Efficient Source Discovery and Service Composition for Ubiquitous Computing Environments

Abir Qasem, Jeff Heflin, Héctor Muñoz-Avila
Dept. of Computer Science & Engineering
Lehigh University
19 Memorial Drive West
Bethlehem, PA 18015
{qasem, heflin, munoz}@cse.lehigh.edu

Abstract. To be truly pervasive the devices in a ubiquitous computing environment have to be able to form a "coalition" without human intervention. The Semantic Web provides the infrastructure for discovery and composition of device functionalities. AI planning has been a popular technology for automatic service discovery and composition in the Semantic Web. However, because the Web is so vast and changes so rapidly, a planning agent cannot make a closed-world assumption. This condition makes it difficult for an agent to know when it has gathered all relevant information or when additional searches may be redundant. To avoid redundancy we incorporate Local Closed World reasoning with HTN planning to compose Semantic Web services. In addition, when performing information gathering tasks on the Semantic Web, we use Local Closed World reasoning and a concept of "source relevance" to control the search process. We also describe a prototype agent that we have developed.

1 Introduction

A decade ago, virtually the only way to access the Web was through a personal computer or workstation. While this is still primarily true, the number of different kinds of device that can access the Web has grown astronomically and so have their capabilities. Smart phones, PDAs, interactive television systems, voice response systems, and even certain domestic appliances have all become regular user of the Web. The range of their capabilities for input and output and the number of markup languages and networks supported mean that it has become quite a challenge to provide a technological infrastructure that would support this new ubiquitous computing on the Web.

To achieve true pervasiveness as envisioned by Weiser [93], the devices have to become aware of each other and be able to form a "coalition" without human intervention Lassila & Adler [03]. For example, if no single device can match the user's need, then it should be possible to determine if there is a sequence of services offered by different devices, which solve the problem. This issue has led to the development of semantic web services. The OWL-S initiative is attempting to define how a web service can be described using OWL. Essentially, OWL-S is an ontology represented in OWL [OWL 04] that describes web services. This ontology focuses on three type of knowledge about a service: a profile of the service, a process model that describes how it works, and a description of how to invoke it.

A natural approach to automatic service composition is to view it as an AI planning problem [Peer 02] [Hendler 99]. This is natural because a service profile describes the inputs and outputs of services, which match with the preconditions and effects of planning operators. McIlraith *et. al.* [01] has proposed an approach to composing web services using Golog They distinguish between information gathering and world altering actions. Instead of using conditional plans, which result in a large search space, they execute information gathering actions as needed during planning and only include world altering actions in the final plan. Of course, the correctness of this approach depends on the assumption that any information that is gathered remains static until the plan is executed. Wu *et. al.* [03] make similar assumptions however, they use hierarchical task network (HTN) planning to compose services that are described using the DAML-S [DAML-S 03] ontology. The idea of a using a planner to automatically discover and compose services offered by numerous ubiquitous "semantic gadgets" (A term coined by Lassila & Adler [03]) may seem

seductive but there are several difficult technical hurdles that still need to be overcome before this becomes a reality. For example, we would need to develop mechanisms for reasoning on the relationship between the information-gathering search and the complex tasks being solved and introduce techniques that reduce and control the search effort of the information-gathering process, while simultaneously maintaining guarantees about the coverage of the search, when possible.

We would also have to address the fact that the information sources may use different schemas (i.e., represent the same domain differently) and / or data models (e.g. XML, relational databases, etc.). The highly distributed nature of ubiquitous computing and the large number of devices, combined with narrow bandwidth of a wireless environment requires that we locate the desired information without querying every possible source. This is critical for ensuring that network communication does not become overloaded. It is therefore critical that we choose a plan generation paradigm that addresses these challenges.

In our work we use HTN planning to compose Semantic Web services. To avoid redundancy in service composition we use Local Closed World reasoning (LCW). In addition when performing information gathering tasks on the Semantic Web we use LCW and a concept of "source relevance" that we introduce to control the search process. We developed an agent prototype, OntoPlan, which demonstrates the feasibility and potential of this research. The rest of the paper is organized as follows. In section 2 we describe HTN in details and show how we have incorporated it in OntoPlan, in section 3 we describe how LCW and relevance is expressed in OWL. In section 4 we describe the architecture of OntoPlan and its functionality. Finally in section 5 we provide some conclusion and future work.

