HotBlu - A system for large scale service discovery and composition

Ion Constantinescu, Boi Faltings, and Walter Binder

Artificial Intelligence Laboratory
Swiss Federal Institute of Technology Lausanne
{ion.constantinescu,boi.faltings,walter.binder}@epfl.ch
http://liawww.epfl.ch

Abstract. It has been widely recognised that matchmaking is an important component for environments populated with heterogeneous services. Several researchers have developed powerful techniques for the matchmaking problem in general. There are also specific representation of service capabilities such as OWL-S which provides a more specific framework for matchmaking.
Most approaches to matchmaking have assumed a sequential search for a service with matching capabilities. This may become intractable when the number of available services gets large.
Our contribution consist in an integrated directory and planning system specially adapted to large scale service discovery and composition.[1]

Keywords: Service description, service discovery, matchmaking, directories, indexing.

1 Overview and demo scenario

Our approach to automated service composition is based on matching input and output parameters of services using type information in order to constrain the ways how services may be composed [4]. Our composition algorithm allows for *partially matching* types and handles them by computing and introducing *switches* in the integration plan. Experimental results show that using partial matches significantly decreases the failure rate [4, 5] compared with an integration algorithm that supports only complete matches (e.g., like for example [8]).

We have developed a directory service with specific features to ease service composition. Queries may not only search for complete matches, but may also retrieve *partially matching* directory entries [7, 6, 3]. As the number of (partially) matching entries may be large, the directory supports *incremental retrieval* of the results of a query. This is achieved through *sessions*, during which a client issues queries and retrieves the results in chunks of limited size [2, 1].

For demo purposes we consider a scenario where an personal agent (PA) has to organise an evening out for it's owner. The PAs task will be to find a cinema running a

[1] The work presented in this paper was partly carried out in the framework of the EPFL Center for Global Computing and supported by the Swiss National Science Foundation as part of the project MAGIC (FNRS-68155), as well as by the Swiss National Funding Agency OFES as part of the European projects KnowledgeWeb (FP6-507482) and DIP (FP6-507483).

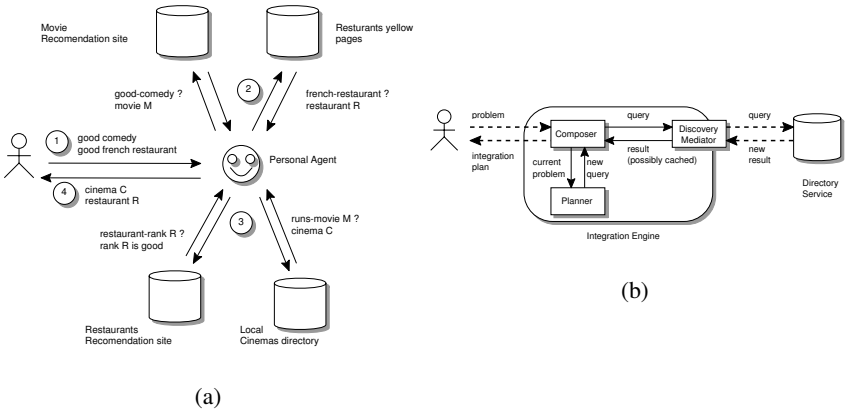

Fig. 1. (a) Planning an evening out. (b) The architecture of our service integration engine.

"good" comedy and a good french restaurant. For that the PA uses a cinema recommendations service for determining what are the good comedy movies today and yellow page directory for finding out what french restaurants are available. Then the PA tries to find a cinema which runs the selected comedy and uses a recommendation service to determine the ranking of the selected restaurant. Finally it returns a restaurant/cinema combination to the user.

The HotBlu engine is a research prototype and is not currently publically available but we aim to release it in the next months as open source.

2 Discovery and Composition

In this section we will present algorithms for computing type-compatible service compositions. Their design is motivated by two aspects specific to large scale service directories operating in open environments:

- **large result sets** - for each query the directory could return a large number of service descriptions.
- **costly directory accesses** - being a shared resource accessing the directory (possibly remotely) will be expensive.

We address these issues by interleaving discovery and composition and by computing the "right" query at each step. For that, the integration engine (see Fig. 1 (b)) uses three separate components:

- **planner** - a component that computes what can be currently achieved from the current query using the current set of discovered services. From that the problem that remains to be solved is derived and a new query is returned.
- **composer** - a component that implements the interleaving between planning and discovery. It decides what kind of queries (partial/complete) should be sent to the directory and it deals with branching points and recoursive solving of subproblems.

– **discovery mediator** - a component that mediates composer accesses to the directory by caching existing results and matching new queries to already discovered services.

