Mapping Semantic Web Service Descriptions to Planning Domain Knowledge

Hyun-Sik Kim and In-Cheol Kim

Department of Computer Science, Kyonggi University, Suwon-si, KOREA

Abstract—**Recently semantic web and web service are considered to be a key technology for realizing interoperability between heterogeneous healthcare information systems. One of the advantages of the web service technology is the automated generation of complex services out of a set of simple atomic services. In this paper, we present a new OWLS2PDDL converter. It is a conversion tool that converts OWL-S 1.1 service descriptions to the corresponding PDDL 2.1 descriptions that are can be used by many AI planners as input to plan a service composition that satisfies a given goal. We also illustrate how to convert some examples of semantic web services into the corresponding PDDL descriptions.**

Keywords— **Semantic Web Service, Service Composition, AI Planning, OWL-S, PDDL**

I. Introduction

Recently there has been some work to apply AI planning techniques to the Web Service composition problem [7, 8, 9]. The straight-forward approach is to map service descriptions to planning operators and directly use existing planners. OWL-S language [14] was developed to provide a set of ontologies to describe services using Web Ontology Language (OWL) [3]. OWL-S allows for describing services in ways amenable to planning. For example, it supports describing the preconditions and effects of atomic processes. Precondition presents logical conditions that should be satisfied prior to the service being requested. Effects are the result of the successful execution of a service. Such atomic process descriptions are easily treated as planning operators.

The prior versions of OWL-S have left the particular language for encoding preconditions and effects unspecified. Consequently, translation schemes from OWL-S to particular planning formalisms have had to insert their own encodings of preconditions and effects into the translated operators. As a result, the translated precondition and effect formulas are easily handled by those planners. Unfortunately, the typical logic for expressing preconditions and effects in a planning system has a radical different expressiveness than RDF and OWL do. So, these systems are not exploring what it would be like to plan against the kinds of encodings of the world state that we expect to find on the Semantic Web. However, the new version, OWL-S 1.1 forces the issue by using SWRL (Semantic Web Rule Language), KIF (Knowledge Interchange Format), or DRS (Declarative RDF System) as a description language for encoding service preconditions and effects.

In the area of AI planning, there has been rapid development of general planners which input PDDL (Planning Domain Description Language) [4] domain models. In this paper, we present a new OWLS2PDDL converter. It is a conversion tool that converts OWL-S 1.1 service descriptions to the corresponding PDDL 2.1 descriptions that are can be used by many AI planners as input to plan a service composition that satisfies a given goal. We also illustrate how to convert some examples of semantic web services into the corresponding PDDL descriptions.

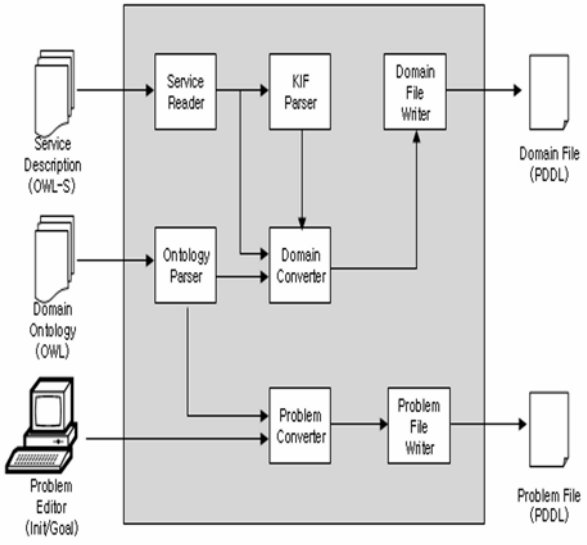

Fig. 1 Components of our OWLS2PDDL converter

II. Organization of the OWLS2PDDL Converter

Inputs to our OWLS2PDDL converter include semantic service descriptions and domain ontologies in OWL(-S). Some of dynamic property instances can be input through a separate problem editor. They usually represent a planning problem consisting of the initial state and the goal state. On

the other hand, outputs from the converter are a PDDL domain file and a PDDL problem file.

