The PSM Librarian: Configuring Problem-solving Applications with Protégé

Monica Crubézy

Stanford Medical Informatics

~~ July 2003 ~~

Reasoning with knowledge bases

- Knowledge bases (KBs) encode reference models and facts in a domain
  - a set of clinical guidelines for hypertension care
  - a set of components and constraints about elevators
  - an anatomy ontology
- KBs further provide a basis for performing reasoning tasks—or **problem solving**
  - diagnosis, therapy advising, design, classification, …

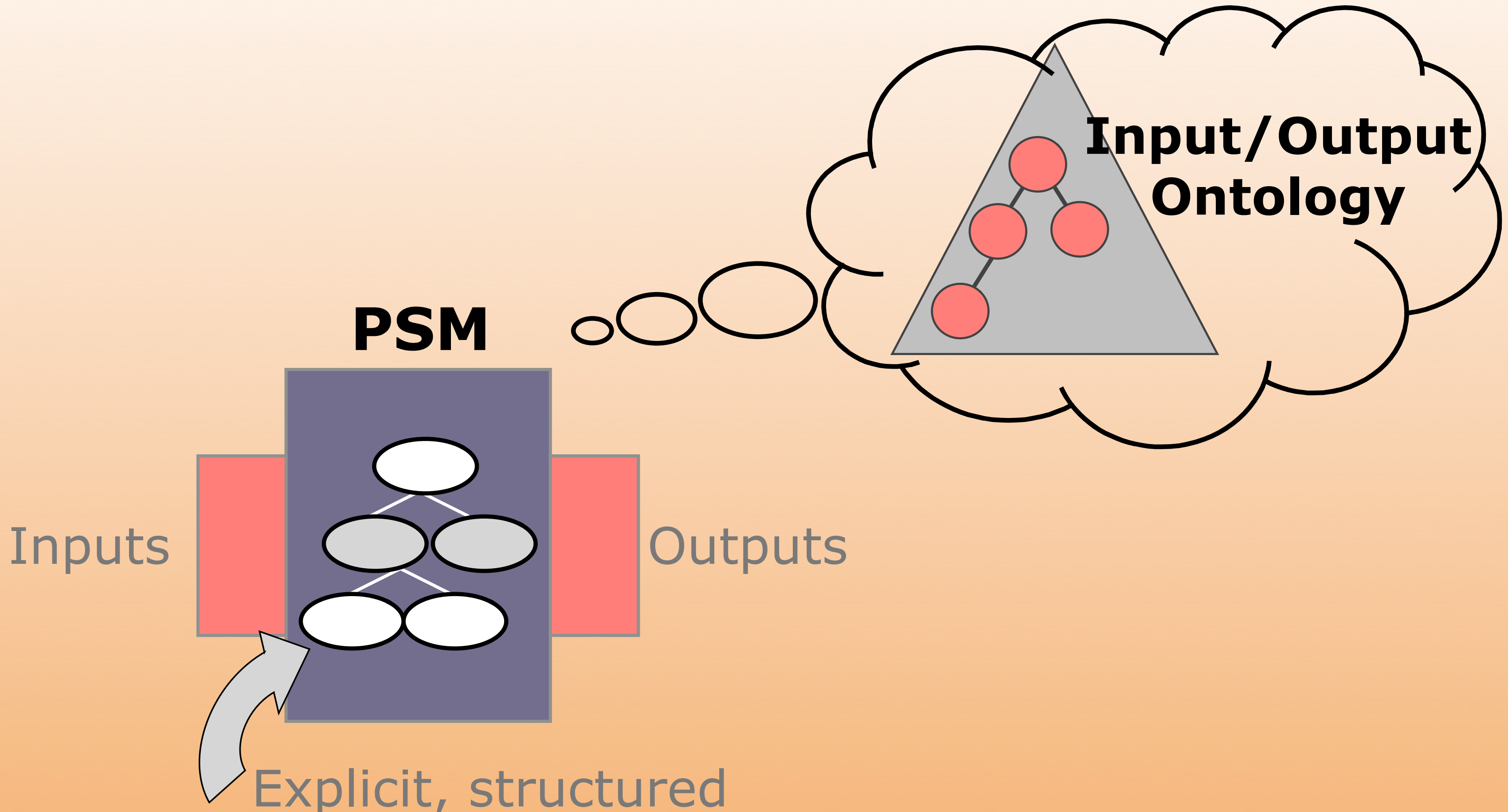
Component-based knowledge systems

- Alternative to rule-based inference systems
- **Separate** problem-solving component(s) encode(s) the reasoning process of the system
  - Reasoning behavior is explicit and understandable
  - Maintenance of system’s behavior facilitated
- KB **only** contains domain models and facts
  - Domain knowledge is explicit, understandable and maintainable too
  - Several problem-solving components can rely on the same corpus of domain knowledge

Problem-Solving Methods (PSMs)

- Standard, explicit algorithms that address stereotypical tasks
  - Design, classification, diagnosis
- Domain-independent components that abstract the reasoning process from factual knowledge
  - Reusable for different applications and domains
  - The *Propose-and-Revise* PSM: configuring elevator designs, predicting conformations of ribosomal units
- Collected and indexed in **libraries** for reuse

Model of a PSM

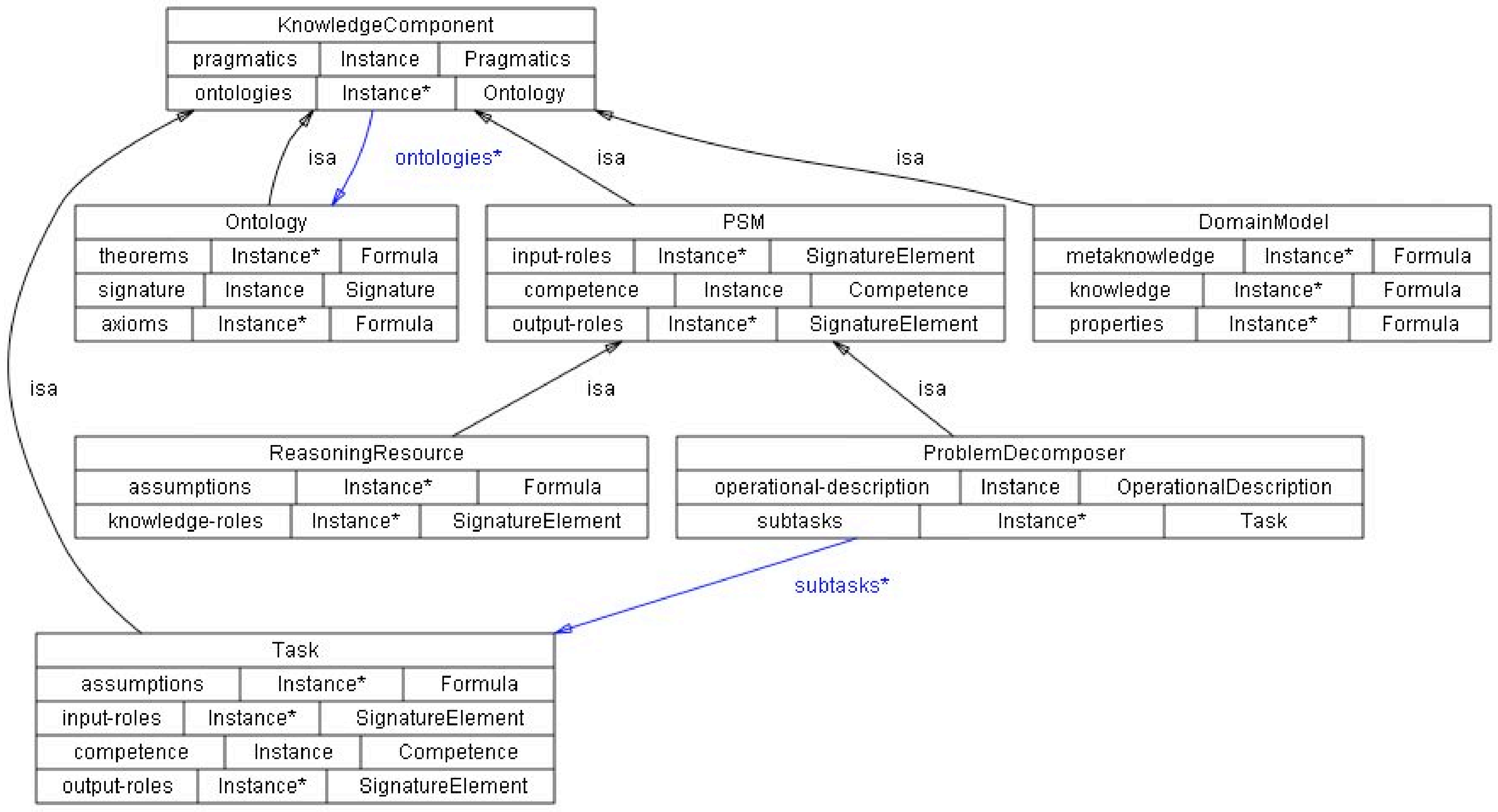


Explicit, structured representation of problem-solving knowledge

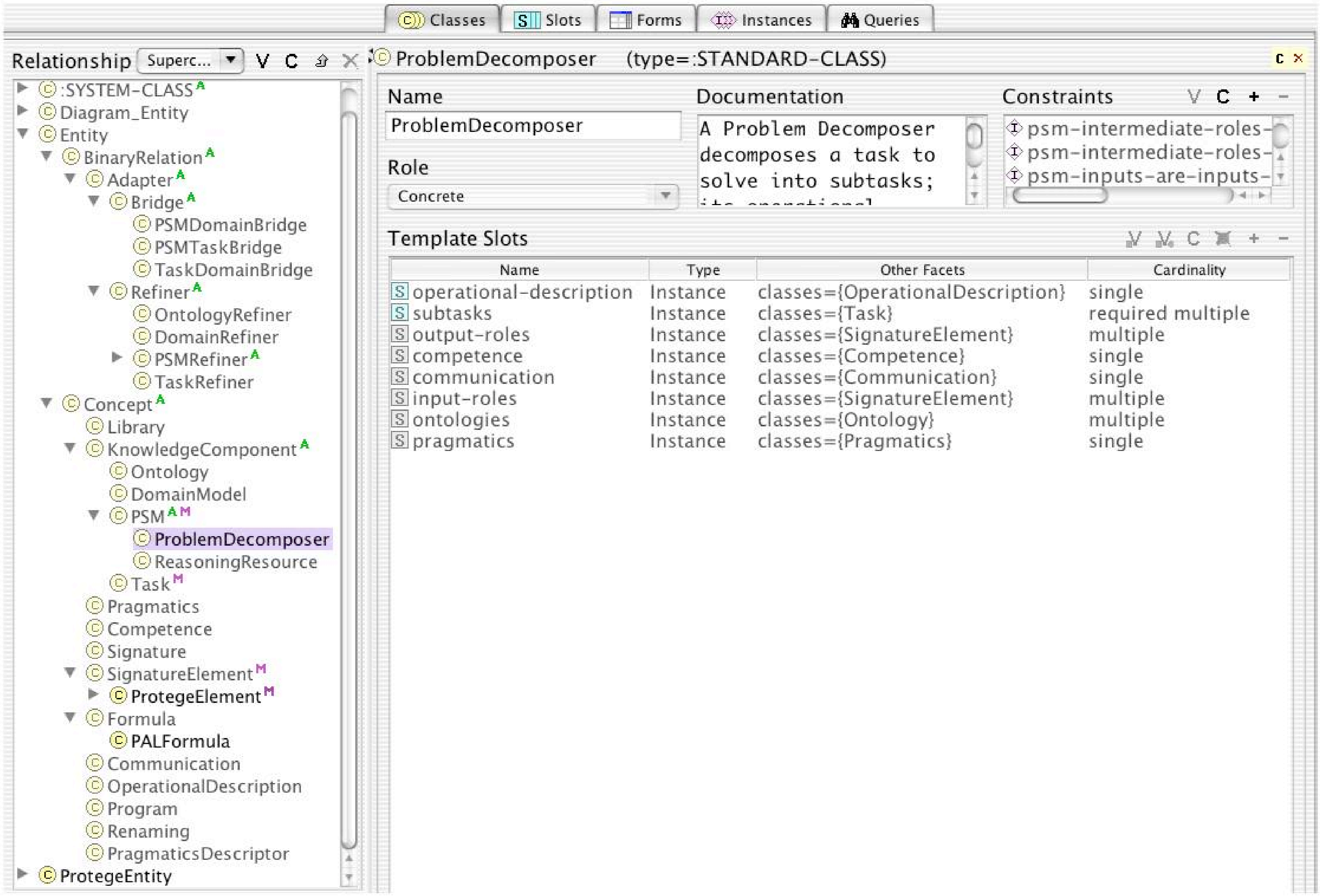
Describing libraries of PSMs

- Formal modeling & metadata annotation of PSMs in context
- The Unified Problem-solving Method development Language (UPML)
  - Task-Method decomposition paradigm
  - PSM: *pragmatics*, *input-output roles*, *pre/postconditions*, *knowledge assumptions*, *subtasks* & *control*
  - Ontology-based modeling of knowledge components
  - Domain/Task/PSM component-adaptation approach (*bridges* & *refiners*)

