Explicit Transformation Modelling

Towards Systematic Transfromation Development

Thomas Kühne, Gergely Mezei, **Eugene Syriani**,
Hans Vangheluwe, Manuel Wimmer

Ph.D. Candidate in the Modelling, Simulation and Design Lab

School of Computer Science

McGill University

OVERVIEW

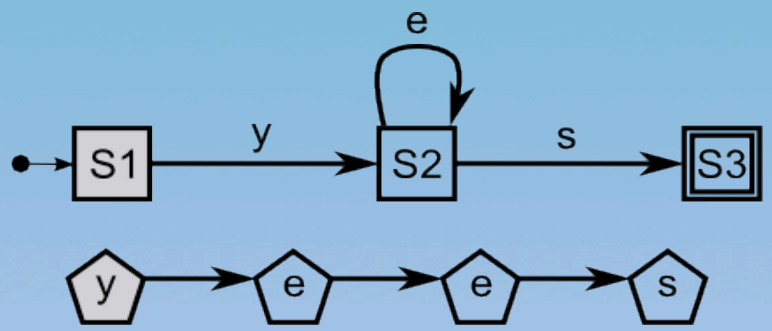
- Motivation
- Running Example
  - FSA semantics from PN simulation
- Explicit Transformation Modelling
  - RAM process
- Higher-Order Transformation
  - Adding animation
- Conclusion

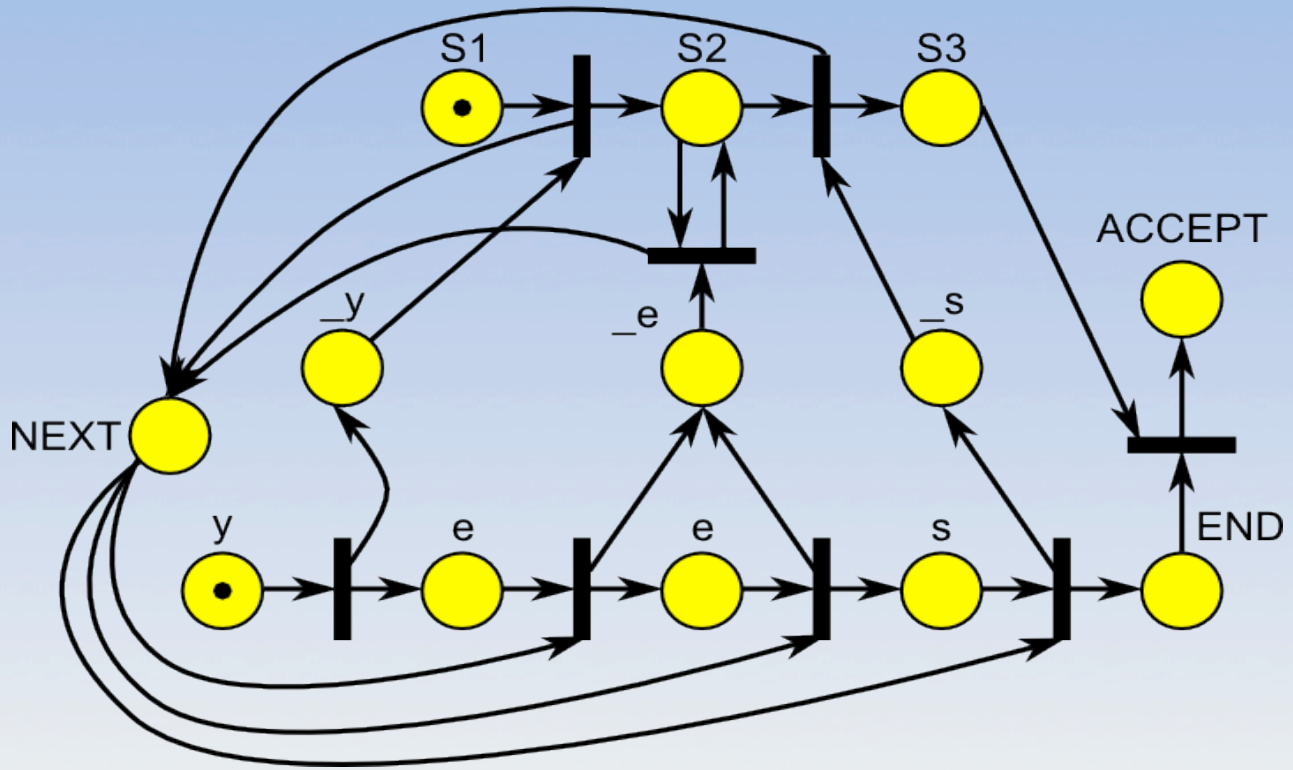
MOTIVATION

- Increase modeller's productivity
  - Raise level of abstraction of specification/design
  - Lower impedance mismatch between modelling language and application domain
- Explicitely model transformations as 1st class entities
  - Easier modification of control flow + mapping + pattern matching at syntax and semantics levels
  - Domain-specific transformation models //vs// generic pattern matching language
  - Facilitate transformation evolution: higher-order transformation (HOT)

