Constraints and Agents for a Decentralized Network Infrastructure

Jean Marc Andreoli
Uwe M. Borghoff
Remo Pareschi
Rank Xerox Research Centre, Grenoble Lab.
6, chemin de Maupertuis. F-38240 Meylan
<surname>@grenoble.rxrc.xerox.com

Stefano Bistarelli
Ugo Montanari
Francesca Rossi
Dipartimento di Informatica, Univ. of Pisa
Corso Italia 40, I-56125 Pisa.
{bista,ugo,rossi}@di.unipi.it

Abstract

The irreversible trend toward decentralized information sources makes less and less adequate the current approaches to search mostly based on a centralized index. The Constraint Based Knowledge Brokers (CBKB) model is a first attempt to provide an efficient decentralized search engine, and it does so by exploiting the power of constraints. One of the reasons of its efficiency comes from sending the information providers requests based on very specific and prefiltered information. This is done via a method which splits the scope of the broker agents as the requests come. However, an exaggerated use of scope splitting may lead to an unmanageable quantity of agents. Techniques to avoid such situations may be based on several approaches, which we briefly describe in this paper, like scope merging based on anti-unification, organizing agents in disjunctive structures, and limiting scope splitting on the basis of the capabilities of the knowledge bases.

Introduction

The need of adequate search capabilities for the growing bulk of on-line information is emerging as perhaps the most urgent challenge for computing at the end of the millenium. The inadequacy of the current generation of search engines becomes particularly evident by contrasting their strongly centralized approach with the irreversible trend toward decentralization of information sources. Take for instance the "father"[1] of all Web-related search engines, Alta Vista (Alta Vista URL). Moving from the Web domain to business applications implies coverage of heterogeneous sources such as catalogues, databases, on-line libraries etc. Alta Vista handles this problem by adding to its basic indexing machine some sort of brokerage service where information coming from multiple sources is homogenized for the user. It does so by attaching more database indexes to its existing index, with coverage of multiple file formats (Notes, HTML, MS etc). This is achieved, however, via a brute force, albeit smart, indexing policy, i.e. by packaging everything into a centralized index that is then searched against with state-of-the-art tools. Other providers of commercial information retrieval tools (e.g. Fulcrum and Verity) follow essentially the same strategy.

What we see is that, in reality, there are problems with processing speed, performance, and document relevance due to the impossible task of managing centrally a dynamically evolving federation of information providers. This applies even to the "simple case" of indexing Web pages. As an example, take again Alta Vista. It serves something like 20 million accesses per day (increasing every day), while its index becomes more and more outdated due to the rapid growth of the Web-pages to be gathered and indexed. To tackle this problem, Alta Vista now offers the possibility of sharing its technology through mirror sites in Europe and elsewhere (to increase revenues but also to distribute the load). Clearly, the load is split, but the gathering problem is still there. In a short period of time, the Web will become unmanageable with this kind of strategy. The problem gets only amplified if we move from the Web and the Internet to the world of corporate intranets. How is a centralized indexing policy for the search of information going to deal with distributed enterprises where empowered and autonomous local departments are responsible for the management of their own information repositories? How can it suit the requirements of "network" organizations, where economic actors can dynamically connect or disconnect from the network?

There are further issues that are not tackled well by today's search systems, e.g. where to search for a given query in a distributed world with thousands of accessible backends, and how to collect, fuse and present the answers from the selected backends. For the resource discovery problem, there are research attempts that work nicely if all information repositories follow the same retrieval model, such as the vector space retrieval

[1] At least in terms of coverage and penetration.

model (Gravano & Garcia-Molina 1995). This situation is, however, unlikely to arise in practice. Also, very few systems take retrieval costs (time, money) and quality of the returned documents into account. In research, however, early approaches are outlined (Fuhr 1997).

Constraint Based Knowledge Brokers

The Constraint Based Knowledge Brokers (CBKB) (Andreoli, Borghoff & Pareschi 1996) model is an early attempt to support search of distributed information in a decentralized manner by leveraging two innovative technologies: constraints and agents. The CBKB model has been used in several practical environments (see for instance (Borghoff *et al.* 1996; 1997)) and part of its technology is in the course of being transferred to a product development project. Our intent here is to describe both how it revolutionizes the concept of search in distributed, heterogeneous environments as well as how it still falls short of some the requirements for decentralized search. Before going into the details of this discussion, we would like however to make clear the use of the terms *constraints* and *agents* in the context of the CBKB framework:

- *Constraints* are used to flexibly define the behavior of broker agents collecting information from information servers on behalf of users. Thus, they provide a refinement on the client side of the basic client-server model of computation which corresponds to the reality of distributed computing nowadays. This refinement can go beyond simple search of information to account for transactional computations needed to support such applications as workflow management and electronic commerce; see for instance (Andreoli & Pareschi 1996) and references cited therein. First results concerning the methods of the refinement on the server side are discussed in (Chidlovskii, Borghoff & Chevalier 1997).

