Partner and Service Discovery for Collaboration Establishment with Semantic Web Services

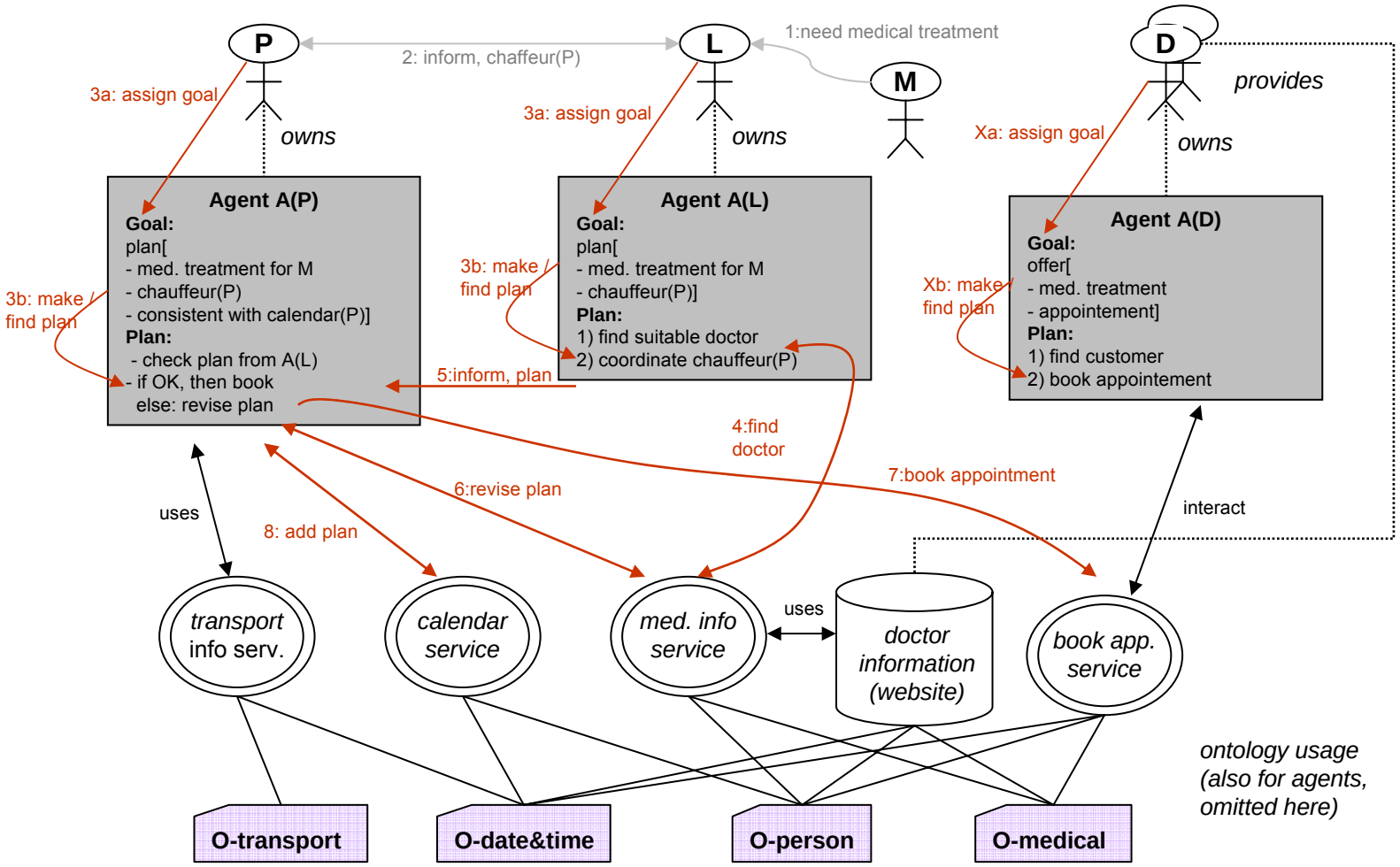
Michael Stollberg, Uwe Keller, Dieter Fensel
DERI – Digital Enterprise Research Institute

3rd International Conference on Web Services (ICWS 2005)
Orlando, Florida, July 12th