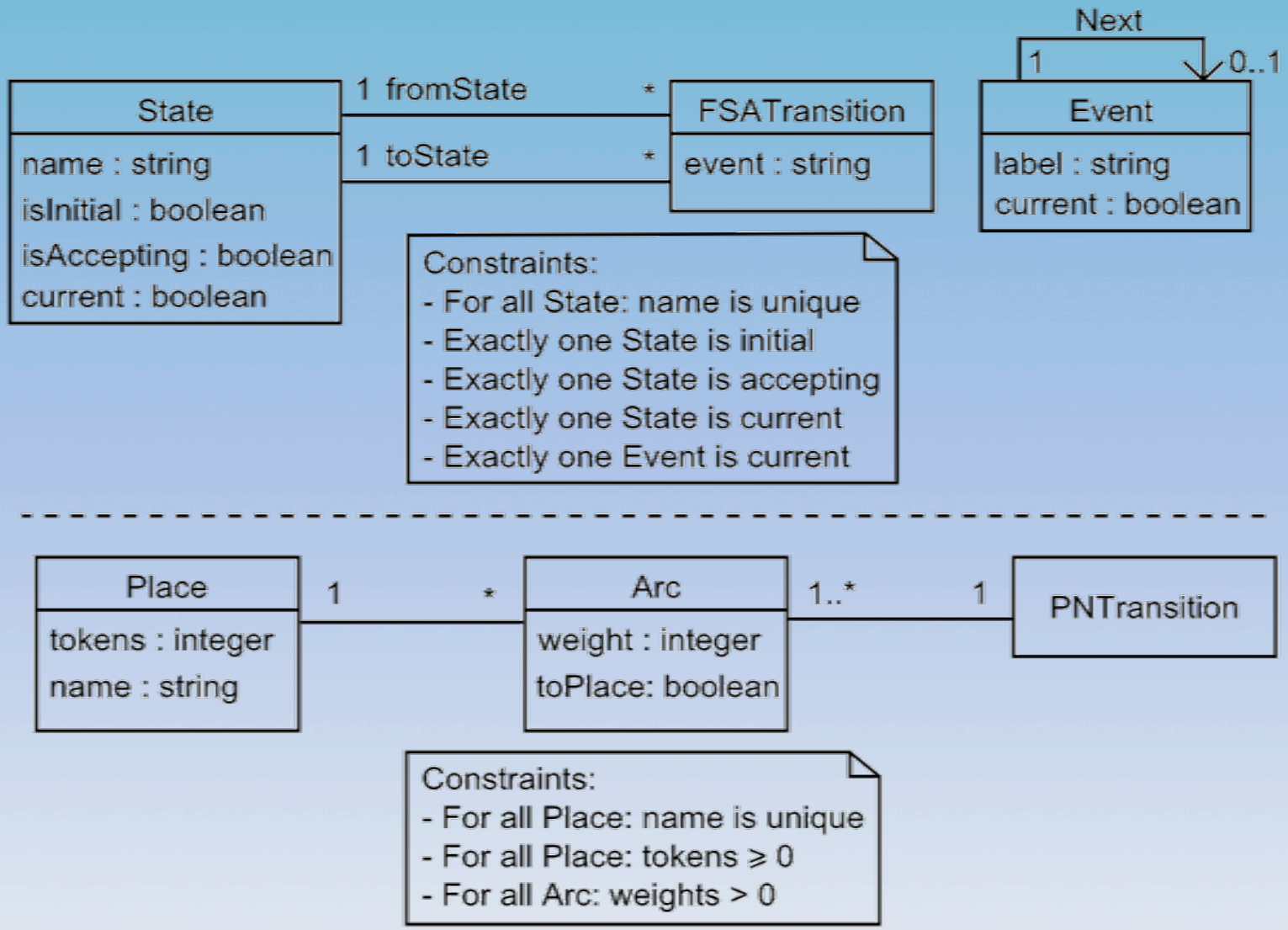
RUNNING EXAMPLE

Finite State Automata (FSA) Simulator from Petri-Net Simulator



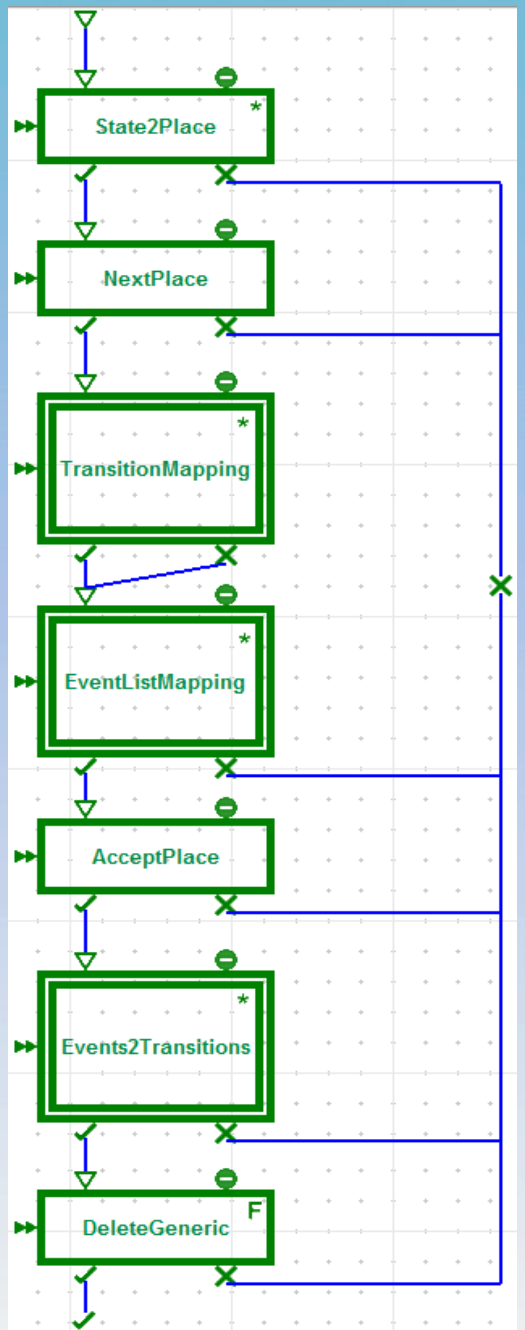


FSA & PETRI-NET META-MODELS

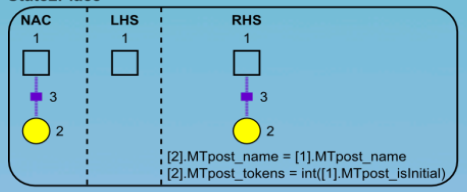


- Classes
- Multiplicities
- Attribute types
- Constraints

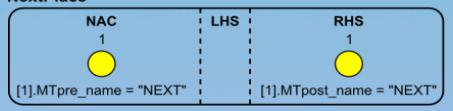
FINITE STATE AUTOMATA TO PETRI-NET



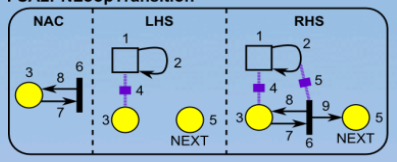
State2Place



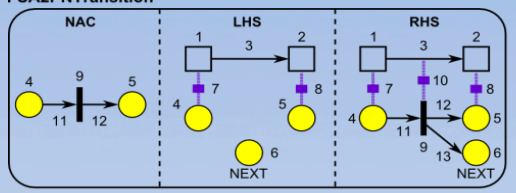
NextPlace



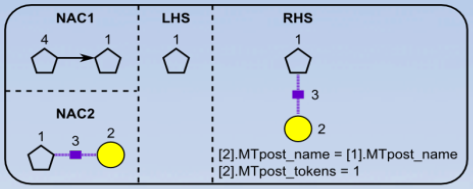
FSA2PNLoopTransition



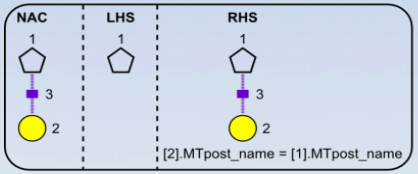
FSA2PNTransition



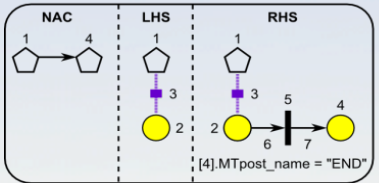
HeadEvent2Place



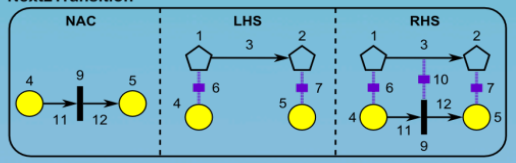
Event2Place



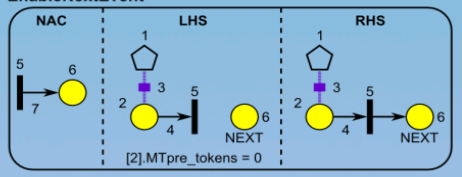
EndPlace



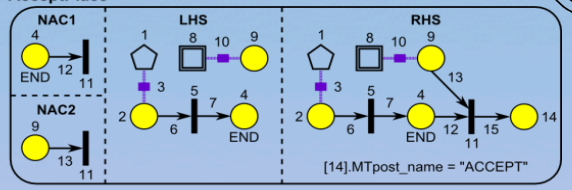
Next2Transition



