

# Business domain oriented AI planning for web service composition

Dai Yu Yang Lei Zhang Bin

(College of Information Science and Engineering, Northeastern University, Shenyang 110004, China)

**Abstract:** In order to improve the effectiveness of the artificial intelligence (AI) planning based web service composition problem, a business domain oriented AI planning algorithm for web service composition is introduced. The algorithm uses the proposed business relation model to minimize the searching space and uses the proposed goal distance as a heuristic rule. Compared with other algorithms tackling web service composition such as AI planning, the proposed algorithm focuses not only on how to convert the web service composition problem to the AI planning problem, but also exerts its effort on how to apply the business feature of web service composition into planning in order to improve the effectiveness. The experiment shows the better performance of the proposed algorithm.

**Key words:** AI planning; web service; web service composition; business relation model

Web services<sup>[1]</sup> are rapidly emerging as a popular standard for sharing functionality among loosely coupled heterogeneous systems. Furthermore, there has been a considerable amount of recent work<sup>[2-6]</sup> on the challenges associated with composing web services to solve a given problem. The problem of web service composition is to find a course of services from a given initial state to a goal state. And several approaches of this field of research have emphasized how to compose automated web services. The space searching algorithm<sup>[7]</sup> can be used to solve this problem. However, the space searching problem is an NP-hard problem. Directly using it to solve a composition problem will be of low effectiveness.

For this reason, a business relation model is proposed in this paper which contains three parts: resource state-resource state relation, service-service relation and service-resource state relation. In order to integrate the business relation model with the composition problem, a heuristic algorithm is proposed. And the experiment shows that the proposed algorithm performs better.

## 1 Business Relation Model

Any interaction with a web service involves sending and receiving messages. Through sending and receiving messages, services can recognize the state of resources and can change the state of resources. Thus,

one way to describe the state of a system which is interacting with such services is in terms of resource states.

**Definition 1** (resource) A resource can be defined as a 3-tuple:

$$\text{Res} : = \langle \text{ID}, \text{StaticAttr}, \text{DynamicAttr} \rangle$$

where ID is the identification of the resource; StaticAttr is the set of static attributes of the resource; DynamicAttr is the set of dynamic attributes of the resource which signifies the state of the resource.

**Definition 2** (resource state) For resource  $r$ , at some time, its state can be defined as  $r. \text{da}_i. \text{name} = r. \text{da}_i. \text{value}$  and  $\text{da}_i \in \text{DynamicAttr}$ . The resource state can be signified as  $s$ .

According to OWL-S, a web service is specified by its name, its input and output messages, its precondition and its effect. The precondition and effect can be modeled as a set of the resource state.

**Definition 3** (web services) A web service can be modeled as

$$\text{WS} : = \langle \text{name}, \text{input}, \text{output}, \text{precondition}, \text{effect} \rangle$$

This paper classifies the business relation into three categories. We will discuss the details later.

Different services may deal with different resources. Thus, when composing these services into one, a common understanding between resource states must exist.

**Definition 4** (resource state-resource state relation) For resource  $r_i$  and  $r_j$ , relation  $\langle r_i. s_i, r_j. s_j \rangle \downarrow \langle r_i. \text{sa}_i, r_j. \text{sa}_j \rangle$  means that if static attribute  $\text{sa}_i$  of  $r_i$  is equal to static attribute  $\text{sa}_j$  of  $r_j$ , state  $s_i$  of  $r_i$  has the same meaning as state  $s_j$  of  $r_j$ .

Received 2007-05-18.

**Foundation item:** The National Key Technologies R & D Program of China during the 10th Five-Year Plan Period(No. 2004BA721A05).

**Biographies:** Dai Yu (1980—), female, graduate; Zhang Bin (corresponding author), male, doctor, professor, zhangbin@ise.neu.edu.cn.

During composition, there exists a set of domain knowledge which restricts the sequence of services.

**Definition 5** (service-service relation) For service  $a_i$  and  $a_j$ , relation  $\langle a_i, a_j \rangle$  means that service  $a_i$  occurs just before service  $a_j$ .

