Service Discovery and Recommendation in Rough Hierarchical Granular Computing

Liang Hu, Jian Cao
Department of Computer Science and Technology
Shanghai Jiaotong University
Shanghai, China
lianghu@sjtu.edu.cn
cao-jian@cs.sjtu.edu.cn

Zhiping Gu
Department of Electrical Engineering
Shanghai Technical Institute of Electronics & Information
Shanghai, China
raingzp@sohu.com

***Abstract*—Service discovery is one of the most vital components involved in almost all service applications. A lot of researches have paid attention to improving the matching accuracy for the given user requirement. Without a clarified requirement specification to describe the goals user really wants to achieve, any matching algorithm is useless. Goal-oriented requirement engineering is a formal requirement analysis methodology which recursively decompose a complex requirement into a set of finer grained goals. Such a hierarchical granulation structure partitions a requirement into a family of fine grain-sized granules. In order to handle uncertainties, rough set theory is employed in granular computing. For any given imprecise user requirement, a set of ordered stratified rough set approximations can be induced over all possible partitions. These approximations are used to iteratively refine imprecise requirement and recommend goals most probably desired. A matching algorithm based on six types of granule approximation is also given to describe different matching strategies for satisfying user requirement. Through the theoretical analysis and case study, it shows that rough set theory combining with granular computing is powerful to handle imprecise requirements and also provide better service quality.**

Keywords-services discovery; services recommodation; requirement engineering; granular computing; rough set theory

I. INTRODUCTION

The service-oriented architecture (SOA) has become the first considered technology to reuse and integrate all kinds of heterogeneous applications. One of the most vital issues covering almost all of service applications is the service discovery which is the basis of service selection, composition, execution and so forth. Some standard interfaces such as UDDI have been proposed to handle such requirement, but, with the upsurge in semantic technology, the keyword based discovery has become more and more powerless. A lot of new technology has introduced meta data to describe services semantically such as OWL-S [1], SAWSDL [2]. They provide a more intelligent and efficient way for services matching [3].

However, when a client (a user or an intelligent agent) intends to retrieve some service, he usually cannot get the services that he really wanted. It is not a surprise because no one has the knowledge covering all domains so that he cannot express his requirement exactly in an unfamiliar area. Such uncertain and imprecise user requirements have to be confronted when discovering services, which have a direct impact on the quality of retrieved service result. Rough Set Theory proposed by Pawlak [4] is such a kind of tool to handle the uncertainty. In his perspective, e-service intelligence requires tools for approximate reasoning about vague concepts. The rough set based methods make it possible to deal with imperfect knowledge [5].

Service discovery can be viewed as a process to find a collection of services whose capabilities are able to satisfy the requirement proposed by the client. The capabilities of a service may be viewed as goals achieved after execution. Therefore, to discover services are equivalent to finding goals that clients really want to achieve. The goal-oriented requirements engineering (GORE) suggested by van Lamsweerde has long been recognized to be an essential component involved in the requirements engineering. Goals can be formulated at different levels of abstraction ranging from high-level, strategic concerns to low-level, technical concerns [6]. Goals at different levels can be seen as a set of requirement granules to be satisfied in the perspective at that level. Naturally, with the rising research on granular computing [7, 8], it is suitable to be introduced to enhance GORE.

The basic issues of granular computing often involve two related aspects, the construction of granules and computation with granules [8]. The former aspect deals with the formation, representation, and interpretation of granule; section 2 will cover this aspect to discuss the construction of hierarchical granules in basis of goal model. The latter aspect deals with the utilization of granules in problem solving; section 3 will cover this aspect to present service recommendation and discovery in the stratified rough set approximations induced by the hierarchical requirement granules. An illustrative example will be given in section 4.

II. Formalize Requirements with Hierarchical Granules

Information granules can be treated as linked collections clumps of objects drawn together by the criteria of indiscernibility, similarity or functionality [7, 9]. In GORE, goals provide a precise criterion for sufficient completeness of a requirements specification [6]. Generally, a collection of goals with correlated functionality to achieve some objective can be regard as a granule in the requirement space. All of the granules at different levels define the requirements specification from general to specific.

A. Hierarchical Goal Model for Requirement Refinement

GORE is an effective way for requirements elicitation, specifying, analysis, negotiation and modification. Goals capture the various objectives to achieve at different levels of abstraction. Once a set of goals at some level is obtained and validated, it can be decomposed into subgoals with refinement links [6]. As shown in Fig. 1, refinement links connect general goals to more specialized goal recursively, which form the skeleton of the goal graph. Such multi-layer structure is derived from Artificial Intelligence (AI) planning where a goal is satisfied absolutely when its subgoals are satisfied. The goal structure can be viewed as the divide and conquer method in order to satisfy some complex requirement, and each goal in this structure is called Action Granule (AG) [10].

