

# TMCL

- a very experimental  
version

Dmitry Bogachev  
Montreal, 2005, 29th July

# What is new

- Modified schema part based on recommendations from meeting in Amsterdam
- More precise notation: using InfoSet now
- Better integration of rule and schema parts
- Some work on better formalism for schema interpretation

# Topic Map Schema

TopicMapSchema:

[schemaID]	# TopicIdentification? - topic map schema identifier
[baseLocator]	# IRI? - a reference to the location where TopicMapSchema is stored
[schemaName]	# TopicName* - a set of topic name items representing name of a schema
[include]	# TopicMapSchema* - a set of TopicMapSchema items: list of included schemas
[topicSchema]	# TopicSchema* - a set of topic schemas
[associationSchema]	# AssociationSchema* - a set of association schemas
[ruleItem]	# RuleItem* - set of rules

# Topic Identification

Topic Identification is used to identify exactly 1 topic.

TopicIdentification:

[sourceLocator] # IRI\*

[subjectIdentifier] # IRI\*

[subjectLocator] # IRI\*

# Set of topic identification constructs

TopicSet:

[topicIdentification]

# TopicIdentification\*

# Scope pattern

ScopePattern:

[simpleTopicExpression]

# TopicIdentificaiton\*

[orTopicExpression]

# OrTopicExpression\*

[typeTopicExpression]

# TopicIdentificaiton\*

# Or Topic Expression

OrTopicExpression:

[orTopics] # TopicIdentificaiton \*

# Scope Pattern: Examples

- fr
- fr or en
- %language%
- (Dmitry or Graham), %language%

# Assert - embedding rules

Assert:

- [parent] # TopicSchema or .... - reference to parent
- [type] # TopicIdentification?, reference to type of the assertion, default type is "Conflict"
- [test] # string?, TMQL expression which can include \$self variable for current context
- [every] # string?, shortcut for "every \$X in \$self satisfies P(\$X)"
- [some] # string?, shortcut for "some \$X in \$self satisfies P(\$X)"
- [message] # string?, which can include \$self variable for current context

# Assert: comments

Each schema creates a path expression which can be used as a selector for specifying constraints

Examples:

test:        count(\$self)>5

every:       contains(\$self,'J')

some:        contains(\$self,'John')

# Topic Schema

[parent]	# TopicMapSchema - reference to topic map schema
[typeSelector]	# TopicIdentification
[subjectLocatorSchema]	# SubjectLocatorSchema*
[subjectIdentifierSchema]	# SubjectIndicatorSchema*
[topicNameSchema]	# TopicNameSchema*
[occurrenceSchema]	# ExternalOccurrenceSchema*
[playRoleSchema]	# PlayRoleSchema *
[oneOf]	# TopicSet? - list of topics
[assert]	# Assert* - list of assertions

# Topic Schema: TODO

- super-type/sub-type constraints
- reification constraints
- disjoint constraints

# Subject Identifier Schema

SubjectIdentifierSchema:

[parent] # TopicSchema - reference to topic schema

[cardMin] # nonNegativeInteger?

[cardMax] # nonNegativeInteger?

[match] # Regular Expression\* , \* is important for merging, has AND meaning

[assert] # Assert\* - list of assertions

DB: can we lose 'match'?

Use assert if needed

# Subject Locator Schema

SubjectLocatorSchema:

[parent]           # TopicSchema - reference to topic schema

[cardMin]           # nonNegativeInteger?

[cardMax]           # nonNegativeInteger?

[match]             # Regular Expression\*

[assert]            # Assert\* - list of assertions

# Topic Name Schema

TopicNameSchema:

[parent]	# TopicSchema - reference to topic schema
[typeSelector]	# TopicIdentification
[scopeSelector]	# ScopePattern?
[cardMin]	# nonNegativeInteger?
[cardMax]	# nonNegativeInteger?
[match]	# Regular Expression*
[variantSchema]	# VariantSchema*
[assert]	# Assert* - list of assertions

# Variant Schema

VariantSchema:

[parent]	# BaseNameSchema - reference to parent base name schema
[scopeSelector]	# ScopePattern?
[dataTypeSelector]	# xsd and custom xml schemas?
[cardMin]	# nonNegativeInteger?
[cardMax]	# nonNegativeInteger?
[match]	# Regular Expression*
[assert]	# Assert* - list of assertions

# Occurrence Schema

OccurrenceSchema:

[parent]	# TopicSchema - reference to parent topic schema
[typeSelector]	# TopicIdentification
[scopeSelector]	# ScopePattern?
[cardMin]	# nonNegativeInteger?
[cardMax]	# nonNegativeInteger?
[dataType]	# xsd and custom schemas?, # (during merging should be the same)
[oneOf]	# String*
[match]	# Regular Expression*, (* for merging)
[assert]	# Assert* - list of assertions

# Play Role Schema

PlayRoleSchema:

[parent]	# TopicSchema - reference to parent topic schema
[associationTypeSelector]	# TopicIdentification
[roleTypeSelector]	# TopicIdentification
[scopeSelector]	# ScopePattern?
[cardMin]	# nonNegativeInteger?
[cardMax]	# nonNegativeInteger?
[otherRoleSchema]	# RoleSchema*
[assert]	# Assert* - list of assertions

# Role Schema

RoleSchema:

[parent]	#TopicSchema or AssociationSchema - # reference to parent schema
[roleTypeSelector]	# TopicIdentification
[cardMin]	# nonNegativeInteger?
[cardMax]	# nonNegativeInteger?
[allPlayersFrom]	# TopicSet* - list of lists of types, # each TopicSet has "OR" interpretation, external list - "AND"
[oneOf]	#TopicSet? - list of specific topics
[assert]	# Assert* - list of assertions

# Association Schema

AssociationSchema:

[parent] # TopicMapSchema - reference to parent topic map schema

[typeSelector] # TopicIdentification

[roleSchema] # RoleSchema+

[assert] # Assert\* - list of assertions

# Occurrence Schema

- ???

# RuleItem

RuleItem:

[parent] # TopicMapSchema - reference to parent topic map schema

[ruleName] # TopicName\* - a set of topic name items  
# representing name of a rule

[selector] # string?, TMQL expression which generates list of tuples

[assertItem] # AssertItem\*

# AssertItem

AssertItem:

- [type] # TopicIdentification?, reference to type of the assertion,  
# default type is "Conflict"
- [test] # string?, TMQL expression, can include variable \$self for  
# addressing list of tuples
- [every] # string?, shortcut for "every \$1, \$2... in 'tuple list' satisfies P(\$1,\$2,...)"
- [some] # string?, shortcut for "some \$1, \$2... in 'tuple list' satisfies P(\$1,\$2,...)"
- [message] # string?, which can include \$self or \$1,\$2, ... variables for current context

# Rule example

RuleItem:

[ruleName]:

[value]="Good Number Of Subtypes"

[selector]="""

for \$X in subtype-supertype(\_ : subtype, \$X : supertype)

let \$BNN:=select count(\$Y) from

    subtype-supertype(\$Y : subtype, \$X : supertype)

return (\$X,\$BNN)

""

[assert]:

[type]:

[subjectIdentifier]="http://...#Notification"

[every]="\$2>7"

[message]="Warning: Type {\$I/bn} has more than 7 subtypes"

# TMQL question

- do we have access to list/set of tuples ?
- $\$self := ((a1,b1,c1) (a2,b2,c3) \dots)$
- $count(\$self)$
- $for \$x in \$self \dots$
- $every \$x in \$self \dots$
- $some \$x in \$self \dots$

# TMCL-model is back

Infoset



Infoset



Infoset



# TMCL Model

- Defines a set of basic constraints which we think are important based on TMCL requirements (mostly smart topic viewing/editing)
- Simple interpretation
- Simple merging
- If something cannot be expressed with model, for sure it can be expressed with rule

# SubjectLocatorCardMinConstraint InfoSet item

SubjectLocatorCardMinConstraint:

[typeSelector] # TopicIdentification

[cardMin] # NonNegativeInteger

# SubjectLocatorCardMinConstraint: mapping from schema

TopicSchema:	# = \$TSch
[typeSelector]	# = \$TypeSel
[subjectLocatorSchema]:	
[cardMin]	# = \$N

```
for $TSch,$TypeSel, $SLSch, $N in
  instanceOf($TSch, TopicSchema)
  and typeSelector($TSch,$TypeSel)
  and subjectLocatorSchema($TS,$SLS)
  and cardMin($SLS,$N)
```

```
infer $C
  instanceOf($C, SubjectLocatorCardMinConstraint),
  typeSelector($C,$T),
  cardMin($Cnf,$N)
```

# Comments

- It is controversial: we introduce inference
- But it is a nice way to describe transformation of one assertion model expressed as an info set to another assertion model expressed as an info set
- It looks like TMQL a little bit, but that's just for convenience

# SubjectLocatorCardMinConstraint: interpretation

```
for $Ex,$C, $T, $N in
  instanceOf($Ex,$T)
  and instanceOf($C,SubjectLocatorCardMinConstraint)
  and typeSelector($C,$T)
  and cardMin($C,$N)
  and not ExistsAtLeast $N $Y SuchAs subjectLocator($Ex,$Y)

let
  $Message:="SubjectLocatorCardMinConflict: Topic: {$Ex/bn}, Topic Type={$T/bn}, cardMin={$N}"

infer $Cnf
  instanceOf($Cnf,SubjectLocatorCardMinConflict),
  typeSelector($Cnf,$T),
  instanceTopic($Cnf,$Ex),
  cardMin($Cnf,$N),
  message($Cnf,$Message)
```

