Implementing Operational Semantics in MetaEdit+

Simon Van Mierlo - 20081499

Universiteit Antwerpen

Contents

- Goals
- Introduction to MetaEdit+
- Hardcoded Operational Semantics
- Graph Rewriting

Goals

- Add operational semantics to the 'Production System' formalism in MetaEdit+
- Investigate the possibility of using graph rewriting to generate and execute rules by mapping to TCore rules

MetaEdit+ [1]

- Define Graphical Languages
  - Metamodelling Language: GOPRR

    Graph, Object, Property, Relationship, Role
  - Forms or Graphically
  - Icon and Symbol Editor
  - Constraints
    - Graphical through Ports
    - Syntactical

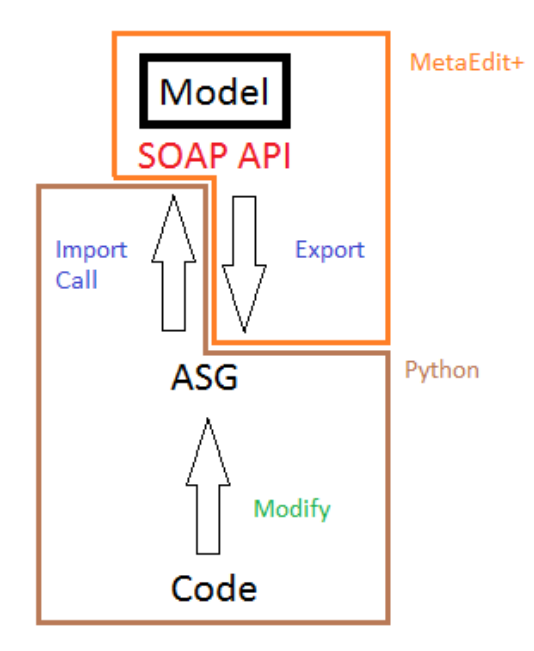
[1] www.metacase.com

MetaEdit+

- Generators
  - Generate code from models
  - Example: GOPRR
- SOAP API
  - Access to models from outside MetaEdit+
  - Define semantics

Hardcoded Operational Semantics

- Import/Export model to Python
- Provide a layer above SOAP API



Hardcoded Operational Semantics

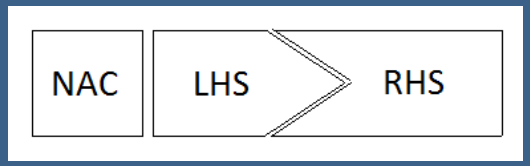
- Concern:
  - API seems not to be built for this kind of job

```
restrictedType = client.factory.create('ns0:METype')
restrictedType.name = 'NonProperty'
rel = client.service.relsForObj(self.parent.instance, self.instance, restrictedType)[0]
bindingReprs = client.service.bindingReprs(self.parent.diagram)
relRep = None
for bindingRep in bindingReprs:
    binding = client.service.inst(bindingRep)
    relationship = client.service.relationship(binding)
    if relationship.objectID == rel.objectID:
    relRep = bindingRep
place = client.service.place(relRep)
```

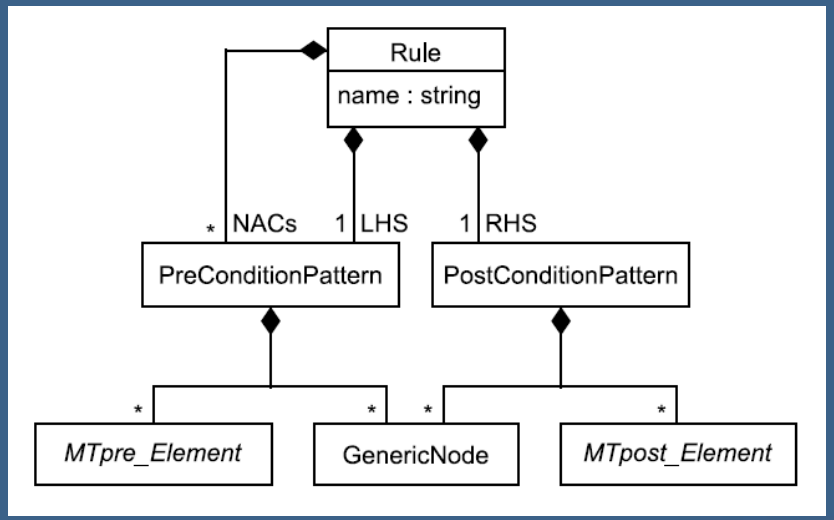
- Solution(s):
  - Exporting instead of importing
  - Optimizations to reduce the number of calls to the API