The UPML ontology


KnowledgeComponent
pragmatics | Instance | Pragmatics
ontologies | Instance* | Ontology
isa
ontologies*
isa
isa
Ontology
theorems | Instance* | Formula
signature | Instance | Signature
axioms | Instance* | Formula
PSM
input-roles | Instance* | SignatureElement
competence | Instance | Competence
output-roles | Instance* | SignatureElement
DomainModel
metaknowledge | Instance* | Formula
knowledge | Instance* | Formula
properties | Instance* | Formula
isa
isa
isa
ReasoningResource
assumptions | Instance* | Formula
knowledge-roles | Instance* | SignatureElement
ProblemDecomposer
operational-description | Instance | OperationalDescription
subtasks | Instance* | Task
subtasks*
Task
assumptions | Instance* | Formula
input-roles | Instance* | SignatureElement
competence | Instance | Competence
output-roles | Instance* | SignatureElement

The UPML ontology



- :SYSTEM-CLASS A
- Diagram_Entity
- Entity
  - BinaryRelation A
    - Adapter A
      - Bridge A
        - PSMDomainBridge
        - PSMTaskBridge
        - TaskDomainBridge
      - Refiner A
        - OntologyRefiner
        - DomainRefiner
        - PSMRefiner A
        - TaskRefiner
  - Concept A
    - Library
    - KnowledgeComponent A
      - Ontology
      - DomainModel
      - PSM A M
        - ProblemDecomposer
        - ReasoningResource
      - Task M
    - Pragmatics
    - Competence
    - Signature
    - SignatureElement M
      - ProtegeElement M
    - Formula
      - PALFormula
    - Communication
    - OperationalDescription
    - Program
    - Renaming
    - PragmaticsDescriptor
- ProtegeEntity

ProblemDecomposer (type=:STANDARD-CLASS)

Name: ProblemDecomposer

Role: Concrete

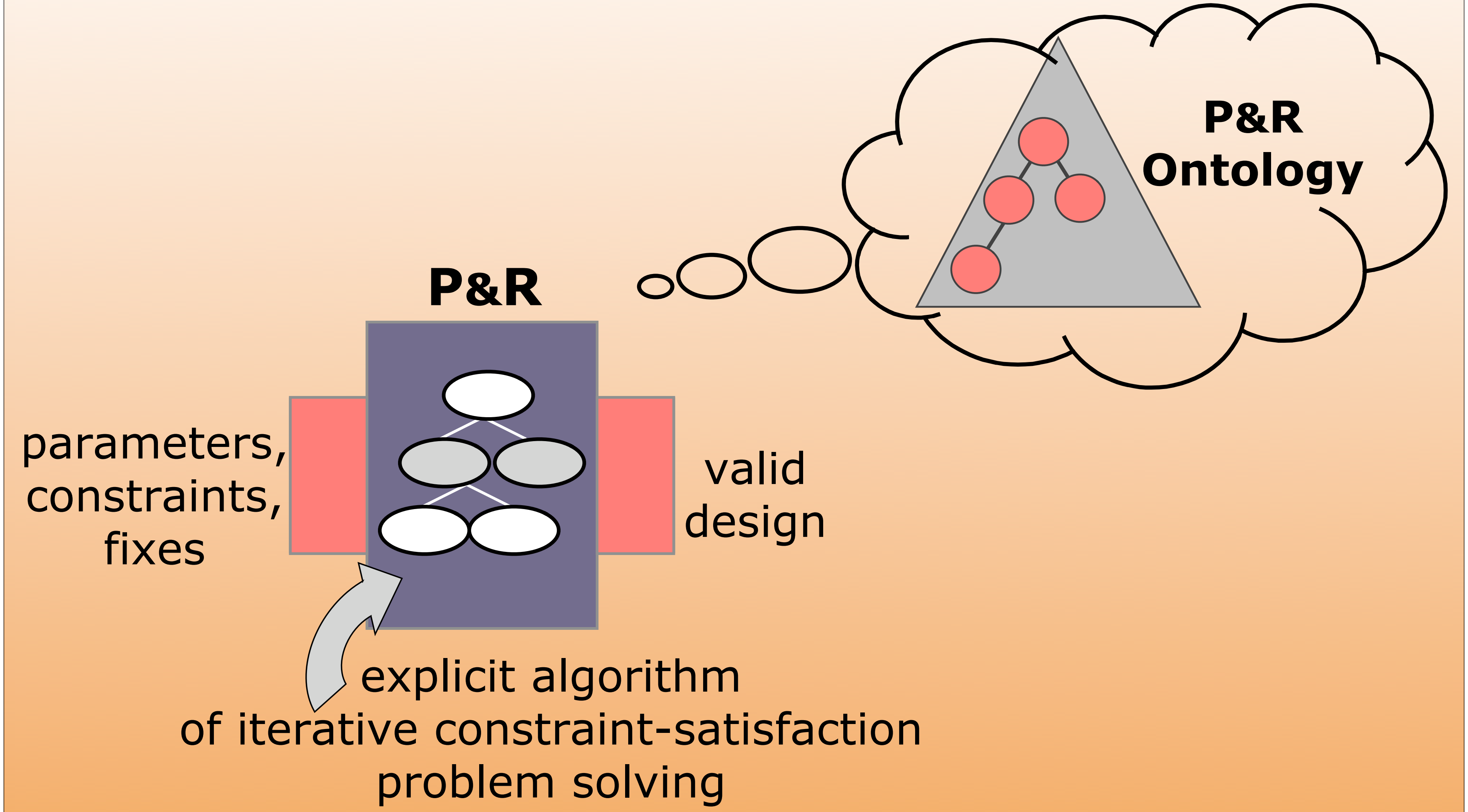
Documentation: A Problem Decomposer decomposes a task to solve into subtasks; its operational

Constraints:
- psm-intermediate-roles-
- psm-intermediate-roles-
- psm-inputs-are-inputs-

Template Slots

Name	Type	Other Facets	Cardinality
operational-description	Instance	classes={OperationalDescription}	single
subtasks	Instance	classes={Task}	required multiple
output-roles	Instance	classes={SignatureElement}	multiple
competence	Instance	classes={Competence}	single
communication	Instance	classes={Communication}	single
input-roles	Instance	classes={SignatureElement}	multiple
ontologies	Instance	classes={Ontology}	multiple
pragmatics	Instance	classes={Pragmatics}	single

The *Propose-and-Revise* PSM


P&R
Ontology
P&R
parameters, constraints, fixes
valid design
explicit algorithm of iterative constraint-satisfaction problem solving

UPML model of *Propose-and-Revise*

Pragmatics
title: Propose and Revise
resource: ChronBackPnR.clp
...
Ontology
element: *parameter*, as defined by class *stateVariable*
element: *constraint*, as defined by class *Constraint* and its subclasses
element: *fix*, defined by class *Fix* as “A condition-expression rule associated to a constraint and a parameter"
element: *consistent*, defined as a logical predicate
...
Input-roles: *parameters*, *constraints*, *fixes*
Output-roles: *parameter values*
Subtasks: *Select* next parameter, *Propose* next set of parameter values, *Verify* against constraints, *Revise* according to fix knowledge.
Competence
preconditions: “Every fix has exactly one associated constraint." ...
postconditions: “The output parameter values are consistent regarding the constraints." ...
Operational Description
...

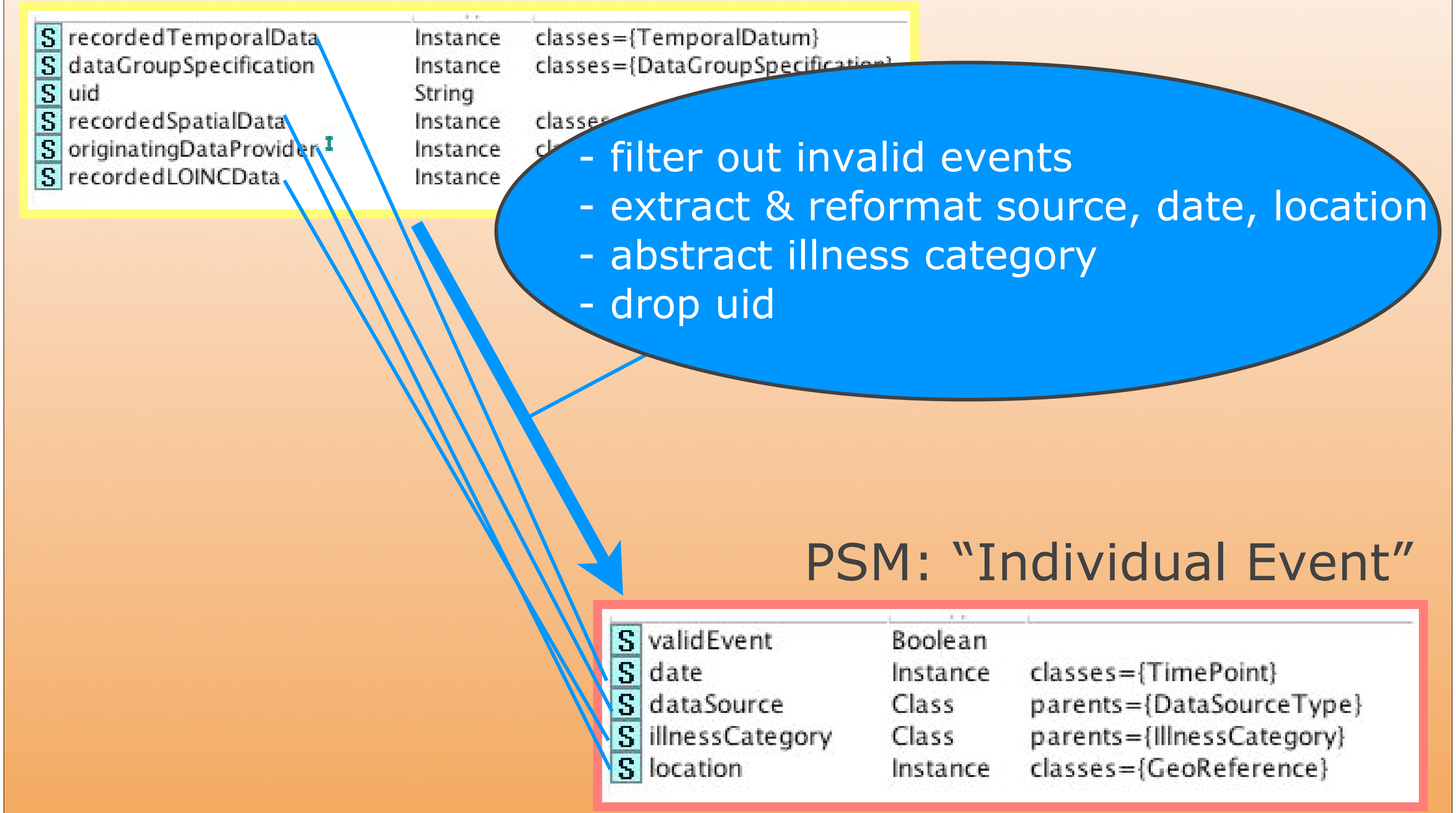
Configuring PSMs for an application

- A problem-solving method (PSM)
  - processes domain knowledge & data in an abstract way
  - defines an ontology of its inputs and outputs in a domain-independent way--*input–output ontology*

