WP3 – Triple Space Interaction

Lyndon J B Nixon

TripCom WP3
Innsbruck Meeting, 8 January 2008

Meeting agenda

- 1. WP3 Status
- 2. Task 3.5 – Implementation of semantic matching in Triple Space
  - Query Typing (20 min, Berlin)
  - Query Evaluation (20 min, Profium)
  - Inconsistent and Incomplete Reasoning (10 min, LFUI)
  - Reasoner benchmarking (10 min, Ontotext)

Sixth Framework Programme Priority 2
Information Society Technologies (IST)
Specific Targeted Research Project

WP3 Status

- In this task (by end March 08) we implement Triple Space querying
- SPARQL based query answering is already in the Triple Store Adapter (OWLIM)!
- The Query Processor should provide querying support
  - On queries on a local space, and
  - On queries to be distributed across the Triple Space
- Foreseen functionalities
  - Query Analysis (provide cost functions on subqueries)
  - Query Engine Selection (provide best reasoning facilities)
  - Query Approximation (incompleteness/inconsistency)
  - Query Distribution (best-case query answering)

Descriptive Typing of SPARQL Queries

Lyndon J B Nixon

TripCom WP3
Innsbruck Meeting, 8 January 2008









Motivation & Introduction (I)

Some essential issues affecting TS Query Processing

- Core of TS Query Processor based on SPARQL
- The TS System is distributed
- Later: Semantic similarity relation for data to support semantic clustering

Idea

- Categorizing queries (i.e. query plans) w.r.t. their **evaluation costs** to support local / distributed query evaluation optimization
- Categorizing queries w.r.t. to the **semantic similarity** of the data read/returned by the query processor to use the semantic clustering property for query evaluation optimization

Motivation & Introduction (II)

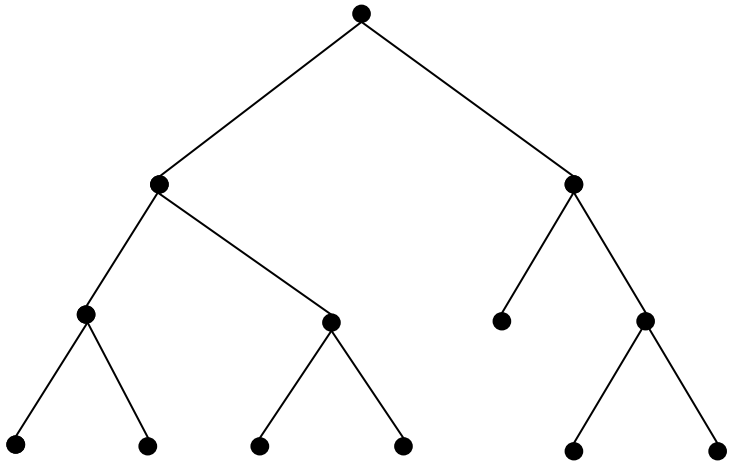
Descriptive Typing Approach

- **Core Principle**: Categorization of queries by approximating their semantics under a certain criteria (e.g. costs, similarity)
- SPARQL Queries given as algebraic trees or tree terms based on abstract SPARQL syntax and **SPARQL Graph Pattern Algebra** of Jerez et al
- **Type** (=distinct set of queries) defined by a **Type Schema** (analog to XML-Documens and XML Schemas)