Graph Rewriting

- Rule:



- Rule Metamodel[2]:



[2] Kühne, T.; Mezei, G.; Syriani, E.; Vangheluwe, H. & Wimmer, M. Explicit Transformation Modeling. MODELS 2009 Workshops, LNCS: 6002, pp. 240-255, Springer. Denver (USA) 2010.

Metamodel Metamorphosis

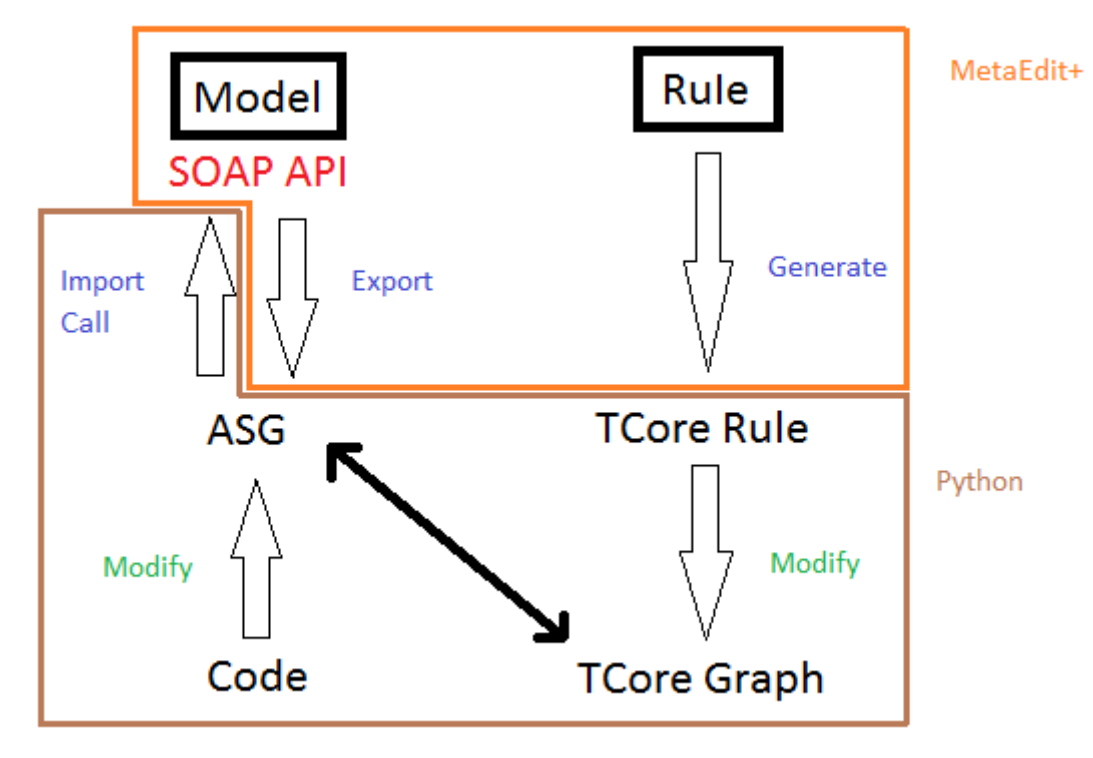
- Create a pattern specification metamodel
  - Modify existing 'Production System' metamodel
  - RAM: Relaxation, Augmentation, Modification
  - These metamodels define the languages used for LHS, RHS and NAC parts of the rule

TCore[3]

- Collection of model transformation primitives
  - Combine with scheduling language to get transformation language
- Implementation in Python: PY-TCore
- Translate rule models to TCore rules

[3] Syriani E., 2011, A Multi-Paradigm Foundation for Model Transformation Language Engineering, http://www.cs.mcgill.ca/~esyria/publications/dissertation.pdf

TCore


Model
SOAP API
Rule
MetaEdit+
Import
Call
Export
Generate
ASG
TCore Rule
Python
Modify
Modify
Code
TCore Graph