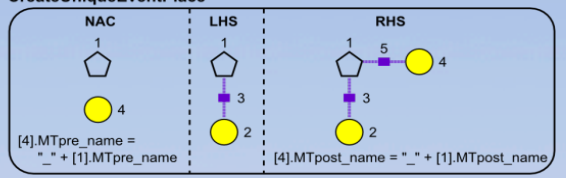
EnableNextEvent



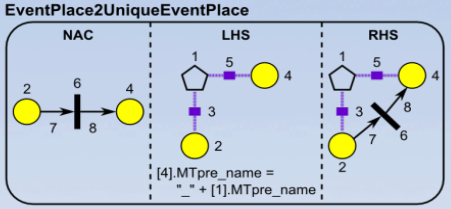
AcceptPlace



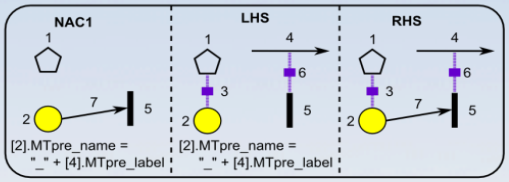
CreateUniqueEventPlace



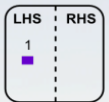
EventPlace2UniqueEventPlace

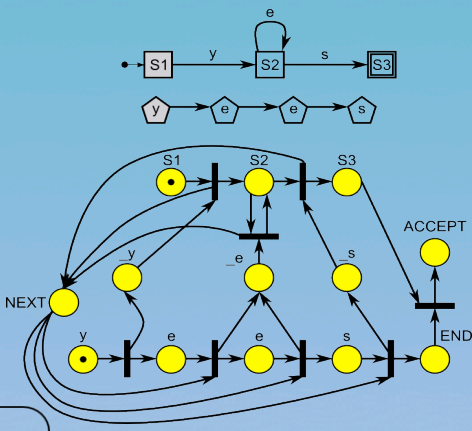


EventLabel2Transition

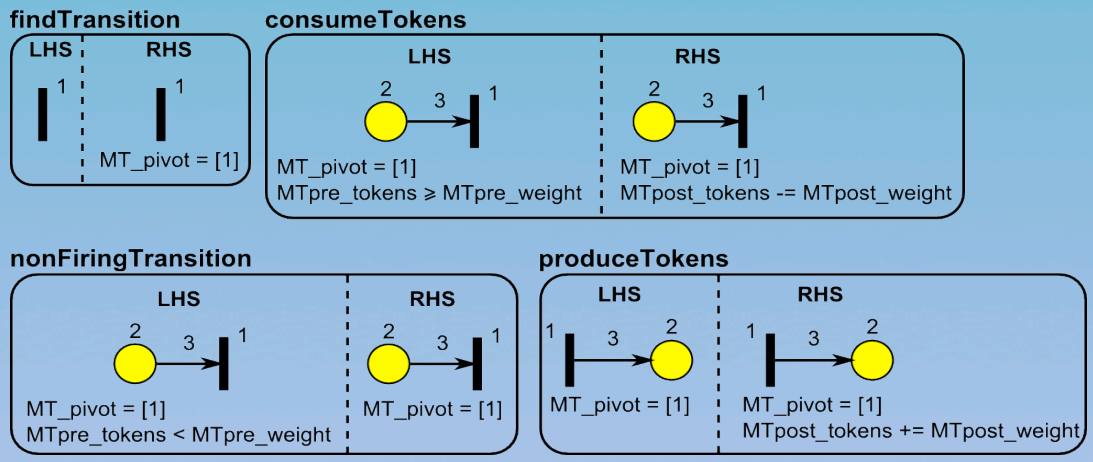


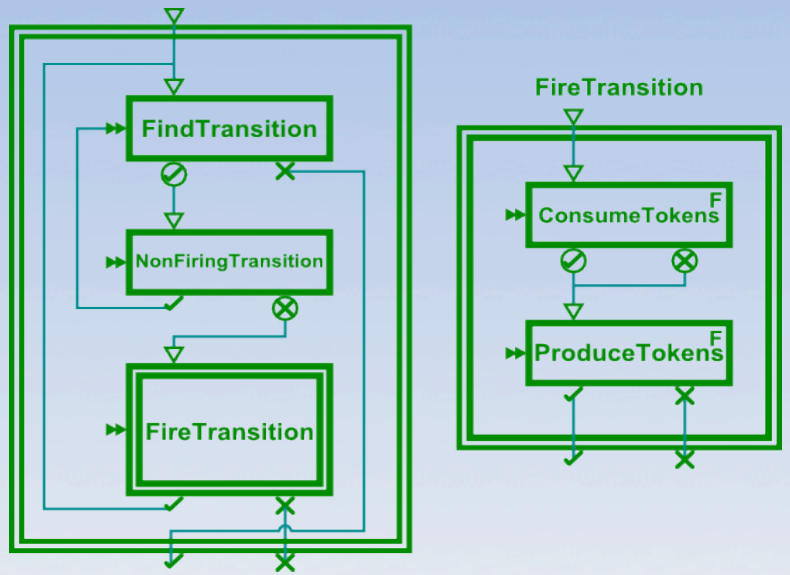
DeleteGeneric





PETRI-NET OPERATIONAL SEMANTICS





EXPLICIT TRANSFORMATION MODELLING

- Generic //vs// Customized pattern language
  - Syntax: pattern specification adapted to languages invloved
  - Design: exclude patterns that would not match

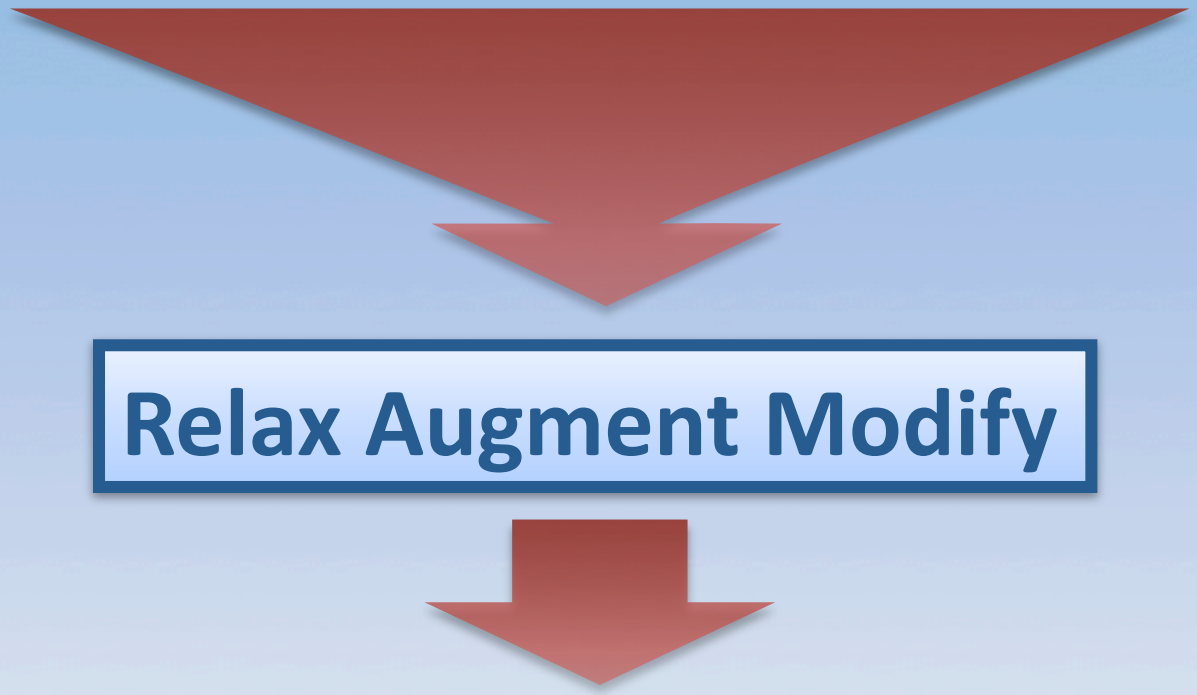
  → Custom
- Meta-model //vs// Conformance check
  - Have a separate meta-model for the pattern specification
  - Have customized conformance checks