- Domain knowledge (defined by a domain ontology) needs to be construed in terms of the PSM’s input–output ontology

Conceptual and syntactic mismatch

Domain: “Data Group”



S	recordedTemporalData	Instance	classes={TemporalDatum}
S	dataGroupSpecification	Instance	classes={DataGroupSpecification}
S	uid	String	
S	recordedSpatialData	Instance	classes
S	originatingDataProvider	Instance	cl
S	recordedLOINCData	Instance	

- filter out invalid events
- extract & reformat source, date, location
- abstract illness category
- drop uid

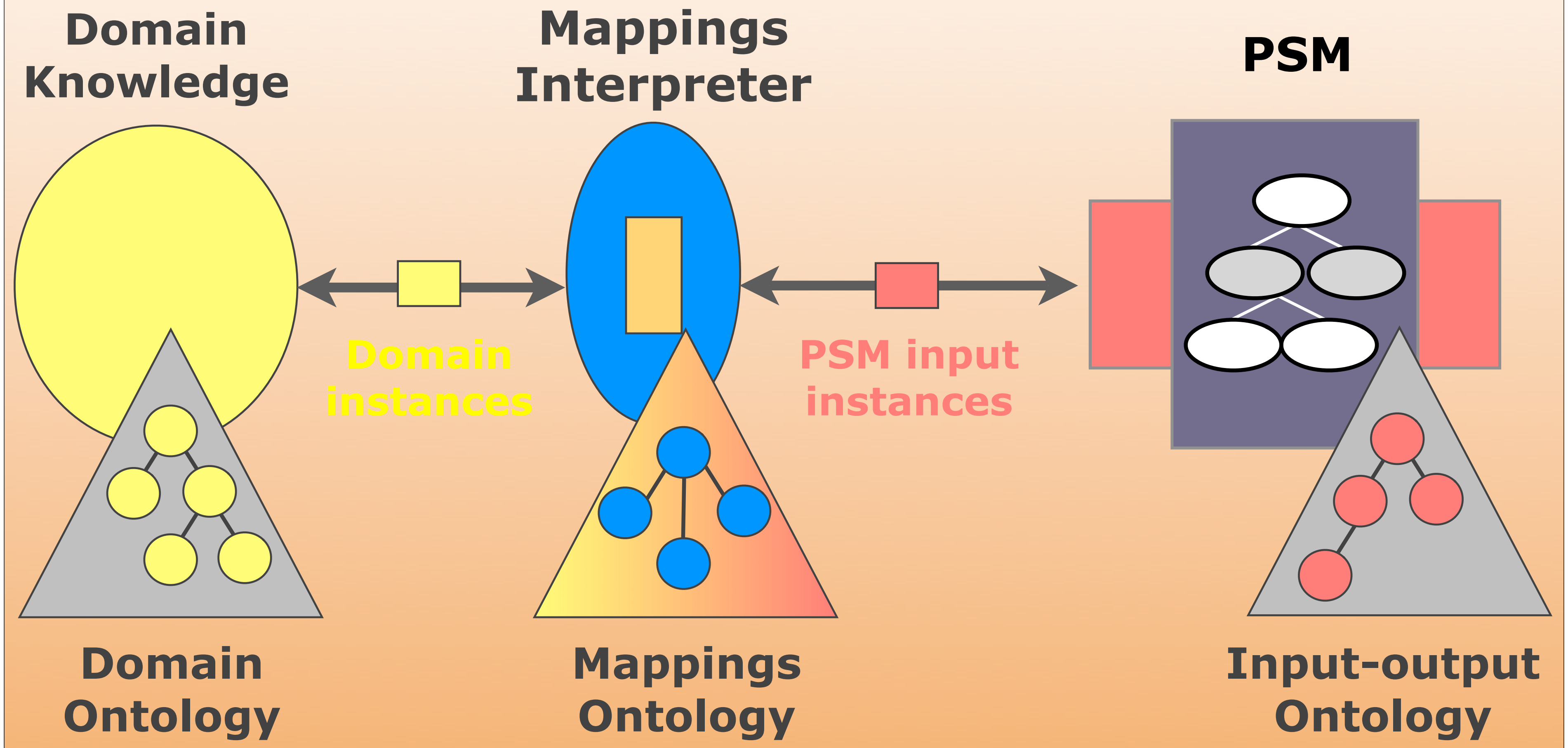
PSM: “Individual Event”

S	validEvent	Boolean	
S	date	Instance	classes={TimePoint}
S	dataSource	Class	parents={DataSourceType}
S	illnessCategory	Class	parents={IllnessCategory}
S	location	Instance	classes={GeoReference}

Data & knowledge exchange

- Conceptual mapping
  - change in domain of discourse
  - difference in the level of knowledge granularity
  - split and join of concepts & attributes
- Value transformation
  - abstraction, reduction
  - aggregation or dispatch
  - format change (unit change)
  - custom computation (functional transformation)

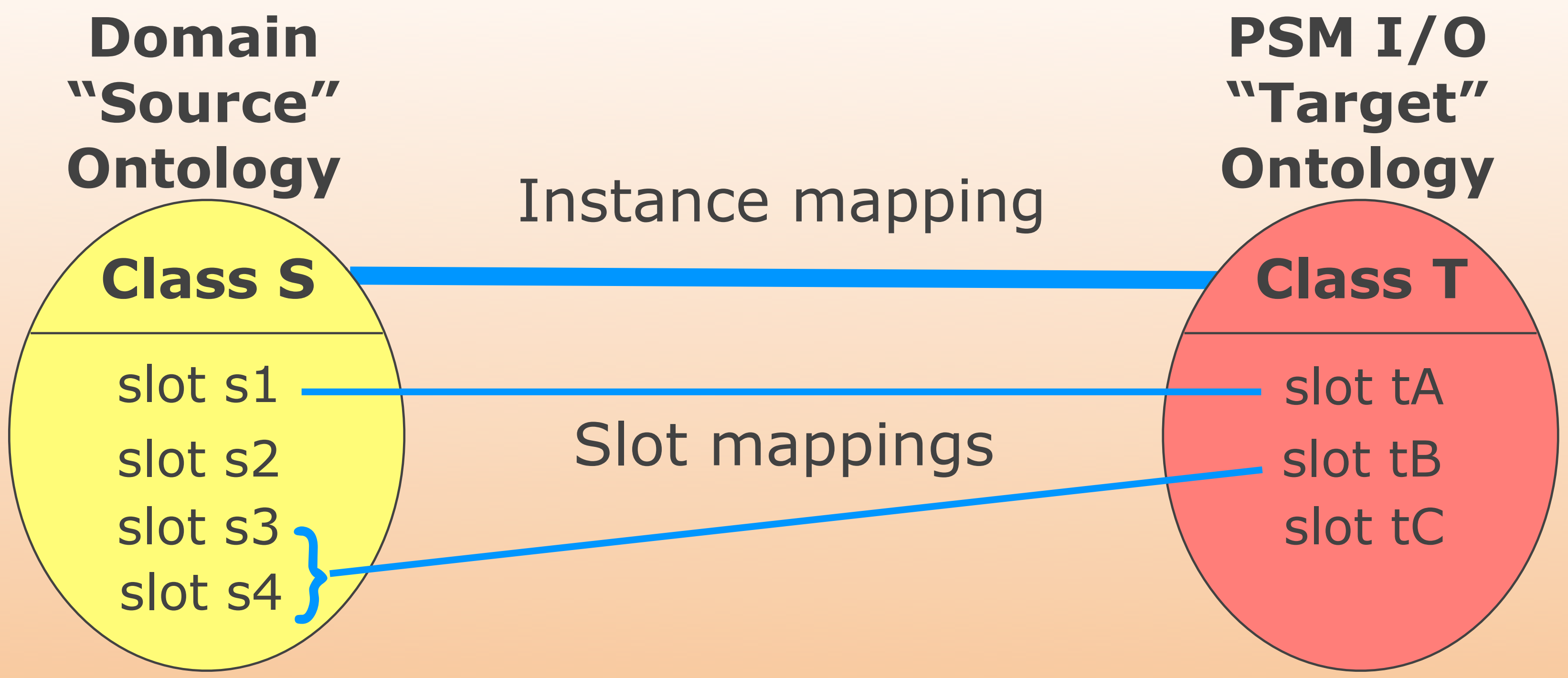
Ontology-mapping approach



Mapping relations

- Domain and PSM each define an ontology of its working concepts (with their attributes)
- A set of mapping relations expresses the connections between the 2 ontologies
- Mapping relations also express rules of transformation needed to mediate data & knowledge between domain and PSM

Mapping relations

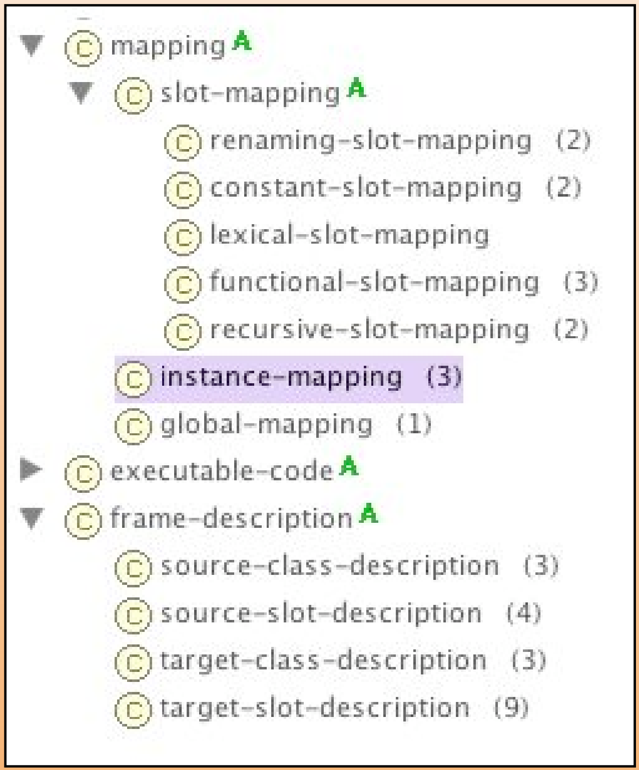


- Each instance of the target class is calculated from an instance of the source class
- The slot values of the target instance are computed according to slot-mapping expressions that involve the source instance's slot values

