Holonic Multiagent Systems: A Foundation for the Organisation of Multiagent Systems

Klaus Fischer, Michael Schillo and Jörg Siekmann

DFKI GmbH,
Stuhlsatzenhausweg 3,
D–66123 Saarbrücken,
{kuf, schillo, siekmann}@dfki.de

Abstract. With the growing usage of the world-wide ICT networks, agent technologies and multiagent systems are attracting more and more attention, as they perform well in environments that are not necessarily well-structured and benevolent. Looking at the problem solving capacity of multiagent systems, emergent system behaviour is one of the most interesting phenomena, however, there is more to multiagent systems design than the interaction between a number of agents: For an effective system behaviour we need structure and organisation. But the organisation of a multiagent systems is difficult to specify at design time in the face of a changing environment.
This paper presents basic concepts for a theory of holonic multiagent systems to both provide a methodology for the recursive modelling of agent groups, and allow for dynamic reorganisation during runtime.

1 Introduction

A multiagent system (MAS) consists of a collection of individual agents, each of which displays a certain amount of *autonomy* with respect to its actions and perception of a domain. Overall computation is achieved by *autonomous computation* within each agent and by *communication* among the agents. The capability of the whole MAS is an *emergent functionality* that may surpass the capabilities of each individual agent [19, 20]. An extremely useful feature in terms of reduction of complexity for the designer of a MAS is that an overall task can be broken down into a variety of specific sub-tasks, each of which can be solved by a specific agentified problem solver.

Jennings notes that "the development of robust and scalable software systems requires autonomous agents that can complete their objectives while situated in a dynamic and uncertain environment, that can engage in rich, high-level social interactions, and that can operate within flexible organisational structures" [9]. Agents acting in organisational structures can encapsulate the complexity of subsystems (simplifying representation and design) and modularise its functionality (providing the basis for rapid development and incremental deployment). Organisations are social structures which have mechanisms of conflict resolution resulting from previously resolved problems or conflicts [7]. They institutionalise

anticipated coordination, which is especially useful for medium- and large-scale applications that require limitation of the agents' communication behaviour.

With this work, we provide some terminology and theory for the realisation of dynamically organised societies of agents. The central concept for our endeavour, which has been iteratively tested, developed, and applied in a series of projects over the course of several years, is the *holonic multiagent system*. According to Arthur Koestler [12], a *holon* is a self-similar or *fractal* structure that is stable and coherent and that consists of several holons as sub-structures. Koestler gives biological examples. For instance a human being consists of organs which in turn consist of cells that can be further decomposed and so on. None of these components can be understood without its sub-components or without the super-component it is part of.

Many distributed problems exhibit an inherent structure and we need to mirror this structure in the structure of the relationship between (agentified) problem solvers. For this purpose in a holonic multiagent systems, an agent that appears as a single entity to the outside world may in fact be composed of many sub-agents and conversely, many sub-agents my decide that it is advantageous to join into the coherent structure of a super-agent and thus act as single entity — just as the swarm of a certain species of fish sometimes takes on the appearance of a (much bigger) fish. We call agents consisting of sub-agents with the same inherent structure *holonic agents*.

Section 2 gives a formal definition of multiagent systems in general. In Section 3, we extend this to a formal definition of holonic multiagent systems building on previous work in this area [5, 6, 8], and highlight the diversity of groupings (links of varying nature between agents and recursion) that are possible with this concept. Section 4 compares the notion of holonic multiagent systems to holonic manufacturing systems.

2 Abstract Specification of Multiagent Systems

For any software system, it is common practice to distinguish the static specification of the system from its runtime instance. While concepts and theories for the static specification of software systems are reasonably well-understood, concepts and theories for the specification and analysis of the dynamic behaviour of a software system are by far not as sophisticated. This is especially true if we look at MAS, in which self-organisation is an important aspect. This makes MAS different from systems that are designed according to a more traditional software development paradigm.

We assume that there is some infrastructure which supports the agents in the process of self-organisation. For example, the FIPA[1] initiative has established standards for such infrastructures in an open environment. This paper takes a more abstract point of view, which assumes that there is an agent directory service (ADS) which allows the agents to find out how they can contact other

[1] See http://www.fipa.org/

agents that currently exist in the system. This means that we require that the ADS provides at least a white pages service, where agents can inquire the addresses of other agents. Other services like yellow pages, i.e. the information on which services are offered by specific agents, may also be provided by the ADS. However, this and possibly other more general services could also be introduced by other specialised agents of the MAS.

