Towards a Constraint-based Multi-agent Approach To Complex Applications

Selvaratnam Indrakumar

Ph.D. Thesis

Submitted to the University of Surrey in partial fulfilment of requirements for the degree of Doctor of Philosophy

Artificial Intelligence Group
Department of Mathematical and Computing Sciences
University of Surrey
Guildford GU2 5XH
Surrey
United Kingdom

October 2000

Abstract

A society of agents based on constraint logic as a formalism that could support an agent environment for solving complex problems is explored. Within such society, the rights of individual agents and duties of these agents to others are often expressed in notions like communication, co-operation, negotiation, autonomy, and so on. The motivation comes from the fact that a contract net is essentially an imposition and cannot be easily reconciled with the notion of autonomy, social order, co-operation, and negotiation, and was related to a logistics problem. Our approach is to use constraint-based theories and methods to introduce an abstraction that can be used to articulate how the agents communicate, co-operate, negotiate, and how social laws are to be introduced.

A CANET (Constraint-based multi-Agent Network) framework is proposed in which various societal notions can be put into operation. The CANET approach is based on relations, not on hierarchy. Within the CANET architecture, agents interact via a constraint store that consists of basic constraints; agents are treated as autonomous, reactive, pro-active, and as a social system. Each agent consists of layers of reactive, planning, and co-operation components. Communication between agents is treated as constraint passing. Co-operation and co-ordination are treated as constraint propagation whereas negotiation is treated as constraint relaxation. Social laws are treated as hard constraints.

A prototype is developed that extends the scope of a logistics system that includes trucks operated by a number of companies. Fischer and Kuhn's (1993) approach has limitations in the sense that a contract net approach is presented, which is a highly regulated, ordered society and does not reflect preferences of agents. The role of constraint logic for such application is not explored. That fact holds in many other application areas.

CANET is implemented in Oz (now called Mozart), an object-oriented language developed at DFKI (German Research Centre for Artificial Intelligence) for logistics application. A CANET agent is seen as 'concurrent objects', and CANET multi-agents as 'concurrent objects with constraints'. We have extended the contract net message passing by constraint passing. That change facilitates communication by constraint passing. Further, we have shown that task allocation, co-operation, negotiation, social laws, and co-ordination can be discussed in a single framework based on constraint logic.

We have developed a framework in which various societal notions can be operationalised. We have made a limited comparison with other approaches, notably with a contract net approach that allows us to look at the definition of an agent, agent architectures, agent communication language, and agency in different perspectives. This research encourages us to incorporate notions like learning and evolution within the CANET architecture, and apply these to complex applications.

Acknowledgements

I am very grateful to my supervisor Professor Khurshid Ahmad for invaluable guidance throughout every stage of this research and for his criticisms, comments and suggestions.

I wish also to thank Professor Roland Price for his generous support and encouragement and to Dr David Pitt for his advice and feedback which helped me a lot to improve my ideas on my work.

Many members of AI group, specially Andrew Salway and Steve Collingham have contributed greatly to my work during the years I have been working on this thesis. I would also like to acknowledge the contribution of Mrs Caroline Jones who offered helpful comments.

Thanks are also due to Professor Gert Smolka, Professor Seif Heiridi, Christian Schultze, Martin Henz, Michael Mehl at DFKI for helpful discussions on Oz.

I wish to express my sincere gratitude to Hydraulic Research plc, and the Department of Mathematical and Computing Sciences, University of Surrey.

My warmest thanks go to my parents and my sister Kanjhana for their constant support and interest in my academic activities. My wife, Bama, has been a source of encouragement and advice. Many thanks for her love and support.

1. Introduction

Modern information systems, amplified by communication networks (such as the Internet), are typically large and complex. Existing software applications (simulation models) and their components are also complex. Agent-based computing (modelling) has provoked enormous interest in research engineering communities; software agents are becoming an essential part of these systems because they mitigate complexity. They achieve this in two important ways: technical and psychological. Technically, each agent provides a locus of intelligence for managing a subset of the information in the system, either on its own initiative or under the direction of a user. Psychologically, people need abstraction by which they can understand, manage, and use complex systems effectively. A natural and convenient abstraction is one based on separation of the complex system into components - objects - and treating them as human agents (multi-agent systems) which is much closer to people's understanding. Therefore very basic research on how such complex systems can be conceptualised and implemented using a multi-agent approach is clearly needed.

This thesis is concerned with the provision of a constraint-based multi-agent approach (CANET) for a wide range of multi-agent task scenarios such as transportation schemas, telecommunication networks, modelling and simulation. The approach is also relevant to unify various existing notions within agent-based systems such as co-operation, negotiation, task allocation, and social laws. In addition, the approach facilitates discussion of the notion of an agent.

1.1 Background to the research

The size and complexity of today's modern information systems is generally owed to the introduction and continuing development of communication networks (such as the Internet. Existing software applications (simulation models) such as telecommunications, air traffic control problems, and enterprise wide applications are complex, involving the use of experiential knowledge, the use of simulation models, access to a set of legacy systems and to various databases.

Within such complex systems, dynamic interaction between various components takes place. The input/output to a component is quite varied, and not always known in advance. The input/output can consist of partial information. There are various restrictions on each component, and how they may be able to interact with others. The relationships between components are quite complex. Changes in one component may need to propagate to other components. Components can be added or be removed from the system. Each component can have different levels of autonomy. Components may be distributed based on knowledge, resource, authority, and control.

Further, such complex systems impose a substantial burden of interpretation on the end-users of such systems. This burden manifests itself in three distinct, yet overlapping stages.

First, the burden manifests itself when the users select input data sets generally emanating from disparate disciplines, amend the data by filtering some input and smoothing others in accordance with the actual and perceived needs of the model that underpins the simulation system.

Second, the user makes assumptions about the limits and the scope of the underpinning model because of the person's training and background, or the person has either not had explained the limits and the scope in the simulation system's documentation, or the person lacks an understanding of certain bases of the model itself.

Third, the burden actually manifests when the user of a simulation system interprets the output of the system. At this stage the cumulative assumptions of the previous two stages have a substantial bearing on how the user infers the relationship between inputs and outputs and how decisions are made on the basis of the output of the simulation system.

The above is a functional, input-compute-output, description of what happens during the entire life cycle of a simulation. The description itself is a surface manifestation of how policy makers, network designers, space scientists work. The working pattern

is through fusing various items of data, by using knowledge of different domains in building and in understanding the workings of the 'real world' system being modelled, and through bringing their own knowledge and experience to bear on the results produced by models. They comprise unarticulated assumptions about the real world.

What usually happens in a simulation situation is that one person uses the knowledge of experts from different disciplines as these individual items of knowledge are encapsulated in a mathematical/numerical or logistic/heuristic model. The same is true of the choice related to the input data sets: each data set is associated with the individual knowledge sources. The data set may be an array of numbers, a collection of axioms, or it may be just a number or axiom – usually referred to as a model parameter like gravitational constant or charge parameters, in themselves a microcosm of empirical augmentation and theoretical speculations.

In general, the tools that have developed tend to stand alone, and there are no interactions between various components. That also presents various problems. Firstly, such tools do not provide overall information for decision making. Secondly, it is extremely difficult to make any changes such as legal, or business rules.

There are tools that address the interaction issues but each interaction is hard-coded. That also presents various problems. The system cannot deal with unplanned scenarios. Further, when new components are introduced, interaction needs to be specified.

Artificial Intelligence (AI) techniques, and expert systems technology in particular, have often been used to tackle some of the more difficult automation problems. After more than a decade of exploitation there are now thousands of expert systems being used in hundreds of companies all over the world to solve complex problems in numerous domains (Feigenbaum et al. 1988). However as this technology has proliferated and individual systems have increased in size and complexity, new problems and limitations have been noted (Partridge 1987; Steels 1985):

- Scalability: the complexity of an expert system may rise faster than the complexity of the domain;

- Versatility: a complex application may require the combination of multiple problem-solving paradigms;

- Reusability: several applications may have requirements for similar expertise, which has to be coded afresh in new situations;

- Brittleness: expert systems operate on a high plateau of knowledge and competence until they reach the extremity of this knowledge when they fall off sharply to the level of ultimate incompetence;

- Inconsistency: as the knowledge base increases in size, it becomes correspondingly more difficult to ensure that the knowledge embodied remains consistent.

Various approaches for circumventing these problems have been advocated. The first proposal involves building an extremely large base of common-sense knowledge (Guha and Lenat 1990). This work is predicated in two assumptions. Firstly, performing a complex task requires a great deal of knowledge about the world. Secondly, to behave intelligently in unexpected situations requires the ability to fall back on increasingly general knowledge and analogising too specific but superficially far-flung knowledge (Lenat and Feigenbaum 1991).

A second approach is to allow the sharing and reuse of knowledge (Neches et al. 1991). In this vision, rather than constructing knowledge base systems afresh, reusable components are assembled. This work is based on the observation that application systems contain many different kinds and levels of knowledge.

Finally, the approach pursued in this research is to build systems of smaller, more manageable components which can communicate and co-operate (Bond and Gasser 1988, Gasser and Huhns 1989, Huhns 1988). In the current parlance of Distributed

Artificial Intelligence that is a subfield of AI, complex problems are solved in the real world through a loosely-coupled network of specialised solvers, an intelligent agent. Each problem solver usually complements the rest, can modify his or her own behaviour as the real-world undergoes temporal or causal change, plan its subsequent actions and is able to communicate with others, resolve conflicts, co-operate with others.

This approach has several advantages. Firstly, divide and conquer has long been championed as a means of constructing large systems because it limits the scope of each processor. The reduced size of the input domain means the complexity of the computation is lower, thus enabling the components to be simpler and more reliable. Decomposition also aids problem conceptualisation; many tasks appear difficult because of the sheer size, they are too big to conceptualise all at once.

A second major advantage is that a distributed approach often provides a more natural fit to the problem. Examples include distributed sensor networks (Lesser and Corkhill 1983), air traffic control (Cammarata et al. 1983). Indeed, Hayes-Roth (1980) even goes so far as to state "all real systems are distributed".

Other potential advantages include: reusability of problem-solving components; greater robustness in the case of component failure; enhanced problem solving due to the combination of multiple problem solving paradigms and sources of information; problem solving speed up due to parallel execution, and increased system modularity (Bond and Gasser 1988).

1.2 Research problems and hypotheses/research questions

1.2.1 Knowledge-based simulation and modelling

The literature on simulation and modelling shows increasing awareness of the fact that for simulation purposes, one needs to use a mixture of analytical (usually mathematical) and heuristic (generally knowledge-based) techniques (Round 1989). And for model building, including specification and implementation, a typical scientist needs access to domain-specific databases that compose not only a data set

but a set of logical and ontological relationships between domain variables (Keller, Rimon, and Pas 1994).

Knowledge-based simulation can be traced both to the emergence of discrete event simulation languages, like SIMULA, and continuous simulation languages, like CSIM, in the late 1960s. Much later, SMALLTALK was also used in such simulations.

During the early 1980s, several systems were developed in domains that involved a number of what we now call *agents*. For instance, within such applications as factory management, cancer therapy, ecological modelling, and control and calibration of complex machinery, various tasks need to be performed.

Knowledge-based simulation systems that were developed for such domains involved a depth mixture of rule-based problem solving and planning in conjunction with the race of sophisticated mathematical models (Table 1 describes systems, their domains, the tasks they perform etc.).

SIGMA, a modelling system developed by NASA, is a good example of a system that supports the construction of complex physical systems – like atmospheric systems (Keller et al. 1994). The modelling system provides access to a suite of self-contained simulation programs, relevant data sets and a library of abstracts of domain texts. However, like knowledge-based simulation systems, SIGMA helps only in accessing a variety of data sets and programs, and still expects the user to reconcile differences and conflicts between the data and programs: co-operation, negotiation, and constraint management between the knowledge sources has to be effected by the users.

Table 1 : Knowledge-based simulation system

Acronym	Domain	Tasks performed	Implementation and Originators
		Discrete Event Simulation	
MOSYS	Factory Management System (FMS)	Use analytical techniques for determining of a FMS model Propose model refinement using a knowledge base to interpret output Refine model	PROLOG Seliger et al (1987)
ONYX	Cancer Therapy Planning	Generate plans using general treatment strategies for a given patient Design/ plan simulation about effect on the therapy on human body Rank plans using decision analysis tools	LISP Langlotz (1987)
		Model building	
ECO	Ecological Modelling	Specify ecological components in an incremental fashion (e.g. trees then grass, then sheep ...) Checking consistency of growing model using a KBS Specify and assert models Stimulate mathematical relationships (FORTRAN) and ontological relationships (PROLOG)	PROLOG/FORTRAN Meutzfield (1987)
ABLE	Knowledge-based Control for Accelerator Magnets	Plan experiments Execute rules (forward chaining) Compute fit to data (backward chaining) Executing planning rules (forward chaining)	KEE Selig (1987)
		Simulation workbenches	
SOFTLAB	Virtual laboratories	Stimulate and control of a chromatography laboratory and an electronics laboratory Used in the stimulation of rigid body dynamics	Hoftman et al (1993)
SIGMA	Knowledge-based software development environment	Construct, modify, share, and understand scientific models	CommonLISP and GINA (Keller et al 1994)
TAEMS	Co-operating problem solving Real-time Scheduling	Specify, reason, analyse, and stimulate computational environments Task oriented approaches works together with agent-based approaches Description of coordination algorithms and agents	LISP Decker and Lesser (1995)

1.2.2 Multi-agent simulation model

Conventionally, a simulation program is defined as a computer program that is used in the simulation of a model of a real-world system, and is regarded as the simulation of a mathematical model. The mathematics usually refers to differential/integral equations, usually in their finite difference, finite element or matrix manifestation approximations, or refers to stochastic description of the real world, generally with one or more distribution functions governed by statistical measures like averages, standard deviation, etc. And this mathematical interpretation of a model might also

well be true for isolated and autonomous events with well-defined boundary phenomena.

What we have in mind here is the volatile movement of stock markets, transportation of goods with strict restrictions on weight, contents, distribution of energy, computation of tax on strict criteria on salary, age, martial status etc. Within the applications, the model is not just a set of equations, but a set of entities. The mathematical/logical model is encapsulated within the entity.

A multi-agent simulation model is based on the idea that programs exhibit behaviours that can be entirely described by their internal mechanisms, the instructions. In a multi-agent simulation, the model is not a set of equations as in mathematical models, but a set of entities that can be described by the quadruple "agents, objects, environment, communications" (Ferber and Drougoul 1992) where agents are the set of all simulated individuals, objects are the set of all passive entities that do not react to stimuli, environment is the topological space where agents and objects are located and communication is the set of all communication categories.

According to Ferber and Drogoul (1992), the aim of using a multi-agent framework for simulation is threefold. Firstly, it can be used to test hypotheses about the emergence of social structures from the interactions of the agents and the reasoning, reacting capabilities of the agent. The second goal relates to the claim that such a simulation can help to build theories that contribute to the development of a general understanding of ethnological systems, by relating to behaviour and structural and organisational properties. Thirdly, such an approach can be used to integrate different theories from various disciplines into a general framework.

1.2.3 A constraint-based approach to agency and its uses in modelling and simulation?

What we are arguing here is that a simulation is situated: situated in a specific physical and temporal location, situated in the context of the modeller and its user. An intelligent simulation system can not only help in the simulation process, but should be able to autonomously articulate its input data requirement, and be explicit about the knowledge it uses in processing the data. An intelligent simulation should

be aware of the multidisciplinary data and knowledge required for even the simplest of simulations, and should be able to help its user in interpreting the output.

An intelligent simulation system, therefore, not only has sophisticated mathematical models implemented on a computer system, but comprises knowledge bases that enable the users to seek help in the input, compute, and output functions.

The aim of this thesis is to discuss some of the crucial issues in simulation and modelling of large-scale systems. We attempt to show that modelling such systems requires a considerable amount of knowledge from diverse sources, and such knowledge can be better managed if there are pro-active entities working in conjunction with conventional simulation and modelling systems.

This brings us to the notion of agency and the societal issues that are connected with the discussion of how groups of people, or, more accurately, computer programs mimicking people, work together to achieve a common set of goals.

Within complex systems, we have mentioned numerous components working together to solve common goals. Within such systems, the behaviour of components, and the interaction between them, may be constrained. Moreover, based on the discussion of the previous section, it is relevant that there is a necessity to have general architecture to unify distinct stand-alone participants such as various databases, knowledge bases, and browsers.

The research question is as follows: how can we create a society of agents for solving complex problems, where the rights and duties are expressed in a common framework? That research question triggered the following sub questions. How is everything connected in complex applications? Would such systems accommodate constraints and irregularities? How is it possible to pass partial information between entities? How can everything dynamically interact? How should a behaviour of a component be constrained as well as the interaction between components? How can changes in one component be propagated? How may components communicate between each other given the consideration of constraints? Examples for interactions include atoms, humans, computers, household appliances, and physical bodies.

Our own experience in the field of constraint satisfaction suggests that it is important to investigate the promise of constraint-based approaches for simulating the behaviour of agents that do, or are expected to, work together.

1.3 Justification for the research

According to BIS strategic decisions: "Agents will be the most important computing paradigm in the next 10 years. By the year 2000, every significant application will have some form of agent enablement" (Aparicio 1996). And according to the Gartner Group Report: "Worldwide market for agent software will grow by an estimated value of $3M in 94 to $2.6B by the year 2000"(Aparicio 1996).

We have discussed that software agents are thus becoming an essential part of complex systems because they mitigate complexity. They achieve this in two important ways: technical and psychological.

Technically, each agent provides a locus of intelligence for managing a subset of the information in the system, either on its own initiative or under the direction of a user.

Psychologically, people need abstraction by which they can understand, manage, and use complex systems effectively. A natural and convenient abstraction is one based on disaggregation of the complex system into components - objects - and treating them as human agents (multi-agent systems) which is much closer to people's understanding. Therefore very basic research on how such complex systems can be conceptualised and implemented using a multi-agent approach is clearly needed.

Complex applications consist of various knowledge sources. Within such knowledge sources there is a degree of autonomy, duty, and social ability. Moreover, there are constraints that are associated with these applications such as precedence, and resource. Such applications require an analytical approach to solve problems that may arise.

Traditionally, the problems in these areas are addressed by distributed artificial intelligence (DAI) methods. In the current parlance, DAI shows multiple inheritance from behavioural and cognitive psychology, sociology, anthropology (particularly ethnography), computational theories and neurobiology. Complex problems like policy planning, network design, war games, are solved in the real world through a loosely coupled network of specialised problem-solvers. Each problem-solver, an intelligent agent, usually complementing the rest, can modify his or her behaviour as the real world undergoes temporal and causal change, can plan its subsequent actions, can communicate with others, can resolve conflicts, can impose his or her own ideas, can adapt other ideas. In another words, research in DAI is concerned with understanding and modelling action and knowledge in collaborative enterprises. People usually distinguish two main areas of research in DAI (Bond and Gasser 1988): distributed problem solving and multi-agent systems.

Distributed problem solving (DPS) considers how the task of solving a particular problem can be divided among a number of modules (or "nodes") that co-operate in dividing and sharing knowledge about the problem and about its evolving solution(s). In a pure DPS system, all interactions (co-operation, co-ordination if any) are incorporated as an integral part of the system.

Research in multi-agent systems (MAS) is concerned with the behaviour of a collection of (possibly pre-existing) autonomous agents aimed at solving a given problem. A MAS can be defined as "a loosely coupled network of problem solvers that work together to solve problems that are beyond their individual capabilities" (Durfee at al. 1989). A multi-agent system, that is a sub-field of DAI, is concerned with co-ordinating intelligent behaviour among a collection of autonomous intelligent agents and how they can co-ordinate their knowledge, goals, skills, and plans jointly to take action to solve problems.

Jennings et al. (1998) argues that there are two major impediments to the widespread adoption of agent technology: (i) the lack of systematic methodology enabling designers to clearly specify and structure their applications as multi-agent systems; and (ii) the lack of widely available industrial strength multi-agent system toolkits. In addition, they show agent-based computing to be chaotic and incoherent.

Constraint-based approaches are applied to applications such as scheduling and so on. However, there is not much emphasis on the idea of agents. Currently, the scope of those applications is very limited. However, it is believed that as these complex applications tend to have various constraints, they can be used to define various behavioural and interaction aspects.

1.3.1 Agent-oriented programming

Agent-oriented programming (AOP) is a term that Shoham (1977) has proposed for a set of activities necessary to create software agents. What he meant by 'agent' is "an entity whose state is viewed as consisting of mental components such as beliefs, capabilities, choices, and commitments". Agent-oriented programming can be thought of as a specialisation of object-oriented programming (OOP), with constraints on state-defining parameters, message types, and methods as appropriate. From this perspective, an agent is essentially "an object with attitude". Table 2 summarises the relation between AOP and OOP.

Table 2: OOP versus AOP

Parameters	OOP	AOP
Basic unit	Object	Agent
Parameters defining state of basic unit	Unconstrained	Beliefs, commitments, capabilities, choices, ...
Process of computation	Message passing, and response methods	Message passing and response methods
Type of message	Unconstrained	Inform, request, offer, promise, decline, ...
Constraints on methods	None	Honesty, consistency

An agent's "mental state" consists of components such as beliefs, decisions, capabilities, and obligations. Shoham formally describes the state in an extension of standard epistemic logics, and defines operators for obligation, decision, and capability. Agent programs control the behaviour and mental state of agents. An agent interpreter executes these programs. In the spirit of speech-act theory, interagent communication is implemented as speech-act primitives of various types, such as inform, request, or refrain.

An agent interpreter assures that each agent will iterate through two steps at regular intervals: read the current messages and update its mental state (including beliefs and commitments); and execute the commitments for current time, possibly resulting in further belief change. Shoham's original interpreter, AGENT-0, implements five language elements:

- Fact statements ("John is an employee of NET");
- Communicative action sequence (inform, request, refrain);
- Conditional action statements ("If, at time *t*, you believe that John is an employee of NET, then inform the agent *A* of the fact");
- Variables; and
- Commitment rules.

The basic concepts described by Shoham have influenced the direction of many other agent researchers. He and his colleagues have continued their investigations on several fronts including mental states, algorithmic issues, the role of agents in digital libraries, and social laws among agents.

Both the theoretical developments of mental categories, and the AGENT-0 programming language, concentrated on a single agent. Indeed, the view promoted was of agents functioning autonomously. However, if a society of agents is to function successfully, some global constraints may be imposed. Such an approach is not suited for industrial strength applications where a robust response is required with distinct constraints.

Apart from Shoham's work, which has weaknesses such as a single-agent solution for complex applications, there is a relative neglect of specific research of constraints on behaviour and interaction of agents for modelling and simulation. In chapter 2, we will review the field of agents, and reiterate the neglect in research on interaction and behaviour with constraints and methodological weaknesses.

1.3.2 Logic programming to constraint logic programming

Logic programming is an appealing language for complex problems, thanks mainly to its relational form and its nondeterminism. Its relational form makes it convenient for stating constraints, as a constraint is nothing other than a relation. Its nondeterminism makes it a powerful conceptual tool for designing backtracking problems.

Unfortunately, logic programming in its current state of development is very inefficient for executing the natural formulation of problems. The reason is that this formulation, when executed on a logic programming system, leads to a generate and test, or a standard backtracking (i.e. depth first search with chronological backtracking) approach. Both search procedures exhibit pathological behaviour and their performance decreases drastically as the problem size grows. As a matter of fact, these search procedures are oriented to recovering from failures and do not try to avoid failures. The basic reason for this inefficiency comes from the way constraints are used, only to reduce the search space after discovering a failure.

The inefficiency of generate and test and standard backtracking must be contrasted with the results of search procedures based on consistency techniques. Consistency techniques are based on the idea of *a priori* pruning, that is, using the constraints to reduce the search space before discovering a failure. They originated from Waltz's filtering algorithm (Waltz 1972). The pruning in consistency techniques is achieved by spending more time at each node of the search tree removing combinations with values that cannot appear in a solution. Thus the procedures are oriented toward the prevention of failures and enable both an early detection of failures and a reduction of the backtracking and the constraint checks. Both experiments and theoretical studies have proved the values of *a priori* pruning. In most cases, a substantial improvement in efficiency over standard backtracking is considered a fundamental primitive of reasoning for solving Constraint Satisfaction Problems (CSPs).

1.3.3 Societal notions and computational agency based on consistency

The implementation of an intelligent simulation system can, perhaps, benefit from developments in distributed problem solving. A simulation system can be construed to involve interaction between a number of autonomous programs – or agents in DAI and a society of agents having the societal attributes of communication, co-operation, and advocacy.

Broadly speaking a simulation can be viewed as an interaction between agents that can help with the input data, agents that are knowledgeable about the simulation model and its implementation, and agents that can help in the interpretation of the output data. The input agents can help in selecting and accessing autonomous data sets and it is possible to benefit from developments in distributed data base systems – a collection of data sets that can be accessed through fuzzy queries together with facilities to transform data and filter 'irregularities'.

Many of the extant simulation systems, ranging from CSIM (continuous simulation modelling) to NASA's SIGMA, to varying degrees, help the user in getting data from external sources, contain help files related to simulation engines and attempt to interpret the output. However, in all these tasks a pro-active user is a must and such a user is generally very experienced.

In order to propose an agent-based intelligent simulation system, it is perhaps useful to briefly describe what we regard to be some salient features of a typically distributed artificially intelligent system, particularly multi-agent systems. We would like to argue that parallel developments in AI, especially constraint-based problem solving theories and methods, can be used to introduce certain societal features within a society of autonomous programs at a greater level of abstraction. Such an abstraction can be used to articulate how the agents communicate, negotiate, and so on.

The point here is that much of the DAI literature discusses the notion of a society of agents, involving the unarticulated assumptions about the rights of individual agents

and duties of these agents to others often expressed in notions like communication, co-operation, negotiation, and so on. Bond and Gasser (1988), and Wooldridge (1995) discuss various interactions. We present a constraint-based multi-agent framework in which these notions can be put in operation.

1.3.4 The transportation-domain' problem revisited

The dominant metaphor in DAI, or, more specifically in distributed problem solving, is that of agents organised in a network: each node represents an agent and the links represent conduits for communication between agents. These nodes are generally expected to be 'sophisticated systems' in their own right in that they are expected to represent a complex real-world entity.

The extent to which multi-agents "can modify...[their] own behaviours as circumstances change and plan [their]...own communication and co-operation strategies with other nodes" (Durfee, Lesser, and Corkhill 1992) varies from one multi-agent system to the other as reported in the literature. For example, Fischer and Kuhn (1993) note that "a central problem in the study of autonomous co-operating is that of ... [establishing] mechanisms for controlling the interaction between different parts ... [or agents] of the system". These authors note that in an implicit sense multi-agent co-operation, in a problem-solving context, has been simulated by using methods based in dynamics programming and operations research – usually through the computation of the so-called cost functions. Within the scope of DAI, the authors describe three models of how a society of agents can be organised and deployed for simulating complex problems in vehicle scheduling in response to customer demands.

For Fischer and Kuhn, a society of agents, represented by $A= \{a_1 ... a_n\}$ is capable of executing a set of tasks $T = \{t_1 ... t_n\}$. A problem can be solved by A by decomposing the problem in subproblems that can be tackled perhaps by the execution of one or more tasks defined in T. These authors deal with the problem of organising the rota for a number of trucks, indeed trucking companies, that may carry goods across Germany.

This is a classical logistic problem, that of optimising the means against a range of ends. The authors have used the extended contract net model for organising agents in

order to manage a society dominated by a broker working together with trucking companies, trucks, and drivers. The contract net model is used when a dominant agent broadcasts a problem and seeks 'contracts' for the solution of the problem from a society. Fischer and Kuhn (1993) have a 'manager' agent, the so-called broker agent, that liases with the customers and seeks their orders. These orders are relayed to a number of 'worker' agents, the trucking companies, who, in turn, send back 'costed' bids for the tasks. This cost is calculated by relaying a message to the subordinate drivers, subordinate to the companies, who after checking the availability of their trucks respond to the trucking companies. In effect, a highly regulated and ordered society of agents with agents at successive levels of hierarchy having less and less autonomy: this situation is more like a military organisation than a society of truckers involved in fierce competition.

The extended contract net approach to the transportation domain problem was operationalised through the use of an object-oriented concurrent language, Oz, a language developed by the German Institute of Artificial Intelligence (DFKI). Using a hierarchical decomposition model, the inter-agent communication is essentially the cost transmitted from the nodes in the agent hierarchy to the apex of the tree. Table 3 summarises the simulation of Fischer and Kuhn (1993) indicating the roles of the agents, the tasks they are supposed to execute and how these tasks have been implemented as encapsulated behaviour through the use of the method construct.

Table 3: The agents, roles, tasks, and methods used in the transportation simulation (Note that rem stands for remove, and init for initiation)

Agent(s)	Role(s)	Task	Method
Broker	Master	Deliver tasks to companies Select minimum priced driver	Init, Add(companies), rem(companies), add(Driver), rem(Driver)
Companies	Slave/Master	Deliver tasks to drivers Select minimum priced drivers	Announce, addcity, init
Drivers	Slave/Master	Computes the cost for tasks Plan, control of driver agents	Announce, init(Window), init(create the truck)
Trucks	Slave		Init, drive, move, route

The limitations of Fischer and Kuhn's (1993) simulation are as follows:

a. The contract net approach involves a highly regulated and ordered society of agents that does not reflect reality. It is difficult to accommodate various preferences of agents.
b. Communication between agents tends to be message passing. That has limitations in passing partial information between agents.
c. Co-operation/co-ordination between agents is via contract net. That does not allow horizontal co-operation between trucks.
d. Social laws are not addressed. Common laws within multi-agent settings for solving problems are not addressed. This would remove the necessity of hardcoding laws to different agents within the application.
e. Task decomposition and allocation is via contract net. The Manager always chooses the cheapest offer in selecting the agents. The approach is very hierarchical.
f. Negotiation is addressed via contract net. Within the simulation, negotiation is not really discussed in the sense that the manager always selects the cheaper cost. There are no negotiations between the agents at the same levels.
g. Various constraints that are applicable such as precedence, resource, temporal are not addressed.
h. Learning and evolving of driver agents is not addressed.

1.3.5 Task decomposition and task allocation in less centralised models

Fischer and Kuhn (1993) have discussed the rather restricted communication available to them even in an extended contract model. The authors have argued for two 'liberal' regimes regarding the operation of the society of agents and discussed the decentralised task model and the completely decentralised task model for task decomposition and task allocation. In these two models, the role of the manager agent is successively reduced such that in the completely decentralised task model the manager is actually surplus to requirements.

