Extended MaMaS[1] DIG Description Logic Interface 1.1

1 Introduction

Most description logic (DL) systems present the application programmer with a functional interface, often defined using a Lisp-like syntax. Such interfaces can be more or less complex, depending on the sophistication of the implemented system, and can be more or less compliant with a specification such as KRSS. As part of the activity of the Description Logic Implementation Group (DIG) a new interface for DL systems has been defined. We extended this interface providing support for non-standard inferences implemented in the MaMaS system, available as an HTTP service at: http://dee227.poliba.it:8080/MAMAS-tng/DIG.

This report presents main features of the MaMaS extension. For what is not covered here, please refer to standard specification [4,5].

2 Protocol

The DIG interface uses HTTP as the underlying transfer protocol. The use of HTTP allows client (and server) developers to use existing libraries for implementation.
For specifying the DIG requests, the protocol prescribes the use of HTTP POST. The body of the request must be an XML encoded message corresponding to a DIG request. There are four types of requests: *identification, kb management, tell and ask*, for, respectively, identifying the reasoner and its capabilities, managing the knowledge bases, adding information to a knowledge base, querying information from a knowledge base. Content-Type is text/xml, and Content-Length must be specified and must be correct. The server will use the root element of the message body to determine the message type. Unless there is a low-level error, the server should return 200 OK. As with requests, Content-Type is text/xml and Content-Length must be present and correct. The body of the response must be an XML encoded message corresponding to a DIG response as described below.

3 Reasoner Identification

An aspect lacking from previous specifications (for instance KRSS) was the ability to identify which reasoner was actually behind the interface. This is particularly important when we may have a number of different reasoners supplying conforming interfaces. Ideally, we would expect all reasoners implementing the specification to support the entire concept language and tell/ask functionality. In reality, this is unlikely to be the case in the short term, and some reasoners may choose not to implement, for example, support for concrete domains. In order to cope with this, along with information regarding their identification, a reasoner should also supply details of the language it supports. This will enable clients to decide whether or not the reasoner will be useful, or guide the clients as to the questions they can ask the reasoner. Such introspective descriptions of tools and services are crucial to supporting dynamic component and service discovery as is being pursued in areas such as the Grid. In the current specification this capability information is rather primitive, and essentially amounts to a list of the concept forming operators, tell assertions and queries supported. It is assumed that all reasoners will support primitive concepts and roles.
An IDENTIFIER request contains a single `<getIdentifier>` element as below:

```
<?xml version="1.0" encoding="UTF-8"?>
<getIdentifier
 xmlns="http://dl.kr.org/dig/2003/02/lang"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://dl.kr.org/dig/2003/02/lang
```

[1] Amended and extended version of the original DIG1.1. specification document, MaMaS compliant. Contributed by Stefano Coppi, Francesco di Cugno and Eufemia Tinelli.

```
 http://dlweb.man.ac.uk/dig/2003/02/dig.xsd"/>
```

The response to an IDENTIFIER request should consist of a single `<identifier>` element. This element includes a string identifying the reasoner along with its version number (which is itself a sequence of integers separated by the . character) and an optional explanatory message. In addition, a <supports> element describes the language and capabilities of the reasoner. Below we show the MaMaS identification response:

```
<?xml version="1.0" encoding="UTF-8"?>
<identifier xmlns="http://dl.kr.org/dig/2003/02/lang"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:schemaLocation="http://dl.kr.org/dig/2003/02/lang http://dl-
web.man.ac.uk/dig/2003/02/dig.xsd" name="MAMAS-tng" version=" 1.0"
message="MAMAS running on http://193.204.59.227:8080/MAMAS-tng/DIG">
        <supports>
                <language>
                        <top/>
                        <bottom/>
                        <catom/>
                        <and/>
                        <all/>
                        <atMost/>
                        <atLeast/>
                        <ratom/>
                        <individual/>
                </language>
                <tell>
                        <defconcept/>
                        <defrole/>
                        <defindividual/>
                        <clearindividual/>
                        <impliesc/>
                        <equalc/>
                        <disjoint/>
                        <instanceof/>
                </tell>
                <ask>
                        <allConceptNames/>
                        <allRoleNames/>
                        <allIndividuals/>
                        <toldValues/>
                        <satisfiable/>
                        <subsumes/>
                        <disjoint/>
                        <parents/>
                        <children/>
                        <ancestors/>
                        <descendants/>
                        <equivalents/>
                        <instances/>
                        <types/>
                        <instance/>
                        <toldDefinition/>
                        <matchType/>
                        <rank/>
                        <abduce/>
                        <contract/>
                </ask>
        </supports>
</identifier>
```