2 Planning for Service Discovery and Composition

Sirin *et. al.* [04] have considered a semi-automatic service composition technique for a sensor network environment and Zhang *et. al.* [03] have used automatic service discovery by capturing user's expected outcome explicitly. The techniques are based on a pipe and filter approach. In this approach the individual services placed earlier in the composition should supply appropriate outputs to the following services in a coordinated assembly line fashion. This implies a total order in the composition of services and assumes a non-hierarchical nature of the devices providing the services. This implication and the assumption may not be very applicable in the ubiquitous computing environments.

For example, if we consider a car's On Board Diagnostic (OBD) system to be able to provide the car's distance from the closest Wal-Mart superstore, we observe the following:

- The service is a composition of several lower level services like getting the location of the car, accessing Wal-Mart's website (that provides a locator service). There is a natural hierarchy among the devices of a ubiquitous computing environment.
- The environment is partially observable and information is distributed
- The services can be composed in partial order. For example, the car's sensors do not have to wait for Wal-Mart's location to be available.

Similar to Wu *et. al.* [03] and as we also previously proposed in Heflin & Muñoz-Avila [02], we use HTN planning to compose semantic web services. HTN representations advocate a top-down view of the world. High-level tasks are refined into simpler tasks. By continuing the decomposition process, eventually non-decomposable tasks (called primitive tasks) are reached. These tasks correspond to actual actions that must be executed. HTN planning seems a natural match for web services composition whereby a top-down task refinement is performed linking the tasks to be achieved with the web services that need to be accessed to accomplish these tasks.

Heflin and Muñoz-Avila [02] have demonstrated how an HTN planner can exploit LCW information encoded in SHOE [Heflin *et. al.* 98] and DAML+OIL [DAML+OIL 00]. Our research builds on this work. We provide an OWL language syntax and semantics for LCW, introduce the idea of relevance statements, and describe a refined system architecture that has been implemented as a prototype system.

In what follows we first give a more detailed description of HTN and then describe how we have adopted LCW in an HTN planning situation.

2.1 HTN Planning

The knowledge artifacts in HTN Planning that describe how to decompose are called ***methods***. A method indicates the conditions that must be met to decompose a task into some subtasks. A fundamental characteristic of HTN planning is that it allows the formulation of high-level tasks to be accomplished and the encoding of high-level strategies (in the methods) before reasoning on low level tasks or concrete actions. This allows a stratified refinement of complex tasks into simpler ones. This flexibility has been crucial for the use of HTN techniques in solving real-world problems [Smith *et. al*. 98], [Currie & Tate 91], [Wilkins 90], [Paolucci *et. al*. 99]. We believe that it is crucial for ubiquitous computing environments, since adequate strategies will constrain the kinds of devices needed to be considered for accomplishing the task. In addition, it is provable that HTN planning is more expressive than operator-based representations [Erol *et. al*. 94]. We used a variant of HTN planning called Ordered Task Decomposition [Nau *et. al*., 99, Nau *et. al*. 01]. Ordered Task Decomposition (OTD) uses forward search and encodes control knowledge in the HTNs. Combining forward search and control knowledge has been the key characteristic for the surprisingly efficient planning systems showcased in recent planning competitions [Bacchus 2001].

Formally, decomposable tasks are called ***compound***, while non-decomposable tasks are called ***primitive***. A ***domain theory*** consists of methods and operators for generating plans. A ***method*** is an expression of the form $M=(h,P,ST)$, where h (the method's ***head***) is a compound task, P is a set of ***preconditions***, and ST is the set of M's (children) ***subtasks***. M is ***applicable*** to a task t, relative to a ***state*** S (a set of ground atoms), iff matches(h,t,S) (i.e., h and t have the same predicate and arity, and a consistent set of bindings _ exists that maps variables to values such that all terms in h match their corresponding ground terms in t) and the preconditions P are ***satisfied*** in S (i.e., there exists a consistent extension of _, named _', such that $\forall p \in P\ \{p\_' \in S\}$), in which case $M(t,S)=ST\ \_'$.