Fischer and Kuhn have actually extended the contract net proposal in order to demonstrate how co-operation can be effected amongst agents on the one hand, and in

order to demonstrate the practical relevance of an agent-oriented paradigm for solving real-world problems (cf. the transportation domain problem) on the other. The extended contract net protocol extends the number of speech acts that were available to the manager from two, i.e. reject and grant, to four, temporal reject, temporal grant, definitive reject, definitive grant (1993:36). By this extension of the speech acts, the authors will deal with much more realistic problems in the domain: namely, the unbooked leg co-operation and the coupling of long-distance transportation and local distribution (1993:91). This allows the authors to argue that this distributed task decomposition and task allocation is possible by allowing agents at the same level to communicate among themselves before they finally report to the manager just above themselves. Hence, orders from customers become more complex and include some notion of time: expected time of departure, expected date of arrival, duration of the journey and so on.

1.4 Methodology

The aim of the thesis is to discuss some of the crucial issues in simulation and modelling of large-scale systems. It is intended to show that modelling such systems requires a considerable amount of knowledge from diverse sources and that such knowledge can be better managed if there are pro-active data and knowledge sources working in conjunction with conventional simulation and modelling systems. This brings us to the emergent notion of agency and the societal issues connected with the discussion of how groups of people, or, more accurately, computer programs mimicking people, work together to achieve a common set of goals.

The aim of the work is to propose a framework to discuss various behavioural aspects of agents and the interactions between agents. The interactions include co-operation, negotiation, task allocation, and social laws.

Our experience in the field of constraint-based systems suggests that it is important to investigate the promise of a constraint-based computing paradigm for simulating the behaviour of agents that do, and are expected to, work together (Selvaratnam 1993, Selvaratnam & Ahmad 1995).

Our approach is to create a society of agents. Within such a society, the rights of individual agents and duties of these agents to others are often expressed in notions like communication, co-operation, negotiation, autonomy, and so on. Our approach is to use constraint-based theories and methods to introduce an abstraction that can be used to articulate how the agents communicate, co-operate, negotiate, and how the social laws are to be introduced. We present a CANET (Constraint-based multi-Agent NETwork) framework in which these notions can be put into operation.

Within the proposed approach, agents communicate between each other. Agents are autonomous entities, with awareness of others. Agents or users send messages to each other to constrain behaviour. Each agent's behaviour may be constrained, and the interaction between agents may also be constrained.

Co-operation is treated as constraint passing, whereas negotiation is treated as constraint relaxation. Social laws are treated as hard constraints that cannot be relaxed. Task allocation is treated as constraint satisfaction.

A transportation scenario is simulated to demonstrate the hypothesis that the behaviour of an agent and the interaction between agents may be related to constraint-based techniques. In particular, constraint communication between agents is demonstrated. We will make a comparison with Fischer and Kuhn (1993).

A constraint-based multi-agent approach for complex applications is proposed. Within such an approach, the behaviour of an agent and the interactions between agents are related to constraint-based interactions. Within the framework, constraint satisfaction is an underlying mechanism compared to the unification mechanism used in Prolog language. The feature of consistency is exploited for agency.

The proposed approach is natural in the sense that it addresses constraints at various levels. Modelling and simulation of complex systems can be interpreted as the interaction between agents and constraints. A wide range of applications can benefit from the method.

Other researchers put more emphasis on discussing various interactions provided by a survey. However, no results are achieved in unifying various interaction approaches. This thesis tries to find a constraint abstraction for agent interactions. A critical review of the existing agent-based systems, notions of an agent, and constraint-based systems is presented.

1.5 Contributions

1.5.1 Inside the agent community

As far as theory is concerned, this research shows that the behaviour of an agent and the interaction between agents can be related to constraint-based techniques. Our contribution to the field of agents is that constraints and agents are synthesised through our system, CANET. Our hypothesis is that the behaviour of an agent and the interactions between agents may be related to constraint-based mechanisms. In addition, various existing interaction notions are unified within such an approach. Thus a distinct interpretation for such notions is provided. Agents work under constraints of duties and rights. As far as the practical framework is concerned, agents are concurrent objects working under constraint logic. That facilitates the existing architecture such as contract net to be extended so that agents interact under constraint logic. The approach allows the agents to communicate by constraint passing compared to message passing in object-oriented language.

It is hoped that this research will contribute towards representing agent behaviour and interaction via constraint logic. As far as the end-user is concerned, this allows the complexity of designing interactions to be radically simplified, their development period sharply reduced and ease potential future modification.

1.5.2 Outside the agent community

Outside the agent-based community, the contribution of the research is to provide a CANET approach for modelling and simulation of complex systems.

Within the constraint-based community, the research contributes to making constraint logic a programming language not only for scheduling type applications, but also applications where agents are applied. Within the constraint-based community, the

emphasis is on variables, domain values, and constraints, and the concept of an agent is not explored. By synthesising the field of agents, constraint logic can be applied within DAI that will be discussed in chapter 3.

1.6 Outline of the thesis

Chapter 2 reviews existing agent-based systems. We critically analyse the definition of an agent, and we argue for a need of an abstraction. The notion allows various definitions to be related. From the evolution of agent-based systems, how a constraints-based agent system plays a role is addressed. Various agent architectures are discussed. We conclude by drawing the conclusions of previous reviews and motivating the research. This allows the role of constraints in such architectures in subsequent chapters to be investigated.

Chapter 3 presents the CANET (Constraint-based multi-Agent NETwork) approach that is a theoretical foundation of the proposed research. Discussion on the evolution of constraint-based systems will position our research in the work. Various notions of constraint-based systems are addressed. This will be applied in chapter 4 when discussing the synthesis of constraints and agents with the aim of addressing constraint-based applications to the proposed methods.

Chapter 4 presents the design and implementation of a CANET approach, and demonstrates the feasibility of the approach on several examples. A transportation scenario is simulated to demonstrate the CANET concepts concerning co-operation, and task allocations, and so on.

Finally, chapter 5 draws together the strands of research presented in this thesis and highlights some areas for further investigation. A brief comment on this research is also given.

1.7 Delimitations of scope and key assumptions

This thesis tries to address how various modules co-operate, negotiate and so on from a constraint-based point of view. How agents learn, and how agents evolve are not addressed. Further, there is potential scope to carry out research on areas such as

merging of agents, cloning of agents, and learning. There is also scope to develop an agent-constraint toolkit for building complex applications. The key assumption of the research is that "everything is connected".

2. Literature review - Agent-based systems

The aim of this chapter is to build a theoretical foundation on which the research is based by reviewing the field of agents to identify research issues that are controversial and have not been adequately answered by previous researchers. It shows the link between research problems and the wider body of knowledge. That is, the literature review includes the immediate disciplines of the research problem. The literature review is focused in the sense that another related discipline, constraint satisfaction, is addressed in the next chapter.

The main goals of this chapter are to overview the rapidly evolving area of agents. This is achieved by discussing the basic concepts of multi-agent systems, by critically reviewing the definition of an agent, by researching into the evolution of agent-based systems, and by listing the agent architectures. Agent-based approaches for applications are also listed.

2.1 Motivation

Agent technology has undoubtedly made a large impact on computing during the last few years (Nwana 1996). Agent software is a rapidly developing area of research. The word *agent* is as currently in vogue in the popular computing press as it is within artificial intelligence (AI) and computer science communities. Agent technology provides an exciting new computing and problem-solving paradigm.

There is a wide range of application domains that are making use of agent-based systems. Agent applications are being developed for fields such as manufacturing, entertainment, electronic commerce, user assistance, service and business management, information retrieval research, energy systems, and autonomous space probes.

2.2 Evolution of agent-based systems

The quest to design intelligent control in artificial systems dates back a long time. Even before electronic computers had been invented, the engineer James Watt (1736-1839) popularised the use of mechanical feedback control as a way of automatically

regulating the velocity of rotation in steam engines, thereby controlling their energy intake. Concepts such as stability in dynamical system, control of systems are still an area for research by using agent-based techniques. Another field that precedes artificial intelligence is cybernetics, the aim of which is to unify mathematically disparate studies of control and communication in animals and machines.

At the early stages of agent development, due to centralised approaches, a single agent tended to perform very complex tasks. However, as the idea of decentralisation became popular, there were developments in distributed systems with multiple agents. Currently, even though various interactions are discussed, they are mostly hard-coded. There is a relative neglect of developing a framework in which various interactions can be simulated. In chapter 3, we will review the field of constraints, and propose a CANET approach for complex applications. Table 4 summarises the evolution of agent-based systems.

Table 4: Evolution of agent-based systems

System/Field	Name/ Examples	Aim/Characteristics
Mechanical feedback control	James Watt (1800)	To refine actions and to produce stability in dynamical systems
Cybernetics	Norbert Weiner (1940-50s) Ross Ashby Grey Walter	To unify mathematically disparate studies of control and communication in animals and machines
AI	John von Neumann (1970s) Alan Turing John McCarthy Marvin Minsky	To build systems that perform some rational operations Machine intelligence, theory of computability Applications: Game playing, diagnosis, planning and natural language understanding
Robotics Shakey project Cart Project	(1980s) Charles Rossen, Nils Nillson John McCarthy, Les Earnet, Hans Moravec	To build intelligent system coupling of computer programs to television cameras and mechanical robot arms. Application: Mobile robots
Distributed AI Multi-agent systems Basic architectures actor-based blackboard contract net Deliberate Non-deliberate Layered/hybrid	 Hewitt (1977) Fennel and Lesser (1977) Smith (1977) e.g. IRMA, AOP, GRATE e.g. PENGI, Agent Network architecture e.g. INTERRAP, Too TouringMachines	Concurrency and distribution Co-ordination of intelligent behaviour among a collection of autonomous agents Symbolic manipulation Reactive behaviour Integration of deliberate and non-deliberate architecture with several hierarchical functional modules General application: Most are prototypes, there are very few applications, e.g. transportation domain, electricity transportation management, telecommunications and so on.

2.3 Terminology of an agent-based system

In this section, the terminology of an agent-based system is reviewed. The emphasis is placed on the definition of an agent. Firstly, we will briefly look at the parent field of agent-based systems.

Researchers in DAI are concerned with understanding and modelling action and knowledge in collaborative enterprises. People usually distinguish two main areas of research in DAI (Bond and Gasser 1988): distributed problem solving and multi-agent systems.

Distributed problem solving (DPS) considers how the task of solving a particular problem can be divided among a number of modules (or "nodes") that co-operate in dividing and sharing knowledge about the problem and about its evolving solution. Multi-agent systems are an outgrowth of the Distributed Artificial Intelligence community. Durfee et al. (1989) define a multi-agent system as "a loosely-coupled network of problem solvers that work together to solve problems that are beyond their individual capabilities".

Research in multi-agent systems is mainly concerned with co-ordinating intelligent behaviour among these agents, how they co-ordinate their knowledge, skills and plans jointly to take action or to solve problems.

These problem solvers, which are essentially autonomous, distributed and maybe heterogeneous in nature, are called *agents* and usually have a single locus of control and/or intention.

An agent is a computer system situated in some environment, and which is capable of autonomous action in this environment in order to meet its design objectives. The key abstraction used is that of an agent. For example, considering a transportation scenario, agents would be a broker agent, or company agent, or driver agent, or truck agent.

In the recent past the term agent has been used unsparingly to refer to any software system which has attributes of intelligence, autonomy, perception, or acts on behalf of a user. There is no standard definition of an agent on which consensus exists, and researchers over time have proposed various definitions of the term (and are still doing so).

The reason why it is so difficult to define precisely what agents are is that, within the software fraternity, the word is really an umbrella term for a heterogeneous body of research and development. As Nwana (1996:6), while having to define the term agent, says:

> "When we really have to, we define an agent as referring to a component of software and/or hardware which is capable of acting exactingly in order to accomplish tasks on behalf of its user. Given a choice, we would rather say it is an umbrella term, meta term or class, which covers a range of other more specific agent types, and then go on to list and define what these other agent types are. This way, we reduce the chance of getting into the usual prolonged philosophical and sterile arguments which usually precede the former definition, when any old software is conceivably recastable as agent-based software".

Agent software is a rapidly developing area of research. The word is overused in the literature. In this section, the specific criteria of an agent will be addressed. There is a necessity of classifying and combining various approaches that exist at the present moment. Recent discussions by various authors vindicate this. There are a few reasons why it is so difficult to define precisely what agents are. Firstly, agent research does not use the term in the same way as other terms might be applied, for example, the term *agent* is used widely in everyday parlance as in travel *agents*, estate *agents* and so on.

Secondly, even within the software fraternity, the word agent is an umbrella term for a heterogeneous body of research and development. The response of some agent researchers to such phenomena is to introduce new terms such as *knobots, softbots, taskbots* and so on. They may have some reasons to invent synonyms. Firstly, agents may come in many physical guises. Secondly, agents can play many roles. Furthermore, due to the multiplicity of roles agents can play, there is a plethora of adjectives which precede the word agent, such as search agents, report agents, presentation agents and so on.

The Reader's Digest Oxford Wordfinder (1993) defines an agent as "a person who acts for another in business, politics, and etc. [L agentia f. L agere do]". One may deduce from this definition that agents do things and act. However, the action-based analysis is not sufficient for the notion of agency. There is a necessity to consider other properties for an agent. The notion of an agent is interpreted in a variety of ways. Researchers in distinct fields tend to put forward their individual notion of an

agent. There is a need to explore these notions to find out some general properties. These definitions vary from 'weak' to 'stronger' categories.

Wooldridge et al. (1994) distinguish two usages of the term 'agent'. Firstly, they discuss an agent as a hardware or software system with properties: autonomy, social ability, reactivity and pro-activity. Secondly, they argue that the agent has a stronger and more specific meaning. In other words, in addition to the properties discussed, a computer system is either conceptualised or implemented using concepts that are more usually applied to humans.

According to Jennings (1995), the term *agent* (and hence *agent-based computing*, *agent-based system*, *multi-agent system*) is being used within information technology to describe a broad range of computational entities. He distinguishes three classes of agents. Firstly, there are "gophers" agents, which can execute straightforward tasks based on pre-specified rules and assumptions. Secondly, there are "service-performing" agents, which execute a well-defined high-level task at the request of a user. Finally, there are "predictive/pro-active" agents, that volunteer information or services to a user, without being asked, whenever it is deemed to be appropriate. He advocates properties such as autonomy, social ability, reactivity and pro-activity. Such notions are explained below:

Autonomy: agents should be able to perform the majority of their problem solving tasks without the direct intervention of humans or other agents, and they should have a degree of control over their actions and their own internal state.

Social ability: agents should be able to interact, when they deem appropriate, with other artificial agents and humans in order to complete their problem solving and to help others with their activities. This requires that agents have, as a minimum, a means by which they can communicate their requirements to others and an internal mechanism for deciding when social interactions are appropriate.

Reactivity: agents should be able to respond to the changes in the environment.

Pro-activity: agents should not simply act in response to their environment, they should be able to exhibit goal-directed behaviour.

Table 5: Definitions of an agent

Name	Definition
Steiner, Mahling & Haugeneder (1990)	Agent as a mouth-head-body (Mouth: communicator; Head: reasons about functions of the body and exerts agent control; Body: describes the application-oriented processing facilities and knowledge of the agent).
Brustolini (1991)	Autonomous agents are systems capable of autonomous, purposeful action in the real world.
Pan & Tenenbaum (1992)	Complex processes such as tasks in cognitive terms (i.e. what to look for? What to do? And who to tell?), and are entrusted to an intelligent agent for execution. Agents interact with each other via a message bus or through a shared, distributed knowledge base.
Ferguson (1992)	Any goal-directed computational process capable of robust and flexible interaction within its environment.
Shoham (1993)	An agent is described in terms of beliefs, goals and commitments.
Muller & Pischel (1993)	An agent is described as a knowledge-based system composed of a knowledge base and a control unit.
Nadoli & Beigel (1993)	An agent is described in terms of a set of rules governing the behaviour of a set of objects. A local 'blackboard' performs the reasoning of the agent. Each blackboard contains the facts known to an agent.
Smith, Cypher & Spohrer (1994). (KidSim agent)	An agent is defined as a persistent software entity dedicated to a specific purpose. 'Persistent' distinguishes agents from subroutines; agents have their own ideas about how to accomplish tasks, their own agendas. 'Special purpose' distinguishes them from entire multifunction applications; agents are typically much smaller.
Wooldridge & Jennings (1994)	In a general sense, an autonomous self-contained, reactive and pro-active system. Properties of an agent are autonomy, social ability, reactivity and pro-activity. However, specifically, an agent is a computer system that is either conceptualised or implemented in terms of concepts applied to humans (belief, desire, and intention). Other properties of an agent are mobility, security and emotion.
Russell & Norvig (1995). (AIMA agent)	An agent is everything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors.
Maes (1995)	Autonomous agents are computational systems that inhabit some complex dynamic environment, and by doing so realise a set of goals or tasks for which they are designed.
Coen (1994) (SodaBot agent)	Software agents are programs that engage in dialogues [and] negotiate and co-ordinate transfer of information.
Minsky (1994)	From the dictionary: "A person who acts on behalf of another person, business, government etc./A person or thing that acts or has the powers to act/..." They all suggest the agent be seen as having some specialised purpose. Agency: A business or other organisation providing a specific service. The term *agency* is used to suggest the image of an office or an organisation that is composed of several interacting agents.
Jennings et al. (1995)	Within their ARCHON architecture, individual problem solving entities are called agents; these agents can control their own problem solving and interact with other community members. The interactions typically involve agents co-operating and communicating with one another in order to enhance their individual problem solving and to better solve the overall application program. Each agent consists of an ARCHON layer (AL) and an application problem (known as an Intelligent Systems or IS).
Nwana (1996)	Presents a complementary view of agenthood to Jennings et al (1995), underpinned by the attributes of autonomy and co-operative ability, but also including learning ability, i.e. the ability to improve performance over time. Nwana (1998) discusses Zeus toolkit.
Tambe (1997)	Teamwork in multi-agent is emphasised.
Muscettola et al. (1998)	A remote agent architecture is proposed. The components of the architecture include temporal planner/scheduler (PS) , with an associate mission manager (MM), reactive executive (EXEC), and a model-based mode identification and reconfiguration system (MIR).

In summary, there are various definitions for the notion of agent. However, among most researchers, there is consent that the agent should be defined as an autonomous, reactive, pro-active, and reactive system. The above discussion is summarised in Table 5.

There is a need to explore agency. Currently, interactions tend to be applications specific, and operational in an ad-hoc manner. In general, there is no consideration to constraints, and irregularities. This thesis tries to address that research gap.

2.4 DAI and classification models

2.4.1 Agent classification models

In Table 5, we summarise various definitions of an agent. Analysing the definition of an agent, we have identified various properties and their meanings that are listed in Table 6.

Table 6: Agent properties

Property	Meaning
Reactive (sensing and reacting)	Responds in a timely fashion to changes in the environment.
Autonomous	Exercises control over its own actions.
Goal-oriented (pro-active)	does not simply act in response to the environment.
Temporally continuous	is a continually running process, not "one shot" computations that terminate.
Communicative (socially able)	Communicates with other agents, perhaps including people.
Learning (adaptive)	Changes its behaviour based on its previous experiences.
Mobile	Able to transport itself from one machine to another. They can carry data along with intelligent instructions, which can be executed remotely.
Flexible	Actions are not scripted.
Character	Believable "personality" and emotional state.
Personisability	The point of an agent is to enable people to do some task better. Since people don't do all the same tasks, and even those who share the same task do it in different ways, agent must be educable in the task and how to do it.
Risk and trust	The idea of an agent is intimately tied up with the notion of delegation. In this situation, one has to balance the risk that the agent will do something wrong with the trust that it will do right.
Graceful degradation	Bound up in the notions of risk, trust, and domain, agents work best when they exhibit graceful degradation in cases of communications mismatch, or a domain mismatch.

Researchers tend to create new agents by composition of various agent properties. For example, an agent may be called a collaborative agent if this agent emphasises

autonomy and co-operation. Having said this, for example, Jennings (1995) argues that the collaborative agent may have to negotiate to reach agreements. The definition of an agent seems to be developed according to the term (e.g. collaborative agent), and to the area of research and development (e.g. taskbot agent). Table 7 depicts the agents.

Table 7: agent definition by composition

Agent	Composition
Smart	Co-operation, learning, and autonomy
Collaborative	Co-operation , and autonomy
Interface	Autonomy, and learn
Sodabot	Co-ordination, and negotiation
Hybrid	Reactive, pro-active, co-operation

Within agent architectures, various subtle notions need to be accommodated. For example, an agent may be autonomous, and co-operative. In another words, if an agent is able to carry out a specific task, that agent is autonomous. If that is not the case, the agent needs to co-operate with another agent to achieve goals. Other tensions include negotiation and compliance, static and dynamic, temporally continuous and one-shot, and tasks to be achieved and constraints. To address the subtle nature of the properties, researchers tend to develop ad-hoc models based mainly on rule-based hard-coded representation.

Various agent properties can be categorised into three parts: agent-centred, agent-environment centred, and agent-agent centred. Table 8 depicts our categorisation.

Table 8: Agent properties are classified into three: agent, agent-environment, and agent-agent centred

Agent	Agent-Environment	Agent-Agent
Autonomy	***Reactive***	***Social abilities***
Flexible		Team work
Personisability		Co-operation
Mobile or static		Communicative
Pro-active (planning)		
Learning		
Veracity, benevolence, rationality		
Emotion		
Belief, desire, and intention		

Among most researchers, there is consent that an agent should be defined as an autonomous, reactive, pro-active, and social system (Wooldridge and Jennings, 1994). From Table 8, the interpretation is as follows: researchers seems to be picking up or abstracting *autonomy*, and *pro-active* from the agent-centred part, *reactive* from agent-environment, and *social abilities* from agent-agent group to discuss the notion of an agent.

However, we believe that reactive, and social ability can all be abstracted further in the sense that reactive and social ability are part of agent-environment and agent-agent interaction categories respectively, and these simply describe how an agent behaves externally – i.e., external aspects of an agent.

The notion of autonomy of an agent is about having a degree of control of its actions and their internal state, and the notion of planning is about introducing actions to obtain a goal, which can be viewed as internal aspects of an agent. In chapter 3, we will discuss how the concept of a constraint can be used to discuss the reactivity, pro-activity, and social abilities of an agent.

In this section, the different types of agents are listed that can be identified by combining some of the attributes described above:

Autonomous agents: Agents that inhabit some complex, dynamic environments, sense and act autonomously in this environment and by doing so realise a set of goals or tasks.

Entertainment agents: Interactive, simulated worlds providing entertainment to a user. These agents are for entertainment purposes (e.g. games, film/video production), rather than strictly utilitarian ones.

Information agents: Agents that have access to many potential information sources and are able to collate and manipulate information obtained from these sources to answer queries posed by users and/or agents. Some people refer to these as Internet agents as such agents may roam about the Internet in order to collect information.

Intelligent agents: Agents that carry out some of the operations on behalf of a user or another program with some degree of independence.

Interface agents: Pattie Maes, a key researcher in this type of agents, states that the key metaphor underlying interface agents is that of a personal assistant who is collaborating with the user in the same environment (Maes 1994). Interface agents learn typically to better assist in four different ways:

- By observing and imitating the user;
- Through receiving positive and negative feedback from the user;
- By receiving explicit instructions from the user; and
- By asking other agents for advice.

Collaborative agents: These agents emphasise autonomy and co-operation in order to perform tasks for their owners. Their key attributes include autonomy, social ability, responsiveness, and pro-activeness. In order to have a co-ordinate set-up of collaborative agents they may have to negotiate to reach mutually acceptable agreements (Jennings 1995, Nwana 1996).

Mobile agents: Mobile agents are computational software processes capable of roaming wide-area networks, such as the World Wide Web (WWW), interacting with foreign hosts, gathering information on behalf of its owner and coming back after having performed the duties set by its user (Nwana 1996). The attribute of mobility has introduced the concept of remote programming (White 1994) where agents interact as peers and each agent can act as both a client and server. Some of the most important issues to be dealt with while implementing mobile agents are that of security, secrecy, transport mechanism, and authentication. The system has to be protected against such hazards as viruses and endless loops that consume all the CPU cycles. Some notable mobile agent implementation platforms are Agent Tcl from Dartmouth University (Agent Tcl 1995), Telescript (Telescript 1996), Odyssey (Odyssey 1997) from General Magic Inc., IBM Aglets Workbench (IBMAglets 1996), Voyager (Voyager 1997) from Objectspace Inc., and Concordia from Mitsubishi Electric (Concordia 1997).

2.4.2 Agent architectures

Muller et al. (1993) have provided several reasons for considering agent architectures for applications. Firstly, the architecture provides a valuable general guideline for the methodology of the design and implementation of an application. Secondly, the modules of the agent architecture precisely structure the classes of operational knowledge. Thirdly, agent architecture provides a basis for the investigation of special strategies and extensions of the modules; finally, predefined mechanisms such as negotiation protocols (e.g. the contract net) are directly applicable.

Agent architectures may be classified into three categories: deliberative, reactive, and hybrid.

2.4.2.1 Deliberative agents

A deliberate architecture is an architecture that relies on explicit, internally held symbolic models and symbolic manipulations. The agents in the deliberate architecture may be seen as deliberate or planning agents. Within the planning approach, given a goal to an agent, an agent will introduce a series of actions to achieve a goal.

The success of these architectures depends on two assumptions: Firstly, an agent should have complete up-to-date knowledge about the state of the world. Secondly, the effect of agent action is always known in advance, and would always be correct. The well-known deliberate architecture reported in the literature are IRMA (Bratman et al. 1988), and GRATE (Jennings 1992).

2.4.2.2 Reactive agents

Reactive agents represent a special category of agents which do not process the internal, symbolic model of their environments; instead they act/respond in a stimulus-response manner to the present state of the environment in which they are embedded. Figure 1 shows the behaviour of reactive agents.

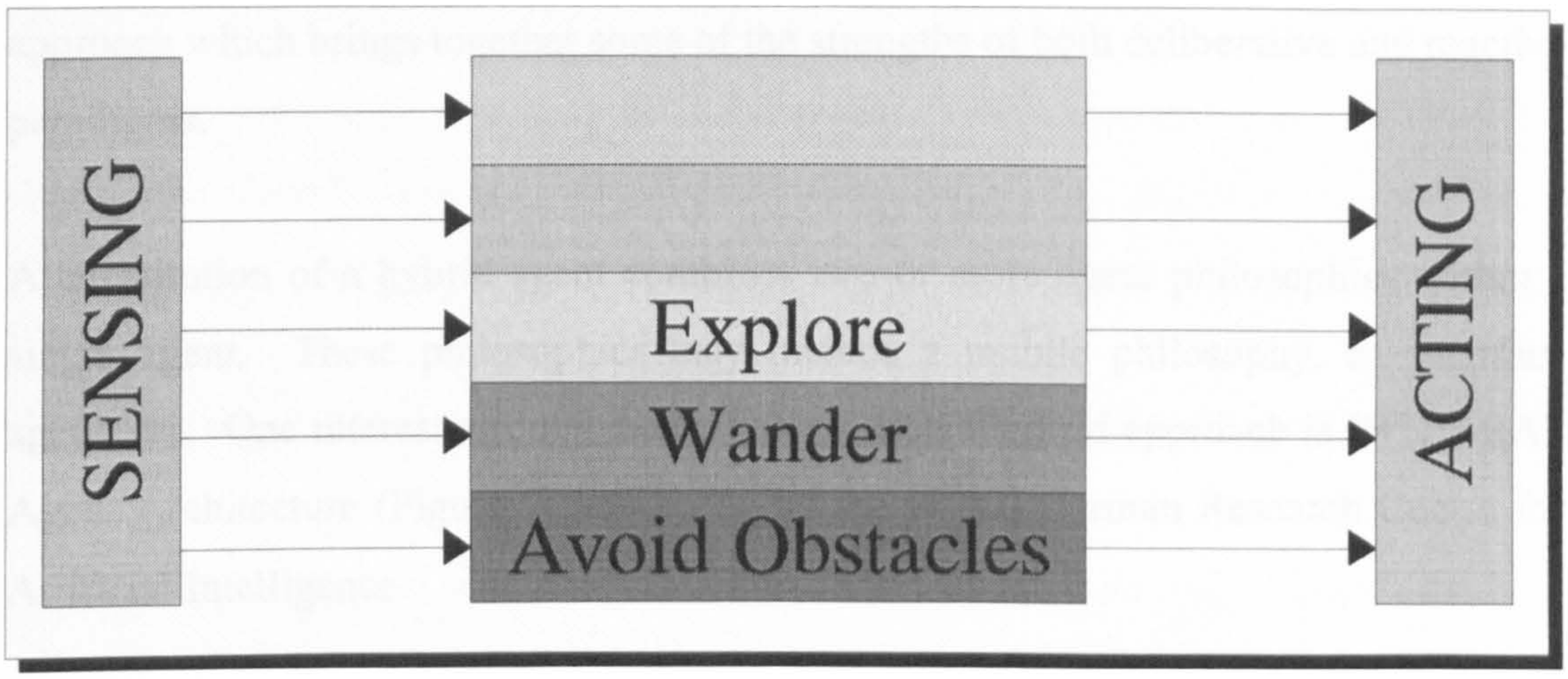

Figure 1: Reactive architecture: Methodology

Table 9 depicts various reactive architectures.

Table 9: Reactive architectures

System	Application
Pengi (Agre and Chapman (1987)	A video game
Wavish and Graham (1995)	CD-i computer game characters Digital video and 3-D graphics-based animations
Ferber (1996)	Simulate ant societies Simulated a limited ecosystem composed of biotapes, shoals of fish and fisherman

The main criticisms of reactive agents are as follows:

- The scope of applicability is currently limited to mainly games and simulations
- How are such systems extended, scaled up or debugged?
- What happens if the 'environment' is changed?
- Not obvious how to design such systems so that the intended behaviour emerges.

Because of their lack of explicit goals and goal-handling capabilities, the designers of reactive systems need to pre-compile or hard-wire the action selections; while a deliberative agent's approach leaves much to the agent, the reactive agent's approach leaves much to the designers.

2.4.2.3 Hybrid agents

Since each of the above mentioned approaches has its own strengths and limitations, it often becomes necessary to maximise the strengths and minimise the limitations of the most relevant technique for a particular purpose. The aim is to adopt a hybrid

approach which brings together some of the strengths of both deliberative and reactive paradigms.

A constitution of a hybrid agent combines two or more agent philosophies within a single agent. These philosophies may include a mobile philosophy, an interface agent, etc. One interesting implementation of such a hybrid approach is INTERRAP Agent Architecture (Figure 2) developed by the DFKI, German Research Centre for Artificial Intelligence.

