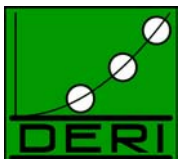
WSMO Discovery

Realization in Semantic Web Fred

Michael Stollberg

- 03 November 2004 -



Content

1. Starting Position

2. General Architecture of a Discoverer

3. Knowlegde Types & Discovery Techniques
   - Action Knowlegde Discovery
   - Object Knowledge Discovery

4. Realization in VAMPIRE

Starting Position

- WSMO Discovery Framework (WSMO D5.1)
  - levels / types of discovery mechanisms
  - matching notions definition
- SWF Framework
  - framework for agent cooperation with usage of WSMO technologies
  - requirements for WSMO Discovery Realization
- preliminary trial & error approaches in WSMO

Discoverer Architecture

initial architecture for

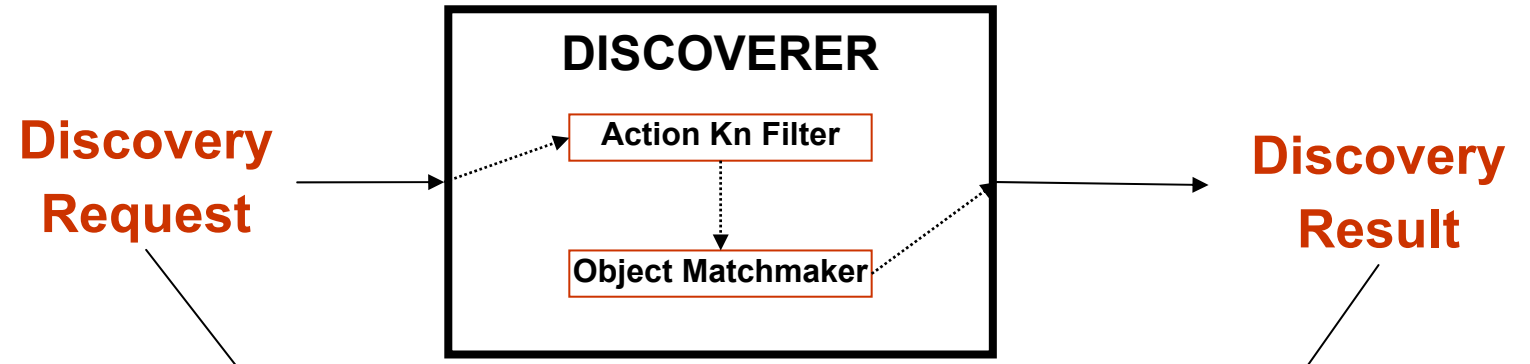
Web Service Discovery

- WSMO Goal (WSMO 1.0)
- Requested Service (WSMO 1.1)

A (Web) Service that can satisfy the Goal

Generalized Architecture for Discoverer

can be used for discovering any WSMO top level notion



- can be aligned with any WSMO top level notion
- has „more detailed“ information for discovery
  - o Action Knowledge
  - o Object Knowledge
  - o Matchmaking Notion

- can be aligned with any WSMO top level notion
- “matching” according to:
  - o Action Compatibility
  - o Object Matching wrt to Matchmaking Notion

Action –vs – Object

- Knowledge Types Distinction
  - Action = activity to be performed on object (e.g., buy, sell, ..)
  - Object = entity that action is performed on (e.g. product, ticket, …)
  - Relation: Action(Object)
- Action and Object Knowledge is different
  - "Action" defines what is to be done; interacting entities need to have **compatible actions** (e.g. buy <-> sell)
  - "Object" defines whereon a action is to be performed; interacting entities need to have **not-contradicting objects**
- Object Matching is not enough, because 2 resources might have not-contradicting objects but not-compatible actions

Action –vs – Object

⇒*different Discovery Technologies needed*

- Action Knowledge Discovery
  - aim: determine **Compatibility of Actions in Resources**
  - approach: **Action Knowledge Ontologies + Filter**

- Object Knowledge Discovery
  - aim: determine **Not-Contraction of Objects in Resources**
  - approach: **Semantic Resource Description + Matchmaking**

In current WSMO Ontologies & resource descriptions Action & Object Knowledge is implicit; it has to be provided explicitly to Discoverer

Action Knowledge Ontology

```
concept action
  compatibleAction ofType set action
concept buy subConceptOf action
  compatibleAction ofType set sell
concept sell subConceptOf action
  compatibleAction ofType set buy
```