To describe a concrete MAS for a given application domain, we specify a set of prototypical agents. This static description of the MAS is given by $\mathcal{MAS}_{prot} := (\mathcal{A}_{prot}, ADS)$, where

$\mathcal{A}_{prot}$ is the set $\{A^1, \ldots, A^n\}, n \in \mathbb{N}$ of prototypical agents, instances of which can be dynamically introduced into the system. These agents are the potentially available problem solvers, where several instances of a specific prototypical agent can be created.

ADS is a specialised prototypical agent providing an agent directory service.

We assume that instances of this finite set of agent types can be dynamically introduced into the MAS that executes (i.e. works on a specific problem) in a given application domain.

The process of problem solving starts with the initial agent system

$$\mathcal{MAS}_{init} = (\mathcal{A}_{init}, ADS_{init}) \text{ where}$$

$$\mathcal{A}_{init} = \{A^1_1, \ldots, A^1_{k_1}, \ldots A^n_1, \ldots, A^n_{k_n}\}, k_1, \ldots k_n \in \mathbb{N} \text{ and}$$

$$\forall A^i_j \in \mathcal{A}_{init} : A^i \rhd A^i_j \wedge A^i \in \mathcal{A}_{prot}.$$

$A^i \rhd A^i_j$ (read "A^i *is instantiated by* A^i_j") denotes that A^i_j is an instance of the prototypical agent A^i. This means that A^i_j inherits its behaviour and initial knowledge from A^i but may also have additional knowledge (like for example its unique identification which can be used as an address to communicate with A^i_j).

Note that the explicit introduction of ADS_{init} does not necessarily mean that the MAS is closed in the sense that the system engineer is in control of all parts of the system. We can assume that ADS_{init} represents some ADS, which is already available and which has the state of ADS_{init} at the time when the first agent of the part of the system that is under the control of the system engineer is started. Along the same line of reasoning we can assume that some of the agents in $\mathcal{A}_{init}$ were also not designed by the software engineer but represent agents that are available in the open environment. Let us without loss of generality assume that $\mathcal{A}_{open} = \{A^1_1, \ldots A^1_{k_1}, \ldots A^m_1, \ldots A^m_{k_l}\}$ for some $1 \leq m < n$ represents the set of these agents. The specification for the corresponding prototypical agents $A^1, \ldots A^m$ is likely to be incomplete in the sense that the system engineer who designs $\mathcal{A}_{prot}$ only needs to have the information about $A^1, \ldots A^m$. This is actually needed for the rest of the agents in $\mathcal{A}_{prot}$ to use services that are offered by the former set of agents.

From $\mathcal{MAS}_{init}$ the dynamic $\mathcal{MAS}_t$ evolves as

$$\mathcal{MAS}_t = (\mathcal{A}^t, ADS_t) \text{ where}$$

$$\mathcal{A}_t = \{A^{1,t}_1, \ldots, A^{1,t}_{l_1}, \ldots A^{n,t}_1, \ldots, A^{n,t}_{l_n}\}, l_1, \ldots, l_n \in \mathbb{N} \text{ and}$$

$$\forall A_j^{i,t} \in \mathcal{A}_t : A^i \blacktriangleright A_j^{i,t} \wedge A^i \in \mathcal{A}_{prot}.$$

$\blacktriangleright$ (read "*is transformed into*") denotes $\rhd \circ \rightsquigarrow^*$ which means that we have $A^i \rhd A_j^i$ where $A_j^i \in \mathcal{A}_{init}$ and $A_j^i \rightsquigarrow^* A_j^{i,t}$ where $\rightsquigarrow$ denotes the transformation of A_j^i by a single step of computation.

The computation goes on while the agents send and receive messages. New agents may be introduced and some of the active agents may be terminated. Each agent has a unique address, which an agent can make accessible to all other agents by registering with the *ADS* agent. All agents automatically know the identification of the *ADS* agent.

