Evaluation of Service Composition Planning With OWLS-XPlan[1]

Matthias Klusch and Andreas Gerber
German Research Center for Artificial Intelligence
Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany
{klusch@dfki.de, andreas.gerber@x-aitment.net}

Abstract

In this paper, we present the implementation, evaluation, and application of our OWL-S service composition planner OWLS-XPlan. Services in OWL-S 1.1 and ontologies are converted to initial state and goal descriptions in PDDL 2.1, which are then used by the fast heuristic FF planner XPlan for generating an execution complete composition plan. Results of experimental evaluation of XPlan shows its top performance compared with other selected AI planners. OWLS-XPlan is used in an agent based eHealth system for medical patient transport planning.

1. Introduction

Though the composition of complex Web services attracted much interest in different fields related to service oriented computing, there are only a few implemented composition planning tools publicly available for the semantic Web such as the HTN composition planner SHOP2 [12] for OWL-S services. One problem with HTN planners is that they require task specific decomposition rules and methods developed at design time, hence are not guaranteed to solve arbitrary planning problems.

That, in particular, motivated the development of our hybrid composition planner for OWL-S 1.1 services, called OWLS-XPlan, which always finds a solution if it exists, though the corresponding planning problem remains to be NP-complete. OWLS-XPlan is integral part of the prototypically implemented medical application service system Health-SCALLOPS. An extended version of OWLS-XPlan, called OWLS-XPlan+, that allows for quasi-online re-planning of service composition is used within the European project CASCOM.

The remainder of this paper is structured as follows. Section 2 briefly introduces OWLS-XPlan, followed by the evaluation results of its core planner XPlan and implementation in sections 3 and 4. Use of OWLS-XPlan in an eHealth application is described in section 5, while we briefly refer to related work and conclude in sections 6 and 7, respectively.

2. OWLS-XPlan Overview

The semantic web service composition planner OWLS-XPlan consists of several modules for pre-processing and planning (cf. figure 1). It takes a set of available OWL-S 1.1 services, related OWL ontologies, and a planning request (goal) as input, and returns a planning sequence of relevant services that satisfies the goal.

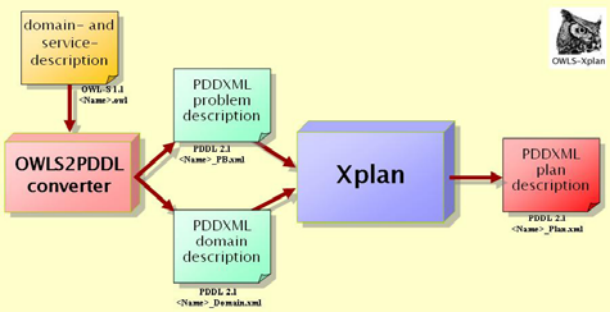


Fig. 1. OWLS-XPlan Architecture

For this purpose, it first converts the domain ontology and service descriptions in OWL and OWL-S, respectively, to equivalent PDDL 2.1 problem and domain descriptions using the integrated OWLS2PDDL converter. The domain description contains the definition of all types, predicates and actions, whereas the problem description includes all objects, the initial state, and the goal state. Both descriptions are then used by the AI planner XPlan to create a plan in PDDL that solves the given problem in the actual domain. For reasons of convenience, we developed a XML dialect of PDDL, called PDDXML that simplifies parsing, reading, and communicating PDDL descriptions using SOAP. An operator of the planning domain corresponds to a service profile in OWL-S: Both operator and profile describe patterns of how an action or service as an instance should

[1] This work has been supported by the German Ministry of Education and Research (BMBF 01-IW-D02-SCALLOPS), http://www.dfki.de/scallops

look like. A method is a special type of operator for fixed complex services that OWLS-XPlan may use during its planning process. Its core AI planning module called XPlan is a heuristic hybrid FF planner based on the FF planner developed by Hoffmann and Nebel [5, 6]. It combines guided local search with relaxed graph planning, and a simple form of hierarchical task networks to produce a plan sequence of actions that solves a given problem. If equipped with methods, XPlan uses only those parts of methods for decomposition that are required to reach the goal state with a sequence of composed services.

For each sub-goal g of the determined goal agenda, at each planning step i, XPlan quickly builds a relaxed planning graph RPG(i) in a fast goal reachability test heuristically ignoring negative effects of actions, and the corresponding relaxed plan RP(i) in a backward pass from g to S_i. The relaxed plan contains all paths of applicable actions that lead from g to S_i, of which only those in its first action-layer 0 are called helpful. In the following, XPlan focuses on the helpful actions of RP(i) only, hence reduces the search space. Please note that the relaxed plan is not necessarily correct.