Since the environment is partially observable, the agent's operators can no longer depend directly on the state of the world, but instead must depend on the agent's beliefs about the world. The agents will also need special information gathering actions that can provide information about the state of the world. These are similar to the standard operators in OTD planning, except that the add list contains knowledge propositions of the form B(□). Standard operators (i.e., those with add lists that do not contain knowledge propositions) are called world-altering actions.

2.2 HTN and LCW

When choosing how to implement a particular task, the agent must test the preconditions of a method to determine its applicability in the current situation. Its knowledge base stores the background knowledge as well as any information it has already gathered. The agent will issue queries to this knowledge base in order to test the validity of the planning conditions. In order to reduce redundant access, the knowledge base keeps track of local completeness information in the form of LCW statements. LCW, as proposed by Golden *et. al*. [94], is a formalism for obtaining closed-world information on subsets of information that is known to be complete.

We can represent completeness by first order logic (FOL) formula of the form LCW (□), where the formulas contain variables. LCW (□) means that for all variable substitutions □, if □□ is true in the world state, then Agent_KB |= □□. Any matching ground sentence that Agent_KB does not entail, is assumed to be false. So for LCW (□), if a sentence matches a substitution for □ then it is either already entailed by the agent's knowledge base or it is false. In this sense, the sentence □ provides a scope for the relative completeness of the knowledge base. Note, that this information is local in the sense that it is local to the knowledge base that it describes.

The typical approach for planning with partial information is to have special information gathering actions that help the agent to learn about the world. However, we don't include such actions in the plan, instead the planner calls the Semantic Web mediator described in section 3. We believe this has two main advantages:

1) In general, it reduces the number of possible actions the agent has to choose from and as such reduces the overall search space, resulting in finding a solution more quickly [Ashish *et. al.* 97].
2) If information gathering actions (operators) were used, each type of query to an information source would have to be modeled as a separate action. When dealing with complex information sources, such as databases, there are an enormous number of queries that can be asked of the system, and thus it is unreasonable to treat them as actions.

Often the knowledge base will not have sufficient information to test the validity of a condition. In these cases, the information must be gathered from information sources. The Semantic Web mediator does this work. It is important to notice that world-altering actions are not executed while planning; instead the expected results are hypothesized in the knowledge base of the planner. In complex scenarios, information sources are queried for pieces of information, which are composed to solve other tasks that, in turn, accomplish the required tasks.

3 Reducing Information Gathering Operations

In order to control the search process during the information gathering phase we need to be able to identify sources that have relevant or complete information with respect to a query. If a source can express that it has relevant information with respect to a query we can choose to query it over other sources that does not express this information. In this way we can locate the desired information without querying every possible source. Duschka [97] has used a similar concept "view-minimality" to control the number of sources that are queried in an information integration scenario. Having relevant information, however, does not mean that the source is capable of answering the query completely. It just says that the source has some useful information on the query.

We will need to query other sources (that also have relevant information) and integrate the results to be able to get a complete response. If a source can express that it has complete information with respect to a query then there is no need to continue our information gathering task any further. When we have complete information on a query it also allows us to reason with overlapping contents of various sources. Overlap of content is a common phenomenon in ubiquitous computing environments, e.g. we may have several sensors scanning one specific environment. We can use completeness information to select an optimal collection of sources that need to be queried in order to form a response. Levy [96] extended LCW (described in section 2.2) to obtain complete answers from databases that have incomplete information. His work can be adapted to express completeness information on ubiquitous computing environments.

In what follows, we show how we have adapted the representation of completeness and relevance information to the Semantic Web. We also provide some OWL examples of expressing source relevance and completeness on the Semantic Web.

3.1 Completeness and Relevance Formalisms for the Semantic Web

FOL formulas that we have used to express LCW in the planner cannot be directly adapted to the Semantic Web. OWL, the de facto standard for the Semantic Web is closer to Description Logic (DL) rather than FOL. To represent LCW using OWL one has to express the formulas in DL. Unlike FOL, DL does not allow us to use variables to refer to unknown objects. DL has notation to express definitions and properties of classes of objects. Classes provide an abstraction mechanism for grouping objects with similar characteristics. OWL classes are described through "class descriptions". Hence we have to express LCW for a class description, which will mean that we have LCW over all the instances of that class. For a given class *C,* if we now consider LCW (*C*) as DL class expressions then we say for a device i with associated

knowledge base KB_i, LCW_i (*C*) indicates that for all x, if KB_i does not entail that x is an instance of *C*, then x is in the complement of *C*. Formally, we can say $\forall x\ KB_i \not\models C(x) \Rightarrow KB_i \models \neg C(x)$.