3 Holonic Multiagent Systems

Multiagent systems represent a new problem solving paradigm [1], where the difficult specification at design time of how a problem should be solved, is all well come by the interaction of the individual agents at run-time and the idea is that the solution of a given problem emerges from this interaction. Looking at nature, an ant hive is a well-known intuitive case, which demonstrates emergent problem solving behaviour :It is impossible to explain the overall behaviour of an ant hive just by the behaviour of an individual ant and the removal of even a significant part of the hive and does not necessarily influence the overall behaviour. Though some parts of the hive seem to be more important than others. Although interesting results have been presented using this approach to problem solving, emergent problem solving behaviour has also been criticised to provide inefficient or even undesirable results.

Divide and conquer is a widely accepted problem solving paradigm of computer science. Here, a centralised problem solving entity accepts a task, separates it into sub-tasks and distributes these sub-tasks to decentralised problem solvers. The problem solvers produce solutions for the sub-problems and send these solutions back to the centralised problem solver which integrates the solutions of the sub-problems into an overall solution for the original task. This approach to problem solving is of course much more structured than the pure emergent problem solving paradigm. The contract-net protocol [17] is a widely-accepted problem solving model in based on the divide and conquer model, where the centralised problem solving entity, called the manager for the task, separates the overall task into sub-tasks. The manager uses a bidding procedure (a first price sealed bid auction) to find the most appropriate decentralised problem solver for each of the sub-problems. The integration of the solutions of the sub-problems into an overall solution is again done by the manager. This procedure can be recursively nested, i.e. the decentralised problem solvers can again use the contract net model to find a set of further problem solvers who are able to solve the given sub-sub-task.

3.1 Definition of a Holonic Multiagent System

The concepts of *fractal* and *holonic* system design in manufacturing were proposed to combine top-down hierarchical organisational structure with decen-

tralised control, which takes the bottom-up perspective [18, 3]. Although it is possible to organise holonic structures in a completely decentralised manner, for efficiency reasons it is more effective to use an individual agent to represent a holon. In some cases, one of the already existing agents is selected as the representative of the holon based on a fixed election procedure. In other cases a new agent is explicitly introduced to represent the holon during its lifetime. Representatives are called the *head* of the holon, the other agents in the holon are called *body*. In both cases, the representative agent represents the shared intentions of the holon and negotiates these intentions with the agents in the holon's environment as well as with the agents internal to the holon. Only the head communicates with the outside of the holon. The binding force that keeps head and body in a holon together can be seen as commitments [16].

Using the formalisation of Section 2, the set $\mathcal{H}$ of all holons in $\mathcal{MAS}_t$ is defined recursively:

- for each $a \in \mathcal{A}_t$, $h = (\{a\}, \{a\}, \emptyset) \in \mathcal{H}$, i.e. every instantiated agent constitutes an *atomic* holon, and
- $h = (Head, Subholons, C) \in \mathcal{H}$, where $Subholons \in 2^{\mathcal{H}} \backslash \emptyset$ is the set of holons that participate in h, $Head \subseteq Subholons$ is the non-empty set of holons that represent the holon to the environment and are responsible for coordinating the actions inside the holon. $C \subseteq Commitments$ defines the relationship inside the holon and is agreed on by all holons $h' \in Subholons$ at creation of the holon h.

A holon h behaves in its environment like any other agent in $\mathcal{A}_t$. Only at closer inspection it may turn out that h is constructed from a set of agents. As any head of a holon has a unique identification, it is possible to communicate with each holon by just sending messages to their addresses. Given the holon $h = (Head, \{h_1, ..., h_n\}, C)$ we call $h_1, ..., h_n$ the *subholons* of h, and h the *superholon* of $h_1, ..., h_n$. The set $Body = Subholons \backslash Head$ (the complement of $Head$) is the set of subholons that are not allowed to represent holon h. Naturally, holons h' are allowed to engage in several different holons at the same time, as long as this does not contradict the sets of commitments of these superholons. We will now outline a treatment of C, a more detailed coverage of this topic can be found in [15].

3.2 Holonic Organisation

For the implementation of a holonic multiagent system, we need to turn to more fine grained issues concerning the commitments that define the intra-holonic relationship. Let us look at some general possibilities for modelling holonic structures. The following notions differ in the degree of autonomy the subholons have and cover the spectrum from full subholon autonomy to a complete lack of autonomy.

