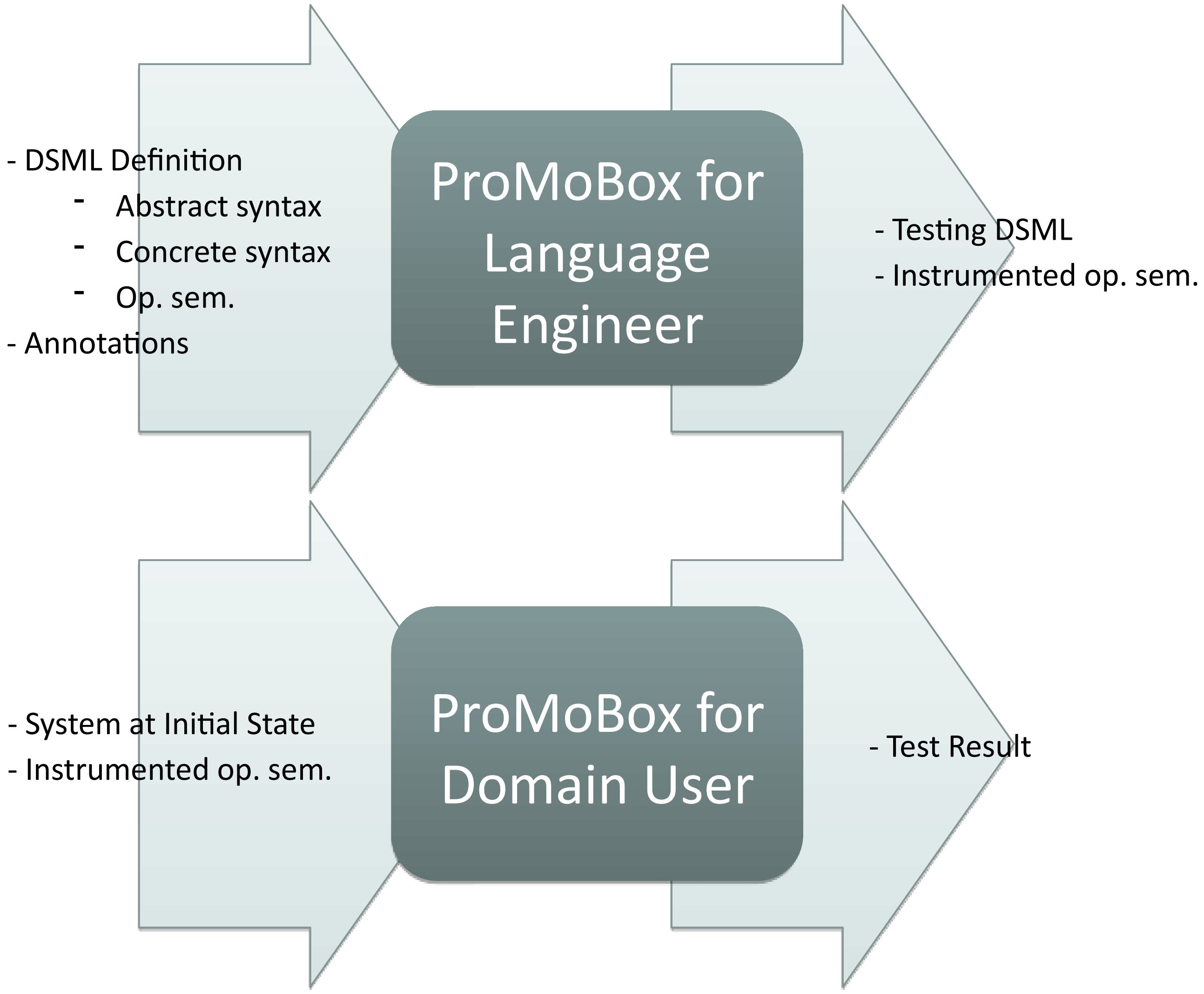
Regression testing?

Oracle specification?



Comparison?