4 Knowledge Base management

A single client can deal with multiple knowledge bases. URIs are used in order to identify the different knowledge bases. When a request is made to the MaMaS system to create a new knowledge base, it (if successful) will return to the client a URI which the client can then use to identify the knowledge base during TELL and ASK requests (see Sections 5 and 6). A request for a new knowledge base consists of a single `<newKB/>` element as below:

```
<?xml version="1.0" encoding="UTF-8"?>
<newKB xmlns="http://dl.kr.org/dig/2003/02/lang"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://dl.kr.org/dig/2003/02/lang
 http://dl-web.man.ac.uk/dig/2003/02/dig.xsd"/>
```

extended DIG 1.1 MaMaS

<newKB/>

MaMaS allows to use two new attributes for <newKB/>:

- shared: the only values to be used are true and false. In MaMas, when a new knowledge base is created, each kb uri is associated with the IP address of the owner client instantiating the kb. If the shared attribute is set to false, only the owner is authorized to submit tells statements and change the knowledge base as well as to submit asks. In this case, requests from IP addresses different form the one of the owner can be only asks. If the shared attribute is set to true, then there is no restriction in sending both tells and asks statements. true is the default value.
- permanent: the only values to be used are true and false. In MaMas, if a knowledge base is not used for more than 300 seconds (5 minutes), the kb is automatically released. If a user wants to maintain the kb indefinitely, the permanent attribute must be set to true. If the permanent attribute is set to false, then the default behavior using a timeout is kept. false is the default value.

The response to a new KB request consists of a single <response/> element. This element contains either a single <kb/> element indicating the URI that the reasoner has allocated for the KB or an <error> message indicating an error occurred. Below there is a response from <newKB/> request:

```
<?xml version="1.0" encoding="UTF-8"?>
<responses xmlns="http://dl.kr.org/dig/2003/02/lang"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:schemaLocation="http://dl.kr.org/dig/2003/02/lang http://dl-
web.man.ac.uk/dig/2003/02/dig.xsd">
  <kb uri="uri:uuid:2adcfcf0-57af-11da-9f6d-acefb10a5c7e"/>
</responses>
```

The server creates a new UUID to refer to the knowledge base. This URI will then be used during TELL and ASK requests made with respect to the knowledge base.
Clients can release a knowledge base through a request consisting of a single <releaseKB/> element with an attribute specifying the KB to release. Once a knowledge base has been released, any requests made using the URI should result in an error. Below you can see an example release request:

```
<?xml version="1.0" encoding="UTF-8"?>
<releaseKB xmlns="http://dl.kr.org/dig/2003/02/lang"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://dl.kr.org/dig/2003/02/lang
 http://dl-web.man.ac.uk/dig/2003/02/dig.xsd"
 uri="uri:uuid:2adcfcf0-57af-11da-9f6d-acefb10a5c7e"/>
```

and the response if the release is successful:

```
<?xml version="1.0" encoding="UTF-8"?>
<response xmlns="http://dl.kr.org/dig/2003/02/lang"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://dl.kr.org/dig/2003/02/lang
 http://dl-web.man.ac.uk/dig/2003/02/dig.xsd">
```

```
    <ok/>
<response>
```

5 Concept Language

The original DIG's specification concept language is based on $\mathcal{SHOIQ}_n^-$, that is a description logic that includes the standard boolean concept operators (and, or, not), universal and existential restrictions, cardinality constraints, a role hierarchy, inverse roles, the one-of construct and concrete domains. For our MaMaS system, we adopted the $\mathcal{ALN}$ description logic, which has polynomial complexity inferences – for bushy TBoxes, both for standard and non-standard inferences. Compared with the original DIG 1.1 specification, our version has a subset of the concept language tags, which are grouped in the following table:

Primitive Concepts	`<top/>` `<catom name="CN"/>`
Boolean Operators	`<and>E1... En</and>`