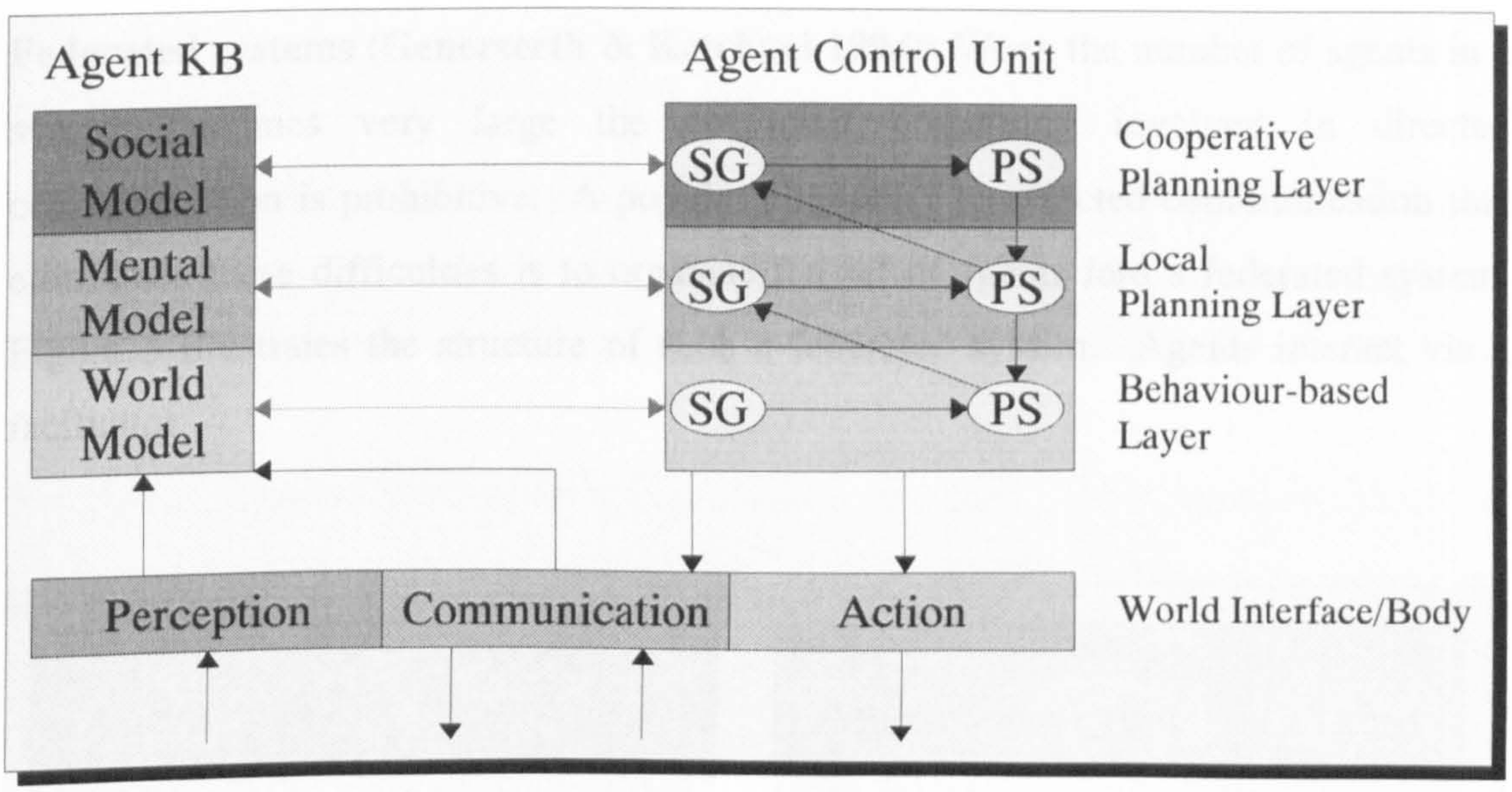

Figure 2: INTERRAP architecture

The main criticisms of hybrid agents are:

- Hybridism usually translates to ad hoc or unprincipled designs;
- Many hybrid architectures tend to be very application specific;
- Theory, which underpins hybrid systems, is not usually specified.

Communication enables the agents in a multi-agent system to exchange information, on the basis of which they co-ordinate their actions and co-operate with each other. The main questions that arise are which communication protocols and mechanisms are conductive to enhance collaboration between communicating agents. Within a multi-agent system, several ways have been proposed for agents to exchange information with each other. Agents can directly exchange messages, or they can organise themselves into a federated system and communicate through special

facilitator agents (Genserserth & Ketchpel 1994), or they can broadcast the messages. Another popular approach used to enable agents to intercommunicate is through a shared blackboard on which information can be posted and retrieved (Chaib-draa et al. 1996).

Directed communication: Directed communication involves establishing direct physical links with other agents using a protocol such as TCP/IP which promises safe arrival of message packets by implementing end-to-end acknowledgements.

Federated systems (Genserserth & Ketchpel 1994): When the number of agents in a system becomes very large the cost and processing involved in directed communication is prohibitive. A popular alternative to directed communication that eliminates these difficulties is to organise the set of agents into a federated system. Figure 3 illustrates the structure of such a federated system. Agents interact via a facilitator.

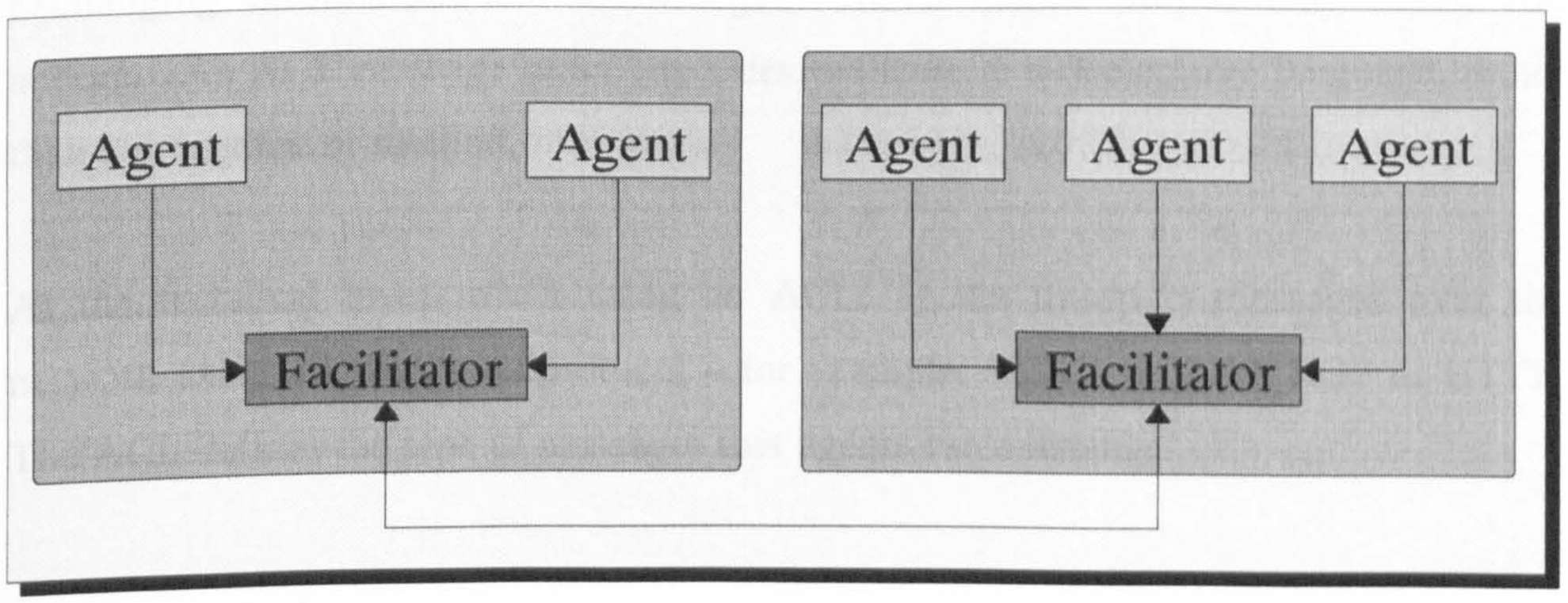

Figure 3: Federated system

Broadcast communication: In scenarios where a message has to be communicated to all the agents in the environment, or the sender agent does not know who the recipient will be, then it can physically broadcast the message to all the agents in the system. Alternatively, it can maintain individual communication links with all the agents in the system and send each one of them a directed message. Two main popular approaches in broadcast communication are the contract net and the specification-sharing approach. Contract net approach is elaborated in Chapter 4. In the

specification sharing approach, agents broadcast their capabilities and needs and other agents use this information to co-ordinate their needs and actions.

Blackboard systems: In AI, the blackboard is an often-used model of shared memory (Chaib-draa & Moulin 1987). It is a store on which agents write messages, post partial results, and obtain information. It is usually partitioned into several levels of abstraction appropriate for the problem at hand, and agents working at a particular level of abstraction have access to the corresponding blackboard level along with the adjacent levels.

2.4.3 Agent communication languages

For interoperability, agents should be able to communicate with agents supplied by different vendors or implementors. The obvious solution is a lingua franca, whereby all the agents who implement the same lingua franca can communicate.

An Agent Communication Language (ACL) provides agents with a means of exchanging information and knowledge. ACLs handle propositions, rules, and actions. An ACL message describes a desired state in a declarative language, rather than a procedure or method.

At the technical level, when using an ACL, agents transport messages over the network using a lower-level protocol – for example, SMTP, TCP/IP, IIOP or HTTP. The ACL defines the type of messages that agents can exchange.

2.4.3.1 KQML

The Knowledge Query and Manipulation Language (KQML) (Finin et al. 1994) was defined under the DARPA-sponsored Knowledge Sharing Effort. KQML assumes a layered architecture.

KQML is a high-level, message-oriented communication language and protocol for information exchange independent of content syntax and applicable ontology. Thus KQML is independent of the transport mechanism (TCP/IP, SMTP, IIOP, or another),

independent of the content language (KIF, SQL, STEP, Prolog, or another), and independent of the ontology assumed by the content.

Conceptually, one can identify three layers in a KQML message: content, communication, and message. The content layer bears the actual content of the message in the program's representation language. The communication layer encodes a set of features to the message that describe the lower-level communication parameters, such as the identity of the recipient and sender, and a unique identifier associated with the communication. The message layer, which encodes a message that one application wants to transmit to another, is the core of KQML.

2.4.3.2 Arcol and The Foundation for Intelligent Physical Agents(FIPA)

The Foundation for Intelligent Physical Agents is a nonprofit association whose purpose is to promote the success of emerging agent-based applications, services, and equipment.

FIPA's agent communication language (like KQML) is based on speech act theory: messages are actions or communicative acts, as they are intended to perform some action by virtue of being sent.

Arcol is another ACL based on speech acts (Breiter and Sadek, 1996). Arcol was the basis for the first version of the proposed FIPA standard, and many of its components survive in the second version as well. Agents conforming to the FIPA specification can deal explicitly with actions. They make requests, and they can nest the speech acts. The FIPA specification has a formal semantics.

2.4.3.3 Comparison

KQML suffers from as yet poorly defined semantics. As a result, of the many implementations of KQML, each seems unique. This makes communication difficult, and the KQML agent might not be understood. Security has not been a major issue in the KQML work.

The FIPA specification, by contrast, attempts to formalise the semantics and provide a security model. However, in view of its recency, it has not been widely accepted or adopted.

2.4.4 Overview of commercial and research products, applications, benefits, and weaknesses

2.4.4.1 Agent-based commercial products

Various agent-based commercial products are presented in Table 10. The products presented vary from development environment to multi-agent protocols. Many of them are implemented on object-oriented languages such as Java, and C++.

Table 10: Agent-based commercial products

Product	Description	Application	Language	Company
AgentBuilder	Integrated Agent and Agency Development Environment	General applications	Java	Reticular Systems, Inc.
AgenTalk	Multi-agent coordination Protocols	General application	LISP	NTT/Ishida
Agent Building environment	Agent Development Environment	General application	C++, Java	IBM
Agent Development Environment (ADE)	Agent Development Environment	General application		Gensym
Agentx	Agent Development Environment	General application	Java	International Knowledge Systems
IGEN tm	Cognitive Agent Toolkit	Intelligent applications	C/C++	CHI Systems
Intelligent Agent Library	Agent library	General applications	Java	Bits & Pixels
JACK Intelligent Agents	Agent Development Environment	Distributed applications	JACK Agent Language	Agent Oriented Software Pty. Ltd.
JAM	Agent architecture	General application	Java	Intelligent Reasoning Systems
Kafka	Agent library	Distributed programming	Java	Fujitsu
Versatile Intelligent Agent VIA	Agent Building Blocks	Web sites and intranets	Java	Kinetoscope
Voyager	Agent-Enhanced ORB	Distributed applications	Java	Object Space
Zeus	Agent architecture	General application	Java	BT

2.4.4.2 Research products

The concept of an agent is being researched by many academic institutions. Many systems have been developed. The trend in agent-based products is to use an agent communication language such as KQML. In general, the object-oriented language Java is used for research in many academic institutions. However how to address constraints is not addressed. This thesis addresses such limitations. Table 11 lists various agent-based research products.

Table 11: Agent-based research products

Systems	Description	Concepts	Language	Research Organisation
Bond Distributed Object System	Agent Framework	Provides a message-oriented environment Uses the KQML for object communication	Java	Purdue University
Cable	System Architecture	Provides ADL (Agent Definition Language)	Agent Definition Language, C++	Logica Corporation
JAFMAS	Multiagent Framework	Uses KQML Analysis system coherency	Java	University of Cincinnati
JATLite	Java packages for Multiagents	Provides a basic infrastructure in which agents interact	Java	Stanford University
Knowbot System Software	Mobile agents	Research infrastructure for mobile agents	Python	CNRI
LALO	Programming Environment	Framework for developing mult-agent systems.	LALO	CRIM

2.4.4.4 Agent applications, strengths, and weaknesses

There are wide ranges of application domain that benefit from the use of agent-based systems. Agent applications are developed for fields as varied as manufacturing, entertainment and electronic commerce. This section describes these application types.

User assistant applications: These systems are those that work with, and in the interests of, an end-user in order to enhance their productivity and to ease the use of complex computer-based systems. More commonly, they communicate with users to help with managing diaries and emails, memory assistance, etc. (Mitchell et al. 1994). They may communicate with other agents (e.g. media agents) for information gathering. They are different from standard user interfaces, in that they are empowered to act at least semi-autonomously, and are not merely tools that the user uses and controls. Some common user applications are:

- User profile learning systems (Caglayan et al. 1996);
- Multimodal interface systems; and
- Personal Digital Assistant or Personal Intelligent Communicator applications (e.g. digital telephone secretary).

Information retrieval applications: These systems involve all the services needed to help users easily and quickly find the information they request (Huhns et al. 1994, Sheth et al. 1993). This can be achieved, for example, by a society of agents. They include:

- Directory services (yellow and white pages);
- Date base inquiry;
- Information brokerage; and
- Media indexing.

Entertainment applications: These are systems, which involve:

- Real-time and non-real-time (store and forward) user avatars for messaging, low-bit-rate communication, and shared virtual environments;
- Games (autonomous interaction between game characters and with environment and multi-player games);
- Gaming and avatar applications deployed in theme parks, arcades, kiosks, WWW, and high-end game machines (Nwana 1993); and
- Film/Video production (i) Camera agents (film/video cameras with automatic motion, focus, reactions, etc.), (ii) 3D graphical agents for storyboard design, (iii) 3D graphical agents and avatars in computer animated feature films, cartoons and advertisements.

Service management applications: These are systems that involve configuration and delivery of user requested services at the right time and cost, while observing required security and privacy issues. Some common service management applications are:

- Multimedia services;
- Buying/selling services (e.g. information, material, goods) (Chavez, A & Maes, P 1996);
- TMN/Intelligent Network Management Services; and
- Trip planning and guidance services (e.g. intermodal route planning, hotel and parking lot reservations, individual traffic guidance, tourism).

Business management applications: These systems deal with the management of business tasks and resources in the provision of services and carrying out of business operations (O'Brien & Wiegand 1996). They include:

- Financial services;
- Electronic commerce (White 1994);
- Workflow management (Levitt et al. 1994);
- Office automation;
- Computer Supportive Co-operative Work; and
- Telecommuting (Appleby & Steward 1994).

Manufacturing management applications: These systems involve physically embodied agents designed to carry out and deal with the management of tasks and processes in relatively structured industrial environments. These processes may involve the control of industrial robots and machines via software interfaces. Some common manufacturing applications areas are:

- Industrial robotics (Brooks 1986);
- Factory automation (Baker 1996);
- Virtual factory management; and
- Load balancing.

Service robotics applications: These systems involve physically embodied agents designed to carry out tasks and processes in relatively unstructured office and domestic environments (e.g. office mail delivery, house cleaning etc.).

Co-operative task management applications: These systems involve the collection of robotics and software agents that are being co-ordinated to achieve higher level tasks.

Research applications: These systems involve using agent technology to further research in other (IT) areas such as:

- Vision processing;
- Learning and adaptive systems (Hermans & Schlimmer 1993);
- Speech processing;
- Distributed knowledge-based systems; and
- Human-computer interface.

In summary, various applications are being considered for agent-based application. The proposed approach that will be elaborated in Chapter 4 is relevant for such applications. The advantages of multi-agent systems are as follows:

Fault-Tolerance (Hatvany 1984): Agents are an inherently distributed mechanism and thus a system made of autonomous agents will not collapse when one or more of its components fail as there will not be any single point of failure.

Modular Software/Scaleable architecture (Parunak 1996): Agents are powerful entities because of the factorisation of the problem-solving they provide. Each agent can be identified as an entity (e.g. a machine, a tool, or a part) and thus help in incremental growth and flexible expansion. The advantage of scalability is provided as each agent can join a system, start working with other agents, or just leave a system once it has finished a plan it was engaged in without effecting the operation of the system.

Self-configuration systems (Parunak 1996): A population of agents can reconfigure itself as it runs. This is an important advantage for systems that must respond to a wide range of different conditions. This is because, as each agent is close to the point of contact with the real world, the system's computational state tracks the state of the world closely, without need for a centralised database. As the overall system behaviour emerges from local decisions, the system readjusts itself automatically to the environment, or the noise, or the removal of other agents. Thus a fully functional self-configuring system can be effectively implemented by merely networking agent resources.

Reduced software costs: As the software becomes more modular, the development time and the complexity is reduced.

Faster problem solving: By exploiting parallelism in the sense that different agents work autonomously to achieve a common vision, problems are solved quicker.

Decreased communication: By transmitting only high level partial solutions to other agents, communication is decreased.

Flexible systems: Flexibility is provided by having agents with different abilities dynamically team up to solve current problems.

This review has acknowledged disagreements between researchers on the definition of an agent, and on the interaction of agents without developing a hypothesis. We have established that agents and agent interactions were an interesting part of the parent field DAI to research, and we have summarised in tables comparisons between compared the current definition of an agent, and agent interactions.

In the preceding section, we have analysed the definition of an agent, agent communication languages, and interaction strategies. It is clear that there are discrepancies in the definition of an agent. There are few agent communication languages such as KQML and we have discussed their limitations. The existing agent architectures are quite rigid in the sense that the behaviour of an agent, and the

interaction between agents, tend to be hard coded. Shoham's AOP has its limitations as the approach is not discussed within multi-agent settings.

Particular concepts and the hypothesised directions of relationships between agent behaviour and interactions will be summarised in a detailed analytical model which grew out of the earlier classification model to structure the literature review.

An emphasis is put on constraint-based systems that is discussed in the next chapter; it is believed that constraint-based systems are relevant to the discussion of behavioural and interaction aspects of agents.

2.5 Conclusion

In this chapter, the definition of an agent was reviewed. Based on the literature review, we have unearthed areas which require researching. With the idea of diversification of agents in the literature, we suggested that there is a need to address interaction strategies. We have asked the question: How do agents interact with the environment, with a user, and between each other with consideration to constraints? Agent architectures have been presented. Agents are seen to be autonomous, reactive, pro-active social systems.

We have consolidated the following testable hypothesis that "the behaviour of an agent, and the interaction between agents can be related to constraint-based techniques" for the research questions. We investigate further constraint-based techniques in the next chapter, and we propose a CANET methodology for complex applications.

3. Proposed modification to existing systems - Constraint logic

"Constraint (n.) The state of being checked, restricted, or compiled to avoid or perform some action". (Webster's Ninth New Collegiate Dictionary).

Based on the literature review in chapter 2, and the research problem outlined in chapter 1.2, the suitability of constraints for agent interactions is now investigated. In this chapter, an overview of a constraint-based system is provided. Basic concepts of constraint-based techniques are given. The aim is to synthesise these techniques with that of agents in the next chapter. The discussion on evolution of an agent-based system allows our work to be located in the research map.

We propose a CANET methodology for complex applications towards the end of this chapter. Within the CANET methodology, the behaviour of an agent, and the interaction between agents can be related to constraint technology. We will describe many issues concerning multi-agent systems (i.e. co-ordination of actions, task allocation, and co-operation) from a constraint-based view. We also compare related methodologies.

3.1 Motivation

Constraints arise in most areas of human endeavour. Constraints formalise the dependencies in physical worlds and their mathematical abstractions transparently, naturally, and implicitly.

A constraint is simply a logical relation among several unknowns, each taking a value in a given domain. The constraint thus restricts the possible values that variables can take; it represents partial information about the variables of interest.

Constraints can also be heterogeneous, so they can bind unknowns from different domains, for example the length (number) with the word (string). The important feature of constraints is their declarative manner, i.e. they specify what relationship must hold without specifying a computational procedure to enforce the relationship.

Constraints are used to guide reasoning as a key part of everyday common sense. For example, a constraint one can use to plan time is as follows: "I can be there from five to six o'clock for a project meeting". Generally, one does not deal with just one constraint, but a collection of constraints that are rarely independent.

Constraint programming is the study of computational systems based on constraints. The idea of constraint programming is to solve problems by stating constraints (requirements) about the problem area and, consequently, finding solutions that satisfy all constraints.

Many problems can be viewed naturally as constraint-satisfaction problems (CSPs) (Freuder and Mackworth 1994, Tsang 1993). In such problems, we seek to find values for problem variables that satisfy or optimise restrictions on value combinations. Applications are found in many fields of AI, including planning, design, diagnosis, temporal reasoning, vision, and language.

3.2 Terminology of constraint-based systems

A constraint-based system is a paradigm for formulating knowledge as a set of constraints without specifying the method by which these constraints are to be satisfied. A variety of techniques have been developed for finding partial or complete solutions for different kinds of constraint expressions. Applications of constraint-based systems include design, diagnosis, truth maintenance, scheduling, logic programming, and user interface.

A constraint network consists of a finite set of variables $X = \{X_1, ..., X_n\}$, each associated with a domain of discrete values, $\{D_1, ..., D_n\}$ and a set of constraints, $\{C_1, ..., C_t\}$. Each of the constraints is expressed as a relation, defined on some subset of variables, whose tuples are all the simultaneous value assignments to the members of this variable subset that, as far as this constraint alone is concerned, are legal.

Constraint Satisfaction Problems have been a subject of research in Artificial Intelligence for many years. A Constraint Satisfaction Problem (CSP) is defined as:

- a set of variables $X = \{x_1, \ldots, x_n\}$,
- for each variable x_i, a finite set D_i of possible values (its domain),
- a set of constraints restricting the values that the variables can simultaneously take.

A solution to a CSP is an assignment of a value from its domain to every variable, in such a way that all constraints are satisfied at once. One may want to find:

- just one solution, with no preferences as to which one,
- all solutions,
- an optimal, or at least a good solution, given some objective function in terms of some or all the variables.

A constraint can be a relation between variables. Examples include $x < y + 3$, $f(x, y)$, and so on. Constraint systems are obtained by sharing variables among constraints. An example is $x \varepsilon \{1, \ldots, 10\}$, $y \varepsilon \{1, \ldots, 10\}$ and $x = y + 1$.

When a set of constraints is used to characterise the solution of a given problem, that problem is set to be *overconstrained* when there is no solution that obeys the set of constraints. Because constraints are defined as relationships or connections between variables, a constraint satisfaction problem is often viewed as a graph or constraint network.

Constraints are multi-directional. Consider the constraints: $X + Y = 5$; and $X = 2$. Constraint reasoning method solves a value for Y. The operation differs from traditional languages in the sense that within the traditional language, the left-hand side variable of the equation is always evaluated, but within the constraint-based system, any constraints can be solved.

The technique of extending or communicating constraints so that they define further relationships between variables is *constraint propagation*. Table 12 shows the effect of constraint propagation.

The results of successive constraints show the effect of the imposition of these constraints on the system X, Y, and Z:

Table 12: The table shows the effect of constraint propagation on three variables X, Y, Z within a set of constraints, indicated by the attachment of subscripts, max and min, that range between 1 to 100. Adding new constraints to the system causes the effects on values of the system shown below.

Constraint propagation	X		Y		Z	
	Xmin	Xmax	Ymin	Ymax	Zmin	Zmax
Initial constraints	1	100	1	100	1	100
Adding constraint #1: 2*X +Y =< Z	1	49	1	98	3	100
Adding constraint #2: Z < 5	1	1	1	2	3	4
Adding constraint #3: X \= Y	1		2		4	

Two branches of Constraint Programming (CP), namely constraint satisfaction and constraint solving, share the same terminology but the origins and application areas are different. Constraint satisfaction deals with problems defined over finite domains and, currently, probably more than 95% of all industrial constraint applications use finite domains. Constraint solving shares the basis of CP, i.e., describing the problem as a set of constraints, and solving these constraints. But, within the constraint solving, the constraints are defined over infinite or more complex domains. Instead of combinatorial methods for constraint satisfaction, the constraint solving algorithms are based on mathematical techniques such as automatic differentiation, Taylor series, or Newton method.

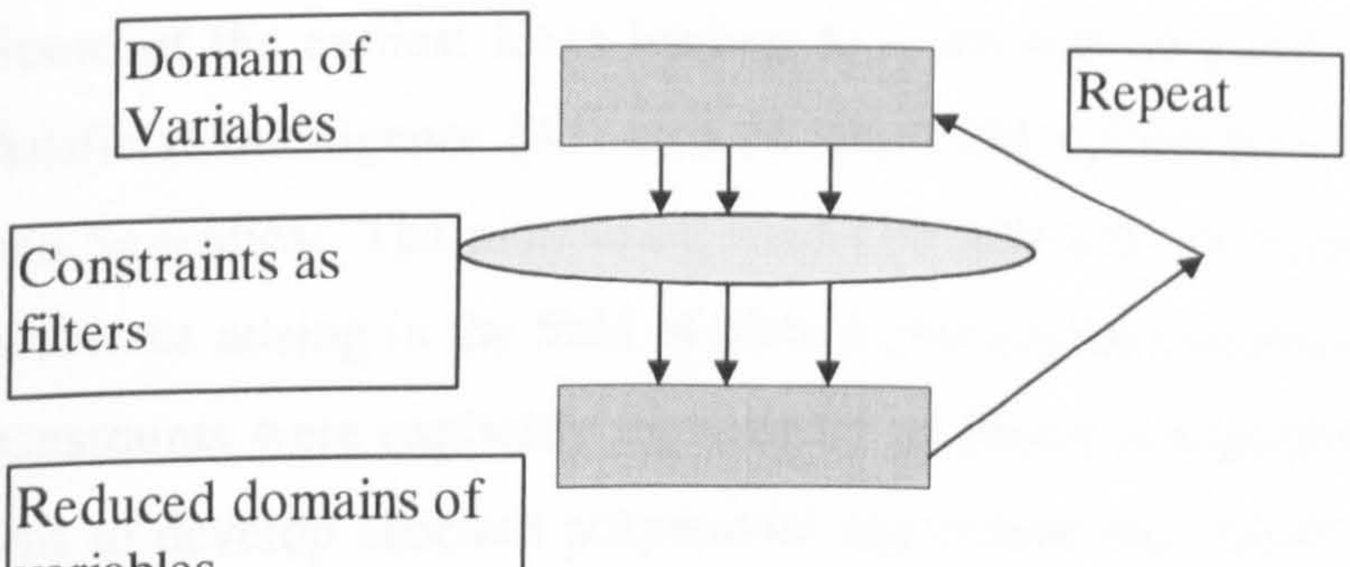

Figure 4: Constraint solving

A basic constraint takes the form x=n, x=y or x ε D, where x and y are variables, n is a non-negative integer and D is a finite domain.

Propagation of constraints is depicted in figure 4. Constraints can be viewed as filters of the domain of variables. The reduced domain can then be passed through the constraints in a repetitive manner.

Propagation constraints are commonly called constraint agents. The behaviour of a constraint agent is to propagate information to the underlying store. In case the underlying store is a constraint store, the information propagated is expressed as primitive constraints. Constraint agents can be built by directly defining their waking behaviour using the notion of a 'guard'. The term 'actor' is used within the Oz community to discuss the propagation constraints.

A new class of constraints is called reified constraints. Reified constraints make it possible to express constraints involving logical connectives such as disjunction, implication, and negation. Reified constraints also make it possible to solve overconstrained problems, for which only some of the stated constraints can be satisfied.

The reification of a constraint C with respect to a variable x is the constraint:
$(C \leftrightarrow x=1) \wedge x \varepsilon 0\#1$ where it is assumed that x does not occur freely in C.

3.3 Evolution of constraint-based systems

Some of the earliest ideas leading to constraint programming may be found in the Artificial Intelligence (AI) area of constraint satisfaction, dating back to the Sixties and Seventies. The pioneering works on networks of constraints were motivated by problems arising in the field of picture processing (Montanari 1974). In these works, constraints were explicitly represented as binary compatibility matrices and the goal was to develop efficient polynomial algorithms that could discover incompatibilities by looking at just a few constraints.

Waltz (1975) dealt with the scene labelling problem. The goal is to recognise the objects in a 3D scene by interpreting lines in the 2D drawings. The types such as

convex (+), concave (-), and occluding edges (<) are used for labelling. The main algorithms developed were related to achieving some form of consistency.

Another application for constraints is interactive graphics where Ivan Sutherland's Sketchpad (Sutherland 1963), developed in the early 1960s was the pioneering system. ThingLab (Borning 1981) were interactive graphics applications that allowed the user to draw and manipulate constrained geometric figures on the computer's display. Table 13 summarises the discussion on the evolution of constraint-based systems.

