Shuttle 1.0

Kao, Chen-yi

2009-04-13

Shuttle 1.0: an open domain model completion tool

- Built on the basis of former Shuttle project
- Performance tuning – utilizing Eclipse concurrency
- Open domain model completion
  - Building research topics
  - Client-server model analysis

Project plan

- Short-term plan (within one or two years)
  - Client-side: (Structural) rule inference result recommendation visualization on GMF editors
  - Server-side: Light-weight rule-base
- Long-term plan
  - From *modeling assistance* to *generally documenting assistance*
  - Even *ubiquitous user interface assistance*

Performance tuning

Utilizing Eclipse concurrency

- For long running operations
  - Client-side: to improve response time
    - **WordNet lexical base search related** - concept mapping & ontology linking
    - **KAON2 reasoning related**
  - Server-side: to utilize I/O (network and/or disk) waiting time
    - Rule base indexing
  - Others recognized in the future

Eclipse concurrency

- **Jobs** (`org.eclipse.core.runtime.jobs.Job`)
  - One job a thread
  - Can be scheduled to re-run
  - Typically completing a **task** per run
- **Progress** (`org.eclipse.core.runtime.IProgressMonitor`)
  - **System Jobs** vs. **User Jobs**
    - Set Jobs as ***User Jobs*** to enable showing progress
  - Each **task** is responsible to a 0~100% progress reporting

Single-job-many-tasks model

- For ***concept mapping*** (concept rule construction)
- Mappings for some accumulated model elements constitute a "task"
- Trade-off between **job run overhead** and **unpredictability of element number**

Open domain model completion

Inferring model via existing concept mapping technique

- Reduce semantic noise
- Extended from concept linking…

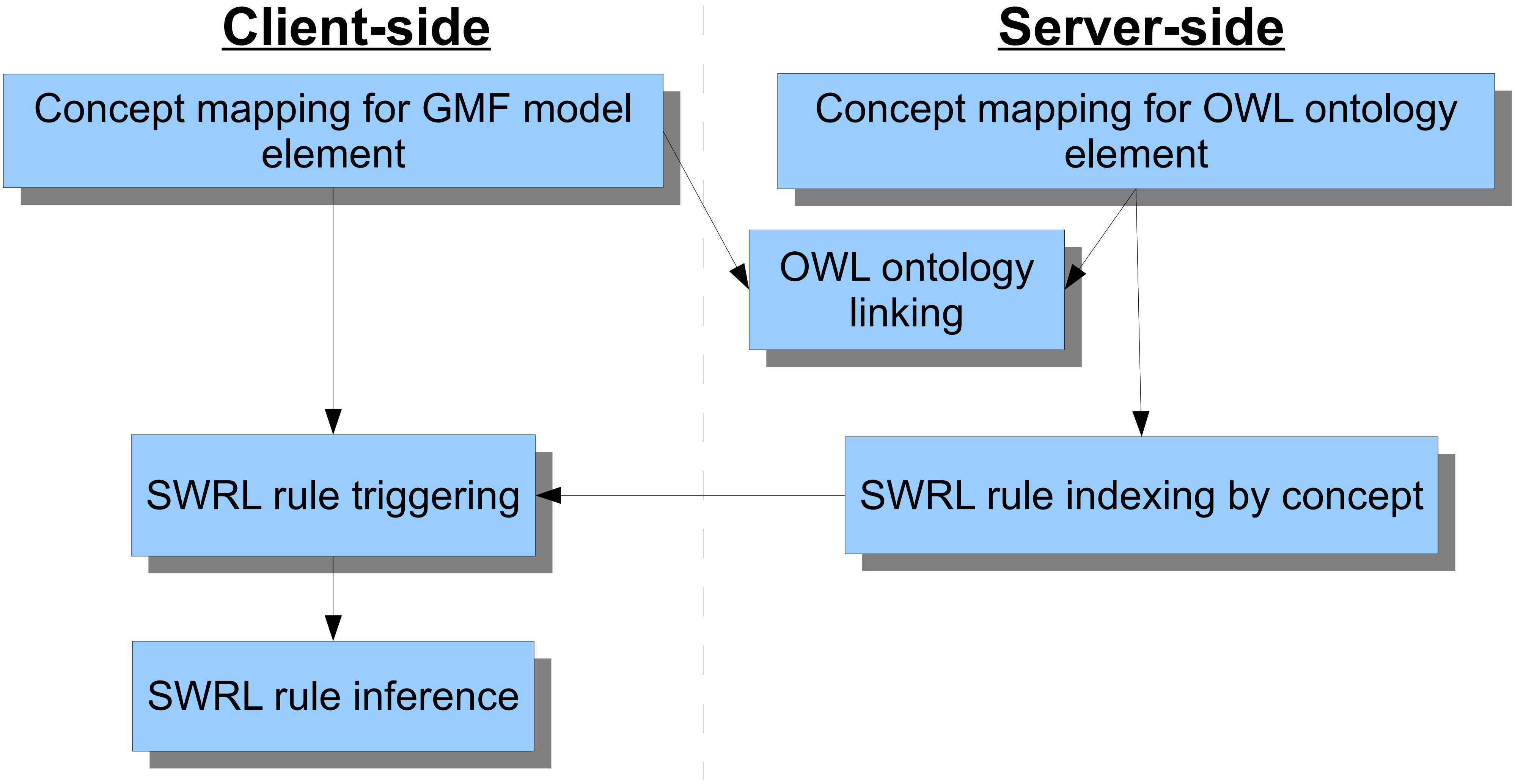
1. OWL ontology linking
2. SWRL rule triggering
    - hasParent(?x1,?x2) ∧ hasBrother(?x2,?x3) ⇒ hasUncle(?x1,?x3)
3. SWRL rule inference
    - Direct inference
    - Indirect inference by KAON2(http://kaon2.semanticweb.org/) reasoner