Property Restrictions[2]	`<all>`
	`  <ratom name="RN"/>`
	`  E`
	`</all>`
	
	`<atleast num="n">`
	`  <ratom name="RN"/>`
	`  <top/>`
	`</atleast>`
	
	`<atmost num="n">`
	`  <ratom name="RN"/>`
	`  <top/>`
	`</atmost>`

Role Expressions	`<ratom name="RN"/>`
Individuals	`<individual name="IN"/>`

From now on, for the sake of conciseness, we will use also the following notation:

RN for `<ratom name="RN"/>`
CN for `<catom name="CN"/>`
IN for `<individual name="IN"/>`
E for either CN or a concept expression nested within an `<and/>`

6 Tell syntax

A TELL request must contain in its body a `<tells>` element, which itself may consist of a number of tell statements. An overview of the concrete forms of the operators is given in the following table:

Primitive Concept Introduction	`<defconcept name="CN"/>`
	`<defrole name="RN"/>`
	`<defindividual name="IN"/>`

Concepts Axioms	`<impliesc>CN E</impliesc>`
	`<equalc>CN E</equalc>`
	`<disjoint>CN1... CNn</disjoint>`

Individual Axioms	`<instanceof>IN CN</instanceof>` **`<clearindividual name="IN"/>`**

!!!NOTE!!!

[2] For property restrictions it is possible to use also unqualified existential restrictions:

```
<some>
  <ratom name="RN"/>
  <top/>
</some>
```

- Currently, MaMaS only supports simple-TBox, that is, allowed concept axioms have a concept name on the left side.
- Since MaMaS does not support full negation but only atomic negation, `<disjoint/>` groups must be composed by concepts specialized by an `<impliesc>` axiom (sub-class axiom). Defined concepts `<equalc/>` (same-class) are not admitted in a disjoint group.

TELL requests are monotonic with respect to the TBox, i.e. once concepts information has been told to a knowledge base, it can never be retracted or removed. To modify the TBox structure, the only available option is to release the knowledge base (see Section 4) and then start again.
Notice that this restriction is valid only for concepts. For individuals it is possible to modify their definition. In order to add new information to an individual the `<instanceof/>` statement has to be used as in the following example:

```
<tells …
uri="uri:uuid:2adcfcf0-57af-11da-9f6d-acefb10a5c7e">
      <instanceof>
          <individual name="Sun_Hotel"/>
          <and>
            <all>
              <ratom name="withFacility"/>
              <and>
                <catom name="WirelessInternet"/>
              </and>
            </all>
          </and>
      </instanceof>
</tells>
```

extended DIG 1.1 MaMaS

<clearindividual/>

With MaMaS it is possible to retract the information related to an individual using the `<clearindividual/>` statement as in the following example:

```
<tells uri="uri:uuid:2adcfcf0-57af-11da-9f6d-acefb10a5c7e">
  <clearindividual name="Sun_Hotel"/>
</tells>
```

A TELL request must be contextualized with respect to a particular knowledge base (which is identified via the `uri` attribute of the `tells` element). The response to a tell will be a response message containing either an `<ok/>` element, indicating the statements have been received and interpreted correctly, or an `<error/>` element which may include an optional error code, message and detailed explanation. In addition, an `<ok/>` message may contain warnings about the tells received. Below a sample tell element is shown . This defines a single room as a bedroom with at least one bed.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<tells ...
 uri="uri:uuid:2adcfcf0-57af-11da-9f6d-acefb10a5c7e">
      <defconcept name="SingleRoom"/>
      <equalc>
        <catom name="SingleRoom"/>
        <and>
          <catom name="Bedroom"/>
          <atleast num="1">
            <ratom name="hasBed"/>
            <top/>
          </atleast>
        </and>
      </equalc>
```

