Fast Dynamic Re-Planning of Composite OWL-S Services[1]

Matthias Klusch, Kai-Uwe Renner
German Research Center for Artificial Intelligence
Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany
{klusch, Kai-Uwe.Renner}@dfki.de

Abstract

In this paper, we present an extension of our OWL-S service composition planner OWLS-XPlan that allows for quasi-online re-planning of composite OWL-S services without full restart of the actual planning process, and preliminary experimental evaluation results.

1. Introduction

Though the AI based composition planning of complex Web services attracted much interest recently [5, 11], only a few planning tools are actually available for the semantic Web, such as the HTN based composition planner SHOP2 [7, 8, 4], or OWLS-XPlan [9] for OWL-S services. However, none of these planners copes with the open world assumption of OWL, but performs more or less efficient CWA based off line planning.

In open environments, like the semantic Web, non-deterministically occurring events such as broken service links, change of facts, or goal, and availability of new services may affect the actual planning process of a composite service. The actual plan, or parts of it, may become invalid or sub-optimal even before its full generation. Invalid plans may be caused by, for example, services that became unavailable, or facts that satisfied a precondition of some service in the current planning sequence changed such that the semantic compatibility with its preceding service is invalid. Newly introduced services may cause sub-optimality of the current plan in terms of its path length to the given goal state. None of the currently available OWL-S service composition planners does allow for dynamic re-planning, which in turn motivated us to extend our own planner OWLS-XPlan to accomplish this task. Basic idea of OWLS-XPlan+ is to re-use as much as possible of the existing plan such that the minimally modified plan as a whole remains valid in the changed world state. Though the state of the world gets checked for any changes that may affect the current plan at the end of each plan step, and if so, triggers immediate re-planning off-line, but actions are executed only after a plan has been eventually created that is guaranteed to reach the given goal. This is in contrast to classical on-line planning approaches where typically a planner generates conditional plans that branch over observations, while a controller executes actions in the plan, and monitors observations to decide which branch to execute. Any kind of interleaving framework, in general, cannot guarantee that a goal state will be reached, unless the domain is proven to be safely explorable. Services provided by autonomous providers cannot be assumed to be executable under full control and observation of the planning site, nor to be delivered charge free even in scenarios of tight collaboration with respective service providers. We set the context by briefly introducing our service composition planner OWLS-XPlan in section 2, and then describe the dynamic re-planning by its extended planning module XPlan+ in section 3. We present preliminary experimental evaluation results in section 4, and conclude in section 5.

2. OWLS-XPlan Overview

The semantic web service composition planner OWLS-XPlan consists of several modules for pre-processing and planning of composite OWL-S services (cf. figure 1).

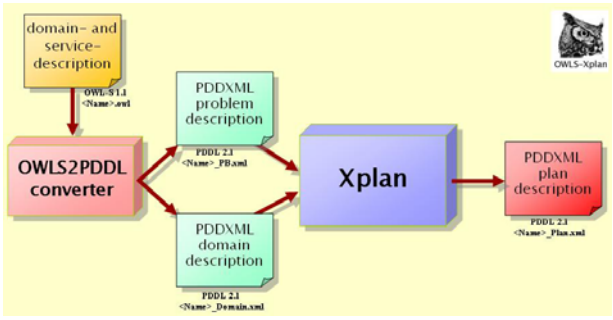

Fig. 1. OWLS-XPlan Architecture

[1] This work has been supported by the German Ministry of Education and Research (BMBF 01-IW-D02-SCALLOPS) and by the European Commission under the project grant FP6-IST-511632-CASCOM.

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 OWL-S services that satisfies the goal. For this purpose, it first converts a given domain ontology and service descriptions in OWL and OWL-S 1.1, respectively, to equivalent PDDL 2.1 problem and domain descriptions using an 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. An operator of the planning domain corresponds to a service profile in OWL-S, while a method is a special type of operator for fixed complex services that OWLS-XPlan may use during its planning process.