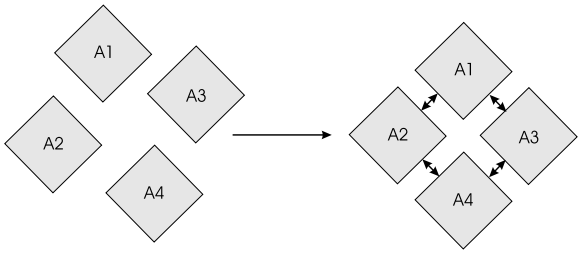

Fig. 1. A holon as a set of autonomous agents.

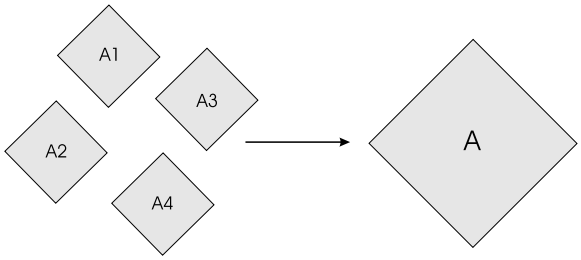

Fig. 2. Several agents merge into one.

A holon as a set of autonomous agents At one end of the spectrum is a model which assumes that the subholons are fully autonomous agents with their predefined architecture and the superholon is just a new conceptual entity whose properties are made up by the properties of the subholons. Figure 1 displays this constellation. In this case no agent has to give up its autonomy, and the superholon is realised exclusively through cooperation among the subholons. The most transparent way of cooperation for this is an *explicit coordination by commitment* via communication, i.e., agents negotiate over joint plans, task distribution or resource allocation. If commitments can not be established through communication, *implicit coordination* can be achieved in two ways: either, the holons are designed such that a goal directed common behaviour emerges from the behaviour of the sub-agents, or some subholons are able to represent goals and intentions of other agents and reason about them; thus, they coordinate their actions without or at least with little communication.

The representation of a holon as a set of autonomous agents is in a sense just another way of looking at a traditional multiagent system. The holon entity itself is not represented explicitly as a piece of code. In this case, holonic structures are only a design aid for structured agent-oriented programming. This is formally described as holon $h = (\{A_1, A_2, A_3, A_4\}, \{A_1, A_2, A_3, A_4\}, C_{autonomous})$.

Several agents merge into one The other extreme of the design spectrum terminates the participating sub-agents and creates a new agent as the union of the

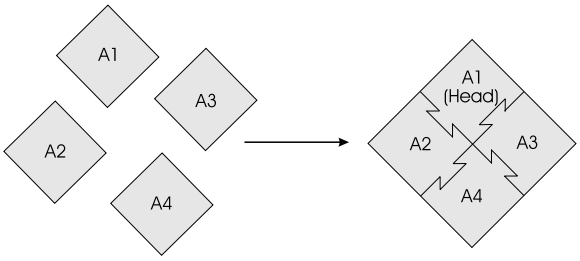

Fig. 3. A holon as a moderated association.

sub-agents with capabilities that subsume the functionalities of the sub-agents (see Figure 2). In this case the merging agents completely give up their autonomy but they may be re-invoked when the superholon is terminated. Naturally, this is a new atomic holon $h = (\{A\}, \{A\}, C_{merge})$.

The realisation of this approach assumes procedures for splitting and merging holons that lead to the creation of a new agent. For agents of the same kind with an explicit representation of goals and beliefs (e.g., BDI agents) merging can be achieved by creating an agent with the union of the sub-agents' beliefs and goals provided consistency. Especially for a heterogeneous sets of agents this can be intractable and in either case may not be very desirable.

A holon as a moderated association The two solutions above are extremes and only useful in very specific circumstances. Hence, we shall propose a continuum, the border lines of which are the two above architectures. Consider a hybrid way of forming a holon, where agents give up only part of their autonomy to the superholon (cf. Figure 3). From a software engineering point of view (in terms of reducing complexity) it is advisable to allow only for a single head which represents the superholon to the rest of the agent population (to reduce coordinational effort). Its competence may range from purely administrative tasks to the authority to give directives to other subholons. Furthermore, the head may have the authority to plan and negotiate for the holon on the basis of its subholons' plans and goals, and even to remove some subholons or to incorporate new subholons. Figure 3 visualises this approach with an example resulting in a holon $h = (\{A_1\}, \{A_1, A_2, A_3, A_4\}, C_{association})$.