```
</tells>
```

6.1 OWL and DIG for MaMaS

In the following table, correspondence between OWL, DIG and DL syntax, for the language supported by MaMaS, is shown.

OWL syntax	DIG syntax	DL syntax
`<owl:Thing/>`	`<top/>`	T
`<owl:Class rdf:ID="CN"/>`	`<catom name="CN"/>`	C
`<owl:ObjectProperty rdf: ID="RN"/>`	`<ratom name="RN"/>`	R

`<rdfs:subClassOf/>`	`<impliesc>`
	`<catom name="CN"/>`
	`E`
	`</impliesc>`
		$CN \sqsubseteq E$

`<owl:equivalentClass/>`	`<equalc>`
	`<catom name="CN"/>`
	`E`
	`</equalc>`
		$CN \equiv E$

`<owl:disjointWith/>`	`<disjoint>`
	`<catom name="CN1"/>`
	`<catom name="CN2"/>`
	`...`
	`<catom name="CNn"/>`
	`</disjoint>`
		$\neg$

`<owl:intersectionOf/>`	`<and>`$E_1 \ldots E_n$`</and>`	$E_1 \sqcap \ldots \sqcap E_n$

`<owl:allValuesFrom/>`	`<all>`
	`<ratom name="RN"/>`
	`E`
	`</all>`
		$\forall$ RN.E

`<owl:maxCardinality/>`	`<atmost num="n">`
	`<ratom name="RN"/>`
	`<top/>`
	`</atmost>`
		$\leq$ n RN

`<owl:minCardinality/>`	`<atleast num="n">`
	`<ratom name="RN"/>`
	`<top/>`
	`</atleast>`
		$\geq$ n RN

`<owl:cardinality/>`	`<and>`
	`<atleast num="n">`
	`<ratom name="RN"/>`
	`<top/>`
	`</atleast>`
	`<atmost num="n">`
	`<ratom name="RN"/>`
	`<top/>`
	`</atmost>`
	`</and>`
		= n RN

7 ASK Syntax

An ASK request must contain in its body an `<asks/>` element, which itself may consist of a number of ask statements. Each ask statement must have an attribute `id` identifying uniquely the query (within the context of the particular collection of queries). Each `asks` element must also have an attribute that identifies the knowledge base that the queries refer to. The value of this attribute is a URI which identifies the KB within the reasoner. An overview of the queries allowed in MaMaS is in the table below:

Primitive Concepts Retrieval	`<allConceptNames/>`
	`<allRoleNames/>`
	`<allIndividuals/>`

Satisfiability & Subsumption	`<satisfiable>E</satisfiable> <subsumes>`E_1 E_2`</subsumes>`

Primitive Concepts Retrieval	<allConceptNames/> <allRoleNames/> <allIndividuals/>
	<disjoint>E_1 E_2</disjoint>
Non Standard Inference Services	**<abduce>E_1 E_2</abduce>** **<contract>E_1 E_2</contract>**
Matchmaking Services	**<rank type="potential|partial">E_1 E_2</rank>** **<matchType>E_1 E_2</matchType>**
Concept Hierarchy	<parents>CN</parents> <children>CN</children> <ancestors>CN</ancestors> <descendants>CN<descendants/> <equivalents>CN</equivalents>
Individual Queries	<instances>CN</instances> <types>IN</types> <instance>IN C</instance> <toldValues>IN</toldValues> **<toldDescription>IN</toldDescription>**

For the semantics of standard DIG asks please refer to [4,5].

extended DIG 1.1 MaMaS

!!!NOTE!!!

For <abduce/>, <contract/>, <matchtype/> and <rank/> an alternative syntax is allowed involving individuals names IN. If an individual name IN is used instead of a concept expression E, then the related ask is performed on its concept description E_{IN}.

The following espression are valid in MaMas:

```
<abduce>IN1 E2</abduce>
<abduce>E1 IN2</abduce>
<abduce>IN1 IN2</abduce>

<contract>IN1 E2</contract>
<contract>E1 IN2</contract>
<contract>IN1 IN2</contract>

<rank type="potential|partial">IN1 E2</rank>
<rank type="potential|partial">E1 IN2</rank>
<rank type="potential|partial">IN1 IN2</rank>

<matchType>IN1 E2</matchType>
<matchType>E1 IN2</matchType>
<matchType>IN1 IN2</matchType>
```

<toldDescription/>

In MaMaS is possible to retrieve the information stated for an individual.