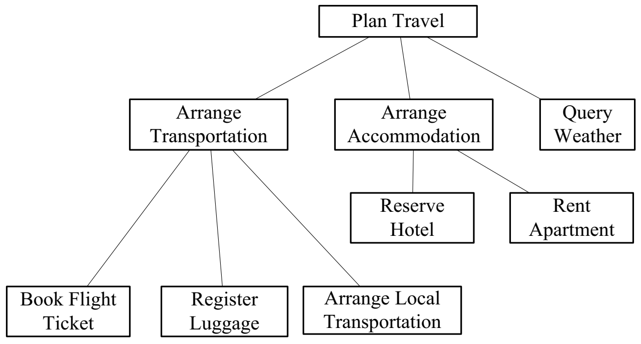

Fig. 1 A hierarchical goal model

The goal in goal model can be formalized as a clause with a main verb and several parameters, where each parameter plays a different role with respect to the verb [11]. For simplicity, in this paper we formalize the goal concept as a 2-tuple (*verb*, *target*). The element *target* designates entities affected by the goal. For example a goal for travel planning can be presented as: (*Plan*$_{verb}$, *Travel*$_{target}$). All of the goal concepts together with the refinement links in the hierarchy can be regarded as an ontology for a certain domain.

B. Partition Requirement in Goal Model

As discussed above, the goal model is a hierarchy in which a goal is satisfied when its subgoals are satisfied. The leaves in goal model can be called atomic goal (without subgoals). The goals in higher levels may be viewed as a successive bottom-up combination of leave goals, and all of the goals in the hierarchy can be represented with leave goals. If the height of a hierarchy is *N*, the root goal is top-down decomposed into a set of leave goals with *(N-1)* times. All goals in goal model are granules of different grain size to cover requirement specification.

The granules space of goal model is defined as:

$$G = \{X \mid X\ is\ a\ node\ in\ the\ goals\ hierachy\} \qquad (1)$$

Example 1, given a goal model illustrated by Fig. 2, all granules in the hierarchy can be given:

$$G = \{Ga, Gb, Gd, G1, G2, G3, G4, G5, G6, G7\}$$

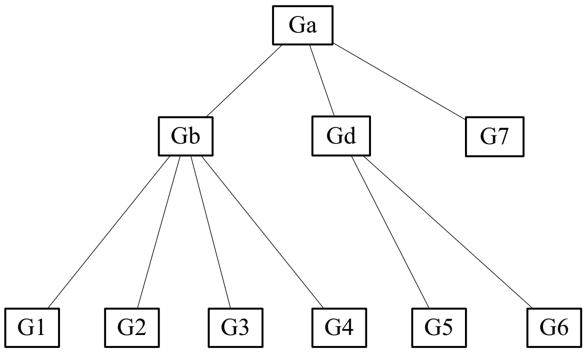

Fig. 2 A 3-layer goal model with ten nodes

A non-leaf goal can be decomposed into a set consisting of its direct subgoals. The refinement for a granule into a set of smaller granules may be viewed as zooming-in operation ω [12].

Given a granule *Gi*, we denote zooming-in operation as:

$$\omega(Gi) = \{all\ direct\ subgranules\ of\ Gi\} \qquad (2)$$

Further, if we denote all atomic goals $\{G_1, G2, \dots, G_n\}$ (leaves in goal model) as *A(G)*, given a granule *Gi*, *Gi* can be refined as a subset of *A(G)* by finite zooming-in operations. The root granule is refined as the entire *A(G)*, and the leaves consist of only singleton subsets of *A(G)*. We denote the zooming-in operation refining a granule into the subset of *A(G)* as $\omega_a(X)$, where $\omega_a(X) \subseteq A(G)$.

Example 2, consider the goal model of Example 1. Refine granules *Gb, Gd, Ga* with the zooming-in operation ω and ω_a:

$$\omega(Gb) = \omega_a(Gb) = \{G1, G2, G3, G4\}$$
$$\omega(Gd) = \omega_a(Gd) = \{G5, G6\}$$
$$\omega(Ga\}) = \{Gb, G7, Gd\}$$
$$\omega_a(Ga) = \{G1, G2, G3, G4, G5, G6, G7\}$$

For any two granules $X, Y \in G$, we can define:

$\omega_a(X) \subseteq \omega_a(Y) \Leftrightarrow X \subseteq Y$ (3)

More general, if we denoted *XS* as a subset of *G,* then, for any $XS,\ YS\ \in\ 2^G$, $\subseteq$ is defined:

$\forall x \in XS \to (\exists y \in YS)(x \subseteq y) \Leftrightarrow XS \subseteq YS$ (4)

From the hierarchy, if we select an arbitrary non-leaf node g and all its direct and indirect sub nodes, then we can obtain a new granule set induced by g denoted as *U(g)*, obviously $U(g) \subseteq G$. Since g can be refined into $\omega_a(g)$, a subset of $U(g)$ can be selected to form a partition of $\omega_a(g)$. The set of all partitions constructed from $U(g)$ is denoted by *P(g)*.