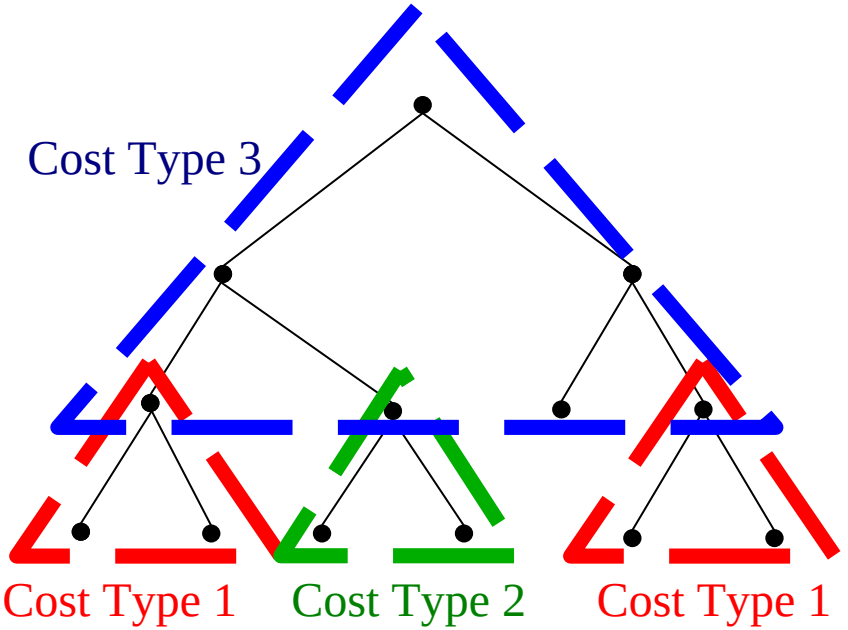
Typing w.r.t. Evaluation Costs (I)

Basic Idea

query tree



cost typed query tree



Typing w.r.t. Evaluation Costs (II)

First Version

- Algebra-oriented cost model: Complexity estimation based on the complexities for operations in SPARQL Graph Patterns
  - ► recursive cost function and solution set cardinality function for types resp. their trees
- Algebra Isomorphism: Evaluation of the operations in SPARQL Graph Patterns reducable to the evaluation of the corresponding Relational Algebra operations
  - ► complexities of best-practice algorithms for the Relational Algebra operations usable

Typing w.r.t. Evaluation Costs (III)

(Possible) further improvements & features

- consideration of ontologies (e.g. cardinality limits) for refinement of the cost function
- [consideration of optimization techniques (rewriting, normal forms, indexes)]
- adaptive strategy to soften exponentially growing estimation error
- For cost function application (?): Usage of statistical cardinality values (i.e. cardinalities of property instances) instead of actual cardinalities of the relevant solution sets as input

Research Works

- Descriptive Typing for SPARQL queries based on schemas
- Schema-driven cost model for SPARQL queries
- Effective SPARQL query segmentation for distributed query evaluation in TS
- Hypergraph-Automaton for graph-structured data

Thank You!

Query Processor

Janne Saarela, Profium

TripCom meeting in Innsbruck

January 8th, 2008

Query Processor
Functional Description (Black Box View)

- Given a query expression SPARQL-- or SPARQL++ …
  - The Query Processor evaluates the query against the tuples known by a dynamically selected query processor (ground and inferred) …
  - … and returns the resulting tuples to the client (potentially in the format required by the client by post-processing the query)
  - The functionality is probably invoked by not other internal components but by external applications

Query Processor Component Interface

- Provided
  - QueryExpression
    - Query expression encoded as a string
  - QueryResult
    - Query evaluation result
- Required
  - ErrorResult
    - Error information about query evaluation
- Required code internally
  - Query Engine implementations - planning integration of 3 existing engines
    - Junit tests for 100% method coverage
    - Scalability test to help decide deployment environment characteristics
  - Query Typing implementation
    - Junit tests for 100% method coverage

Query Processor
Current Status

- Initial design done (Processor, Analyzer, Executor)
- Interfaces developed and committed to svn repository with mock junit tests

- Benchmarking environment has been set up at Profium:
  - 2 * HP Proliant DL380 G5 - 2 x Intel(R) Xeon(R) CPU 5160 @ 3.0GHz with 4 GB main memory both running RHEL V4
  - Ready to perform component level performance

Inconsistent Reasoning for TripCom

WP 3 Meeting

Omair Shafiq
Lyndon J B Nixon

TripCom WP3
Innsbruck Meeting, 8 January 2008









Inconsistency in Ontologies

- Inconsistency can occur in many different ways
- Ambiguous representation of rules
  - if new statements are added to ontology and the new statements are not consistent with the old ones
- Using terminologies that have dual or multiple meanings
- Migrating ontologies from one data source to another data source
- Creating ontology by multiple sources

Example 1/3…

- Birds are animals.
- Birds are flying animals
- Eagle is bird
- Penguin is bird

- One can conclude that Penguins can fly
- Which is actually not true

Example 2/3…

- zoo:Bird rdf:type rdfs:Class.
- zoo:Animals rdf:type rdfs:Class.
- zoo:FlyingAnimals rdf:type rdfs:Class.
- zoo:Eagle rdf:type rdfs:Class.
- zoo:Penguin rdf:type rdfs:Class.