Client-server model analysis

- Inferring model via Concept mapping technique

  1. **Concept mapping for both GMF model and OWL ontology elements**
  2. (OWL ontology linking)
  3. **SWRL rule triggering**
  4. **SWRL rule inference**
     - Indirect inference by KAON2 reasoner / Direct inference

Client-server model of model inference



Client-side

Server-side

Concept mapping for GMF model element

Concept mapping for OWL ontology element

OWL ontology linking

SWRL rule triggering

SWRL rule indexing by concept

SWRL rule inference

Client-side scheme

1. OWL ontology linking
2. SWRL rule triggering
3. SWRL rule inference
4. **Rule inference result recommendation visualization**
    - MDA-lized rules
    - GMF model completion recommendation

Using MDA to bridge I and II

- Model Driven Architecture (MDA)
  - **PIM**(medicine) + **PM**(Java) = **PSM**
  - PIM: platform independent model
  - PM: platform model
  - PSM: platform specific model
- PM-tagging rules
- GMF editor definitions as modeling PM (MPM)
  - GMF editor rules as MPM rules

MDA-lized Rules

- PM rules
  - Tag rule & GMF editor elements for **concept mapping**
- GMF editor definition rules
  - For GMF-involved MPM
  - *Mine/extract GMF editor definition rules for **model completion recommendation***
    - When editor definitions are not available for some reason

Shuttle 1.0:
Divided in *four* research topics...

I. Inferring model via Concept mapping technique

II.1. Generating rules from GMF editor definitions for model completion recommendation

II.2. Extracting GMF editor rules for model completion recommendation without editor definitions

III. MDA-based inference for model completion recommendation visualization

Research topics

I. Inferring model via Concept mapping technique

II.1. Generating rules from GMF editor definitions for model completion recommendation

II.2. Extracting GMF editor rules for model completion recommendation without editor definitions

