

Faculty of Informatics

Faculty of Informatics - Papers

University of Wollongong

Year 2006

HCLP Based Service Composition

Y. Guan*

A. Ghose†

Z. Lu‡

*University of Wollongong, yguan@uow.edu.au

†University of Wollongong, aditya@uow.edu.au

‡University of Wollongong, lu@uow.edu.au

This article was originally published as: Guan, Y, Ghose, AK & Lu, Z, HCLP Based Service Composition, 2006 IEEE/WIC/ACM International Conference on Web Intelligence and International Agent Technology Workshops (WI-IAT 2006 Workshops), Hong Kong, China, December 2006, 138-141. Copyright IEEE 2006.

This paper is posted at Research Online.

<http://ro.uow.edu.au/infopapers/479>

HCLP Based Service Composition

Ying Guan, Aditya K. Ghose, Zheng Lu

Decision Systems Laboratory
School of IT and Computer Science
University of Wollongong, Australia
{yg32, aditya, lu}@uow.edu.au

Abstract

A key impediment to the widespread adoption of web services is the relatively limited set of tools available to deal with Quality-of-Service (QoS) factors. QoS factors pose several difficult challenges in how they may be articulated. While the functional requirements of a service can be represented as predicates to be satisfied by the target system, QoS factors are effectively statements of objectives to be maximized or minimized. QoS requirements occur naturally as local specifications of preference. Dealing with QoS factors is therefore a multi-objective optimization problem. In effect, these objectives are never fully satisfied, but satisfied to varying degrees. In evaluating alternative design decisions, we need to trade-off varying degrees of satisfaction of potentially mutually contradictory non-functional requirements.

One key contribution of this work is the use of the Hierarchical Constraint Logical Programming (HCLP) framework in dealing with functional requirements, business process rules and Quality of Service (QoS) factors. We show how functional requirements and business process rules can be defined as hard constraints, QoS factors can be formulated as soft constraints and how the machinery associated with constraint hierarchies can be used to evaluate the alternative trade-offs involved in seeking to satisfy a set of QoS factors that might pull in different directions. We apply also this approach to the problem of reasoning about web service selection and composition, and establish that significant value can be derived from such an exercise.

1 Introduction

A key impediment to the widespread adoption of web services is the relatively limited set of tools available to deal with Quality-of-Service (QoS) factors [7]. For instance, UDDI based look ups for web services are entirely based on the functional aspects of the desired services with quality factors playing no role. QoS factors encompass a wide range of non-functional attributes of a service such as capability, performance, reliability, integrity, security etc. [5]. Although much progress has been made over the past several years, it is widely acknowledged that dealing with QoS factors for services remains an important open question.

QoS factors pose several difficult challenges in how they may be articulated. While the functional requirements and business process rules of a service can be represented as predicates to be satisfied by the target system, QoS factors are effectively statements of objectives to be maximized or minimized and must be represented as such. Yet it is difficult and impractical to insist that QoS requirements be articulated by users as objective functions in the tradition of operations research techniques. QoS requirements occur naturally as local specifications of preference, and any robust approach to dealing with them must support such specifications. In evaluating alternative design decisions, we need to trade-off varying degrees of satisfaction of potentially mutually contradictory QoS factors. Dealing with QoS factors is therefore a multi-objective optimization problem.

According to [15], in large applications, it is common to mix business rules with the main business logic. A business rule can also be a statement that defines or constraints some aspect of the business. It is intended to assert business structure or to control the behavior of the business [14]. Business rules are usually expressed either as *constraints* or in the form *if conditions then action*. The conditions are also called *rule premises*. The business rule approach encompasses a collection of *terms* (definitions), *facts* (connection between terms) and *rules* (computation, constraints and conditional logic) [2]. *Terms* and *Facts* are statements that contain sensible business relevant observations, whereas *rules* are statements used to discover new information or guide decision making.

Our premise is that functional requirements and business rules can be modeled as goals or hard constraints, while QoS factors can be modeled as soft constraints in Hierarchical Constraint Logical Programming framework. The machinery of *constraint hierarchies* used in HCLP can be brought to bear on the service composition problem (and also on the problem of service selection). We require that those required business rules specified as goals in a HCLP rule, and QoS factors articulated as inequalities relating key system parameters to thresholds on their values.

This paper is organized as follows. In Section 2, we give a brief introduction to the HCLP framework. In Section 3, we present the HCLPWS framework. In Section 4, we talk about the application of HCLPWS for selection and branch and bound composition of web

service. Finally, we discuss related work in Section 5 and conclude in Section 6.