Any partition $\gamma \in P(g)$ has following properties:

(i) $\omega_a(X) \cap \omega_a(Y) = \emptyset$ $(X, Y \in \gamma, X \neq Y)$

(ii) $\bigcup_{X \in \gamma} \omega_a(X) = \omega_a(g)$

Example 3, If we have the granules given in Example 1 and select *Ga* as the root, we can construct all possible partitions $P(Ga)$:

$\gamma 1: \{Ga\} = \{\{G1, G2, G3, G4, G5, G6, G7\}\}$

$\gamma 2: \{Gb, Gd, G7\} = \{\{G1, G2, G3, G4\}, \{G5, G6\}, \{G7\}\}$

$\gamma 3: \{G1, G2, G3, G4, Gd, G7\}$

$= \{\{G1\}, \{G2\}, \{G3\}, \{G4\}, \{G5, G6\}, \{G7\}\}$

$\gamma 4: \{Gb, G5, G6, G7\}$

$= \{\{G1, G2, G3, G4\}, \{G5\}, \{G6\}, \{G7\}\}$

$\gamma 5: \{\{G1\}, \{G2\}, \{G3\}, \{G4\}, \{G5\}, \{G6\}, \{G7\}\}$

It is easy to see all above partitions satisfy the properties (i) & (ii).

Furthermore, $P(g)$ is a bounded lattice $L(g)$ [13] whose order relation $\preccurlyeq$ is the inclusion relation given in definition (4). Fig.3 depicts the Hasse diagram of $P(Ga)$ for Example 3.

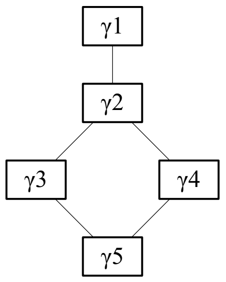

Fig. 3 The Hasse diagram of *P(AG)*

Different partitions in $P(g)$ can been viewed as different grain-sized solutions to achieve the goal *g*. Given two partitions $\gamma 1, \gamma 2$, if $\gamma 1 \preccurlyeq \gamma 2$, we say $\gamma 1$ is finer than $\gamma 2$, or $\gamma 2$ is coarser than $\gamma 1$. That is, $\gamma 1$ provides a more detailed view of requirements than $\gamma 2$. Some clients show no care for much detail, their requirements can be achieved in a coarsely granular way, whereas other clients prefer involving into the details for certain purpose. Therefore, various granularities provide both coarse and detailed views to satisfy different users.

III. Deliver Services over Imprecise Requirements

Since clients (users or intelligent agents) are not the domain experts, with poor domain knowledge, they always submit a very imprecise requirement. Even if services are retrieved according to such imprecise requirement, actually these services cannot satisfy the goals that clients really want to achieve. Hence, it is necessary to help clients clarify their requirement iteratively, adding goals they really desire to achieve and removing goals they do not want to involve.

A. Stratified Rough Set Approximation Space

In Pawlak's rough set model, the partition induced by an equivalence relation *R* on universe *U* is denoted as *U/R= {C1, C2, . . ., Cn}*, where *Ci* is an equivalence class of *R*. Let an arbitrary subset *X* of *U*. Every rough set we associate two crisp sets, called lower and upper approximation.

The lower approximation of X:

$\underline{apr}\,(X)\ =\ \{x\ \in\ Ci\ |\ Ci\ \subseteq\ X\}$ (5)

The upper approximation of X:

$\overline{apr}(X)\ =\ \{x\ \in\ Ci\ |Ci\ \cap\ X \neq\ \emptyset\}$ (6)

The boundary region of X:

$bnd(X) =\ \overline{apr}(X) -\ \underline{apr}\,(X)$ (7)

Intuitively, the lower approximation of a set consists of all elements that definitely belong to the set, whereas the upper approximation of the set constitutes of all elements that possibly belong to the set, and the boundary region of the set consists of all elements that cannot be classified uniquely to the set or its complement, by employing available knowledge [5].

Dubois and Prade [14] defined a rough set as a pair of subsets of *U/R*, and the pair of approximation is given:

$\underline{apx}\,(X)\ =\ \{\ Ci\ |\ Ci\ \subseteq\ X\}$ (8)

$\overline{apr}(X)\ =\ \{Ci\ |Ci\ \cap\ X \neq\ \emptyset\}$ (9)

The pair of approximation may be viewed as extensions of Pawlak's lower and upper approximations. Actually, they are consistent with each other [15]. Every Ci can be seen as a granule.

As discussed in section 2.2, the partitions in *P(G)* form a lattice granulation structure *L(G)*. With a given subset *X*, we can use equation (8) (9) to compute lower and upper approximations for each partition in *P(G)*. Further, the stratified rough set approximations can be produced from *L(G)*.

