A Framework for Intelligent Web Services: Combined HTN and CSP Approach

Incheon Paik[*], Daisuke Maruyama[*], Michael N. Huhns[**]
[*] *School of Computer Science and Engineering*
University of Aizu
E-mail: paiki@u-aizu.ac.jp, dmaru@ebiz.u-aizu.ac.jp
[**] *Dept. of Computer Science and Engineering*
University of South Carolina
E-mail: huhns@cs.edu

Abstract

Solving general real-life problems requires a set of appropriate services to be composed via planning, scheduled, and then executed. Web service composition is the most difficult aspect and is our focus. In this paper, we describe a new framework for intelligent semantic Web services that supports the planning and scheduling aspects by a combined HTN planner and CSP. The framework covers all of the procedures needed to deal with a user's request, including domain analysis of the request, task flow decisions and CSP creation by the planner, and solving the CSP by a distributed CSP solver.

1. Introduction

Web users require various types of information and constraints, and automatic service composition requires several rounds of planning, because of trial and error, or for flexibly coping with dynamic exceptions. Web service composition by a planner alone has limitations that apply to a more general and intelligent composition of services. First, it is inefficient for autonomously finding a solution in planning, because it does not provide a suitable basis for dealing with the evaluation of planning results with constraints. Second, although it works well for task ordering in planning, it is not good for dealing with a Web user's various requests for information. As real-life problems[1] involve planning, scheduling, and executing, Web service composition in real life requires not only planning information, but also additional information requests with constraints, which can be met by scheduling tasks jointly. A Constraint Satisfaction Problem (CSP) formulation provides a strong basis for scheduling in a variety of real-life problems on the Web. Third, it is weak regarding maintenance, because of the frequent invocation of services on the Web. Although an Hierarchical Task Network (HTN) planner can invoke outside Web services during planning, this causes severe restrictions and inefficiency, because service invocations in the planner are merged with the planning strategy.

In this paper, we suggest a combined architecture of planning and CSP for a basic problem-solving engine to automate Web service composition that tackles the problems above, giving an entire framework of intelligent Web services for users. We claim that a HTN and CSP combination is better than an HTN alone when problems involve scheduling plus other parameters. Composed Web service problems are of this type. The framework of a combined architecture will give an intelligent Web service composer a better environment for solving various problems, from tactical planning with well-established process fragments to puzzle-mode planning that characterizes domains such as the blocks-world.

2. Sample Scenarios

Users' constraints can be different in different domains, and there can be many details in a domain. Here, we illustrate an example of scenarios for intelligent Web services.

A user, who lives in Aizu City in Japan, wants to go to South Carolina in the U.S.A for a vacation. If the user wants to go by train to the Narita international airport near Tokyo, there are three stages: by local train from Aizu to Koriyama bullet train station, by bullet train from Koriyama to Tokyo, and by JR Express from Tokyo to Narita, from where a series of flights completes the journey to South Carolina.

Therefore, the user calls an agent to construct an itinerary to South Carolina. For this, the user provides basic information such as the departure date and location, and the arrival date and location. Suppose that the user wants to depart at 2:00 PM from Aizu because of a special business meeting. Therefore, he adds this new constraint to the basic input information.

Now, when the travel planner solves this problem, the solution may produce other internal spontaneous constraints temporarily. For instance, the planner should reserve a one-night stay in a hotel near Narita and a flight

[1] The planner determines a sequence of actions, and the scheduler maps activities and their respective operating times to resources. In this paper, we involve the executor with the HTN planner.

the next day when there is no flight to South Carolina at Narita on that day. On another occasion, the user may specify the arrival time in South Carolina as a constraint, which the planner will also need to accommodate.

3. Framework of Intelligent Web Service

In this section, an outline of our framework for intelligent Web services is explained. A user presents a request with some constraints to a service interface of the intelligent Web service. The user's request, in general domains, can be in the form of natural language or a flexible Web graphical user interface. Then, the request is passed to the domain analyzer (DA).

The DA analyzes the request to capture its goal and domain. According to this analysis, the DA decides the domain of the problem, a rule to solve the top-level problem, the initial variables and states, i.e., the initial CSP tuple, and the constraints.

The information is then passed to the problem-solving engine. The problem solving engine, which also plays the role of composing Web services automatically, using an HTN planner and a CSP solver, comprises three parts: the HTN planner, the CSP tuple, and the DCSP solver. This engine is the core element of our research. Here, we combine the HTN planner and CSP problem solving.

When the HTN planner in the problem-solving engine receives information about the goal problem such as the initial states and the goal, clues for HTN planning, and the initial CSP tuple, it selects a suitable method for the problem, planning from a KB or from available Semantic Web rules. As the planner operates, it produces the final CSP tuple according to the planning results.

Then, the CSP constructor elaborates the prestage CSP tuple to create complete CSP tuples, considering the ontology and grammar of CSP in that domain. We have developed a novel way of describing the CSP tuple in our framework. The CSP information will be converted into the form of a CSP tuple. The final CSP solver reads this input concerning CSP tuples to produce the final solution sets. The solver solves the problem by filling out the variables in the domain with values satisfying the constraints using backtracking search and a DCSP solving technique.

4. From HTN to CSP Domain

4.1 Input and Domain Analysis

In our framework, a user's request is analyzed in the DA, which will then be passed to the planner part of the problem-solving engine. A user's request mainly comprises three elements: basic information, additional information and the user's constraints, and a domain information goal. For example, this is a user request in the trip domain: "Plan a trip from Aizu-Wakamatsu to South Carolina, starting on Sep. 20, 2005 1:00 PM, returning to Aizu on Sep. 29, 2005."