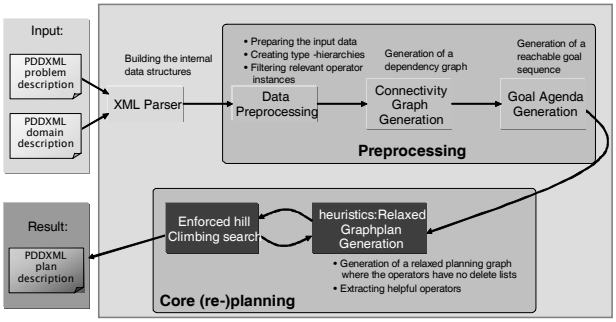

Fig. 2: The planning module XPlan

The planning module XPlan (cf. figure 2) is a heuristic hybrid FF planner based on the FF planner developed by Hoffmann and Nebel [1, 2, 3]. 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. Due to space restrictions, for more details on OWLS-XPlan in general, and XPlan in particular, we refer the reader to [9]. The sources are available at [12].

3. Dynamic Re-Planning by XPlan+

We modified the original XPlan module of OWLS-XPlan to allow for event driven heuristic re-planning of composite services during the actual planning process. The corresponding OWL-S composition planner is called OWLS-XPlan+. The modified planner XPlan+ does perform, in essence, highly frequent event driven off-line re-planning under closed world asumption with heuristic computation of best re-entry points for re-planning at the end of each planning step if the currently produced plan, or plan fragment gets affected by the observed change. External changes in the world state concern converted OWL ontologies, individuals and the set of available services during the internal planning process each of which potentially affecting the respective operators, actions, predicates, facts and objects in the PDDXML problem and domain descriptions as well as already generated partial plans. For event monitoring, we equipped XPlan+ with an event listener for distinguished classes of events (cf. figure 3).

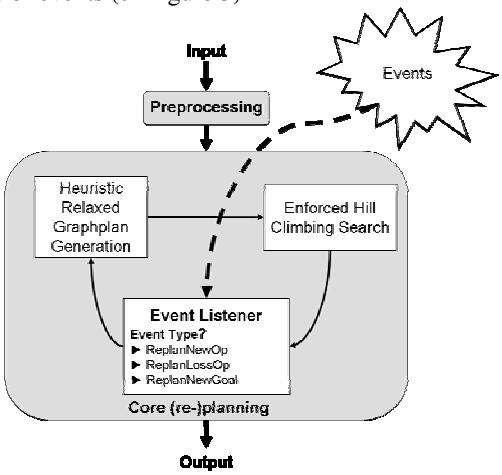

Fig. 3: Modified planning module XPlan+

In each plan step i, before applying selected helpful action A to the state S_i, however, XPlan+ listens for events of state changes. If no events are in its event queue, it applies A to S_i and proceeds with plan step i+1. The plan fragment from initial state S_0 to S_i is correct and, due to the selection of helpful actions in the minimal relaxed plan, approximated optimal. XPlan+ triggers re-planning in the following cases of observed events of world state changes: (1) An operator (service) instantiation (action) becomes available. This is the case if (a) a new operator has been introduced, or (b) the world state (set of facts) changed such that an operator whose instantiation was impossible before can be instantiated now, or (c) new predicates which are part of the preconditions or effects of an operator are introduced, making it possible to instantiate this operator; (2) An operator (service) of the plan is not possible anymore, if any of the opposites of cases 1.a - 1.c holds; (3) The goal state changed due to a change of the original planning request. Each of these cases is handled separately as described in subsequent sections. If facts or objects change, it searches for the first operator which precondition is satisfied by the new fact, and starts re-planning from there, while the helpful actions get instantiated with the new fact(s). The case in which a predicate p( ) changes can be reduced (a) to the latter case of changed facts, if new facts are added; (b) to the case of

change of operator o, if preconditions or effects of o include p( ); or (c) to the case of fact changes, if the deletion of p() implies the deletion of all instances of p( ). It is assumed that the planning state consistency is checked by means of an appropriate module as intergal part of both XPlan and XPlan+.