  → Meta-model

RAM PROCESS

(quasi-)Automatically generated environment for pattern language

Input Meta-Model **Output Meta-Model**



Customized Pattern Meta-Model

RAM PROCESS

(quasi-)Automatically generated environment for pattern language

Relaxation

- Concretize abstract entities
- Reduce minimal multiplicity constraint
- Constraints *(automatic?)*
  - Free form: no constraints
  - Valid elements: ensure typing
  - Valid multiplicities: enforce (relaxed) multiplicities
  - Valid constraints: enforce (a subset of) meta-model constraints

RAM PROCESS

(quasi-)Automatically generated environment for pattern language

Augmentation

- Type all entities to MT(pre/post)_Element
- Add MT specific attributes
  - Labels
  - Pivot passing
  - Allow subtype matching *
  - Other (e.g., isProcessed)
- Generic elements

RAM PROCESS

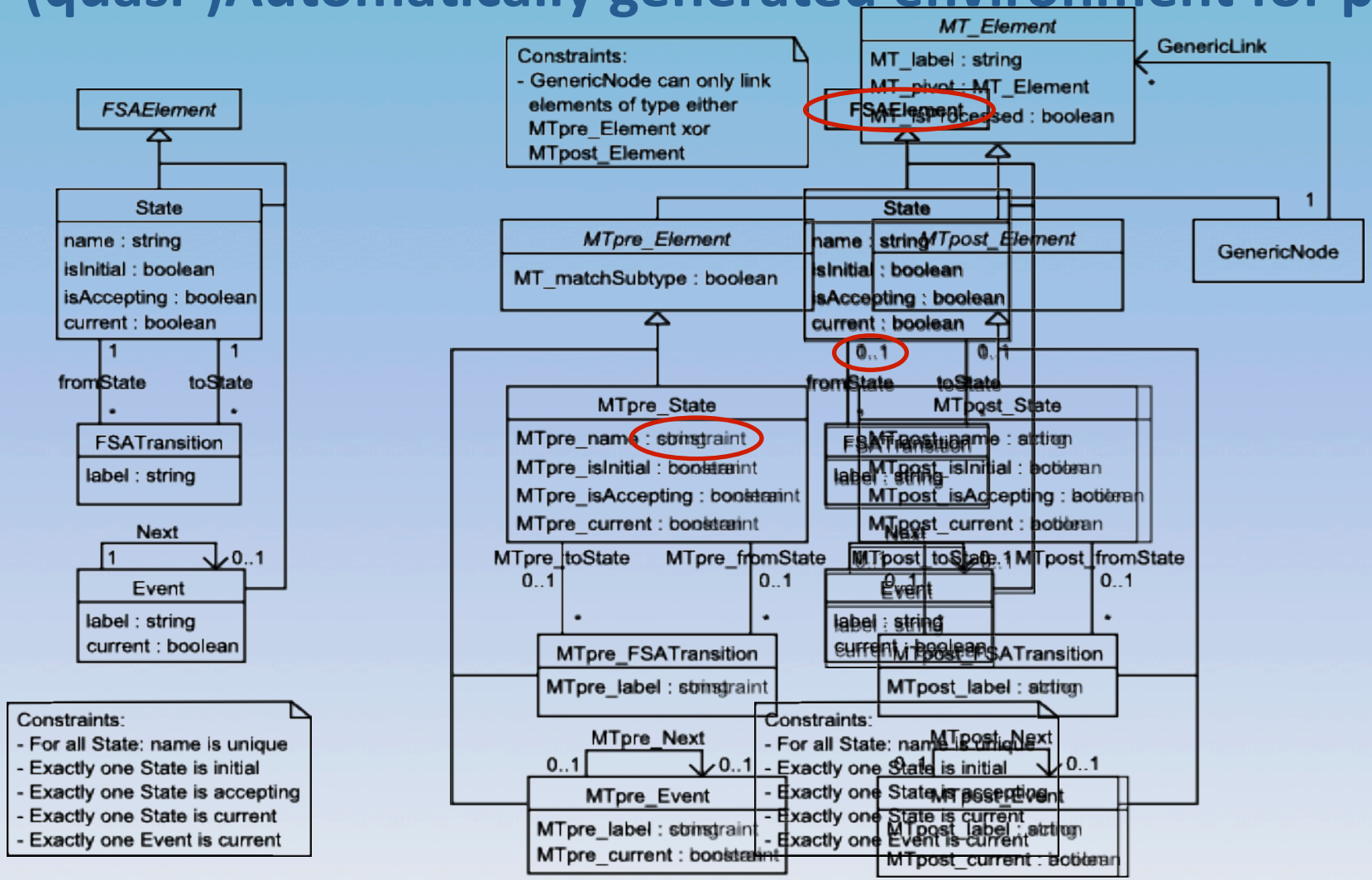
(quasi-)Automatically generated environment for pattern language

Modification

- Pre-condition pattern
  - Attributes are of type 'constraint'
- Post-condition pattern
  - Attributes are of type 'action'
- Modify concrete syntax
  - Abstract classes
  - Association ends
  - Other (e.g., replace topological visual syntax constraints)

RAM PROCESS

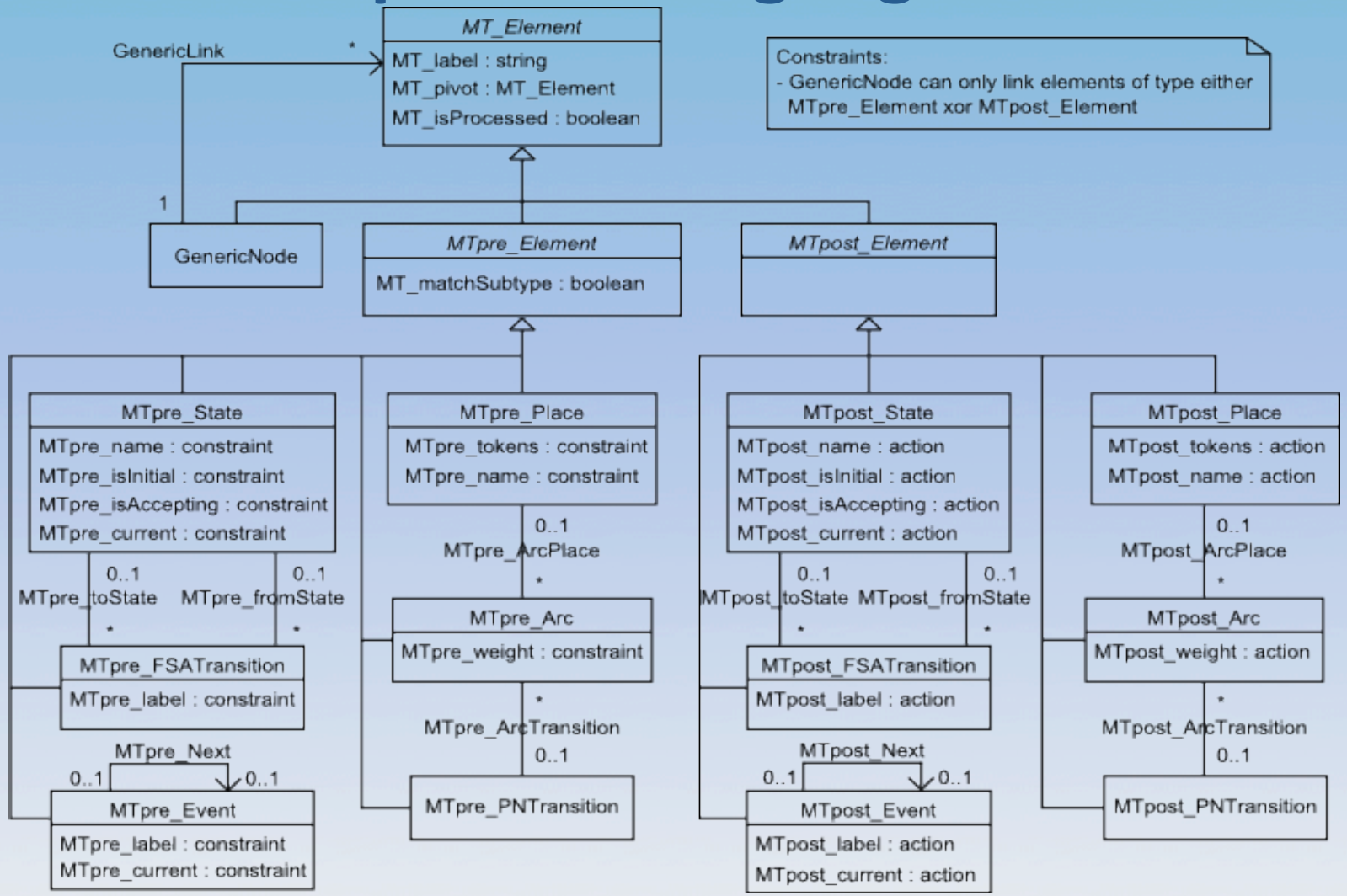
(quasi-)Automatically generated environment for pattern language