4.2 Creation of HTN Input Information (HTNInput)

The set of UserRequest needs to be mapped into the contents of the HTN planner.

From UserRequest to HTNInput, mapping function f_{U2H}, f_{U2H}: UserRequest → HTNInput. Here, HTNInput = {Operator, Method, Axiom, InitialState, Goal}.

We can define the functions that map the elements of UserRequest into the HTNInput set as follows.

$$f_{U2H} = \{(BasicInfo,\ InitialState \cup Axiom),\ (UserConstraint,\ Method \cup Operator),\ (DomainInforGoal,\ Goal)\}$$

The information for HTNInput generated from UserRequest will be added to the real entity for input to the HTN planner.

4.3 Creation of CSP-Tuple Set from the HTN Planner

The objective of an HTN planner is to produce a sequence of actions that perform some activity or task to reach a goal. The description of the planning domain includes a set of operators similar to those of classical planning, and a set of methods, each of which is a prescription for how to decompose a task into subtasks. Planning proceeds by using methods to decompose tasks recursively into smaller and smaller subtasks, until the planner reaches primitive tasks that can be performed directly, using the planning operators.

Within this framework, the HTN is to generate the CSP set. Resulting from the planning operations of SHOP2, SHOP2 produces a sequence of instantiated operators that will achieve the task list from a state in a set of axioms, operators, and methods. These operators represent the main stream of work required to reach the goal state.

The planning result does not contain additional information, such as scheduling information, required to satisfy the final state from the user's original problem. In our trip domain example, the HTN planner will produce a basic route and transportation, such as "Aizu (Train)→ Koriyama (Shinkansen) → Tokyo (JR Express) → Narita (Airplane) → South Carolina". To fulfill the user's original request, we need to include time, fare, and transportation information, number of passengers, constraints among these data, etc. Therefore, although the planner develops a plan, it needs to produce additional information for a final solution in the form of a CSP.

5. Distributed CSP Solver

In the previous section, we showed the creation of the CSP set by the planner to fulfill a user's request. The CSP solver in this section inputs this CSP set to produce a final solution set, by solving the CSP. The CSP solver starts its operation by input of the CSP set. We define a format to describe the CSP set for the first-level CSP in our framework, called Planner-CSP Interchange Format (PCIF). As CSP comprises the triple <Variable, Constraint, Domain>, PCIF describes this triple. As the problem sets in CSP may belong to many problem domains, we adopt a distributed CSP (DCSP) architecture. The candidate values can be calculated from the internal system, in the usual CSP solving world. However, for Web services, we meet several different situations and variations, such as different domain applications, different service discovery and composability, and network situations. In addition, for better performance on the Internet, we need to invoke Web services simultaneously and independently. These considerations lead us towards using a DCSP set. For DCSP, we also adopt the concept of "affiliation" according to the application domain. Affiliation makes the CSP set differentiate between domains.

From the DCSP tuple, the DCSP solver searches for a final solution to the user's request to invoke Web services in the relevant domain.

6. Implementation and Evaluation

We implemented the whole sequence of the proposed framework in order to provide proof of concept for it. The implementation mainly comprises three parts: domain analyzer, HTN planner, and CSP handler.

JSHOP2 was used as the planner. The planner generates two pieces of information for the CSP: a task flow (action sequence), and a CSP set corresponding to the scheduling input for the final solution of the user's request. JSHOP2 was revised to generate the CSP sets used by the HTN planner. The planner and CSP solver communicate with Web services by Axis Simple Object Access Protocol engine.

In the experiment, the problem solving time and the number of Web service invocations were counted. Table 1 shows the experimental results. In our test samples, we found that the number of invocations of Web services and the problem solving time for the two models were similar.

The processing takes a long time because the generated CSP set is supplied by the I/O operation, and the object mapping and instantiation take considerable time. A revision of the processing method to remove I/O operations, and of the lightweight design of the CSP solver, will enhance the system's performance. However, as Web service invocations take far more processing time than problem solving, we need to decrease the number of service invocations for better processing-time performance.

Table 1. Experimental Result

Item Solving System	**Web Service Invocations**	**Solving Time (ms)**	**Activity in Composition**
HTN Planner only	4	3039	Planning
Combined HTN and CSP	4	5789	Planning Scheduling

7. Conclusions and Future Work

We presented in this paper a framework that provides a user with intelligent services based on Semantic Web services. We describe a complete architecture to capture the domain from a user's request, generate a solution plan through semantic Web service composition, and then execute the solution plan to satisfy the user's requirement. The main part of the framework, a model for problem solving to automate Web service composition with additional scheduling information by the combined HTN planner and CSP, is described. Our combined system removes the limitations very well. The HTN produces the main task flow for problem solving in the form of a CSP set, which contains all information for the user's initial requirement, including additional scheduling information, and meshes well with the task flow information to produce a final solution. We also implemented a prototype for giving proof-of-concept of the framework.

8. References

[1] E. Sirin, B. Parsia, D. Wu, J. Hendler, D. Nau, HTN planning for Web service composition Using SHOP2, Journal of Web Semantics, Vol. 1, No. 4, 2004, pp. 377-396.

[2] Ugur Kuter, Evren Sirin, Bijan Parsia, Dana Aau, James Hendler, "Information Gathering During Planning for Web Service Composition", In *Proceedings of the Third Internatonal Semantic Web Conference (ISWC2004)*, Hiroshima, Japan, November 2004.

[3] A. Nareyek and E. C. Freuder, etc, "Constraints and AI Planning", IEEE Intelligent System, Mar./Apr., 2005, pp. 62-72.