

TMAssert - Introduction

TMAssert – Presentation for the TMRM workshop in Montreal-2004

- Basic assertion model (BAM)
- BAM+S (scope)
- TMDM mapping
- BAM+I (inference) TODO
- BAM + CD (constructor/destructor) TODO
- BAM +C (constraints) TODO
- BAM +Q (query) TODO

@ 2004 Dmitry Bogachev

Basic Assertion Model (BAM)

Assertion Map:

- Finite set of GUIDs (globally unique identifiers)
- Finite set of assertions:
(GUID0, ARG1,...ARGN) as GUID
ARGi is GUID or literal

- Operations:

Add GUID

Delete GUID

Add assertion

Delete assertion

Merge two GUIDs into one GUID

Add other Assertion Map

BAM - comments

- (GUID0, ARG1,...ARGN) - has informal interpretation as relationship of type GUID0 between GUIDs and literals
- Each assertion has own GUID
- It is possible to make assertions about assertions
- Merging two GUIDs into new GUID replace old GUIDS in Assertion Map
- Assertions are merged only if they have same meta assertions

BAM+S (scope)

Let's introduce subset of GUIDS in Assertion Map

– scope assertion types

Let's introduce set of contexts SC

Let's introduce Context Evaluation Operator - @

<Assertion Map with scopes> @ <context> =>

<Basic Assertion Map>

BAM+S comments

- This system allows to represent assertions which are true in specific context.
- BAM+S splits all assertions about assertions into two subsets:
 - first subset can change truth of an assertion
 - second subset cannot change truth of an assertion
- when we apply context operator additional merging of assertions can happen because assertions become “context free”

TMDM and BAM+S

General ideas:

- TMDM is a different “language”
- It assumes additional “ontological commitments”
- We define necessary constructs in BAM+S to support TMDM commitments
- We define translation rules from TMDM to BAM+S (with commitments)
- Translation rules record results of translation
- These results can influence further translation

Translation of TMDM associations

- TMDM associations are very different from BAM assertions
- BAM assertions do not have named roles
- Of course, no multiple players of the same role

We allow to use assertions such as

(nthArgRole, <assertion type>, <role type>, <position>)

Let's call it a “signature” of an assertion type

Pseudo code:

- For each association in TMDM check if there is an assertion type with the same “signature” as association
- If not - create a new assertion type, connect it with TMDM association type with assertion
(implementsTMDM, <association type>, <assertion type>)
- If yes – use existing assertion type to translate association
- If there is a role with multiple players introduce a GUID to represent an implicit group of role players. Connect each player with this group using (member, <role player>, <implicit group>) assertion

TMDM Example

authors(book_1 : book, author_1 : author, author_2: author)

authors(opera_1 : opus, author_3: author)

=>

(implementsTMDM authors, authors_1)

(nthArgRole, authors_1,book,1)

(nthArgRole, authors_1,author,2)

(authors_1, book_1, authors_group_1)

(instance_of, authors_group_1, implicit_group)

(member, author_1, authors_group_1)

(member, author_2, authors_group_1)

(implementsTMDM authors, authors_2)

(nthArgRole, authors_2,opus,1)

(nthArgRole, authors_2,author,2)

(authors_2, opera_1, author_3)

TMDM Translation comments

- If TMDM association represents simple relation there is one to one correspondence between TMDM association and BAM assertion.
- If TMDM associations use different signatures for the same association type these associations are mapped to several different BAM assertion types
- If TMDM association uses multiple players of the same role it is mapped to an assertion with an additional GUID for an implicit group

As a result:

- Nicely designed TMDM associations are mapped to simple BAM assertions
- Badly designed TMDM associations are mapped to complicated BAM assertion structures