A source needs to express that it contains relevant information with respect to a query when it has some (but not all) useful information to respond to a query. This allows us to locate the desired information without querying every possible source. Informally, we define the relation $REL_s(q)$ to indicate if an information source *s* has relevant information about a query *q*. In DL we will have to express this information by stating that a source has relevant information on some instance of a class. For a given class *C*, if we now consider REL (*C*) as DL class expression then we say for a device i with associated knowledge base KB_i, REL_i (*C*) indicates that there exists an instance in KB_i of class *C*. Formally, $\exists x\ KB_i \models C(x)$.

3.2 Expressing REL and LCW in OWL

To represent above in OWL we propose that an OWL document can use new properties lcw: isCompleteFor and lcw: isRelevantFor to state that it has complete or relevant information on some subset of information respectively. These properties are in a new namespace identified by the lcw prefix, and have rdf:Resource in its domain and owl:Class in its range. As such, it can be applied to any resource.

Since OWL has features for composing complex class expressions it is possible to use this method to represent LCW on any binary FOL atom with at most one variable. The following examples show how to apply these properties to represent LCW on various binary atoms. REL statements will essentially work in similar way. It is worth mentioning that OWL-S essentially allows the description of a service that accepts a handful of predefined queries. Our approach allows us to manage rich information sources that allow complex ad hoc queries.

For a given property p, class c, and some individual x we use the following to represent LCW (p (x, c)) on source s.

```
<rdf:Description rdf:about="s">
     <lcw:isCompleteFor>
         <owl:Restriction>
             <owl:onProperty rdf:resource="#p" />
             <owl:hasValue rdf:resource="#c" />
         </owl:Restriction>
       </lcw:isCompleteFor>
 </rdf:Description>
```

Here `isCompleteFor` is applied to an individual of a class that has at least one of its property values equal to the resource c.

It is somewhat difficult to represent complete information on an object's values for a specific property. So to represent LCW (p (c, x)), we create an anonymous property that is the inverse of p, and restrict the value of the inverse (essentially restricting the value of the subject of p). This is shown below:

```
<rdf:Description rdf:about="s">
     <lcw:isCompleteFor>
         <owl:Restriction>
             <owl:onProperty>
              <rdf:Property>
                 <owl:inverseOf rdf:resource="#p" />
                  </rdf:Property>
             </owl:onProperty>
             <owl:hasValue rdf:resource="#c" />
         </owl:Restriction>
     </lcw:isCompleteFor>
 </rdf:Description>
```

4 OntoPlan

We have developed a prototype planner agent, which demonstrates the techniques of efficient discovery and composition of Semantic Web Services we have proposed above. Figure 1 below shows the main components and the environment of OntoPlan.