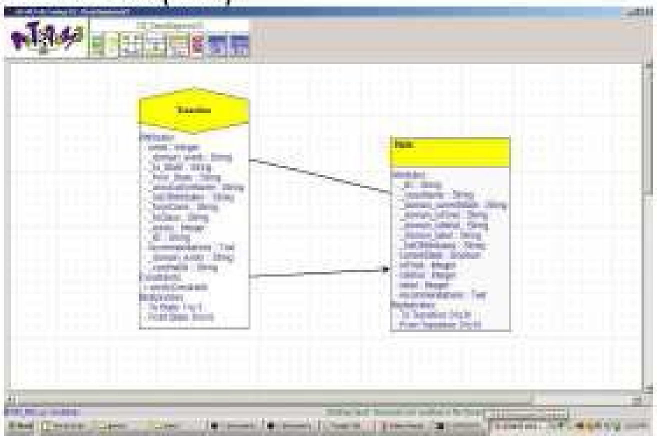
III. MDA-based inference for model completion recommendation visualization

Related research

- Domain-specific Model Editors with Model Completion (http://www.irisa.fr/triskell/publis/2007/Sen07b.pdf)
  - For AToM^3 (A Tool for Multi-formalism and Meta-Modelling) platform (http://atom3.cs.mcgill.ca/)
  - Forward editor generation

The AToM^3 example of FSM: A Domain-specific Model Editor with Prolog Model Completion Rules

Step 1: Specify a meta-model as an AToM3 Meta-model (MM)



Step 2. Specify Constraints in Prolog on MM

atleastOneFinal

sum(listOfisFinals,>=,1)

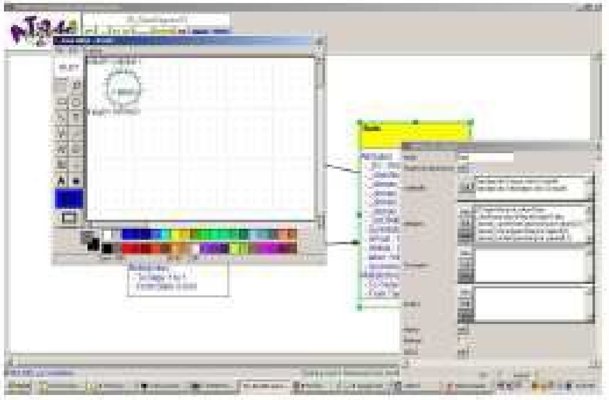
exactlyOneInitial

occurrences(listOfisInitial,1,1)

.

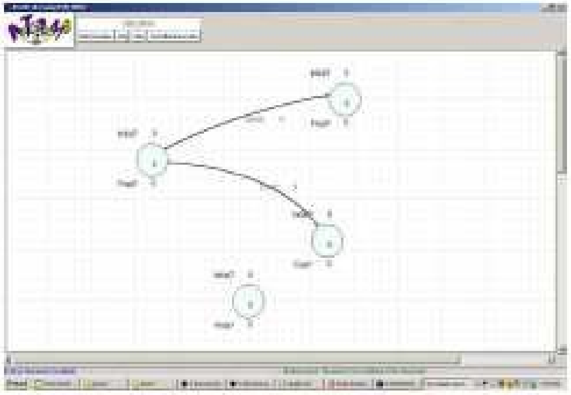
.