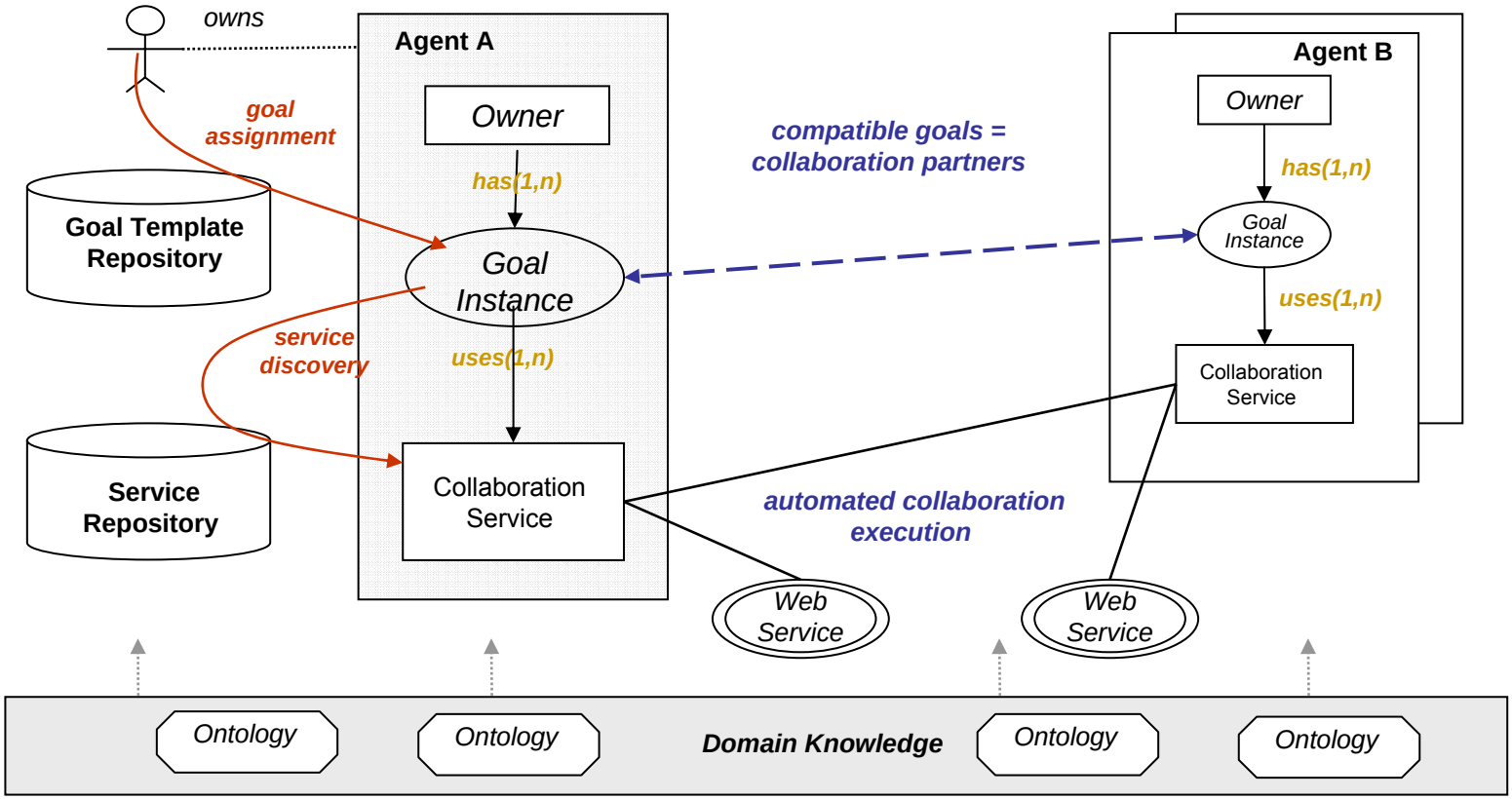
Content

1. Automated Collaboration on the Semantic Web
2. Semantically Enabled Discovery
3. Partner and Service Discoverer Architecture
4. Findings
5. Conclusions

Motivating Example



Conceptual Architecture



Agent Semantic Description

Agents

electronic representative of real world entity that wants to achieve an objective by automated collaboration