Example 4, consider *P(G)* given in Example 3 and the corresponding lattice *L(G)* illustrated in Fig.3. Give a subset *X={G3,G5,G6}*, then the stratified rough set approximations are depicted in Fig.4.

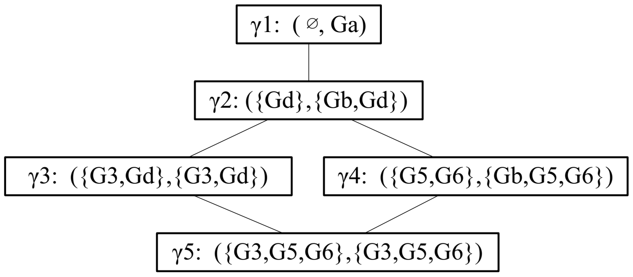

Fig. 4 The stratified rough set approximations of set *{G3,G5,G6}*

B. Clarify User Requirements in Stratified Rough Set Approximations Space

The goal model is often constructed by domain experts to formalize domain requirements specification. Usually, a user requirement in some domain may be represented by a part of goals in a goal model. For example, given the goal model of domain "*Travel Planning*" as depicted in Fig.1, user requirement can be given as: {*G1:(Book, Flight Ticket), G2:(Arrange, Accommodation)*}.

However, in fact, the goal model is often much more complex than this example. Due to a client not being the domain experts, he has less knowledge of that domain. Initially, he can only give a very imprecise requirement which does not cover all goals he wants to achieve. Consequently, we should recommend some goals, which are possible desired by clients, to help clarifying their requirements. Rough set model is a tool used to handle imprecision and uncertainty. Our proposal is try to refine user requirements in the stratified rough approximation spaces discussed previously.

The accuracy of rough set approximation is defined as [4]:

$$\alpha(X) = |\underline{apr}(X)|/|\overline{apr}(X)| \qquad (10)$$

where $|\cdot|$ denotes the cardinality of a set. Obviously, $0 \leq \alpha(X) \leq 1$. If $\alpha(X) = 1$ then X is called definable ($\underline{apr}(X) = X = \overline{apr}(X)$), and otherwise, if $\alpha(X) < 1$ then *X* is indefinable or rough ($\underline{apr}(X) \subset X \subset \overline{apr}(X)$).

According to equation (1), if we denote G as the universe for all granules in a goal model, with a given goal set *X*, rough set approximations can be used to measure the degree of *X* satisfying the requirements. If we regard *Ci* in equation (5), (6), (8), (9) as a granule in *G*, $Ci \subseteq X$ represents that *X* can fully satisfy *Ci,* whereas $Ci \cap X \neq \emptyset$ represents that *X* can partially satisfy *Ci.*

The user requirement *R* can be clarified through four kinds of operations: (1) refine *R* into finer granules; (2) generalize *R* into coarser granules; (3) add desired goals into *R*; (4) remove undesired goals from *R*.

We demonstrate our method the with the goal model illustrated by Fig. 2. If the initial user requirements are formalized as goals *R*:*{Gb,G5},* then we can obtain the stratified rough set approximations and corresponding accuracy as shown in Table 1.

TABLE 1. STRATIFIED ROUGH SET APPROXIMATIONS FOR PARTITIONS

Partition	$\overline{apr}$	$\underline{apx}$	Bnd	α
$\gamma 1$	*{Ga}*	$\emptyset$	*{Ga}*	*0*
$\gamma 2$	*{Gb,Gd}*	*{Gb}*	*{Gd}*	*4/6*
$\gamma 3$	*{G1,G2,G3, G4,Gd}*	*{G1,G2,G3, G4}*	*{Gd}*	*4/6*
$\gamma 4$	*{Gb,G5}*	*{Gb,G5}*	$\emptyset$	*1*
$\gamma 5$	*{G1,G2,G3, G4,G5}*	*{G1,G2,G3, G4,G5}*	$\emptyset$	*1*

In Table 1, the accuracy values indicate certainty degrees of the user requirement in the stratified rough set approximations. The approximations with the accuracy value 1 are definable and they all have a pair of equal approximations. If we denote the approximation induced by partition γi as $\mathrm{apr}_{\gamma i}$, where $\mathrm{apr}_{\gamma i}(X) = \underline{apr}_{\gamma i}(X) = \overline{apr}_{\gamma i}(X)$, then the set $apr_{\alpha=1}(X)$ is also a bounded lattice, where:

$$apr_{\alpha=1}(X) = \left\{apr_{\gamma i} \middle| \underline{apr}_{\gamma i}(X) = \overline{apr}_{\gamma i}(X)\right\} \qquad (11)$$