```
<?xml version="1.0"?>
<asks ...
 uri="uri:uuid:2adcfcf0-57af-11da-9f6d-acefb10a5c7e">
      <toldDescription id="q0">
            <individual name="Sun_Hotel"/>
      </toldDescription>
</asks>
```

The response to the previous query is the following:

```
<?xml version="1.0"?>
<responses ...
      <conceptset id="q0">
            <synonyms>
              <and>
               <all>
```

```
                <ratom name="withFacility"/>
                <and>
                  <catom name="WirelessInternet"/>
                </and>
              </all>
            </and>
          </synonyms>
        </conceptset>
</responses>
```

That is, all the information stated for the individual `Sun_Hotel` is returned.

<abduce/>

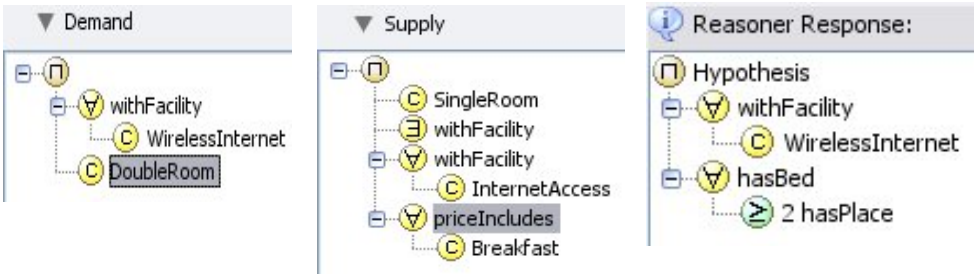
With Concept Abduction [2], given a TBox $\mathcal{T}$ and two concept expressions **D** (for Demand) and **R** (for Resource) to be matched with **D**, both satisfiable with respect to $\mathcal{T}$, if **D** is compatible with **R** – Potential Match – i.e. their conjunction is satisfiable with respect to $\mathcal{T}$, then with Concept Abduction it is possible to compute a concept expression **H** representing what is under-specified in **R** in order to completely satisfy **D** – **R** is classified by **D** with respect to $\mathcal{T}$ – taking into account the information modeled in $\mathcal{T}$. In other words, H represents an explanation on why **R** is not classified by **D** with respect to $\mathcal{T}$. In DL words:

if $\mathcal{T} \not\models \mathbf{D} \sqcap \mathbf{R} \sqsubseteq \bot$ *and* $\mathcal{T} \not\models \mathbf{R} \sqsubseteq \mathbf{D}$ *then*
find a concept **H** *such that*
$\mathcal{T} \not\models \mathbf{H} \sqcap \mathbf{R} \sqsubseteq \bot$ *and*
$\mathcal{T} \models \mathbf{H} \sqcap \mathbf{R} \sqsubseteq \mathbf{D}$

As an example consider:

```
<asks...
uri="uri:uuid:2adcfcf0-57af-11da-9f6d-
acefb10a5c7e">
  <abduce id="q1">
    <!-- D -->
    <and>
      <catom name="DoubleRoom"/>
      <all>
        <ratom name="withFacility"/>
        <and>
          <catom name="WirelessInternet"/>
        </and>
      </all>
    </and>
    <!-- R -->
    <and>
      <catom name="SingleRoom"/>
      <some>
        <ratom name="withFacility"/>
        <top/>
      </some>
      <all>
        <ratom name="withFacility"/>
        <and>
          <catom name="InternetAccess"/>
        </and>
      </all>
      <all>
        <ratom name="priceIncludes"/>
        <and>
          <catom name="Breakfast"/>
        </and>
      </all>
    </and>
  </abduce>
</asks>
```