- *Agents* correspond here to simple information filters organized in topologies that can dynamically evolve through their interaction with an environment given by the users requesting information and the servers providing it. Thus, they are agents more from the point of view of artificial life than from the point of view of classical artificial intelligence: what counts is the collective intelligence obtained through their social relationships, rather than their own single capabilities.

The CBKB model tackles the issue of scalability of information gathering by partly following the approach taken by Harvest (Harvest URL), an earlier system that was not based on constraint technology. In Harvest, information is gathered from many (possibly millions) local Web administrators who have the task of supplying up-to-date information to the "Harvest gatherers". These in turn provide meta-index-information to "brokers." Thus, Harvest maintains a hierarchy of brokers that scales well. Similarly, in the CBKB model, brokers are responsible for a part of the world domain (scope) and can be distributed. They communicate through a simple "coordination" language (ForumTalk). Furthermore, by using constraints to describe the scope of search of the agents, the CBKB model radically extends the capabilities of Harvest. In fact, the possibility of dynamically refining constraint expressions can be leveraged in order to modify dynamically the hierarchy (or genetic tree) of brokers by keeping into account their interaction with the users requesting information (Andreoli *et al.* 1995). To use a biological metaphor, constraints act as a kind of DNA for describing the genotype of broker agents; this genotype can be optimally evolved according to the interaction with that part of the environment corresponding to the users accessing the information and constraints offer the means for coding genetic evolution.

The constraints used in CBKB to specify the scope of brokers are *signed feature constraints* (SFC) (Andreoli, Borghoff & Pareschi 1997). They are based on the intuition of providing the user with a simple constraint-based query language through which information requests can be formulated, and of accounting for changes in the topology of agents through an extended version of the same language.

Feature constraints (FC) were originally introduced in linguistics to represent lexical information, and are possibly complex constraints built from atomic constraints which are either sort or label constraints (Aït-Kaci, Podelski & Smolka 1994). A *sort constraint* expresses a property of a single entity. For example, `P:person` expresses that the entity `P` is of sort `person`. On the other hand, a *label constraint* expresses a property linking two entities. For example, `P:employer -> E` expresses that entity `P` has an employer, which is the entity `E`. Feature constraints can also contain *built-in relations* such as equality and disequality.

SFCs are obtained by taking the subset of FCs where disjunction is not allowed, and by adding signs (positive or negative) to express the fact that we want the entities to have a certain property or not. For example, the scope of a broker agent in charge of the domain of all books published in 1990 and whose author is not Balzac is represented by the following SFC constraint:

```
X
+ X:book
+ X: published -> P        P:1990
- X: author -> A           A:"Balzac"
```

The user requests are instead simpler SFCs where negation is not allowed (and not even signs, meaning that all properties are requested), called *basic feature constraints* (BFC). The mechanism by which the brokers' scopes are modified depending on the users' requests can be described by the following example: suppose we have an initial broker in charge of all books. That is,

```
X
+ X:book
```

Then a user request arrives for books by Honoré de Balzac:

```
X
X: book
X: author -> A        A:"Balzac"
```

This leads to a first branching in the genetic tree, with the creation of a specialist constrained to search for books by Balzac, and another agent in charge of all other books. That is,

```
X
+ X: book
+ X: author -> A        A:"Balzac"
```

and

```
X
+ X:book
-  X: author -> A        A:"Balzac"
```

Then another request arrives for books of more than 400 pages:

```
X
X: book
X: npage -> N        N>400
```

So the genetic tree evolves again with the creation of a specialist constrained to search for non-Balzac's books of more than 400 pages (the corresponding refined search for books by Balzac of more than 400 pages is handled by the previous Balzac-specialist through a simple projection into the constraint store, that is, a sifting of the relevant Balzac books):

```
X
+ X:book
+ X: npage -> N        N>400
- X: author -> A       A:"Balzac"
```

and a remaining agent now in charge for all other books, i.e. non-Balzac books with less or equal to 400 pages, etc.

```
X
+ X:book
- X: author -> A        A:"Balzac"
- X: npage -> N         N>400
```