2.1 Composition with complete type matches

Composing complete matching services using forward chaining is straightforward: once the condition for complete type matches is fullfilled (all inputs required by the service S are present in the query Q and the types in the query are more specific than the types accepted by the service) a new query Q' can be computed by adding to the set of available inputs of the current query Q all the outputs provided by the service S.

2.2 Composition with forward partial type matches

Conceptually the algorithm that we use for composing services with forward partial type matches has three steps:

– Discovery of complete matching services (see above Section 2.1).
– Discovery of services for full coverage of available inputs.
– Discovery of services for correct switch handling.

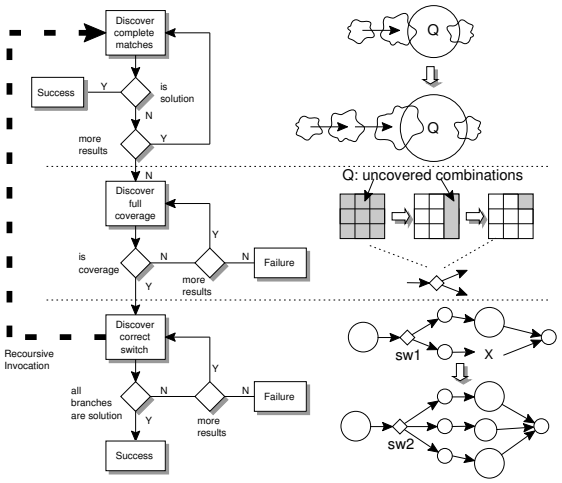

Fig. 2. Flow of algorithm for composition with partial type matches.

Discovering full input coverage The second step of the algorithm assumes that a solution using only complete matches was not found and that services with partial type matches have to be assambled in order to solve the problem. By definition any of the partially matching services is able to handle only a limited sub-space of the values available as inputs. In order to ensure that any combination of input values can be handled, the space of available inputs is first discretized in parameter value cells. One cell is a rectangular hyperspace containing all dimensions of the space of available inputs but

only a single interval for each dimension. A cell corresponds to the guard condition of the switch. Cells are built in such a way that any of the required inputs for the retrieved partially matching services could be expressed as a collection of cells. Each of the retrieved partially matching services is assigned to the cells that it can accept as input. The coverage is considered complete when all cells have assigned one or more services. When all cells are covered the algorithm proceeds at the next step. If no more partially matching services can be found and a complete coverage was not achieved the algorithm returns failure.

Discovering solution switch The last step of the algorithm assumes that a coverage was found and a first switch can be created. The goal of this step is to ensure that the switch will function correctly for each of its branches. For each cell and its set of assigned services the algorithm will compute the set of output parameters that those services will provide. Then a new query is computed, having as available inputs the output parameters of the cell and as required outputs the set of required outputs of the complete matching phase. The whole composition procedure is then invoked recursively. In the case that all cells return a successful result the switch is considered to be correct and the algorithm returns success. Otherwise a new service is retrieved and the process continues. When no more services can be retrieved the algorithm returns failure.

References

1. Ion Constantinescu, Walter Binder, and Boi Faltings. An Extensible Directory Enabling Efficient Semantic Web Service Integration. In *3rd International Semantic Web Conference (ISWC04)*, Hiroshima, November 2004.
2. Ion Constantinescu, Walter Binder, and Boi Faltings. Directory services for incremental service integration. In *First European Semantic Web Symposium (ESWS-2004)*, Heraklion, Greece, May 2004.
3. Ion Constantinescu and Boi Faltings. Efficient matchmaking and directory services. In *The 2003 IEEE/WIC International Conference on Web Intelligence (WI'03)*, Halifax, Canada, October 2003.
4. Ion Constantinescu, Boi Faltings, and Walter Binder. Large scale, type-compatible service composition. In *IEEE International Conference on Web Services (ICWS-2004)*, San Diego, CA, USA, July 2004.
5. Ion Constantinescu, Boi Faltings, and Walter Binder. Type-based composition of information services in large scale environments. In *The 2004 IEEE/WIC/ACM International Conference on Web Intelligence (WI'04)*, Beijing, China, September 2004.
6. Lei Li and Ian Horrocks. A software framework for matchmaking based on semantic web technology. In *Proceedings of the 12th International Conference on the World Wide Web*, 2003.
7. Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, and Katia Sycara. Semantic matching of web services capabilities. In *Proceedings of the 1st International Semantic Web Conference (ISWC)*, 2002.
8. Wu, Dan and Parsia, Bijan and Sirin, Evren and Hendler, James and Nau, Dana. Automating DAML-S Web Services Composition Using SHOP2. In *Proceedings of 2nd International Semantic Web Conference (ISWC2003)*, 2003.