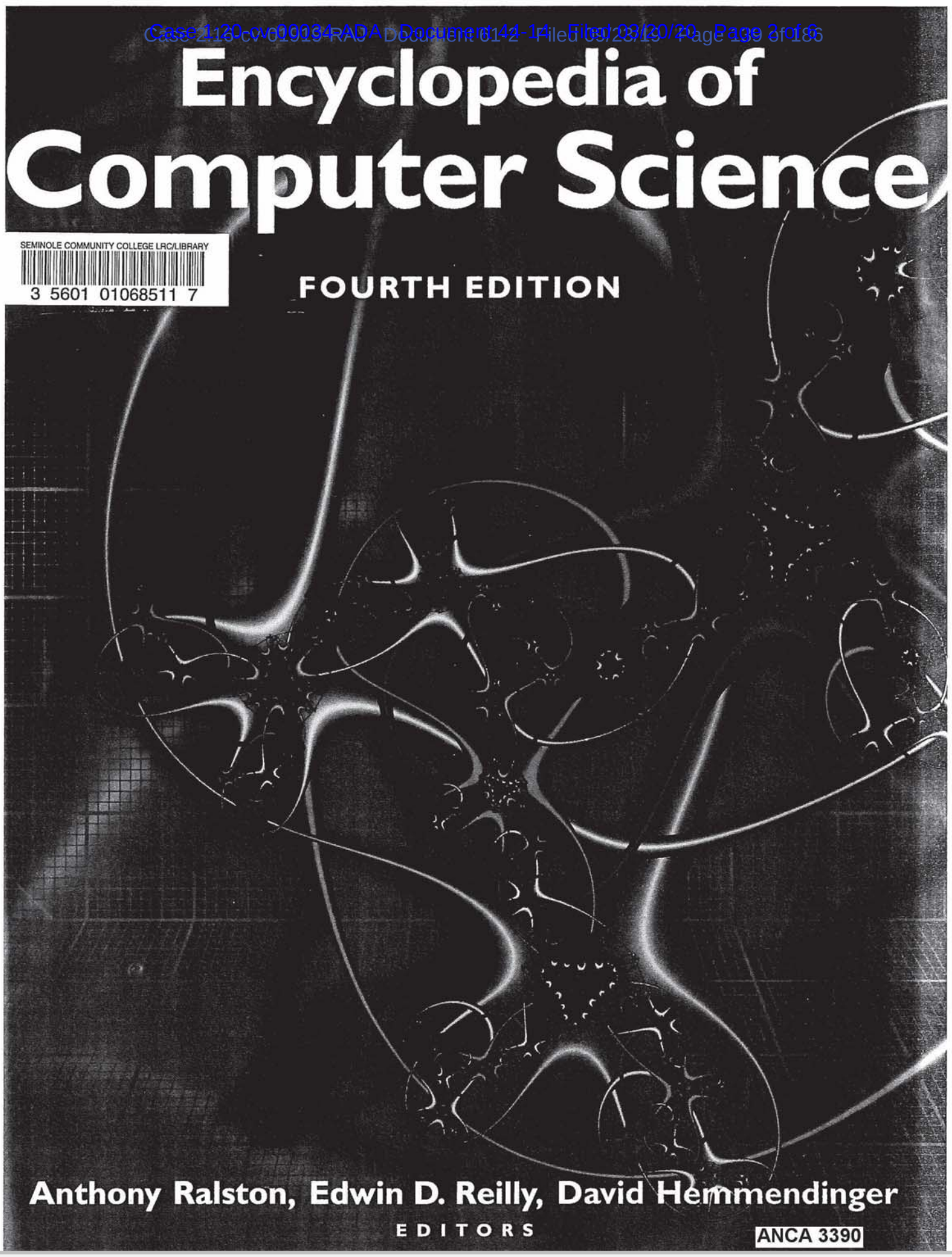
Exhibit 13

Encyclopedia of Computer Science

FOURTH EDITION



Anthony Ralston, Edwin D. Reilly, David Hemmendinger
EDITORS

Encyclopedia of Computer Science
Fourth Edition
Edited by Anthony Ralston, Edwin D. Reilly and David Hemmendinger

Published in the United Kingdom by
NATURE PUBLISHING GROUP, 2000
25 Eccleston Place, London, SW1W 9NF, UK
Basingstoke and Oxford
ISBN: 0–333–77879–0
Associated companies throughout the world
http://www.nature.com