3.1. Case of new operator

If a new operator (service) becomes available, XPlan+ first checks whether re-planning might yield a shorter plan by comparing the old with the newly generated relaxed plan. If positive, it heuristically determines the point in the plan where the new operator might first be helpful to start the re-planning from there.

Re-planning decision. XPlan+ first uses the same initial state as for the original (partial) plan P plus the new operator o to build the relaxed plan graph RPG, and extract a new relaxed plan RP'. Second, it estimates the length h(RP') of RP' by applying the same relaxed plan length heuristic as done for determining h(RP), that is the sum of all actions in all action-layers of the RP'. It is the number of all (helpful) actions of RP' as solution paths in the RPG for the initial state. If h(RP') < h(RP) holds, it continues with re-planning. Otherwise no re-planning is performed.

Re-planning. How much of the old plan P can be reused for the new plan P', means what would be the best position to restart planning with the new operator o? In order to determine this position, XPlan+ heuristically takes the index of the layer of the new RP' at which o occurs first as position e, else (if o is not in RP') sets position e = -1 and stops, and retains the old plan (fragment) P until this position. In other words, the position e of starting re-planning is the minimal number of actions before first occurrence of o. Second, it applies all operators from the old plan occurring before e in the new plan, and then tries to identify the instances of o which are applicable to the current state. For this purpose it checks whether the precondition of o is satisfied in the RP'. If no instance of o is applicable, it tries to apply more operators from the old plan until an instance of o eventually becomes applicable. If this fails, a complete re-planning has to be started. Otherwise it applies the new operator o. That is, XPlan+ identifies a re-entry point in the original plan by searching for already planned operators (actions) which correspond to helpful ones in the current state, and continues with the first step from this position. If no such position can be found, start full re-planning of the plan. If the goal is not yet reached, extend the plan until the goal is reached by continuing with the normal planning process like XPlan.

3.2. Case of lost operator

If a planned operator becomes unavailable, the actual plan is invalid. XPlan+ tries to replace the affected operator(s) by replacing it with alternative ones which achieve the same effect as the lost one. In case of success, the remainder of the plan can be re-used, which reduces re-planning time significantly.

Re-planning decision. XPlan+ first marks all actions in the plan which are affected, because of the fact that the respective operator does not exist anymore, or some precondition does not hold anymore. If no actions are marked, it continues with the normal planning process like XPlan.

Re-planning. For each affected action, XPlan+ creates a relaxed plan RP' from S_0. It then uses (inverse) enforced hill climbing search to circumvent the affected operator by applying alternative operators if possible. Basically, the planner identifies a re-entry point in the old plan P by searching for already planned actions in P which correspond to helpful actions in the current state, and continues with re-planning from this position. If no such position can be found, the remainder of the plan has to be re-planned completely. Otherwise, if the goal is not yet reached, extend the plan until the goal is reached by continuing with the normal planning process like XPlan.

3.3. Case of new goal

If the given planning goal did change, re-planning is necessary in case the new goal cannot be satisfied by the current plan at all, or could even be achieved by a shorter plan.

Re-planning decision. XPlan+ quickly creates a relaxed plan for the new goal from the initial state S_0, and marks all actions in the already existing plan P which are also contained in the new relaxed plan.

Re-planning. For each non-marked action, XPlan+ uses enforced hill climbing search to circumvent the action by applying alternative operators, identifies a re-entry point in the old plan by searching for planned actions in the old plan P which correspond to helpful actions in the current state, and continues planning from this position. That is, XPlan+ starts heuristic re-planning with the action in currently valid P that precedes first occurrence of o. If no such position can be found, the remainder of the plan has to be re-planned completely. If the goal is not yet reached, XPlan+ extends the plan until it is reached by continuing with the normal planning process like XPlan.