Class agent
- hasNonFunctionalProperties **type** nonFunctionalProperties
- importsOntology **type** ontology
- usesMediator **type** ooMediator
- owner **type** owner
- collaboration **type** collaboration
- history **type** collaboration

Class owner
- hasNonFunctionalProperties **type** nonFunctionalProperties
- owner **type** instance
- preference **type** axiom
- policy **type** axiom
- serviceUsagePermission **type** service

Class collaboration
- hasNonFunctionalProperties **type** nonFunctionalProperties
- goal **type** goal *single-valued*
- service **type** service

SWF

Goals Semantic Description

Goal Template

predefined schema of user objective, described as WSMO 1.0 goal

Class goalTemplate **is-a** wsmov1.0Goal

hasNonFunctionalProperties **type** nonFunctionalProperties
importsOntology **type** ontology
usesMediator **type** {ooMediator, ggMediator}
hasPostcondition **type** axiom
hasEffect **type** axiom

Goal Instance

concrete objective assigned to an agent, instantiated Goal Template, client for service usage

Class goalInstance **sub-Instance** goalTemplate

hasNonFunctionalProperties **type** nonFunctionalProperties
importsOntology **type** ontology
usesMediator **type** ooMediator
hasPostcondition **type** axiom
hasEffect **type** axiom
hasSubmission **type** instance
hasResult **type** instance
goalResolutionStatus **type** goalResolutionStatus

