Journal of Universal Computer Science, vol. 1, no. 12 (1995), 762-789
submitted: 11/12/95, accepted: 27/12/95, appeared: 28/12/95

Constraint Agents for the Information Age

Jean-Marc Andreoli, Uwe M. Borghoff and Remo Pareschi
Rank Xerox Research Centre, Grenoble Laboratory
6, chemin de Maupertuis. F-38240 Meylan, France.
Email: {andreoli,borghoff,pareschi}@xerox.fr

Johann H. Schlichter
Institut für Informatik, Technische Universität München
D-80290 München, Germany.
Email: schlicht@informatik.tu-muenchen.de

Abstract: We propose constraints as the appropriate computational constructs for the design of agents with the task of selecting, merging and managing electronic information coming from such services as Internet access, digital libraries, E-mail, or on-line information repositories. Specifically, we introduce the framework of *Constraint-Based Knowledge Brokers*, which are concurrent agents that use so-called *signed feature constraints* to represent partially specified information and can flexibly cooperate in the management of distributed knowledge. We illustrate our approach by several examples, and we define application scenarios based on related technology such as Telescript and workflow management systems.

Key Words: multiagent coordination, agent-interaction, distributed problem solving, signed feature constraints, negotiation, cooperation strategies.
Category: H.3.3., I.2.

1 Introduction

New electronic sources of information, such as E-mail, Internet access and on-line information repositories flood the desktop environment of users with an evergrowing flow of information which, in order to be exploitable, demand efficient management. The inundation of electronic data coming from all kind of sources must convert into real knowledge in order to benefit the whole range of users from business people to casual surfers and shoppers on the Internet. Intelligent agents [CACM, 1994; Wooldridge and Jennings, 1995] interacting through multiagent systems have been proposed as the appropriate answer to this demand. Indeed, software processes of this kind may one day manage distributed knowledge by "living on the network" and manipulating electronic information on users' behalf.

A central question faces us in order to reach an effective deployment of such technology: How intelligent agents can be best designed and customized to meet users' individual information needs. The issue at stake concerns essentially one of adequate computational support. However, the motivations differ from those underlying linguistic frameworks for multiagent systems such as Actors [Agha, 1986] and Agent-oriented Programming (AOP) [Shoham, 1993] as well as of multiagent architectures, either "reactive" (see e.g. [Brooks, 1991]) or "deliberative" [Bratman et al., 1988] or "hybrid" [Kaelbling and Rosenschein, 1990]. In these cases the agents are assumed to be situated in an environment which they can modify while pursuing their own goals. These goals range from collecting empty

cans, for simple robotic agents, or optimally solving scientific programming problems, for software agents in massively parallel multiprocessing architectures, to more complex types of activities for correspondingly more complex types of agents. Furthermore, the agents communicate either by passing messages (as in Actor languages) or by issuing, declining, or committing to requests, as well as performing other speech acts (as in the higher-level AOP languages). Thus, these agents explicitly communicate with their fellows and implicitly assume their situatedness in given environments. By contrast, agents primarily concerned with the elaboration and manipulation of information must have a more direct and explicit relationship with the environment since by its exploration they derive their very *raison d'etre*. Communication with fellow agents will at times be implicit and other times explicit: these agents effectively elaborate information and then communicate it, either to other agents or to humans. The recipients of information may be unknown to the senders – communication resembles more a radio broadcast or a conference presentation than a conversation between entities that know each other.

A computational model should satisfy a few precise requirements in order to support this notion of agency:

1. By definition these agents continually watch for information that meets preestablished criteria. For instance, in a given company, they can routinely scan news wires for breaking reports about the company's current customers, whoever they happen to be. Thus, it must be possible to express and implement agents' behavior in terms of a set of criteria through which information is filtered and selected. These criteria act as a partial specification of the information to come.
2. Electronic information domains are wide open lands where most of the times we do not know exactly what we are looking for, nor what we are going to find. In these conditions, a good way to guide our search is to explicitly exclude things we are not interested in. This can be conveniently expressed by freely mixing "positive" and "negative" requirements in the specification of the behavior of agents. Thus, users should be allowed to feed agents with such requests and criteria as "find me all books written by Umberto Eco which are not novels" or "I am not interested in reports on sales reps from Canada Customer Operations".
3. The scope of exploration of agents should be dynamically readjustable to optimize their work. As a minimal requirement, they should be capable of "focusing on targets," thus incrementally reducing their scope as they proceed. More intelligence could plausibly come from long-term memory, that is the remembrance of things past: They should be able to reuse the knowledge they have gained from executing a certain request in the context of other requests. Take for instance a request such as "find me all books by Umberto Eco which are not novels" and a subsequent request such as "find me all books by Umberto Eco which are literary essays."
4. We would also like to implement cooperative behavior of multiple agents on given tasks. Cooperation should arise naturally from handling queries involving the selection and composition of information from different knowledge repositories (often called backends) reachable through the Internet. Another example is the creation of compound documents on the fly from preexisting documents, according to some hierarchical description language

like SGML [ISO, 1986], with each part of the final document being assigned to a specific agent.

5. Finally, it should be possible to tune interagent communication in terms of different communication protocols according to such parameters as the nature of the problem to be solved, the underlying system architecture, etc.

In this paper we investigate these issues from the point of view of a computational construct that has already found widespread application in artificial intelligence and computer science, namely the notion of *constraint*. Constraints have been exploited mainly in the context of search and combinatorial optimization but their significance is more general and extends to the management and manipulation of information. In fact, constraints can be used to provide partial specifications on the possible values of variables. This paper illustrates how this capability can be exploited to implement predefined criteria for filtering and selecting information. The specific constraint framework we shall adopt is the Constraint-Based Knowledge Brokers (CBKBs) [Andreoli et al., 1994; Andreoli et al., to appear] model, which exploits constraints to support knowledge-intensive tasks executed by concurrent agents and views the management and manipulation of information in distributed environments as a form of distributed problem solving. A CBKB is capable of understanding and enacting both "requests" and "negations of requests," is self-sufficient in managing its own scope of exploration over a given information domain and is capable of knowledge reuse. Furthermore, different communication protocols for CBKBs have been defined that can be used to tune interagent communication and cooperation.

The remainder of this paper is organized as follows. In Sect. 2, we characterize the notion of multiagent interaction in the context of distributed problem solving. Agents are classified and given a set of general requirements. Agent cooperation is then illustrated in terms of CBKBs. Two different protocols for interagent communication are introduced and described, one supporting direct, explicit communication and the other supporting group-oriented communication. Sect. 3 introduces a specific type of constraints suitable for representing electronic information, namely signed feature constraints (SFC). In Sect. 4, SFCs are used to illustrate a number of specific issues of information management, such as interdependencies, thresholds, and reuse of information. Sect. 5 explains related scenarios. In particular we discuss negotiation in the contract-net protocol, Telescript as a promising agent infrastructure, and workflow management as an interesting application domain for remote programming. In Sect. 6, related work is discussed. Sect. 7 concludes the paper.

2 Multiagent Interaction

The area of Distributed Problem Solving (DPS) has led to various approaches which allow distributed (semi-)autonomous agents to cooperate in order to solve complex problems and accomplish tasks which might not be solvable by one individual system. From the problem solving point of view, distribution implies the decomposition of the problem into a set of subproblems and the dissemination of the subproblems to the appropriate agents which solve them autonomously and concurrently. The final solution of the global problem can be generated by composing the solutions of the subproblems. Thus, agents can be viewed as problem solvers which cooperate to generate the solution of the global problem.

2.1 Classification

We distinguish between passive and active agents. *Passive* agents act under direct user control. The user explicitly triggers the execution of agent functions, e.g. sorting and filing electronic messages in the user's mailbox. Unlike passive agents, *active* agents react to incoming messages, such as requests for information or the execution of functions, autonomously or semi-autonomously. *Autonomous* agents may perform actions without user involvement. They have enough knowledge about the problem domain and the contextual constraints to interpret received messages and react appropriately. During execution, the user has no direct control over the agent's behavior. On the other hand, *semi-autonomous* agents perform routine tasks for the user. Exceptional requests or situations are referred to the user who handles them personally. The behavior of semi-autonomous agents is directly controlled by the user who has read and write access to the rules which specify the agent's behavior.