Test Support for DSMLs



- DSML Definition
  - Abstract syntax
  - Concrete syntax
  - Op. sem.
- Annotations

ProMoBox for Language Engineer

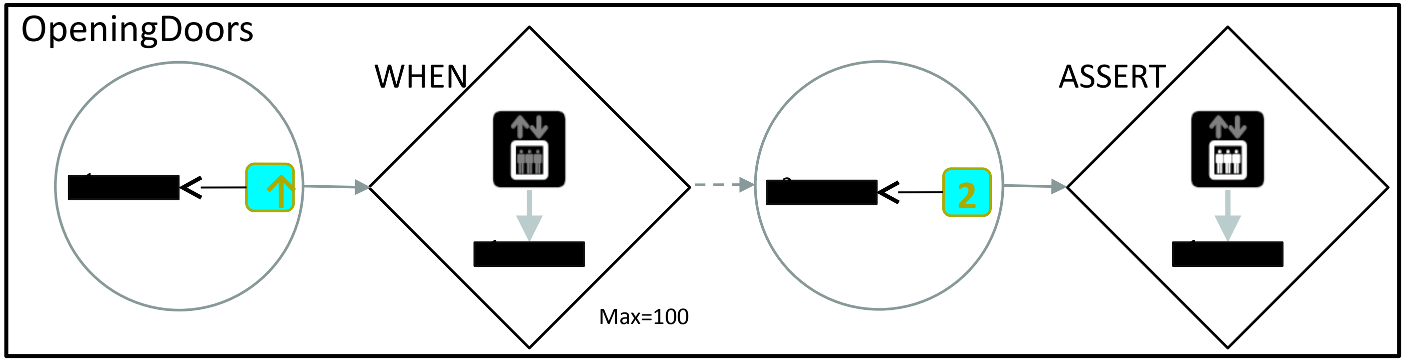
- Testing DSML
- Instrumented op. sem.

- System at Initial State
- Instrumented op. sem.

ProMoBox for Domain User

- Test Result

Test case in Testing models in a DSML (caveat: not testing the DSML)