Relax Augment Modify

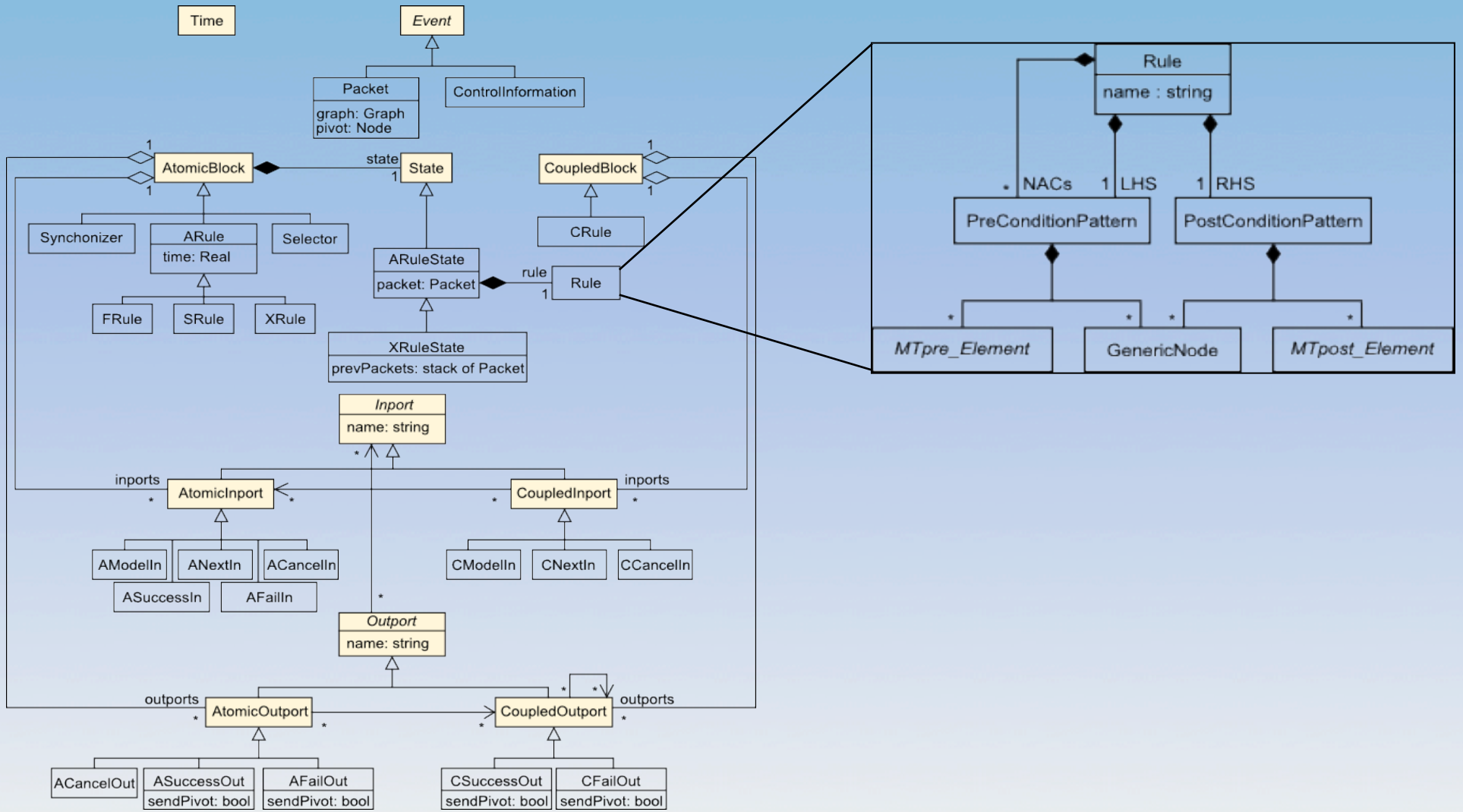
RAM PROCESS

Customized pattern language meta-model



RAM PROCESS

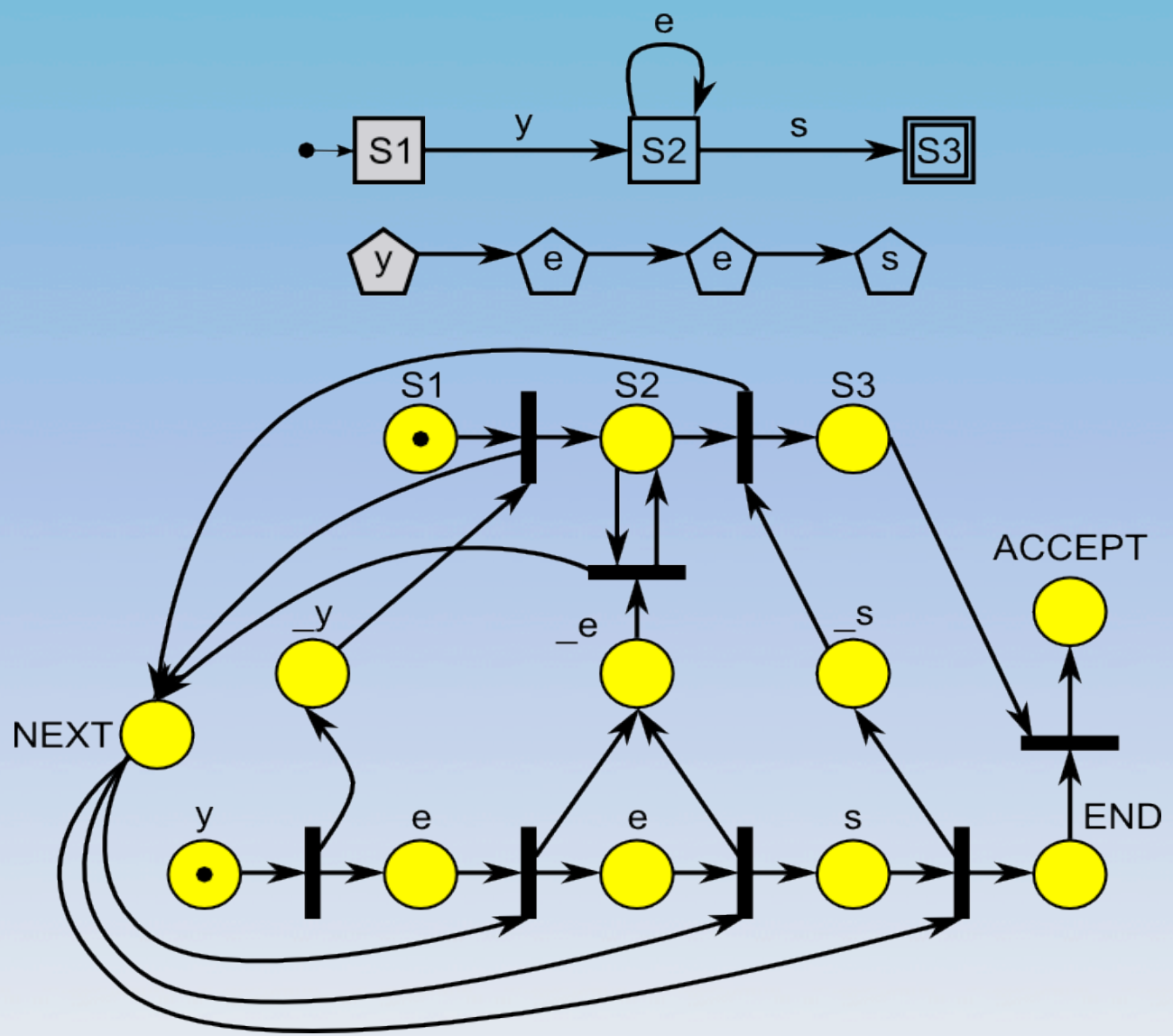
Transformation Meta-Model



HIGHER-ORDER TRANSFORMATION (HOT)

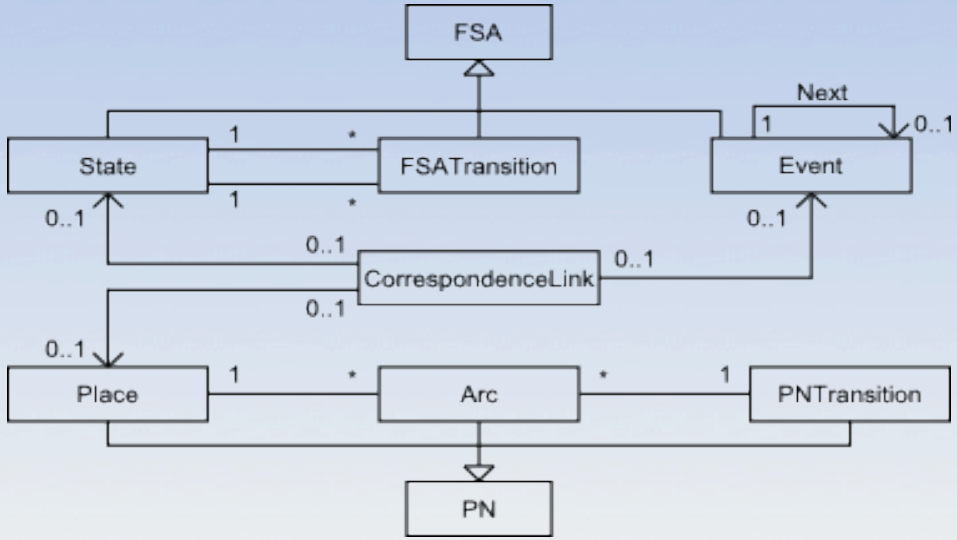
- Language evolution
  - Adapting the transformation of an evolving language
- Transformation optimization/refactoring
  - Modify for more efficient results
- Translational semantics
  - From operational semantics, for more advance analysis oportunities