```
<responses...>
  <conceptSetid="q1">
    <synonyms>
      <and>
        <all>
          <ratom name="hasBed"/>
          <atleast num="2">
            <ratom name="hasPlace"/>
            <top/>
          </atleast>
        </all>
        <all>
          <ratom name="withFacility"/>
          <catom name="WirelessInternet"/>
        <all>
      </and>
    </synonyms>
  </conceptSet>
</responses>
```



<contract/>

With Concept Contraction[3], given an ontology $\mathcal{T}$ and two concept expressions representing a request **D** (for Demand) and the resource **R** to be matched with **D**, both satisifiable with respect to $\mathcal{T}$. If **D** is NOT compatible with **R** – Partial Match – i.e. their conjunction is NOT satisfiable with respect to $\mathcal{T}$, then with Concept Contraction it is possible to compute a contraction **K** (for Keep) of **D** which is compatible with **R** taking into account the information modeled in $\mathcal{T}$. The solution computed for the Concept Contraction problem is a pair of concept expressions **G** (for Give up) and **K** whose conjunction is equivalent to **R** with respect to $\mathcal{T}$. In other words, **G** represents an explanation on what in **D** is not compatible with **R** (causing a Partial Match). In DL words:

if $\mathcal{T} \models \mathbf{D} \sqcap \mathbf{R} \sqsubseteq \bot$ *then*

find a pair of concept **K** *and* **G** *such that*

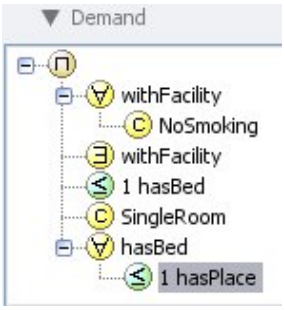
$\mathcal{T} \models \mathbf{K} \sqcap \mathbf{G} \equiv \mathbf{D}$ *and*

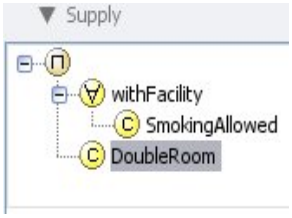
$\mathcal{T} \not\models \mathbf{K} \sqcap \mathbf{R} \sqsubseteq \bot$

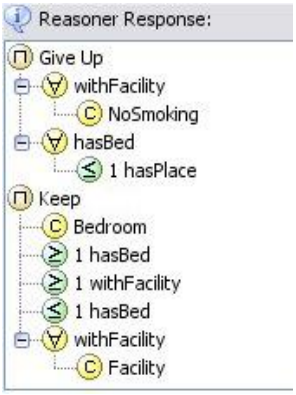
As an example consider:

```
<asks...
uri="uri:uuid:2adcfcf0-57af-11da-9f6d-
acefb10a5c7e">
  <contract id="q3">
    <!-- D -->
    <and>
      <all>
        <ratom name="withFacility"/>
        <and>
          <catom name="NoSmoking"/>
        </and>
      </all>
      <some>
        <ratom name="withFacility"/>
        <top/>
      </some>
      <atmost num="1">
        <ratom name="hasBed"/>
        <top/>
      </atmost>
      <catom name="SingleRoom"/>
      <all>
        <ratom name="hasBed"/>
        <and>
          <atmost num="1">
            <ratom name="hasPlace"/>
            <top/>
          </atmost>
        </and>
      </all>
    </and>
    <!-- R -->
    <and>
      <all>
        <ratom name="withFacility"/>
        <and>
          <catom name="SmokingAllowed"/>
        </and>
      </all>
      <catom name="DoubleRoom"/>
    </and>
  </contract>
</asks>
```

```
<responses ...">
  <conceptSet id="q3">
    <!-- G -->
    <synonyms>
      <and>
        <all>
          <ratom name="withFacility"/>
          <catom name="NoSmoking"/>
          </all>
          <all>
            <ratom name="hasBed"/>
            <atmost num="1">
              <ratom name="hasPlace"/>
              <top/>
            </atmost>
          </all>
        </and>
      </synonyms>
       <!-- K -->
      <synonyms>
        <and>
          <catom name="Bedroom"/>
          <atleast num="1">
            <ratom name="hasBed"/>
            <top/>
          </atleast>
          <atleast num="1">
            <ratom name="withFacility"/>
            <top/>
          </atleast>
          <atmost num="1">
            <ratom name="hasBed"/>
            <top/>
          </atmost>
          <all>
            <ratom name="withFacility"/>
            <catom name="Facility"/>
          </all>
        </and>
      </synonyms>
    </conceptSet>
  </responses>
```







<matchtype/>

With Match Type detection, given an ontology $\mathcal{T}$, allows to determine the match category between a request **D** and a resource **R**. In particular, matches are classified within the following categories:

Exact: The resource **R** is equivalent to the request **D**

$$\mathcal{T} \models \mathbf{R} \equiv \mathbf{D}$$

Full: The resource **R** is more specific than the request **D** and then the former completely satisfies the latter

$$\mathcal{T} \models \mathbf{R} \sqsubseteq \mathbf{D}$$

Plug-In: The request **D** is more specific than the resource **R** and then the former completely satisfies the latter

$$\mathcal{T} \models \mathbf{D} \sqsubseteq \mathbf{R}$$

Potential: The conjunction of **D** and **R** is satisfiable with respect to the ontology $\mathcal{T}$

$$\mathcal{T} \not\models \mathbf{D} \sqcap \mathbf{R} \sqsubseteq \bot \text{ and } \mathcal{T} \not\models \mathbf{R} \sqsubseteq \mathbf{D}$$

Partial: The conjunction of **D** and **R** is NOT satisfiable with respect to the ontology $\mathcal{T}$

$$\mathcal{T} \models \mathbf{D} \sqcap \mathbf{R} \sqsubseteq \bot$$

<rank type="potential"/>

With Rank Potential [1], given an ontology $\mathcal{T}$ and two concept expressions representing a request **D** (for demand) and the resource **R** to be matched with **D**, both satisifiable with respect to $\mathcal{T}$. If **D** is compatible with **R** – Potential Match – i.e. their conjunction is satisfiable with respect to $\mathcal{T}$, then Rank Potential returns a score measuring how many characteristics are unspecified in **R** in order to completely satisfy **D** – **R** is classified by **D** with respect to $\mathcal{T}$ – taking into account the information modeled in $\mathcal{T}$. The score returned by Rank Potential can be seen as a measure of **H** for the related Concept Abduction Problem.

As an example consider:

```
<rank type="POTENTIAL" id="q6">
    <and>
      <catom name="DoubleRoom"/>
      <all>
        <ratom name="withFacility"/>
        <and>
          <catom name="WirelessInternet"/>
        </and>
      </all>
    </and>
    <and>
      <catom name="SingleRoom"/>
      <some>
        <ratom name="withFacility"/>
        <top/>
      </some>
      <all>
        <ratom name="withFacility"/>
        <and>
          <catom name="InternetAccess"/>
        </and>
      </all>
      <all>
        <ratom name="priceIncludes"/>
        <and>
          <catom name="Breakfast"/>
        </and>
      </all>
    </and>
</rank>

<responses...">
  <ival id="q6">2</ival>
</responses>
```

!!!NOTE!!!
In order to obtain the upper bound for the score returned by Rank Potential, the resource **R** must be equivalent to `<top/>`. Hence, if you want to evaluate how far is a resource **R** from a demand **D**, you can use this simple formula:

$$\frac{rankPotential(R, D)}{rankPotential(top, D)} * 100$$

`<rank type="partial"/>`

With Rank Potential [1], given an ontology $\mathcal{T}$ and two concept expressions representing a request **D** (for demand) and the resource **R** to be matched with **D**, both satisifiable with respect to $\mathcal{T}$. If **D** is NOT compatible with **R** - Partial Match - i.e. their conjunction is NOT satisfiable with respect to $\mathcal{T}$, then Rank Partial returns a score measuring how many characteristics in **D** are in conflict with **R**.

References

[1] T. Di Noia, E. Di Sciascio, F.M. Donini, M.Mongiello. *A system for principled Matchmaking in an electronic marketplace*. International Journal of Electronic Commerce – IJEC, 8 (4), 2004. (Short version available in refereed paper track of Proceedings of the Twelfth International World Wide Web Conference WWW 2003, (ACM press), pp.321-330, Budapest, Hungary, May 20-24 2003.)

[2] T. Di Noia, E. Di Sciascio, F.M. Donini, M. Mongiello. Abductive matchmaking using description logics. in Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI '03), pp. 337-342, Morgan Kauffmann, Acapulco, Messico, August 9-15 2003.

[3] S. Colucci, T. Di Noia, E. Di Sciascio,F.M. Donini, and M. Mongiello. Concept Abduction and Contraction in Description Logics. In Proceedings of the 16th International Workshop on Description Logics (DL'03), volume 81 of CEUR Workshop Proceedings, September 2003.

[4] S. Bechhofer. The DIG Description Logic Interface: DIG/1.1. Available at http://dl-web.man.ac.uk/dig/2003/02/interface.pdf

[5] I. Dickinson. Implementation experience with the DIG 1.1 specification. Technical Report HPL-2004-85. Availabel at http://www.hpl.hp.com/techreports/2004/HPL-2004-85.pdf