1. Mappings ontology

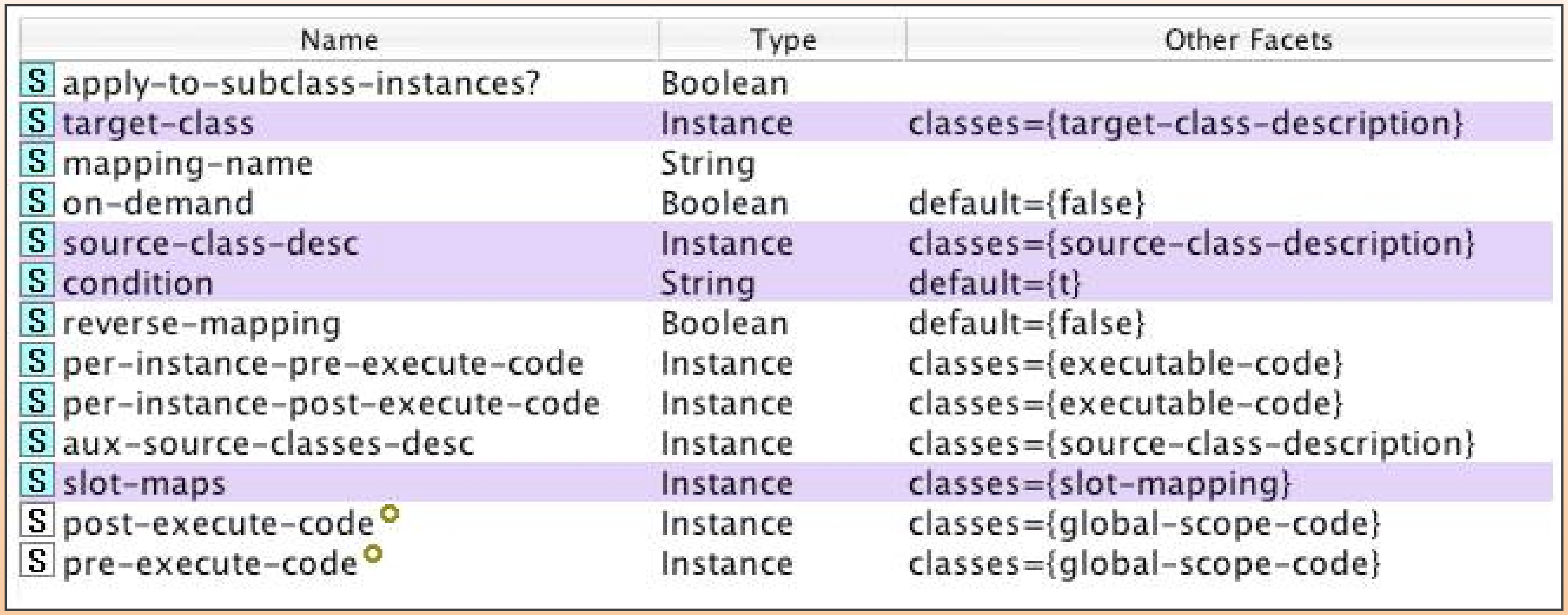
- A small, generic & controlled set of possible mapping relations between classes and slots of a source (domain) ontology and of a target (PSM I/O) ontology

- For each instance required by the target component, a specific set of rules define the transformation of source instances and their attribute values



```
▼ Ⓒ mapping A
    ▼ Ⓒ slot-mapping A
          Ⓒ renaming-slot-mapping  (2)
          Ⓒ constant-slot-mapping  (2)
          Ⓒ lexical-slot-mapping
          Ⓒ functional-slot-mapping  (3)
          Ⓒ recursive-slot-mapping  (2)
      Ⓒ instance-mapping  (3)
      Ⓒ global-mapping  (1)
▶ Ⓒ executable-code A
▼ Ⓒ frame-description A
      Ⓒ source-class-description  (3)
      Ⓒ source-slot-description  (4)
      Ⓒ target-class-description  (3)
      Ⓒ target-slot-description  (9)
```

Instance mappings



Name	Type	Other Facets
S apply-to-subclass-instances?	Boolean	
S target-class	Instance	classes={target-class-description}
S mapping-name	String	
S on-demand	Boolean	default={false}
S source-class-desc	Instance	classes={source-class-description}
S condition	String	default={t}
S reverse-mapping	Boolean	default={false}
S per-instance-pre-execute-code	Instance	classes={executable-code}
S per-instance-post-execute-code	Instance	classes={executable-code}
S aux-source-classes-desc	Instance	classes={source-class-description}
S slot-maps	Instance	classes={slot-mapping}
S post-execute-code	Instance	classes={global-scope-code}
S pre-execute-code	Instance	classes={global-scope-code}

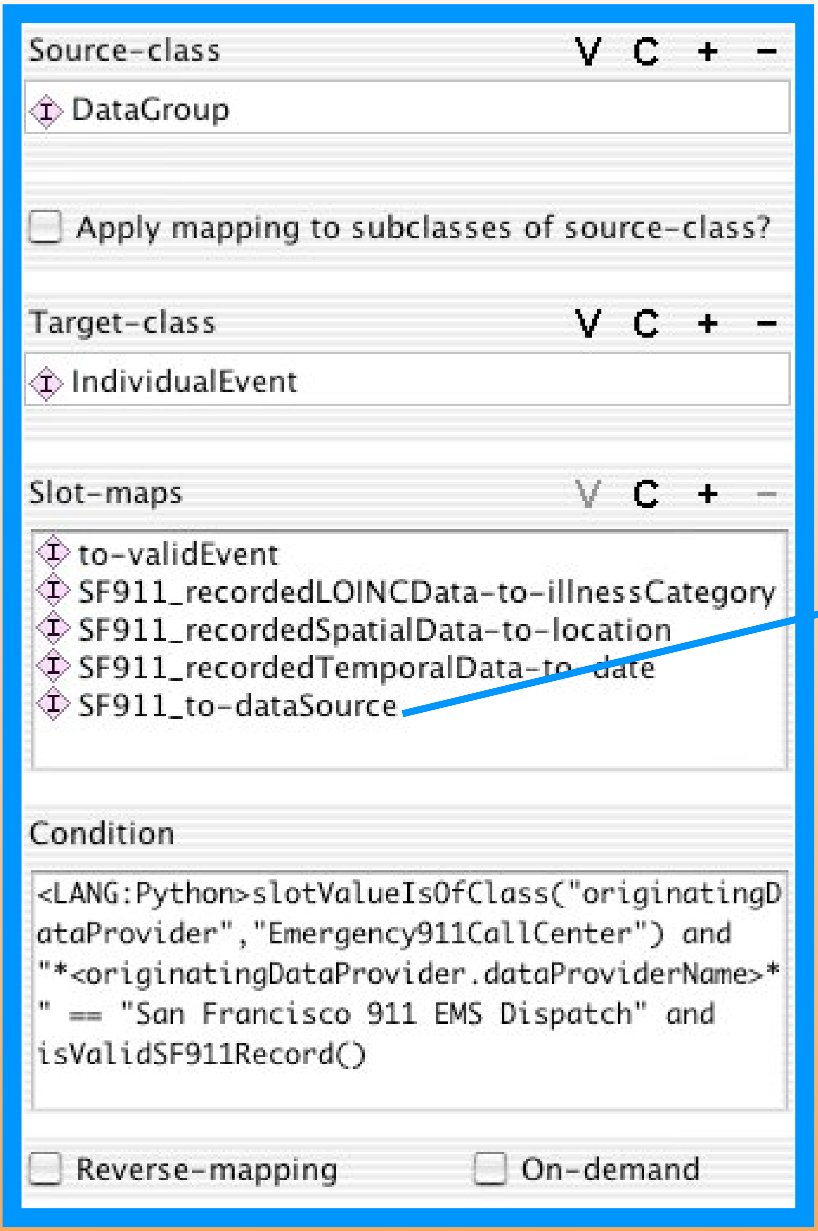
- The class **S** of source instances
- The class **T** of target instances
- A condition to filter source instances
- A set of associated slot-level mappings

Slot mappings

- The target slot (tX)
- The slot-value computation expression, possibly involving source slots (si)

  local access to (sub)instance slot values: *<s1.s11>*
- Different types of slot mappings:
  - *renaming*: value(tA) = value(s1)
  - *constant*: value(tC) = constant
  - *lexical*: value(tB) = "*<s2>* / 20*<s3>*"
  - *functional*: value(tC) = function()
  - *recursive*: value(tA) = instance (w/ auxiliary mapping)

Mapping Data Groups to Individual Events



Source-class

DataGroup

Apply mapping to subclasses of source-class?

Target-class

IndividualEvent

Slot-maps

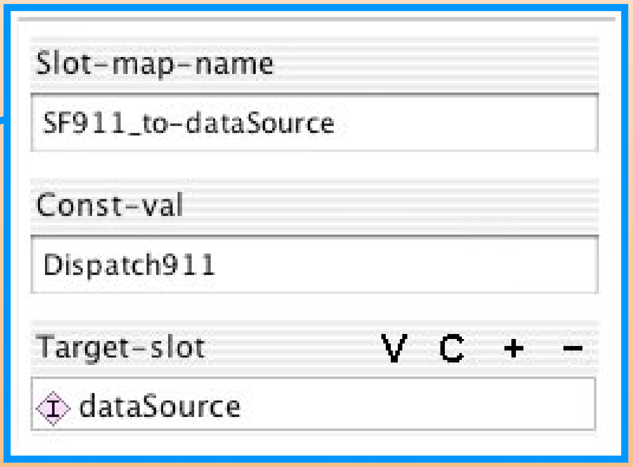
- to-validEvent
- SF911_recordedLOINCData-to-illnessCategory
- SF911_recordedSpatialData-to-location
- SF911_recordedTemporalData-to-date
- SF911_to-dataSource

Condition

```
<LANG:Python>slotValueIsOfClass("originatingD
ataProvider","Emergency911CallCenter") and
"*<originatingDataProvider.dataProviderName>*
" == "San Francisco 911 EMS Dispatch" and
isValidSF911Record()
```