  - Define meaning of transformation by mapping onto a standard one
- Separation of transformation concerns
  - Multi-stage transformation is a special case

ANIMATION SUPPORT



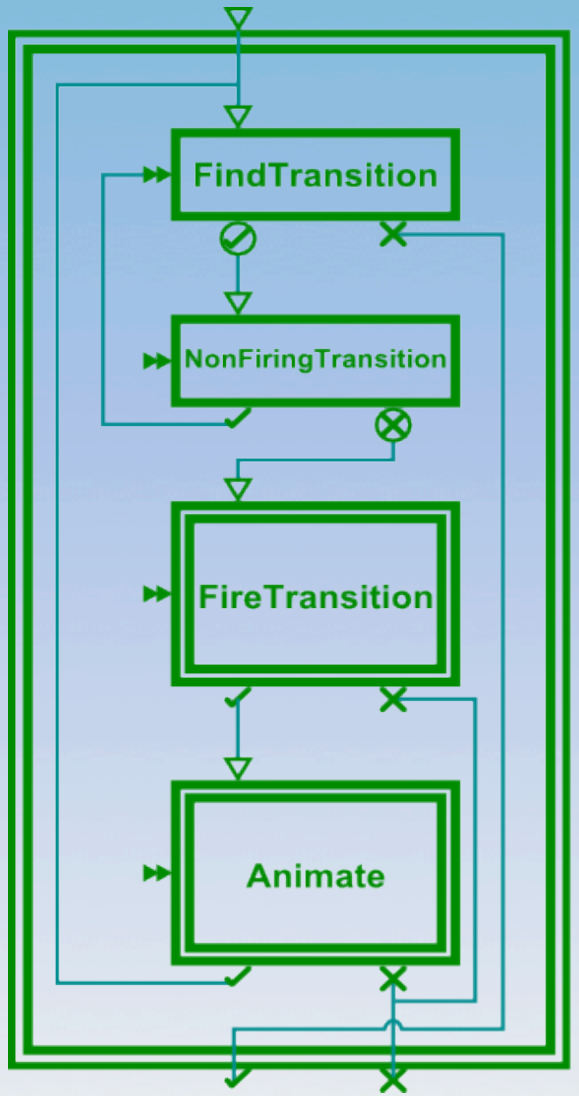
ANIMATION SUPPORT

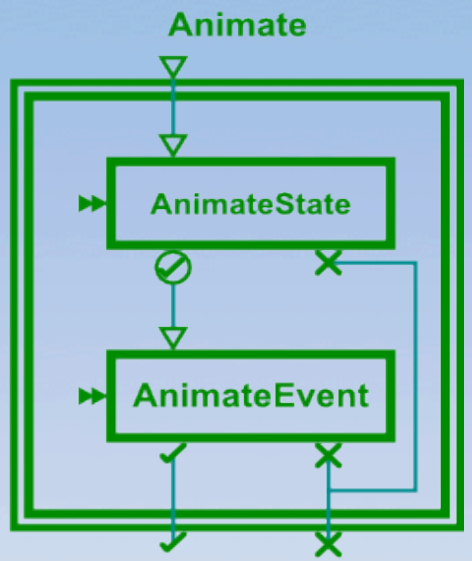
- ! User modifies meta-model: correspondence links
- ? Evolve the transformation without changing current transformation
- ➢ Transform the transformation to include animation support