Our OWLS2PDDL converter consists of several modules shown in Fig. 1. *Service Reader* parses the given service descriptions and extracts annotation information such as input, output, preconditions and effects to hand over *Domain Converter*. *KIF Parser* analyzes the KIF preconditions and effects in service descriptions and submits the result to *Domain Converter*. *Ontology Parser* scans domain ontologies in OWL and extracts class and property information for constructing PDDL domain models. And then *Domain Converter* collects the relevant information to generate action descriptions in PDDL. *Domain File Writer* stores these action descriptions in a file. On the other hand, *Problem Converter* generates a PDDL problem description from class instances and property instances extracted by Ontology Parser or input by the user. *Problem File Writer* stores the problem description in a file.

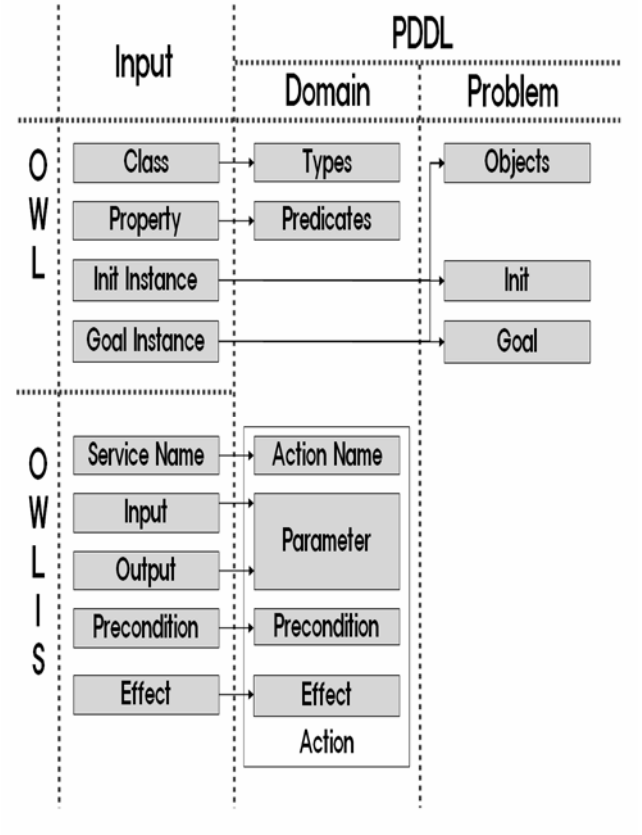

Fig. 2 Conversion rules

III. Conversion rules

Fig. 2 describes a set of conversion rules by which service descriptions and relevant domain ontologies in OWL(-S) are translated into the corresponding domain model and problem description in PDDL. The domain model contains the definition of all types, predicates and actions, whereas the problem description includes all objects, the initial state, and the goal state.

Classes and properties in OWL can be converted into PDDL types and predicates. Service descriptions in OWL-S can be translated into PDDL action descriptions. The service name is mapped to the corresponding action name, whereas the preconditions and effects of the service are mapped to the corresponding ones of the action respectively. We assume that the preconditions and effects of each service should be described in KIF, which is one of formal languages recommended in OWL-S 1.1 standards. So, KIF preconditions and effects are converted to PDDL ones. Input and output of each service are collectively mapped to the parameters of the corresponding action.

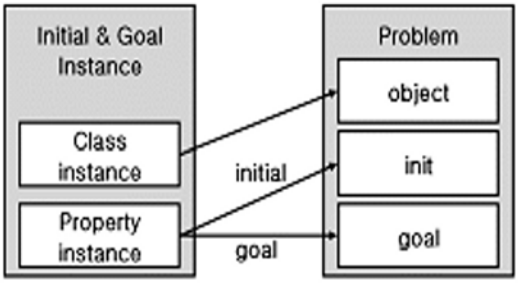

Fig. 3 Problem conversion

Class instances in the initial and goal ontologies are mapped to PDDL objects, whereas property instances are converted to the corresponding PDDL initial and goal state descriptions. A set of static property instances represent an invariant part of the state descriptions. So, all static property instances have to be added to the initial state description after conversion. Fig. 3 depicts this conversion rule in details.