Collaboration Service Description

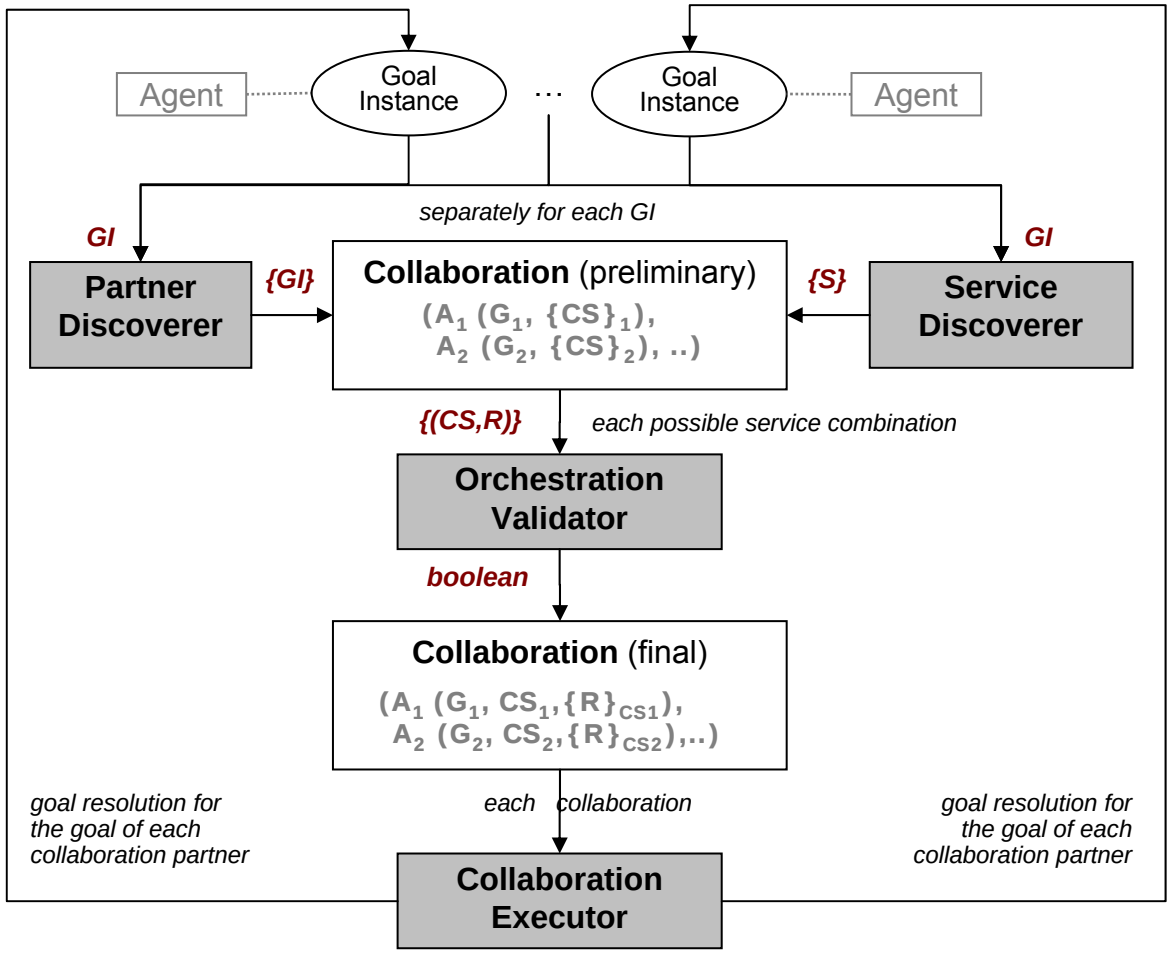
Collaboration Services

facility an agent uses for participating in an automatically executed collaboration
interacts with other agents and utilizes Semantic Web resources via its orchestration

Class collaborationService **is-a** wsmoService
- hasNonFunctionalProperties **type** nonFunctionalProperties
- importsOntology **type** ontology
- usesMediator **type** ooMediator
- hasCapability **type** capability *single-valued*
  - hasSharedVariables **type** sharedVariable
  - hasPrecondition **type** axiom
  - hasAssumption **type** axiom
  - hasPostcondition **type** axiom
  - hasEffect **type** axiom
- hasInterface **type** interface
  - clientInterface **type** serviceInterfaceDescription
  - orchestration **type** serviceInterfaceDescription

Class serviceInterfaceDescription **sub-Class** wsmoServiceInterface
- hasNonFunctionalProperties **type** nonFunctionalProperties
- importsOntology **type** ontology
- usesMediator **type** ooMediator
- hasVocabulary **type** concept_in, concept_out, concept_controlled
- hasState **type** ontology
- hasGuardedTransition **type** if (condition) then *update* – for client interface
  if (condition) then *operation(CS, R, update)* – for orchestration

Collaboration Management



Semantically Driven Discovery

find appropriate facility (Web) Service for automatically resolving a goal as the objective of a requester

Key Word Matching
match natural language key words in resource descriptions

Controlled Vocabulary
ontology-based key word matching

Semantic Matchmaking
… what Semantic Web Services aim at

SWF

Action and Object Knowledge

- 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, ...)
- Action and Object Knowledge is different
  - "Action" defines what is to be done; interacting entities need to have **compatible actions** (e.g. buy <-> sell)

  => Action-Resource Ontology
  - "Object" defines whereon an action is to be performed; interacting entities need to have **not-contradicting object definitions**

  => Set-theoretic Object Matchmaking

- **Combination is needed** (agents / resources might have not-contradicting objects but incompatible actions)

Action-Resource Ontology

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

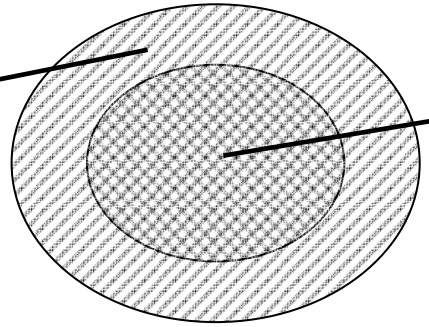
all resources are defined as instances of resource types

- **allows determining action compatibility / equality of agents & resources**
- **supports handling multi-party collaborations**

Object Matchmaking

Set-Based Resource Descriptions

Information Space
all possible instances
of used ontologies



Description Notion
all possible instances that
satisfy restricted information space

```
postcondition
  definedBy
    exists ?PurchaseItem(?PurchaseItem[
      item hasValue ?PurchaseFurniture
    ] memberOf swfmo:product) and
    exists ?PurchaseFurniture(?PurchaseFurniture[
      material hasValues {wood},
    ] memberOf furn:chair) and
  ?X[
     purchaseItem hasValue ?PurchaseItem,
     buyer hasValue kb:MichaelStollberg,
     purchasePayment hasValue kb:MSCreditCard
   ] memberOf swfmo:purchaseContract .
```

