Goal Representation and Reasoning based on Description Logics(DLs)

Xiuguo Wu[1,2] Guangzhou Zeng[1] Fangxi Han[1] Rui Wang[1,2]

[1]School of Computer Science and Technology, Shandong University, Jinan 250014, P. R. China
[2]School of Information Management, Shandong University of Economic, Jinan 250014 , P. R. China

Abstract

Over the past decades, goal models have been used in Computer Science in order to represent business objective, design qualities and desirable states. The main merit of goal-driven systems is that they bridge the human and the machine level problem solving. This paper presents first steps towards the definition of goals based on Description Logics(DLs), which are able to represent structural knowledge in a formal and well-understood way. In particular, the paper proposes a goal planning algorithm to achieve an agent's goal, and the preliminary results demonstrate that our implementation provides a practical solution.

Keywords: Goal Representation, Goal Reasoning, Description Logics (DLs), Agent

1. Introduction

Over the past decades, the concept of goal has been used in many areas of Computer Science for quite some time. In AI planning problems, an agent is given a description of the environment and a set of admissible actions, and searches of a plan- i.e., a sequence of actions and intermediate sub-goals- which allows for achieving a final goal from a given initial state. More recently, goals have been used in Software Engineering to model early requirements and non-functional requirements for a software system[1]. In agent theory, goals are introduced to explain and specify an agent's(proactive) behavior. In this view, agents are assumed to have their own objectives, for the achievement of which they initiate behavior.

The use of an explicit representation of goals provides for added flexibility, as the fact that a goal represents a desired state. The abstraction of goal could be particularly useful and appropriate, especially when adopting the agent-oriented paradigm, which provides interesting abstractions related to autonomous entities for the development of software systems whose requirements are not entirely known at design time(e.g. when running in rapidly changing environments). At this point, the explicit representation of goals and the ability to reason about them play an important role in several requirements analysis and modeling techniques.

Actually, several authors have argued the importance of declarative representation of goals in agent deliberation processes, especially in dynamic environments. Among them, M.Winikoff[2] stated that "by omitting the declarative aspect of goals the ability to reason about goals is lost". What is actually lost is the ability to know if goals are impossible achieved, incompatible with other goals. M.Stollberg[3] has elaborated and defined a goal model, which distinguishes three goal types along with the distinction of goal templates and instances. P. Giorgini[1] presented a formal framework for reasoning with goal models, which not only use AND/OR goal relationships, but also allow more qualitative relationship, as well as contradictory situations. In conclusion, all these notions are structures built from the actions and therefore similar in nature to plans.

When considering the application of goal-oriented techniques to workflow, especially Migrate Workflow, seem to have a natural relation with goals. For example, if a client C wants to buy a book named 《Intelligent Agent and its Application》 on the Internet. He has to hunt for this book in each of web site. Once finding it, he needs to pay for the book; transact the book. But in a goal-oriented migrating workflow system, what he needs to do is telling the agent "what he wants", then all the rest affairs will be accomplished automatically.

In this paper, we focus in particular on the representation and reasoning of goals based on Description Logics(DLs) and AI planning technologies. Our algorithm for goal planning uses backward-chaining search method to find potential candidate goal. The plans are stored in queue and can be reuseful.

The remainder of this paper is structured as follows. Section 2 provides a simple description on goal and Description Logics(DLs). in Section 3 we present our definition on goals based on Description Logics(DLs). In Section 4, we propose a method to check goal consistency and give a recursive algorithm

about goal planning. In the final section we draw some conclusions and discuss future work.

2. Preliminaries

2.1. Goals

A Goal represents a formal, machine-processable knowledge level specification of a client objective that needs to satisfy the following requirements:

- Abstracting from technical details to the highest possible extent;
- Support all possible kind of objectives that clients may have;
- Carry all information needed for automated resolution.

In common sense, and in most logics of agents, the goals of an agent describe the state of affairs an agent would like to be realized. These goals are declarative in nature, and are also called goals-to-be.

Moreover, there is a second type goals-procedural notion of goal. These goals are also called goals-to-do, because they specify a plan of action the agent is intending to execute. Goals in this sense are a kind of imperative program, which given a set of simple goals can be composed into more complex goals by means of the usual operators from imperative programming.