Table 13: Evolution of constraint-based systems

System/Field	Researchers	Aim/Characteristics
'network of constraints' Scene labelling	Montanari (1974) Waltz (1975)	Constraints were explicitly represented as binary compatibility matrices; the goal was to develop efficient polynomial algorithms for discovering incompatibilities by examining constraints
Interactive graphics Circuit modelling and diagnosis e.g. Sketchpad ThingLab	Sutherland (1963) Borning (1981)	Constraint as a declarative relation
CONSTRAINT LOGIC- PROGRAMMING CLP, CLP® Prolog III CHIP ECLIPSE	Jaffar et al. (1987, 1992) Colmeraur (1990)	Algorithm = Logic + Control Logic programming as a kind of constraint programming
Concurrent constraint programming Oz AKL CIAO	Saraswat (1993) Smolka (1995) Hermenengildo (1994)	Saraswat (1992, 93, and 95) and Gupta (1997) discusses abduction, concurrent logic, default CC, Timed CC for Concurrent constraint languages

Constraint programming derives from logic programming, operational research, and artificial intelligence. Logic programming offers the general non-deterministic host language which accommodates dedicated constraint solvers from Operational Research (OR) and AI such as linear programming or constraint satisfaction techniques.

There are two main directions of approach to uncovering the mathematical content of logic – model theory and proof theory (c.f. Figure 5). Model theory examines the relationships between sentences of logic once associated with external domains, such

as truth-values. The vocabulary of elementary model theory employs such terms as *true*, *false*, *interpretation*, *satisfaction*, *model*, *implication*, and *semantic consequence*.

Proof theory examines the relationships between sentences in terms of their derivability from other sentences using rules, which operate only upon the structural content of sentences. The vocabulary of elementary proof theory uses terms such as *axiom, inference, rule, theorem, proof, consistency*, and *syntactic consequence*. Both approaches are of value in understanding logic programming.

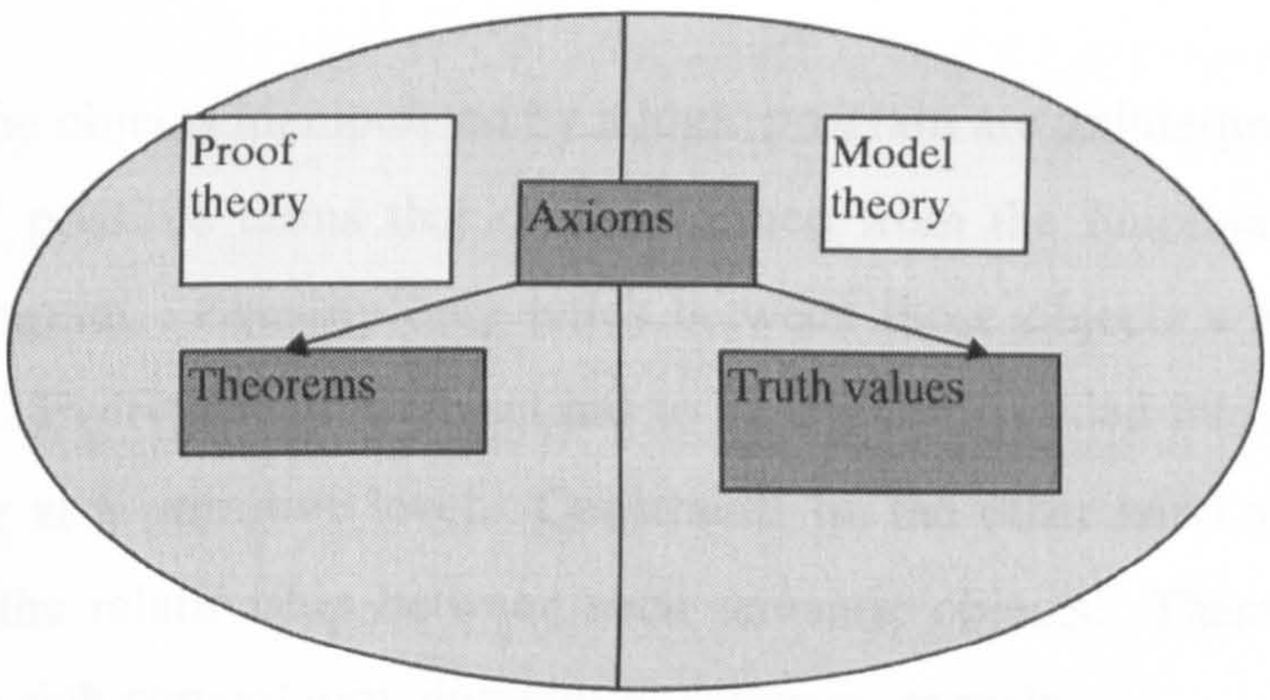

Figure 5: The two views of logic

Logic programming is a computational formalism, which combines these two central principles:

It uses logic to express knowledge. It uses inference to manipulate knowledge.

The logic programming formalism adds a particular sort of control strategy in the pursuit of efficient implementation to the kernel system (Figure 6)

clausal-form logic + resolution

What does a logic program look like? — Simply a set of clauses describing relations, as does the following example:

```
likes(indran, Anyone) if reads(Anyone, this_thesis)
reads(Anyone, this_thesis) if sensible(Anyone)
sensible(you)
```

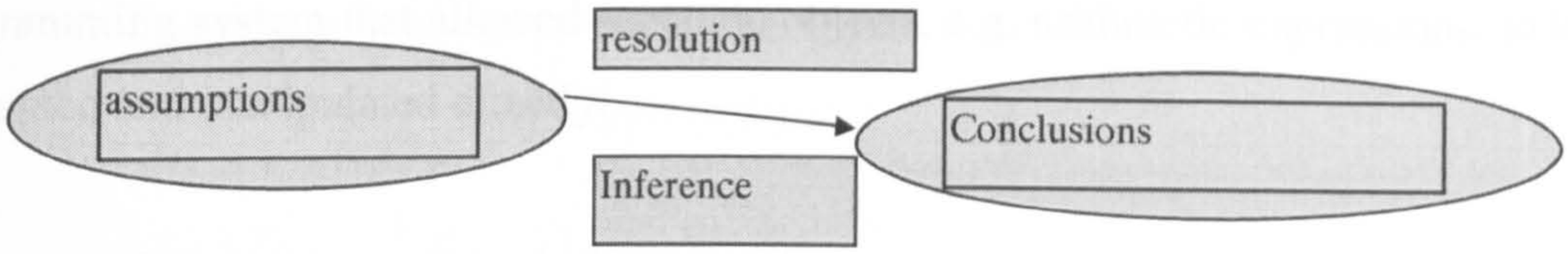

Figure 6: The essence of logic programming

Logic programming has the unique property that its semantics (operational and declarative) are both simple and elegant and coincide in a natural way. These semantics, however, have their limitations.

Firstly, the objects manipulated by a logic program are uninterpreted structures — the set of all possible terms that can be formed from the functions and constants in a given program. Equality only holds between those objects which are syntactically identical. Every semantic object has to be explicitly coded into a term; this enforces reasoning at a primitive level. Constraints on the other hand are used to implicitly describe the relationship between such semantic objects. These objects often range over such rich computation domains as integers, or reals.

Secondly, logic programming stems from its uniform but simple computation rule, a depth first search procedure, resulting in a generate-and-test procedure with its well-known performance problems for large-scale applications.

Constraint manipulation and propagation were studied in the Artificial Intelligence community in the early 1970s and 1980s (Montanari 1974, Steele 1980, Mackworth 1986) to make search procedures more intelligent.

Constraint logic programming (CLP) is an attempt to overcome the limitations of logic programming by enhancing it with constraint solving mechanisms. Strangely, both of these limitations of logic programming can be lifted using "constraints". However, each limitation is treated by a quite different notion of constraint. Hence CLP has two complementary lines of descent:

First it descended from work that aimed at introducing richer data structures to a logic programming system that allowed semantic objects, e.g. arithmetic expressions, to be expressed and manipulated directly.

Secondly, CLP has been strongly influenced by the work on consistency techniques. With the objective of improving the search behaviour of a logic programming system, Gallaire (1985) advocated the use of these techniques in logic programming. "Constraint and test" replaced the performance problems of the "generate and test" method.

From a theoretical point of view the extension of logic programming to constraint logic programming has been very useful. That, in turn, inspired development of concurrent constraint languages (Saraswat 1993, Smolka 1995).

Van Hentenryck et al. (1996) discussed promising directions in constraint programming. They are listed below:

More realistic constraint systems and languages: There is a necessity to develop more automatic and systematic ways to acquire and model domain-specific and problem-specific knowledge, developing a richer paradigm to cope with the properties and uncertainties of real-world information.

Towards constraint-based distributed systems: Another challenge for constraint programming systems is related to the role of such systems in network-wide programming. Our research can be categorised within this direction in the sense that we have explored how agents and constraints can be synthesised.

Towards faster, more efficient systems: While the performance and computing resource economy of current CP systems has proved to be adequate in significant industrial applications, competing very favourably with other techniques and approaches, it appears that there still remain many avenues for improvement, which would make the technology even more competitive.

Constraint databases: Many challenges in constraint databases are yet to be addressed. Specific directions of work include constraint modelling, canonical forms and algebrae; data models and query languages.

User interfaces: In user interface applications, there is a constant need for constraint satisfaction algorithms that can handle a wider range of constraints that arise in such applications, and algorithms and data structures with improved time and space efficiency.

3.4 Justification for constraint-based systems, constraint-based applications, and methodology

3.4.1 Justification for constraint-based systems

Constraints and agents have a potential synergy. On the one hand, agent behaviour, for example, can be modelled as constraint satisfaction. Constraint computation provides a general problem-solving framework. On the other hand, agents can be used to accomplish constraint satisfaction to solve distributed scheduling problems where agents are self-directed problem-solving entities.

Software agents can benefit by using constraint computation to improve the efficiency of individual agent problem solving (Tambe 1996). They may assist in knowledge acquisition (Freuder and Wallace 1997); or model the difficult issues of negotiation, collaboration, and competition among agents with differing interests (Freuder and Eaton 1997; Liu and Sycara 1994).

Constraint-based reasoning systems can be enhanced by using software agents to improve performance by combining the expertise of multiple, heterogeneous problem solvers (Petrie, Jeon, and Custosky 1997; Anderoli at al. 1997); improve solution quality when the different interest of multiple agents is necessary (Freuder and Eaton 1997), and improve the performance of constraint-satisfaction methods by distributing the problem over multiple agents (Petrie, Jean, Custosky 1997).

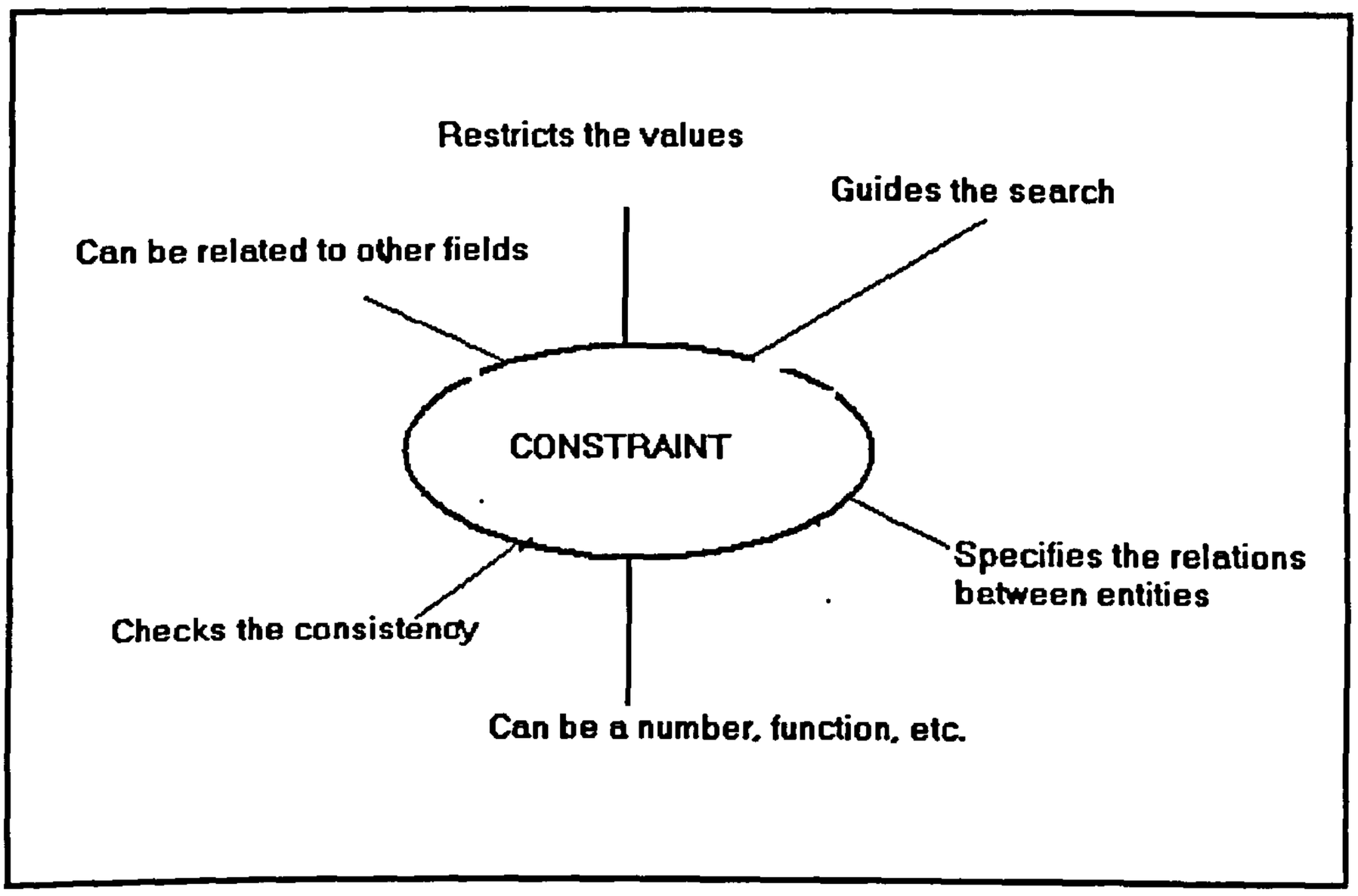

Figure 7: The relevance of constraints for applications

Figure 7 depicts the relevance of constraints for applications.

3.4.2 Constraint-based applications

Various applications using constraints for modelling are discussed below:

Constraints as relations: The simplest formal model of a constraint is a relation.

Circuit verification: The idea is that complex systems can be broken down and can be modelled in quite different ways — into functional components, physical components, causal sequences, etc. If the different models can be expressed in a common, constraint-based formalism, then one can learn much from the interaction of these different models. For the circuit verification, the different models — the behavioural model and the functional model — should be equivalent.

Real-time control systems: Constraint programming is now being exploited for building control software for eletro-mechanical systems with a finite number of inputs, outputs, and internal states. Each component within a complex system is only connected to a small part of the overall state of the system, and its behaviour can be

captured quite simply, but when the system is considered as a whole the number of global states becomes very large.

Constrained objects: A typical constraint applicable to an object is to constrain a property of the object. For example, a particular parking bay will not admit vehicles of more than a certain dimension.

Constraint programming has been successfully applied to many different problem areas as diverse as DNA structure analysis, time tabling for hospitals or industry scheduling. CP proves itself to be well adapted to solving real-life problems because many application domains evoke constraint description naturally.

Assignment problems were a type of industrial application that was solved using constraint tools. Examples include stand allocation for airports, where aircraft must be parked on the available stand during the stay at airports (Dincmas and Simonis 1991).

Personnel assignment problems are problems where work rules and regulations impose difficult constraints. Examples include the Gynnaste system (Chan et al. 1998), developed for the production of rosters for nurses in hospitals, for crew assignments to flights, or goods assignment in railway companies (Focacci et al. 1997).

Constraint-based software is used for well-activity scheduling (Johansen and Hasle, 1997), forest treatment scheduling (Adhikary et al, 1997), production scheduling in the plastics industry (InSol), or for planning production of military and business jets (Bellone et al, 1992).

Within the network management and configuration area, problems include planning the capability of the telecommunication networks in buildings or electric power network reconfiguration maintenance scheduling without disrupting customer services (Creemers et al. 1995).

Recent applications include:

- Computer graphics (expressing geometric coherence in the case of scene analysis, drawing programs, user interfaces);
- Natural language processing (construction of efficient parsers);
- Database systems (to ensure and/or restore consistency of the data);
- Molecular biology (DNA sequencing, hypothetical reasoning);
- Business applications (option trading);
- Electrical engineering (to locate faults); and
- Circuit design (to compute layouts).

3.4.3 A constraint store and constraint types

The traditional model of a computer store admits only two possible states for a variable: assigned or unassigned. Constraint programming uses an abstraction of this model in the sense that a so-called constraint store can hold partial information about a variable, expressed as constraints on the variable. Within such a model, an unassigned variable can be seen as an unconstrained variable whereas an assigned variable can be seen as maximally constrained in the sense that non-further non-redundant constraints can be imposed on the variable, without introducing an inconsistency.

Definition: A constraint store is a storage model, which admits primitive constraints of a specific class. Each new primitive constraint that is added to the store is automatically checked for consistency with the currently stored constraints.

The constraint store contains a constraint, and a named abstraction. Primitive constraints are the constraints that can be kept in the constraint store. Within the constraint store, constraints have the form variable = value. Constraints are formulae of first-order predicate logic with equality.

The constraint store contains a conjunction of basic constraints up to logical equivalence. An example for such a constraint is:

$$X \varepsilon 0\ \#5 \wedge Y = 8 \wedge Z \varepsilon 13\ \#23.$$

The storage model applied by logic programming has a weakness that can be shown by the following example. The equation x - 3 = y + 4 is not applicable because logic programming does not yield any meaning with - or + in such an equation.

Linear equations and inequations are examples of primitive constraints that are stored in the constraint store. Further constraint stores can be built for different classes of primitive constraints, by designing constraint solvers specifically for those classes of constraints. It is also possible to apply different constraint stores for different problem solving.

The constraint on the constraint store is always satisfiable. It is said that a constraint store entails a constraint Y if the implication O → Y is valid. The constraint store is consistent with a constraint Y if O and Y are satisfiable, where O is the constraint stored on the constraint store. Elaboration of a constraint O checks whether O is consistent with the constraint store.

A few constraints propagated to store by agents are as follows:

$i \varepsilon \{0, 1, 2\}, x \varepsilon \{2, 3\}$	Agent zena
$j \varepsilon \{2, 3, 4\}, y \varepsilon \{0, 1, 2\}$	Agent wendy
$i + j < 4, x + y = 3$	Agent celine

The constraint store hosts mainly basic constraints such as $i \varepsilon \{0, 1, 2\}$, $x \varepsilon \{2, 3\}$, $x = y$, and so on. Constraint propagators realise non-basic constraints such as $x > y$, $2 * x - y = z$. Propagators amplify the constraint store. This is elaborated by the following figure 8:

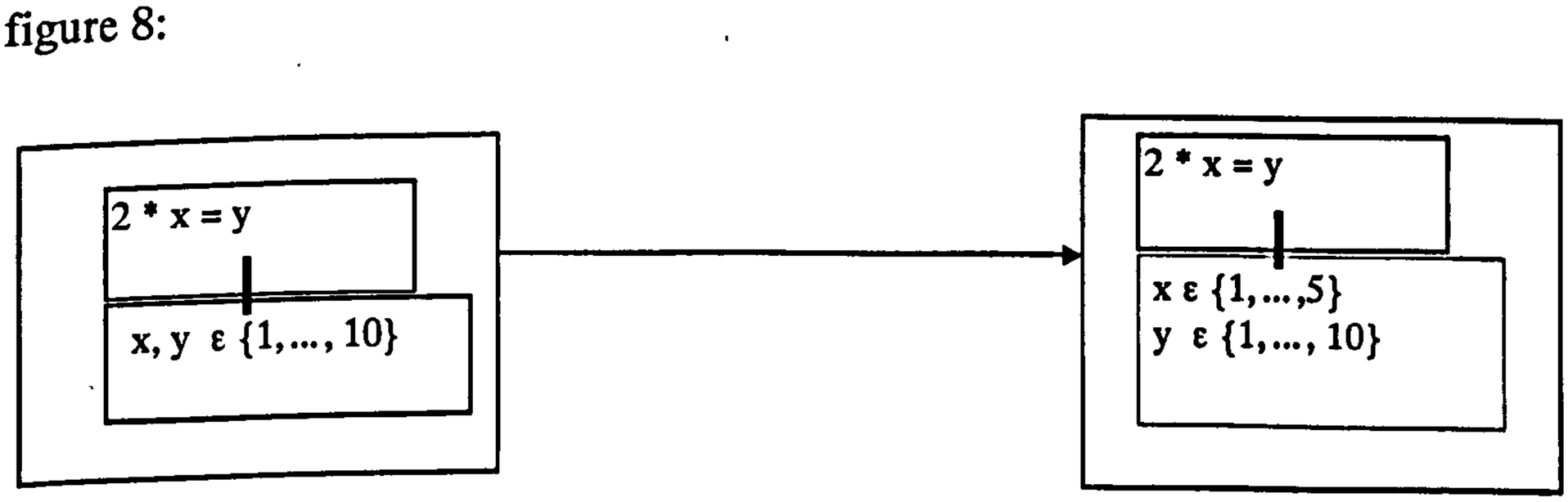

Figure 8: Effects of propagators on the constraint store

The notion storage model is discussed instead of the data model, because the facility of the constraint is independent of the choice of data model — object-oriented, temporal etc.

The reasons for considering the concurrent constraint approach are as follows:

- Simple, powerful;
- Store-as-computation by von Neumann is replaced by store-as-valuation;
- Instead of reading and writing the values of variables, processes may now ask (check if a constraint is entailed by the store) and tell (augment the store with a new constraint); and
- A system of partial information.

In this section, various constraint domains are discussed.

Van Hentenryck et al. (1996) discusses a relatively small number of constraint systems that have been used as a basis for several concrete implementations. The four most important domains are Boolean constraints, finite domains, real intervals, and linear constraints; other examples include lists and finite sets.

Boolean constraints are either treated by a specialised constraint solver, as in CHIP or Prolog III, or seen as a specialised case of finite domain constraints. In the latter, a Boolean constraint is considered as an integer between 0 (false) and 1 (true).

Finite domain constraints are constraints on integer valued variables. These constraints are useful in many applications. Combining propagation techniques with backtracking search usually solves them. Each variable is associated with a finite set of possible values that are domain variables. Inconsistent values are removed from the domain variable during propagation, and then the search tries to assign a value to each variable.

Real interval constraints are the analogue of finite domains when reals are considered instead of integers. As it is impossible to explicitly represent the set of

reals that a variable can take, the domain of a real variable is an interval whose bounds are floating point numbers.

Linear constraints are constraints posted on real variables, which have a special form: they only involve weighted sums of variables (no product or more complex expressions). For such constraints, very efficient constraint solvers have been implemented using the Simplex algorithm as a starting point. Some linear constraint solvers use infinite precision (rational numbers), some others use floating point computations.

Global constraints: The removal of inconsistent values can be tricky for more complex constraints. An important line of work aims to define a good propagation algorithm for more complex constraints. This is sometimes referred to as global constraints. In this context, scheduling, all-different (a set of variables takes on values that are all different), cardinality constraints (the number of constraints within a set that must be satisfied is required to be within given lower and upper bounds), and spatial constraints have been considered in detail in the literature.

User-defined constraints: It is found that from the application of CP tools in practice, domain specific constraints are often needed. In other words, the user of these systems often needs to extend the constraint system with some constraints that are specific to the application in hand. Several approaches have been made for making it possible for the user to add domain specific constraints to the system and to tailor the underlying constraint solver to these specific constraints.

The constraints systems that are discussed have been integrated into different programming languages, ranging from subsets of first order logic to imperative languages such as C++, or even specialised languages. One of the most popular approaches is to use Horn clauses as a basis (as in Prolog), and then extend this with one or more constraint systems, in addition to unification over Herbrand terms. This constraint logic programming approach has led to many important tools that are in the following Table 14:

Table 14: Constraint programming tools

Tools	Constraint systems
CLP ®	linear constraints
Prolog III	Booleans, linear constraints, and lists
Oz	Finite domains
CHIP	Booleans, linear constraints, finite domains
Clp(fd)	finite domains, Booleans
ECL^{i}PS^e	finite domains, linear constraints

3.5 Towards a Constraint-based multi-Agent (CANET) approach

The aim of this section is to propose a framework called CANET (Constraint-based Multi-agent NETwork) to discuss various interactions. The architecture is based on a higher level assumption that "everything is connected", and on two founding premises: first, constraints are an ideal system for representing the many regularities in agent activities; and second, constraints that guide searches, can be used to exploit these regularities, and can focus the knowledge, resources, authority, and control toward useful behaviour, planning, and interaction with other agents.

The proposed approach is a synthesis of constraints and agents for complex applications. Figure 9 illustrates this. The behaviour of an agent and the interactions are related to constraint-based mechanisms. Behaviour and interactions include reactive, planning, co-operation, negotiation, task allocation, and social laws.

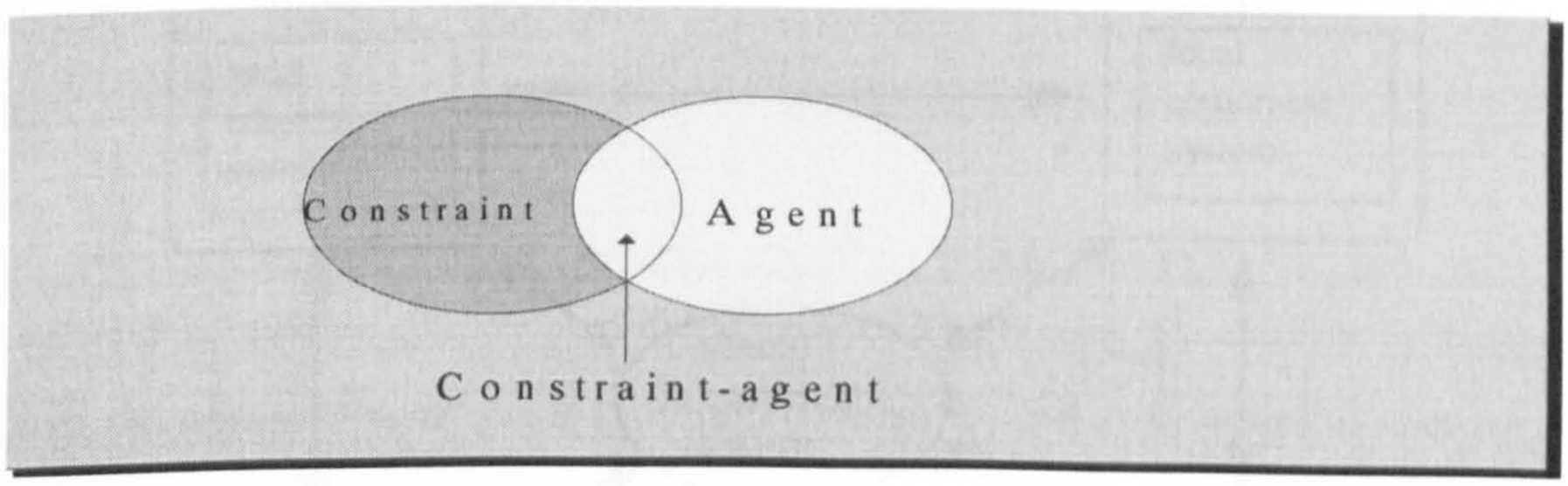

Figure 9: Synthesis of constraints and agents

Within the proposed approach, agents interact via a constraint. Figure 10 depicts this. The constraint store is elaborated in the previous chapter. Agents are able to release various constraints to the constraint store. Various expressions are elaborated in that manner. Constraint propagation allows various expressions to be processed.

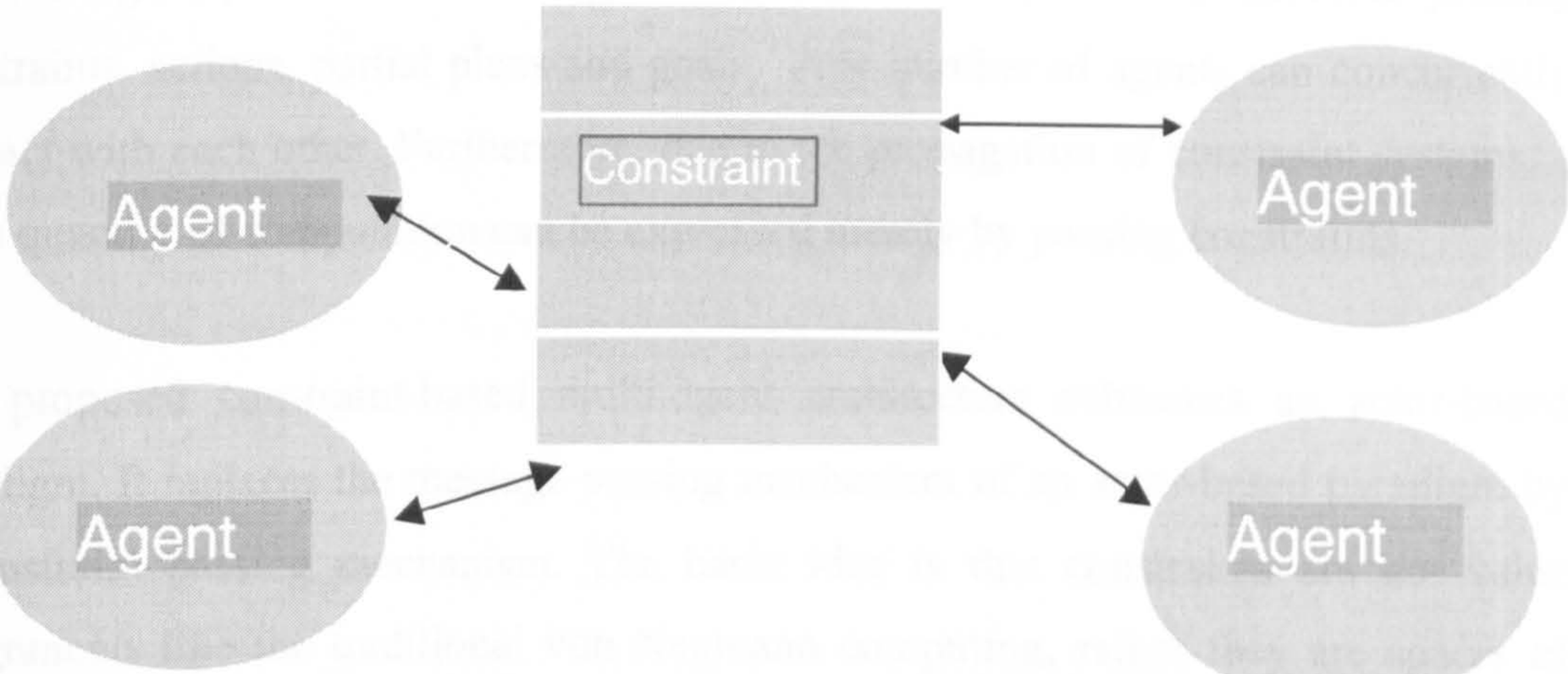

Figure 10: Agents interact via a constraint store

The proposed architecture can be seen as a loosely connected architecture in the sense that it allows a local constraint system for each agent. Figure 11 demonstrates this. Moreover, there is also a shared constraint store for all agents. The architecture enables specialised techniques to be associated with each agent to improve the coherence of the system.