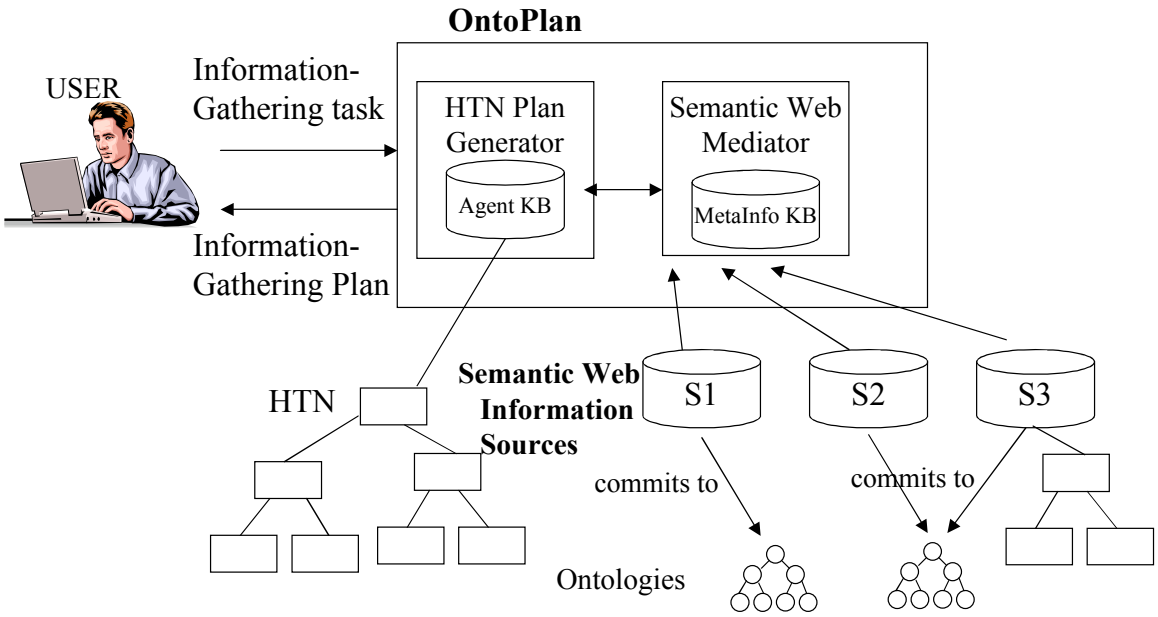

Fig. 1: Architecture of the OntoPlan

OntoPlan has two main components. The first is the HTN Plan Generator, which performs HTN planning to generate a plan that represents a service composition. When choosing how to implement a particular task, the system must test various planning conditions. It has a knowledge base, Agent KB, which stores the background knowledge as well as any information it has already gathered. The HTN Plan Generator will issue queries to Agent KB to test the validity of the planning conditions. In order to reduce redundant access, Agent KB keeps track of local completeness information in the form of LCW statements. However, often Agent KB will not have sufficient information to test the validity of a condition. In these cases, the information must be gathered from information sources available in the Web.

The second component, the Semantic Web mediator, provides an interface to the information sources. The information sources are devices in the ubiquitous computing environments. The information is distributed and commits to different ontologies and some of the sources may contain distributed pieces of the HTN. In what follows, we first describe the architecture of the Semantic Web mediator and give an overview of its components. We then describe how the HTN plan generator and the Semantic Web mediator interact to provide the functionality for OntoPlan.

4.1 Semantic Web Mediator

The Semantic Web mediator is based on the Common Integration Architecture (CIA) proposed by Wiederhold [97]. Figure 2 below shows the architecture of the Semantic Web mediator.