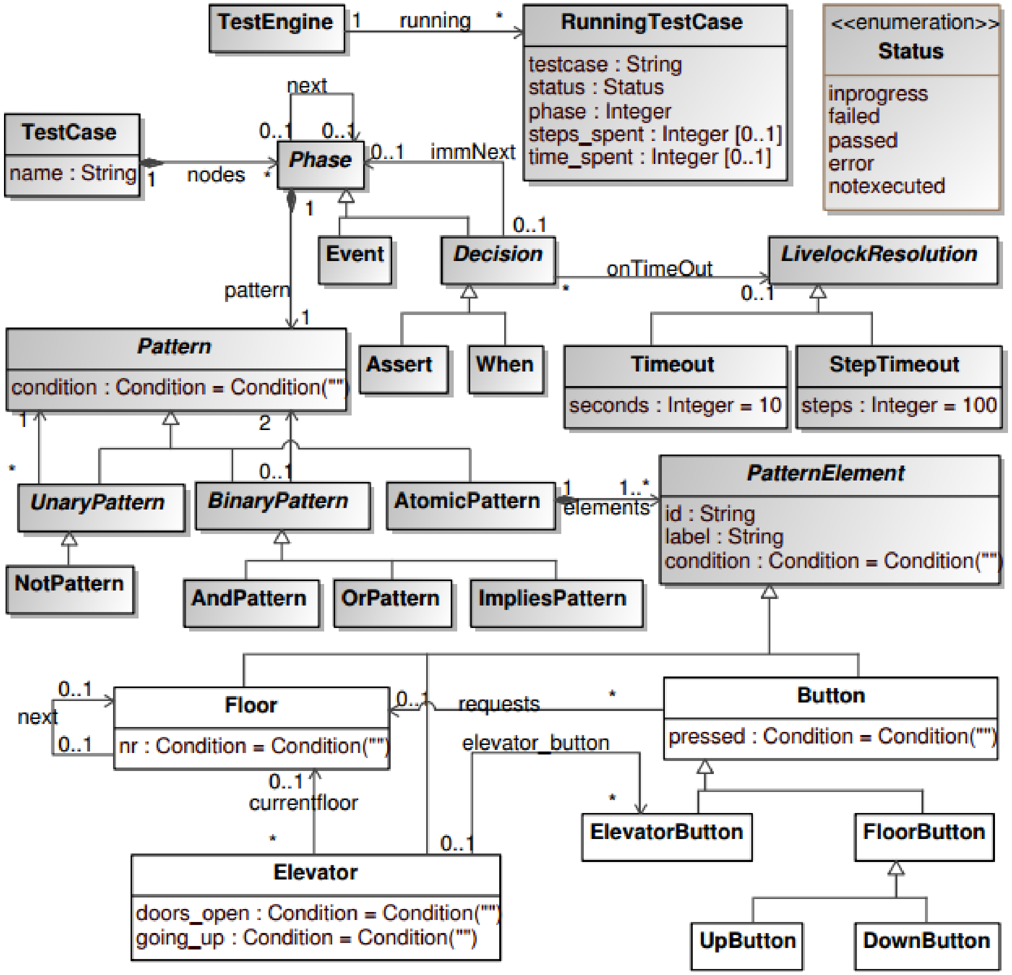
Inspired by unit testing

Domain-specific syntax

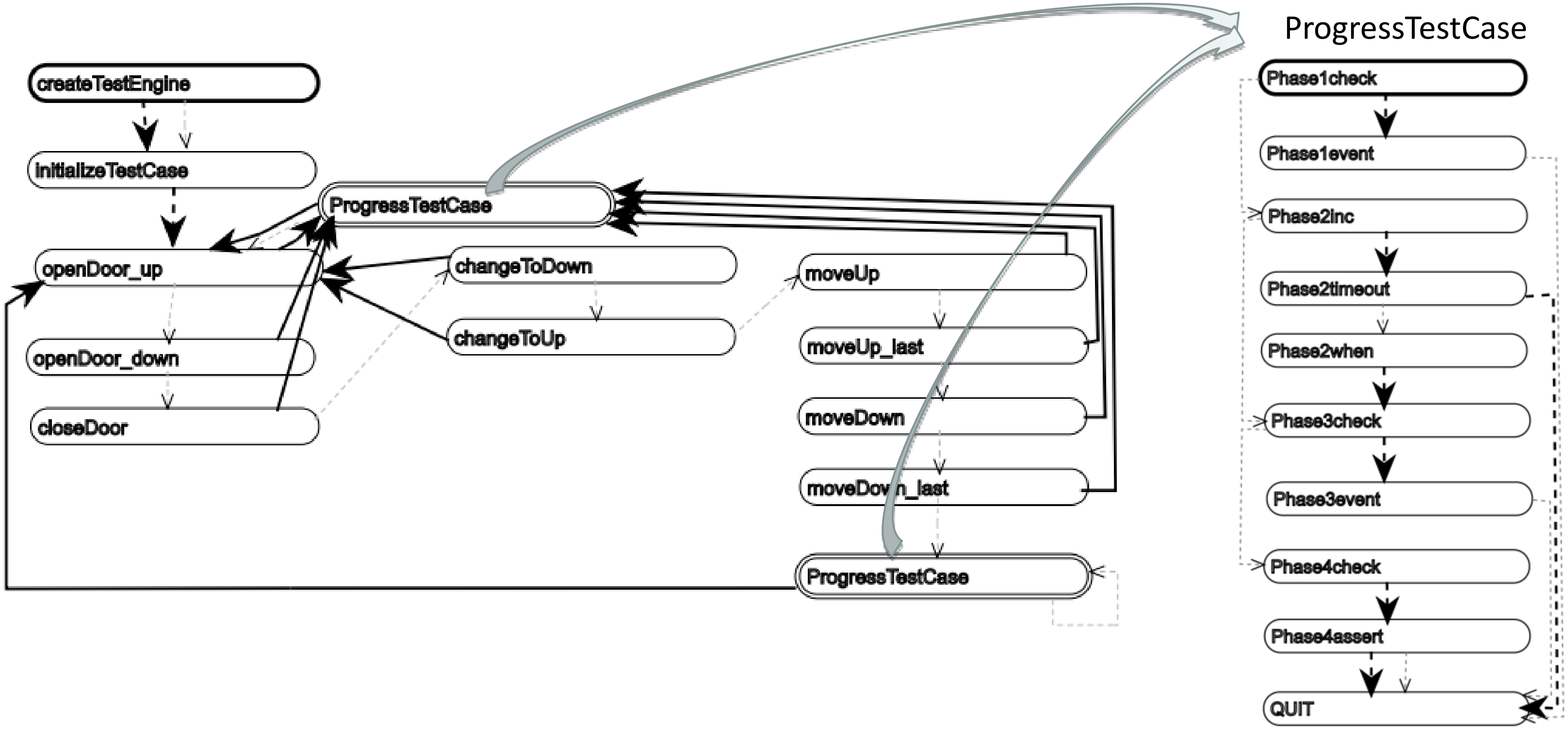
Includes oracle

Can also be used for runtime monitoring (~ trace-matches)

