Assessing bidirectional model transformation tools

Alcino Cunha





Universidade do Minho

BIRS BX workshop, Banff December, 2013

Motivation

- Survey (operational) BX tools
- Focus on expressiveness
- Example driven
- Systematic exploration of the consistency design space
- Clarification of supported BX laws
- Repository of unit tests for (future) tool developers

Constraint maintainers



Basic properties of relations

$$\frac{\mathsf{R}(a,b) \quad \mathsf{R}(a,b')}{b=b'} \qquad \mathsf{R} \subseteq A \times \mathsf{lone} \ B \tag{Simple}$$

$$\frac{\mathsf{R}(a,b) \quad \mathsf{R}(a',b)}{a=a'} \qquad \mathsf{R} \subseteq A \text{ lone } \times B \tag{Injective}$$

$$\exists b.R(a,b)$$
 $R \subseteq A \times \text{some } B$ (Entire)

$$\exists a.R(a,b)$$
 $R \subseteq A \text{ some } \times B$ (SURJECTIVE)

The bestiary of relations

 $lone \hookrightarrow lone$ $set \hookrightarrow lone$ $set \times set$ (Partial Injection) (Partial Function) (Relation) $set \rightarrow one$ **Partiality** (Function) lone \rightarrow one set × some (Injection) some \hookrightarrow lone (Entire Relation) (Partial Surjection) one \rightarrow one some \rightarrow one some × some (Surjection) (\dots) (Bijection)

Non-determinism

Data domain

- Shape
 - Tree
 - Graph
- Constraints
 - None
 - Order
 - Keys
 - Other

Some laws of interest

$$\frac{\overleftarrow{\mathsf{R}}(a,b')=a'}{\mathsf{R}(a',b')}$$

$$\frac{\mathsf{R}(a,b')}{\mathsf{R}(a,b')=a}$$

$$\frac{\exists a'.\mathsf{R}(a',b')}{\overline{\mathsf{R}}(a,b')\downarrow}$$

$$\frac{\overleftarrow{\mathsf{R}}(a,b') = a' \quad \mathsf{R}(a',b)}{\overleftarrow{\mathsf{R}}(a,b) = a'}$$

For curators

- Propose simple but illustrative examples:
 - Specify meta-models + consistency relation + unit tests.
 - UML diagrams + (hopefully) non-ambiguous natural language?
- Propose and formalize laws to be evaluated.
- Synthesize results by example (tool vs law).
- Synthesize global results (tool vs consistency vs domain).
- Publish results in a web-site (BX Wiki?).

For tool developers

- For each example:
 - Show how meta-models + consistency relation can be encoded.
 - Describe how unit tests can be implemented:
 - Show how to encode models.
 - Describe tool configuration.
 - Report results.
- Ideally, package tool + examples in a virtual machine.

Bijection #1 (one \rightarrow one), Sets

Person name : String Employee name : String

Every person is an employee and vice-versa

Ø

:Person name = "A"

:Person name = "A"

:Person name = "A" :Person name = "A"

:Person name = "B" Ø

Ø

:Employee

:Employee name = "B" :Employee name = "B"



Bijection #2 (one \rightarrow one), Sets with keys

Person name: String { id }

Employee name : String { id }

Ø

Ø

:Employee

Every person is an employee and vice-versa

Ø

:Person name = "A"

:Person name = "A"

:Person name = "A" :Person name = "B" name = "B"

:Employee :Employee name = "C"



Surjection #1 (some \rightarrow one), Sets with keys

Person

name : String { id } age : Integer

Employee name : String { id }

Every person is an employee and vice-versa

Ø

:Person

name = "A" age = 30

:Person name = "A" age = 30 :Person name = "B" age = 40

:Person name = "A" age = 30 :Person name = "B" age = 40 Ø

:Employee name = "A"

Ø

:Employee :Employee name = "C"

Surjection #2 (some \rightarrow one), Sets with order

Person
names : String [1..*] { ordered }

Employee name : String

Every person is an employee with its first name and vice-versa

Ø

:Person names : { "A", "A", "B" }

:Person names : { "A", "A", "B" }

:Person names : { "A", "A", "B" } :Person names : { "C" }

:Person names : { "C" } Ø

Ø

:Employee name = "A"

:Employee name = "A" :Employee name = "B"

Injection #1 (lone \rightarrow one), Sets with keys

Person name : String { id }

Employee name : String

Every person is an employee and vice-versa

Ø

٦

:Person name = "A"

:Person name = "A"

:Person name = "A"

:Person name = "B"

:Person name = "B" Ø

Ø

:Employee

name = "B"

:Employee name = "A"

> :Employee name = "B"

Function #1 (set \rightarrow one), Sets with keys

Person

name : String { id } age : Integer

Employee name : String

Every person is an employee and vice-versa

Ø

:Person

name = "A" age = 30

:Person name = "A" age = 30 :Person name = "B" age = 40

:Person name = "A" age = 30 :Person name = "B" age = 40 :Employee

Ø

Ø

:Employee :Employe

:Employee :Employee name = "B"

Surjective entire relation #1 (some \times some), Sets

Person name : String Employee name : String

Any pair of models is consistent

Ø

Ø

:Person name = "A"

Ø

:Person name = "A" :Person name = "A" :Employee name = "A"

:Person name = "A" :Person name = "B" :Employee name = "B" :Employee name = "C"

Surjective entire relation #2 (some \times some), Sets

Person name : String Employee name : String

There are persons iff there are employees

Ø

:Person name = "A"

:Person name = "A"

:Person name = "A" :Person name = "A"

:Person name = "B" Ø

Ø

ی

:Employee name = "A"

:Employee :Employee name = "C"

We want you!



Acknowledgements

- Perdita Stevens (inspiration, examples)
- Soichiro Hidaka (GRoundTram)
- Nuno Macedo (Echo)
- Tiago Jorge (web-site)