In order to decide which helpful action to select as the next action in a valid plan sequence, it applies each of them to S_i and adds the previously ignored Del-list facts yielding the complete state S_{ij}, where j in {1,.., l}, denotes the j-th helpful action applied to state S_i .

For each of these states the relaxed plan RPG(i,j) is built to heuristically search for the relaxed plan RP(i,j) with heuristically minimal length h(RP(i,j)). In this context, the "plan length" h(RP(i,j)) just denotes the sum of all actions in all action-layers of the RP. Finally, XPlan retains the action A_{ij} with heuristically minimal goal-distance and starts the next planning step i+1 with S_{ij}. If there are multiple RPs of equal length, it repeats the same decision process starting at state S_{i1} (like a breadth first search restricted on helpful actions), and then S_{i2}, ..., S_{il} until a minimum is found.

Eventually, all created plans for sub-goals g of the goal agenda are respectively concatenated which yields the final plan sequence P. The plan then gets executed, and if it fails, XPlan allows re-planning from the most recent valid state produced by action execution, to avoid a total re-planning, if possible.

For more details on the service composition planner OWLS-XPlan, we refer the reader to [14].

3. Evaluation of XPlan

We evaluated the performance of XPlan, using the benchmark of the international planning competition in 2003 (IPC3) [5], and compared the results with that of the four top performing IPC3 participants, i.e. FF planner, SimPlanner, and the HTN planners TLPlan, and Shop2. XPlan was tested without task specific methods. Planning performance was measured in terms of (a) the planning completeness, i.e. the total percentage of solved problems (cf. figure 2), (b) the average plan length (cf. figure 3), and (c) the average plan quality, i.e. the average distance of individual plans from the optimal plan length (cf. figure 4) in relation to the complexity of the given problems. The complexity of a planning problem is defined as the number of objects of the type definitions specified in the given planning problem domain description. We grouped all test cases of the IPC3 test scenarios leading to 122 problems in total into complexity classes with an increasing number of objects.

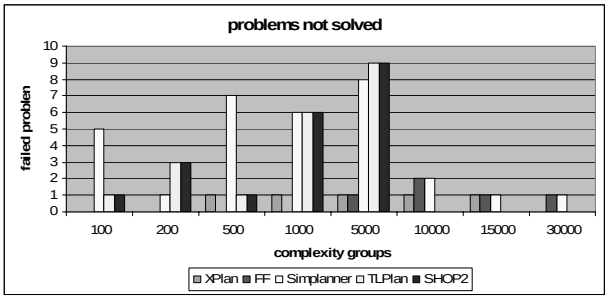


Fig. 2: Completeness

First, we tested the completeness of planning (cf. figure 2). It turned out that XPlan and FF planner failed to solve only a few mid ranged complex problems; in fact, both solved nearly 97% of given problem cases. There were no results reported for TLPlan and SHOP2 for the last six test cases, they failed a lot in solving problems of low and mid range complexity, but performed very well in solving more complex problems. Main reason is that the HTN planners turned out to be equipped with methods that better enabled them to solve highly complex problems in most domains. Figure 3 summarizes the results of testing the average plan length in relation to the complexity of the problem definition. The HTN planners produced shorter plans than their competitors with increasing complexity of the problem, whereas XPlan outperformed all other planners for given problems of low and mid range complexity.