Reverse-mapping

On-demand

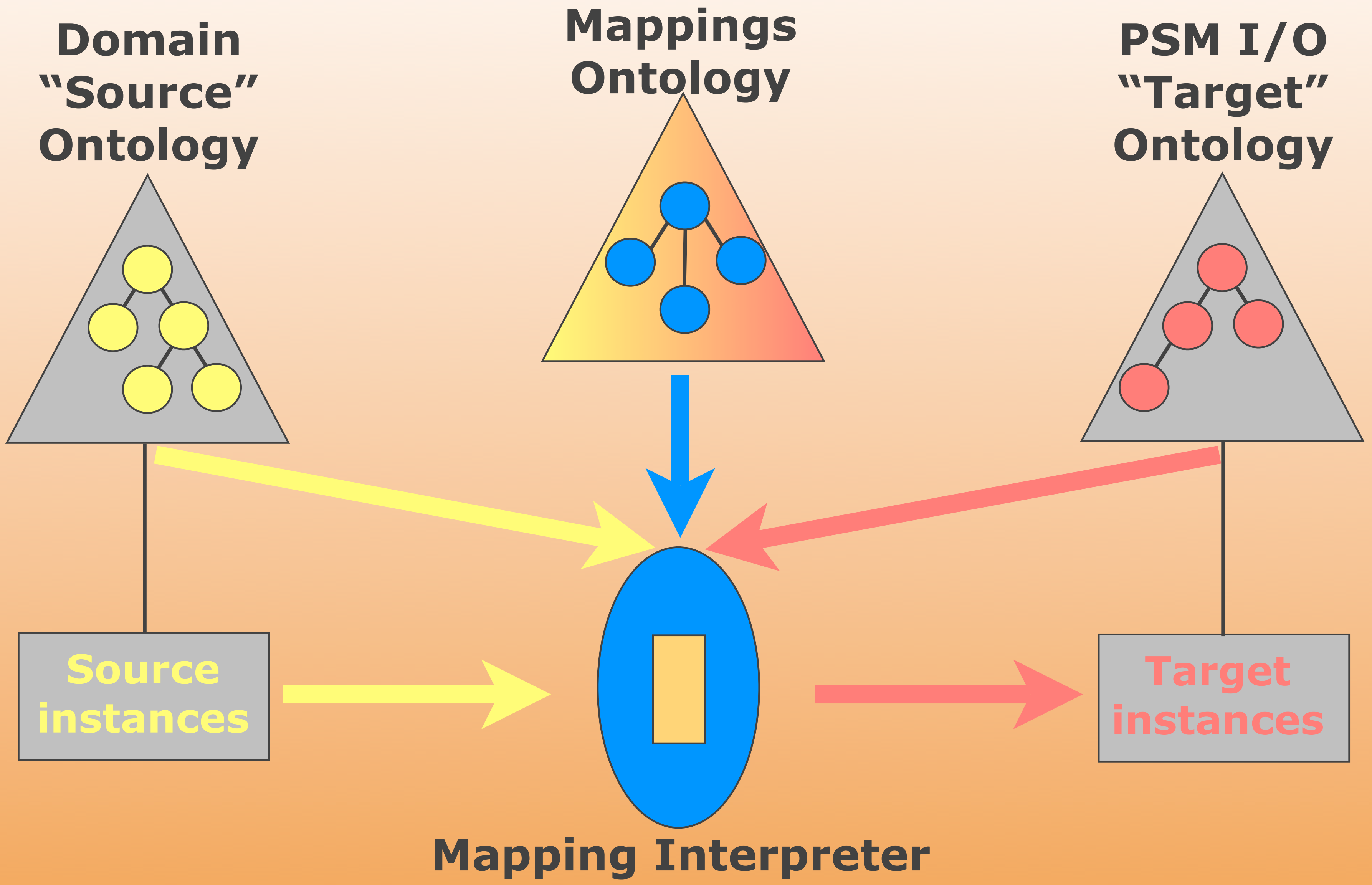


Slot-map-name: SF911_to-dataSource

Const-val: Dispatch911

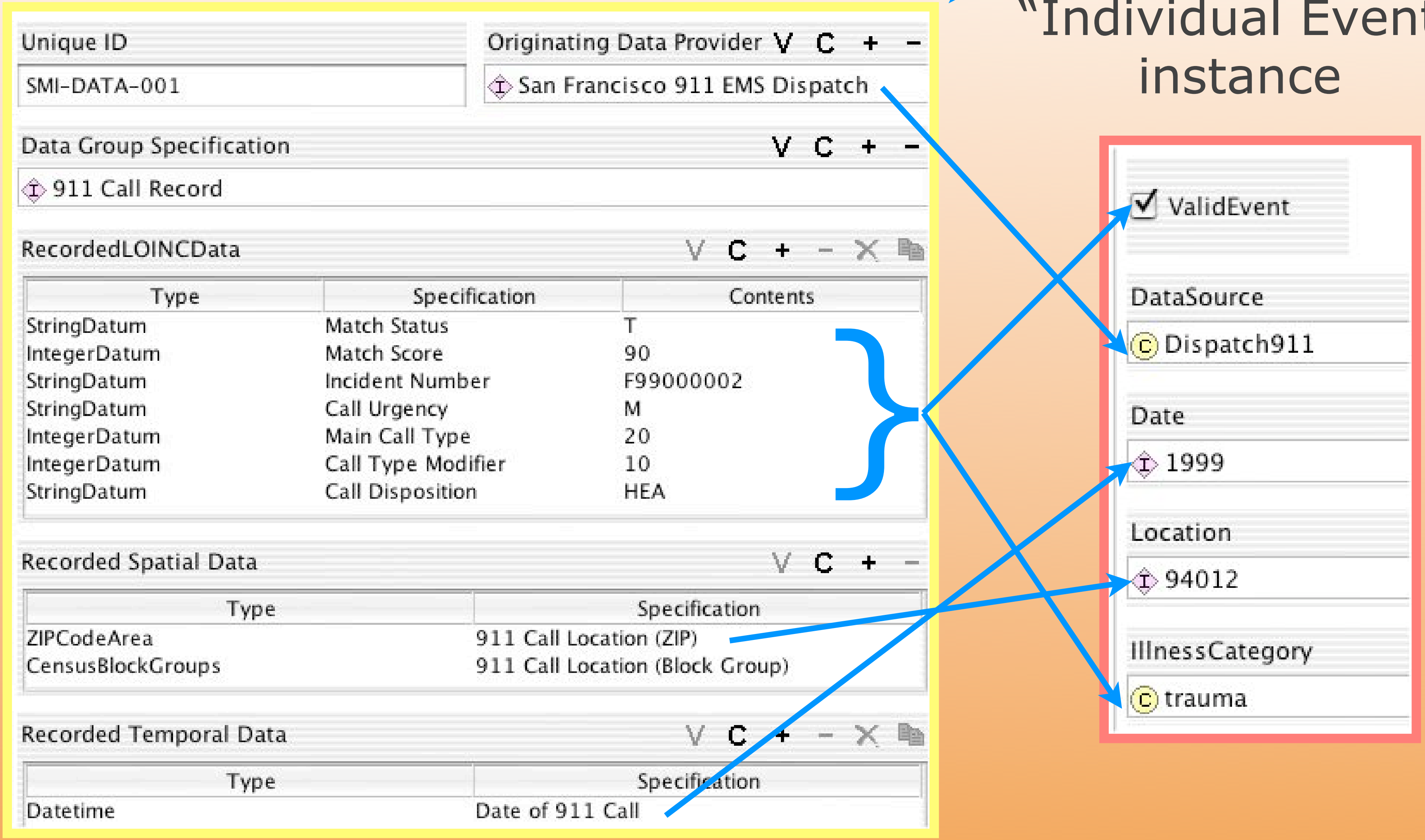
Target-slot: dataSource

2. Mapping interpreter



Results of mapping interpretation

Source "Data Group" instance → Resulting target "Individual Event" instance



Unique ID

SMI-DATA-001

Originating Data Provider

San Francisco 911 EMS Dispatch

Data Group Specification

911 Call Record

RecordedLOINCData

Type	Specification	Contents
StringDatum	Match Status	T
IntegerDatum	Match Score	90
StringDatum	Incident Number	F99000002
StringDatum	Call Urgency	M
IntegerDatum	Main Call Type	20
IntegerDatum	Call Type Modifier	10
StringDatum	Call Disposition	HEA

Recorded Spatial Data

Type	Specification
ZIPCodeArea	911 Call Location (ZIP)
CensusBlockGroups	911 Call Location (Block Group)