Actually, constraints like

```
+ X: npage -> N        N>400
```

do not match the syntax of SFCs, which says that every relation (also >) is binary. However, in this paper we use them because they are equivalent and more intuitive than the correct ones. For example, the correct way to write the above constraint would be the following:

```
+ X: npage -> N        N > N'        N'>400
```

In theory, this approach based on scope splitting allows optimal use of network resources, by requesting the information providers with very specific, prefiltered information, as well as maximal concurrency and asynchronous behavior, thus avoiding "waiting lists" of requests. Furthermore, constraints offer a simple solution to the collection and presentation problem, i.e. how to merge results from different sites having different ranking algorithms in use and different corpus statistics. With the CBKB model, results can be ranked according to constraint satisfaction ratio, avoiding direct use of the non-comparable ranks coming from the backends.

In reality, this approach remains partly theoretical, due to the fact that just a few user interactions generate quickly an unmanageable proliferation of specialists; see for instance the complexity results in (Andreoli, Borghoff & Pareschi 1996). So, in practical implementation of the CBKB model, the opposite approach has been followed, with a fixed topology where, for each information domain, there is always a single agent in charge. The constraints are still used effectively, but only for filtering and ranking information coming from the backends, and for creating dependencies among agents responsible for different information domains. To overcome this situation, we need to complement genetic specialization with another feature of biological systems, namely the possibility of genetic cross-breeding, that would allow passing the capabilities of multiple specialists to single individuals, thus permitting a control on the size of the population of broker agents. From a computational point of view, this amounts to defining a sound method for dynamically combining brokers by finding the greatest lower bound of the constraints that define their scope of search. This combination should effectively merge genotypes by discarding unwanted specific genes, and thus should not trivially sum up constraints (e.g. through disjunctions). At the same time, it should maintain certain general properties of constraint expressions that are relevant for the CBKB model, such as stability of scope splitting (Andreoli, Borghoff & Pareschi 1997). These requirements lead to the program of research that we describe in the next section.

A program of research for constraint-based decentralized search

In the current framework, a new generation of specialists is created for each new request. The specialists explicitly try to enumerate the knowledge tokens in their scope and thus generate answers to the requests from which they derive. Enumerating knowledge tokens may either be achieved by directly querying a database or by issuing subrequests and combining the answers. On the other hand, the non-specialists, which cover the domain not covered by specialists, do not try to explicit the knowledge tokens implicitly described in the scopes. Their role is only to spawn new specialists at each new request.

To avoid an unmanageable proliferation of specialists, we explore several directions.

First, we could avoid over-specialization by letting the agents decide how to split their scopes, creating a specialist and a non-specialists, upon reception of a request. For example, a request for novels by Balzac may split a book agent into a specialist for Balzac writings (and not just novels) and a non-specialist covering the other authors. A later request for Balzac's mail correspondence would be handled by the same specialist and would not generate a new one.

Second, we could allow not only scope splitting, as occurs when specialists are created, but also *scope merging*. Typically, a specialist for novels before 1851 and a specialist for novels after 1850 could be merged into a single specialist for novels. A possible approach to scope merging could rely on *inductive generalization* methods (Plotkin 1970), which find the most restricted property (or constraint) which is more general than all those given as input. However, one has to be careful in that merging several scopes may lead to constraints which are not expressible in the available constraint language. Thus, restrictions have to be posed on the sets of constraints to be merged. For example, if we have a specialist for the novels before 1800 and one for the poems before 1400, then the scope merging would need a disjunction in the resulting constraint, which is not allowed by the language used in the CBKB formalism. As noted above, in the current state of the system, we impose that all scopes of the agents be expressed as SFCs, and we have designed an algorithm for combinator agents in that case. The choice of SFCs as uniform format for the scope of our agents has been influenced by our (naive) strategy for splitting-no-merging (i.e. split exactly what is needed, no merging). Indeed, SFCs are the smallest subset of feature constraints which is stable by splitting, assuming requests are all expressed as BFCs.

The merging of the scope of two specialists is intuitively their logical *or*. Let us now see how this can be computed. Each scope can be seen as the logical *and* of several SFCs. First we represent each SFC as a conjunction of predicates. For example,

```
P
+ P : person
+ P :  spouse -> P'
       P':person
       P':name -> N        N:"Genny"
```

is represented as *person(P), spouse(P,P'), person(P'), name(P',N), N="Genny"*, and

```
P
+ P:person
- P : spouse -> P'
      P': person
      P':employer -> E        E:"Xerox"
```

is represented as *person(P), not(spouse(P,P'), person(P'), employer(P',E), E = "Xerox")*. Note that in this representation unary predicates represent sorts while the others represent labels.