British Library Cataloguing in Publication Data

Encyclopedia of Computer Science
1. Computer science–Encyclopedias
I. Ralston, Anthony II. Reilly, Edwin D. III. Hemmendinger, David
004/ . 03

Published in the United States and Canada by
GROVE'S DICTIONARIES, INC
345 Park Avenue South, New York, NY 10010-1707, USA
ISBN: 1–561–59248–X

Library of Congress Cataloging in Publication Data

A catalog for this record is available from the Library of Congress

Typeset in the United Kingdom by Aarontype Limited, Bristol.
Printed and bound in the United Kingdom by Bath Press Ltd, Bath.

no mouse pad is needed (*see* Fig. 2). An optical sensor captures images of the work surface at a rate of 1,500 images per second, and a digital signal processor (DSP) translates changes between the images into on-screen movements. This technique, called *image correlation processing*, results in smooth, precise pointer movement. The mouse features a glowing red underside and tail light, a scrolling and zooming wheel, and two customizable buttons on its left side which facilitate Internet (*q.v.*) navigation and other routine tasks.

Bibliography

1992. Soberanis, P. "Of Mice and Trends," *CompuServe Magazine*, **11**, *2* (February), 29–30.
1998. White, R. "The Mechanical Mouse," in *How Computers Work*, 4th Ed., 160–161. Indianapolis, IN: Que/Macmillan

Edwin D. Reilly

MULTI-AGENT SYSTEMS

For articles on related subjects *see* ARTIFICIAL INTELLIGENCE; DISTRIBUTED SYSTEMS; EXPERT SYSTEMS; and HEURISTIC.

Multi-agent systems are computational systems in which several artificial "agents", which are programs, interact or work together over a communications network to perform some set of tasks jointly or to satisfy some set of goals. These systems may consist of homogeneous or heterogeneous agents. Examples of agents would be ones for detecting and diagnosing network problems occurring on a segment of a local area network; for scheduling the activities of a group of machines in a workcell on a factory floor; or for locating agents that are selling a specific product and deciding on what price to pay. Agents may be characterized by whether they are benevolent (cooperative) or self-interested. Cooperative agents work toward achieving a set of shared goals, whereas self-interested agents have distinct goals but may still interact to further their own goals. For example, in a manufacturing setting, where agents are responsible for scheduling different aspects of the manufacturing process, agents in the same manufacturing company would behave in a cooperative way, while agents representing two separate companies, where one company was outsourcing part of its manufacturing process to the other company, would behave in a self-interested way. Agents often need to be semi-autonomous and highly adaptive due to their "open" operating environments, where the configuration and capabilities of other agents and network resources change dynamically. Agent autonomy relates to an agent's ability to make its own decisions about what activities to do, when to do them, and to whom information should be communicated. Scientific research and practice in this area, which is also called *Distributed Artificial Intelligence* (DAI), focuses on the development of computational principles and models for constructing, describing, and analyzing the patterns of interaction and coordination in both large and small agent societies.

Multi-agent systems provide a potential model for computing in the twenty-first century, in which networks of interacting, real-time, intelligent agents integrate the work of people and machines, and in which the effectiveness of computational agents in large distributed systems is improved by exploiting the efficiencies of organized behavior. Application domains in which multi-agent system technology is appropriate typically have a naturally spatial, functional, or temporal decomposition of knowledge and expertise among agents. By structuring such applications as a multi-agent system rather than as a single agent, the system will have some or all of the following advantages:

- Speed-up due to concurrent processing;
- Less communication required because processing is located nearer the source of information;
- More reliability because of the absence of a single point of failure;
- Real-time (*q.v.*) responsiveness due to processing, sensing, and effecting being collocated;
- Easier system development due to the modularity produced by dividing the program into agents.

Domains which have used a multi-agent approach include: distributed situation assessment (e.g. network diagnosis, information gathering, and monitoring on the Internet); distributed resource scheduling and planning (e.g. factory scheduling, network management); and distributed expert systems (e.g. concurrent engineering). A multi-agent approach is also useful in applications in which agents represent the interests of different organizational entities (for example, in electronic commerce (*q.v.*) where agents representing the interests of different buyers and sellers negotiate over an acceptable price for delivery of goods or services). Other emerging uses of multi-agent systems are in layered systems architectures, in which agents at different layers need to coordinate their decisions (e.g. to achieve appropriate configurations of resources and computational processing) and in the design of resilient systems in which agents dynamically reorganize to respond to changes in resource availability, software and hardware malfunction, and intrusions. In general, multi-agent systems provide a framework in which both the distribution of processing and information in an application and the complexities that come from issues of scale can be handled in a natural way.