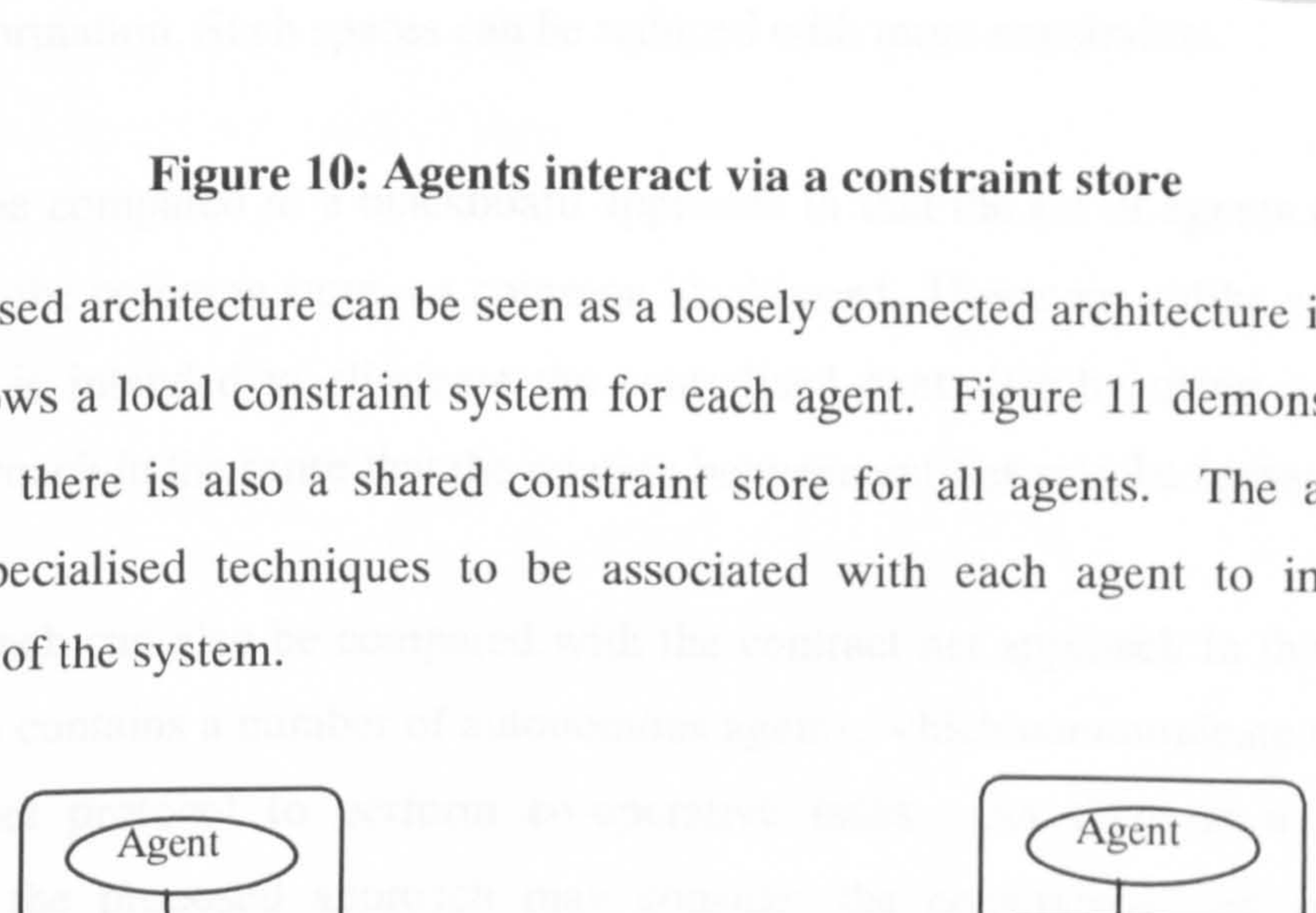

Figure 11: Outline of architecture for application

Communication in the architecture is asynchronous and can take place locally or between agents, or between an agent and the common store. It involves passing constraints, actions, partial plans and goals. Any number of agents can concurrently interact with each other. Furthermore, due to the propagation of constraint systems, a large quantity of computation can be expressed merely by passing constraints.

The proposed constraint-based multi-agent architecture subsumes an actor-based paradigm. It replaces the message passing mechanism of an actor-based paradigm by a constraint passing mechanism. The basic idea is that constraints are not value assignments like the traditional von Neumann computing, rather they are spaces of partial information. Such spaces can be reduced with more constraints.

This can be compared to a blackboard approach in that the set of agents co-operates by sharing the common store – a common blackboard. However, unlike a blackboard system, it is intended to eliminate the centralised controller by using a constraint-based approach in the sense that the relation between entities may be loosened.

The approach can also be compared with the contract net approach in the sense that the system contains a number of autonomous agents, which communicate through the contract net protocol to perform co-operative tasks. As there is a knowledge overhead, the proposed approach may consider the constraint-based approach in allocating tasks, consideration of time and so on.

This system, based on constraint satisfaction, is efficient and concurrency may increase the speed of computation. Since the system is loosely coupled, autonomy of agents is achieved. As the system is distributed, it is much easier to develop and manage various agents. This system is natural as some problems are better described as collections of separate agents. The system deals with resource allocations in that each agent has its own resource and there are also shared resources.

3.5.1 Proposed CANET agent

Within the proposed approach, CANET agents are seen as a knowledge-based system with constraints. The behavioural and interaction capabilities are discussed in terms of constraints. Capabilities such as co-operation, negotiation, task allocation, social laws are treated in such a manner.

Constraints can be used for many purposes for CANET agents, the most significant of which are their uses as guides, describing the structure of agent duties, and focusing its efforts appropriately with regard to the structure. We now investigate the structure, attributes, and types of constraints within the CANET framework.

Constraints can guide agent duties in three ways. First, a constraint may restrict the agent by eliminating one or more implausible alternatives. Second, a constraint can compel the agent towards a duty by support to alternatives that obey the constraint, and finally, a constraint may have a number of relaxation associated with it, when a constraint cannot be obeyed.

Constraints can serve as inhibitors of duty for CANET agents: constraints can inhibit the agent from working with certain agents, performing certain behaviour. Constraints may control the amount of effort spent on a duty. For example, the agent may restrict the amount of time spent on the duty. Constraints can also be used to modify an agent's routine based on current conditions within the environment.

Constraints can be used to represent the agent's expectations of the future that can be represented as restrictions on a future world. Constraints can be used to compel the agent to perform a certain duty. Constraints can represent the interactions between duties or agents. Constraints can also be used to limit the recollection of agents and the amount of information examined during the course of a duty.

Each constraint has a set of criteria under which the constraint is applicable. A constraint may only be applicable in the presence of a certain agent, or when some

events occur. A constraint may be applied to a collection of agents. A constraint may have a specific piece of knowledge on which it is explicitly dependent. Each constraint has a specific lifetime during which it is active.

3.5.2 The structure of the proposed CANET agent

Below, an agent model for application is discussed in detail. The notion of the agent is defined in terms of a knowledge base and a control unit. The main contribution of the approach is the introduction that the behaviour of an agent and the interaction between agents may be related to constraint-based mechanisms.

The layers in the proposed model are a negotiation/co-operation layer for multi-agent interactions and a behaviour/planning layer for the agent to interact with its environment.

Knowledge abstraction needed for describing the behaviour of agents and interaction between them are planning, co-operation, negotiation, task allocation, and social laws.

The aim of various components is as follows. There is a co-operation layer that is responsible for maintaining co-operation and to deal with joint plans in actual interactions with other agents. Below the co-operation layer, the local planning component is placed. It deals with the goals that an agent can achieve on its own. Below this, the reactive layer deals with situations that require an immediate reaction.

Knowledge abstraction reflects the complexity of the real-world application concerned. Therefore, not all the components need to be involved in real-world problems. If the real-world problem requires an immediate response, then the behaviour-based component deals with it (these are implemented as if-then rules).

However, if the problem is complex, the agent may need to shift to the planning component. The planning component allows the achievement of goals from given initial states. During the process, the knowledge about action interference and/or hazards may be taken into consideration.

If one single agent cannot solve the problem, the task needs to be allocated to other agents. Therefore, the agent needs to shift to the co-operation component. For the co-operation process, agents may have preferences with which agents to co-operate.

In certain scenaria, not all the preferences need to be satisfied. In such situations, the agent may need to negotiate with other agents. In such situations, we propose two strategies: a preference for maximal constraints, and the sum of preferences.

Unlike a single agent-based architecture, each agent needs to react additionally to changes resulting from interactions with other agents as well as to changes induced by the external world. The *external* world can be seen as another agent, which can pass constraints and occurred actions to other agents or to the shared system. The difference is this agent is not negotiable. Its constraints are regarded as non-relaxable.

Within the CANET approach, belief, desire, goals and so on are treated as constraints. The intention of an agent is seen as constraint satisfaction. Awareness of other agents comes from interaction with others. Reactivity of an agent is treated as a constraint that can itself be a rule. Pro-activity of an agent is treated as a constraint search that involves satisfying various constraints.

Table 15: Synthesis of constraints and agents

Constraints	Agents	Constraints and Agents
There are precedence/resource constraints, which can also be used in agent interactions. Constraints can be a number (integer, real)/ list of numbers/ text strings/ and so on.	Agents can have a unique name, attributes, and methods. Agent <name> (attributes: Features: methods:<set of behaviours planning|rules|) Properties of an agent : Autonomy/social abilities/reactivity/pro-activity (Wooldridge et al (1995)	Each agent is associated with constraints and the interactions between agents may be related to constraint-based mechanisms. Constraints can be used in agent communications, and constraints can be used to co-ordinate the activity of various agents. Reactivity (behavioural) is seen as a constraint, and pro-activity (planning) is treated as constraint search.

Table 15 depicts the synthesis of constraints and agents for applications. Constraints include precedence, resource, and so on. Properties of an agent include autonomy, social ability, reactive, and pro-active.

3.5.3 Proposed interaction between CANET agents

Within the CANET approach, communication between agents is treated as constraint passing. The contents of the communication can be a number, a list of elements, various message types, partial information, or even a complex task.

The interaction of agents – a constraining behaviour – may itself be constrained. We propose a constraint-based view of interaction that includes co-operation, negotiation, task allocation and so on.

In table 16, characteristics of interaction and their loose definition and constraint-based view of that interaction are tabulated. These will be elaborated in the next chapter.

Table 16: A constraint-based view of CANET agent interactions

Characteristics	Elaboration	Mechanism
Communication	Dissemination of information among agents	Constraint passing
Co-operation	Working together to achieve a common goal	Constraint passing
Negotiation	Reconciling the differences among agents	Constraint relaxation
Co-ordination	Bringing into a required or proper relation to ensure effective operation or harmony	Constraint passing/ constraint satisfaction
Social law	Agent society adopts a set of laws which specify how individual should behave	Hard constraints
Task allocation	Matching capabilities of agents and tasks	Constraint satisfaction

Research is carried out on the synthesis of constraints and agents in one form or another. These works tend to focus on either co-operation, or social law, and so on. The proposed method unifies various methods in the sense that the constraint-agent view is sufficient to treat various interactions. These works enhance our overall hypothesis that the behaviour of an agent and the interactions between agents may be related to constraint-based mechanisms.

In the preceding chapter, an overview of an agent is provided. In this chapter, constraint-based techniques are addressed. Our aim is to synthesise them for applications. Table 16 summarises a constraint-based view of behaviour and interaction between agents. This will be elaborated in chapter 4.

3.5.4 Strengths and limitations of the CANET approach

The strength of the CANET approach lies in the fact that it deals with various constraints. The approach synthesises constraints and agents. The agents interact via a constraint store. Constraint solving is a better defined search strategy that can be applied to solve complex problems. In addition, constraint-based techniques can be applied in a situation where changes can be propagated along the network.

Propagation constraints are commonly called constraint agents. The behaviour of a constraint agent is to propagate information to the underlying store. In this case, the underlying store is a constraint; the information propagated is expressed as primitive constraints.

The specific constraints associated with the message are usually derived from the content of those messages and general principles of agent behaviour. Typical examples of these principles are *veracity* (an agent must tell the truth), *autonomy* (an agent may not constrain another agent to perform a service unless the other agent has advised its willingness to accept such a request), and *commitment* (if an agent advertises its willingness to perform a service, then it is obliged to perform that service when asked to do so). All these general principles of agent behaviour are known as the 'mental state of the agent' (Shoham 1993).

Constraint agents in the literature are processes that involve a fixed set of variables. During their lifetime they alternate between suspended and waking states. They are woken from suspension when an extra primitive constraint on one or more of their variables is recorded.

3. 6 Related work

There are few works that put emphasis on constraints and agents. For example, Freuder and Wallace (1997) describe a paradigm for content focused matchmaking based on a recently proposed model for constraint acquisition-and-satisfaction. Matchmaking agents are conceived as constraint-based solvers that interact with others, providing potential solutions based on partial knowledge. Such a constraint acquisition-and-satisfaction approach can also be incorporated into the CANET approach.

Carlson et al. (1997) describe how global constraints and local agents can be combined to control the overall behaviours of smart matter in a simple and robust manner. Within the CANET approach, we discuss global constraints as social laws for a society of agents.

Paredis (1994) introduced CCS, a co-evolutionary approach to constraint satisfaction. Two types of objects - constraints and solutions - interact in a manner modelled after predator and prey relations in nature.

Anderson (1995) examines the nature of everyday activities and develops a computational architecture for an agent able to participate in such activities. He presents a theory of improvisation to address everyday activities. He demonstrates architecture embodying the improvisational approach based on the use of constraint-directed reasoning.

Languenou et al. (1998) present a virtual cameraman which provides the user with camera movements satisfying user defined constraints specified in the image space and/or constraints on the objects of the scene.

3.7 Conclusion

In this chapter, the CANET architecture for complex application has been proposed. Within this approach, each agent has a local constraint system, and agents interact via a constraint store. We have proposed the structure of an agent, and agent interaction on a constraint-based view. In the next chapter, we will discuss how the CANET architecture can be implemented, and how CANET can be applied to logistics applications.

4. System Design/ Implementation of constraint-based multi-agent system, and applications

4.1 Motivation

The main aim of this chapter is twofold. First, we will discuss how the CANET approach can be implemented. Agents are implemented as concurrent objects, and multi-agents are implemented as concurrent objects under constraints.

Second, the proposed approach is applied to a transportation scenario. Our strategy is as follows: firstly, we want to move away from the contract net approach due to its limitations. However, we also extend the contract net approach to pass constraints between each other. Secondly, we create a transportation scenario in which various societal notions are related to constraint-based techniques.

4.2 Implementation of CANET

4.2.1 Objects to concurrent objects

The world in which we live is concurrent in the sense that there are multiple active entities; *distributed*, meaning that there is a distance between entities that yields a propagation delay in communication between them; and *open*, meaning that the entities and their environments are always changing. Computation can be considered as a simulation of part of the real or an imaginary world.

To solve a simple, small problem, sequential computing is usually sufficient. However, when the problem becomes larger and more realistic, it is much easier to model it as concurrent, distributed, or open computing. For example, if a system has multiple users at a time such as in banking or airline reservation systems, one would model the problem in the form of concurrent or distributed computing.

The notion of concurrent objects is an extremely valuable development. A concurrent object contains a virtual processor. Here one can eliminate the notion of processes, which is necessary in concurrent programming using sequential objects.

Programmers don't have to describe execution control. Concurrent objects are executed in the same way as in time-sharing systems.

For complex applications, we need a higher level module than a concurrent object for constructing larger systems.

4.2.2 Agent as 'concurrent objects' under constraints

An agent is composed of concurrent objects, in the same way as a person is composed of cells living concurrently. An agent is the unit of individual software that interfaces with humans, other agents, and the real world. Each agent has its own goal, and reacts to its environment. The collections of agents form a society. Agent behaviour and agent interaction are seen as constraints. The exchange of information is sent by constraint passing. The knowledge layers are reflected by the Oz objects.

4.2.3 Multi-agents as concurrent objects, agent behaviours, agents interaction as Constraint Satisfaction

Our approach is that a multi-agent system consists of a collection of agents, and can be implemented as concurrent objects. The interaction between agents and the behaviour of an agent are related to constraint-based techniques.

Objects are the primary concurrent structuring concept of Oz. They combine data encapsulation through procedural abstraction with state and mutual exclusion. Objects can be seen as service providing agents. The agent layer is a hierarchy of Oz classes.

Consider an example of how to represent a Truck agent in Oz. The truck agent has attributes such as step of movement, and state. The truck has a unique identification that is represented as a feature in Oz. The Truck agent might have different behavioural aspects such as reverse, move, and so on.

The representation of a class agent is as follows:

```
class Truck attr step:3 state:off feat id
meth reverse
  Step<~1*step
end.
...
```

The user can then create various instances of driver agents with specific id's.

Different agents can be represented in the above manner. The knowledge layers are also implemented in the same manner.

The interactions between agents are represented as constraints. Suppose in a situation where two goals of a truck agent (speed1, speed2) are the same. That is simply represented as speed1 = speed2.

4.3.4 Suitability of Oz language (now called Mozart system) for CANET applications

The Mozart system provides state-of-art support in two areas: open distributed programming, and constraint-based inference. Mozart implements Oz, a concurrent object-oriented language with dataflow synchronisation.

4.3.4.1 About Oz

Oz is a concurrent object-oriented language. It is based on a new computation model for higher order concurrent constraint programming (CCP) that provides a uniform foundation for functional programming, constraint, and logic programming, and concurrent objects with multiple inheritance. From functional languages, Oz inherits full compositionality, and from logic languages, logic variables and constraints.

Oz supports a number of search strategies, and Oz is a good platform for adding new kinds of search strategies. Searching in Oz is encapsulated and programmable, so it is easy to program, e.g., one solution, best solution, all solutions, and branch and bound strategies.

DFKI Oz is an interactive implementation of Oz featuring a programming interface based on GNU Emacs, a concurrent browser, an object-oriented interface to Tcl/Tk, with powerful interoperability features, an incremental compiler, and support for stand-alone applications. Performance is competitive with commercial Prolog and Lisp systems. Oz is designed as a successor to languages such as Lisp, Prolog, and Smalltalk, which fail to support applications that require concurrency, reactivity, and real-time control.

4.3.4.3 Why Oz?

To implement the CANET approach, we have chosen the Oz language. We will explore various features of Oz in this section.

Oz is based on logic variables and fair concurrent control. Variables can be used before they are assigned values, and multiple computations are advanced fairly. If a computation requires the value of a not yet assigned variable, the computation suspends automatically.

Oz can compute with variables whose values are only partially specified. Information about the values of variables can be specified by means of constraints.

Oz comes with powerful predefined search abstractions, including depth-first one solution, demand driven multiple solution, all solutions, and best solution (branch and bound search).

Oz comes with powerful constraints for variables constrained to finite sets of non-negative integers (so-called finite domain variables), including addition, multiplication, and comparisons.

Objects are the primary concurrent structuring concept of Oz. They combine data encapsulation through procedural abstraction with state and mutual exclusion. The services of an agent are provided through methods and can be requested by sending messages to the object. Objects are created as an instance of classes. Classes define

methods, attributes, and features. The definition of a class may involve inheritance of the classes.

Oz is a higher-order language. Functions, procedures, objects, classes, methods, and modules are created dynamically and are designated by first-class values, which may be passed as arguments.

Oz is well suited for reactive programming with soft real-time requirements. Timers and access to real-time are available. Control with time-outs is easily expressible.

A CANET agent is implemented as concurrent objects, and the behaviour and interaction of an agent are implemented as constraints in Oz.

4.3.4.2 Strengths and limitations of Oz/Mozart

The Mozart programming system is a general-purpose development platform designed to support concurrency, distribution, resource-aware computation, and symbolic computation and inferencing. Mozart implements Oz, providing the abilities of constraint, logic languages. Current research in Mozart includes high-level abstractions for fault tolerance, security, and implementations for devices with restricted resources. The high level representation of Oz enables clean prototyping because it frees the developer from low-level details.

Mozart/Oz system is an excellent tool for research and development of agent-based systems due to the reasons discussed in the preceding section. However, we explore below the limitations of the Oz language.

The Mozart system does not address issues like database connectivity, or web enablement for industrial applications in its current form. Due to the difficulty in implementing persistent objects, recovery from failures is currently not possible.

Oz does not provide a debugger that makes it difficult to trace during the development. Oz is provided with an Emacs interface that has difficulties in moving files.

Oz also does not provide a real-interval constraint system, but allows one to implement interval constraints over real numbers.

4.3 Towards a constraint network formulation

4.3.1 Logistics scenario

In the last section, the CANET architecture for complex applications was proposed. The CANET approach is based on the idea of the synthesis of constraints and agents for complex applications.

The aim of this section is to demonstrate the relevance of the CANET approach, the agent model and the interactions discussed in the previous section to a logistics management problem. This would enable us to make a limited comparison with that of Fischer and Kuhn (1993).

4.3.2 Logistics problem

Rittmann (1991) states that more than one third of the trucks in the streets of Europe are driving without goods, since they are on the way to pick up goods or on their way back home. This shows that the actual planning in the transportation domain is far from satisfactory. Due to the distribution of knowledge, resources, authority, and control, due to the complexity of worldwide technology, and due to environmental and economical reasons, there is a necessity for an agent framework for such applications. Within such applications, various knowledge sources need to interact. Interaction concepts such as co-operation, negotiation, task allocation, social laws and so on are introduced. This interaction tends to reflect the agent's social abilities. There are constraints such as precedence, resources, and so on, to consider.

In chapter 1, we discussed briefly the 'transportation problem', and discussed Fischer and Kuhn's (1993) approach. We have argued that the contract net approach is a highly regulated and ordered society of agents at successive levels of hierarchy having less and less autonomy: this is more like a rigid structure with only a consideration to cost rather than a society of companies involved in fierce competition. The

simulation does not address issues like preferences of drivers, specialisation of drivers, and various conflicts, dynamic environments in which an agent may not possess knowledge about other agents, and may have changeable goals, and be subject to interruption from external events. We have also argued that within the completely decentralised model, the manager is actually surplus to requirements.

4.3.3 The relevance of constraints, agents for logistics applications

AI/DAI approaches appear well suited to tackle such a scenario. First, there is the complexity of the scheduling problem. Second, common-sense knowledge (e.g., topological, temporal, and expert knowledge) is necessary to tackle such problems. Third, local knowledge about the capabilities of a Transportation Company as well as knowledge about competitive companies massively influences the solutions.

4.4 CANET experiments

In this section, experiments have been carried out to satisfy our hypothesis that the behaviour of an agent and the interaction between agents can be related to constraint-based mechanisms. In Chapter 1, we have listed the limitations of Fischer and Kuhn's (1993) approach. We will address most of the limitations within this section.

The scope of this scenario is extended to Fischer and Kuhn's (1993) in the following manner. We simulate interaction between the similar agents (i.e. drivers), simulate preferences, and external (dynamic) interaction.

Within the scenario, trucks move from one city to another to deliver goods. The agents within such application have various reactive, reasoning, and social abilities.

The main objective of the experiment is to simulate the interaction between agents. The agents that participate in the scenario are the broker, transportation companies, driver, and truck agents. Constraint interactions are addressed. In particular, constraint communication between agents is explored.

4.4.1 Constraint-based contract net, communication as constraint passing

Smith (1977) introduced the contract net approach. In a contract net, a certain task may be given to a society of agents. A special agent, the manager, receives that task and possibly divides it into subtasks. The manager then broadcasts to all eligible agents. These eligible agents calculate the cost of performing the task and report to the manager. The manager selects the best one and a deal is struck between them. Communication between agents within the contract net is via message passing.

Within this section, an extended constraint-based contract net approach is firstly presented for a transportation application. The approach differs from Fischer and Kuhn in the sense that the communication variable is not just treated as variable but as a constraint. That has interesting effects. For example, within the transportation simulation, companies and drivers are able to communicate not just numbers, but a constraint. Examples of communication include: Xcost = 105.5; Ycost=104; Zcost ='message'; Icost=[23 34 56]; and Jcost= {Min 23 34}, where Xcost, Ycost, Zcost, Icost, and Jcost are the communication constraints.

The main objective of this experiment is to simulate constraint communication between agents.

4.4.2 Social laws as hard constraints, co-operation, co-ordination as constraint passing

The aim of the experiments is to explore horizontal interaction, social laws, co-operation, and co-ordination. Unlike the contract net approach that was developed at DFKI, there were no horizontal interactions between agents.

Within the following scenario, there are three trucks T1, T2, and T3. The truck agents move from one location to another. The aim is to co-ordinate/co-operate their movements based on their individual speed, and their relative speeds. It is interesting to note the idea of co-operation discussed in multi-agent systems, and co-ordination discussed within Distributed Problem Solving (DPS), both constraints within the CANET approach. By adding the constraints in a sequential manner, appropriate

behaviour is obtained. Within this scenario, external users can set the constraints. The behaviours of agents are as follows: setSpeed, communication, and travel.

Adding the following three constraints allows the agents T1, T2, and T3 to expect a speed value X, Y, and Z to be provided:

{T1 setSpeed(X)}; {T2 setSpeed(Y)}; {T3 setSpeed(Z)}

Possible options are as follows:

X = 30; Y= 40; Z =50.

Adding the above constraints allows the truck to move in an autonomous manner.

In the following instructions, we demonstrate that it is possible to communicate constraints:

{T3 communication(stop)}; {T2 communication(reverse)} ;{T1 communication({Min 50 49 $)}
{T1 setSpeed(N1+N2)} N1=23 N2=34

In the following scenario, the agent's goals are co-ordinated/co-operated by constraints:

Let the goals of T1, T2, and T3 be X, Y, and Z respectively. The relationship between goals are listed below:

{T1 setSpeed(X)}, {T2 setSpeed(Y)}{T3 setSpeed(Z)}; X = Y+Z; X=2*Y; X>:45

We intend to remove the number of interactions such as co-operation by negotiation, or by other means, by giving consideration to social laws. A law may be seen as a rule of conduct put down by a controlling authority to a society of agents. Our research interest lies in that of constraints to describe the phenomena. Social laws may be addressed in very complex applications such as in a logistic management scenario.

We consider an example in which two Trucks agents may be present. Let them be T1 and T2 and let us say that their movements may need to be co-ordinated:

{T1 speed (S1)}
{T2 speed (S2)}

The effect of the constraints is seen on the movements of trucks. Within the experiment carried out, it is possible to select the trucks to double or halve their speed, set different speeds to different trucks, give different speed limits on different days and so on. It is possible to specify some incomplete information to the speeds and so on. If the speed of T1 is twice of that of T2, then such constraint is described below:

```
S1 = 2*S2;
```

A user or a central authority can introduce social laws such as by adding a constraint that SPEEDLIMIT <= 50, and assigning various goals of an agent such as S1 = S2 SPEEDLIMIT. Due to the propagation of constraints, the range of speed is forced to less than 50. From a constraint-based point of view, social laws are implemented hard constraints that cannot be relaxed.

Using trucks as agents, we have shown that it is possible to co-ordinate agents at the same level, which demonstrates that agents at the same level may be co-ordinated by a constraint based approach. The aim of this experiment is to achieve co-ordination between agents at the same level, for example, in an existing framework such as a contract net for horizontal co-operation.

The advantage is that one could simply accommodate horizontal co-operation between agents. This has far reaching implications. Unlike the approach implemented by Fisher and Kuhn for a transportation domain, one could consider not only interaction between agents at different hierarchy levels but also between agents at the same level.

4.4.3 Task allocations as constraint satisfaction

Consider the transportation domain that was discussed extensively by Fischer and Kuhn (1993), stated as a contract net problem. We treat the problem as a problem involving a group of agents organised through the extended contract net. Task allocation between agents may be seen as matching capabilities of agents and tasks. There are a number of trucks (T_1, T_2,..., T_n), and a number of drivers (D_1, D_2, ..., D_m) that can carry a range of goods, including animals and so on. These trucks belong to different companies (Ama, Cala, Ola, etc.) and the drivers that drive the trucks have also indicated a preference over when they would prefer to drive, morning or afternoon etc. 'Animals' can only be transported during the day, and 'oil' can only be shipped during the night, due to safety issues.

We intend to discuss the problem in terms of preferences rather than a centralised mechanism. To demonstrate the approach, we started by listing the preferences of agents in terms of constraints. For examples, certain trucks for certain goods, certain drivers for certain goods, certain products at certain times and so on.

The overall aim is to consider all the constraints and allow a search strategy to satisfy these. The result of the task allocation process is computed through a constraint network. The aim of the experiment is to satisfy all constraints. This is achieved by means of a constraint search mechanism.

The proposed approach is quite flexible in the sense that most constraints can be dealt with. There is no fixed organisational structure required such as hierarchical or decentralised. Due to the use of constraints, correctness is guaranteed theoretically.

There are trucks ranging from 1 to 5, where 1 is the first, and 5 is the last truck in the order. There are 25 different properties, and each of these properties must hold for exactly one truck. The aim is to assign values such that all constraints are satisfied.

Drivers = [D1, D2, D3, D4, and D5]
Company Names = [BAMA, CALA, OLA, AMA, PALA]
Time = [Morning, Evening, EarlyMorning, Afternoon, Night]

Goods = [Biological, Chemical, Oil, Animal]

Trucks = [T1, T2, T3, T4, T5]

These and other constraints were coded as Oz instructions (see Table 17); these constraints are important preliminaries for task allocation. A front end was written in Oz, which allowed a user to describe his or her 'transportation' requirements, and the program computed the deployment of trucks. Drivers are scheduled according to what time of day each of the trucks leave: in effect our constraint-based program allocated tasks.

Table 17: Constraints and instructions in the task allocation process

Constraints	Oz instructions
Certain goods can be carried at certain time *Oil may be transported at night, and biological may transported in evening*	Oil = Night Biological = Evening
The preference and dislikes of agents *Driver 2 does not want to work at night*	Driver2 \= Night
Possible conflict *Truck 2 should not be loaded with oil*	Truck2 \= Oil
Companies specialising in transporting certain goods	Bama = Biological, Cala = Chemical, Ama = Animals, Ola = Oil, Miscellaneous = Pala
Drivers specialist in driving with goods/ companies	D1 = Oil, D2 = Ama
Restrictions on the transit times of certain products	Or Morning = Animal [] EarlyMorning = Animal
Truck ordering *The truck ordering can be in terms of goods they carry*	Animal = Biological + 1 Chemical = Animal +1 Oil = Chemical + 1 Physical = Oil + 1
Options *Driver 5 specifies a preference in a truck based on truck ordering*	OR D5 = Ama –1 [] D5 = Ama + 1

Figure 12 illustrates the results of the task allocation scenario.