Now we must compute the logical *or* of these two conjunctions. By distributing the *or* over the *and*, we get a conjunction C of disjunctions, where each disjunction has the form (A or B) and A and B are BFCs with possibly a sign. At this point, we must compute a SFC constraint C' which is more general than C but less general than any other generalization of C. We call it the *least generalization*; see (Plotkin 1970) where they use the same term to generalize clauses of C. We do this by considering each logical *or* and computing its least generalization. We have to take into account several cases:

- (A or B) where $B = A$: this obviously generates A.
- (A or B) where $B \neq A$:
  - if any of A or B is not a built-in constraint, then we get the empty SFC; otherwise
  - if both A and B are built-in constraints (over the same variable X), we consider the interval represented by each of A and B over X, say $IA = (A1, A2)$ and $IB = (B1, B2)$. The union of these two intervals is given by $I =$

(min($A1, B1$),max($A2, B2$)). If I is the whole integer line, then we get the empty constraint; otherwise we write the built-in constraint(s) representing the interval I.

- (A or not A): we get the empty constraint.
- (A or not (A and B)): empty constraint.
- (A or not C), where C does not contain A:
  - if A is a built-in constraint and C = (B and D) where B is a built-in constraint (over the same variable as A), then we get: not((not(A) and B) and D), where (not(A) and B) is a built-in obtainable from A and B by a suitable constraint solver. If instead C does not contain any built-in constraint, we get the empty constraint.
  - if A is not a built-in constraint and it is not a sort constraint, it generates the empty constraint; otherwise, if C contains a sort (over the same variable as A), it generates not(C). This is due to the fact that sorts are assumed to be disjoint. Otherwise we get the empty constraint.
- (not A or not B): it generates not(A and B).

After applying these reduction rules, we may still simplify the obtained SFC because there may be situations like not (A and B) and also not(A), which can be simplified to not(A), or also duplications which can be eliminated.

Let us consider the following two scopes to be merged:

```
X
+ X : book
+ X : topic -> T        T:"Art"
+ X : published -> Y        Y >1950
- X :  author -> A     A:nationality -> N
       N:"American"
```

```
X
+ X : topic -> T        T:"Art"
- X :  author -> A     A:nationality -> N
       N:"American"
- X : book     X : published -> Y     Y >1950
```

We get: *topic(X,T), T="Art", not(author(X,A), nationality(A,N), N="American")*, which corresponds to the SFC

```
X
+ X : topic -> T        T:"Art"
- X : author -> A     A:nationality -> N
      N:"American"
```

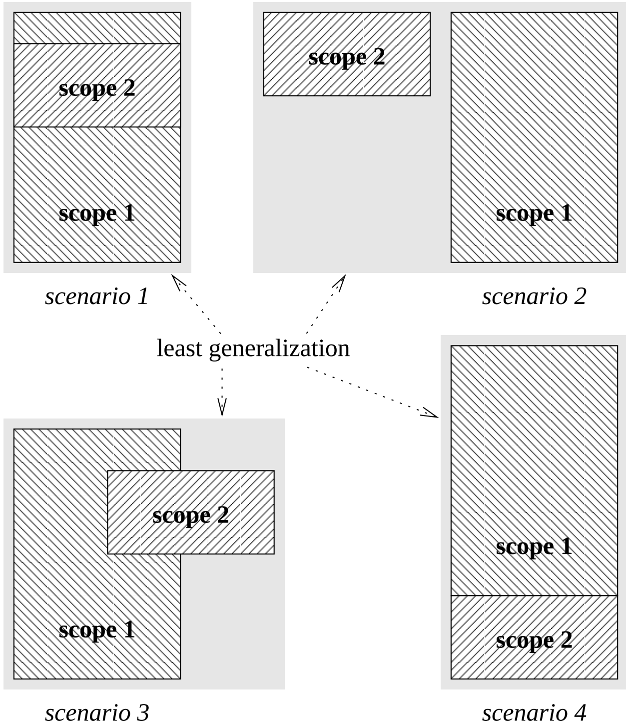

Figure 1: Geometrical characterization of scope merging

Scope merging can also be characterized geometrically. Each SFC describes an n-dimensional space where n indicates the number of different labels for each sort, e.g. *name*, *age*, *employer*, *spouse* etc. for the sort *person*. SFCs have been constructed such that this n-dimensional space is *convex*.[2] If the union of two convex spaces (representing the scopes of the specialists in question) leads to a convex space too, the merging of the scope of two specialists is straight-forward (see scenarios 1 and 4 of Fig. 1). If this union, on the other hand, is not convex, we have to look for a convex "shell" that wraps both spaces (see scenarios 2 and 3 of Fig. 1). In the most general case, this is the entire n-dimensional universe (represented as an empty constraint in the discussion above).