and the order relation $\preccurlyeq$ is the inclusion defined in (4). That is, the least element in the lattice implies that the user requirement should be satisfied in the most detailed way. In contrast, the greatest element implies that the user requirement should involve the least details. A client can choose a proper grain-sized $\mathrm{apr}_{\gamma i}$ from $apr_{\alpha=1}(X)$ to describe his requirement. For example, according to Table 1, if a client chooses $apr_{\gamma 5}$, it indicates that he is willing to involve more details comparing with $apr_{\gamma 4}$. Choosing different $apr_{\gamma i}$ can be viewed as clarifying user requirement through *operation (1)* or *(2)*. When a client has selected a $\mathrm{apr}_{\gamma i}$ from $apr_{\alpha=1}(X)$, the new user requirement *R* was assigned as $\mathrm{apr}_{\gamma i}$. Moreover, a client may do the *operation (4)* to remove some goals that he does not desire from *R*.

The approximations with the accuracy value less than 1 may be more interesting, because the granules in boundary region contain some goals in user requirements. Since elements in the same granule are highly correlated, those granules in boundary region most potentially desired by clients but they have not given them in *R*. Therefore, such goals in these granules are best candidates for recommendation.

These recommendations are ranked by the accuracy (not including approximations with accuracy value 1) in a descendent order, because the higher the accuracy is, the less modification is required to make user requirements more complete.

Example 5, if the user requirement *R:{G2,G3,G4,G5}* and the corresponding stratified rough set approximations are give in Table 1. List the recommendations:
The recommended granules order by α are:
1. $\{Gd\}_{\alpha=4/6}$ *2.*$\{Gd\}_{\alpha=4/6}$ *3.* $\{Ga\}_{\alpha=0}$
Remove duplicated granules and granules have existed in *R*. Following gives the final recommendations:

Rank	***Recommendation***	***Comment***
1	*G7*	Elements in *Gd* and remove *G5* existed in *R*
2	*Gb,Gc,Gd*	Elements in *Ga*

Clients adding desired goals from recommendations can be viewed as clarify *R* through *operation (4)*. It should be noted that the added granules possibly absorb some small granules which are included in the new added ones. For example, if a client adds *Gd* into *R*: *{G2,G3,G4,G5}*, *G5* is absorbed by *Gd* because *G5* is contained in *Gd*. The new *R* will be $\{G2,G3,G4,Gd\}$.

When clients do any operation described above, we get a more clarified user requirement *R*' from *R*. If *R*' is still not clear, we can compute the approximations, boundary and accuracy based on *R*' and clarify it through above 4 kinds operations. Such process makes user requirements to be clarified iteratively, which can be stopped any time if clients no longer want to do any modification. We denote the clarified user requirement as $\mathbb{R}$, and it is supposed to find services to satisfy $\mathbb{R}$.

C. Services Annotation and Retrieval

1) Advertise Services with Granules: Services can be executed to achieve some goals. Any service may be annotated with these goals to advertise its capabilities. As discussed previously, a goal model can be formalized according to requirements specification for a given domain. Correspondingly, the granules space G of a goal model can be given as defined by equation (1). A subset of G should be selected as the advertisement for every service, because further service retrieval bases on the matching between advertisements and user requests. The advertisements annotation for services can be done manually or by some semi-automatic tools [16].

A service *S* may achieve multiple goals by given different parameters. If we denote the advertisement set of *S* under all parameters as *A(S)={A1,...An}*, then for each *Ai* in *A(S)* can be viewed the advisement under a certain parameter, in which $Ai \in 2^G$. That is, every *Ai* is a granule, and the service can only achieve all the goals in *Ai* as a whole instead of achieving any of them separately. Following example is provided for intuitive understanding.

Example 6, travel agencies usually arrange tour for customer as a package including transportation and accommodation. They hardly sell flight tickets or hotel rooms individually.
Given the goal model shown in Fig. 1, the service S provided by travel agencies is advertised as: *A(S)={A1}*, where *A1={G1,G2}*, $G1 \overset{\text{def}}{=} (Arrange, Transportation)$ and $G2 \overset{\text{def}}{=} (Arrange, Accommodation)$. It implies that the execution of *S* will achieve *{G1,G2}* as a whole. You cannot achieve the goal *G1* or *G2* separately by the service.
If a travel agency sells *G1*,*G2* separately in addition to *A1*, the advertisement will be:
A(S)={A1,A2,A3}
where *A1={G1,G2}, A2={G1}* and *A3={G2}*

2) Match Services in Granule Approximations: Service discovery is to find a collection of services which capabilities can satisfy user requirements. In the granular computing view, the process of service discovery is to select a set of services whose advertisements can cover all granules in the clarified user requirements $\mathbb{R}$.