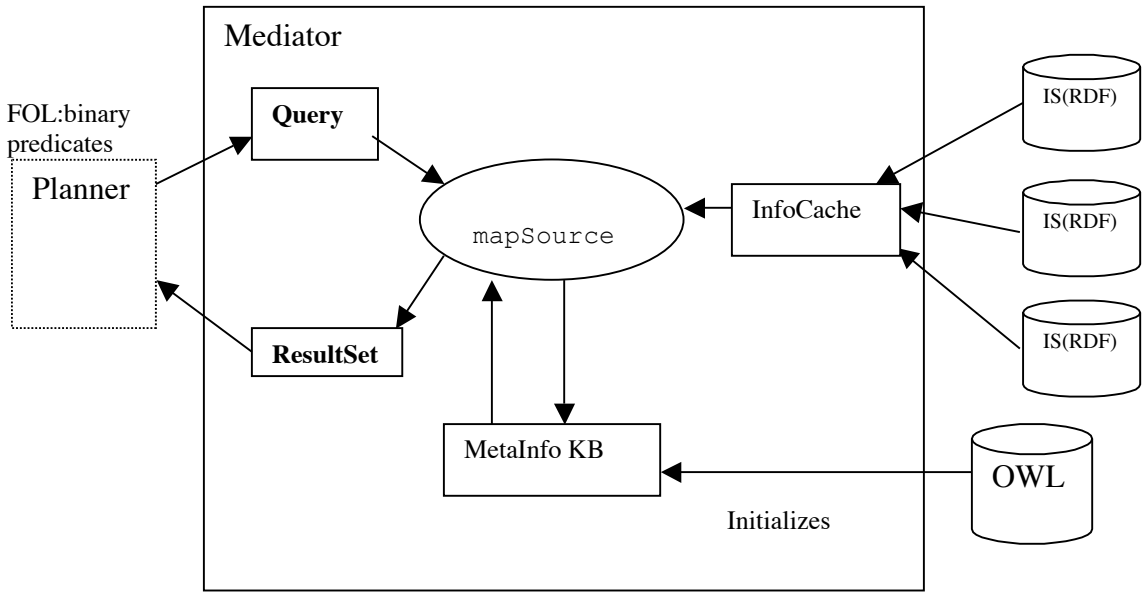

Fig. 2: Architecture of the Semantic Web Mediator

The main objective of the Semantic Web mediator is to identify appropriate information sources with respect to a query. To accomplish this task, the Semantic Web mediator uses and maintains information about the information sources available. The information sources are wrapped with an RDF interface. Its knowledge base, MetaInfo KB, contains two kinds of meta-information: completeness and relevance of information sources with respect to queries. It is currently initialized from an OWL file that has this meta-information encoded using the proposed `isComleteFor` and `isRelevantFor` OWL properties described in section 3. We plan to augment this with an ability to dynamically update the MetaInfo KB using data from specialized pages and/or meta information that accompanies newly discovered sources. The function `mapSource` uses this meta information and the query to determine the sources that can provide a reasonable response.

The queries in our system are represented in FOL while the Semantic Web is an information space described in DL. So, to process a query in the Semantic Web, we have to map the FOL representation to a DL representation. In Section 3 we have described how LCW and REL statements expressed in FOL are mapped to OWL. The queries are translated in a similar fashion. For example an FOL query location (PDA, x) maps in the RDF/OWL documents as follows: location is a property, PDA is the subject and x is the object. So we find every triple that has the property location and the subject PDA. Then the object of each of those triples is returned as a binding for x. As we only support a binary representation for class description in our queries and the predicate "type" is assumed to signify a class membership. For example, for a query `type (Bluetooth, x)` matches an RDF triple with property `rdf:type`, subject `Bluetooth` and unknown object, where `Bluetooth` is an `owl:Class`. Therefore, we find all triples that match this pattern and return each object as a binding for x. When we have complete information on the query we also set a flag to inform the plan generator so that it can update Agent KB with this fact.

The RDF pages that describe information sources are read only once in a "planning session". These are stored in InfoCache. In addition to speeding up the system by avoiding expensive http calls, it preserves a notion of consistency. This is critical in the ubiquitous environments where the information may change at a rapid rate. For example if a source said P when first queried, and then NOT P when queried the second time, there could be a problem if these were both used in the same plan.

OntoPlan is a proof of concept system that allowed us to examine the interaction of an HTN planner and a Semantic Web mediator. Specifically we wanted to study how source selectivity and overlap reasoning improve when we introduce local completeness and relevance meta-information about various information sources. In order to achieve this goal in a reasonable time we decided to make several simplifying assumptions in the system. In section 5, we discuss these assumptions, explain why we had to make them and propose how they can be removed in a deployable system

4.2 Functionality of OntoPlan

Given an HTN description, OntoPlan chooses how to implement it by testing the preconditions of every method to determine its applicability in the current situation. The precondition evaluation is described by the following pseudo code:

```
For all preconditions
   If a precondition is satisfied in Agent KB
         Return variable bindings satisfying precondition
   Else
         if LCW on precondition is contained in Agent KB
               precondition is false
         Else
               Call mediator with the precondition
         if mediator has responded with complete flag
               update Agent KB
End For
```

The precondition passed onto the mediator as the query that the mediator tries to find a result for. The mediator selects the information sources to be queried based on the LCW and REL information that it has in its MetaInfo KB. The source selection is done using the following method:

```
If MetaInfo KB has LCW on query
   select and query that source
         return result and a completeness flag
Else
   find all sources in MetaInfo KB that has REL on the query
   select and query those sources and return result
```

When LCW information is not available, the mediator may have to search every single relevant data source. In practice, this may be limited by resource-bounded constraints such as time limits or maximum number of access.

5 Conclusion and future work

As mentioned before OntoPlan is a proof of concept system and is very much a work in progress. We made several simplifications that allowed us to build a system quickly and examine the effect of introducing meta-information about local completeness and relevance in a service composition and mediation scenario in the Semantic Web. In this section we discuss these simplifications and propose how they can be removed in a deployable system.