An alternative to scope merging consists of splitting the agents and then smartly organizing them in a disjunctive form (for example, and *or-tree*), and then an efficient search method could be used to find the most suitable agent for the request at hand. For example, the search method could be based on *decision tables*, as described in (Martelli & Montanari 1978).

[2] In reality this issue is more tricky. Some of the built-in constraints may have *gaps* and *holes* or may consist of the union of (e.g. integer intervals) and still be conform with the SFC definition.

Another aspect to consider is the other side of the environment where the broker agents interact, namely the kind of information servers which the agents will refer to: if they are not able to handle a very detailed request, it is useless to further split the current specialist if it already has reached that level of detail. Thus, a reasonable approach has to decide upon a limit to the level of specialization of the agents, by considering also the knowledge bases which will be most used by them.

Probably, the best approach will derive from a suitable and flexible combination of these four intuitions. We plan to investigate the effects of these approaches to improve the state of the art of distributed knowledge information retrieval systems, starting specifically from the system which uses the CBKB formalism.

References

Aït-Kaci, H.; Podelski, A.; and Smolka, G. 1994. A feature-based constraint-system for logic programming with entailment. *Theoretical Computer Science* 122:263–283.

URL. Alta Vista: http://altavista.digital.com/.

Andreoli, J.-M., and Pareschi, R. 1996. Integrated computational paradigms for flexible client-server communication. *ACM Computing Surveys* 28(2):295–297.

Andreoli, J.-M.; Borghoff, U. M.; Pareschi, R.; and Schlichter, J. H. 1995. Constraint agents for the information age. *J. Universal Computer Science* 1(12):762–789. Electronic version available at http://www.iicm.edu/jucs.

Andreoli, J.-M.; Borghoff, U. M.; and Pareschi, R. 1996. The constraint-based knowledge broker model: Semantics, implementation and analysis. *J. Symbolic Computation* 21(4):635–667.

Andreoli, J.-M.; Borghoff, U. M.; and Pareschi, R. 1997. Signed feature constraint solving. In *Proc. 3rd Int. Conf. on the Practical Application of Constraint Technology (PACT '97).* London, U. K.: Blackpool, U. K.: The Practical Application Company Ltd.

Borghoff, U. M.; Pareschi, R.; Karch, H.; Nöhmeier, M.; and Schlichter, J. H. 1996. Constraint-based information gathering for a network publication system. In *Proc. 1st Int. Conf. on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM '96)*, 45–59. London, U. K.: Blackpool, U. K.: The Practical Application Company Ltd.

Borghoff, U. M.; Hilf, E. R.; Pareschi, R.; Severiens, T.; Stamerjohanns, H.; and Willamowski, J. 1997. Agent-based document retrieval for the European physicists: A project overview. In *Proc. 2nd Int. Conf. on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM '97).* London, U. K.: Blackpool, U. K.: The Practical Application Company Ltd.

Chidlovskii, B.; Borghoff, U. M.; and Chevalier, P.-Y. 1997. Towards sophisticated wrapping of web-based information repositories. In *Proc. 5th Int. RIAO Conf. on Computer-Assisted Information Searching on Internet.* Montreal, Canada.

Fuhr, N. 1997. A decision-theoretic approach to database selection in networked ir. In *submitted to SIGIR.*

Gravano, L., and Garcia-Molina, H. 1995. Generalizing gloss to vector-space databases and broker hierarchies. In Dayal, U.; Gray, P. M. D.; and Nishio, S., eds., *Proc. 21st Int. Conf. on Very Large Data Bases*, 78–89. Zurich, Switzerland: San Francisco, CA: Morgan Kaufmann.

URL. Harvest: http://harvest.cs.colorado.edu/.

Martelli, A., and Montanari, U. 1978. Optimizing Decision Trees Through Heuristically Guided Search. In *Communications of the ACM* 21(12): 1025-1039.

G.D. Plotkin 1970. A Note On Inductive Generalization. In *Machine Intelligence* 5: 153–163.