Several ways to determine the head are possible. Either, a new agent is created for the lifetime of the holon, or one of the members of the holon takes the role of the head and gains the additional functionality. Or, either one member of the holon is pre-destined for the leadership or an election procedure is needed to promote one of the agents to leadership. Depending on the application domain, the competence of the representative may vary: the resulting structure can range from a loosely moderated association to a authoritative, hierarchical structure. However, the members of the superholon are always represented as agents, and, hence, we do not lose the capability to solve problems in a distributed fashion.

This approach allows for an explicit modeling of holons, a flexible formation of holonic associations, and a scalable degree of autonomy of the participating agents that are subject to negotiation and make up the commitments $C_{association}$ of the superholon (for a more detailed discussion see [15]). The most challenging problem in this design is the distribution of individual and overall computation of the holonic multiagent system.

4 Holonic Multiagent Systems vs. Holonic Manufacturing Systems

Although similar in name, there are several important differences between holonic multiagent systems as proposed here and holonic manufacturing systems as presented in the literature (e.g. [4, 14]):

- Modelling the recursion of agent grouping is an integral part of holonic multiagent systems. Mirroring the complex composition of a task, holonic agents can engage in a complex nested structures and nested structures of arbitrary depth are possible and meaningful (depending on the complexity of the task). This is not the case for holonic manufacturing systems.
- Holonic manufacturing systems make no assumption about the internal architecture of the head of a holon, it is only required to act as the control unit. For holonic multiagent systems however, the head is required to possess agent properties (cf. [21]).
- The head of holonic multiagent systems are not required to co-ordinate the work of a physical resource, but instead co-ordinate the work of several information agents that exist only virtually (information agents collaborating for increase of efficiency, to combine competencies or resources, to resolve bottlenecks).
- Holonic manufacturing systems use a market metaphor to design inter-holon co-ordination. Research on holonic multiagent systems is concerned with choosing long-term partners (and is thus related to coalition formation) as well as researching the diversity of possible organisational structures [11].
- In a holonic multiagent system, a holon is not a piece of code. It is merely a concept that is realised by commitments between agents (which exist as code) to maintain a specific relationship concerning goals and has as a result an emergent structure between agents.

While this shows the conceptual differences, it does not rule out the application of holonic multiagent systems to the manufacturing domain, which we have done successfully in several industrial projects.

5 Applications

The proposed theory has been iteratively tested, developed, and applied in a series of projects over several years with a big variation in requirements. In one

domain (flexible manufacturing) agents form holons because they have different abilities and can only as a group achieve the task at hand [5]. A second example (train coupling and sharing) demonstrates that even in a setting where we have agents with identical abilities holonic structures can be beneficial [13]. Several other projects focused on special aspects of holonic modelling (e.g. RoboCup [10], Socionics [15]) The most striking application that used the presented approach to holonic multiagent systems is the TeleTruck system, which was designed to do order dispatching in haulage companies [2].

In this system, the basic transportation units (trucks, trailers, drivers, chassis, and containers) are modeled by agents which temporarily form holons that represent vehicles for the execution of transportation tasks. The vehicle holons are headed by a special agent that is equipped with planning capabilities. All the vehicle holons and the agents representing currently idle transportation units form a super-holon that represents the whole transportation company. The head of the company holon, called the *company agent* coordinates the interaction with the user and communicates with other companies that employ the TeleTruck system. Agents representing transportation units are autonomous in their decision to participate in a vehicle holon. Participating in the holon however restricts the autonomy of the subholons for this time span, since they have to execute the sub-tasks allocated to them. The agents forming a vehicle holon cooperate in order to pursue the goal of executing a set of transportation tasks. Sometimes, even different vehicle holons cooperate for a task. A vehicle holon is able to transport the cargo, which none of its components could do on its own.