There exists a set of domain knowledge which restricts; that is, if state  $s$  holds, service  $a$  will occur.

**Definition 6** (service-resource state relation) For resource state  $s$  and service  $a$ , relation  $\langle s, a \rangle$  means that if  $s$  holds, service  $a$  will occur.

## 2 Business Relation Driven Web Service Composition

Before introducing the problem of web service composition, this paper first introduces the user requirements regarding composition. Based on the formal planning language EaGLE<sup>[6]</sup>, the composition requirement can be expressed as follows:

**Definition 7** (composition requirement) A composition requirement can be a 3-tuple:

$$\text{Req} ::= \langle S_0, S_F \rangle$$

where  $S_0$  is the initial resource state;  $S_F$  is the set of resource states which the user wants to achieve.

The space searching algorithm<sup>[7]</sup> can be used to solve the composition problem. The problem of space searching based planning can be divided into two phases: one is to find the appropriate node; the other is to find the path from the initial node to the final one. As for the problem of finding the appropriate node, there are business relations which can be used and which will limit the searching space; and for the problem of finding the path, some graph traversing algorithm can be applied. In this paper, we use the deepness-first traversing algorithm, and for a goal distance based, the heuristic algorithm is applied.

In the following, we will illustrate how to use the business relation model in the process of state searching. Algorithm 1 gives the algorithm of business relation based state searching. In this algorithm, the service-service relation and service-state relation are used in order to filter out some undesired services. The resource state-resource state relation is also used in order to have a common knowledge regarding state.

**Algorithm 1** Business relation model based state searching

```

StateSet StateSearching( PreState, PreService, ServiceSet)
{
  for each servicei ∈ ServiceSet
  {
    if ∃ ar = ⟨ PreService, service ⟩ ∈ service-service relation
    {
      if ( similar(servicei, service) = 1)
      {
        put servicei into StateSet[ ][0];
      }
    }
  }
}

```

```

        put servicei. effect into StateSet[ ][1];
        put PreState into StateSet[ ][2]; }
        continue; }
    else if ∃ as = ⟨ PreState, service ⟩ ∈ service-service relation
    {
      if ( similar(servicei, service) = 1)
      {
        put servicei into StateSet[ ][0];
        put servicei. effect into StateSet[ ][1];
        put PreState into StateSet[ ][2]; }
      continue; }
    else { for each si ∈ servicei. Precondition
      {
        if ( ∃ sk ∈ PreState ∧ sk ⇒ si ) or ( ∃ sk ∈ PreState ∧ ⟨ sk, si ⟩ ∈ resource state-resource state relation) continue; else break;
        put servicei into StateSet[ ][0];
        put servicei. effect into StateSet[ ][1];
        put PreState into StateSet[ ][2];
      } } }
    } } }
  } } }
  int Similar(servicei, servicej)
  {
    for each si ∈ servicei. Precondition
    {
      if ( ∃ sk ∈ servicej. Precondition ∧ sk ⇒ si ) or ( ∃ sk ∈ servicej. Precondition ∧ ⟨ sk, si ⟩ ∈ resource state-resource state relation) continue; else return 0; }
    }
    for each si ∈ servicei. effect
    {
      if ( ∃ sk ∈ servicej. effect ∧ sk ⇒ si ) or ( ∃ sk ∈ servicej. effect ∧ ⟨ sk, si ⟩ ∈ resource state-resource state relation) continue; else return 0; } return 1; }
  }
}

```

Although the business relation model can filter out sets of undesired ones, there also may be a number of candidate states that need to select and each of these states may conduct a number of state searching processes. How to select the proper state in order to enable the planning process to have the lowest number of state searching processes becomes the key problem. For this problem, this paper uses goal distance as an evaluation of states.