Goal Instance Postcondition

- Objective: receive a purchase contract for a wooden chair for Michael Stollberg, payment with credit card

- meta-varibale X (dynamically quantified by matchmaking notion)

- restrictions on several ontology notions

- WSML syntax

Set Theoretic Matchmaking Notions

1. Exact Match:

   $D_Q, D_R, O, M \models \forall x.\ (D_Q <=> D_R)$

2. PlugIn Match:

   $D_Q, D_R, O, M \models \forall x.\ (D_Q => D_R)$

3. Subsumption Match:

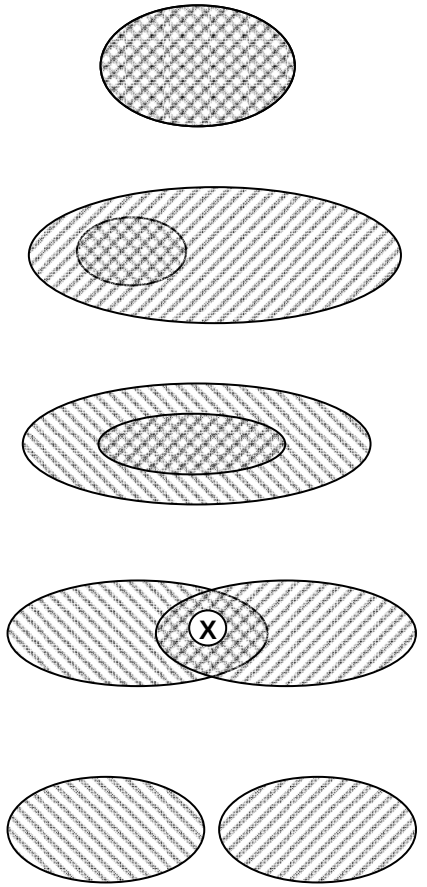
   $D_Q, D_R, O, M \models \forall x.\ (D_Q <= D_R)$

4. Intersection Match:

   $D_Q, D_R, O, M \models \exists x.\ (D_Q \wedge D_R)$

5. Non Match:

   $D_Q, D_R, O, M \models \neg\exists x.\ (D_Q \wedge D_R)$



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”

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*

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

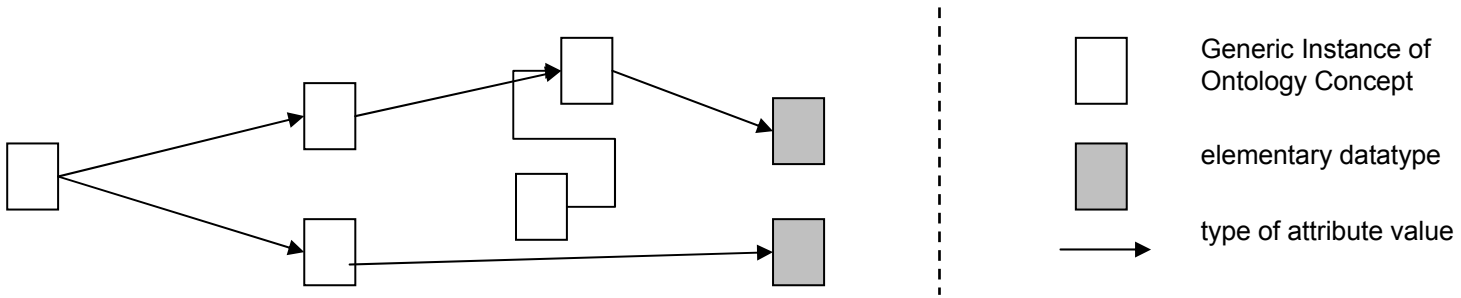
Generic Instance in Universe

- 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

VAMPIRE (+) and (–)

- **Pro:**
  - Not bound to limited reasoning facilities supported by existing reasoners

  => we can create object matchmaking as
  - Ontology Theory + Universe as the only pre-defined knowledge resources needed