6 Conclusion

The paper presents a general framework for holonic multiagent systems, whose advantage is threefold. First, the model preserves compatibility with standard multiagent systems by addressing every holon as an agent, whether this agent represents a set of agents or not. The complexity of a group of agents is encapsulated into a holon represented by its head, the number of agents involved in the holon becomes irrelevant for other agents communicating with it. Secondly, holonic multiagent systems are one way to introduce recursion into the modelling of multiagent systems, which has proven to be a powerful mechanism in software design. Of course a holonic multiagent system is more than just the recursive decomposition into its agents, as we have solved the additional structure preserving problem. Third, there is no restriction to a specific or static association between agents, so it leaves room to introduce a variation of organisational concepts, which can dynamically change at run-time. There is no comparable programming construct that would support the design of such systems in a purely object-oriented programming approach.

References

1. A. H. Bond and L. Gasser. *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann, 1988.

2. H. Bürkert, K. Fischer, and G. Vierke. Holonic transport scheduling with TELETRUCK. *Applied Artificial Intelligence*, 14(7):697–726, 2000.
3. J. Christensen. Holonic manufacturing systems — initial architecture and standard directions. In *Proc. of the 1st European Conference on Holonic Manufacturing Systems*, Hannover, December 1994.
4. S. M. Deen. A cooperation framework for holonic interactions in manufacturing. In S. M. Deen, editor, *Proceedings of the Second Internatinal Working Conference on Cooperating Knowledge-Based Systems (CKBS-94)*, pages 103–124. DAKE Centre, University of Keele, 1994.
5. K. Fischer. Agent-based design of holonic manufacturing systems. *Journal of Robotics and Autonomous Systems*, 27:3–13, 1999.
6. K. Fischer. Holonic multiagent systems – theory and applications. In *Proceedings of the 9th Portuguese Conference on Progress in Artificial Intelligence (EPIA-99), LNAI Volume 1695*, LNAI. Springer Verlag, 1999.
7. L. Gasser. Social conceptions of knowledge and action: DAI foundations and open systems semantics. *Artificial Intelligence*, 47:107–138, 1991.
8. C. Gerber, J. Siekmann, and G. Vierke. Flexible autonomy in holonic multiagent systems. In *AAAI Spring Symposium on Agents with Adjustable Autonomy*, 1999.
9. N.R. Jennings. Agent-based computing: Promise and perils. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 1429–1436, 1999.
10. C. G. Jung. Experimenting with layered, resource-adapting agents in the robocup simulation. In *Proc. of the ROBOCUP'98 Workshop*, 1998.
11. T. Knabe, M. Schillo, and K. Fischer. Inter-organizational networks as patterns for self-organizing multiagent systems. In *Proceedings of the Second International Conference on Autonomous Agents and Multiagent Systems (AAMAS'03)*, In print.
12. A. Koestler. *The Ghost in the Machine*. Hutchinson & Co, London, 1967.
13. J. Lind, K. Fischer, J. Bcker, and B. Zierkler. Transportation scheduling and simulation in a railroad scenario: A multi-agent approach. In *Proc. of the* 4^{th} *int. Conf. on The Practical Application of Intelligent Agents and Multi-Agent Technology*, pages 325–344, London, UK, 1999. The Practical Application Company Ltd.
14. R. Rabelo, L. Camarinha-Matos, and H. Afsarmanesh. Multi-agent perspectives to agile scheduling. In R. Rabelo, L. Camarinha-Matos, and H. Afsarmanesh, editors, *Intelligent Systems for Manufacturing*, pages 51–66. Kluwer Academic Publishers, 1998.
15. M. Schillo. Self-organization and adjustable autonomy: Two sides of the same medal? *Connection Science*, 14(4):345–359, 2003.
16. M. P. Singh. Commitments among autonomous agents in information-rich environments. In *Modelling Autonomous Agents in a Multiagent World*, pages 141–155, 1997.
17. R. G. Smith. The contract net: A formalism for the control of distributed problem solving. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI-77)*, page 472, 1977.
18. H. Warnecke and M. Hüser. *The Fractal Company — A Revolution in Corporate Culture*. Springer-Verlag, 1995.
19. G. Weiss. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 1999.
20. M. Wooldridge. *An Introduction to Multiagent Systems*. John Whiley & Sons, 2002.
21. M. Wooldridge and N. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.