# Comments

- We define what kind of conflicts can be inferred based on constraints and existing topic map which we apply constraints to
- Conflicts are represented as InfoSet

# Extended TMCL Schema: for discussion

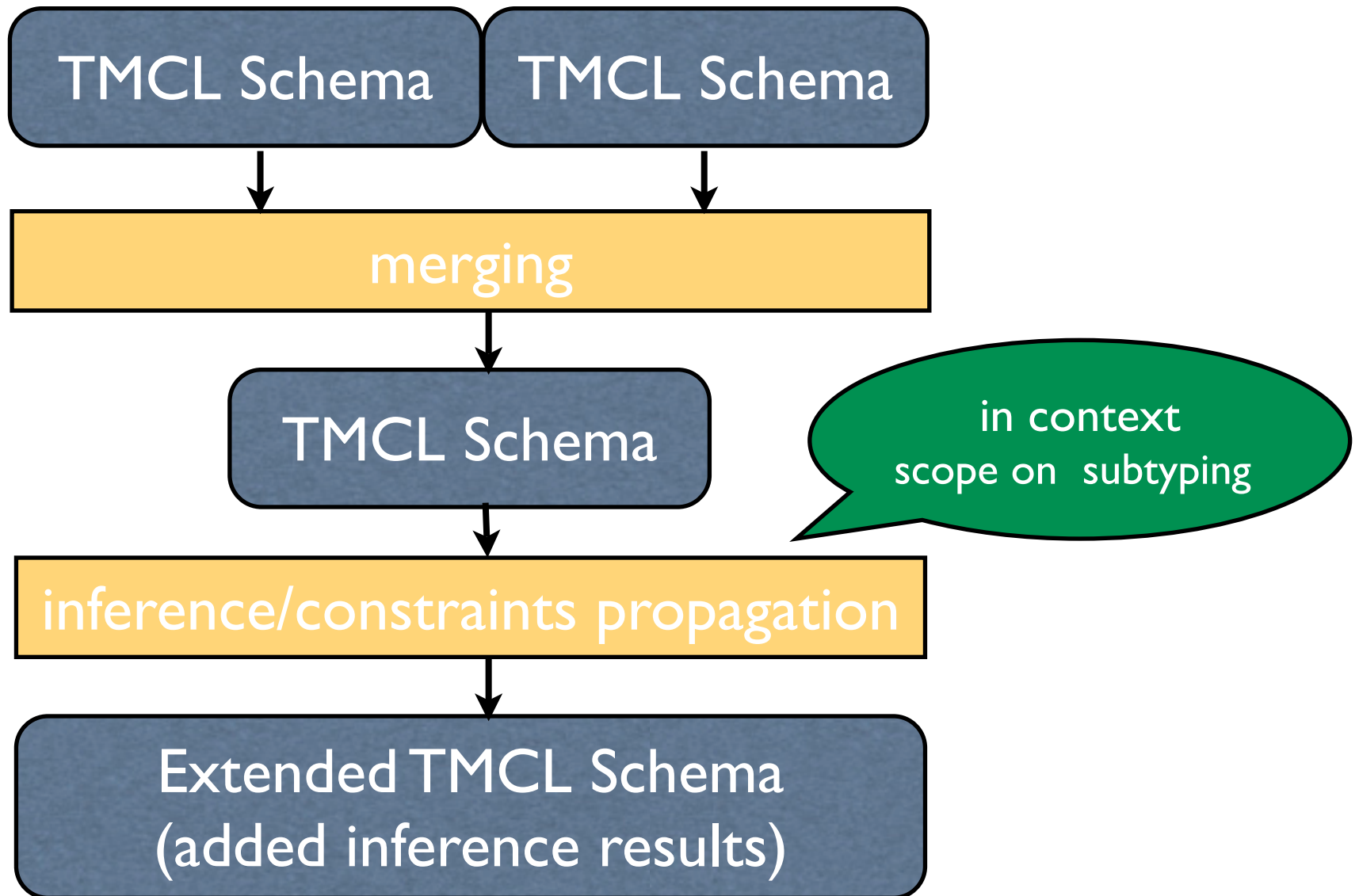
Additional properties on schemas:

- isUserProvided
- isInferred

and additional constraint properties:

- cardMinUserProvided
- cardMinInferred
- cardMin (summary)

# Extended TMCL Schema



# Example of extended Schema

TopicSchema:

[isUserProvided]=true

[isInferred]=true

[typeSelector]:

    [subjectIdentifier]="http://.../#composer"

[occurrenceSchema]:

    [typeSelector]:

        [subjectIdentifier]="http://.../#webPage"

    [isUserProvided]=true

    [isInferred]=true

    [cardMinInferred]=0

    [cardMinUserProvided]=1

    [cardMin]=1

# Summary

- Response to NBs
- New draft
- More intensity in research :)