First, we introduce basic actions, which constitute one of the simple goals. Basic actions specify the capabilities which an agent has to achieve a particular state of affairs. It is important in this context to emphasis that an agent is viewed as a mental entity. Basic actions are actions which update or change the beliefs of an agent. This is most natural, since these updates change the representation of the environment the agent is supposed to control by means of performing actions.

There are two other simple goals, i.e. achievement goals and test goals. Achievement goals are atomic propositions from the logical language *L*. Test goals allow an agent to introspect its beliefs. A test is evaluated relative to the current beliefs of an agent. Their main use, however, is not just to check whether or not the agent believes a particular proposition, but to compute values or bindings for the free variables which occur in the test.

Together, the basic actions, achievement goals, and the test goals are the basic goals in agent description. The complex goals are composed from basic goals by using the programming constructs for sequential composition and nondeterministic choice. An agent also may have goals which are executed in parallel, which is not explicitly represented by some operators, but as a result of the fact that an agent may have more than one goal at the same time.

Definition 2.1 (goals) Let Bact be a set of basic actions, Atom be a set of atom goals, then the set of Goal is inductively defined by:

(1) Bact $\subseteq$ Goal;
(2) Atom $\subseteq$ Goal;
(3) if g$\in$Goal, then g? $\in$Goal;
(4) if g_1,$g_2\in$Goal, then$(g_1;g_2)$,$(g_1+g_2)\in$Goal.

2.2. Description Logics

This section gives a brief introduction to Description Logics(DLs) and discusses why it is suitable as a representation framework of goals[4][15].

Description Logics(DLs) are a family of knowledge representation languages that are able to represent structural knowledge in a formal and well-understood way. A description logic system consists of four parts: constructors which represent concept and role, Terminological assertion(TBox) subsumption assertion, Assertions about individual(ABox) instance assertion, and reasoning mechanism of TBox and ABox.

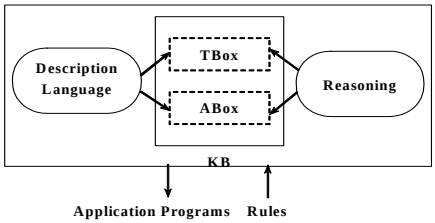

Fig. 1: Architecture of a Knowledge Representation System based on Description Logics.

The constructors determine the expressive power of the Description logics. Given mutually disjoint sets N_C of concept names, N_R of role names, and N_I of individual names, concept and role constructors in ALC, can be defined using the following syntax:

C,D$\rightarrow$A|	atomic concept
T|	universial concept
$\perp$|	bottom concept
$\neg$A|	atom negation
C$\sqcap$D|	intersection
$\forall$R.C|	value restriction
$\exists$R.C|	limited existential quantification

Description Logics commonly have a set-theoretic semantics. The semantics of a Description Logic knowledge base are given via interpretation $I = (\triangle^I, \bullet^I)$, where $\triangle^I$ is a non-empty set of objects, and $\bullet^I$ maps:

- each individual name a to an element $a^I\in\triangle^I$;

- each concept name C to a subset of $\triangle^I$, i.e., $C^I \subseteq \triangle^I$;
- each role name R to a binary relation on $\triangle^I$, i.e., $R^I \subseteq \triangle^I \times \triangle^I$.

A concept definition is an identity of the form:

$$A \equiv C$$

where A is a concept name and C an ALC-concept. A TBox T is a finite set of concept definitions with unique left-hand side. Concept names occurring on the left-hand side of a definition of T are called defined in T whereas the others are called primitive in T. The TBox T is acyclic iff there are no cyclic dependencies between the definitions, i.e., the recursive substitution of defined concepts by their definitions always terminates. This process is called expansion of the TBox.

The semantics of TBox definitions is defined in the obvious way: the interpretation I is a model of the TBox T iff it satisfies all its definitions, i.e.,

$A^I=C^I$ holds for all $A \equiv C$ in T

Any interpretation of the primitive concepts and of the role names can uniquely be extended to a model of the acyclic TBox T. This is an easy consequence of the fact that acyclic TBox can be expanded.