- **zoo:Bird rdfs:subClassOf zoo:Animals.**

- **zoo:canfly rdf:type rdf:Property**
- **rdfs:domain zoo:Bird;**
- **rdfs:range zoo:FlyingAnimals.**

- **zoo:Eagle rdfs:subClassOf zoo:Bird**
- **zoo:Penguin rdfs:subClassOf zoo:Bird**

Sixth Framework Programme Priority 2
Information Society Technologies (IST)
Specific Targeted Research Project

Example 3/3…

- Sample SPARQL query below may give the result which may be correct for the given RDF statements but is logically incorrect in real world.

- *SPARQL query:*
- *SELECT ?zoo:birds WHERE (?x rdf:type zoo:FlyingAnimals)*

- Result if the query will be:
  - Eagle
  - Penguin

PION framework …

- A framework for reasoning with inconsistent ontologies
- Based on the approach to simply avoid the inconsistency and to apply a non-standard reasoning method to obtain meaningful answers
- Suitable in the web setting.
- The selection function is based on finding the syntactic connections between axioms in order to decide which axioms are relevant to a query.
- The relevance requirement is decreased gradually during reasoning in order to obtain a increasing subset of the axioms until it has selected a subset which is small enough to avoid the inconsistency, but large enough to answer the query
- 2nd selection function takes into account that it is dealing with ontologies and uses the concept hierarchy for the selection of related axioms.

Sixth Framework Programme Priority 2
Information Society Technologies (IST)
Specific Targeted Research Project

Rating the correctness of answers

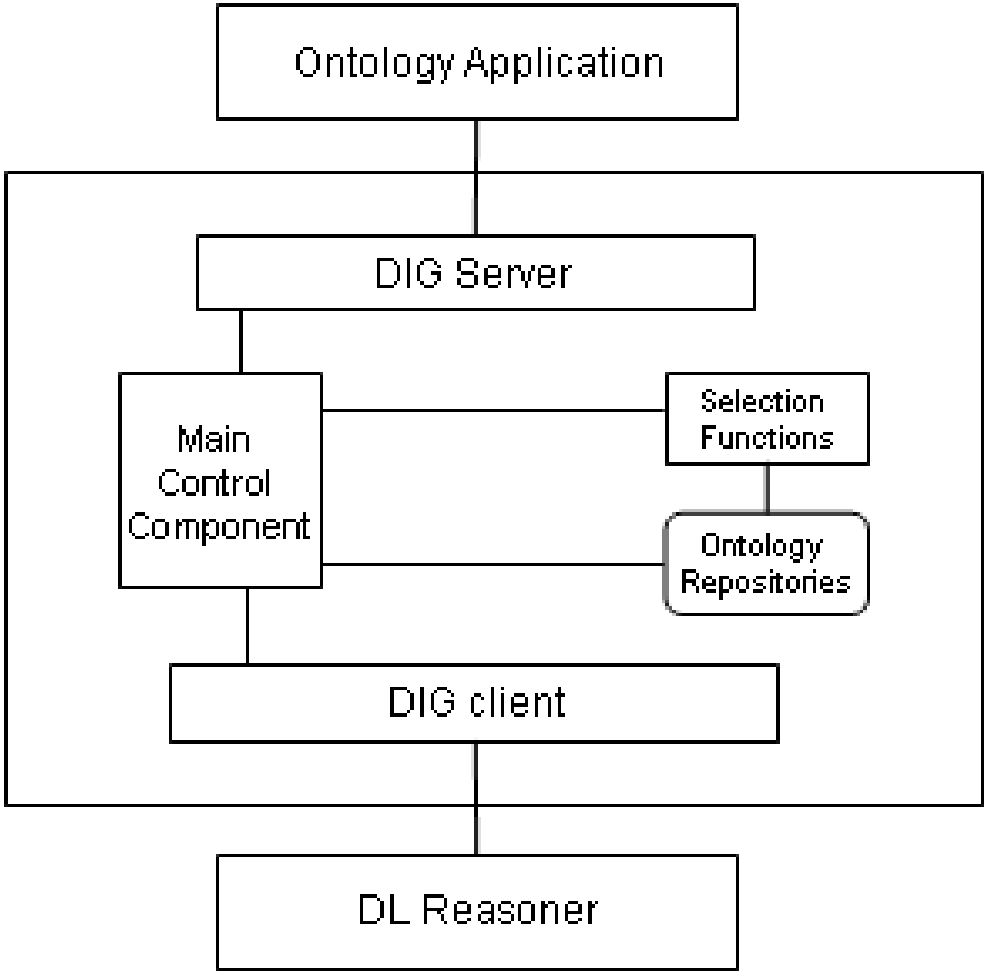
- Over-determined: if the result falls in the range or query and also falls outside the range of query (which is realistically not possible)
- Accepted: if the results falls in the range of the query and doesn't fall outside the range of the query
- Rejected: if the results does not falls in the range of the query and only fall outside the range of the query.
- Undetermined: if the results does not falls in the range of the query and also does not fall outside the range of the query.