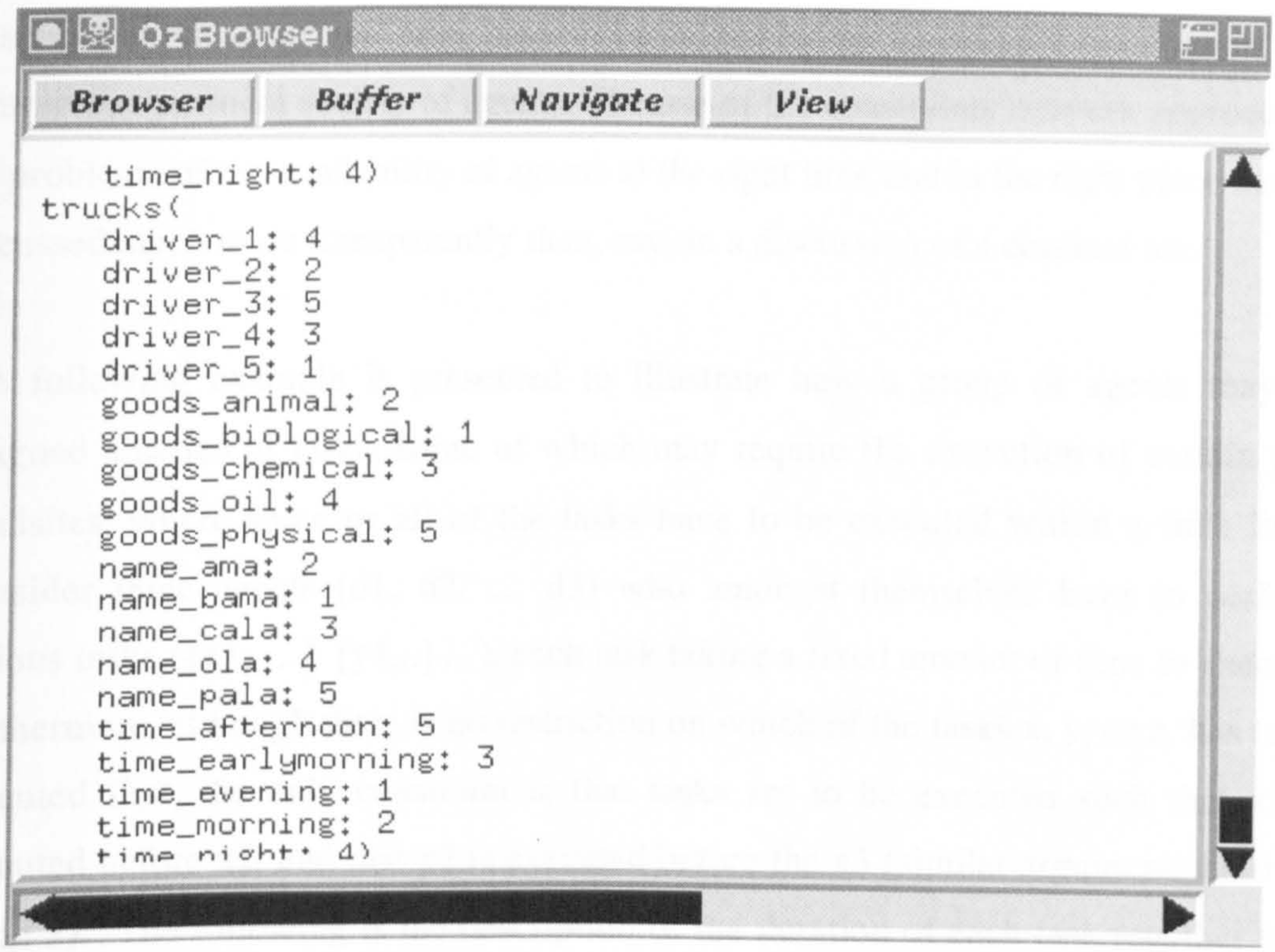

Figure 12: Result of task allocation simulation

The result of the simulation through a constraint network is depicted in the following Table 18.

Table 18: The results of the task allocation

Truck #	Driver #	Goods	Company	Time
1	5	Biological	Bama	Evening
2	2	Animal	Ama	Morning
3	4	Chemical	Cala	Early Morning
4	1	Oil	Ola	Night
5	3	Miscellaneous	Pala	Afternoon

4.4.4 Constraints network and task scheduling

We have thus far argued about constraints that were descriptive in nature and static in the way we specified these constraints. These included the preferences of a driver, the kinds of goods that can be carried on some trucks (and not others) and at certain specific times only. In a real world scenario such static descriptions have to be augmented with considerations on how long it takes an agent to match the task to be executed, as the agents may not have enough time to execute the task or may not be in the right place to do so. Such problems, related as they are directly or indirectly to

reasoning about space and time, have to be solved by the so-called broker agents in a contract net protocol society of agents. The use of the constraints network approach to the problems of the availability of agents at the right time and in the right place can be discussed much more transparently than, say, in a discussion of a contract net.

The following example is presented to illustrate how a group of agents may be assigned a series of tasks, some of which may require the execution of certain pre-requisites, where some or all of the tasks have to be executed within a time limit. Consider three agents (d1, d2, ... d3) who amongst themselves have to perform various tasks ({x1, ...} {y1,..}...), each task taking a fixed amount of time to execute. Furthermore, although there is no restriction on which of the tasks x, y, or z, has to be executed first, the only constraint is that tasks are to be executed such that x1 is executed before x2, and that x2 is executed before the x3 (similar arguments hold for y and z). The following is the description of the duration of each task together with the agent involved in the execution of the task.

Table 19: Distribution of tasks for three agents and the allowed time duration. Recall that in addition to the time constraint, there are precedence constraints such that x1 should be executed before x2, and x2 should be executed before x3.

Agent	Tasks	Duration	Predecessor
d1	x1	3	x2, x3
	x3	2	none
	y1	3	y2,y3
	Z2	2	z3
d2	x2	2	x3
	y2	2	y3
d3	y3	5	none
	z1	3	z2,z3
	z3	4	none

The model introduces for each task a variable which stands for the start time of the task. The end time of each task is its start time and duration. For the time origin, 0 is assumed. The obvious upper bound is the sum of the duration of all tasks.

From the predecessor relation, it is possible to obtain precedence constraints. That can be generalised as: Start.Pred + Dur.Pred =<: Start.Task

The problem specification which is a direct implementation of Table 19 is given below to deliver goods:

```
Goods = goods(tasks:[x1(dur:3 res:d1)
                     x3(dur:2 res:d1)
                     y1(dur:3 res:d1)
                     z2 (dur:2 res:d1)
                     x2 (dur:2 res:d2)
                     y2 (dur:2 res:d2)
                     y3 (dur:5 res:d2)
                     z1 (dur:3 res:d3)
                     z3 (dur:4 res:d3) ])
```

The scheduling program uses a procedure to compute duration and start records from the specification satisfying all precedence constraints, and assigning start times. The results of the scheduling between agents are shown in Fig 13.

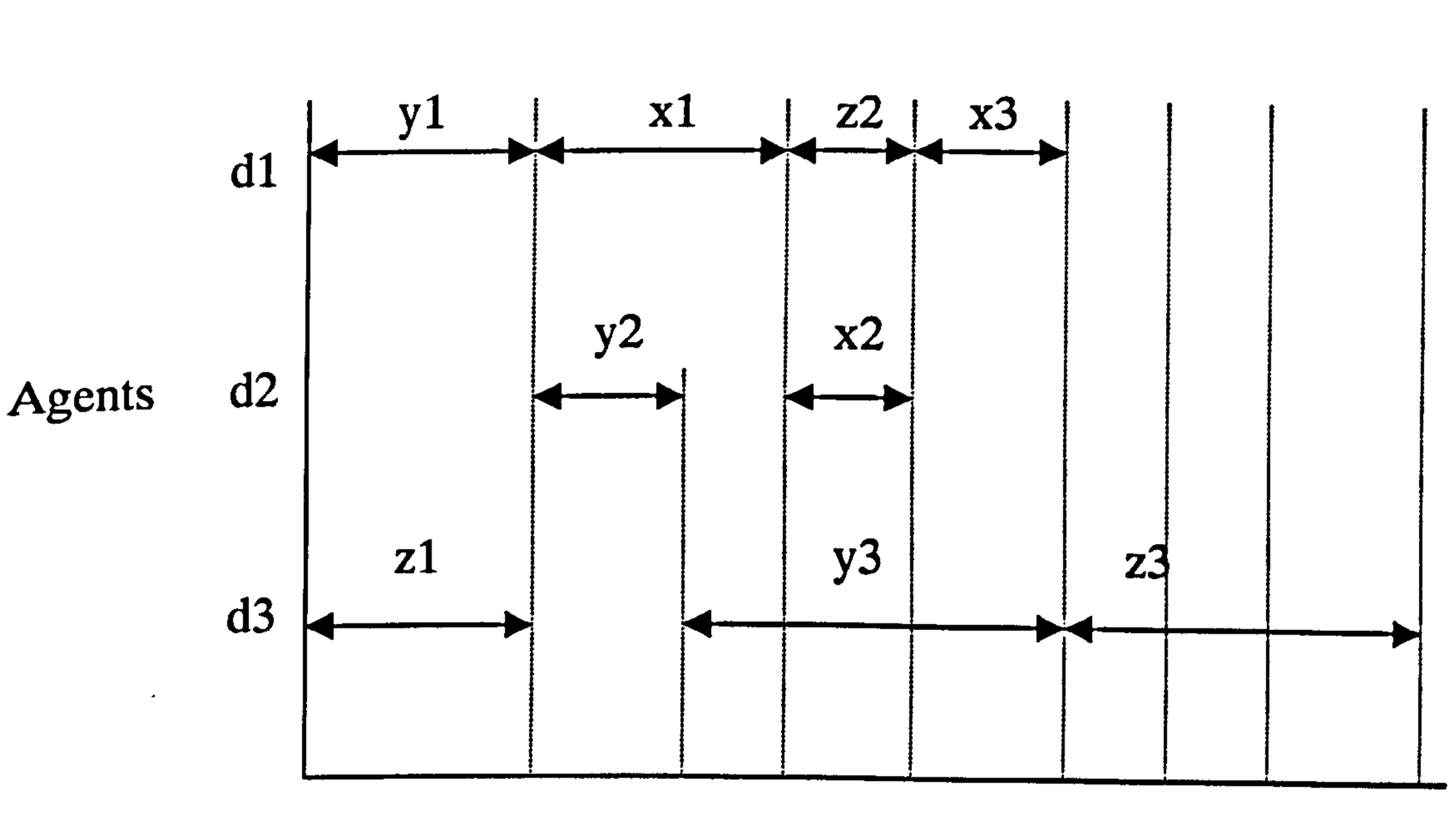

Figure 13: Task scheduling with consideration to time constraints

4.4.5 Negotiation as constraint relaxation for conflict resolution

We now consider an overconstrained problem for which it is impossible to satisfy all constraints. The problem specification will distinguish between primary and secondary constraints, and the goal is to find a solution that satisfies all primary constraints and as many secondary constraints as possible.

4.4.5.1 A preference for maximal constraints

The following example is presented to demonstrate the approach of maximal constraints satisfied. Consider five drivers C1, C2, C3, C4, and C5, which amongst them have to co-operate to achieve tasks. Due to the nature of the business, each of them is assigned to work with two drivers. However, drivers are allowed to specify as many preferences of partners as possible. In this example, six preferences are given:

Preferences = [C1#C2, C3#C4, C4#C5, C5#C1, C2#C3, C1#C3]
Drivers = [C1 C2 C3 C4 C5]

Where C1#C2, C2#C3 states that C1 prefers to co-operate with C2 whereas C2 prefers to co-operate to C3.

The model has a variable Cp for every person, where p stands for the order p takes in the drivers' alignment. Since there are exactly 5 persons, we have Cp ε 1#5 for every person p. We have Cp <> Cq for every p, q are different drivers. The model has a variable Si ε 0#1 for each of the 6 preferences, where Si = 1 if and only if the ith preference is satisfied. To express the constraint, we constrain the control variable S of a preference "driver p wants to co-operate with driver q" by means of the reified constraint:

$$(|Cp-Cq| = 1 <-> S=1) \wedge S \varepsilon 0\#1$$

Finally, there is a variable Satisfaction = S1 +...+S6 denoting the number of satisfied preferences. We need to find a solution that maximises the value of satisfaction.

The result of the simulation is shown in Table 20. The best result is the satisfaction of 5 preferences, and the constraint C1#C3 (C1 wants to co-operate with C3) was not satisfied.

Table 20: The result includes the number of preferences satisfied, the agents who co-operate, and the constraints not satisfied

Preferences satisfied	Agents to co-operate	Constraints not satisfied
5	C1 C2 C3 C4 C5 Means C1 with C2, C5 C2 with C1, C3 C3 with C2, C4 C4 with C3, C5 C5 with C4, C1	C1#C3
4	C1 C2 C3 C4 C5 C1 C3 C2 C4 C5 C3 C1 C2 C5 C4	C5#C1, C2#C3 C1#C2, C3#C4 C2#C3, C5#C1

4.4.5.2 The sum of preferences of agent

Unlike the previous approach, which is restrictive, we are assigning preferences for agents or for constraints. The approach is preferences-based in the sense that either the preferences of agents are all treated equally or agents themselves may be assigned some preferences values and the maximum preferences can be preferred and so on.

The following example is presented to demonstrate the approach of the sum of preferences of agents. Consider five drivers C1, C2, C3, C4, and C5 who amongst them have to co-operate to achieve tasks. As mentioned above, due to the nature of the business, each of them is forced to work with two drivers. In this example, six preferences are given:

Preferences = [C1#C2 C3#C4 C4#C5 C5#C1 C2#C3 C1#C3]

Drivers = [C1 C2 C3 C4 C5]

Attribute = [5 4 3 2 1]

Where C1#C2, C2#C3... states that C1 prefers to co-operate with C2 whereas C2 prefers to co-operate with C3.

However, unlike the previous strategy, the agents themselves are given certain priority values that will be taken into consideration. The attributes of agent preferences are listed in Table 21. There is a constraint that the maximum sum of agent attributes should not be greater than 14.

The approach is similar to the previous approach but instead of calculating the maximal satisfaction, we need to consider the sum of preferences of agents.

Table 21: The agent preference

Agent	Preference attribute
C1	5
C2	4
C3	3
C4	2
C5	1

The result of the simulation is shown in Table 22, where the first column describes the number of preferences satisfied. The second column depicts the total preferences of values assigned to agents. In the last column, the constraint satisfied with agent preferences is listed.

Table 22: Conflict resolution using sum of preferences

Preferences satisfied	Sum of agents preferences	Agents to co-operate	Constraints and agent preferences
5	15	C2 C1 C5 C4 C3	C1#C2(5), C3#C4(3), C4#C5(2), C5#C1(1), C2#C3(4)
4	15	C2 C1 C3 C4 C5	C1#C2(5), C1#C3(5), C3#C4(3), C4#C5(2)
3, 2, 1	15		

4.5 Analysis of results

We have implemented CANET architecture to create a society of agents in Oz language for a logistics application. We have implemented agents as concurrent objects, and the multi-agent systems as concurrent objects under constraints. We have carried out experiments to satisfy the approach using constraint-based methods to

introduce an abstraction that can be used to articulate how the agents communicate, co-operate, negotiate, and how social laws are to be introduced.

First, we have extended the contract net approach in which communication between agents is implemented as constraint passing. That differs from the traditional message passing of Fischer and Kuhn (1993) for the logistics application. Constraint communication allows the agents to communicate partial information, numbers, messages, range of values, and so on.

Second, we have simulated the agency within the logistics scenario in which co-operation/co-ordination is treated as constraint passing. Negotiation is treated as constraint relaxation. Social laws are treated as constraints. Task allocation between agents is seen as constraint satisfaction. Task scheduling is also addressed with consideration to time.

Based on the simulation, Tables 23 and 24 depict the behavioural and reasoning capabilities of an agent for the transportation scenario.

Table 23: The behaviour capabilities of agents

Agents	Reasoning
Broker agent	AddCo getCos, remCo, addDriver, remDriver, addHazard, remHazard, getTrucks, addTruck, remTruck, addMap, announce, and so on.
Transportation companies	init, announce, addCity
Driver agent	init, announce
Trucks agent	init, move, toggle, route, change,...
Track agent	AddHazard, remHazard
Mediator agent	init, potentialHazard

(Note that rem stands for 'remove', and init for 'initiation')

Table 24: Agent reasoning

Agents	Reasoning
Broker agent	Decision making about contracts
Transportation companies	Select the minimum priced drivers
Driver agent	Plan to achieve goals
Trucks agent	No reasoning
Mediator agent	Decision making about path interferences via constraint reasoning

Table 25 illustrates the agents, role, tasks, and methods used in the transportation domain.

Table 25: The agents, role, tasks, methods used in the transportation domain

Agent(s)	Task	Method
Brokers	Deliver tasks to companies Select minimum priced company	init, add(companies), rem(companies), get(companies), add(driver), rem(driver).
Companies	Deliver tasks to drivers Select minimum priced drivers	announce, addcity, init
Drivers	Compute the cost for tasks. Plan, control of driver agents	announce, init(Window), init(create the truck, controller)
Trucks		init, drive, move, route

5. Conclusions and future directions

5.1 Introduction

We have set out the research problem in section 1.2. Chapter 2's literature review on intelligent agents starts putting the pieces together towards understanding the current state of the art, but shows that some research has to be done to discuss agent interaction. Then chapters 3 and 4 describe the path to achieve the objectives. In this chapter, we summarise the end of chapter 2 and explain our perspective.

This thesis has been concerned with the problem of tools for modelling and simulation, and of the constraints at various levels of application. As described in chapter 1, the requirement for tools is important, since not only does the interaction between entities depend on various constraints but input-compute-output of 'safe' simulation has to be achieved. The approach presented in this thesis for investigating the behaviour and interactions of agents has been to use a combination of agent-based systems with constraint-based techniques to provide new insights to the modelling and simulation. The aim of this work has been to use improved understanding of agent behaviours and interactions to provide either a new technique for simulating interactions or to improve upon methods of interaction.

5.2 Conclusions about research questions or hypothesis

In this section, we discuss findings for each research question summarised from chapter 4 and explained within the context of this and prior research examined in chapter 2; for example, with which of the researchers discussed in chapter 2 does this research agree or disagree, and why? Disagreement suggests that the research is making a contribution to knowledge.

5.2.1 Comparisons of CANET with Fischer and Kuhn (1993)

In section 1.3.4, we have discussed Fischer and Kuhn's approach for a transportation application, and have identified various limitations. Now we will evaluate the CANET approach for such scenarios.

The contract net approach is a highly regulated and ordered society of agents that does not reflect reality. It is difficult to accommodate various preferences of agents. Within CANET, the structure is based on relations. Unlike the contract net approach in which the manager has too much responsibility, the CANET approach puts emphasis on satisfaction of constraints of agents.

Communication between agents tends to be message passing within the contract net approach. That has limitations in passing partial information between agents. The CANET approach, on the other hand, allows the agents to communicate partial information in an incremental manner.

Co-operation/co-ordination between agents is via contract net with Fischer and Kuhn (1993). That does not allow horizontal co-operation between trucks. We have simulated a truck scenario, in which we have demonstrated that agents can be able to co-operate by specifying constraints on goals.

Social laws are not addressed within Fischer and Kuhn's approach. Common laws within multi-agent settings for solving problems are not addressed. That would remove the necessity of hardcoding laws to different agents within the application. We have simulated a scenario in which we have introduced social laws as constraints.

Task decomposition and allocation is also via contract net within Fischer and Kuhn's approach. The manager always chooses the cheapest offer in selecting the agents. The approach is very hierarchical. We have demonstrated task decomposition and allocation as a constraint satisfaction problem.

Negotiation is addressed via contract net within Fischer and Kuhn's approach. Within the simulation, negotiation is not really discussed in the sense that the manager always selects the cheaper cost. There are no negotiations between the agents at the same levels. We have demonstrated negotiation as constraint relaxation based on either preference for maximal constraints, or sum of preference of agents.

Various constraints that are applicable such as precedence, resource, and temporal are not addressed by Fischer and Kuhn. We have shown that task scheduling can be

carried out by giving considerations to constraints such as precedence, resource, and temporal.

Learning and evolving of driver agents are not addressed. We have not carried out experiments to explore learning, or the evolution of agents. However, we believe that the constraint-based techniques can be used for learning and evolution.

The CANET approach is much easier to program for applications in the sense that the agency is programmed simply as constraints. There are no hard coded approaches to program various interactions. The CANET scripts are faster than Prolog's depth-first search mechanism, and an efficient constraint search approach is used for planning, and task allocation. The CANET approach provides an easier way to represent knowledge as constraints, and the programs are, in general, very short.

We have demonstrated that the existing agent framework, such as a contract net, can benefit from constraint passing in the sense that a traditional message passing mechanism is replaced by constraint passing. The findings in section 4 confirm this. The benefits of constraint-based approaches include no centralised control; knowledge is easy to express, and correctness is guaranteed theoretically.

Table 26: Comparisons between the CANET approach and Fischer and Kuhn (1993).

Properties	Fischer and Kuhn's (1993)	CANET approach
Communication	Message passing	Constraint passing
Co-operation, Co-ordination	Contract Net	Constraint propagation
Negotiation	Contract Net	Constraint relaxation
Task allocation	Contract Net	Constraint Satisfaction Problem
Social laws	None	Hard constraints

Table 26 shows the comparison between the proposed approach and Fisher and Kuhn (1993).

5.2.2 CANET 'Society of agents'

The CANET 'Society of agents' are based on the assumption that is "Everything is connected". The CANET agents are not organised in a hierarchical structure as in contract net but based on relations to achieve a decentralised framework. Agents communicate between each other not just via messages but also by constraints. That allows the agent to communicate partial information in an incremental manner. The behaviour of an agent may be constrained. The planning capability of an agent is seen as constraint search. Co-operation is seen as constraint passing. Negotiation is treated as constraint relaxation. External users may introduce certain social laws that cannot be relaxed. Belief, knowledge, desire, and the goal of an agent are all treated as constraints.

CANET agents are seen as fully committed to tasks. CANET agents are receptive to external stimuli for unexpected scenarios, can plan to works complex goals, co-operate and negotiate between each other, follow social laws. CANET agents may be extended to include emotion (Bates 1990), learning, and evolution. Concerning learning, the connection between neurons can be interpreted as constraints. Concerning evolution, there are technologies such as Simulated annealing, and Genetic algorithms within which cost function can be interpreted as constraints. Concerning emotion, various states such as 'happy' can be defined and treated as constraints.

To address the limitations of deliberate and non-deliberate architectures discussed in chapter 2, we have proposed the CANET approach that combines both reactive and planning layers. The CANET approach may be seen as a hybrid approach. However, the main criticisms of hybrid agents are that hybridism translates to ad hoc or unprincipled designs; hybrid architectures tend to be application specific; and theory that underpins hybrid system is not usually specified. The CANET approach is achieved by using constraint logic as a underlying theory for agent behaviours, and agent interactions.

In the DAI literature, various interactions are discussed such as co-operation, negotiation, task allocation, and so on. Traditional research tends to focus on either co-operation, or social law, and so on. Our view is that most of the interactions can be related to constraint-based mechanisms, and an agent performs various interactions at various circumstances. This facilitates in dealing with various interactions in the sense that a constraint-agent approach may be seen as an umbrella term, and is sufficient to simulate various interactions. The approach can be seen as unifying various existing interaction strategies. The unification in addition allows one to explore new areas such as self-organisation within an agent-constraint research map.

5.3 Conclusions about the research problem

In this section, the implications of our research towards further understanding of the research problem are explored. This section examines qualitative research that was not considered in the literature reviewed in chapter 2. The contribution of the research to the body of knowledge is clearly expressed. In this section, we include a summary listing of the contributions of the research together with justification. The contribution to parent disciplines, and other disciplines is outlined in the following section.

Our findings illustrate that the behaviour of an agent and the interactions between agents can be related to constraint-based mechanisms.

A summary list of the contributions of the research is as follows: First, we have provided a CANET (Constraint-based multi-Agent NETwork) approach for complex applications. Second, we have presented various societal notions, and proposed a constraint-based view of interactions. Third, we have extended the contract net approach by incorporating constraint passing within the framework. Fourth, we have presented in chapter 4 a series of experimental results to demonstrate the behaviour of an agent, and interactions can be related to constraint-based techniques for applications.

5.4 Implications for the theory

In this section, the full picture of the research findings within the body of knowledge is discussed. Theoretical implication of the research in its immediate field and wider body of knowledge is discussed.

Based on the theory, the model discussed in chapter 2, which is contract net, needs to be modified to be classified as a constraint-based contract net. Similarly, we argue various existing approaches can be converted to constraint-based ones.

The CANET methodology for analysing complex applications is very appealing. The approach is easy and can be applied widely. This CANET visualisation of modelling and simulation is appealing and the representation and reasoning of an agent and agent interactions are very efficient. The CANET approach allows one to simulate a society of agents.

A transportation scenario was simulated. Within the scenario, various agents such as broker, transportation companies, drivers, and trucks and so on, interact with each other by means of communication, co-operation, task allocation, and so on to deliver goods from one location to another. These interactions are based on constraint-based techniques. The advantage of this approach is that within this application, behaviours of an agent, such as scheduling, are related to constraint-based techniques.

Within the scenario, constraint communication was explored. This allows the agents to communicate not only just numbers but lists of numbers, partial information, programs, and so on. This agent interaction through a constraint store is also very appealing. The intention of an agent is seen as propagating constraints to the store.

Constraints are also used to co-ordinate the activities of various agents. This allows one to consider various constraints such as days, speed, and actions, planning and so on to achieve common goals. This approach is relevant to real-time applications. Agents in addition are required to express their preferences, dislikes and so on. The approach models real-world situations with consideration to hazards and so on.

Constraint co-operation was explored. The idea was that the interactions between various constraints filter out the solution. This can be applied in situations where agents or experts interact between each other their knowledge, resources, authority, and control.

5.5 Implications for further research

Notions such as learning, and emotion can be implemented in the CANET approach in the sense that those concepts can be related to constraint-based techniques.

We started the thesis by talking about the simulation of complex physical systems. We have highlighted the aspects of agency that might be simulated using a constraints-based approach. Such an approach is of considerable use at two levels.

First, a model itself is an implicit statement of how parts of a physical system work coherently with each other to manifest the phenomena they do. Therefore, the rules for selecting the input of such a model may be expressed, for example, as plans, as frames and so on.

Similarly, a constraint network can be used for a knowledge base that helps in the interpretation of the output of the simulation: a constraint network can not only be used for maintaining truth during the reasoning process but can also be used to guide a novice user. Roughly similar arguments can be applied to the discussion about the limits and scope of a given simulation model and how such things can be articulated using constraints.

Second, components of a large simulation model can be regarded as expressions of constraints within the real world. Again such constraints are implicitly described such as the interaction of a number of data sets, simulation engines and so forth. A constraints based approach can help in the articulation of the underlying dynamics of the model through the explication of the constraints that exist between large components of a given system.

As discussed in chapter 1.2, further research can be carried out to remove the limitations. Merge and de-merge of agents from the field of business can be explored.

This allows agents to merge and de-merge with the situations. An experiment was also carried out to see how the abilities of a society might be enhanced. A simulation of a society of trucks showed this. The method is appealing in that it can include new behaviours for agents to existing systems. This will continue as a research area for the evolution of an agent. Further, finite domain constraint examples are provided. There is a potential to incorporate research progress in the field of constraint satisfaction and DAI.

The proposed approach has implications in the parent field of DAI. Knowledge representation and reasoning in machines, theories of actions, search, planning, and pattern recognition can be evaluated to a constraint-agent framework.

There are other implications in related fields:

- Mathematics (differential equations, operational research problems can be treated as a Constraint Satisfaction Problems, and agents can be used to co-ordinate the computation of different topics);
- Physics (the conception of action, concept of constraint, distributed systems such as multi-particle motion, gases, liquids, and magnetism);
- Philosophy (agency in philosophy of mind, functioning of the universe);
- Sociology (social interactions such as co-operation, negotiation);
- Economics (group behaviour);
- Psychology (emotion, cognition, social structures, and processes).

A concrete example of where we wish to use an agent-based simulation that is based on constraints network is described.

The current research is part of the Safe-DIS project. The objective of the Safe-DIS project is to emphasise the role of Information Systems in making safety-related information available to design engineers in a timely and efficient manner.