Since any two granules in $\mathbb{R}$ are disjoint, the matching process may be viewed as finding services to satisfy each granule g in $\mathbb{R}$ respectively. The matching algorithm is described in detail as following:

ALGORITHM 1. SERVICES MATCHING FOR GRANULE *g*

```
exactMatchedList = ∅;
/*to store exact matched goals & services*/

pluginMatchedList = ∅;
/*to store target goals, matched advertisement & services*/

unmatchedList = ∅;
/*to store unmatched goals*/

/****Main Procedure****/
if ExactMatching(g) = true then
    matchDegree=Exact;
else if WeakExactMatching(g) = true then
   matchDegree=WeakExact;
else if PlugInMatching (g) = true then
   matchDegree= PlugIn;
else
   WeakPlugInMatching();
   if unmatchedList = ∅ then
      matchDegree= WeakPlugin;
   else if (exactMatchedList≠ ∅ or
            pluginMatchedList≠ ∅) then
      matchDegree= Subsumes;
```

```
  else
     matchDegree= Fail;
  end if
end if

/****Sub Procedures****/
ExactMatching(g)
{
if exists a service S,
$(\exists Ai \in A(S))(\{g\} = Ai)$ then
    exactMatchedList.append(g, S);
    return true;
end if
 return false;
}

WeakExactMatching(g)
{
 if g is not the atomic goal
   for each x in $\omega(g)$
   /*$\omega(g)$ is the zooming-in operation defined in (2)*/
      if not ExactMatching(x) then
        WeakExactMatching(x);
      end if
   end for
else
     unmatchedList.append(x);
end if
}

PlugInMatching(g)
{
if exists a service S,
$(\exists Ai \in A(S))(\{g\} \subset Ai)$ then
/*$\{g\}, Ai \in 2^G$,
$\subset$ relation can be judged by the definition (4)*/
    pluginMatchedList.append(g, Ai, S);
    return true;
end if
return true;
}

WeakPlugInMatching ()
{
for each x in unmatchedList
    if PlugInMatching (x)=true then
        unmatchedList.remove(x);
    end if
end for
}
```

Above algorithm can be explained intuitively in granule approximations. These approximations can be classified into six types: {***Exact, WeakExact, Plugin, WeakPlugin, Subsumes, Fail***}. We borrow such measurements from the classic service matching algorithm [3] and add two additional metrics to describe matching degree.

Exact. This is the perfect way to satisfy the granule g in user requirement. That is, we can find a service S, one of the advertisements Ai in *A(S)* is equal to *{g}*.

WeakExact. Usually, we cannot always find a service perfect matching the requirement g. Since g can be decomposed into finer granules, we try to find a set of services satisfying each finer granule exactly. If there some granules still cannot be exactly matched, they would be zoomed in recursively up to atomic goals. Actually, *{g}* is the greatest element in the bounded lattice *L(g)* (see 2.2). *WeakExact* matching is to find a partition $\gamma \preccurlyeq \{g\}$, and every element in γ is exactly matched.

Plugin. Given a service S, Ai is an advertisement in *A(S)*. If $\{g\} \subseteq Ai$ (see the definition (4) in 2.2), we call it as *Plugin* matching.

WeakPlugin. If *WeakExact* matching fails, $\{g\}$ has been refined as a partition γ in which all unmatched granules belong to *A(g)*, i.e. these granules cannot be zoomed in any more. Then we try to find a *Plugin* matching for these unmatched granules.

Subsumes. After *WeakPlugin* matching, if some granules are still unmatched, we say g can only be satisfied partially and call thus matching is *Subsumes*.

Fail. If nothing matches.

This algorithm produces three lists: *exactMatchedList*, *pluginMatchedList* and *unmatchedList*. In which, *unmatchedList* is used for exception handling for these unsatisfied goals; *pluginMatchedList* can be used for extra cost estimation for further decision. If we denote $\delta(g)$ is the cost function (cost may be money, time, etc.) to compute the cost for achieving g, then extra cost can be estimated:

$$\Delta = \sum_{x \epsilon P} \delta(x.Adv) - \delta(x.g)$$
$$\cong \sum_{x \epsilon P} \delta(x.Adv - \{x.g\}) \qquad (12)$$

where P represents *pluginMatchedList*, *Adv* represents *Ai* ***Plugin*** matching the target goal g ($\{g\} \subseteq Ai, Ai \in A(S)$, S is the matched service).

Other granules in requirements $\mathbb{R}$ can use the same algorithm to find services. Finally, we obtain a set of services satisfying each granule in $\mathbb{R}$

IV. An Illustrative Example

Let us consider the domain “Travel Planning”. If the goal model of this domain is depicted in Fig. 2, we obtain all granules:

G={Ga,Gb,Gc,Gd,G1,G2,G3,G4,G5}, where
Ga:{Plan, Travel},
Gb:{Arrange, Transportation },
Gc:{Arrange, Accommodation },

G1:{Book, FlightTicket},
G2:{Register, Luggage},
G3:{Arrange, LocalTranportaion},
G4:{Reserve, Hotel},
G5:{Rent, Apartment},
G6{Query, Weather}

If a user submits a request: "I want to book a flight ticket and reserve a hotel room", by some NLP method [16], it is formalized as the initial requirement, *R:{G1,G4}*.