Step 3. Specify a Visual Syntax Specified in an Icon Editor

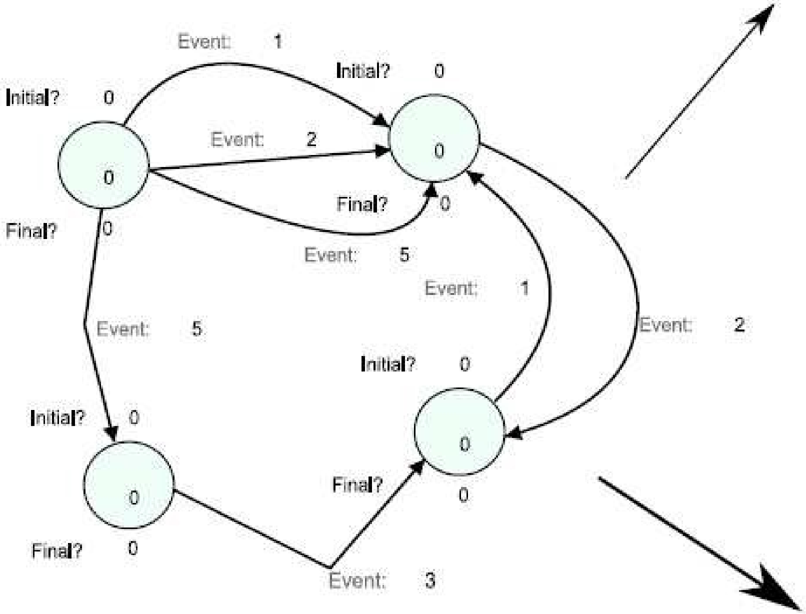


Synthesis

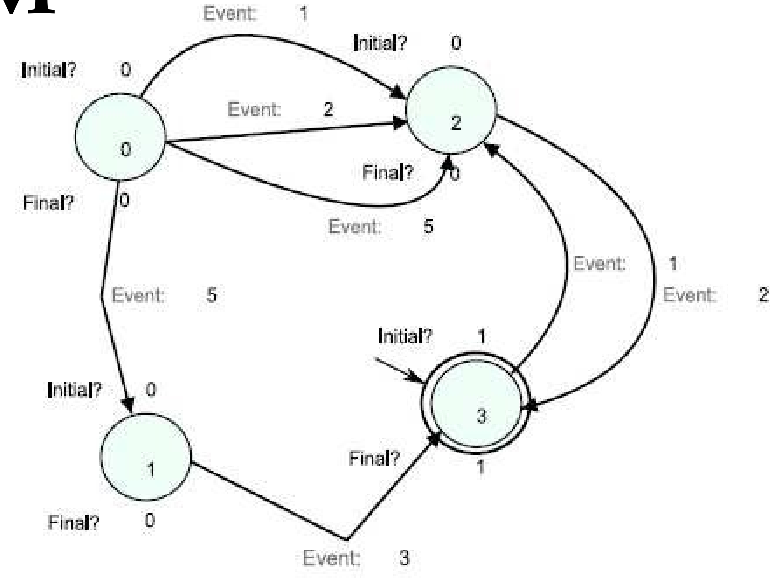
Domain-specific Model Editor



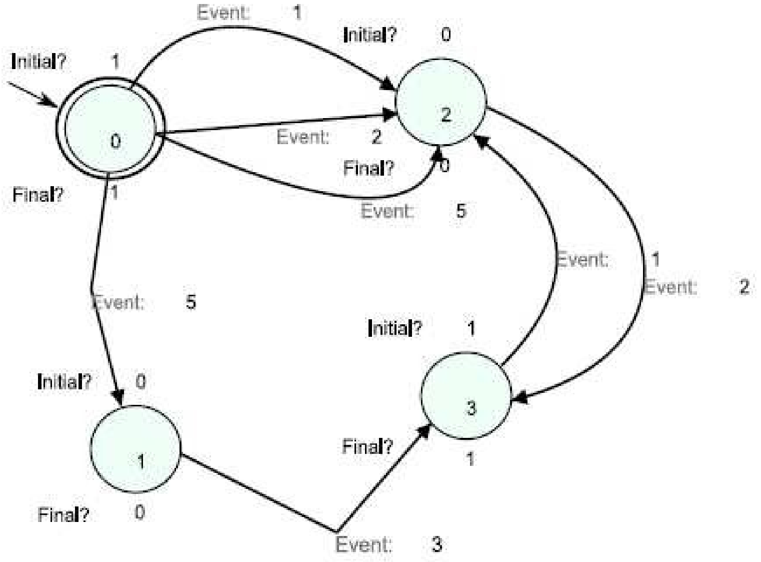
Model Completion example of the AToM^3 FSM



Partial Model



Recommendation 1 (Time taken = 1.39 s)