Comparing the quality of answers

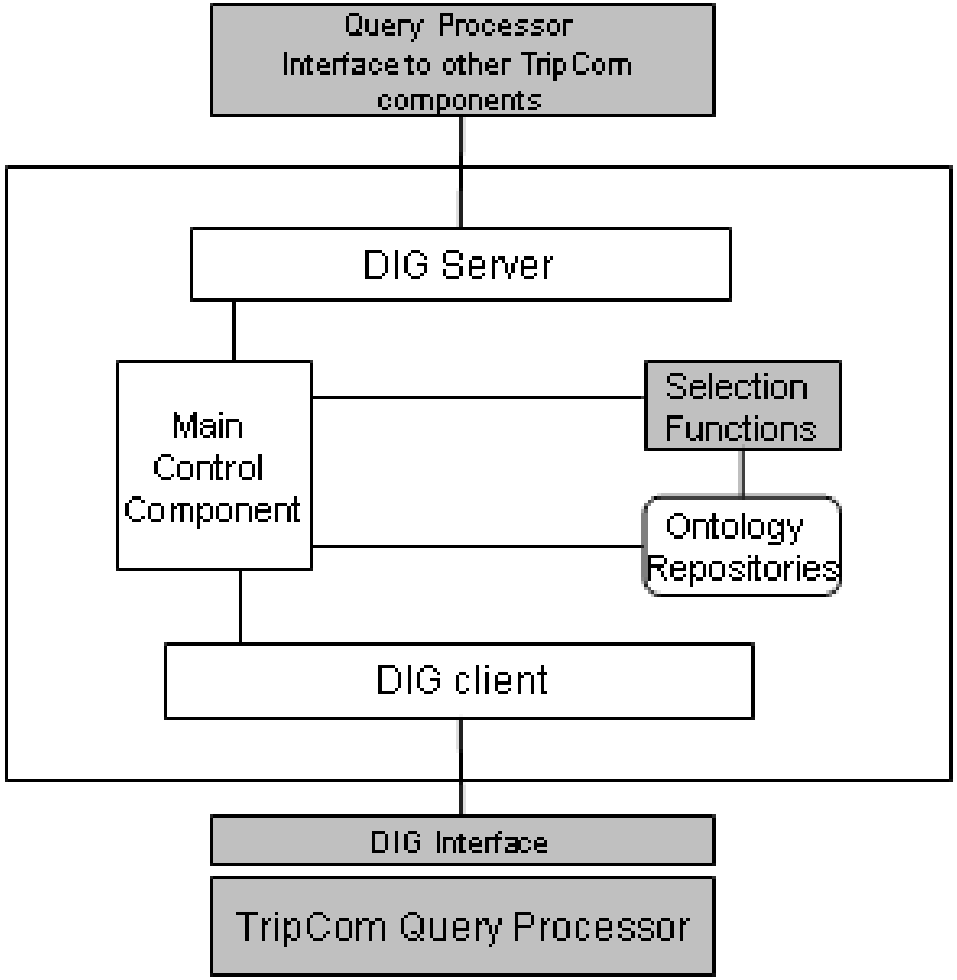
- Intended Answer: if it is exactly equal to the intuitive answer
- Counter-intuitive Answer: if the actual answer is exactly opposite to the intended answer
- Cautious Answer: The intuitive answer is ’accepted’ or ’rejected’, but the actual answer is ’undetermined’.
- Reckless Answer: the actual answer is ’accepted’ or ’rejected’, but the intuitive answer is ’undetermined’