Recorded Temporal Data

Type	Specification
Datetime	Date of 911 Call

✓ ValidEvent

DataSource

Dispatch911

Date

1999

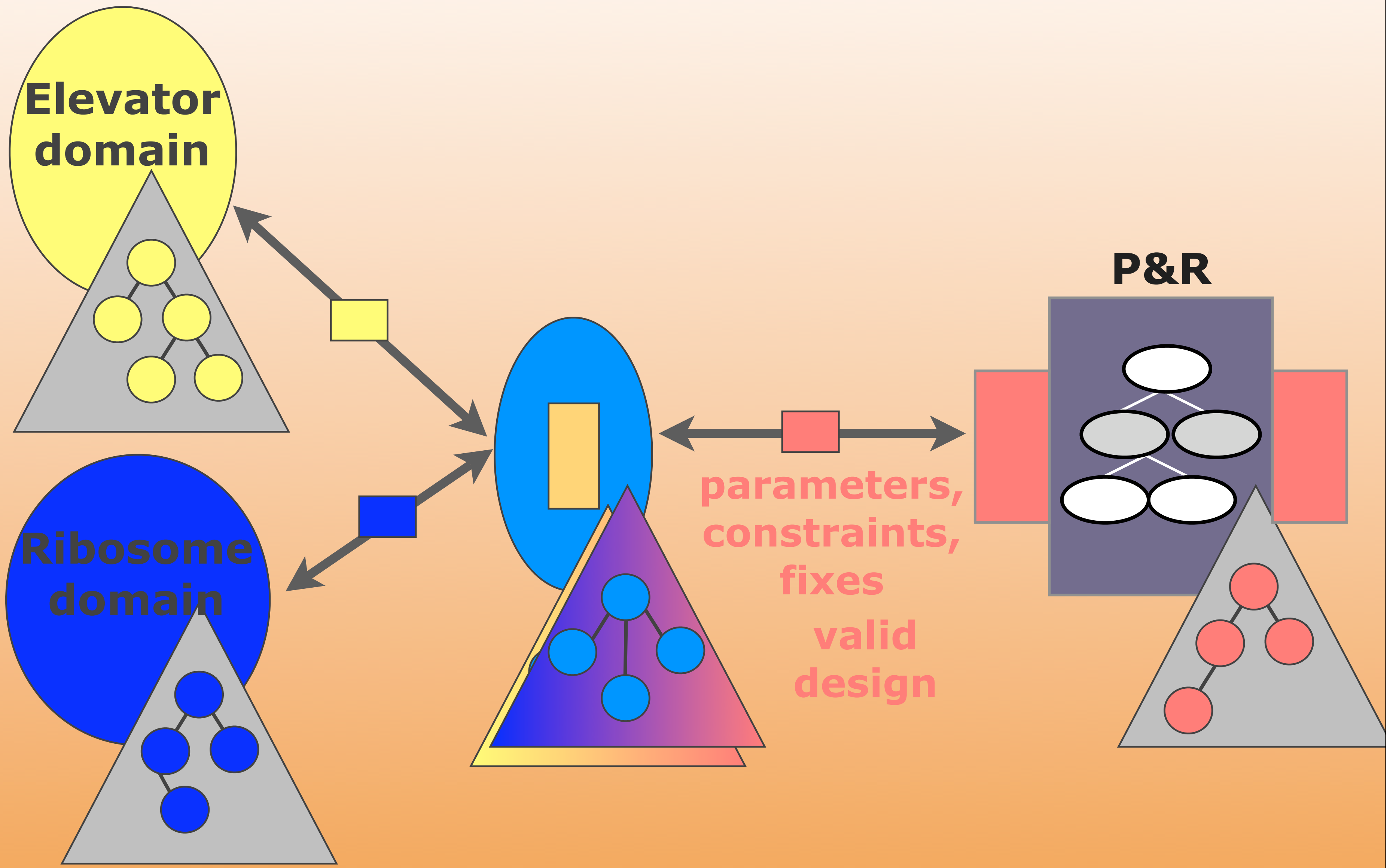
Location

94012

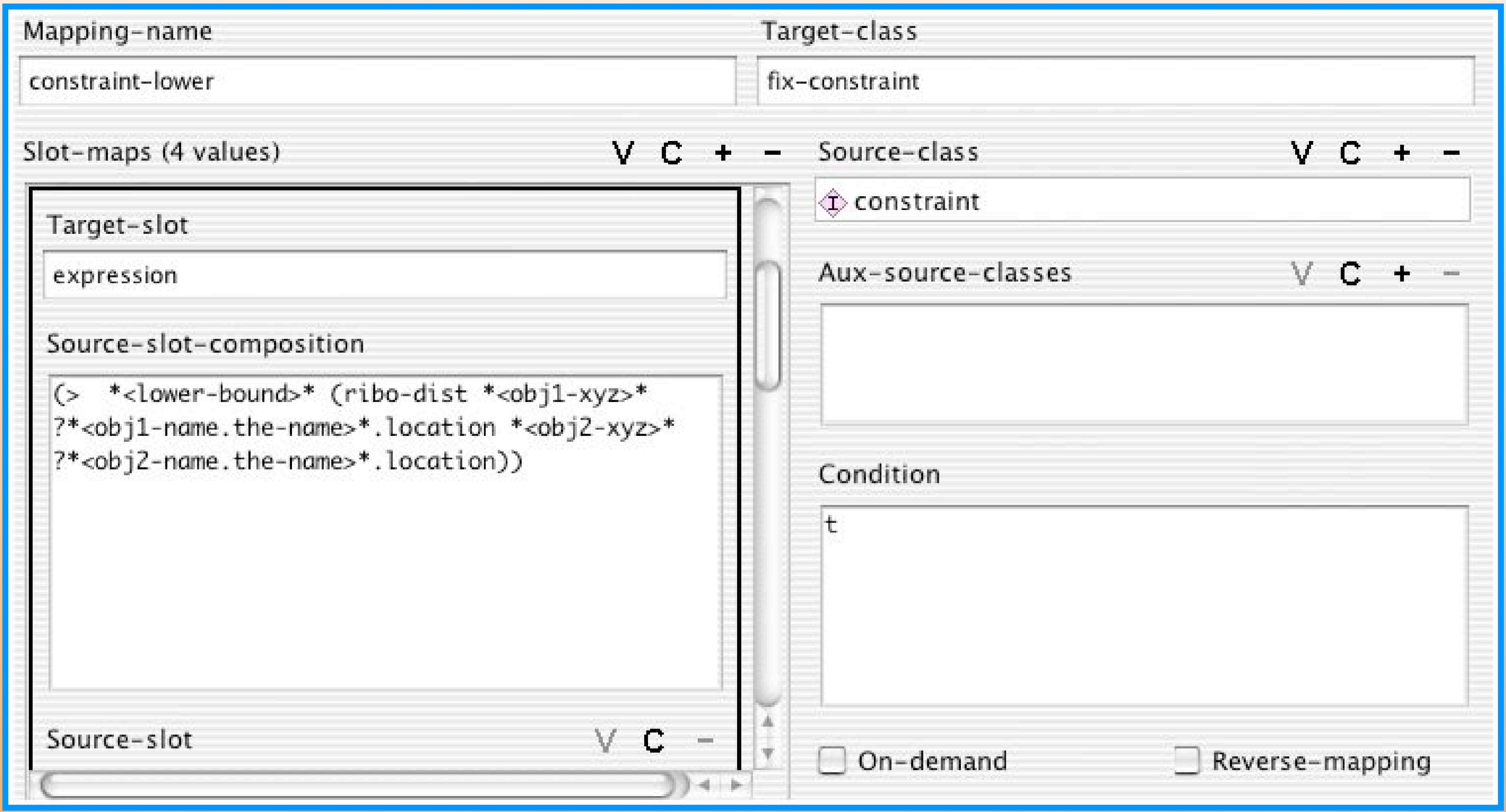
IllnessCategory

trauma

The *propose-and-revise* example



Mapping ribosome range constraints to PnR constraints



Mapping-name
constraint-lower

Target-class
fix-constraint

Slot-maps (4 values)

Target-slot
expression

Source-slot-composition

```
(>  *<lower-bound>* (ribo-dist *<obj1-xyz>*
?*<obj1-name.the-name>*.location *<obj2-xyz>*
?*<obj2-name.the-name>*.location))
```

Source-slot

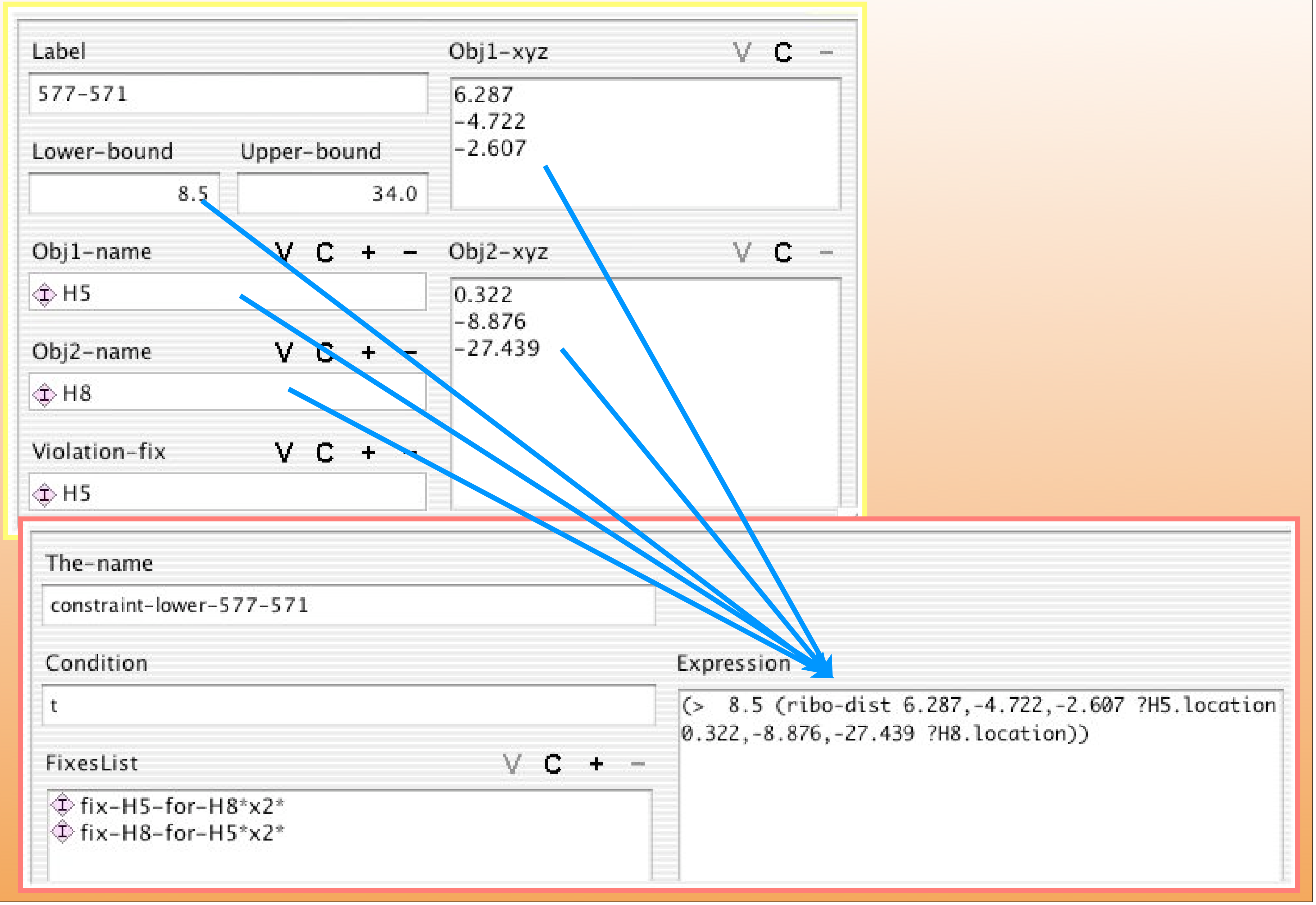
Source-class
constraint

Aux-source-classes

Condition
t

☐ On-demand
☐ Reverse-mapping

Mapping ribosome range constraints to PnR constraints



Label: 577-571

Lower-bound: 8.5

Upper-bound: 34.0

Obj1-name: H5

Obj2-name: H8

Violation-fix: H5

Obj1-xyz:
6.287
-4.722
-2.607

Obj2-xyz:
0.322
-8.876
-27.439

The-name: constraint-lower-577-571

Condition: t

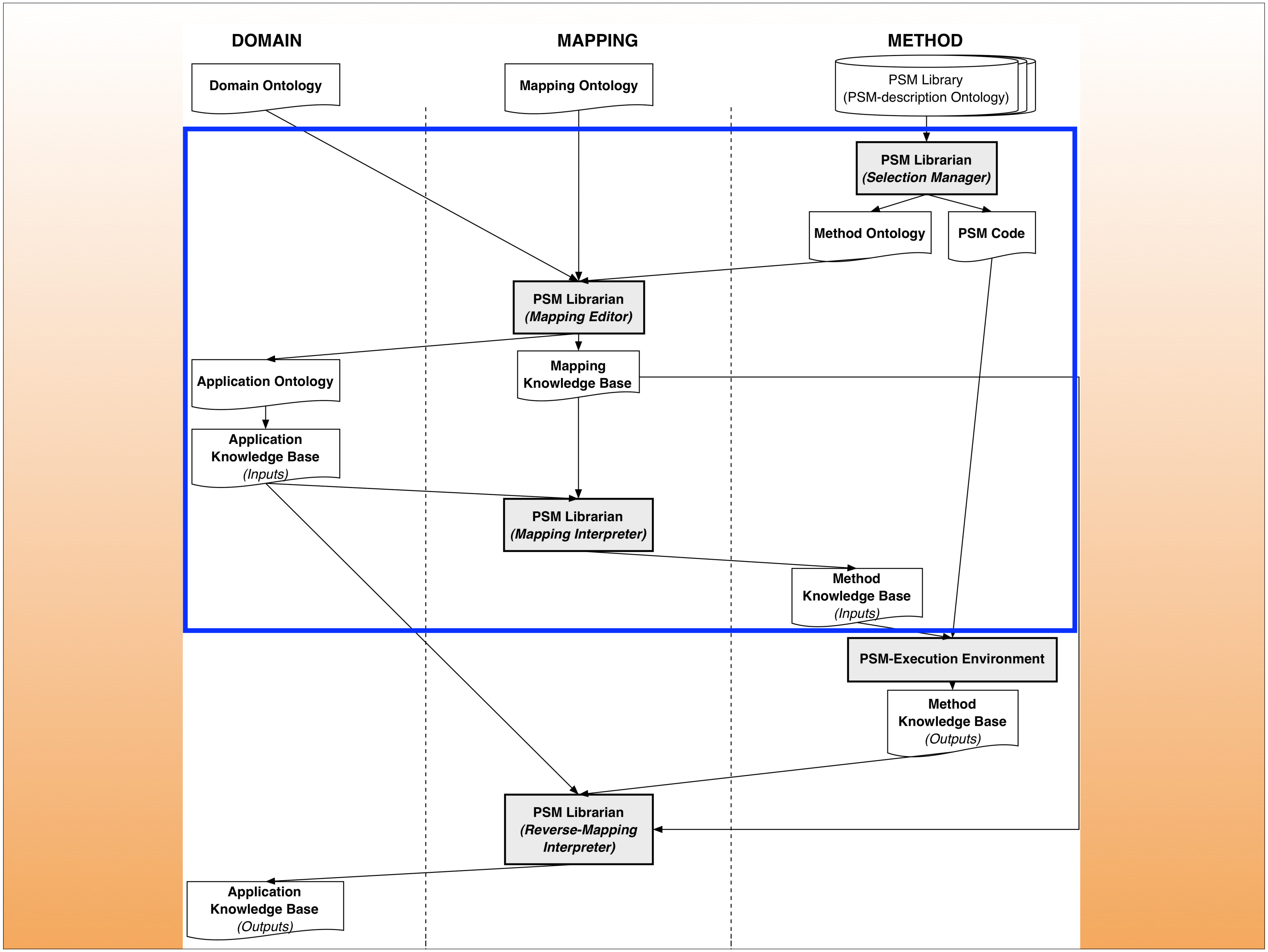
FixesList:
fix-H5-for-H8*x2*
fix-H8-for-H5*x2*