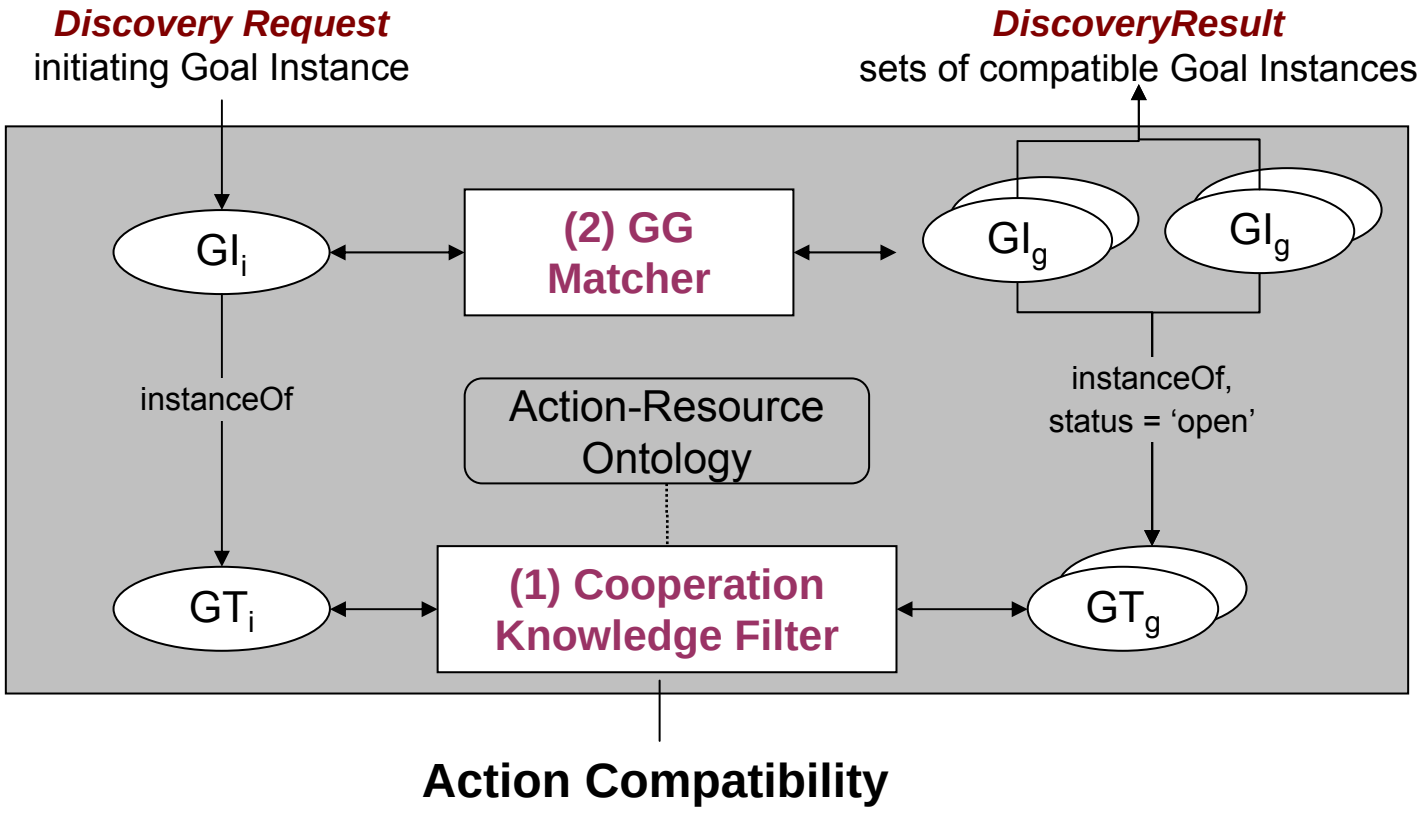
  => efficient & easy to handle at runtime
- **Con:**
  - does not have any built-in functions
    - no support for basic datatype processing
    - no arithmetics