2 HCLP

HCLP which includes hard and soft constraints is an extension of CLP scheme. A HCLP problem consists of the domain of constraints and a comparator which is used to select among alternative ways of satisfying the soft constraints. The soft constraints in a HCLP problem are specified using Constraint hierarchies (CHs) which belong to traditional frameworks for the handling of over-constrained systems of constraints by specifying constraints with hierarchical preferences or strength. It allows one to specify not only hard constraints (the constraints that are required to hold), but also several preference levels of soft constraints (which violations are minimized level by level subsequently) at an arbitrary (finite) number of strengths [9]. To introduce the constraint hierarchies, we will use the definition of constraint hierarchies in [11].

A *constraint hierarchy* H is a finite set of labeled constraints. A *labeled constraint* is a constraint labeled with a strength, written lc where c is a constraint and l is a strength. A valuation for a set of constraints is a function that maps free variables in the constraints to elements in domain D over which the constraints are defined. A solution to a constraint hierarchy is such a set of valuations for the free variables in the hierarchy that any valuation in the solution set satisfies at least the required constraints. An error function $E(c\theta)$ is used to indicate how nearly constraint c is satisfied for a valuation θ . Major error functions are the predicate and metric error. In our model, we adopt the metric error function. The metric function is mainly adopted for arithmetic constraints composed of arithmetic functions and relations [2]. It expresses constraint errors as some distances. Typically, for arithmetic equality constraints, it uses the differences between the left- and right- hand sides. For example, the error of the constraint $x = y$ may be given as follows: $e("x=y", \theta) \equiv |\theta(x) - \theta(y)|$.

Constraint hierarchies define the so called comparators aimed to select solutions (the best assignment of values to particular variables) via minimizing errors of violated constraints. Currently, there are three groups of comparators: *global*, *local* and *regional comparators*. For a *local comparator*, each constraint is considered individually, for a *global comparator*, the errors for all constraints at a given level are aggregated using the combining function g . For a *regional comparator*, each constraint at a given level is considered individually. There are a number of comparators defined by combining function g and the relations $\langle \rangle_g$ and $\langle \rangle_g$ (the symbol $\langle \rangle$ means equal). All of these three comparators can be chose to measure CH solutions. In our model, we use the global metric comparator, which aggregate errors of violated constraints at each level. If a solution θ is better than a solution σ , there is some level k in the hierarchy such that for $1 \leq i < k$,

$g(E(H_i\theta)) \langle \rangle_g g(E(H_i\sigma))$, and at level k , $g(E(H_k\sigma)) \langle \rangle_g g(E(H_k\theta))$.

For the requirements of a service composition, we propose to use HCLP rules to specify them. An HCLP rule takes the form:

$$p(\mathbf{t}) :- q_1(\mathbf{t}), \dots, q_m(\mathbf{t}), l_1c_1(\mathbf{t}), \dots, l_nc_n(\mathbf{t}).$$

where \mathbf{t} is the list of terms, $p(\mathbf{t})$, $q_1(\mathbf{t})$, \dots , $q_m(\mathbf{t})$ are atoms and $l_1c_1(\mathbf{t}), \dots, l_nc_n(\mathbf{t})$ are labeled constraints. An HCLP program is a collection of rules.

For each service, it can be represented using HCLP rules. Functional requirements and business rules can be written like atoms $p(\mathbf{t})$, $q_1(\mathbf{t})$, $q_m(\mathbf{t})$ which will be treated as goals in CLP. These goals must be satisfied. For QoS factors which are quantifiable (i.e. execution time, cost, etc) we can use labeled constraints ($l_1c_1(\mathbf{t}), \dots, l_nc_n(\mathbf{t})$) to compose a constraint hierarchy.

In the next section, we will introduce a framework which can be used for service composition based on HCLP.

3 HCLPWS Framework

In this section, we will lay the groundwork for the application of HCLP in reasoning about business rules of web service and the QoS factors of web service. As discussed earlier, we shall use such reasoning to support service selection and service composition. Here we propose a general framework, the HCLP for Web service framework, which is applicable for web service selection and web service composition. This HCLPWS framework is an extension of the QoSCH framework we discuss in [13]. The QoSCH framework uses the constraint hierarchy for reasoning about the functional requirements and non-functional requirements of web services. It only uses Constraint hierarchy model of HCLP. HCLPWS deploys the full capability of the HCLP framework in dealing with functional requirements, business process rules and non-functional requirements services. This framework is in a high-level abstraction without considering a particular language, algorithm, platform and other factors in the process of service selection and service composition.