An ABox assertion is of the form:

C(a), R(a, b), or ¬R(a, b)

Where a, b $\in N_I$, C is a concept, and R a role name. To improve readability, we will sometimes write the assertion C(a) in the form a: C. An ABox is a finite set of ABox assertions. The interpretation I is a model of the ABox A iff it satisfies all its assertions, i.e.,$a^I \in C^I((a^I, b^I) \in R^I, \neg(a^I, b^I) \notin R^I)$ for all assertions C(a)(R(a, b), ¬R(a, b)) in A. if φ is an assertion, then we write $I \models \varphi$ iff I satisfies φ.

Various reasoning problems are considered for Description Logics(DLs). For the purpose of this paper, it suffices to introduce concept satisfiability and ABox consistency:

- the concept C is satisfiable with respect to the TBox T iff there exists a model I of T such that $C^I \neq \varnothing$;
- the ABox A is consistent with respect to the TBox T iff there exists an interpretation I that is a model of both T and A.

Several reasons exist to use Description Logics(DLs) for goal representation and reasoning. The first one is that Description Logics(DLs) is decidable, i.e. given a concept definition it is possible to determine if this definition is consistent with others. Also given an instance definition, it can be decided which is the concept definition that most suits it. The next reason is that Description Logics(DLs) has sound and complete reasoning mechanisms which guarantee the results accuracy and reliability. Finally, wide range of logics has being developed till now, so we can choose which suits our needs by least computational complexity. Besides these, we must consider that modern Description Logics(DLs) reasoners are quite efficient which let the result of model checking be available in a reasonable time.

3. Goal Representation and Semantics based on Description Logics(DLs)

From above, we know that Description Logics(DLs) have many advantages such as sound and complete reasoning mechanisms which guarantees the results accuracy and reliability. We now introduce the formalism for representation about goals. For simplicity, we concentrate on ground goals, the complex goals can be handled in the same way.[5]-[7]

3.1. Goal Representation based on Description Logics(DLs)

Definition 3.1 (Condition) Let N_C be a set of individual names, N_X a set of individual variables, and L a description logic. We use N_I as an abbreviation for $N_X \cup N_C$. A condition is an expression of the form:

$\forall$C, C(p), R(p, q), p=q, p≠q

for a, b$\in N_I$, C an L-concept and R a possibly negated L-role. [14]

Definition 3.2 (Goal) Let T be an acyclic TBox. An atomic goal for T can be defined in the from of g= < *pre*, *post* > ,where:

- *g* is the name of goal;
- *pre* is the pre-condition of the goal, which must be satisfied before the goal is achieved. *pre* is defined as a set of conditions.
- *post* is finite set of conditional post-conditions, which denote the effects of the goal. *post* is a set of pair φ/ψ ,where φ and ψ are an ABox assertions for T, and defined as a set of conditions.

A composite goal for T is a finite sequence g_1, g_2, …, g_k of atom goal for T. A goal is a composite or an atomic goal.

Intuitively, the pre-conditions specify under which conditions the goal is applicable. The conditional post-condition φ/ψ say that, if φ is true before executing the goal, then ψ should be true afterwards. If φ is tautological, e.g. $\top$ (a) for some individual name a, then we write just ψ instead of φ/ψ. By the law of inertia, only those facts that are forced to change by the post-conditions should be changed by applying the goal.

To illustrate the definition of goal, consider a goal "planning travel", which include four sub-goals:(1)g_1:Applying for visa; (2)g_2:Obtaining the

ticket; (3)g_3:Reserving hotel; (4)g_4:Reserving the transportation.

If we know that the pre-condition of applying for visa is in possession of passport of that country, then the sub-goal g_1 can be represented as follows:

pre_1= {Eligible(a),∃Has.Valid(a, b), Person(a), passport(b)}

$post_1$= {Holds(a, c), Person(a), Visa(c)}

Suppose that one can obtain the ticket if one has enough money in off-season, then the sub-goal g_2 can be represented as follows:

pre_2= {∃Holds.Enough(b), Money(b)}

$post_2$= {Receive(a, b), Person(a), Ticket(b)}

3.2. Semantics of Goals

To define the semantics of goals, we must first define how the application of an atomic goal changes the world, i.e., how it transforms a given interpretation I into a new on I′.