  => everything has to be provided as explicit FOL theories

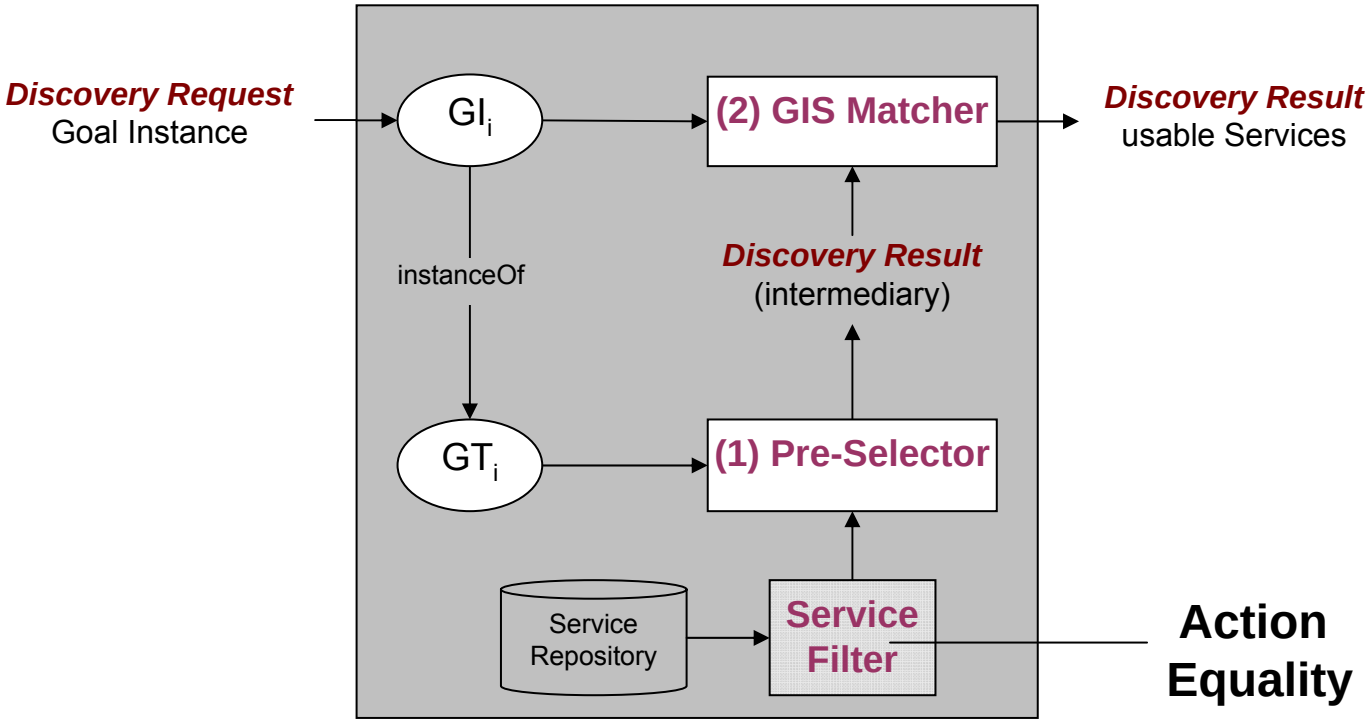
Partner & Service Discovery Components

- run independently, invoked by agents
- apply ARO & OMM as defined above
- Design Principles:
  1. Common Discoverer Architecture
  2. Modularized Functionality
  3. Layered Architecture (stepwise narrow search space)
  4. Reduction of expensive operations (efficiency)

Partner Discoverer Architecture



Service Discoverer Architecture



Findings

- Functional Correctness approved in use cases
  - Distinction Action – Object Knowledge needed
  - Performance (Vampire):
    - average 1-2 sec (domain dependent !!)
    - correctness & scalability more important than celerity
- Goal and Service descriptions are mostly constrained object definitions without complex logical expressions => object matchmaking rather logical reasoning
- Industrial applicability:
  - "hide logics from users and system developers"
  - semantic descriptions by Knowledge Engineers
  - exhaustive tool support required for users and system developers
  - abandon logical expressivity if it hampers automation

Conclusions

- Framework for Collaboration on the Semantic Web
  - agents, goals, Web Services
  - clear separation of partner & service discovery
- Realization of Semantic Discovery
  - distinction Action-Object Knowledge
  - semantic discovery techniques
    - Action-Resource Ontology (action compatibility / equality)
    - Set-based Object Matchmaking (not-contradicting objects)
  - discoverer architectures (modularized, layered, efficient)
- based on Web Service Modeling Ontology WSMO
  - semantic element descriptions
  - set-based object matchmaking