Different from standard web service architecture, we add a "Services HCLP Solver" in the architecture. This component consists of two sub-components: a HCLP Service Representation and a HCLP Interpreter. The scenario is: Services HCLP Solver accepts web services requirements from services requester and a set of services discovered by Discovery Engine, and then transforms those services requirements and descriptions into HCLP rules format. After all these preparations, HCLP Interpreter will work on those HCLP rules. The output of this component is an optimized services set.

HCLPWS framework that we presented in this section defines the following:

1. How business rules of the composite services are defined.

2. How service requirements (both functional and QoS) are defined.
3. The assumptions about services descriptions that must be satisfied for this framework to be applicable.
4. A measure of “distance” of a specific service from the “as-described”, i.e., the QoS requirements.
5. A \oplus operator which is associative to enable us to discuss service composition in abstract terms.
6. A \otimes operator to enable us to discuss the composition/ aggregation of the QoS factors of individual service to obtain QoS descriptions of composite services, in abstract terms.

The HCLPWS framework requires the following elements to be specified:

1. The business rules of the composite services and functional service requirements. These can be represented in constraint-based form (which would be treated as hard constraints) or as assertions in some other formal or even semi-formal language.

In HCLPWS, antipant service composition functional requirements and business rules will be modeled as goals or queries in HCLP rules, in the format of $q_1(\mathbf{t}), \dots, q_m(\mathbf{t})$. For example, functional requirement *flight ticket booking* and business rule “*if more than two persons travel together, the third one pays half flight price*” can be written as: `book_flight_ticket(N)`, `calculation_price(N, P)`, in which N and P are parameter terms that stands for number of person and price of tickets.

2. QoS requirements. These will be represented as a constraint hierarchy (which is $l_1c_1(\mathbf{t}), \dots, l_n c_n(\mathbf{t})$ in a HCLP rule), with each constraint relating a system parameter to a value, typically through an inequality for a parameter whose value one seeks to maximize, a constraint might be constructed requiring that the parameter in question be assigned the highest possible value, viewed as a soft constraint, this would oblige the system to assign to this parameter as high value as possible, even if the highest value cannot be assigned. Similarly, the minimization objective for a parameter could be represented by a soft constraint that seeks to assign to this parameter the lowest possible value.
3. An instance of the \oplus operator referred to above. The most common instance of this operator is sequential composition, but parallel composition and other control structure may also be of interest.
4. An instance of the \otimes operator referred to above. In general this is a commutative, associative operator that seeks to combine QoS descriptions of individual services to a QoS descriptions the composition service. For example, if processing speed is the only

QoS factor of interest in a setting involving sequential composition of services, then arithmetic sum would be the appropriate instance of \otimes . More generally, the operator may be viewed as a vector of concrete operators, one for each QoS factor. If two QoS factors, processing speed and a reliability measure, were of interest in a sequential composition setting, then would be the vector [sum, min, max] where sum would be used to aggregate processing speed (for obvious reasons), min would be used to aggregate reliability (a sequence of services is as reliable as the least reliable service in the sequence) and max would be to aggregate response time (when compose two parallel services).

5. A machinery for measuring the distance of a given service from a service requirements specification. We shall discuss this machinery in the rest of this section.

4. Apply HCLPWS for Service Composition

Web service composition is the ability of one business to provide value-added services through composition of basic web services, possibly offered by different companies [7]. A composite web service is an aggregation of web services which interact with each other based on a process model [4]. Web service standards, such as UDDI, WSDL, SOAP, do not deal with the composition of existing services. The industry solution for web service composition is using WSDL and BPEL4WS (a business protocol specification language proposed by IBM and Microsoft). Many researchers have worked on this problem ([10, 4, 6]).

In this section, we propose a branch and bound Services Composition technique that builds on the HCLPWS framework.

The branch and bound composition process consists of two steps:

1. Construct the HCLP model for each web service, requested one and available ones.
2. Find the first composite service that meets all functional requirements, business rules and hard constraints. This step will be done as in CLP and temporarily ignoring the non-required constraints which are quantifiable non-functional QoS properties. After the functional requirements, business rules and hard constraints have been successfully reduced, if the solution is still not unique, and then calculate the distance from the available constraint hierarchy to requested constraint hierarchy. Let the distance be d_i .
3. Try to construct another composite service for the same requirements. At each step, compare distance d with d_i , if $d > d_i$, then prune this branch.

When executing the web services composition, we

adopt the aggregation functions proposed in [4] for each step composition.