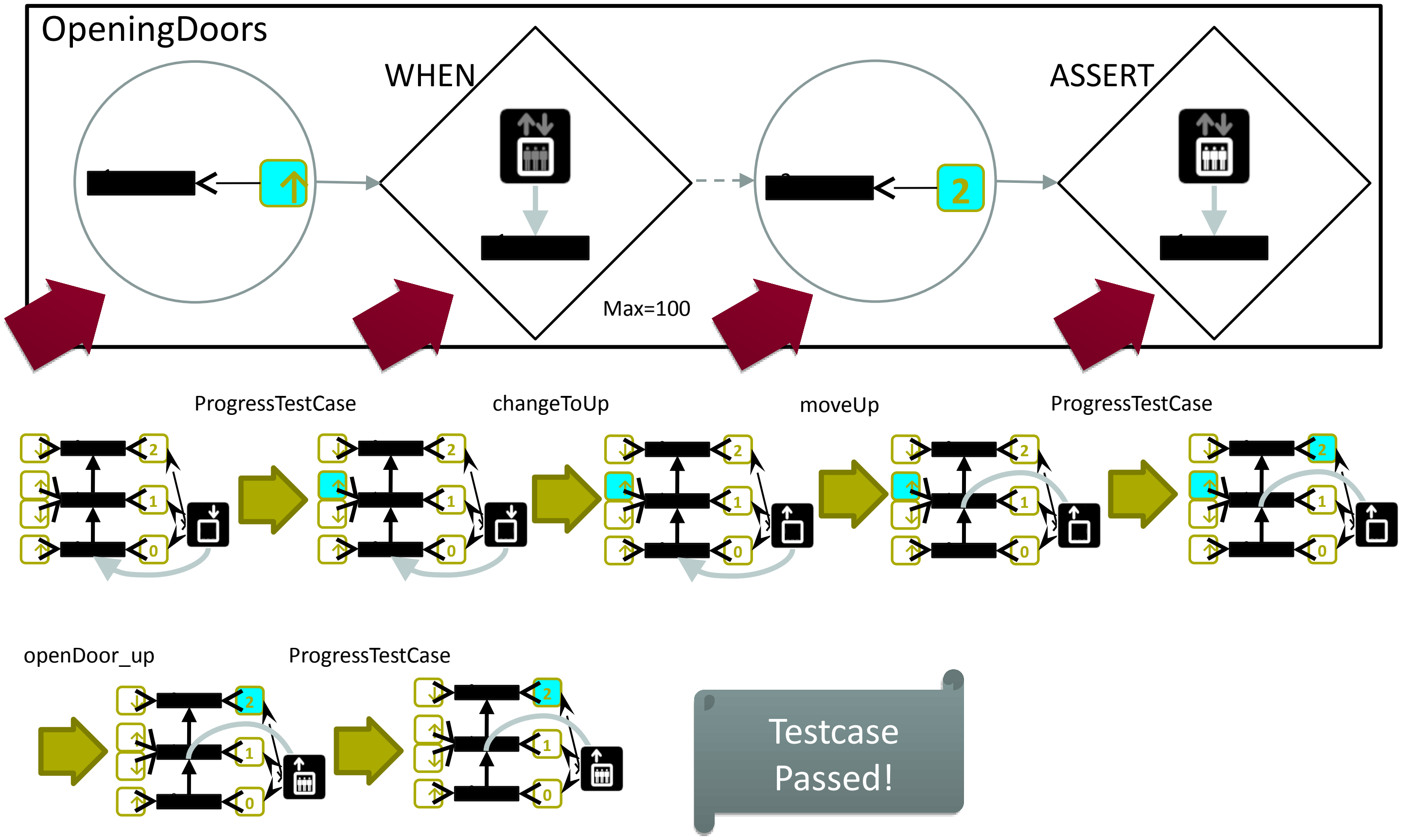
Testing DSML (generated)



Instrumented Operational Semantics (generated)