Construct all possible partitions *P(Ga)* for the root *Ga*:
$\gamma 1: \{Ga\}$
$\gamma 2: \{Gb, Gc, G6\}$
$\gamma 3: \{G1, G2, G3, G4, Gc, G6\}$
$\gamma 4: \{Gb, G4, G5, G6\}$
$\gamma 5: \{G1, G2, G3, G4, G5, G6\}$

The corresponding order relation of the bounded lattice *L(Ga)* is: $\gamma 5 \subseteq \gamma 4, \gamma 3 \subseteq \gamma 2 \subseteq \gamma 1$

Now we can compute the stratified rough set approximations for the given requirement *R*:

Partition	$\overline{apr}$	$\underline{apx}$	Bnd	α
$\gamma 1$	*{Ga}*	$\emptyset$	*{Ga}*	*0*
$\gamma 2$	*{Gb,Gc}*	$\emptyset$	*{Gb,Gc}*	*0*
$\gamma 3$	*{G1,G4}*	*{G1,G4}*	$\emptyset$	*1*
$\gamma 4$	*{Gb,G4}*	*{G4}*	*{Gb}*	*1/4*
$\gamma 5$	*{G1,G4}*	*{G1,G4}*	$\emptyset$	*1*

The recommended granules order by α are:
1. $\{Gb\}_{\alpha=1/4}$
2. $\{Gb, Gc\}_{\alpha=0}$
3. $\{Ga\}_{\alpha=0}$

Add them into an ordered list: *{1:Gb, 2:Gc, 3:Ga}*. The duplicated recommended granules should be removed when adding to the list, such as adding *Gb* in the second time. Then, the recommendations will be returned to the user:

Rank	*Recommendation*	*Comment*
1	*G2,G3*	Elements in *Gb* and remove *G1* existed in *R*
2	*G5*	Elements in *Gc* and remove *G4* existed in *R*
3	*Gb,Gc,G6*	Elements in *Ga*

From the recommendations, the user finds that he also desires to achieve following goals not included in *R*:
G2:{Register, Luggage},
G3:{Arrange, LocalTranportaion}
G6{Query, Weather}

Besides, the user prefers renting an apartment to reserving a hotel room. So he removes *G4* and adds *G5*. The refined requirement becomes: *R:{G1,G2,G3,G5,G6}*.

Using this new *R,* the user continues to clarify his requirement in second round:

Partition	$\overline{apr}$	$\underline{apx}$	Bnd	α
$\gamma 1$	*{Ga}*	$\emptyset$	*{Ga}*	*0*
$\gamma 2$	*{Gb,Gc,G6}*	*{Gb,G6}*	*{Gc}*	*4/6*
$\gamma 3$	*{G1,G2,G3 ,Gc,G6}*	*{ G1,G2,G3, G6}*	*{Gc}*	*4/6*
$\gamma 4$	*{Gb,G5,G6}*	*{Gb,G5,G6}*	*{Gb}*	*1*
$\gamma 5$	*{G1G2,G3, G5,G5}*	*{G1G2,G3, G5,G5}*	$\emptyset$	*1*

In this round, the user no longer wants to do any change. Two definable approximations are obtained, which represent the clarified user requirement in different granularities: *{Gb,G5,G6}* and *{G1,G2,G3,G5,G6}*. If the user prefers that a service provider can arrange transportation for him as a whole, then *{Gb,G5,G6}* should be chosen as the final clarified requirement $\mathbb{R}$.

Assume all services and corresponding advertisements related to this domain are listed below:
S1: A(S1):{{Gb,Gc}}
S2: A(S2):{{G6}}
S3: A(S3):{{G1,G4}}
S4: A(S4):{{G3,G4},{G5}}
S5: A(S5):{{G2}}

For granule *Gb* in $\mathbb{R}$, it cannot find a service with a ***Exact*** matching degree. We zoom in *Gb* by the operation $\omega(Gb)$, then we obtain a finer granules set *{G1,G2,G3}*. $\{G2\} \in A(S5)$, so *S5* can satisfy *G2* exactly. $\{G1\} \subset A(S3)$, $\{G3\} \subset A(S4)$, hence *S3*, *S4* are ***Plugin*** matching for *G1* and *G3* respectively. Since all granules in $\omega(Gb)$ are matched, we can say *Gb* is *a* ***WeakPlugin*** matching with the retrieved services: *{G1:S3:Plugin, G2:S5:Exact, G3:S4:Plugin}*. Furthermore, it is easy to see *S1* is a ***Plugin*** matching of *Gb*. Then we have two solutions to satisfy *Gb*:

WeakPlugin Solution:
{G1:S3:Plugin, G2:S5:Exact, G3:S4:Plugin}

Plugin Solution:
{G1:S1:Plugin }

To determine which solution is better, we can quantify them by computing the extra cost (using the extra cost function defined in (12)) for those plug-in matched goals for each solution:

$$\Delta_{WeakPlugin} = \delta(\{G1, G4\} - \{G1\}) + \delta(\{G3, G4\} - \{G3\}) = 2\delta(\{G4\})$$
$$\Delta_{Plugin} = \delta(\{Gb, Gc\} - \{Gb\}) = \delta(\{Gc\})$$

The cost can be estimated the by historical data. Assume $2\delta(\{G4\}) > \delta(\{Gc\})$, namely $\Delta_{WeakPlugin} > \Delta_{Plugin}$, we can say that the ***Plugin*** solution is better than ***WeakPlugin*** one. Therefore, *S1* is the best candidate to achieve *Gb*.

Similarly, $\{G5\} \in A(S4)$, $\{G6\} \in A(S2)$, so *S4* and *S2* are ***Exact*** matching of *G5* and *G6* respectively.

Finally, services retrieved to satisfy $\mathbb{R}$ are returned to the user: *{Gb:{S1}, G5:{S4}, G6:{S2}}*

V. Conclusion

In this paper we pay no attention to improving the precision of service discovery with a given request, but try to help clients to clarify their initial imprecise requirement in a multi-layer goal model. Each goal in the goal model is regard as a granule, and these granules cover requirements in various grain sizes. According to such hierarchical granulation, granular computing and rough set theory are employed to handle uncertainties, which distinguishes what has been definable and what is still uncertain. The iterative clarification process is to do four kinds of operations in the stratified rough set approximation space and produce a clarified requirement with a client desired granularity at length. Finally, a matching algorithm is provided in basis of six types of granule approximations, which deals with the matching process in the granular computing view.

We believe that the combination of granular computing and rough set theory will lead to many new methodologies for problem solving in service computing.

Acknowledgements

The paper is partially supported by China National Science Foundation (Granted Number: 60873230, 61073021), Scientific Research Projects of STCSM (Granted Number 08JC1411700, 10511501503 and 10692101400), National High-Tech Research and Development Plan of China (Granted Number: 2006AA01A124) and Program for New Century Excellent Talents in University (Granted Number: NCET-08-0347).

References

[1] D. Martin*, et al.*, "OWL-S 1.2 Release," *Available at:* http://www.daml.org/services/owl-s/1.2, 2008.
[2] J. Farrell and H. Lausen, "Semantic annotations for WSDL and XML schema," *Available at:* http://www.w3.org/TR/sawsdl/, 2007.
[3] M. Paolucci*, et al.*, "Semantic matching of Web services capabilities," *Semantic Web - Iswc 2002,* vol. 2342, pp. 333-347, 2002.
[4] Z. Pawlak, *Rough sets: Theoretical aspects of reasoning about data*: Springer, 1991.
[5] Z. Pawlak and A. Skowron, "Rough Sets and Conflict Analysis," *E-Service Intelligence,* pp. 35-74, 2007.
[6] A. van Lamsweerde, "Goal-oriented requirements engineering: A guided tour," *Fifth Ieee International Symposium on Requirements Engineering, Proceedings,* pp. 249-262, 2000.
[7] L. Zadeh, "Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic," *Fuzzy sets and systems,* vol. 90, pp. 111-127, 1997.
[8] Y. Yao, "Granular computing: basic issues and possible solutions," 2000, pp. 186-189.
[9] A. Skowron and J. Stepaniuk, "Information granules: towards foundations of granular computing," *International Journal of Intelligent Systems,* vol. 16, pp. 57-85, 2001.
[10] L. Zadeh, "From computing with numbers to computing with words¡ªfrom manipulation of measurements to manipulation of perceptions," *Intelligent Systems and Soft Computing,* pp. 3-40, 2000.
[11] C. Rolland*, et al.*, "Guiding goal modeling using scenarios," *Ieee Transactions on Software Engineering,* vol. 24, pp. 1055-1071, Dec 1998.
[12] Y. Yao, "A partition model of granular computing," *Transactions on Rough Sets I,* pp. 232-253, 2004.
[13] Y. Y. Yao, "Information granulation and rough set approximation," *International Journal of Intelligent Systems,* vol. 16, pp. 87-104, Jan 2001.
[14] D. Dubois and H. Prade, "Rough Fuzzy Sets and Fuzzy Rough Sets," *International Journal of general systems,* vol. 17, pp. 191-209, 1990.
[15] Y. Yao, "Combination of rough and fuzzy sets based on a-level sets," *Rough sets and data mining: analysis for imprecise data,* pp. 301-321, 1997.
[16] L. Hu*, et al.*, "Modeling Semantic Web Service using Semantic Templates," 2008, pp. 165-172.