Agents are used in a wide area of different application domains ranging from robotics, distributed sensoring to Computer-Supported Cooperative Work (CSCW) and information gathering [Wayner, 1994]. The emerging field of CSCW provides a demanding area for distributed problem solving. CSCW systems need to support the interaction of humans and overcome the difficulties of temporal and spatial distribution. For instance, agents may be used to support the scheduling of meetings (see [Sen and Durfee, 1991]). Another related application domain is that of workflow management and document processing where agents might be used to coordinate the tasks and information exchange between tasks and humans. The rapid growth of the Internet and the World-Wide Web have demonstrated the need for innovative and efficient ways of information gathering, and this provides the main focus for this paper. The World-Wide Web makes available an incredible amount of information; however, in many cases the user is unable to find and extract the desired information effectively. In this case agents may be used to collect relevant information, filter the search results according to contextual constraints, and present the resulting information to the user in an appropriate form. *Telescript* [White, 1994b] is an example of system providing infrastructural support for this type of agent application.

The *contract-net protocol* [Smith, 1980] was one of the first approaches to provide a general framework for DPS. It supports an application protocol for communication between problem solving agents and facilitates distributed control during the problem solving effort. Special emphasis is put on

- localizing those agents which are eligible for solving the created subproblems;
- the negotiation between agents for the information exchange with respect to subproblem descriptions, required agent capabilities and subproblem solutions [Davis and Smith, 1983].

The Rank Xerox Research Centre at Grenoble has developed the model of *Constraint-Based Knowledge Brokers* (CBKBs) which uses constraints to provide computational support for DPS. CBKBs explicitly separate aspects of local problem solving, based on computations specific to a single agent, from aspects of global problem solving, deriving from the interaction of different agents. CBKBs model active agents which act autonomously and concurrently. In the following sections some of the specific capabilities of the CBKB model will be discussed in more detail.

In order to effectively cooperate and participate in the problem solving effort, an agent must satisfy the following requirements:

- an agent must be able to communicate with other agents of the system (e.g. send and receive request/answer messages);
- an agent must be able to act upon receipt of messages.

2.2 Cooperation between Agents

The phenomenon of cooperation which is well-known in the human environment may also be applied to agent interaction. A number of different cooperation strategies between agents have been proposed, ranging from strongly hierarchical master-slave relationship, to the less hierarchical contract-net [Smith, 1980], to the sharing of common goals. In the latter case agents not only exchange information with respect to their individual tasks and problems, but they also communicate their goals. Thus, the agents follow shared goals when pursuing the problem solving activities.

In general, cooperation between agents is based on explicit communication, i.e. agents send messages to transfer knowledge and requests. The message content can range from values, formal and informal descriptions, to constraints. The CBKB model uses values and constraints to represent knowledge and requests for problem solving. The basic message types in the context of DPS are requests and answers. Usually messages are completely structured and are only intended for agent consumption; the messages are not in human-readable form. The message structures can be tailored to reduce network bandwidth and interpretation complexity by the agents. Both the contract-net protocol and the CBKB model apply structured messages to model agent interaction. An example for a system which uses semi-structured messages is *Object Lens* [Malone and Lai, 1988], which provides intelligent filtering and dissemination of electronic mail messages. Semi-structured messages are based on the notion of a semi-formal system [Malone, 1989] which:

- represents and interprets information that is formally specified,
- permits the human user to create and interpret formal information informally,
- allows the formal interpretation by the computer and the informal interpretation by the user to be easily changed.

Semi-formal systems are especially useful in heterogeneous environments where there is no clear separation between human tasks and agent tasks. They support the co-existence of humans and agents in the same environment. For example, some people use personal agents to cooperate in the distributed meeting scheduling process, while other people perform the required requests manually. Thus, semi-formal systems facilitate a smooth transition from a purely human-oriented environment to a completely agent-based environment. However, semi-formal systems use rather complex messages. This creates a significant network load and requires complex interpretation functionality by the agents.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.