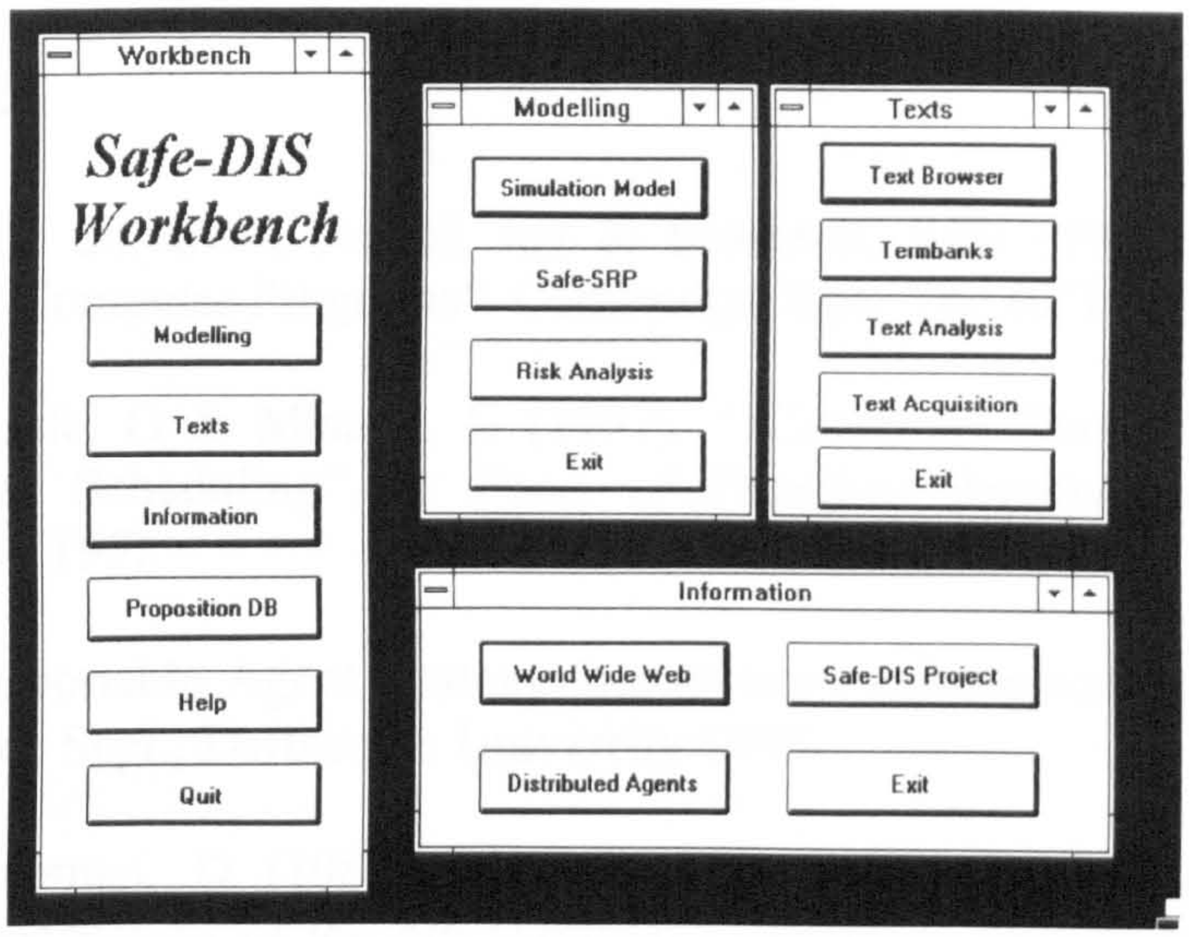

Figure 14: Safe-DIS workbench

The Safe-DIS workbench serves as a research development environment and as a prototype of the final system. The tool allows access to one of the major simulation networks in drainage, HYDROWORKS produced by Wallingford Software, and it supplements textual knowledge by providing access to problem-specific engineering knowledge within the corpus of texts that is relevant to the discipline.

The animation mechanisms have been developed under the Safe-DIS workbench (see Figure 14); the latest version of which is the front-end to a machine-assisted modelling facility, text analysis and management tools, with World Wide Web connections to various water and safety related resources, and termbanks.

This research into agents may be applicable to Safe-DIS in the light of the many diverse information sources which need to be managed during design, e.g. texts, rule-bases, modelling tools, etc. The agents are useful in analysing the inputs from diverse sources, in interpreting results, in helping the model building process, and so on. Agents may be used in monitoring the values of parameters such as Tank level, which then inform other agents, and/or trigger some actions. The potential candidate agents are input analysis agents, result interpretation agents, model building agents, model verification agents, textual agents and knowledge assistants.

REFERENCES

Abelson, Harold., Sussman, Gerald, Jay & Sussman, Julie (1985). "Structure and Interpretation of Computer Programs". Cambridge, MA: The MIT Press.

Adhikary, J., Hasle, G & Misund, G (1997). " Constraint Technology Applied to Forest Treatment Scheduling". In *Proc. of Practical Application of Constraint Technology, (PACT97).*

Agent Tcl, Transportable Agent System. *http://www.cs.dartmouth.edu/ agenttcl.html*, Computer Science Sept., Dartmouth University 1995.

Agre, P & Chapman, D (1987). "PENGI: An Implementation of a Theory of Activity". In *Proceedings of the 6th National Conference on Artificial Intelligence*, AAAI-87, Seattle, WA. pp. 268-272.

Agre, E, Philip (1995). "Computational Research on Interaction and Agency". In *Artificial Intelligence*, 72 (1-2).

Ahmad, Khurshid (1995). "A Knowledge-based Approach to the Safe Design of Distributed Networks". In (Eds.) Felix Redmill & Tom Anderson. *Achievement and Assurance of Safety (Proceedings of the Safety-critical Systems Symposium, Brighton 1995)*. London: Springer-Verlag Ltd. pp. 290-301.

Ahmad, Khurshid., Collingham, Steve., Salway, Andrew & Indrakumar, Selvaratnam (1995). "SAFE-DIS Briefing". Issue No. 3, Internal report.

Allen, James, F (1982). "Planning Using a Temporal World Model". In *International Joint Conferences on Artificial Intelligence*, IJCAI-83. pp. 741-747.

Allen, James, F (1984). "Towards a General Theory of Action and Time". *Artificial Intelligence*, vol. 23(2). pp. 123-154.

Allen, James, F & Kooman (1983). "Planning Using a Temporal Possible World Model". In *International Joint Conference on Artificial Intelligence*, IJCAI-83.

Allen, James, F & Perrault, C, R (1980). "Analysing Intention in Utterance". In *Artificial Intelligence.* Vol. 15. pp. 143-178.

Anderoli, J-M., and Pareshi, R.; and Schlichter, J, H. (1995). "Constraint agents for the information age". In *J. Universal Computer Science* 1(12): 762-789. Anderoli, J-M et al (1997).

Anderoli, J-M et al. (1997). "Constraints and Agents for a Decentralised Network Infrastructure". In *AAAI-97 Workshop on Constraints and Agents*, July 1997, Providence, Rhode Island.

Anderson, John, Eric (1995). "Constraint-directed Improvisation for Everyday Activities", *Ph.D. thesis*, Department of Computer Science, University of Manitoba.

Anke, Kay., Staudte, Rainer & Dilger, Werner (1997). "Producing and Improving Time Tables by Means of Constraint and Multi-agent Systems". In *AAAI-97 Workshop on Constraints and Agents*, July 1997, Providence, Rhode Island.

Aparicio, G (1996). "IBM Intelligent Agents". In *FIPA Opening Forum Proceedings*, Yorktown, New York, 1996.

Appelt, D, E (1985). "Planning English Sentence", Cambridge University Press, New York.

Appleby, S & Steward, S (1994). "Mobile Software Agents for Control in Telecommunications Networks". In *BT Technological Journal* 12 (2), pp. 104-113, April 1994.

Armstrong, Aaron & Durfee, Edmund (1997). "Dynamic prioritisation of complex agents in distributed constraint satisfaction problems". In *IJCAI-97 (Fifteenth International Joint Conference on Artificial Intelligence)*, 1997.

Atkinson, Will & Cunningham, Jim (1990). "Proving Properties of a Safety Critical System". In *Research Report* Doc. 90/28.

Avouris, N, M (1992). "User Interface Design for DAI applications: An Overview". In (Eds.) Avouris, N, M & Gasser, L. *Distributed Artificial Intelligence: Theory and Praxis*. pp. 141-161.

Axling, Tomas., Fahlen, Lenat & Haridi, Seif (1995). "Virtual Reality Programming in Oz". In *Proceedings of WOz'95*, International Workshop on Oz Programming (Nov. 29 - Dec. 2), Martigny, Switzerland. pp. 17-24.

Baker, A, D (1996). "Metaphor or Reality: A case study where agents bid with actual costs to schedule a factory". In (Eds.) Schott H., *Market-based Control*, Clearwater World Scientific, pp. 184-223, 1996.

Bamberger, K, Stefan (1997). "Co-operating diagnostic expert systems to solve complex diagnosis tasks". In (Eds.) Ras, W, Zbigniew., and Skowron, Andrzej, *Foundations of Intelligent Systems, ISMIS'97, Lecture Notes in Artificial Intelligence.*

Barbuceanu, Mihai (1997). "Co-ordinating agents by role based social constraints and conversation plans". In *AAAI-97 (Fourteenth National Conference on Artificial Intelligence), IAAI-97*, Providence, Rhode Island.

Barney, Pell., Douglas, E, Bernard & Steven, A, Chien (1996). "A Remote Agent Prototype for Spacecraft Autonomy". In *Proceedings of the SPIE Conference on Optical Science engineering, and Instrumentation, 1996.*

Barney, Pell., Douglas, E, Bernard & Steven, A, Chien (1997). " An Autonomous Spacecraft Agent Prototype". In *Proceedings of the First International Conference on Autonomous Agents, Marina del Rey, CA 1997.*

Bates, J (1994). 'The Role of Emotion in Believable Agents". In *Communications of the ACM.* Vol. 37(7). pp. 122-125.

Baujard, O., Pesty, S & Garbay, C (1994). "MAPS: A Language for Multi-agent Systems Design". In *Expert Systems.* Vol. 11(2).

Bellone, J., Chamard, A & Pradeless, C (1992). " PLANE : - An Evolutive Planning System for Aircraft Production". In *Proc. of Practical Application of Prolog (PAP92).*

Bird, S, S (1993). "Towards a Taxonomy of Multi-agent Systems". In *International Journal of Man Machine Studies.* Vol. 36. pp. 689-704.

Bond, Allen, H & Gasser, Les (1988). "An Analysis of Problems and Research in DAI". In (Eds.) *Readings in Distributed Artificial Intelligence.* Los Angles: Morgan Kaufmann.

Borning, Alan (1981). "The Programming Language Aspects of a ThingLab, a Constraint-Oriented Simulation Laboratory", *TOPLAS 3* (4), pp. 353-387, 1981.

Borning, Alan., Freeman-Benson, Bjorn & Wilson, Molly (1992). "Constraint Hierarchies, Lisp and Symbolic Computation". In *An International Journal*, 5, 223-270, 1992.

Borning, Alan., Lin, Richard & Marriot, Kim (1997). "Constraints for the web". In *Electronic proceedings*, November 8 - 14, 1997, Crowne Plaza Hotel, Seattle, USA.

Bradshaw, M, Jeffrey (1997). "An introduction in software agents". In (Eds.) Bradshaw, M, Jeffrey in Software Agents, The MIT Press, 1997.

Bratman, M, E., Israel, D, J & Pollack, M, E (1988). "Plans and Resource-bounded Practical Reasoning". In *Computational Intelligence.* Vol. (4). pp. 349-355.

Breitner, P. & Sadek, M.D (1996). " A rational agent as a kernel of a cooperative dialogue system: Implementating a logical theory of interaction". In *ECAI-96 Worshop on Agent Theories, Architectures, and languages.* Springer-Verlag, Heidelberg,Germany.

Brooks, Rodney, A (1986). "A Robust Layered Control System for a Mobile Robot". In *IEEE Journal of Robotics and Automation.* Vol. RA -2(1), pp. 14-23, 1986.

Brooks, Rodney, A (1991). "Intelligence without Representation". In *Artificial Intelligence*, Vol.(47). pp. 139-159.

Brustolini, Jose, C (1991). "Autonomous Agents: Characterisation and Requirements". In *Carnegie Mellon Technical Report* CMU-CS-91-204, Pittsburgh: Carnegie Mellon University, 1991.

Caglayan, A., Snorrason, M., Jacoby, J., Mazzu, J. & Jones, R., (1996). "Lessons from Open Sesame! A user interface learning agent". In *Proceedings the First International conference on the Practical Application of Intelligent Agents and Multi-agent technology* (PAAM 1996), London 22-24 April, pp. 61-74, 1996.

Cammarata, S., McArthur, D & Steeb, R (1983). "Strategies of Co-operation in Distributed Problem Solving". In *Proc. International Joint Conference on AI*, pp767-770.

Chaib-draa., & Moulin, P (1987). "Architecture for Distributed Artificial Intelligent Systems", *IEEE Proceedings*, Montreal, pp. 191-198, 1991.

Chaib-draa, B., Moulin, B., Mandiau, R & Millot, P. (1996). "Chapter 1- Trends in DAI", *Foundations of Distributed Artificial Intelligence*. In (Eds.) G.M.P.O'Hare and N.R. Jennings John Wiley & Sons Inc, pp. 3-55, 1996.

Chan, P., Heus, K & Weil, G (1998). " Nurse Scheduling with Global Constrainsta in CHIP: GYMNASTE". In *Proc of Practical Application of Constraint Technology (PACT98)*, London, UK.

Chavez, A & Maes, P (1996). "Kasbah: An agent marketplace for buying and selling goods". In *Proceedings the First International conference on the Practical Applications of Intelligent Agents and Multi-agent Technology* (PAAM 96), London, 22-24 April, pp. 75-90, 1996.

Chu, D (1993). "I.C. Prolog II: A Language for Implementing Multi-agent System". In S.M. Deen, editor, *Proceedings of the 1992 Workshop on Co-operating Knowledge-based Systems (CKBS- 92)*. pp. 61-74. DAKE Centre, University of Keele, UK, 1993.

Chess, David., Grosof, Benjamin., Harrison, Colin., Levine, David., Parris, Colin & Tsudik, Gene (1995). "Itinerant Agents for Mobile Computing". In *Technical Report* RC 2001, IBM T.J.Watson Research Center.

Coen, Nichael, D (1994). "Sodabot: A Software Agent Environment and Construction System". In (Eds.) Yannis Labrou and Tim Finn, *Proceedings of the CIKM Workshop on Intelligent Information Agents, Third International Conference on Information and Knowledge Management* (CIKM 1994), Gaithetsburg, Maryland.

Cohen, R, Philip & Levesque, J, Hector (1997). "Communicative actions for artificial agents". In (Eds.) Bradshaw, M, Jeffrey, Software Agents, The MIT press, 1997.

Collinot, Anne & Hayes-Roth, Barbara (1991). "Real-time performance on intelligent autonomous agents". In (Eds.) Werner, Eric, and Demezeau, *Yves in Decentralised AI3*, 1991.

Collis, Jaron.., Ndumu, Divine., Nwana, Hyancinth & Lee, Lyndon (1998). "The Zeus Agent Building Tool-Kit". In *BT Technology Journal* 16(3), July 1998, p.60-68.

Collis, Jaron & Lee, Lyndon (1998). "Building electronic marketplaces with the Zeus agent tool-kit". In *Proceedings of the Agent Mediated Electronic Trading (AMET) Workshop*, May 1998, p17-32.

Crabtree, Barry & Rhodes, Brad (1998). "Wearable computing and the remembrance agent".In *BT Technology Journal* 16(3), July 1998, p118-124.

Creemers, T., Giralt, L.R., Riera, J., Ferrarons, C., Rocca, J & Corbella, X (1995). "Constraint-Based Maintenance Scheduling on a Electric Power-Distribution Network". In *Proc. of Practical Application of Prolog (PAP95)*, Paris, France.

Cucchiara, R., Lamma, E., Mello, P & Milano, M (1997). "An interactive constraint-based system for selective attention in visual system". In (Eds.) Breka, Gerhard., Habel, Christopher, and Nebel, Bernhard in *KI 97*: Advances in Artificial Intelligence, *21st Annual German Conference on Artificial Intelligence*, Freiburg, Germany, 1997.

Dechter, Rina (1992). "Constraint Networks". In *the Encyclopaedia of Artificial Intelligence*, second edition, Wiley & Sons, pp. 276-285.

Decker, Keith (1996). "TAEMS: A framework for environment centred analysis and design of co-ordination mechanisms". In *foundations of Distributed Artificial Intelligence*, Edited by G.M.P.O' Hare and Jennings, N, R, 1996, John Wiley & Sons, Inc.

Decker, Keith., Sycara, Katia & Williamson, Mike (1997). "Middle-agents for the Internet, *IJCAI-97"*. In *Fifteenth International Joint Conference on Artificial Intelligence*, 1997.

Dincbas, M & Simonis, H (1991). " APACHE – A Constraint Based, Automated Stand Allocation Systems". In *Proc. of Advanced Software Technology in Air transport (ASTAIR91), London, UK.*

Douglas, E, Bernard (1998). " Design of the Spacecraft Autonomy". In *Proceedings of IEEE Aerospace Conference, Snomass, CO, 1998.*

Douglas, E, Bernard & Barney, Pell (1997). " Designed for Autonomy: Remote Agent for the New Millenium Program". In *Preceedings of the Fourth International Symposium on Artificial Intelligence, Robotics, and Automation for Space (I-SAIRAS), 1997.*

Durfee, Edmund, H (1992). "What Your Computer Really Needs to Know, You Learned in Kindergarten". In *Proceedings of the National Conference on Artificial Intelligence.* AAAI-92.

Durfee, E. H., Lesser, V, R & Corkill, D (1987). "Coherent Co-operation among Communicating Problem Solvers". In *IEEE Transactions on Computers* C-36(11), pp.1275-1291, 1987.

Durfee, Edmund H., Lesser, Victor R & Corhill, Daniel D. (1992). "Distributed Problem Solving". In (Eds.) Stuart C. Shapiro *The Encyclopaedia of Artificial Intelligence* (Part 1: A-C). New York: John-Wiley Interscience. pp 379-388.

Durfee, E. H & Montgomery, T, A.,(1989). "MICE: A Flexible Testbed for Intelligent Co-ordinatrion Experiments". In *Proceedings of the 1989 Distributed Artificial Intelligence Workshop,* pp.25-40,1989.

Eaton, Peggy., Freuder, Eugene & Wallace, Richard (1997). "Constraint-based agents: Assistance, co-operation, compromise". In *Internal report*, Constraint computation centre, University of New Hampshire.

Edmonds, Ernest, A., Candy, Linda., Joseph, Rachel & Soufi, Bassel (1994). "Support for Collaborative Design: Agents and Emergence". In *Communications of the ACM.* Vol. 37. pp. 41-47.

Eisinger, N., Elshshiewy, N & Pareshi, R (1991). "Distributed Artificial Intelligence - An Overview". In *Technical Report* ECRC- 91.

Epharti, Eithan & Rosenchein, Jeffrey, S (1991). "The Clarke Tax as a Consensus Mechanism among Automated Agents". In *Proceedings of the National Conference on Artificial Intelligence,* AAAI- 91.

Epharti, Eithan & Rosenchein, Jeffrey, S (1992). "Constrained Intelligent Action: Planning under the Influence of a Master Agent", In *Proceedings of the national Conference on Artificial Intelligence.* AAAI-92.

Esquirol, Patrick., Fargier, Helene., Lopez, Pierre & Schieux, Thomas (1996). "Constraint Programming". In *Belgian Journal of Operational Research, Statistics, and Computer Science,* 1996.

Fennel, R, D & Lesser, V, R (1977). "Parallelism in Artificial Intelligence Problem Solving : A Case Study of Hearsay II".In *IEEE Transactions on Computers.* Vol. 26(2).

Ferber, Jacques (1996). " Reactive Distributed Artificial Intelligence: Principles and Applications". In (Eds.) O'Hare, G.M.P. & Jennings, N.R, *Foundations of Distributed Artificial Intelligence,* John Wiley & Sons, 1996.

Ferber, Jacques & Drogoul, Alexis (1992). "Using Reactive Multi-Agent Systems in Simulation and Problem Solving". In (Eds.) Avouris, N, M & Gasser, L. *Distributed Artificial Intelligence: Theory and Praxis,* 1992.

Feigenbaum, E, A., McCorduck, P & Nii, H. P (1988). "The Rise of the Expert Company", Times Books.

Ferguson, I, A (1992). "TouringMachines : An Architecture for Dynamic, Rational, Mobile Agents", *Ph.D. Thesis,* Clare Hall, University of Cambridge, UK.

Finin, T & Fritzon, R (1994). "KQML - A Language and Protocol for Knowledge and Information Exchange". In *Proceedings of the 13th International Distributed Artificial Intelligence Workshop*, Seattle, WA. pp. 127-136.

Finnin, Tim., Labrou, Yannis & Mayfield, James (1997). "KQML as an agent communication language". In (Eds.) Bradshaw, M, Jeffrey in Software Agents, The MIT press, 1997.

Fikes, R, E & Nilsson, N (1971). "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving". In *Artificial Intelligence.* Vol. 5(2).

Fischer, Klaus & Kuhn, Norbert (1993). "A DAI Approach to Modelling the Transportation Domain". In *Research Report* RR-93-25.

Fischer, Michael (1994). "Representing and Executing Agent-Based Systems". In (Eds.) Wooldridge & Jennings. *ECAI 94 (European Conference on Artificial Intelligence)*. Berlin: Springer-Verlag Ltd. pp. 307-323.

Fischer, Michael (1994). "A Survey of Concurrent METATEM - The language and its applications". In (Eds.) Gabbay et al., *Temporal Logic - Proceedings of the First International Conference* (LNAI Volume 827), pp. 480-505, Springer-Verlag.

Focacci, F., Lamma, E., Mello, P & Milano, M (1997). " Constraint Logic Programming for the Crew Rostering Problem". In *Proc. of Practical Application of Constraint Technology (PRACT97),* London, UK.

Fox, S, Mark (1994). ISIS: "A retrospective". In (Eds.) Morgan, B, Michael in Intelligent Scheduling, Morgan Kaufman Publishers, Inc, pp.3-29, 1994.

Franklin, Stan & Graesser, Art (1996). "Is it an agent, or just a program?: A Taxonomy for autonomous agents". In *Proceedings of the Third International Workshop on Agent theories, Architectures, and Languages*, Springer-verlag.

Freeman-Benson, Bjorn, N (1993). " Converting an existing user interface to use constraints". In *Proceeding of UIST 93*, Atlanta, Georgia, November 1993.

Freuder, E. C & Mackworth, A. K. (Eds.) (1994). "Constraint based reasoning". Cambridge, MA: MIT Press.

Freuder, Eugene, & Eaton, Peggy, S (1997). "Compromise Strategies for Constraints Agents". In *AAAI-97 Workshop on Constraints and Agents*, July 1997, Providence, Rhode Island.

Freuder, Eugene & Wallace, Richard, J. (1997). "Suggestion strategies for constraint-based matchmaker agents". In *AAAI-97 Workshop on Constraints and Agents*, July 1997, Providence, Rhode Island.

Fruhwirth, Thom., Harold, Alexander., Kuchenhoff, Volker., Provost, Thierry., Lim, Pierre., Monfroy, Eric & Wallace, Mark (1992). "Constraint Logic Programming : An informal introduction". In *Internal Report* ECRC-92-6i.

Gallaire, H (1985). "Logic Programming : Further developments". In *IEEE Symposium on Logic Programming*. pp. 88-99, IEEE, Boston, July 1985.

Galliers, J. R. (1988). "A strategic framework for multi-agent cooperative dialogue". In *Proceedings of the Eighth European Conference on Artificial Intelligence* (ECAI-88), pp. 415-420, 1988.

Galliers, J. R. (1988). "A Theoretical Framework for Computer Models of Cooperative Dialogue, Acknowledging Multi-Agent Conflict", *PhD thesis*, Open University.

Gallimore, J, R., Jennings, R, N., Lamba, S, H., Mason, L, C & Orenstein, J, B (1999). "Co-operating Agents for 3D Scientific Data Interpretation". In *IEEE Trans. On Systems, Man and Cybernetics*, Part C. 29 (1), p. 110-126.

Gasser, Les (1992). "An Overview of DAI". In (Eds.) Avouris, N, M & Gasser, Les. *Distributed Artificial Intelligence: Theory and Praxis*. pp. 10-24.

Gasser, Les & Huhns, Michael, N (1989). *Distributed Artificial Intelligence*, Volume II, San Mateo, California: Morgan Kaufmann Publishers, Inc.

Gasser, L., Rosenchein, J, S., & Ephrati, E (1995). "Introduction to Multi-agent Systems". In *Tutorial A Presented at the 1st International conference on Multi-agent Systems*, San Franscisco, CA, June 1995.

Geneserth, Michel, R., Ginsburg, Matthew, L & Rosenchein, Jeffrey, S (1988). "Co-operation without Communication". In *Proceedings of the National Conference on Artificial intelligence*, AAAI-86. pp. 561-567.

Genesereth, M, R & Ketckpel, S, P. (1994). "Software Agents", *Communications of the ACM* 37(7), pp. 48-53, 1994.

Georgeoff, Michael (1983). "Communication and Interaction in Multi-agent Planning". In *Proceedings of the National Conference on Artificial Intelligence.* AAAI-83. pp.125-129.

Georgeoff, Michael (1988). "A Theory of Action for Multi-agent Planning". In Bond and Gasser (1988).

Georgeoff, Michael & Lansky, Amy (1987). "Reactive Reasoning and Planning". In *Proceedings of the National Conference on Artificial Intelligence*, AAAI-87.

Ghanea-Hercock, Robert., Collis, Jaron & Ndumu, Divine (1999). "Co-operating mobile agents for distributed parallel processing". In *3rd Int. Conference on Autonomous Agents*, Seattle, May 1999.

Gray, Robert, S (1995). "Agent Tcl: A Transportable Agent System". In *Internal Report*, Department of Computer Science, Dartmouth College, Hanover, New Hampshire.

Guha, R, V & Lenat, D, B (1990). "CYC: A Mid Term Report", *AI Magazine*, pp32-59, 11 (3).

Gupta, Vineet, Jagadeesam, Radha, and Saraswat, A, Vijay (1996). "Computing with Continuous Change". In *Science of Computer Programming*, 1996.

Gupta, Vineet; Jagadeesam, Radha, and Saraswat, A, Vijay (1997). "Probabilistic Concurrent Constraint Programming". In *Proceedings of CONCUR 97*, edited by Mazurkiewicz and Winkowski, Sporinger Verlag 1997.

Gupta, Vineet., Jagadeesam, Radha., Saraswat, A, Vijay, and Bobrow, G, Danny (1994). "Programming in Hybrid Constraint Languages". In *Hybrid Systems Workshop*, Cornell, October 1994. Hybrid Systems II, LNCS 999, Springer Verlag.

Gupta, Vineet., Saraswat, A, Vijay and Struss, Peter (1995). "Modelling a Photocopier Paper Path". In *Proceedings of the Second IJCAI Workshop on Engineering Problems for Qualitative Reasoning*, Montreal, August 1995.

Gupta, Vineet & Struss, Peter (1995). "Modelling a Copier Paper Path : A Case Study in Modelling Transportation Process". In *Proceedings of the Ninth Qualitative Reasoning Workshop*, Amsterdam, May 1995.

Hanschke, Philipp (1993). "A declarative integration of terminological, constraint-based, data-driven, and goal-directed reasoning", *Research report*, RR-93-46.

Haridi, S & Janson, S (1990). "Kernel andorra prolog and its computation model". In *Proc. International Conference on Logic Programming*, pages 31-46. MIT Press, 1990.

Hatvany, J (1984). "Intelligence and Co-operation in Heterarchical Manufacturing Systems". In *The 16th CIRP International Seminar on Manufacturing Systems Proceedings*, Tokyo, Japan, pp.1-4, 1984.

Hayes-Roth, F (1980). "Towards a Framework for Distributed AI". In *SIFART Newsletter* pp 51-52.

Henz, Martin & Wurtz, Jorg (1995). "Using Oz for College Timetabling". In *Proceedings of the 1995 International Conference on the Practice and Theory of Automated Timetabling*, Edinburgh, Scotland, 20 August-1 September.

Henz, Martin & Muller, Martin (1994). "Programming in Oz". In *DFKI Oz documentation series*, German Research Center for Artificial Intelligence.

Henz, Martin., Muller, Martin & Wolf, Markus (1995). "A Shell for Distributed Multi-User Games". In *Proceedings of WOz'95*, International Workshop on Oz Programming (Nov. 29 - Dec. 2), Martigny, Switzerland. pp. 63-64.

Henz, Martin., Smolka, Gert & Wuertz, Jorg (1993). "Oz - A Programming Language for Multi-agent Systems". In *International Joint Conference on Artificial Intelligence,* IJCAI-1993, Chambery, France, 28 August - 3 September). pp. 404-409.

Henz, Martin., Smolka, Gert & Wuertz, Jorg (1995). "Object-oriented Concurrent Constraint Programming in Oz". In (Eds.) V.Saraswat & P. van Hentenryck, *Principles and Practice of Constraint Programming,* chapter 2, pages 29-48. Cambridge: MIT press.

Hermenegildo, M and the CLIP group. (1994). "Some Methodical Issues in the Design of CIAO - A Generic parallel Concurrent Constraint System". In *Principles and Practice of Constraint Programming,* LNCS 874, pp. 123-133, Springer-Verlag, May 1994.

Hermans, L & Sclimmer, J (1993). "A Machine Learning Apprentice for the Completion of Repetitive Forms". In *Proceedings of The 9th IEEE Conference on Artificial Intelligence Applications*, Orlando, Florida: IEEE Press, pp. 164-170, 1993.

Hewitt, C (1977). "Viewing Control Structures as a Pattern of Passing Messages". In *Artificial Intelligence.* Vol. 8(3). pp. 323-364.

Huhns, M, N (1988). In (Eds.) Distributed Artificial Intelligence. Vol I. London: Pitman Publishing.

Huhns, M, N. & Singh, M, P., (1994). "Distributed Artificial Intelligence for Information Systems". In *CKBS-94 Tutorial*, University of Keele, UK, June 1994.

IBM Aglets, (1996). "IBM Aglets: Programming Mobile Agents in Java, A White Paper", *http://www.ibm.co.jk/trl/aglets/whitepaper.htm*, IBM Tokyo Research Laboratory, 1996.

Ishida, Toru (1997). "Parallel, distributed and multi-agent production systems". In *Lecture notes in Artificial Intelligence.*

Jary, David & Jary, Julia (1991). Collins Dictionary of Sociology, Harper Collins Publishers.

Jaffar, J & Lassez, J, L. (1987). "Constraint Logic Programming", In *Proceedings of the ACM Symposium on Principles of Programming Languages*, ACM, 1987.

Jaffar, J., Michaylov, S., Stukey, P & Yap, R (1992). "The CLP® Language and System". In *ACM Transactions on Programming Languages and Systems*, 1992.

Jennings, N, R (1995). "Agent Software". In *Proc. UNICOM Seminar on Agent Software*, London, UK, pp. 12-27, 1995.

Jennings, N, R (1995). "Co-ordination techniques for Distributed AI". In (Eds.) G.M.P. O'Hare and N. R. Jennings, *Foundations of Distributed Artificial Intelligence*, John Wiley & Sons, pp. 187-210, 1995.

Jennings, N, R., Corera, J M., Laresgoiti, L., Mamdani, E., Perriollat, F., Skarek, P & Varga, L(1995). "Using ARCHON to Develop Real-World DAI Applications for Electricity Transportation and Particle Accelerator Control". In *IEEE Expert*, Special Issue on Real World Applications of DAI systems.

Jennings, N, R & Mamdani, E, M (1992). "Using Joint Responsibility to Coordinate Collaborative Problem Solving in Dynamic Environments". In *Proceedings of the National Conference on Artificial Intelligence.* AAAI-92.

Jennings, Nicholas, R., Sycara, Katia & Wooldridge, Michael, (1998). "A Roadmap of Agent Research and Development". In *Autonomous Agents and Multi-agent Systems*, 1, pp. 275-306, Kluwer Academic Publishers, Boston, 1998.

Johansen, B.S & Hasle, G. " Well Activity Scheduling – An application of Constraint Reasoning". In *Proc. of Practical Applications of Constraint Technology* (PACT97), London, UK.