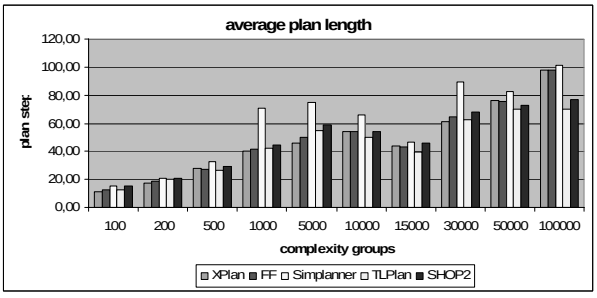


Fig. 3: Average plan length

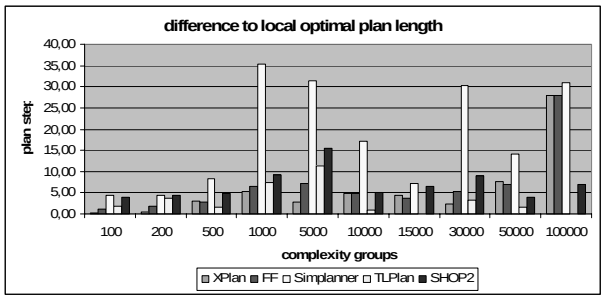
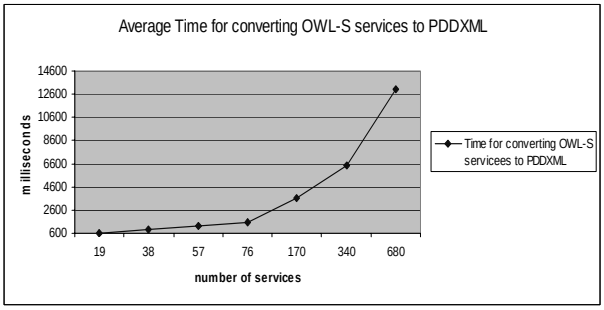


Fig. 4: Average plan quality

Finally, we measured the average plan quality in terms of the average distance of individual plans from the optimal solution of a given problem (cf. figure. 4). That is, we counted the number of additional plan steps of a solution generated by an individual planner compared to that of the shortest plan created for the given problem, averaged over all test cases per complexity class. In this respect, except for the most complex problems, XPlan outperformed the other planners.



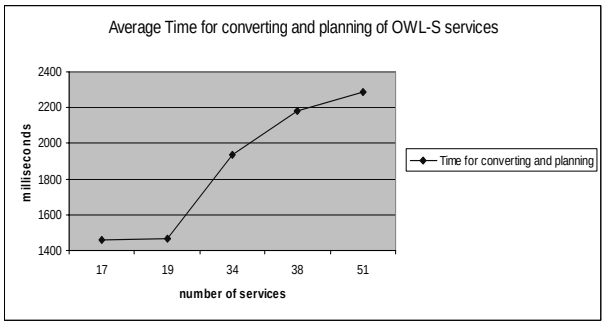


Fig. 5: Average run time for conversion and planning

We did not have specific information about the underlying computing hardware used in the IPC3 competition for run time measurement. Figure 5 shows the reasonably fast run time of converting and planning by OWLS-XPlan on a Siemens-Fujitsu Amelio 1425 notebook with 1.8 Ghz Intel Centrino, and 1 GB RAM.

4. Implementation

OWLS-XPlan has been implemented in Java and C++, and provides an integrated graphical user interface (cf. figure6). XPlan uses the Microsoft MSXML parser for PDDXML definitions and generating plans in XML format. In addition, OWLS-XPlan provides an integrated PDDXML editor that allows the experienced user to edit the goal, and the initial state ontology of given planning problem.

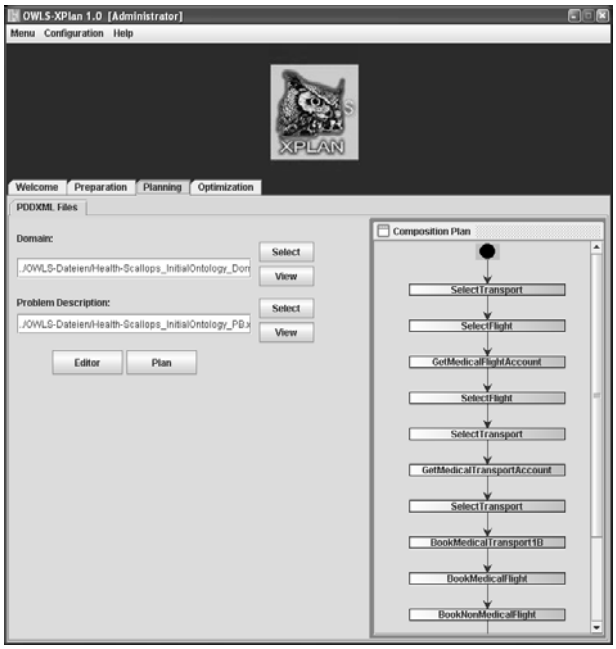


Fig. 6: OWLS-XPlan GUI (part of)

The initial (world) state ontology is assumed to be provided to the system; we acknowledge that this is a major hurdle for inexperienced users, hence are working on a more convenient user interface.
The generated plan is being displayed, and can be further optimized with respect to given QoS parameters by means of ILP based optimization with newly provided semantically equivalent services. OWLS-XPlan 1.0 is available under GPL at [17]. We are currently working on an improved version 2.0 that also allows for dynamic re-planning in case of non-deterministically occurring changes during planning.