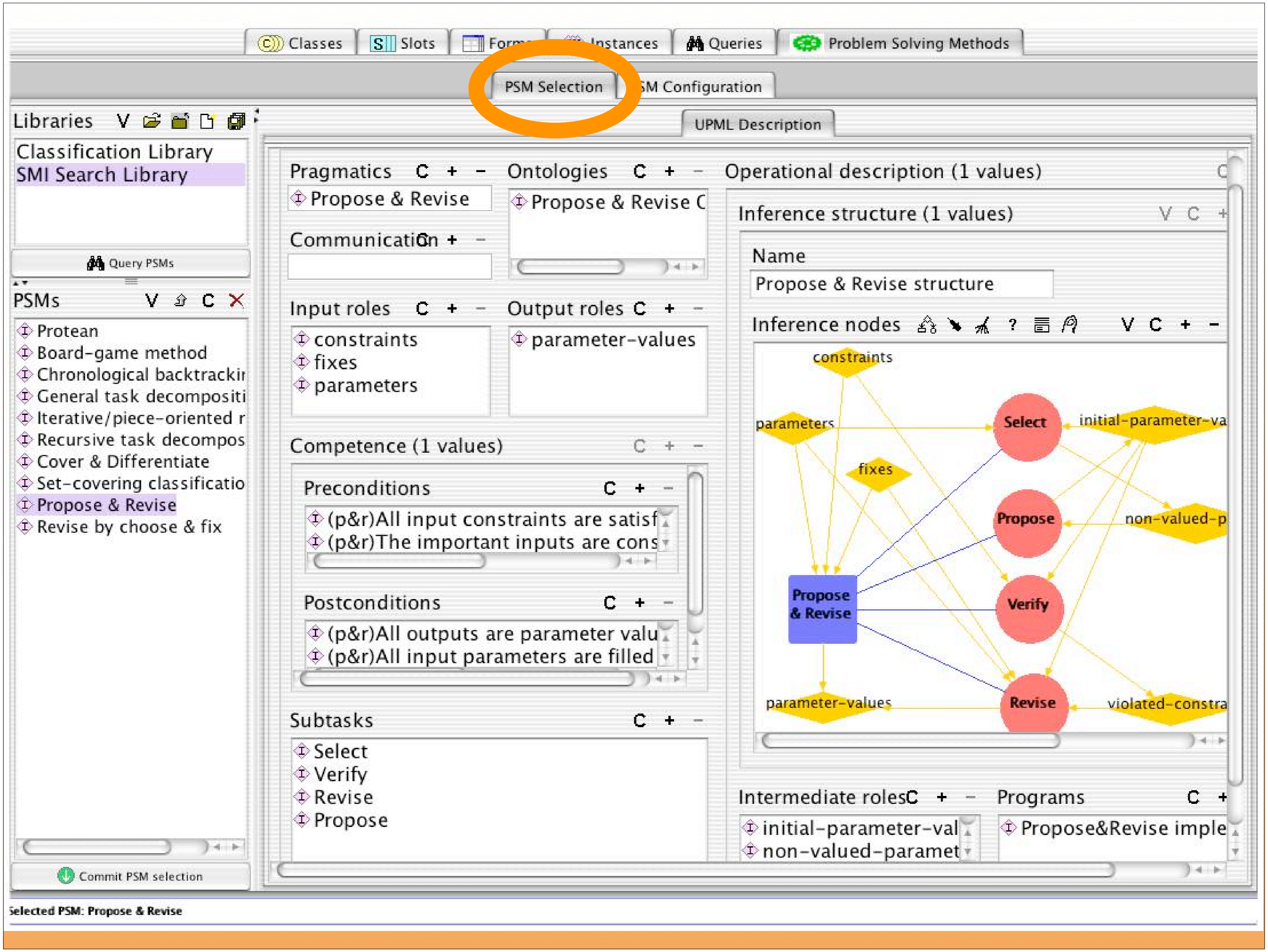
Expression:

```
(>  8.5 (ribo-dist 6.287,-4.722,-2.607 ?H5.location
0.322,-8.876,-27.439 ?H8.location))
```

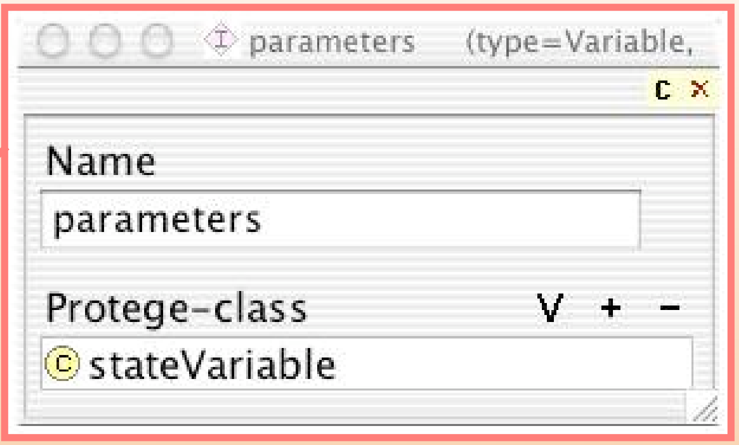
The PSM Librarian tab

- PSM selection support
  - browsing & searching of UPML libraries
  - access to all elements of a PSM’s model
- PSM configuration support
  - integrated browsing of domain, PSM I/O & mappings ontologies
  - authoring of mapping relations
  - execution of the mapping interpreter
  - inspection of resulting instances (i.e., PSM inputs)


DOMAIN
MAPPING
METHOD
Domain Ontology
Mapping Ontology
PSM Library (PSM-description Ontology)
PSM Librarian (Selection Manager)
Method Ontology
PSM Code
PSM Librarian (Mapping Editor)
Application Ontology
Mapping Knowledge Base
Application Knowledge Base (Inputs)
PSM Librarian (Mapping Interpreter)
Method Knowledge Base (Inputs)
PSM-Execution Environment
Method Knowledge Base (Outputs)
PSM Librarian (Reverse-Mapping Interpreter)
Application Knowledge Base (Outputs)

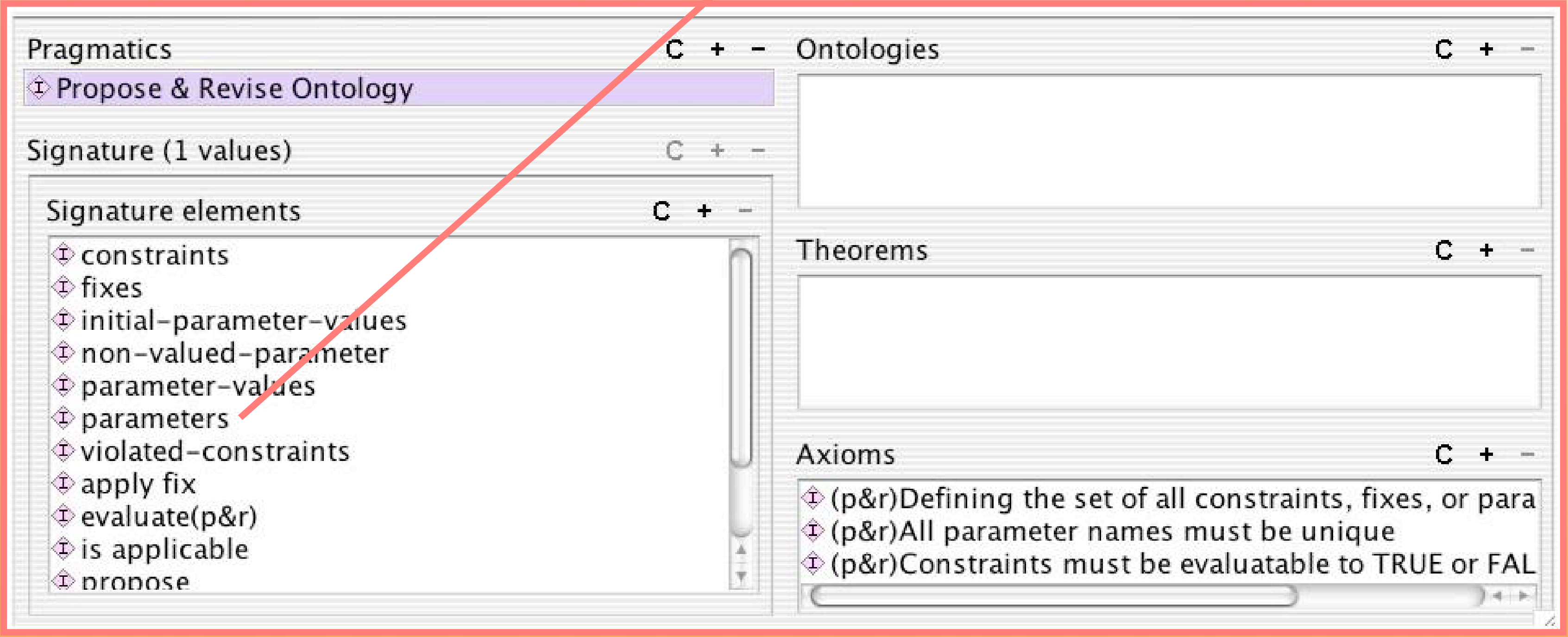

Classes
Slots
Forms
Instances
Queries
Problem Solving Methods
PSM Selection
PSM Configuration
UPML Description
Libraries V
Classification Library
SMI Search Library
Query PSMs
PSMs V C
Protean
Board-game method
Chronological backtrackir
General task decompositi
Iterative/piece-oriented r
Recursive task decompos
Cover & Differentiate
Set-covering classificatio
Propose & Revise
Revise by choose & fix
Commit PSM selection
Pragmatics C + −
Propose & Revise
Communication C + −
Ontologies C + −
Propose & Revise C
Input roles C + −
constraints
fixes
parameters
Output roles C + −
parameter-values
Competence (1 values) C + −
Preconditions C + −
(p&r)All input constraints are satisf
(p&r)The important inputs are cons
Postconditions C + −
(p&r)All outputs are parameter valu
(p&r)All input parameters are filled
Subtasks C + −
Select
Verify
Revise
Propose
Operational description (1 values)
Inference structure (1 values) V C +
Name
Propose & Revise structure
Inference nodes V C + −
constraints
parameters
fixes
Select
initial-parameter-va
Propose
non-valued-p
Propose & Revise
Verify
parameter-values
Revise
violated-constra
Intermediate roles C + −
initial-parameter-val
non-valued-paramet
Programs C +
Propose&Revise imple
Selected PSM: Propose & Revise

PSM input-output ontology


parameters (type=Variable,

Name
parameters

Protege-class
stateVariable


Pragmatics
Propose & Revise Ontology

Signature (1 values)