Keller, Richard., Rimon, Michal & Das, Aseem (1994). "A Knowledge-Based Prototyping Environment for Scientific Modelling Software". In *Automated Software Engineering*, Vol. 1 (No. 1, March 1994). pp. 79-128.

Kiss, George (1996). "Agent Dynamics", *Foundations of Distributed Artificial Intelligence.* In (Eds.) G.M.P.O.'Hare and N.R. Jennings, Hohn Wiley & Sons, 1996.

Kraus, Sarit (1997). "Negotiation and co-operation in multi-agent environments". In *Artificial Intelligence* 94 (1997), p 79-97, Elsevier B. V.

Kraus, Sarit & Wilkenfield, Jonathan (1991). "The Function of Time in Cooperative Negotiations". In *Proceedings of the National Conference on Artificial Intelligence.* AAAI-91.

Kuchenoff, Volker (1991). "Novel Search Techniques". In *ECRC report.*

Kuchenhoff, Volker (1993). "Novel Search and Constraints - An Integration". In *Working paper* ECRC-CORE-93-9 (European Computer-Industry Research Centre).

Kuipers, Benjamin (1994). "Qualitative reasoning : modelling and simulation with incomplete knowledge". In *Artificial Intelligence*, MIT press (1994).

Kumar, Vipin (1992). "Algorithms for Constraint Satisfaction Problems : A Survey". In *AI Magazine* 13 (1): 32-44, 1992.

Labrou, Yannis & Finnin, Tim. (1997). "Semantics and conversations for an agent communication language". In *IJCAI-97* (Fifteenth International Joint Conference on Artificial Intelligence), 1997.

Langlotz, C, P et al. (1987). "A Therapy Planning Architecture That Combines Decision Theory and Artificial Intelligence Techniques". In *Computers and Biomedical Research.* Vol. 20 . pp. 279-303.

Languenou, Eric., Benhamou, Frederic., Goualard, Frederic & Christie, Marc (1998). "The Virtual Cameraman: an Interval Constraint-based Approach". *In Proceedings of the Workshop "Constraint Techniques for Artistic Applications"*. August 1998, Brighton.

Lee, Lyndon., Nwana, Hyancinth., Ndumu, Divine & De Wilde, Philip (1998). "The stability, scalability, and performance of multi-agent systems". In *BTTJ* 16(3), July 1998, p94-103.

Lekkas, G & Liedekerke, M, Van (1992). "Prototyping Multi-agent Systems: A Case Study". In (Eds.) Avoris, N, M & Gasser, Les. *Distributed Artificial Intelligence: Theory and* praxis. pp. 129-140.

Lenat, D, B (1975). "BEINGS: Knowledge as Interacting Experts". In *Proceedings of International Joint Conference on Artificial Intelligence.* IJCAI-75. Morgan Kauffmann.

Lenat, D, B., and Feigenbaum, E, A (1991). "On the Thresholds of Knowledge". In *Artificial Intelligence*, 47, pp 185-220.

Lesser, Victor R. & Corkill, Daniel G (1983). "Remote Agent: To Boldly Go Where No AI System Has Gone Before". *The Distributed Vehicle Monitoring Testbed: A Tool for Investigating Distributed Problem Solving Networks.* AI Magazine 4(3): 15-33 (1983)

Levitt, R., Cohen, P., Kunz, J., Nass, C., Christiansen, T & Jin, Y (1994). "The Virtual Design Team: Simulating How Organisational Structure and communication Tools affect Team Performance". In (Eds.) Carley, K. & Prietula, M, *Computational Organisation Theory*, San Francisco: Lawrence Erlbaum, pp. 67-91, 1994.

Lingrad, Andy, R (1992). "A Comparison of Temporal Reasoning Strategies Used in Planning". In *Technical report*, ARL/02, Department of Computing, Imperial College of Science, Technology and Medicine.

Mackworth, A, K (1986). "Constraint Satisfaction". In *Encyclopaedia of Artificial Intelligence*, 1986.

Maes, P (1991). "The Agent Network Architecture (ANA)". In *SIGART Bulletin.* Vol. 2(4). pp. 115-120.

Maes, P (1994). "Agents that Reduce Work and Information Overload". In *Communications of the ACM* 37(7), pp. 31-40, 1994.

Maes, P. (1995): "Artificial Life meets Entertainment: Interacting with Lifelike Autonomous Agents". In Special Issue *on New Horizons of Commercial and Industrial AI, Communications of the ACM*, Vol. 38, No. 11, 1995.

Malone, T, W (1988). "Modeling Coordination in Organizations and Markets". In Bond and Gasser (1988).

Marriott, Kim & Stuckey, J, Peter (1998). "Programming with constraints", MIT Press, 1998.

Mauichi, Takeo., Ichikawa. Masaki & Tokoro, Mario (1991). "Modeling agents and their groups". In (Eds.) Werner, Eric, and Demazeau, Yves, Decentralised AI3, 1991.

Mayfield, James., Labrou, Yannis & Finnin, Tim (1997). "Evaluation of KQML as an agent communication language". In (Eds.) Wooldridge, Michael., Muller, P, Jorg, and Tambe, *Milind in Intelligent Agents II, IJCAI'95 Workshop (ATAL),* Montreal, 1995.

McAllester, David (1992). "Constraint Satisfaction Search". In *Lecture Notes* for 6.824, Artificial Intelligence, 1992.

McCarthy, John (1986). "Applications of Circumscription to Formalizing Common-sense Knowledge". In *Artificial Intelligence.* Vol. 28, pp. 89-116.

Mc.Cabe, F, G & Clark, K, L (1994). "APRIL - Agent process interaction language". In Wooldridge & Jennings (1994).

Meutzelfeldt et al. (1986). "ECO- An Intelligent front end for ecological modelling". In *Society for Computer Simulation.* Simulation Series 18(1). pp. 67-70.

Minsky, Marvin & Reicken, Doug (1994). "A Conversation with Marvin Minsky about Agents". In *Communications of the ACM.* pp. 22-30.

Mitchell, T., Caruana, R., Freitag, D., mcDermott, J. & Zabowski, D. (1994). "Experience with a Learning Personnel Assistant", *Communications of the ACM* 37 (7), pp. 81-91, 1994.

Montanari, U (1974). "Network of Constraints : Fundamental properties and applications to picture processing". In *Information Science,* 7(2):95-132, 1974.

Montanari, U & Rossi (1996). "Constraint solving and programming: What Next?". In *ACM Computing Surveys* 28 A (4), December 1996.

Moore, R (1985). "Semantical Considerations on Non-monotonic Logic". In *AI Journal.* Vol. 25. pp. 75-94.

Muller, J, P & Pischel, M (1993). "The agent architecture INTERRAP: concept and application". In *Technical Research Report* RR-93-26, German Artificial Centre for Artificial Intelligence, Saabrucken.

Muller, J, P., Pischel, M & Thiel, Michael (1994). "Modeling a Reactive Behaviours in Vertically Layered Agent Architectures". In (Eds.) Wooldridge and Jennings. *ECAI-94 (European Conference on Artificial Intelligence).* Berlin: Springer-Verlag Ltd. pp. 261-276.

Muller, Martin., Muller, Tobias & Van Roy, Peter (1995). "Multiparadigm programming in Oz". In (Eds.) Donald Smith, Olivier, Ridoux, and Peter Van Roy, "Visions for the Future of logic Programming: Laying the foundations for a Modern

Successor of Prolog", *A workshop in association with ILPS"95*, December 7, Portland, Oregon.

Muller, Tobias and Wurtz, Jorg (1996). "Interfacing propagators with a concurrent constraint language". In *JICSL Post-conference Workshop and compulog Net Meeting on Parallelism and Implementation Technology for (Constraint) Logic Programming Languages*, 1996.

Muscettola, Nicola., Nayak, P, Pandurang., Pell, Barney & Williams, Brian (1998). "Artificial Intelligence"103(1-2):5-48, August 1998.

Musliner, David & Boddy, Mark (1997). "Contract-based distributed scheduling for distributed processing". In *AAAI-97 Workshop on Constraints and Agents*, July 1997, Providence, Rhode Island.

Murthy, Seshahayee et al. (1997). "Agent-based co-operative scheduling", *AAAI-97 Workshop on Constraints and Agents*, July 1997, Providence, Rhode Island.

Nadoli, Gajanana & Beigel, John, E (1993). "Intelligent manufacturing-simulation agents (IMSAT)". In *ACM Transactions on Modelling and Computer Simulation.* Vol. 3(1). pp. 42-65.

Nayak, P, Pandurang (1999). "Validating the DS1 Remote Agent Experiment". In *Proceedings of the 5th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, iSAIRAS-99.

Neches, R., Fikes, R., Finin, T., Gruber, T., Patil, R., Senator, T & Swartout, T (1991). "Enabling Technology for Knowledge Sharing". In *AI Magazine*, pp36-56.

Nareyek, Alexander (1997). "Constraint-based Agents". In *AAAI-97 Workshop on Constraints and Agents*, July 1997, Providence, Rhode Island.

Ndumu, Divine., Collis, Jaron, and Nwana, Hyancinth (1998). "Towards desktop personal agents". In *BT Technology Journal* 16(3), July 1998, p.69-78.

Ndumu, Divine & Nwana, Hyancinth (1997). "Research and development challenges for agent-based systems". In *IEE Proceedings on Software Engineering*, 1997, Volume 144, No. 01, January 1997.

Ndumu, Divine., Nwana, Hyancinth., Lee, Lyndon & Collis, Jaron (1999). "Visualising and debugging distributed multi-agent systems". In *3rd int. Conference on Autonomous Agents*, Seattle, May 1999.

Ndumu, Divine., Nwana, Hyancinth., Lee, Lyndon & Haynes, Hayden (1999). "Visualisation of distributed multi-agent systems". In *Applied Artificial Intelligence Journal*, Vol. 13 (1), 1999, p187-208.

Ndumu, Divine, and Tah, Joseph (1998). "Agents in computer assisted collaborative design". In *Artificial Intelligence in Structural Engineering* (LNAI No. 1454), May 1998, p249-270.

Nwana, H, S., (1993). "Simulating a Children's Playground in ABLE". In *Working Report*, Department of Computer Science, Keele university, UK, 1993.

Nwana, S, Hyacinth (1996). "Software Agents : An Overview". In *Knowledge Engineering Review*, Vol. 11, No. 3, pp.205-244, October-November 1996.

Nwana (1998). "Zeus: A Collaborative Agents Tool Kit". In Abridged version of *Nwana et al. 1997*, submitted to *Autonomous Agents 1998*, Minneapolis/St. Paul, USA.

Nwana, H, S., Ndumu, D, T., & Lee, L, C (1997). "An Advanced Tool-Kit for Engineering Distributed Multi-Agent Systems". In *Consideration to the AA'98 Conference.*

Nwana, Hyancinth & Ndumu, Divine (1999). "A perspective on software agents research". In *Knowledge Engineering Review*, 1999.

Nwana, Hyancinth, Ndumu, Divine & Lee, Lyndon (1998). "ZEUS: An advanced tool-kit for engineering distributed multi-agent systems". In *Proceedings of PAAM'98*, London, March 98, p377-392.

Nwana, Hyancinth., Ndumu, Divine., Lee, Lyndon & Collis, Jaron (1999). "ZEUS: A tool-kit for building distributed multi-agent systems". In *Applied Artificial Intelligence Journal*, Vol 13 (1), 1999, p. 187-208.

Nwana, Hyancinth., Rosenchein, Sandholm., Sierra, Maes & Guttman (1998). "Agent-mediated electronic commerce: issues, challenges, and some viewpoints". In *Proceedings of the Agents'98*, Minneapolis, May 1998, p189-196.

O'Brien, P. & Wiegend, M (1996). "Agents of Change in Business Process Management". In *British telecommunications Technology Journal* 14(4), pp. 1-24, Oct 1996.

Obrst, Leo(1997). "Constraints and Agents in MADE-smart". In *AAAI-97 Workshop on Constraints and Agents*, July 1997, Providence, Rhode Island.

Odyssey, Odyssey Frequently Asked Questions,
http://www.genmagic.com/agents/odyssey-faq.html, General Magic Inc., 1997.

Ossowski, Sascha., Garcia-Serrano, Ana & Cuena, Jose (1998). "From theory to practice in multi-agent system design: The case of structural co-operation". In (Eds.) Herzog, Otthein, and Gunter, Andreas in KI-98, *Advances in Artificial Intelligence. 22nd Annual German Conference on Artificial Intelligence,* Bremen, Germany, 1998.

Pan, Jeff, Y-C & Tenenbaum, Jay, M (1992). "Towards an intelligent agent framework for Enterprise Integration". In *Knowledge Representation*, KR-92.

Pape, Le, Claude (1994). "Scheduling as intelligent control of decision making and constraint propagation". In (Eds.) Morgan, B, Michael in Intelligent Scheduling, Morgan Kaufman Publishers, Inc, pp. 67-99, 1994.

Paredis, Jan (1994). "Co-evolutionary constraint Satisfaction", In *Proceedings of the Third Conference on Parallel Problem Solving from Nature* (PSPN 94), Lecture Notes in computer Science, vol. 866. In (Eds.) Davidor, Y., Schwefel, H-P., Manner, R. , Springer Verlag.

Parunak, H. Van Dyke (1996). "Case Grammar: A Linguistic Tool for Engineering Agent-based Systems", http://www.iti.org/¬van, 1996.

Parunak, H, Van Dyke. (1996). "Applications of Distributed Artificial Intelligence in Industry". In (Eds.) G, M, P, O'Hare and N. R. Jennings , *Foundations of Distributed Artificial Intelligence,* John Wiley & Sons, pp. 139-163, 1996.

Parunak, Van et al. (1997). "Distributed component centred design as agent-based distributed constraint optimisation". In *AAAI-97 Workshop on Constraints and Agents*, July 1997, Providence, Rhode Island.

Patridge, D (1987). "The Scope and limitations of Future Generation Expert Systems". In *Future Generations Computer Systems*, Vol3.1, pp1-10.

Pell, Barney., Bernard, E, Douglas., Chien, A, Steven., Gat, Erran., Muscettalo, Nicola., Nayak, P, Pandurak., Wagner, D, Michael & Williams, C, Brian (1998). Autonomous Robots 5(1), 1998.

Pesant, Gilles (1995). "Une Approache Geometrique aux Contraintes Arithmetiques Quadratiques en Programmation Logique avec Contraintes", *Ph.D. thesis*, Department d'Informatique et de Recherche Operationnelle, Universite de Montreal.

Petrie, Charles., Jeon, Heecheol & Cutkosky, Mark (1997), "Combining Constraint Propagation and Backtracking for Distributed Engineering". In *AAAI 97 Workshop on Constraint and Agents*, 1997.

Platzner, Marco., Rinner, Bernhard & Weiss, Reinhold (1995). "Exploiting paralleleism in constraint satisfaction for qualitative simulation". In *Journal of Universal Computer Science*, vol. 1, no. 12 (1995). p81-1120.

Plaunt, Christian., Rajan, Kanna., Pell, Barney & Muscettola, Nicola (1998). "Integrated planning and execution for satellite tele-communications". In *AAAI 1998 Fall Symposium.*

Poggi (1994). "DAISY: An Object-oriented System for Distributed Artificial intelligence. In (Eds.) Wooldridge et al (1994), *Intelligent Agents- Proceedings of the 1994 Workshop on Agent Theories, Architectures and Languages.*

Pont, M.J & Moreole, E(1996). "Towards a Practical Methodology for Agent Oriented Softwarte Engineering with C++ and Java". In *Technical Report 96-33*, Department of Leicester University, Dec 1996.

Poole. D (1988). "A Logical Framework for Default Reasoning". In *AI Journal.* Vol. 36. pp. 27-47.

Popov, Konstantin (1995). "An Exercise in Concurrent Object-oriented Programming: the Oz Browser". In *Proceedings of WOz'95, International Workshop on Oz Programming(Nov. 29 - Dec. 2),* Martigny, Switzerland. pp. 101-108.

Pountain, Dick (1995). "Constraint Logic Programming". In *BYTE Magazine,* McGraw-Hill, Inc, New York, NY.

Power, June (1990). "Towards a Framework for Self-organising Distributed Systems". In *Research Note* RN/90/34.

Prigogine, Ilya & Stengers, Isabelle (1985). "Order Out of Chaos", Flamingo.

Reiter, R (1980). "A Logic for Default Reasoning". In *AI Journal.* Vol. 13. pp. 81-132.

Rich, Elaine & Knight, Kevin (1991). Artificial Intelligence, McGraw-Hill, 2nd edition.

Rittmann (1991). "Die Macht der Trucks". In *Bild der Wissenschaft,* 9, pp. 112-114, 1991.

Rosenchein, J, S & Geneserth, M, R (1992). "Deals among rational Agents". In *International Joint Conference on Artificial intelligence,* IJCAI-92. pp. 91-99.

Rosinus, Michael., Muller, Jorg, P & Pischel, Markus (1995). "An Agent Specification Language".In *Proceedings of WOz'95, International Workshop on Oz Programming(Nov. 29 - Dec. 2),* Martigny, Switzerland. pp. 35-44.

Roth. Al (1993). "Constraint Programming: A Practical Solution to Complex Problems". In *AI Expert.* pp. 36-37.

Round, Alfred (1989). "Knowledge-based Simulation". In (Eds) Avon Barr, Paul, R, Cohen and Edward A. Feigenbaum, *The Handbook of Artificial Intelligence* - Vol. IV. (Chap XXII). Reading, MASS : Addisson Wesley Pub. Co. pp. 416-518.

Russell, Stuart, J., & Norvig, Peter (1995). "Artificial Intelligence: A Modern Approach", Englewood, Cliffs, NJ: Prentice Hall, 1995.

Sabin, D., Sabin, M & Russell, R (1995). "A Constraint-based Approach to Diagnosing Software Problems in Computer Networks". In *Proceedings of the First International Conference on Principles and Practice of Constraint Programming,* CP'95, U.Montanari, ed. Springer-Verlag.

Sabin, Mihaela et al.(1997). "Automated construction of constraint-based diagnosticians". In *AAAI-97 Workshop on Constraints and Agents,* July 1997, Providence, Rhode Island..

Sandholm, Tuomas & Lesser, Victor (1997). "Issues in automated negotiation and electronic commerce: extending the contract net framework". In *AAAI-97 Workshop on Constraints and Agents*, July 1997, Providence, Rhode Island.

Saraswat, A, Vijay (1992). "Concurrent Constraint Programming: A Brief Survey". XEROX Parc Report.

Saraswat, A, Vijay (1993), "Concurrent Constraint Programming", MIT Press, 1993.

Saraswat, A, Vijay., Jagadeesam, Tadha, and Gupta, Vineet (1994). "Foundations of Timed Concurrent Constraint Programming". *Proceedings of the Ninth Annual IEEE Sym on Logic in Computer Science*, Paris, July 1994.

Saraswat, A, Vijay; Jagadeesam, Tadha, and Gupta, Vineet (1994). "Programming in a Timed Concurremt Constraint Languages. Constraint Programming", Edited by Mayoh, E Tougu, and J Penjam, Springer Verlag, 1994.

Saraswat, A, Vijay., Jagadeesam, Tadha & Gupta, Vineet (1995). "Timed Default Cocurrent Constraint Programming". In *Journal of Symbolic Computation, 1996. Extended abstract published in Proc of the 22nd An. ACM SIGPLAN SIGACT Sym. on the Principles of Prog. Lang.*, San Francisco, Jan 1995.

Saraswat, A, Vijay & Patrick Lincoln (1992). "Higher-order, linear, Concurrent Constraint Programming". In *Technical Report*, Xerox PARC, 1992.

Saraswat, A, Vijay., Rinaed, Martin & Panangaden, Prakash (1991). "Semantic Foundations of Concurrent Constraint Programming". In *Proc. Of the 18th Ann. ACM-SIGPLAN-SIGACT Sym. On the Principles of Prog. Lang.*, Orlando, January 1991.

Schmeier, Sven & Schupeta, Achim (1995). "PASHA - Personal Assistant for Scheduling Appointments". *Proceedings of WOz'95*, International Workshop on Oz Programming(Nov. 29 - Dec. 2), Martigny, Switzerland. pp. 59-64.

Schulte, Christian (1994). "Open Programming in DFKI Oz". In *DFKI Documentation Series*, German Research Centre for Artificial Intelligence(DFKI), Saarbrucken, Germany.

Schulte, Christian & Smolka, Gert (1994). "Encapsulated Search for Higher-order Concurrent Constraint Programming". In (Eds.) M. Brynooghe. Logic Programming, *Proceedings of the 1994 International Symposium*. New York: The MIT press.

Schulte, Christian (1995). "An Oz Search Debugger". In *Proceedings of WOz'95*, International Workshop on Oz Programming (Nov. 29 - Dec. 2), Martigny, Switzerland. pp. 109-115.

Schulte, Christian., Smolka, Gert & Wurtz, Jorj (1998). "Finite Domain Constraint Programming in Oz : A Tutorial". In *DFKI OZ Documentation Series*, February 12, 1998.

Schupeta, Achim (1992). "Main Topics of DAI: A Review". In DFKI German Research Centre for Artificial Intelligence(DFKI), Saarbrucken, Germany.

Selig, L (1987). "An Expert System Using Numerical Simulation and Optimisation to Find Particle Beam Line Errors". In *Second Workshop on AI and Simulation. AAAI conference*, Seattle.

Seliger, G., Viehweger, B., Wieneke-Toutouai, B & Kommana, S, R. (1987). "Knowledge-based Simulation of Flexible Manufacturing Systems". In *Proceedings of the Second European Simulation Multiconference*, Vienna, Austria pp. 65-68.

Selvaratnam, Indrakumar (1993). "A Constraint-based Approach to Multi-agent Planning and Scheduling", *M. Sc. Dissertation.* Dept. of Computer Science, Imperial College of Science, Technology and Medicine, London, UK.

Selvaratnam, Indrakumar & Ahmad, Khurshid (1995). "Multi-agent Systems in Modelling and Simulation". In *Proceedings of WOz'95*, International Workshop on Oz Programming(Nov. 29 - Dec. 2), Martigny, Switzerland. pp. 1-15.

Selvaratnam, Indrakumar & Ahmad, Khurshid (1998). "Multi-agent Systems in Modelling and Simulation". In *Computing Sciences Report*, University of Surrey, CS-98-02..

Sheth, N. & Maes, P (1993). "Evolving Agents for Personalised Information Filtering". In *Proceedings of the 9th Conference on Artificial Intelligence for Applications*, pp. 113-121, 1993.

Shevestov, I et al (1997). "Technology of active objects". In *AAAI-97 Workshop on Constraints and Agents*, July 1997, Providence, Rhode Island.

Shoham, Yoav (1997). "An overview of agent-oriented programming". In (Eds.) Bradshaw, M, Jeffrey in Software Agents, The MIT press, 1997.

Shoham, Y (1990). "Agent-oriented Programming". In *Technical Report* STAN-CS-1335-90, Computer Science Department, Stanford University, Stanford, CA 94305.

Shoham, Y (1993). "Agent-oriented Programming". In *Artificial Intelligence.* Vol. 60 (1). pp. 51-92.

Shoham, Y & Tennenholtz, M (1992). "On the synthesis of useful social laws for AI societies". In *Proceedings of the National Conference on Artificial Intelligence.* AAAI-92. pp. 276-281.

Shoham, Yoav and Tennenholtz, Moshe (1997). "On the emergence of social conventions: modelling, analysis, and simulations". In *Artificial Intelligence 94* (1997), p 139-166.

Sloman, Aaron (1996). "What sort of architecture is required for a human-like agent?". In *Cognitive Modelling Workshop*, AAAI-96, Portland, Oregon, Aug. 1996.

Smith, D, C., Cypher, A & Spohrer, J., (1994). "KidSim: Programming Agents Without a Programming Language". In *Communications of the ACM*, 37 (7), pp. 55-67, 1994.

Smith, Reid, G (1977). "The Contract Net: A Formalism for the Control of Distributed Problem Solving". In *International Joint Conference on Artificial Intelligence*, IJCAI.

Smolka, Gert (1993). "Residuation and Guarded Rules for Constraint Logic Programming". In (Eds.) F. Benhamou & A. Colmerauer. *Constraint Logic Programming,* MIT press, pp. 405 - 419.

Smolka, Gert (1995). "The Oz Programming Model", In Jan Van Leeuwan, editor, *Computer Sciences Today*, number 1000 in LNCS, pp. 324-343, Springer-Verlag, Berlin, 1995.

Smolka, Gert & Treinen, Ralf (1995). Editors in *DFKI Documentation Series*, German Research Centre for Artificial Intelligence(DFKI), Saarbrucken, Germany.

Soltysiak, Stuart & Crabtree, Barry (1998). "Knowing me, knowing you: Practical issues in the personalisation of agent technology". In *Proceedings of PAAM'98*, London, March 1998, p377-392.

Steele, G. L. (1980). "The Definition and Implementation of a Computer Programming Language Based on Constraints", PhD thesis, MIT

Steels, L. (1985). "Second Generation Expert Systems". In *Future Generation Computer Systems*, Vol. 1, 4, pp.213-221.

Steels, L (1990). "Co-operation between Distributed Agents through Self-organisation". In (Eds.) Demazeau, Y & Muller, J, P., *Decentralised AI - Proceedings of the 1st MAAMAW.*,Amsterdam: Elsevier, pp.175-196, 1990.

Steiner, D., Mahling, D & Haugeneder, H (1990). "Human Computer-supported Cooperative Work". In *Proceedings of the 10th International Workshop on Distributed Artificial Intelligence.* MCC Technical Report Nr. ACT-AI-335-90.

Stone, Peter & Veloso, Manuela (1997). "Multi-agent Systems: A Survey from a Machine Learning Perspective". In *Under Review for Journal Publication*, February 1997.

(Sun 1994). "The Java language: A White Paper". In *Sun Microsystems White Paper*, Sun Microsystems.

Sutherland, I (1963). "Sketchpad: a Man-Machine Graphical Communication System. In *Proceedings of the IFIP Spring Joint Computer Conference*, 1963.

Sycara, K, P (1988). "Resolving Goal Conflicts via Negotiation". In *Proceedings of the 7th National Conference on Artificial Intelligence*, St. Paul, Minesota. pp. 245-250.

Sycara, K, P (1989). " Multi-agent Comprise via Negotiation". In (Eds) Gasser and Huhns. Distributed Artificial intelligence. San Mateo, California: Morgan Kaufmann. Vol. II. pp. 119-137.

Tambe, M. (1996). "Teamwork in real-world, dynamic environments". In *International conference on multi-agent systems (ICMAS96)* .

Tambe, Milind (1997). "Agent architectures for flexible, practical teamwork". In *AAAI-97 (Fourteenth National Conference on Artificial Intelligence), IAAI-97,* Providence, Rhode Island.

Tambe, Milind (1997). "Implementing Agent Teams in Dynamic Multi-agent Environmnts", Extended version of "Teamwork in real-world dynamic environments". In *Proceedings of the International Conference on Multi-agent Systems,* (ICMAS), December 1996.

Telescript, Telescript Technology: The Foundation for the Electronic Marketplace, *http://www.genmagic.com/Telescript/Whitepapers/wpl/ whitepaper-1.html*, General Magic Inc., 1996.

Thomas, S, R (1993). "PLACA- An Agent Oriented Programming Language", *Ph.D. Thesis*, Computer Science Department, Stanford University, Stanford, CA 94305.

Tsang, E (1993). Foundations of constraint satisfaction. London: Academic Press.

Tulloch, Sara (1993). The Reader's Digest Oxford Wordfinder, Ckarendon Press, Oxford.

Van Hentenryck, Pascal &_Saraswat V.A.(1996) "Strategic Directions in Constraint Programming". In *Computing Surveys* 28(4): 701-726.

Voyager, Voyager Technical Review, *http://www.objectspace.com/voyager/voyager_white_papers.html*, ObjectSpace Inc., 1997.

Waltz, D.L. (1972). "Generating Semantic Descriptions from Drawings of Scenes with Shadows", Ph. D. Thesis, Department of Electrical Engineering, Massechusetts Institute of Technology.

Waltz, D, L (1975). "Understanding Line Drawings of Scenes with Shadows. In P. Winston, editor, *The Psychology of Computer Vision*, McGraw-Hill, 1975.

Wavish, P. & Graham, M. (1994), "Roles, Skills and Behaviour". In (Eds.) Wooldridge, M. & Jennings, N. (1995b), *Intelligent Agents*, Lecture Notes in Artificial Intelligence **890**, Heidelberg: Springer Verlag, 371-386.

Weerasooriya, D., Rao, A & Ramamohanara, K (1994). "Design of Concurrent Agent-oriented Language". In (Eds.) Wooldridge & Jennings (1994). Intelligent

Agents - *Proceedings of the 1994 Workshop of Agent Theories, Architectures and Languages.*

Werner, Eric (1991). "The design of multi-agent systems". In (Eds.) Werner, Eric, and Demezeau, Yves in Decentralised AI3, 1991.

Werner, E (1988). "Cooperating Agents: A Unified Theory of Communication and Social Structure". In Gasser, Les & Huhns, Michael, N (1987).

White, James, E. (1994) "Telescript technology: The Foundation for the Electronic Marketplace". *General Magic White Paper*, General Magic, 1994.

Winston , Patrick, Henry (1992). Artificial Intelligence. Addison-Wesley, 3rd edition.

Wooldridge, Michael, J & Jennings, Nicholas, R (1994). "Intelligent Agents-Theories, Architectures, and Languages: A Survey". In (Eds.) Wooldridge & Jennings. *ECAI 94* (*European Conference on Artificial Intelligence*). Berlin: Springer-Verlag Ltd. pp. 1-39.

Wooldridge, Michael., & Jennings, Nicholas, R (1995). "Agent Theories, Architectures, and Languages: a Survey". In (Eds.)Wooldridge and Jennings , *Intelligent Agents*, Berlin: Springer-Verlag, pp. 1-22.

Wurtz, Jorg (1997). "Oz Scheduler: A Workbench for Scheduling Problems". In *Proceedings of the 6th IEEE International Conference on Tools with Artificial Intelligence*, Nov 16-19, 1996, IEEE Computer Society Press.

Yang, Q (1992). "A Theory of Conflict Resolution in Planning". In *Artificial Intelligence.* Vol. 58.

Yonezeva (1990). (Eds.) ABCL - "An Object-oriented Concurrent System". The MIT Press.

Yumi, Iwasaki., Farquhar, Adam., Bobrow, G, Daniel & Saraswat, A, Vijay (1995). "Modeling Time in a Hybrid Systems: How Fast is Instaneous?". Inp *The Proceedings of IJCAI 1995*, Montreal, August 1995.

Figures and tables