5. Related work

Existing approaches to service composition planning can roughly be classified into process oriented, and data or signature oriented approaches. Members of the first presume a goal that specifies the global behaviour of the desired service in terms of the set of possible desired

conversations, or a process flow to be accomplished by synthesizing the process models of available services that can either be modified during composition [2], or not [1]. Specification of service behaviour usually takes the form of FSMs, Petri Nets [13, 16], situation calculus [8], or linear temporal branching logic formulas. Signature-oriented or data-driven composition approaches do not take the process of a service into account but try to instantiate a goal specification given by the I/O signature of a desired service together with constraints and user preferences only. Such an instance is a sequence of atomic or other composite services considered as black boxes that accomplishes the goal. OWLS-XPlan falls into the latter category, and is tightly related to classical planning in AI [10]. An accessible approach to solutions of the problem of cyclic composition planning via model checking is in [15]. To the best of our knowledge, it holds that (a) none of the implemented planners including OWLS-XPlan does cope with the open world assumption of OWL, and (b) OWLS-XPlan and Shop2 are the only implemented OWL-S service planners publicly available.

6. Conclusion

The implemented OWL-S service composition planner OWLS-XPlan is used in a prototyped medical health information service system. Its hybrid core planner XPlan exploits both relaxed Graph-Plan based FF-planning with local search and HTN based planning. According to experimental evaluation, XPlan performs reasonably well compared to other relevant planners. OWLS-XPlan is publicly available for downloading, and used in an agent based emergency medical assistance system.

8. References

[1] D. Berardi, D. Calvanese, G. D. Giacomo, M. Lenzerini, and M. Mecella. Automatic service composition based on behavioral descriptions. *Cooperative Information Systems, 14(4)*, 2005.

[2] T. Bultan, X. Fu, R. Hull, and J. Su. Conversation specification: A new approach to the design and analysis of e-service composition. *Proc. World Wide Web Conference WWW, Budapest, Hungary*, 2003.

[3] International Planning Competition 2002. IPC3. http://planning.cis.strath.ac.uk/competition/

[4] J. Hoffmann. A heuristic for domain indepndent planning and its use in an enforced hill-climbing algorithm. *Proc .12th Intl Symposium on Methodologies for Intelligent Systems, Springe*, 2000.

[5] J. Hoffmann. The Metric-FF planning system: Translating Ignoring Delete Lists to Numeric State Variables. *Artificial Intelligence Research, 20*, 2003.

[6] J. Hoffmann and B. Nebel. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Artificial Intelligence Research*, 14, 2001.

[7] A. Lotem, D. Nau, and J. Hendler. Using planning graphs for solving HTN problems. *Proceedings of AAAI/IAAI conference, USA*, 1999.

[8] S. McIllraith and T. Son: Adapting Golog for composition of semantic Web services. *Proceedings of Intl Conference on Knowledge Representation and Reasoning KRR, Toulouse, France*, 2002.

[9] B. Medjahed, A. Bouguettya, A.K. Elmagarmid. Composing Web services on the semantic Web. *Very Large Data Bases (VLDB), 12(4)*, 2003

[10] J. Peer. Web Service Composition as AI Planning: A Survey. *Technical Report, U St. Gallen, Switzerland http://elektra.mcm.unisg.ch/pbwsc/docs/pfwsc.pdf*, 2005.

[11] M. Schmidt. Ein effizientes Planungsmodul fuer die lokale Planungsebene eines InteRRaP Agenten. Master Thesis, U Saarland, Germany, 2005.

[12] E. Sirin, B. Parsia, D. Wu, J. Hendler, D. Nau. HTN planning for web service composition using SHOP2. *Journal of Web Semantics, 1(4)*, 2004.

[13] R. Hamadi, B. Benatallah: A Petri-Net Based Model for Web Service Composition. *Proc. 14th Australian Conference on Database Technologies*, ACM Press, 2003

[14] M. Klusch, A. Gerber, M. Schmidt: Semantic Web Service Composition Planning with OWLS-XPlan. *Proceedings of the AAAI Fall Symposium on Semantic Web and Agents, Arlington VA, USA, AAAI Press*, 2005.

[15] A. Cimatti, M. Pistore, M. Roveri, P. Traverso: Weak, strong, and strong cyclic planning via symbolic model checking, *Artificial Intelligence,147(1/2)*, 2003.

[16] P. Traverso, M. Pistore: Automated Composition of Semantic Web Services into Executable Processes. *Int Semantic Web Conference, LNCS 3298, Springer*, 2004.

[17] OWLS-XPlan: http://projects.semwebcentral.org/projects/owls-xplan/