Agents in such systems need to interact because they are solving subproblems that are interdependent, either through contention for resources or through relationships among the subproblems. This need for interaction may require them to cooperate extensively during problem-solving based on reasoning about subproblem interdependencies, the agents' current state of problem-solving, and the status of network resources. Such agent interactions are exemplified in a recently developed commercial multi-agent system for restoring service in an electricity transportation grid. This application has agents for fault detection, fault isolation and diagnosis, and network reconfiguration. Consider the example of two expert agents in this system performing different forms of fault diagnosis. Each of these agents, operating concurrently, uses very different algorithms to do its diagnosis and the information that they use is not identical. Both can make mistakes, but generally will not make the same mistake. They interact by exchanging partial results to focus their local diagnostic search processes towards promising areas of the grid where the fault probably originated, and away from unpromising ones. They also exchange final results to increase the confidence in the eventual diagnosis that they agree to. Thus, by working together, they not only produce a solution in which they have more confidence, but they also accomplish the task quicker.

Multi-agent systems must be designed to enable an agent to modify its problem-solving activity in response to the emerging state of the group problem-solving effort. Agents must be flexible, as they work with information of varying degrees of completeness and accuracy, and use resources of varying capabilities. For example, in the multi-agent system described above the agents doing the diagnosis should be able to work in a standalone manner, but also be able to take advantage of information from the other diagnostic agent if and when it arrives. In other words, hard-coded assumptions about information and resources are typically avoided. This flexibility requires agent autonomy and is in direct contrast to the less autonomous characteristics of agents in usual distributed processing applications.

The design, implementation, and assessment of multi-agent systems raises many specific issues. The major conceptual problem that researchers face in dealing with these issues is the possibility that the information an agent is using to make its decisions is incomplete, out-of-date or inconsistent with that of other agents. Obtaining all the appropriate non-local information is often not practical due to:

1. Limited communication bandwidth (*q.v.*) and computational capabilities which make it infeasible to transfer, package, and assimilate pertinent information in a timely manner.
2. The heterogeneity of agents, which makes it difficult to share information and the possibility that competitive agents, out of self-interest, are not willing to share certain information.
3. The dynamic character of the environment due to changing problems, agents, and resources, and the inability to predict with certainty the outcome of agents' actions.

In order to deal with this uncertainty in problem-solving and coordination decisions, a number of formal and heuristic techniques have been developed. These techniques are oriented towards achieving effective, though not necessarily optimal, agent problem-solving and interaction, while limiting the computational and communication requirements. These include: group problem-solving strategies that can reach acceptable solutions even though an individual agent's local information may be incorrect or incomplete; coordination strategies that enable groups of agents to solve problems effectively through decisions about which agents should perform specific tasks and when, and to whom they should communicate the results of task execution; negotiation mechanisms that serve to bring a collection of agents to an acceptable state; and protocols (*q.v.*) by which agents may communicate and reason about interagent communications. Where formal techniques have been used, they have generally been based on game-theoretic ideas, market mechanisms, or logical formalisms, while heuristic approaches have their roots in knowledge-based AI search, planning and scheduling mechanisms. A recent trend is the use of machine learning (*q.v.*) to acquire the information necessary to implement these approaches.

The use of multi-agent systems technology is still in its infancy. There are only a handful of commercial applications to date. However, given the great interest in the field, the emerging multi-agent application development infrastructures, and the next generation of sophisticated network applications beginning to take shape, we may expect the impact of multi-agent systems on computer science to increase significantly during the next decade.

Bibliography

1988. Bond, A., and Gasser, L. (eds.) *Readings in Distributed Artificial Intelligence.* San Francisco: Morgan Kaufmann.

1994. Rosenschein, J. S., and Zlotkin, G. *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers* (eds. M. Brady, D. Bobrow and R. Davis). Cambridge, MA: MIT Press.

1994. Jennings, N. R. *Cooperation in Industrial Multi-Agent Systems.* Singapore: World Scientific.

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.