IV. Examples

Fig. 4 shows an example of OWL-S service description. It represents the online BabelFish translator service to convert text from one language to another language.

```
        <process:hasEffect>
          <expr:KIF-Expression rdf:ID="dictionary_KIF-Expression">
            <expr:expressionBody rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >(and (nonTransed ?x) (not (transed ?x)) (usedDic))</expr:expressionBody>
          </expr:KIF-Expression>
        </process:hasEffect>
      </process:Result>
    </process:hasResult>
    <process:hasPrecondition>
     <expr:KIF-Condition rdf:ID="dictionary_KIF_Condition">
       <expr:expressionBody rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
       >(and (transed ?x))</expr:expressionBody>
     </expr:KIF-Condition>
    </process:hasPrecondition>
    <service:describes>
      <service:Service rdf:ID="trans_model">
        <service:supports>
          <grounding:WsdlGrounding rdf:ID="trans">
            <service:supportedBy rdf:resource="#trans_model"/>
          </grounding:WsdlGrounding>
        </service:supports>
        <service:describedBy rdf:resource="#dictionary"/>
      </service:Service>
    </service:describes>
  </process:AtomicProcess>
  <process:AtomicProcess rdf:ID="babelfish_translator">
    <process:hasPrecondition>
      <expr:KIF-Condition rdf:ID="babelfish_translator_KIF_Condition">
        <expr:expressionBody rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >(and (nonTransed ?from))</expr:expressionBody>
      </expr:KIF-Condition>
    </process:hasPrecondition>
    <process:hasOutput>
      <process:Output rdf:ID="to">
        <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.owl-ontologies.com/unnamed.owl#lang</process:parameterType>
      </process:Output>
    </process:hasOutput>
    <process:hasInput>
      <process:Input rdf:ID="from">
        <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.owl-ontologies.com/unnamed.owl#lang</process:parameterType>
      </process:Input>
    </process:hasInput>
    <process:hasResult>
      <process:Result rdf:ID="babelfish_translator_Result">
        <process:hasEffect>
          <expr:KIF-Expression rdf:ID="KIF-Expression">
            <expr:expressionBody rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

Fig. 4 Example of OWL-S description

This service has three inputs: the text to be translated, its input language and the desired output language. There are total of nine languages supported by the translator. The precondition of the service requires that the input language and output language should be different. Fig. 5 shows the converted domain model and problem description in PDDL. The domain model includes two distinct action descriptions: babelfish-translator and dictionary. The problem description requires the meaning of a given French word to be expressed in French. From the domain model and the problem description, a general PDDL-based planner such as Sapa can generate a solution plan shown in Fig. 5, The resulting plan represents a sort of composed Web service.