The formal semantics of goals can be defined by means of a transition relation on interpretations. The goal g may transform I to I′($I \Rightarrow_{g}^{T,A} I'$), if C is a primitive concept and R a role name then

- $C^{I'}$:=($C^I \cup \{c^I | \varphi / C(c) \in post$, and $I \models \varphi\} \setminus \{ c^I | \varphi / \neg C(c) \in post$, and $I \models \varphi\}$)
- $R^{I'}$:=($R^I \cup \{(a^I,b^I) | \varphi / R(a,b) \in post$, and $I \models \varphi\} \setminus \{ (a^I,b^I) | \varphi / \neg R(a,b) \in post$, and $I \models \varphi\}$)

Then we present executability and projection of goals as follows:

Definition 3.3 (Executability and Projection**)** Let T be an acyclic TBox, $g_1,\ldots, g_k$ a goal for T with g=(pre_i, $post_i$) and A an ABox.

- Executability:$g_1,\ldots,g_k$ is executable in A with respect to T iff the following conditions are true in all models I of A and T:

  ⑴ $I \models pre_1$ and

  ⑵ for all I with $1 \le i \le k$ and all interpretation I′ with $I \Rightarrow_{g_1,\ldots,g_k}^{T,A} I'$, we have $I' \models pre_1$
- Projection: An assertion φ is a consequence of applying $g_1,\ldots,g_k$ in A with respect to T, iff for all models I of A and T, and all I′ with $I \Rightarrow_{g1,\ldots,gk}^{T,A} I''$,we have $I' \models \varphi$.

Note that executability alone does not guarantee that we cannot get stuck while executing a composite goal. It may also happen that the goal to be achieved is inconsistent with the current interpretation.

3.3. Relationship among Goals

There exists relationship between different goals when considering goal composition. In this section, we categorize goals into following relationships.

Let T be an acyclic TBox, A be an ABox, and goal g_i and g_j be sub-goals of the composite goals while g_i different from the goal g_j. The relationship R between sub-goals g_i and g_j can be identified as follows:

⑴ Independent Relationship:

If I be *any* models of T and A, and there exists another models I' of T sharing the same domain and interpretation of all individual names. Then g_i and g_j is independent iff

① if there exists I′ such that $I \Rightarrow_{g_i;g_j}^{T,A} I'$, then $I \Rightarrow_{g_j;g_i}^{T,A} I'$;

② if there exists I′ such that $I \Rightarrow_{g_j;g_i}^{T,A} I''$, then $I \Rightarrow_{g_i;g_j}^{T,A} I''$.

In this case, sub-goal is freely independent of other and the order of execution of these two sub-goals does not affect the composition goal.

For example, sub-goals g_3: Reserving hotel and sub-goals g_4: Reserving the transportation are independent, and have no special order.

⑵ Equal Relationship

If I be *any* models of T and A, and there exists another models I′ of T sharing the same domain and interpretation of all individual names. Then g_i and g_j are equal iff

① if there exist I′ such that $I \Rightarrow_{g_i}^{T,A} I'$, then $I \Rightarrow_{g_j}^{T,A} I'$;

② if there exist I″ such that $I \Rightarrow_{g_j}^{T,A} I''$, then $I \Rightarrow_{g_i}^{T,A} I''$.

If g_i and g_j are equal, it means that the two sub-goals seem to provide the same result but they have some different attributes.

⑶ Weakly Equal Relationship

If I be a *certain* models of T and A, and there exists another models I′ of T sharing the same domain and interpretation of all individual names. Then g_i and g_j are weakly equal iff

① if there exists I′ such that $I \Rightarrow_{g_i}^{T,A} I'$, then $I \Rightarrow_{g_j}^{T,A} I'$;

② if there exists I″ such that $I \Rightarrow_{g_j}^{T,A} I''$, then $I \Rightarrow_{g_i}^{T,A} I''$.

If g_i and g_j are weakly equal, it means that a sub-goal g_i can provide the same function as g_j in *some* situation.