Sixth Framework Programme Priority 2
Information Society Technologies (IST)
Specific Targeted Research Project

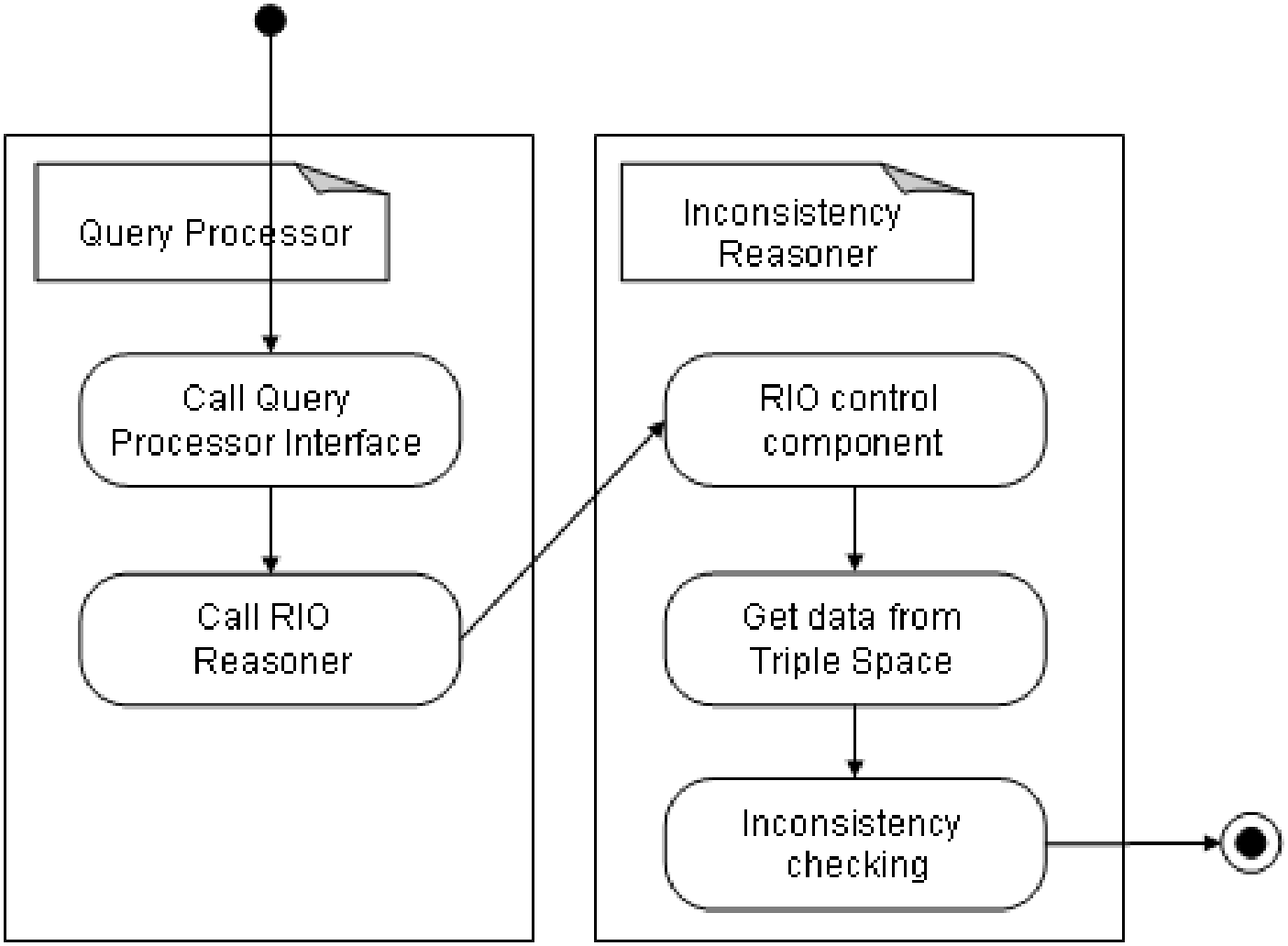
RIO – A Reasoner for Inconsistent Ontologies