4. Preliminary Evaluation

The comparative analysis of the computational complexity of planning with XPlan+ and XPlan, as depicted in figure 5, is concerned with situations where a new operator becomes available, or an operator is deleted just before the initial planning is finished (for plans with at least 20 steps). The denotation "Online(n)", with n =

0,1,2 refers to the case where an observed event does affect the plan at n positions. As a consequence, XPlan+ has to build n relaxed plans during its partial re-planning, whereas the pure off-line planner XPlan denotated in the figure as "Offline" would do a full re-planning.

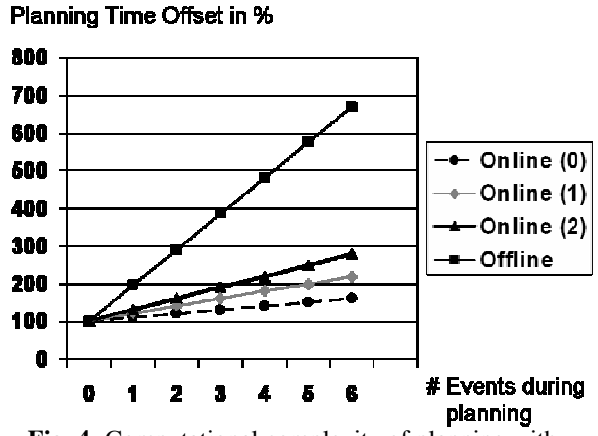

Fig. 4: Computational complexity of planning with XPlan+ (quasi.-)online vs. XPlan (full restart/offline)

The resulting planning time offsets for all cases applied to a simple blocks world related plan of 28 steps with initially five operators is shown in figure 4. Only new operators where introduced during the planning process that theoretically would lead to a shorter plan. XPlan+ gained more momentum compared to XPlan with increasing number of such events, and the later in the plan they did occur. This is also experimentally confirmed by the measured run time of XPlan+ (cf. figure 5) which decreased in absolute terms mainly due to its heuristic re-use of plan parts as described above.

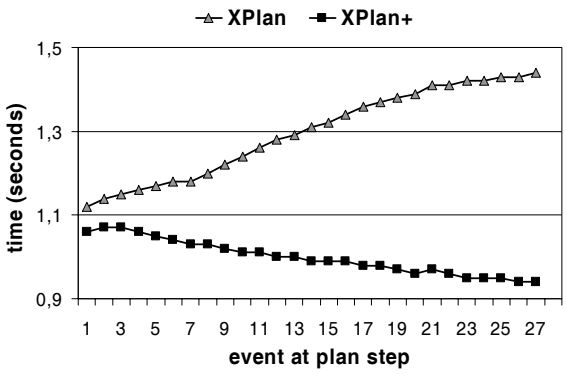

Fig. 5: Measured run time of XPlan+ vs. XPlan

5. Conclusion

We presented an extension of the planning module of our OWL-S service composition planner OWLS-XPlan, named XPlan+, that allows for quasi-online re-planning of composite OWL-S services with reasonable performance according to preliminary evaluation results. We are currently working on the integration of the implemented XPlan+ into OWLS-XPlan, and plan to make the resulting service composition planner OWLS-XPlan+ publicly available at semwebcentral.org

6. References

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

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

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

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

[5] 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.

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

[7] 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.

[8] D. Wu, B. Parsia, E. Sirin, J. Hendler, and D. Nau. Automating DAML-S web services composition using SHOP2. *Proceedings of the 2nd International Semantic Web Conference (ISWC2003), Florida, USA*, 2003.

[9] 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.

[10] L. Pryor, G. Collins. Planning for Contingencies: A Decision-based Approach. *Artificial Intelligence Research, 4:287-339,* 1996.

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

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

[13] R. Dearden et al.. Incremental Contingency Planning. Proc. Int. Conf. on Automated Planning and Scheduling ICAPS, Workshop on Planning under uncertainty and incomplete information, Trento, Italy, 2003