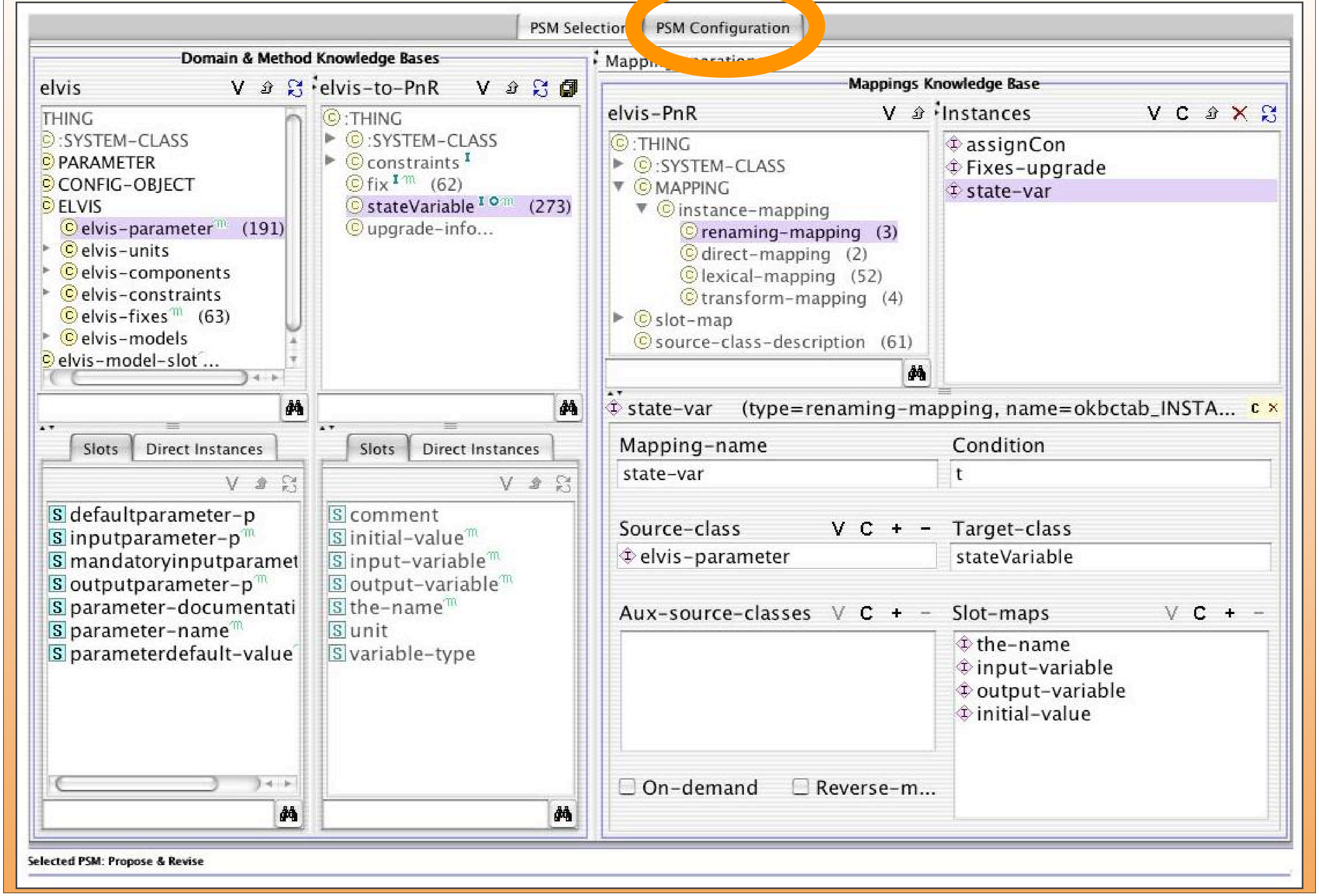
Signature elements
constraints
fixes
initial-parameter-values
non-valued-parameter
parameter-values
parameters
violated-constraints
apply fix
evaluate(p&r)
is applicable
propose

Ontologies

Theorems

Axioms
(p&r)Defining the set of all constraints, fixes, or para
(p&r)All parameter names must be unique
(p&r)Constraints must be evaluatable to TRUE or FAL

PSM configuration support



elvis: THING, :SYSTEM-CLASS, PARAMETER, CONFIG-OBJECT, ELVIS, elvis-parameter (191), elvis-units, elvis-components, elvis-constraints, elvis-fixes (63), elvis-models, elvis-model-slot…

elvis-to-PnR: :THING, :SYSTEM-CLASS, constraints, fix (62), stateVariable (273), upgrade-info…

Slots (elvis): defaultparameter-p, inputparameter-p, mandatoryinputparamet, outputparameter-p, parameter-documentati, parameter-name, parameterdefault-value

Slots (elvis-to-PnR): comment, initial-value, input-variable, output-variable, the-name, unit, variable-type

elvis-PnR: :THING, :SYSTEM-CLASS, MAPPING, instance-mapping, renaming-mapping (3), direct-mapping (2), lexical-mapping (52), transform-mapping (4), slot-map, source-class-description (61)

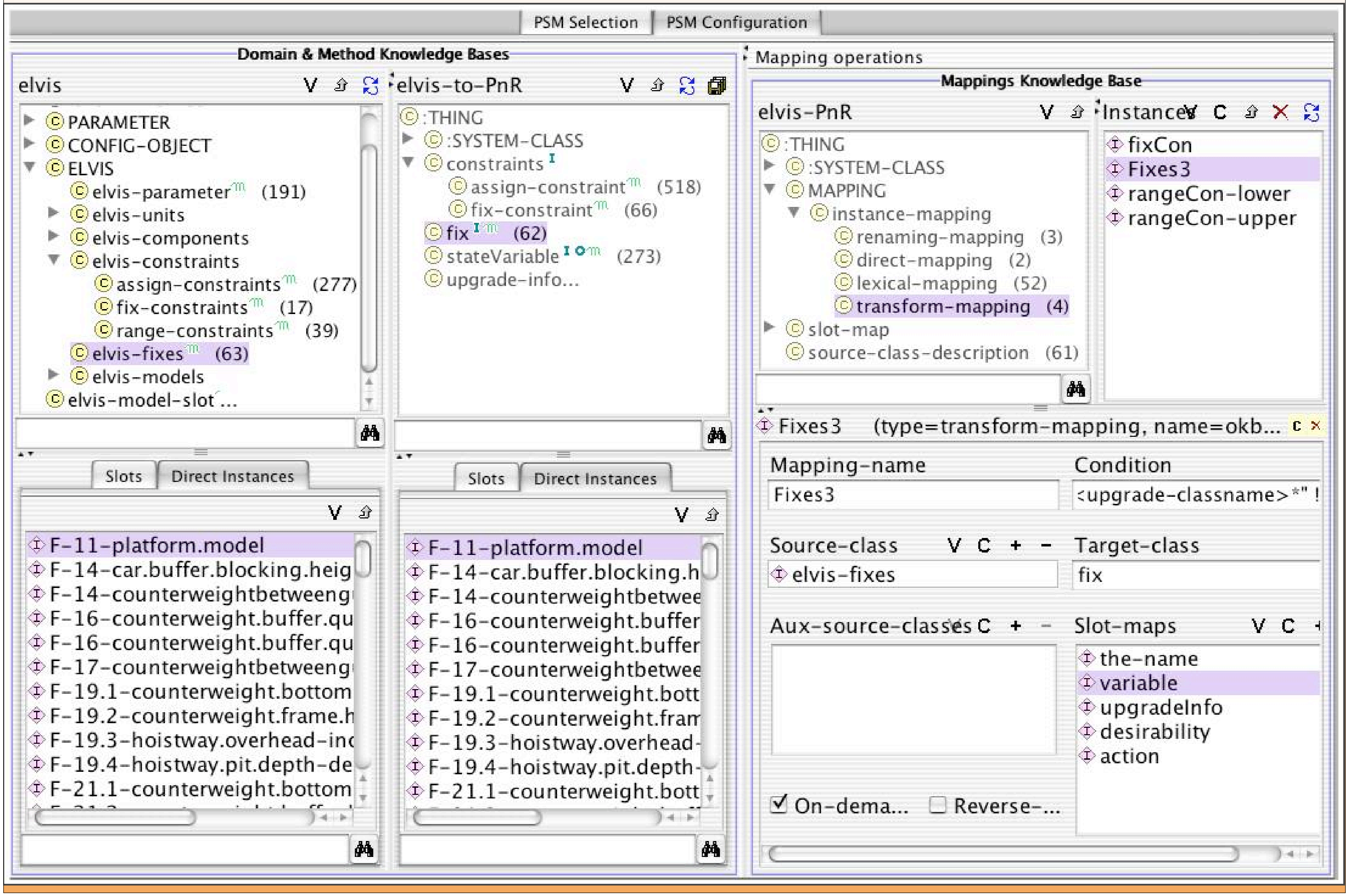
Instances: assignCon, Fixes-upgrade, state-var

state-var (type=renaming-mapping, name=okbctab_INSTA…

Mapping-name: state-var	Condition: t
Source-class: elvis-parameter	Target-class: stateVariable
Aux-source-classes:	Slot-maps: the-name, input-variable, output-variable, initial-value

☐ On-demand ☐ Reverse-m…

PSM configuration support (2)



PSM Selection | PSM Configuration

Domain & Method Knowledge Bases

elvis

- PARAMETER
- CONFIG-OBJECT
- ELVIS
  - elvis-parameter (191)
  - elvis-units
  - elvis-components
  - elvis-constraints
    - assign-constraints (277)
    - fix-constraints (17)
    - range-constraints (39)
  - elvis-fixes (63)
  - elvis-models
- elvis-model-slot ...

Slots | Direct Instances

- F-11-platform.model
- F-14-car.buffer.blocking.heig
- F-14-counterweightbetweeng
- F-16-counterweight.buffer.qu
- F-16-counterweight.buffer.qu
- F-17-counterweightbetweeng
- F-19.1-counterweight.bottom
- F-19.2-counterweight.frame.h
- F-19.3-hoistway.overhead-inc
- F-19.4-hoistway.pit.depth-de
- F-21.1-counterweight.bottom

elvis-to-PnR

- :THING
- :SYSTEM-CLASS
- constraints
  - assign-constraint (518)
  - fix-constraint (66)
- fix (62)
- stateVariable (273)
- upgrade-info...

Slots | Direct Instances

- F-11-platform.model
- F-14-car.buffer.blocking.h
- F-14-counterweightbetwee
- F-16-counterweight.buffer
- F-16-counterweight.buffer
- F-17-counterweightbetwee
- F-19.1-counterweight.bott
- F-19.2-counterweight.fram
- F-19.3-hoistway.overhead-
- F-19.4-hoistway.pit.depth-
- F-21.1-counterweight.bott

Mapping operations

Mappings Knowledge Base

elvis-PnR

- :THING
- :SYSTEM-CLASS
- MAPPING
  - instance-mapping
    - renaming-mapping (3)
    - direct-mapping (2)
    - lexical-mapping (52)
    - transform-mapping (4)
  - slot-map
  - source-class-description (61)

Instances

- fixCon
- Fixes3
- rangeCon-lower
- rangeCon-upper

Fixes3 (type=transform-mapping, name=okb...

Mapping-name: Fixes3

Condition: <upgrade-classname>*" !

Source-class: elvis-fixes

Target-class: fix

Aux-source-classes

Slot-maps:
- the-name
- variable
- upgradeInfo
- desirability
- action

☑ On-dema... ☐ Reverse-...

Concluding remarks

- Benefits of explicit, structured mappings
  - Maximize independence & reuse of components
  - Minimize adaptation of components to work together
  - Isolate connection & transformation knowledge
  - More general than domain-to-PSM mapping paradigm
  - When both conceptual and syntactic transformation are needed
- Mapping is still hard
  - Generality and reuse of mappings defined by a target component as templates for other applications?
  - Complementarity with ontology-merging approaches? (PROMPT)

- PSM Librarian tab now online
- Mapping tools soon available standalone

http://protege.stanford.edu/plugins/

crubezy@smi.stanford.edu