**Definition 8** (goal distance) The goal distance factor of  $gd(\text{state}, \text{goal})$  is defined as

$$gd(\text{state}, \text{goal}) = \frac{| \{ s \mid s \in \text{goal} \} |}{| \{ s \mid s \in \text{state} \wedge s' \in \text{goal} \wedge (s \Rightarrow s' \vee \langle s, s' \rangle \in \text{resource state-resource state relation}) \} |}$$

The basic idea of composition is to select the state which has the shortest goal distance from the goal state. Based on the goal distance, the composition algorithm can be described as follows:

**Algorithm 2** Goal distance based heuristic algorithm for composition

```

ServiceSet Composition( GoalSet, InitialState, ServiceSet)
{
  ServiceSet = ∅; maxStep = 20;
  StateStack[ ][ ][ ] = ∅;
  StateStack[ ][ ][ ]. visit = not-visited;
  TempGoalSet = ∅; n = 1;
  If StateSearching( InitialState, null, ServiceSet) is not null
  Put StateSearching( InitialState, null, ServiceSet) into StateStack
  [ n ][ ][ ];
  Else
  Return null; // means that no plan will be found;
  While ( step < maxStep)
  {
    ...
  }
}

```

```

{a: For each  $ss_i \in \text{StateStack}[n][][ ]$  and  $ss_i$ . visit = not-visited
{gdi = Evaluate( $ss_i$ , GoalSet); //compute the goal distance of each
state;
CurrentState =  $ss_i$  which having the shortest goal distance gdi;
CurrentState. visit = visited; Step + +;
Put currentState[ ][0] into ServiceSet;
If gdi = 1 Return ServiceSet;
Else Put states of GoalSet not deduced from currentState[ ][1] into
TempGoalSet;
If StateSearching( CurrentState[ ][1], currentState[ ][0], ServiceSet)
is not null;
Put StateSearching( CurrentState[ ][1], currentState[ ][0], ServiceSet)
into StateStack[ n + 1 ][ ][ ];
Else {Remove currentState[ ][0] from ServiceSet;
Put state of TempGoalSet caused by currentState[ ][0] to GoalSet;
Step - -; n - -; }
If  $\forall ss_i \in \text{StateStack}[n][ ][ ]$  and  $ss_i$ . visit = visited //trace back to
the last state;
{ If n > 0 Goto a;
Else Return null; } }
float Evaluate( state, GoalSet)
{ for each  $s_i \in \text{state}$ 
{ if (  $\exists s_k \in \text{GoalSet} \wedge s_i \Rightarrow s_k$  ) or (  $\exists s_k \in \text{GoalSet} \wedge \langle s_i, s_k \rangle$ 
 $\in \text{resource state-resource state relation}$  )
n + +; } return Getlength( GoalSet ) / n; }

```

### 3 Related Work and Experiment

In order to implement the automated composition of web services, approaches of template based composition are proposed<sup>[8-10]</sup>. The disadvantages of these approaches are that the construction of templates needs the help of humans and thus reduces the automation of composition. For this reason, approaches of automated generation of composite services have been proposed<sup>[11]</sup>. The work of this paper is among these works. Based on the above algorithms, a web service composition system SDOWSCP<sup>[12]</sup> is implemented.

In order to verify the effectiveness of the proposed algorithm, experiments are also conducted. It is run with a PC of Intel Pentium IV 2.4 GHz CPU and 1 GB RAM. The operation system is Windows Server 2000. And the algorithms involved in the composition part are written in C language. The test results address the effectiveness of the proposed algorithm compared with the classical space searching algorithm.

Here, we conduct the experiments in six situations where the number of services are 5, 10, 20, 30, 50, 100, respectively, and the goal set contains 15 states. The experimental results are shown in Fig. 1.

From this experiment, we find that with the increase in the number of services, the runtime of the proposed approach is much less than that of the classical one.