Our first simplification is to restrict the queries to the Semantic Web Mediator to binary atoms that contain at most one variable. This allowed for a set of bindings to be determined from a single source in isolation. There was no need to query multiple sources to get a single answer. For similar reasons, the determination of the completeness (and relevance) of a query needs to be done with respect to sources in isolation. For example, had we allowed a query "foo(A,x) AND bar(A,y)" then the answer may have had to come from combining two sources, one with "foo(A,B)" and the other with "bar(A,C)". One would also have needed one source with "LCW(foo(A,x))" and another with "LCW(bar(A,x))" to determine the completeness of the query. However, by limiting the query to a single atom, one reduces the determination of completeness of a query to a simple matter of pattern matching.

Although this assumption allows us to observe source selectivity it does not allow us to reason with overlapping contents. We realize, this is a serious limitation and we plan to address this in our next version of the system.

Our second simplification is in the initialization of the Semantic Web Mediator's knowledgebase, MetaInfo KB. Currently we initialize the knowledgebase from a static OWL file. This implies that the LCW and the REL information are known beforehand. This scenario does not reflect the real Semantic Web by any stretch of the imagination, but it does not affect our analysis of source selectivity or overlap reasoning either. The system could easily be extended to accept new metadata on additional sources when they are discovered. For example, the sources can themselves have metadata about the local completeness of their contents. In such cases we have to decide if given a set of local completeness statements, is a query Q□a complete answer to Q [Knoblock and Kambhampati 02]. We can generate relevance statements by analyzing the content of a source. Since the relevance basically states that we have some but not all information on a topic, we can use a generalization process to develop a set of representative concepts for each source we discovered. Conceptually this is similar to key word generation process of HTML Web pages.

Our third simplification is to assume that all sources commit to a single ontology. This allowed us to postpone work of the well-known and difficult ontology alignment problem. Clearly, if the sources commit to different ontologies we will have to provide some form of translation mechanism. Possible approaches include using a reasoner to provide alignment axioms during the query process, directly translating queries and sources into a global merged ontology, or translating queries at runtime into the vocabularies of all possible sources. An additional problem occurs in determining relevance and local completeness of sources with respect to queries. If the meta-descriptions and the query are heterogeneous, then we have to address these problems as well. We plan to augment OntoPlan's capability to achieve this in the future.

Our immediate planned change for OntoPlan is as follows:

- Give the Semantic Web Mediator the ability to answer queries with any number of atoms
- Design a system that computes the relevance information of data sources through analysis of their contents.
- Allow data sources to commit to heterogeneous ontologies.

The above amount to removing all the simplifications that we have made in our system. In the future we would also like to extend OntoPlan with the following:

- Redesign the system for scalability with respect to the amount of data that must be accessed and processed
- Express the domain knowledge used by the HTN planner in an OWL ontology. HTNs can be seen as an ontology defining relations between tasks and the resources needed to accomplish them
- Allow the HTN domain knowledge descriptions to be distributed across data sources.

These changes will enable OntoPlan to operate with partial domain knowledge and to augment its knowledge while achieving its tasks.

In this paper we have described a system, OntoPlan, which uses HTN planning to compose Semantic Web services. To avoid redundancy in service composition it uses Local Closed World reasoning. In addition when performing information gathering tasks on the Semantic Web it uses LCW and a concept of "source relevance" to control the search process. Although this is a proof of concept system it demonstrates the potential of research in efficient source discovery and service composition using local completeness and relevance information.

References

Ashish, N., Knoblock, C.A., and Levy, A. Information gathering plans with sensing actions, In Sam Steel and Rachid Alami, editors, Recent Advances in *AI Planning: 4th European Conference on Planning, ECP'97*. Springer-Verlag, New York, 1997.

Bacchus, F. The AIPS'00 planning competition. *AI Magazine*, 22(3), 47-5

Currie K., and Tate, A. O-Plan: The Open Planning Architecture. *Artificial Intelligence*, 52:49 – 86, 1991

DAML-S (and OWL-S) 0.9 Draft Release retrieved June 30th, 2004 from http://www.daml.org/services/daml-s/0.9/

DAML+OIL (March 2001) retrieved June 30th, 2004 from http://www.daml.org/2001/03/daml+oil-index.html

Duschka, O. 1997. Query Optimization using Local Completeness. In *Proc. of AAAI-97*.