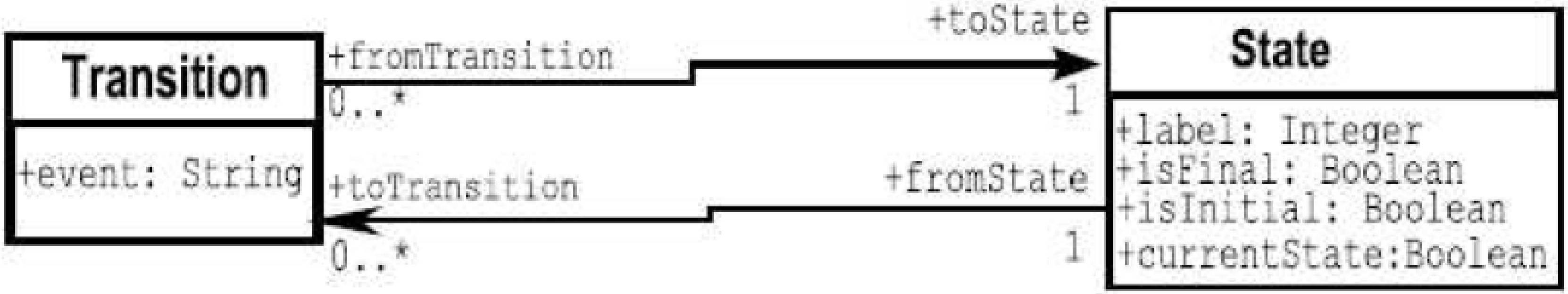


Recommendation 2 (Time taken = 3.5 s)

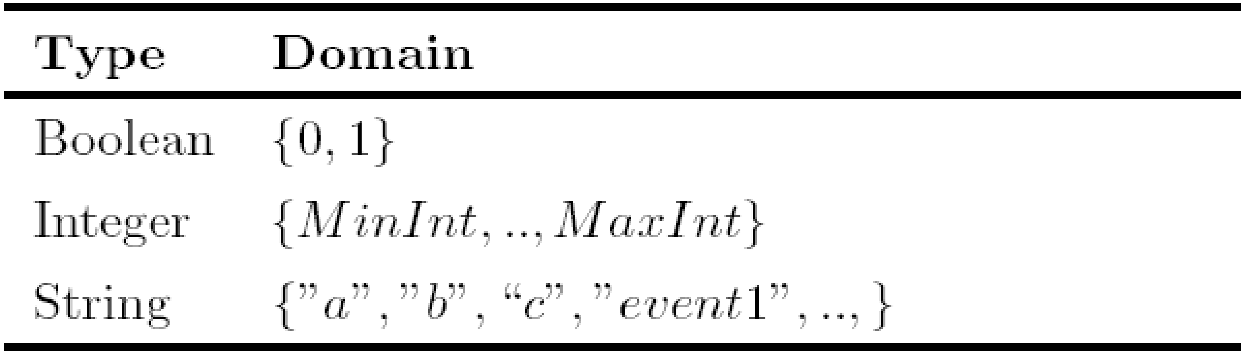
A SWRL rule example

- A UML state chart finite state machine based on the AToM^3 example
  - For comparison purpose
    - To Shuttle, SWRL rule source is *open*,
      - can be translated from in-model constraints
      - or from third-party libraries
  - As a test case
    - UML2 State Machine editor is bundled in Eclipse Ganymede

Meta-model of the AToM^3 example



The Finite State Machine Meta-model



Domains for Primitive Datatypes

Type	Domain
Boolean	$\{0, 1\}$
Integer	$\{MinInt, .., MaxInt\}$
String	$\{”a”, ”b”, “c”, ”event1”, .., \}$

Rules correspondent to the meta-model

- UML diagram part
  - By UML to RDF converters/transformers?
    - Duet (http://projects.semwebcentral.org/projects/codip/)
      - Support ONLY older version 1.3/1.4 of UML
        - UML 1.3 (http://infolab.stanford.edu/~melnik/rdf/uml/), State Machine (http://infolab.stanford.edu/~melnik/rdf/uml/uml-state-20000507.rdf) meta-model in RDF is available
    - Eclipse Atlas Transformation Language (ATL) UML2OWL transformation (http://www.eclipse.org/m2m/atl/usecases/ODMImplementation/)
      - Unknown error in processing the conversion
  - Finally manual transformations with model completion recommending scenarios

Rules correspondent to the meta-model (cont'd)