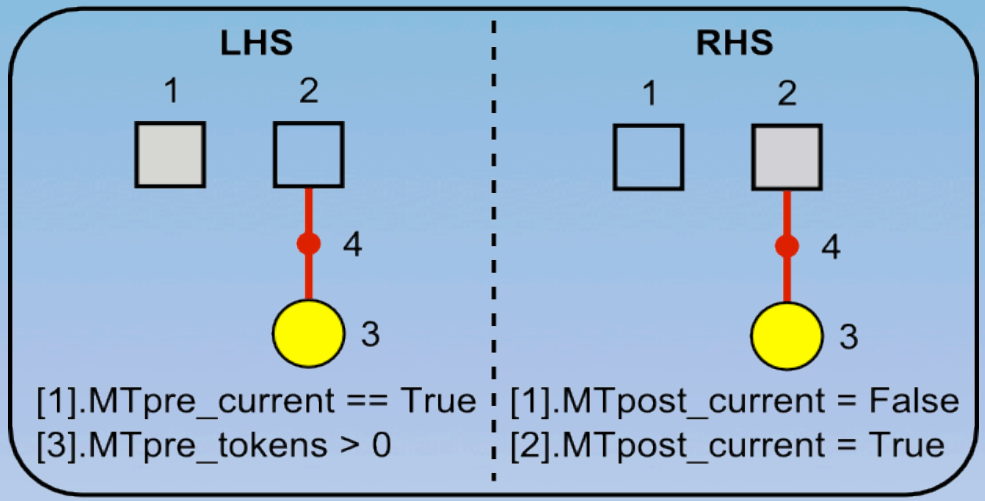
ANIMATION SUPPORT

Add animation rules in simulator

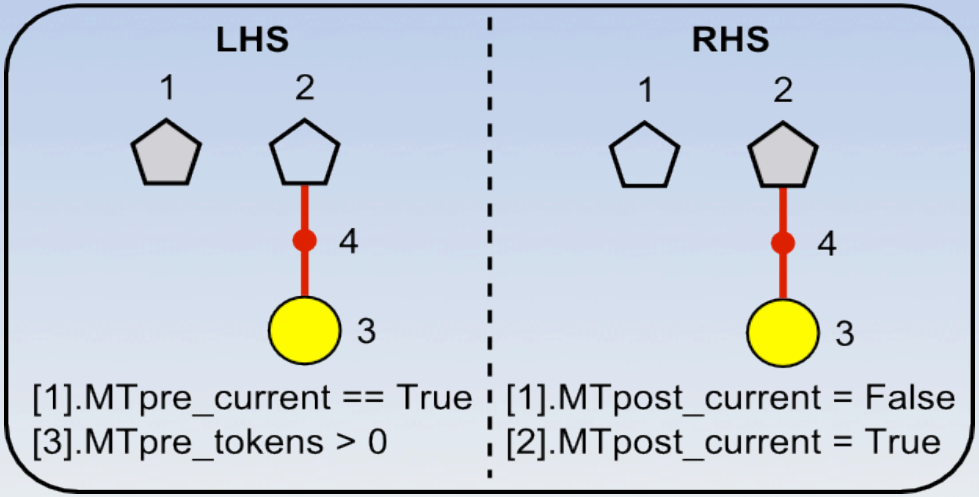




animateState

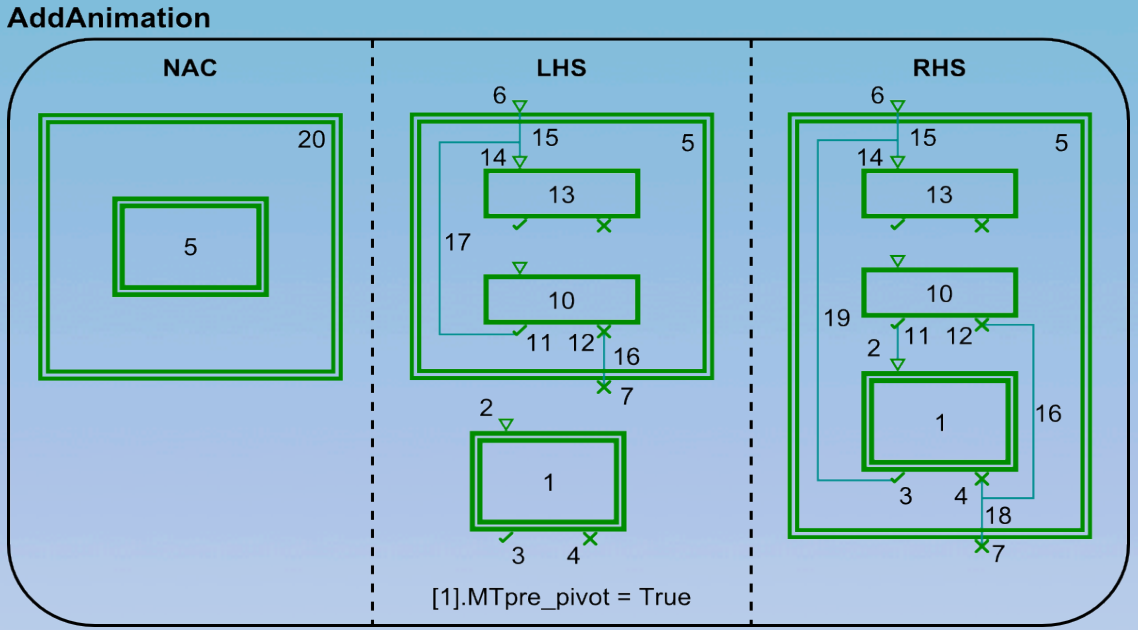


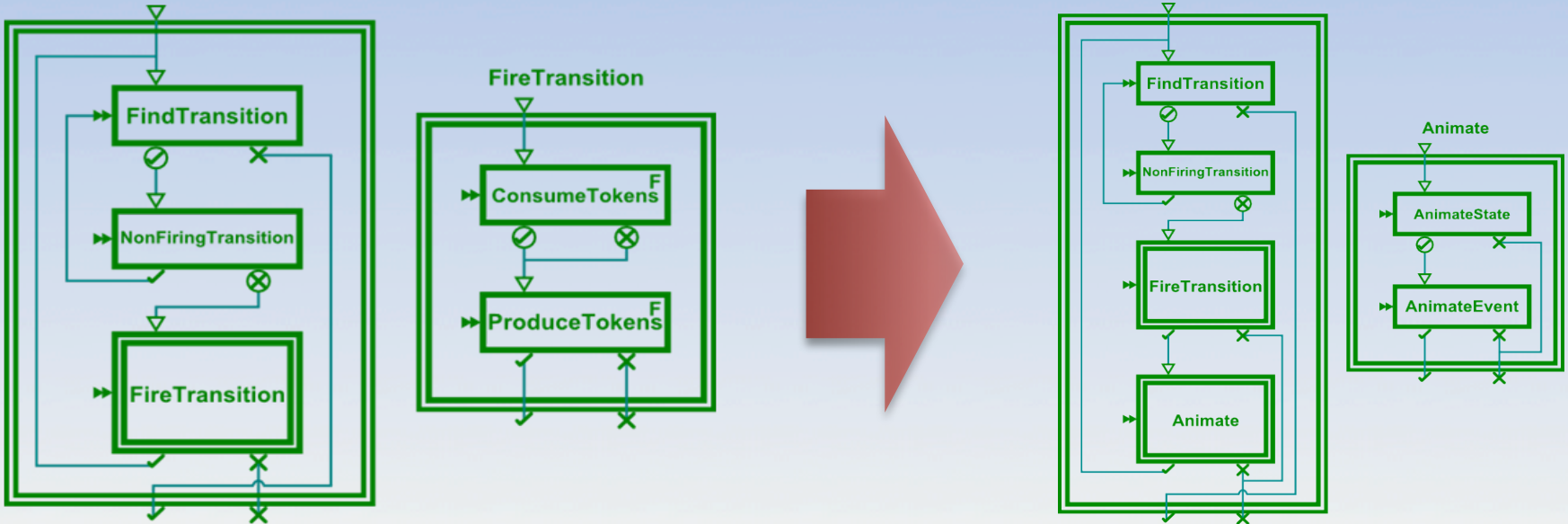
animateEvent



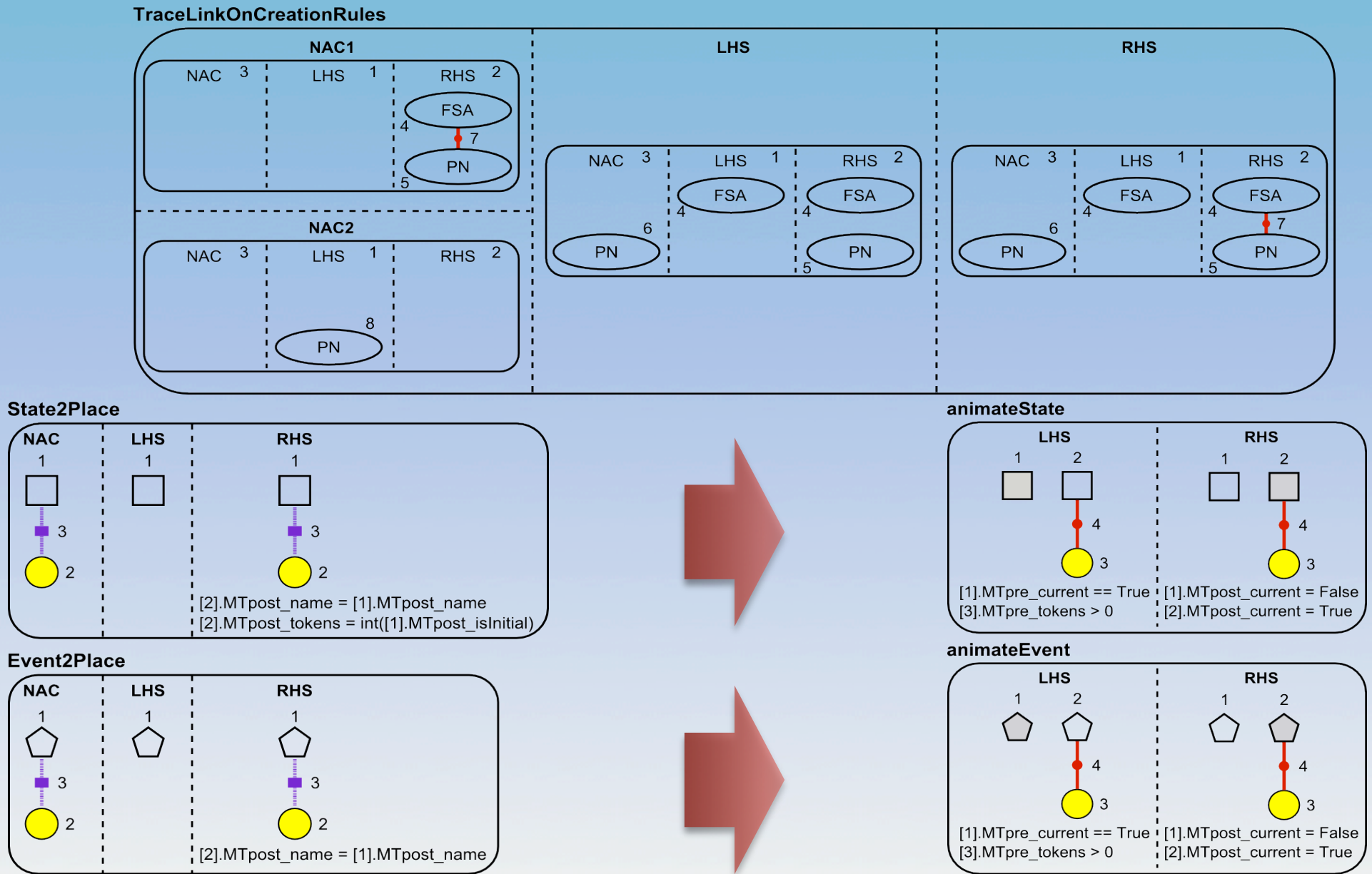
ANIMATION SUPPORT

Control flow level HOT rule





ANIMATION SUPPORT
Rule level HOT rule



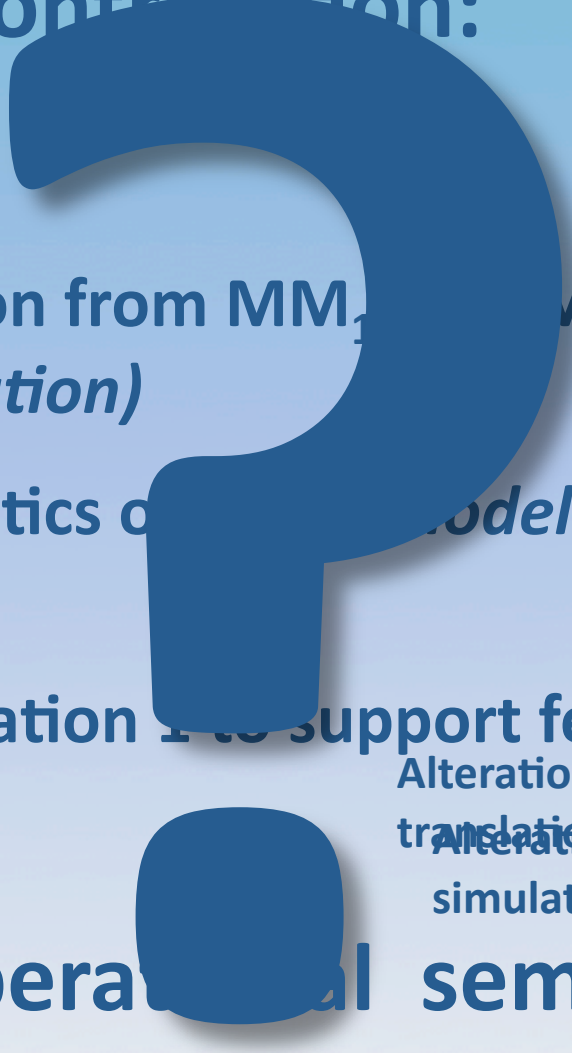
CONCLUSION

- Proposed a *systematic procedure* for the development of model transformations
- Allows to:
  - Offer customized specification of model transfromation rules
  - *Cleanly* design higher-order transformation rules

CONCLUSION

Side-effect of the contribution:

- Given
  1. Semantic translation from MM_1 to MM_2 *(model transformation)*
  2. Operational semantics of MM_2 *(model transformation)*
  3. Alterate transformation 1 to support feed-back *(HOT)*



Translation of FSA to PN

PN simulator

Alteration of the rules of the FSA to PN translation

Alteration of the control flow of the PN simulator

- You get an "operational semantics" for MM_1