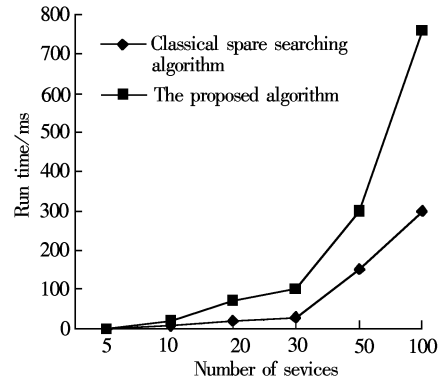


Fig. 1 Comparison of runtime

### 4 Conclusion

In order to solve the problem of the AI planning based web service composition, this paper proposes a business domain oriented AI planning for web service composition. Compared with other approaches of tackling web service composition as space searching problems, this work focuses on how to apply the business feature of the web service composition into the composition in order to improve its effectiveness. Finally, the experiment shows the better performance of the proposed algorithm. In the future, we will work on how to assist users to express their composition requirements and how to improve the effectiveness of the proposed algorithm based on some AI algorithms such as the generic algorithm.

### References

- [1] Alonso G, Casati F, Kuno H, et al. *Web services: concepts, architecture, and applications* [M]. Springer Verlag, 2003.
- [2] Korhonen J, Pajunen L, Puustjarvi J. Automatic composition of web service workflows using a semantic agent [C]//*Proceedings of the IEEE/WIC International Conference of Web Intelligence*. Halifax, Canada, 2003: 566 – 569.
- [3] Peer J. Bringing together semantic web and web services [C]//*Proceedings of the International Semantic Web Conference*. Sardinia, Italia, 2002: 279 – 291.
- [4] Ankolekar A, Burstein B, Hobbs J, et al. DAML-S: semantic markup for web services [C]//*Proceedings of the 1st International Semantic Web Working Symposium*. Stanford, CA, 2001: 411 – 430.
- [5] Liao Jun, Tan Hao, Liu Jinde. Describing and verifying web service using PI-calculus [J]. *Chinese Journal of Computers*, 2005, 28(4): 635 – 642. (in Chinese)
- [6] Ugo D L, Marco P, Paolo T. Planning with a language for extended goals [C]//*Proceedings of the 18th National Conference on Artificial Intelligence and Fourteenth Con-*

- ference on Innovative Applications of Artificial Intelligence*. Alberta, Canada, 2002: 447 – 454.
- [7] Barrett A, Weld D. Partial order planning: evaluating possible efficiency gains [J]. *Artificial Intelligence*, 1994, **67** (1): 71 – 112.
- [8] Tosić V, Mennig D, Pagurek B. On dynamic service composition and its applicability to e-business software system [C]//*Proceedings of Workshop on Object-Oriented Business Solutions*. Budapest, Hungary, 2001: 95 – 108.
- [9] Narayanan S, McIlraith S A. Simulation, verification and automated composition of web services [C]//*Proceedings of International World Wide Web Conference*. Honolulu, USA, 2002: 77 – 88.
- [10] Benatallah B, Dumas M, Sheng Q Z, et al. Declarative composition and peer to peer provisioning of dynamic web services [C]//*Proceedings of IEEE International Conference on Data Engineering*. California, USA, 2002: 297 – 308.
- [11] Thakkar S. Dynamically composing web services from on-line source [C]//*Proceedings of AAAI Workshop on Intelligent Service Integration*. Edmonton, Canada, 2002: 1 – 7.
- [12] Yang Lei, Dai Yu, Zhang Bin, et al. A service oriented web services composition platform [J]. *Journal of Wuhan University: English Edition*, 2006, **11** (1): 160 – 164.

## 面向业务领域基于智能规划的 web 服务组合

代 钰      杨   雷      张   斌

( 东北大学信息科学与工程学院, 沈阳 110004 )

**摘要:** 为了提高基于智能规划的 web 服务组合的效率, 提出了一个面向业务领域的 web 服务组合的智能规划算法. 该算法通过所提出的包含资源间关系、资源动作间关系以及动作和动作间关系的业务关联关系模型缩小规划问题的搜索空间, 将所提出的目标距离作为启发式规则以提高规划的效率. 与其他的基于智能规划的 web 服务组合算法相比, 该算法不仅解决了 web 服务组合问题向智能规划问题的转换, 同时也解决了如何将 web 服务组合的业务特点应用于智能规划中以提高规划效率的问题. 最后, 实验证明了所提出算法的有效性.

**关键词:** 智能规划; web 服务; web 服务组合; 业务关联关系模型

**中图分类号:** TP311