⑷ Substitutable Relationship

If I be *any* models of T and A, and there exists another models I′ of T sharing the same domain and interpretation of all individual names. Then g_i and g_j are substitutable iff

① if there exists I′ such that $I \Rightarrow {}^{T,A}_{g_i} I'$, then there must exist I" such that $I \Rightarrow {}^{T,A}_{g_j} I'$;

② I' ⊆ I".

If g_i and g_j are substitutable, then it means a sub-goal g_i can be substituted by sub-goals g_j in any case.

⑸ Weakly Substitutable Relationship

If I be a *certain* models of T and A, and there exists another models I′ of T sharing the same domain and interpretation of all individual names. Then g_i and g_j are weakly substitutable iff

① if there exists I′ such that $I \Rightarrow {}^{T,A}_{g_i} I'$ then there must exist I" such that $I \Rightarrow {}^{T,A}_{g_j} I''$;

② I' ⊆ I".

If g_i and g_j are weakly substitutable, then it means a sub-goal g_i can be substituted by sub-goals g_j in *some* situation.

⑹ Prerequisite Relationship

If I be a certain models of T and A in which g_j is executable, and there exists I(I ≠ I′), such that $I' \Rightarrow {}^{g} I$, then there must have g=g_i.

If g_i and g_j are prerequisite, it means that one goal has to finish before the other starts, g_i has to be finished before g_j starts.

From the analysis, we can deduce that:

⑴ If g_i and g_j are equal, then they must be substitutable;

⑵If g_i and g_j are weakly equal, then they must be weakly substitutable in the same pre-condition.

4. Goal Reasoning based on Description Logics(DLs)

Assume that we want to apply a composite goal g_1, g_2, …, g_k for the acyclic TBox T. Usually, we do not have complete information about the world(i.e., the model I of T is not known completely). All we know are some facts about this world, i.e., we have an ABox A, and all models of A together with T are considered to be possible states of the world. The central problem is how to compose services based on AI planning methods, which means the reasoning problem. Firstly, we propose the algorithm about goal consistency in order to avoid the conflict in Agent's goal. Secondly, we give the goal planning method for finding goal execution path.

4.1. Goal Consistency

In general, the goals that assign to the agent are not always consistent. As a result, a set of conflict goals will not achieved for ever by the agent. Before trying to execute goals, we want to know whether it is consistent, which is the basic inference problem considered in the reasoning about Description Logics(DLs). For example, if an agent has such goals: g_1=Arrive(Beijing) ⊓ g_2=Leave(Beijing), it is easy to see that the goal $g_1 \sqcap g_2$ is conflict.

Definition 4.1(Conflict Goals) A goal formula α is conflict iff it has such forms:

⑴ {⊥(a)}

⑵ {C (a), C (a)}

⑶ {R (a, b), ¬R (a, b)}

where a, b an individuals, C an concept and R a role.

Definition 4.2(Consistent Goals) A goal formula α is consistent iff there is no conflict in α, or else it is not consistent.

In the next, we will give the algorithm checking the consistency in a goal formula, which mainly use the axiom in Description Logics(DLs) to extend the original formula. If a goal formula does not have standard form, which means all the negate sign do not appear in the front of atom goal, then we can convert into standard form using following rules:

¬(α⊓β)⇔¬α⊔¬β

¬(α⊔β)⇔¬α⊓¬β

Algorithm use the next steps to extend and check the consistency of the goal formula α:

Step1. if C(x) ∈α, and ∀x(C(x)→D(x)), then add D(x) to α;

Step2. using the following rules to extend α, until there is no rules to adopt:

⑴⊓-rule if $C_1 \sqcap C_2(x) \in \alpha$, and $C_1(x) \notin \alpha$, $C_2(x) \notin \alpha$, then $\alpha = \alpha \cup \{C_1(x), C_2(x)\}$;

⑵⊔-rule if $C_1 \sqcup C_2(x) \in \alpha$, and $C_1(x) \notin \alpha$, $C_2(x) \notin \alpha$, then $\alpha = \alpha \cup D(x)$, where $D = C_1$ or $D = C_2$;

⑶∃-rule if ∃R.C(x)∈α, and there is no y such that R(x, y), and C(y) ∈α then $\alpha = \alpha \cup \{C(y), R(x, y)\}$;