Erol, K., Hendler, J., and Nau, D.S. UMCP: A Sound and Complete Procedure for Hierarchical Task-Network Planning. *In Proc. of AIPS-94*.

Friedman, M., and Weld, D. 1997. Efficiently Executing Information Gathering Plans. In *Proc. of IJCAI-97*.

Golden, K., Etzioni O., and Weld, D. 1994. Omnipresence Without Omniscience: Efficient Sensor Managment for Planning. *In proc. of AAAI-94*.

Heflin, J., Hendler J., and Luke S. Reading Between the Lines: Using SHOE to Discover Implicit Knowledge from the Web. 1998. In *AI and Information Integration. Papers from the1998 Workshop. WS-98-14*. AAAI Press, Menlo Park, CA, 1998. pp. 51-57.

Heflin, J., and Munoz-Avila, H. 2003. LCW-Based Agent Planning for the Semantic Web. In *Ontologies and the Semantic Web. Papers from the 2002 AAAI Workshop WS-02-11*. AAAI Press, Menlo Park, CA, 2002. pp. 63-70.

Hendler, J. Is There an Intelligent Agent in Your Future? In *www.nature.com* retrieved June 15th, 2004 http://www.nature.com/nature/webmatters/agents/agents.html/

Knoblock C., and Kambhampati S. 2002. Tutorial on Information Integration on the Web. In Eighteenth National Conference on Artificial Intelligence.

Lassila O., & Adler M. "Semantic Gadgets: Ubiquitous Computing Meets the Semantic Web", in: Dieter Fensel et al (eds.): " Spinning the Semantic Web ", pp.363-376, MIT Press, 2003

Levy, A. 1996. Obtaining Complete Answers from Incomplete Databases. In *Proceedings of the 22'nd VLDB Conference*.

Mcllraith S.A., Son T.C., and Zeng H. Semantic Web Services. In *IEEE Intelligent Systems, Special Issue on the Semantic Web*, Volume 16, No. 2, pp. 46-53.

Nau, D., Cao, Y, Lotem, A., and Muñoz-Avila, H. 1999. SHOP: Simple Hierarchical Ordered Planner. *In Proceedings of IJCAI-99.*

Nau, D., Muñoz-Avila, H., Cao, Y., Lotem, A., and Mitchell, S. 2001. Total Order Planning with Partially Ordered Subtasks. *In Proceedings of IJCAI-2001*.

OWL Web Ontology Language Guide, retrieved March 15th, 2004 from http:// http://www.w3.org/TR/owl-guide/

Peer J. Bringing Together Semantic Web and Web Services. *International Semantic Web Conference* 2002 : 279-291

M. Paolucci, D. Kalp , A.S. Pannu, Shehory, O.,and Sycara K. 1999. A Planning Component for RETSINA Agents. *Lecture Notes in Artificial Intelligence, Intelligent Agents* VI , 1999.

World Wide Web Consortium Issues RDF and OWL Recommendations, retrieved March 15th, 2004 from http://www.w3.org/2004/01/sws-pressrelease

Zhang, R., Budak Arpinar I., Aleman-Meza B., Automatic Composition of Semantic Web Services ,*The 2003 International Conference on Web Service (ICWS'03)* , Las Vegas NV, 2003

Sirin, E.. Parsia, B., and Hendler, J.. Composition-driven filtering and selection of semantic web services. In *AAAI Spring Symposium on Semantic Web Services*, 2004.

Smith S. J., Nau, D., and Throop T.. Computerbridge: A big win for AI planning. *AI Magazine* 19(2), 93-105, 1998

Weiser M. Some Computer Science Problems in Ubiquitous Computing. *Communications of the ACM,* July 1993

Wilkins. D., Can AI planers solve practical problems? *Computational Intelligence* 6(4) 232-246, 1990

Wu, D., Parsia, B., Sirin, E., Hendler, J., and Nau, D. Automating DAML-S web services composition using SHOP2. In *Proceedings of 2nd International Semantic Web Conference (ISWC2003)* , Sanibel Island, Florida, October 2003.