- Table part - manually converted rules
  - Boolean as xsd:boolean
  - Integer as xsd:integer
  - String as

```
DatatypeProperty( #event
  range( oneOf( “a” “b” “c” “event1” ... )
  ) )
```

Manually combined rules from both diagram and table part

- For diagram classes

  -
    ```
    Class( State 'partial'
      restriction( #label cardinality(1) )
      restriction( #label allValuesFrom(xsd#integer) )
      restriction( #isFinal cardinality(1) )
      restriction( #isFinal allValuesFrom(xsd#boolean) )
      restriction( #isInitial cardinality(1) )
      restriction( #isInitial allValuesFrom(xsd#boolean) )
      restriction( #currentState cardinality(1) )
      restriction( #currentState allValuesFrom(xsd#boolean) )
      )
    ```

For diagram classes (cont'd)

```
– Class( Transition 'partial'
    restriction( #event cardinality(1) )
    restriction( #event
      someValuesFrom( oneOf( “a” “b” “c”
      “event1” ... ) ) )
```

For diagram associations (including explicit and implicit navigability & cardinality)

- `ObjectProperty( toState Functional domain(#Transition) range(#State) )`
- `ObjectProperty( toTransition InverseFunctional domain(#State) range(#Transition) )`
- `ObjectProperty( fromState inverseOf(#toTransition) Functional )`
- `ObjectProperty( fromTransition inverseOf(#toState) InverseFunctional )`

For the Prolog constraint rules

- Correspondent SWRL rules maybe (given *only one machine* per diagram):
  - AtLeastOneFinalState: sum(listOfisFinal,>=,1)

```
fsm:FinalState( ?x ) ⇒ swrlb:member( ?x, ?fslist )
builtIn(
    swrlb:booleanNot,
    swrlb:empty( ?fslist ), 'true' )
shuttle:subLabel( ?x, 'final' ) ^
  shuttle:subLabel( ?x, 'state' ) ^
  shuttle:subLabel( ?x, 'false' )
    ⇒ builtIn(
        swrlb:booleanNot,
        fsm:FinalState( ?x ), 'true' )
```

For the Prolog constraint rules (cont'd)

- exactlyOneInitial: occurrences(listOfisInitial,1,1)

```
fsm:InitialState( ?x )
   ⇒ builtIn(
       swrlb:booleanNot,
       fsm:InitialState( ?y ), 'true' )
shuttle:subLabel( ?x, 'initial' ) ^
 shuttle:subLabel( ?x, 'state' ) ^
 shuttle:subLabel( ?x, 'false' )
   ⇒ builtIn(
       swrlb:booleanNot,
       fsm:InitialState( ?x ), 'true' )
```

For the Prolog constraint rules (cont'd)

- alldifferent: all_different(listOfVariables)