RIO with TripCom Query Processor



Interaction b.w RIO and TripCom inconsistent Reasoning



PION Demonstration …

References

- The DIG Interface. http://dig.sourceforge.net/

- Zhisheng Huang, Frank van Harmelen, Annette ten Teije, Perry Groot, and Cees Visser, "Reasoning with Inconsistent Ontologies: a general framework", Deliverable D3.4.1.1, EU-IST Integrated Project (IP) IST-2003-506826 SEKT (SEKT: Semantically Enabled Knowledge Technologies)

- Zhisheng Huang, Frank van Harmelen, and Annette ten Teije, “Reasoning with inconsistent ontologies”, in the proceedings of International Joint Conference on Artificial Intelligence (IJCAI 2005), August 2005, held at Edinburgh, Scotland.

WP3 Resources

WP 3: Triple Space Interaction						FUB	Prof	LFUI	TUW	ONTO	TID
					Target PM	20	17	15	6	12	2
		In charge	Start	End	Duration						
T3.1	Specification of a semantic Linda model		1	6	6	4	1	2	1	0	0
T3.2	Implementation of a semantic Linda model		7	12	6	6	0	0	0	6	0
T3.3	SoA and requirements of query languages		1	12	12	2	5	1	0	0	1
T3.4	Specification of semantic matching		13	18	6	3	1	2	0	2	1
T3.5	Implemenation of semantic matching in distributed spaces		19	23	5	3	0	5	3	1	0
T3.6	Implemenation of a distributed semantic query tool		24	28	5	1	7	3	0	2	0
T3.7	Evaluation/refinement of implementation		29	33	4	1	3	1	2	1	0
					Sum	20	17	14	6	12	2

FUB	1 / 3
Profium	1 / 1.8
LFUI	1 / 5
TUW	0.2 / 3
ONTO	0.1 / 1

WP3 next steps (T3.5)

- Local query optimisation
  - Query analysis
    - Cost model currently based on the algebraic SPARQL semantics
    - Structural analysis by schemas needs details from the TS Adapter
  - Query engine selection
    - Based on chosen query language and queried schema
    - Support for multiple query engines needs to be factored in
  - Support for incomplete/inconsistent reasoning
- Distributed query optimisation
  - Query deconstruction
    - Use query costs and space content (from DM/MM)
  - Query approximation
    - Weakening the query to win efficiency
    - Potentially reconstruct results in the post-processing

Sixth Framework Programme Priority 2
Information Society Technologies (IST)
Specific Targeted Research Project

WP3 next steps (API)

- TS and Management API specifications as PDF in the Tripcom CMS -> comments on these are CLOSED
- SPARQL as base query language (supported in ORDI)
- Considerations to support restrictions and extensions of SPARQL for more efficiency and expressiveness resp. (Ontosearch2, DLVhex)
- Use cases must consider particular querying needs that they have (revision) e.g. SPARQL GRAPH clause
- Use cases must provide ontologies and sample data
- Reasoner benchmarking to be done ASAP – it acts as input to the query engine selection work

Sixth Framework Programme Priority 2
Information Society Technologies (IST)
Specific Targeted Research Project

End of Document