⑷∀-rule if ∀R.C(x)∈α, R(x, y)∈α, and C(y)∉α, then $\alpha = \alpha \cup C(y)\}$;

Step3. checking the conflict in α, α is consistent if there is no conflict; else α is not. The algorithm is end. ■

In the previous steps, the "⑵⊔-rule" in step2 is indeterminable, and may have two branches α_1 and α_2. Once this rule is used many times, we will get a finite set $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$. At this point, the formula is consistent if one of sub-goals $\alpha_1, \alpha_2, \ldots, \alpha_n$ does not include conflict; if all the sub-goals $\alpha_1, \alpha_2, \ldots, \alpha_n$ include conflict, then α is not consistent. In the practical application, once a conflict appears in any sub-goals α_k, we can throw it away.

Lemma 1. The problem to check consistency of goal formula α is determinant.

Proof. The main task to check the consistency of goal formula α is look for conflict in formula α. A formula can be extended using all the rules, what we

need do is to show that the algorithm is terminable in each step.

⑴In the first step, we use axioms to extend the formula, which is substitution on concept. So it can be done in polynomial time, and is determinant;

⑵ The second step correspond to four rules relative to the concept, one of which ⊔ is not determinant, but there may be binary-tree at worst, which means that it can be done in exponential time. So the second step can be done in exponential time at most.

From the analysis above, we can see that checking consistency of goal formula α is terminable, and finding the conflict is determinant. Consequently, the process of checking consistency is determinant.

4.2. Goal Planning

In practice, a goal is divided into many sub-goals, which can be implemented at one time. The executed order of sub-goals will be formed a chain. In this section, we will introduce how to make a goal into set of sub-goals.

Definition 4.3(Goal planning) A goal planning problem can be represented as a four-tuple: 〈T, A, G, g〉 , where,

T: the set of concept in the domain, describes the vocabulary of the application domain;

A: the set of assertion of including concept, also it define the initial state;

G: the set of all the realizable goal before;

g: a set of assertions, which represent the goal attempting to reach.

Essentially, goal panning problem is to decompose the goal g into set of sub-goals(g_1,…,g_k) for the sake of achieving the goal g. The planning procedure will not be finished until the agent finds a goal executable sequence.

It is important to arrange a right order for the execution of sub-goals, because the different executable order will not get the same result. Let g_1 and g_2 be the sub-goals of g, and g_1 and g_2 can be decomposed into {g_{11},…,g_{1i} } and { g_{21},…,g_{2j} } separately. We will give out the algorithm to judge the order relationship between goals g_1 and g_2. Let pre(g_i) denote the pre-condition of the goal g_i.

Let $F_i \equiv \exists g_{1k} \in g_1$ and model I′ such that $I \rightarrow^{g_{1i}} I'$, also $\exists g_{2m} \in g_2$ such that I′⊨ pre(g_{2m}).

Also, we let $F_j \equiv \exists g_{2m} \in g_2$ and model I' such that $I \rightarrow^{g_{2m}} I'$, also $\exists g_{1k} \in g_2$ such that I′⊨ pre(g_{1k}).

Then the procedure is as follows:

⑴ If $F_i \wedge F_j$ is true, then g_1 and g_2 can be executed in any order, they will not affect each other;

⑵ If $F_i \wedge \neg F_j$ is true, then g_1 must be executed before g_2;

⑶ If $\neg F_i \wedge F_j$ is true, then g_2 must be executed before g_1;

⑷ If $\neg F_i \wedge \neg F_j$ is true, then g_1 and g_2 are conflict goals, and can not be achieved at one time.

After arrange the order of goals, we can plan each sub-goals based on the order relationship. Many sub-goals will be produced in order to satisfy the pre-condition. When one sub-goal is planning completely, another subsequent sub-goal will be on. We will propose the planning algorithm on goal g. In algorithm 1, p is a data structure of queue.