Action Taxonomy

```
concept resource
  hasAction ofType set action
concept goal subConceptOf resource
concept service subConceptOf resource

concept buyergoal subConceptOf goal
   hasAction ofType set buy
concept sellerservice subConceptOf service
   hasAction ofType set sell
```

Resource Taxonomy

ALL resources of application need to be instantiated

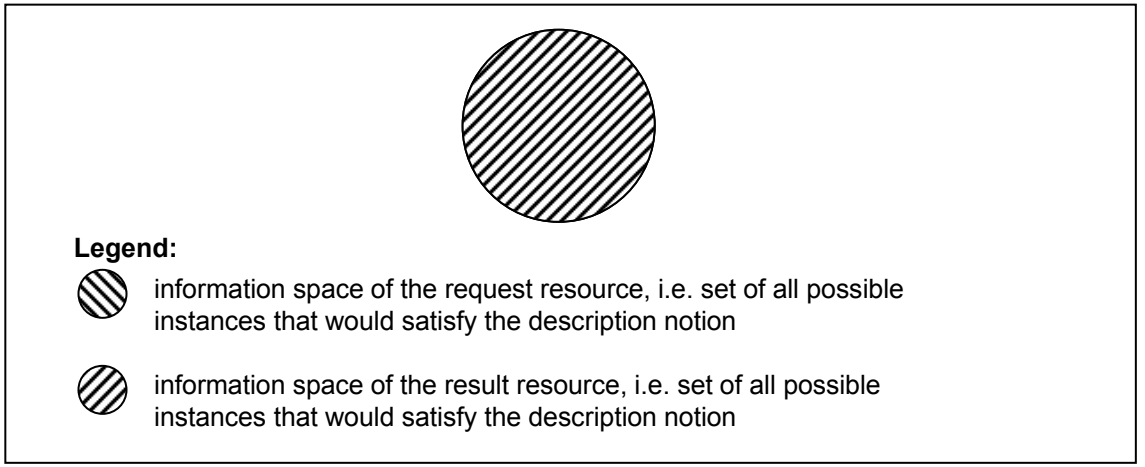
```
instance <<http://.../goal1>> memberOf buyergoal
instance <<http://.../ws7>> memberOf sellerservice
```

Object Matchmaking

- Based on WSMO D5.1 "WSMO Discovery"
  - defines different approaches
  - defines matching notions
- further insights on Object Matchmaking
  - applicability of matching notions
  - why matching notion needs to be specified in Discovery Request
- Realization with Vampire

Exact Match

```
exactMatch(request, result) impliedBy
   forAll X ( request(X) equivalent result(X))
```

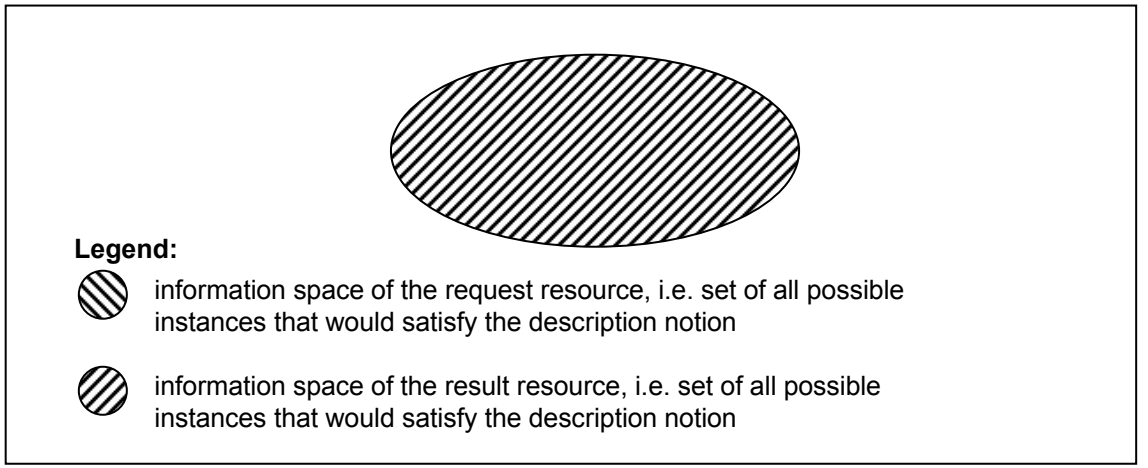


Usage:

- e.g. effect matching
  - all effects of request resource shall be satisfied
  - no more effects by result resource

PlugIn Match

```
pluginMatch(request, result) impliedBy
  forAll X ( request(X) implies result(X))
```



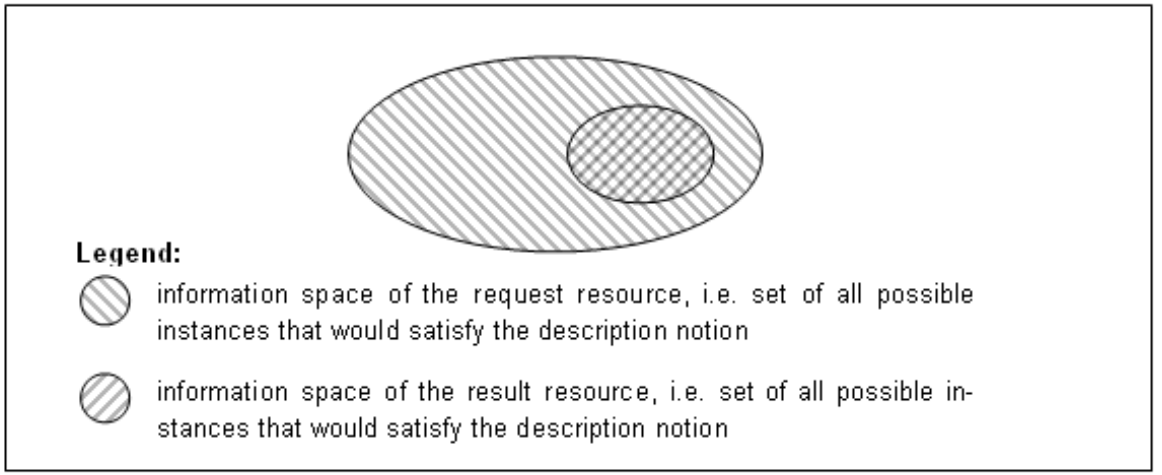
Usage:

When all instances provided by result resource shall satisfy request

- e.g. “give product information wherefore holds …”
- e.g. “find all restaurants wherefore holds …”

Subsumption Match

```
subsumptionMatch(request, result) impliedBy
  forAll X ( request(X) impliedBy result(X))
```



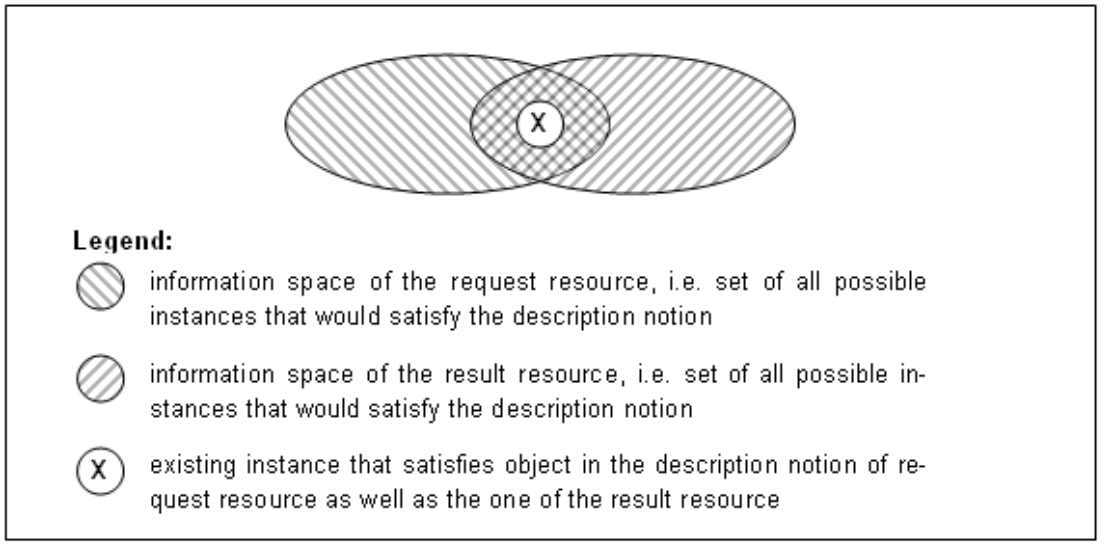
Usage:

When several result resources are needed to satisfy the request

- not used in SWF
- applicable for discovering result resources that have to be composed to satisfy request

Intersection Match

```
intersectionMatch(request, result) impliedBy
  exists X ( request(X) and result(X))
```



Usage:

When there shall be 1 object retrieved that satisfies the request

- e.g. receive 1 purchase contract

Realization in VAMPIRE

- Theorem Prover
  - developed at University of Manchester
  - winner of several CASC competitions
- works with TPTP
  - FOL language, “standardized”
  - Transformation WSML -> TPTP without loss of information & “easy”

Knowledge Representation

1. **Ontology Theory**
   - basic ontological relations (transitive subsumption, etc.)
   - WSMO ontology meta model (when working on WSMO)
2. **Universe**

   all domain ontology knowledge needed
   - Ontology schemas (allows to work on resource description)
   - Generic Instances
3. **Knowledge Base**
   - pre-defined instances
   - optional (but very useful)
4. **Resource Description**
   - WSMO resource description notions (postcondition, effects, etc)
   - only those notions which are needed for object matchmaking

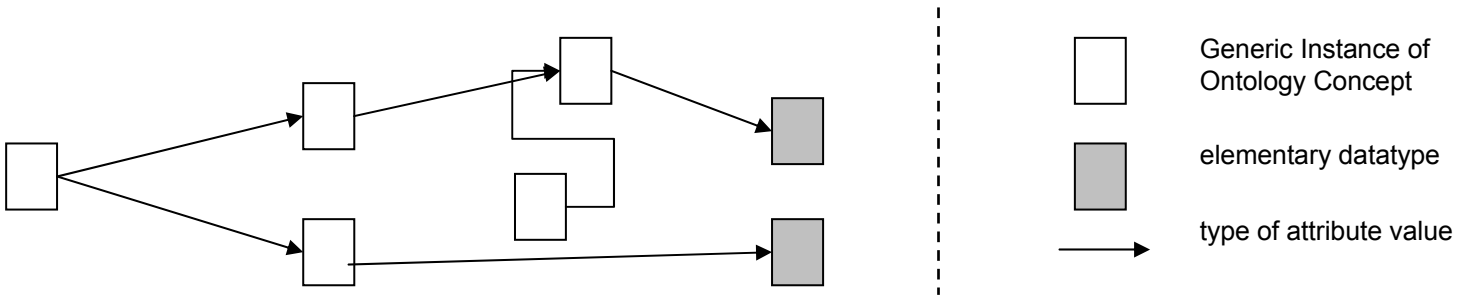
Generic Instance in Universe

- Generic Instance definition:
  - existentially quantified object
  - complete fact with universally quantified variables as attribute values

```
forAll ?A, ?B, ?C(
        exists ?X(?X instanceOf concept[
          attributeA hasValue ?A,
          attributeB hasValue ?B,
          attributeC hasValue ?C
            ]
      ) ) .
```

Generic Instances – Why?

- support for Intersection match:
  - Intersection Matching Notion searches for an existing object wherefore the request and result predicate do not contradict
  - If there are Generic Instances for all concepts of all used domain ontologies, then the existence of an instance of every possible ontology object can be proved by constructing a hypothetical graph wherein for each node there exists a generic instance.



not new – same technique in other approaches for realization of intersection match

Structure of Proof Obligation

1. **Needed Knowledge Resources**
   - include Ontology Theory and Universe
   - Knowledge Base optionally

2. **Resource Description notion to be matched**
   - Request Resource & Result Resource
   - Only those notions that are to be matched

3. **Object Matchmaking notion**
   - Include as specified in the Discovery Request

=> *easy to generate dynamically*

VAMPIRE (+) and (–)

- **Pro:**
  - works on structural level, i.e. no need for intermediate insertion of hypothetical facts in system's knowledge base
  - Ontology Theory + Universe as the only pre-defined knowledge resources needed
  - stable & quite fast compared to other CASC theorem provers
- **Con:**
  - does not have any built-in functions =>
    - no support for basic datatype processing
    - no arithmetics
  - everything has to be provided as explicit FOL theories

</ WSMO Discovery Realization – Theoretic Part>

usage in SWF + Prototype upcoming …