```
fsm:State( ?x ) ^ fsm:label( ?x, ?a ) ^
 fsm:State( ?y ) ^ fsm:label( ?y, ?a )
   ⇒ sameAs( ?x, ?y )
```

- *RDF-style lists* are NOT supported by **OWL-DL**, so we have to deal with them specifically

Scenarios of recommendings

1. New element recommending
    - Recommending "AtLeastOneFinalState"
2. Modified element recommending
    1. Partial modification
    2. Deletion

Process of model completion recommending

1. Pre-modeling knowledge base construction
    - Meta-model axioms + constraint rules
2. Incremental modeling:
    - Empty model or
    - Current element binding → KAON2 axiom building →
3. Rule triggering
    - KAON2 (OWL-DL)/Shuttle rule inference → propagating inference results upward

Process of model completion recommending (cont'd)

4 Recommendation activating:

- inference results on MPM →
  - new element extraction / current element binding from results →
  - recomm. (completion) model construction →
  - hint activation →
  - recomm. model displaying

5 Reaction to applying/denying recommendation

Nested (recursive) MPM-PM rule triggering/recommending

- Hierarchical PMs
  - Applied with hierarchical MDA inference
  - For *reusable* and *scalable* PM rules
- **Downward triggering**
  - MPM→PM
  - PM→PM
  - antecedent→consequent
  - Unconditional fact (OWL fact)

Nested (recursive) MPM-PM rule triggering/recommending (cont'd)

- **Upward recommending**
  - PM→PM
  - PM→MPM
  - consequent→antecedent
  - Unconditional fact

Open (unbounded) domain inference

- Compared to AToM^{3}'s closed (pre-compiled) domain inference
  - More *flexible & scalable* for usual documents with *heterogeneous concepts*

1 **Heterogeneous concept inference**

  - ex. coexisting UML State Machine & other (application) domain concepts
  - For the highest scalability
    - Various inference result recommendations depending on *the scope of MPM*

Open domain inference (cont'd)

- **Open model element bindings**
  - Exhaustive (wildcard) binding
  - To RDF object/property/subject
  - To SWRL variables
  - To other domains to form multiple MPMs

Open domain inference (cont'd)

3 **Open editor (MPM) rules**

- *Native editors*
  - From editor code/API
  - UML2 State Machine editor example
    - GMF editor definitions as MPM
- *Potential editors*
  - From common PM tagging
  - The FSM-over-UML2 example
    - FSM rules as MPM

MPM tagging

- **For native MPM**
  - Extracting GMF editor rules…
- **For potential MPM**
  - By *ontology mapping*
  - Firstly parsing ontology for *concept mapping*
    - ex. parsing user-given FSM rules

Built-in PMs

- SWRL PM
  - Handling **variable binding** to model elements
    - Exhaustive (wildcard) binding
  - Handling *RDF-style list* related SWRL built-in atoms
    - `swrlb:member`, `swrlb:empty`, etc.
- RDF PM
  - Handling *RDF-style lists*
    - *List* implies collection of model elements
  - **SWRL_PM↔RDF_PM** triggering/recommending