Algorithm 1:Plan(T, A, G, g)

1: T is the TBox;
2: A is the ABox, add the initial state to A;
3: G is the set of all realizable goal G={g_1, g_2, …, g_n};
4: g is agent's Goal;
5: if T, A ⊨ g then
6: Return true;
7: else
8: Search for the pre-condition of goal g, each pre-condition is a sub-goal;
9: Computing the order among sub-goals;
10: Select the optimal planning of each sub-goals;
11: Combining time order to the sub-goals, and get a sequence g_1, …, g_n;
12: For π=g_1 to g_n do
13: p=Plan(T, A, G, π); //Planning each sub-goal
14: G=(G－Pre(g))∪Post(g);
15:Return p ■

The algorithm will be end with some sub-goals planned completely, which constitutes a leaf node of a planning-tree. Then p will be returned forming a sub-goal queue.

5. Conclusions and Future Work

Goals as used in agent programming describe situations that the agent wants to reach. The use of an explicit representation of goals provides much more flexibility in problem solving. The main technical result of this paper is that goal representation based on Description Logics(DLs) .Comparing to those existing works, we propose a method to represent goals and a algorithms for goal planning.

This is only a first proposal for a formalism describing the functionality of goals, which is widely used in agent, especially in mobile agent fields, which aims to provide a goal-oriented system. Various experiments and application have been undergoing in our current research. Future work includes extending learning plan scenarios during planning process based

on user's feedback and considering context modeling based on Description Logics.

Acknowledgement

Our work is partially supported by National Science Foundation of China (Grant No. 60573169).

References

[1] P. Giorgini, J. Mylopoulos, E. Nicchiarelli and R. Sebastiani, Reasoning with Goal Models. *Proceedings of 21st International Conference on Conceptual Modeling*, pp. 167-181, 2002.

[2] M. Winikoff, L. Padgham, J. Harland and J. Thangarajah, Declarative and Procedural Goals in Intelligent Agent Systems. *Proceedings of 8th International Conference on Principles of Knowledge Representation and Reasoning(KR2002)*, pp. 480-491, 2002

[3] M. Stollberg and M. Hepp, Goal Description Ontology, DIP, 2006

[4] F.Badder, C.Lutz, M.Milicic, U.Sattler and F.Wolter， A Description Logics based Approach to Reasoning about Web Services. *Proceedings of the WWW 2005 workshop on Web Service Semantics(WSS2005)*, 2005

[5] F.Lin, L.Qiu, H.Huang, Q.Yu and Z.Shi. Description Logic based Composition of Web Service. *Proceedings of the 9th Pacific Rim International Workshop on Multi-Agents.PRIMA*, pp.199-210,2006

[6] I.Horrocks, U. Sattler and S.Tobies. Practical reasoning for very expressive description logics. *Logic Journal of the IGPL.*8(3):239-264, 2000

[7] A.Schaerf. Reasoning with individuals in concept languages. *Proceedings of the Third Conference of the Italian Association for Artificial Intelligence*, pp.141-176, 1994

[8] U.Sattler, "Description Logics for the representation of Aggregated objects". *Proceedings of the 14th European Conference on Artificial Intelligence*. IOS Press,Amsterdam,2000

[9] G. Japaridze, The Logic of Tasks. *Annals of Pure and Applied Logic*.117(1-3):263-295, 2002

[10] F. Baader, L. Deborah, M. Guinness, D. Nardi. and F. Peter, The Description Logic Handbook: Theory, Implementation and Application. *Cambridge:Cambridge University Press*, 2002

[11] F. Baader, U. Scttler. An Overview of Tableau Algorithms for Description Logics. *Studia Logica*,69:5-40, 2001

[12] L. Qiu, F. Lin, C. Wan and Z.Shi, Semantic Web Services Composition Using AI Planning of Description Logics. *Proceedings of the 2006 IEEE Asia-Pacific Conference on Services Computing(APSCC'06)*.pp.340-347, 2006

[13] L. Qiu, Z. Shi, F. Lin and C. Liang, Agent-Based Automatic Composition of Semantic Web Services. *Journal of Computer Research and Development,* 44(4):643-650, 2007

[14] C. Lutz, U. Sattler, A Proposal for Describing Services with DLs. *Proceedings of the 2002 International Workshop on Description Logics, Aachen : CEUR-WS, 2002*. pp.129-140, 2002

[15] F. Wolter and M. Zakharyaschev, Dynamic description logic. *In Segerberg K, et al, ed. Advances in Modal Logic, Vol 2. Stanford: CSLI Publications*, pp.449-463, 2000.