V. Conclusions

In this paper, we presented a new OWLS2PDDL converter. It is a conversion tool that converts OWL-S 1.1 service descriptions to the corresponding PDDL 2.1 descriptions that are can be used by many AI planners as input to plan a service composition that satisfies a given goal. Different from the existing OWL2PDDL converter, it has additional facilities to deal with KIF preconditions and effects of service descriptions and also pragmatically simplifies handling the input and output.

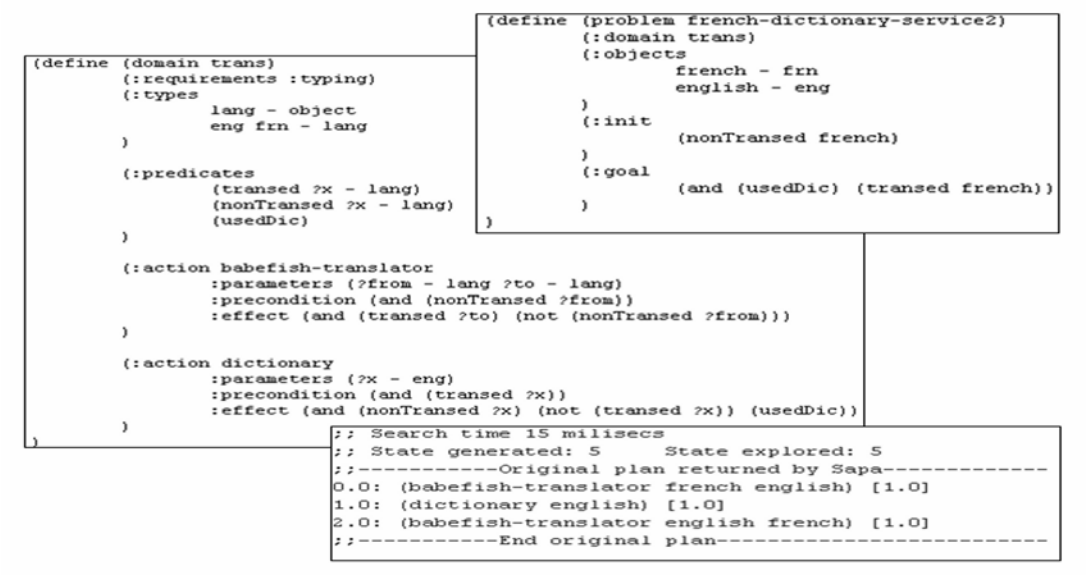

Fig. 5 The resulting domain, problem and plan files

REFERENCES

1. Biplav Srivastava, Jana Koehler (2003) Web Service Composition – Current Solution and Open Problems, , ICAPS2003 Workshop on Planning for Web Services, Trento, Italy, June 2003
2. Cabral, L., Domingue, J., Motta, E., Payne, T., Hakimpour, F. (2004) Approaches to Semantic Web Services: An Overview and Comparisons. Proceedings of the 1st European Semantic Web Symposium (ESWS2004), Springer-Verlag, 2004, pp 225-239
3. Deborah L. McGuinness, Frank van Harmelen (2004) OWL Web Ontology Language Overview, W3C Recommendation 10 Feb 2004
4. Drew McDermott (1998) PDDL--the planning domain definition language, AIPS'98 IPC Committee, 1998
5. Evren Sirin, Bijan parsia (2004) Planning for Semantic Web Services, Semantic Web Services Workshop at 3rd International Semantic Web Conference (ISWC2004), 2004
6. Fiona McNeill, Alan Bundy, Chris Walton (2005) Planning from Rich Ontologies through Translation between Representations, ICAPS2005 Workshop on the Role of Ontologies in Planning and Scheduling, June 2005
7. Jingjai Rao, Xiaomeng Su (2004) A Survey of Automated Web Service Composition Methods, First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004), San Diego, CA, USA, July 6, 2004
8. Joachim Peer (2005) Web Service Composition as AI Planning-a Survey, Technical Report, Univ. of St.Gallen, March 22, 2005
9. Mark Carmen, Luciano Serafini, Paolo Traverso (2003) Web Service Composition as Planning, ICAPS2003 Workshop on Planning for Web Services, Trento, Italy, June 2003
10. Matthias Klusch, Andreas Gerber, Marcus Schmidt (2005) Semantic Web Service Composition Planning with OWLS-Xplan, AAAI Fall Symposium on Agents and the Semantic Web, Arlington VA, USA, AAAI Press, 2005
11. Paolucci, M., Sycara K. (2003) Autonomous Semantic Web Services. IEEE Internet Computing, 7:34-41
12. T.L. McCluskey, S.N. Cresswell (2005) Importing Ontological Information into Planning Domain Models, ICAPS2005 Workshop on the Role of Ontologies in Planning and Scheduling, June 2005
13. Knowledge Interchange Format, draft proposed American National Standard (dpANS) CITS.T2/98-004
14. http://www.daml.org/service/owl-s/1.1

Address of the corresponding author:

Author:	Kim, Hyun-Sik
Institute:	Department of Computer Science, Kyonggi University
Street:	San94-6, Yiui-dong, Youngtong-gu
City:	Suwon-si
Country:	KOREA
Email:	advance7@kyonggi.ac.kr