  - Model element bindings for object/property/subject

Built-in PMs (cont'd)

- Shuttle PM (MPM)
  - Handling complex label text issues
- **PM priority** to handle *contradiction*
  - For now, Shuttle PM

    > native MPM(s)
    > potential MPM(s)
    > other built-in PMs

Built-in inference

- **Description logic inference**: Using KAON2's built-in inference API
  1. **Equivalence** of descriptions
  2. **Satisfiability** of descriptions
  3. **Subsumption** of descriptions

Research topics

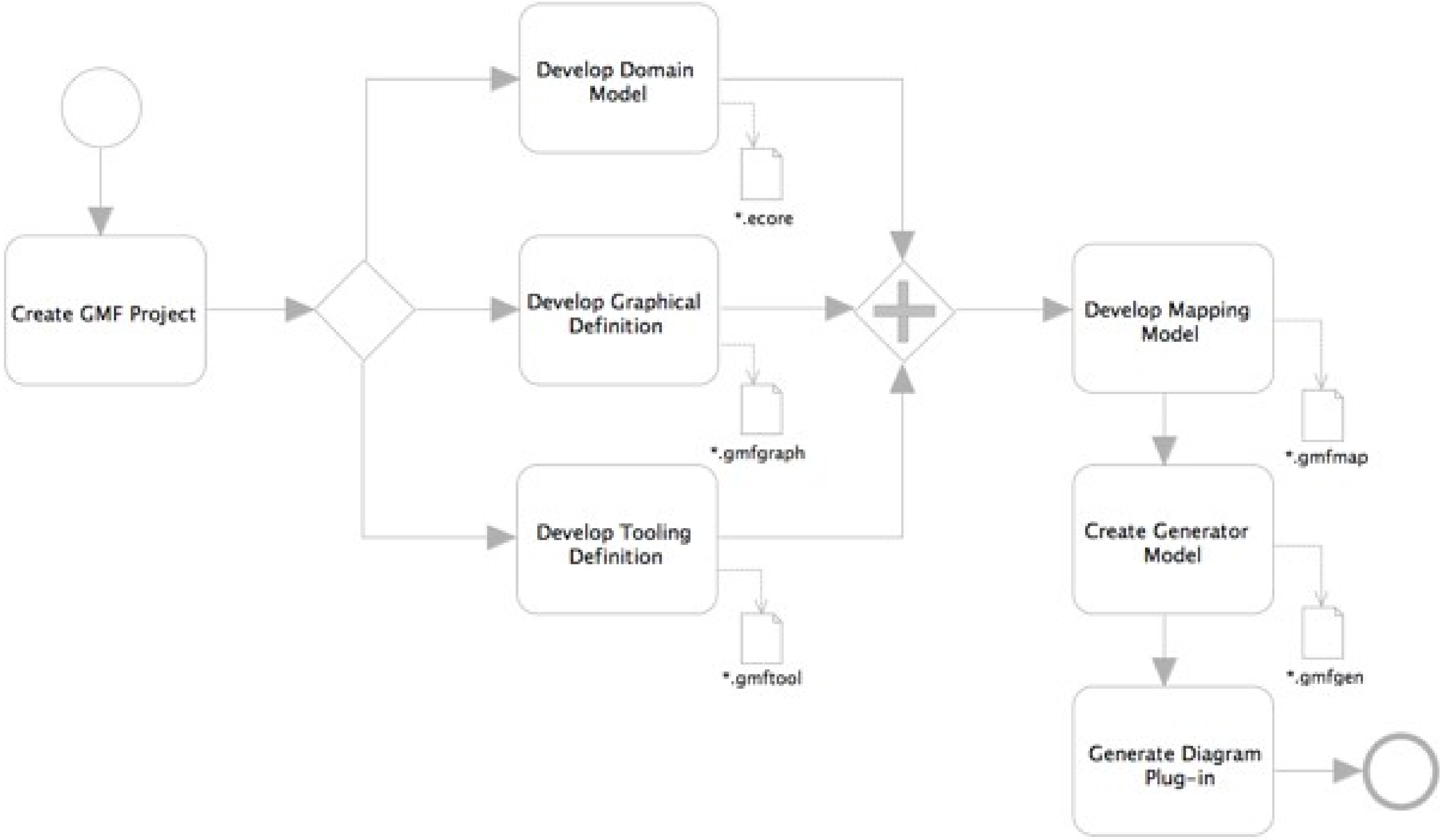
I. Inferring model via Concept mapping technique (with client-server collaboration)

II.1. Generating rules from GMF editor definitions for model completion recommendation

II.2. Extracting GMF editor rules for model completion recommendation without editor definitions

III. MDA-based inference for model completion recommendation visualization

Generating rules from GMF editor definitions for model completion recommendation



Generating rules from GMF editor definitions for model completion recommendation (cont'd)

- **GMF editor definitions**
  - Domain Model Definition
  - Graphical Definition
  - Tooling Definition
  - Mapping Definition
- **Model completion recommendation** – rule inference result recommendation visualization

Research topics

I. Inferring model via Concept mapping technique (with client-server collaboration)

II.1. Generating rules from GMF editor definitions for model completion recommendation

II.2. Extracting GMF editor rules for model completion recommendation without editor definitions

III. MDA-based inference for model completion recommendation visualization

Extracting GMF editor rules for model completion recommendation without editor definitions

- When editor definitions are not available
  - ex. hidden source
- Extracting by…
  - Is there any right-prepared rum-time editor API?
  - Or reverse engineering / code mining?
- A universal solution

Tracing bundled GMF editor code

- UML2 editors
  - Beginning with palette creation tool title (`Messages.`*`(...)CreationTool_title`*)
    - Not easy to get the bound view through creation tools
  - Or get model *element name* & *view* via the element type registry?
    - Studying **GMF Extensible Type Registry**
- Other simple example editors
  - The simple digital logic editor

Tracing bundled GMF editor code (cont'd)

- Investigating: element title in pallete → creation tool → element EditPart view (without EditPart model, just for *possibly temporary recommendations*)
- Studying **GMF Extensible Type Registry**
- Code-let prototype

Logical and physical recommendations

- Logical recommendations
  - Rule inference results
- Physical recommendations
  - Results to be *displayed*
  - Using **recommendation factory** for adapting various displaying techniques
    - ex. cloned, gray-scale or accessibility styles

    For prototype development & future extension...