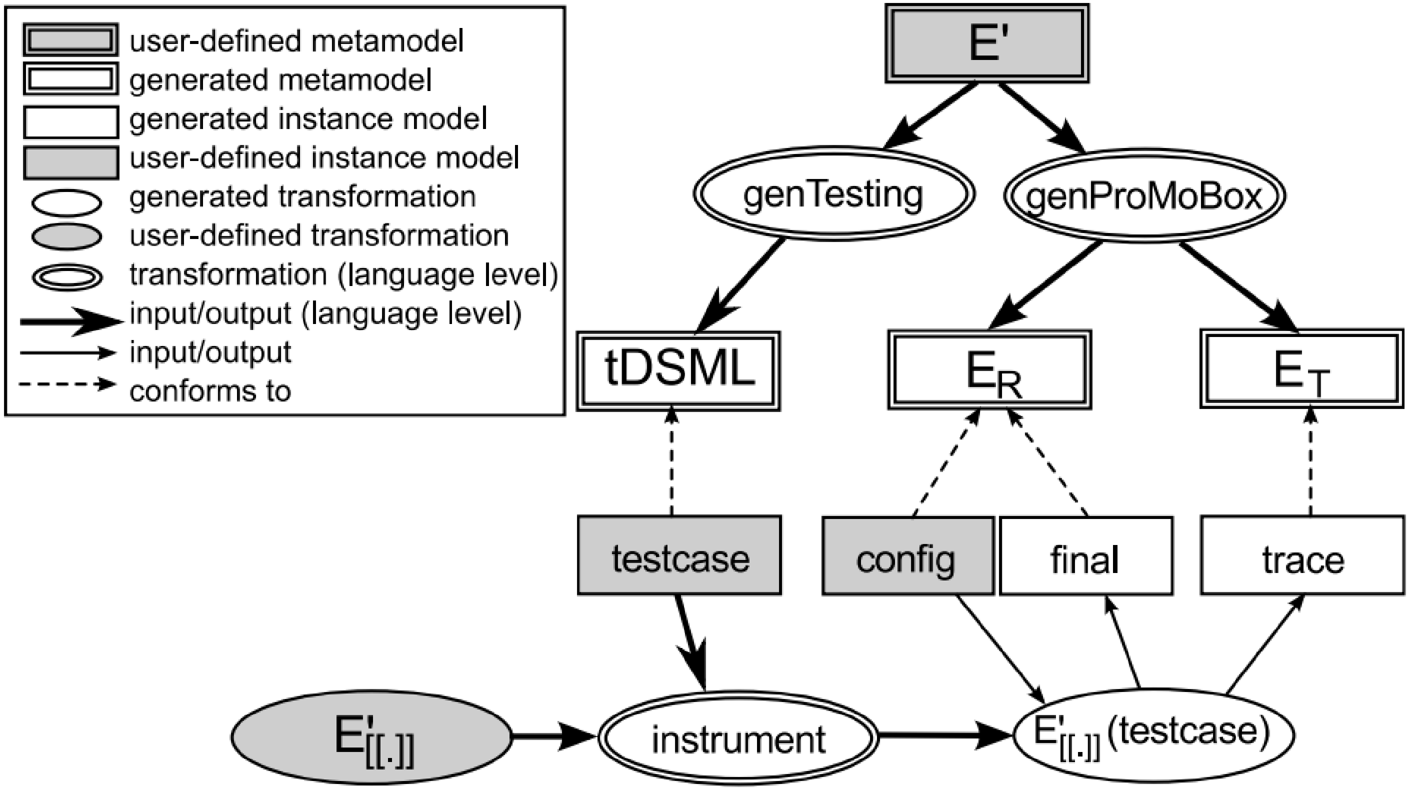


Test Case Execution



Testcase Passed!

Architecture of the Approach



Conclusion

- Definition of **family** of five related domain-specific sublanguages
- **Generative** approach
  - Language generation
  - Generic compiler
- **Insight** in and **support** for various tasks, e.g.,
  - Simulation
  - Model checking
  - Test case generation
- Future Work: Test Case Generation(with good coverage) from properties
- Future Work: Application to Design Space Exploration
  - Generate a language for specifying DSE optimization rules
  - Generate tool support

References

- Bart Meyers, Hans Vangheluwe, Joachim Denil, Rick Salay. A Framework for Temporal Verification Support in Domain-Specific Modelling. IEEE Trans. Software Eng. 46(4): 362-404 (2020).

- Bart Meyers, Romuald Deshayes, Levi Lucio, Eugene Syriani, Manuel Wimmer and Hans Vangheluwe. ProMoBox: A Framework for Generating Domain-Specific Property Languages. In "*Proceedings of the 7th International Conference on Software Languages Engineering (SLE 2014)*", Lecture Notes on Computer Science, vol. 8706, p. 1-20, 2014.

- Bart Meyers, Joachim Denil, István Dávid, and Hans Vangheluwe. Automated Testing Support for Reactive Domain-Specific Modelling Languages. In "Proceedings of the 2016 ACM SIGPLAN International Conference on Software Language Engineering". ACM digital library, p. 181-194, 2016.

- Bart Meyers, Manuel Wimmer, and Hans Vangheluwe. Towards Domain-specific Property Languages: The ProMoBox Approach. In "Proceedings of the 2013 ACM Workshop on Domain-specific Modeling", p. 39-44, ACM New York, NY, USA, 2013.

- Romuald Deshayes, Bart Meyers, Tom Mens and Hans Vangheluwe. ProMoBox in Practice : A Case Study on the GISMO Domain-Specific Modelling Language. In "Proceedings of the 8th Workshop on Multi-Paradigm Modeling (MPM 2014)", CEUR Workshop Proceedings, vol. 1237, p. 21-30, 2014.UR Workshop Proceedings, vol. 1321, p. 1-8, 2014.