5 Related Work

There are many related works in this area. In [8], the author proposed a QoS model which offers a QoS certified to verify QoS claims from the web service suppliers. This approach lacks the ability to meet the dynamics of a market place where the need of both consumers and providers are constantly changing [12]. In [4], Zeng et al proposed a global planning approach to optimally select component services during the execution of a composite service. This proposed approach is quality-driven and by using Multiple Attribute Decision Making approach select optimal execution plan. This approach is not very efficient for large scale composite services, because it requires generating all possible execution plans, the computation cost is high. Whereas, our branch and bound based web services composition, improve the composition efficiency in great extent. In [1], Aggarwal et al propose a Constraint Driven Web Service Composition tool in METEOR-S in which composition conforms to the given constraints. In their approach, queries contain a collection of tuples of *features*, *weight*, and *constraints*. Similar to our work, their approach adopts constraint satisfaction problem to solve service composition problem. Our interest is in identifying alternative services where the deviation from given requirements is *minimal*. We would like to be able to use this measure of distance to rule out less viable compositions early in the process. The framework we present in this paper addresses all of these requirements. In [3], Channa et al propose to deal with the service composition as a constraint satisfaction problem. In their approach, they add a constraint optimizer in the process of service composition to find the optimal services set. Although their approach can deal with the business constraints and some QoS properties, our HCLPWS has an advantage that it will do some relaxation on QoS properties constraints (soft constraints) when the requested quality factors can not be satisfied. [14] is a research work about integrating business rules in service composition. They only focus on the business process and do not consider the non-functional QoS properties.

6 Conclusions

In this paper we proposed to use hierarchical constraint logical programming framework in dealing with service composition business rules and requirements. Hierarchical constraint logic programming (HCLP) was developed to deal with the fact that many of the constraints articulated by users in real-life problems are soft constraints. In the HCLPWS framework presented in this paper, functional requirements, business rules are represented as goals in HCLP rules, while QoS requirements

for a service are represented as constraint hierarchies, which permit local specifications of optimization objectives as well as local specifications of preferences amongst objectives. The constraint hierarchies approach permits us to use a well-founded notion of distance, both for service composition and selection. We use this notion of distance to define a branch-and-bound procedure for service composition. Currently, this work is still in progress, implementation of an Interpreter for HCLPWS and the selection of a suitable Web Ontology Language for specifications of constraint hierarchy of QoS factors will remain for future work.

References

- [1] Aggarwal R., Verma K., Miller J., Milnor W. "Constraint Driven Web Service Composition in METEOR-S", IEEE International Conference on Services Computing, 2004.
- [2] H. Hosobe. A foundation of Solution Methods for Constraint Hierarchies. Kluwer Academic Publishers, 2003.
- [3] Nizamuddin Channa, Shanping Li, Abdul Wasim Shaikh and Xiangjun Fu, Constraint Satisfaction in Dynamic Web Service Composition, Proceedings of the 16th International Workshop on Database and Expert Systems Applications, 2005
- [4] M. D. J. K. Liangzhao Zeng, Boualem Benatallah and Q. Z.Sheng. Quality driven web services composition. In Proceedings of the twelfth international conference on World Wide Web, pages 411–421, 2003.
- [5] D. A. Menasce. Qos issues in web services. IEEE Internet Computing, 6:72–75, 2002.
- [6] G. R.-G. Michael C. Jaeger and G. Mhl. Qos aggregation for web service composition using workflow patterns. In Proceedings of Eighth IEEE International Enterprise Distributed Object Computing Conference, 2004.
- [7] M. R. B. Paulo F. Pires and M. Mattoso. Building reliable web services composition. Lecture Notes In Computer Science, volume 2593, pages 59–72, 2002.
- [8] S. Ran. A model for web services discovery with QoS. ACM SIGecom Exchanges, 4:1–10, 2003.
- [9] H. Rudov. Constraint Satisfaction with Preferences, PhD thesis. Faculty of Informatics, Masaryk University, 2001.
- [10] B. Srivastava and J. Koehler. Web service composition current solutions and open problems. In Proceedings of International Conference on Automated Planning and Scheduling, 2003.
- [11] M. Wilson. Hierarchical Constraint Logic Programming, PhD Thesis. University of Washington May 1993.
- [12] N. Y. Liu, AHH and L. Zeng. Qos computation and policing in dynamic web service selection. In Proceeding of International World Wide Web Conference on Alternate track papers & posters, pages 66–73, 2005.
- [13] Y. Guan, A. K. Ghose and Z. Lu. Using Constraint Hierarchies to support Qos-guided service composition, In Proceedings of the International Conference on Web Service, 2006
- [14] Florian Rosenberg and Schahram Dustdar, Business Rules Integration in BPEL—A Service-Oriented Approach, Proceedings of the Seventh IEEE International Conference on E-Commerce Technology, 2005
- [15] D. A. Manolescu. Orchestration Patterns in Service Oriented Architectures. URL: <http://www.orchestrationpatterns.com/Orchestration-Patterns.html>, January 2005.