Report of Dagstuhl Seminar 02481
Programming Multi Agent Systems based on Logic

[$25/11/02 \rightarrow 29/11/02$**]**

Juergen Dix[†] Michael Fisher[‡] Yingqian Zhang[†]

† Department of Computer Science, University of Manchester, U.K.
‡ Department of Computer Science, University of Liverpool, U.K.

6th December 2002

1 Nature and importance of the subject

Multi-agent systems are set to be the key technology for software organisation during the next decade. While there have already been a number of multi-agent systems developed, the programming technology available for constructing such systems is relatively immature. Hence, there is a need for a powerful, general purpose programming technology for multi-agent systems.

The intention of this seminar is to bring together the leading researchers in these areas and to foster interaction between the various groups and thus get a better understanding of the ways in which multi-agent systems may be programmed in the future. As well as targeting logical approaches, a key element is to consider the requirements for efficient systems scaling within real world applications.

Over many years, work on computational logic has spawned research areas such as knowledge representation (KR), nonmonotonic reasoning (NMR), automated deduction (AR), and deductive databases (DDB). Each of these can be seen as an essential component within multi-agent systems, as agents need to

- *describe* the world (KR),
- *reason* somehow about how the world behaves (AR),
- *decide* in the light of uncertain information (NMR), and
- *deal* with massive data stored in heterogeneous formats (DDB).

In parallel, work within the multi-agent systems community has involved developing, often via logic, concepts concerned with communication languages and distributed computation (CC), cooperation and teamwork (TW), and the dynamic development of agent organisations (ORG). Again, each of these aspects can be seen as being required in complex multi-agent systems, as agents need to *communicate* with other distributed agents (CC), *cooperate* with other agents in order to achieve some goal (TW), and *evolve*, dynamically, organisational structures appropriate to the particular situation (ORG).

2 Goals of the Seminar

The seminar was set up in a way to allow ample time for discussions. We restricted the presentations to 30-35 minutes and allowed 10-15 minutes time for discussion after each presentation. This concept allowed for four talks in the morning and two talks after lunch.

We also set up four working groups: (1) Programming negotiation in agents, (2) Programming deliberation/rationality in agents, (3) Information/Data management via logic-based agents, (4) Programming cooperation in agents. Participants had been allocated to these groups three weeks before the seminar started. Each working group was chaired by two senior researchers[1] who contacted the participants and distributed material before the seminar. The groups met on Monday and Tuesday from 4-6 pm.

The idea behind these working groups was:

1. to identify key exemplars/problems that are relevant to that area;
2. to describe these exemplars/problems concisely/abstractly (can some of them be used as benchmarks/prototypical examples to check particular frameworks against?); and
3. to find out if, and to what extent, logic-based programming of multi-agent systems is useful for solving these problems.

Results were presented on Thursday, where all participants met from 4-6 pm.

An ambitious outcome that we aimed for was

> *A set of challenge problems/exemplars for logic-based programming of multi-agent systems. In addition, some criteria to determine whether a logic-based approach is useful or not. Or a list of problems where other methods are superior.*

[1] With the exception of Working Group 4, where one of the co-chairs dropped out at the last moment.

3 Outcomes of the Seminar

A homepage for the seminar has been set up, at `http://www.cs.man.ac.uk/~zhangy/dagstuhl`, containing all the presentations, the results of the working groups, and, last but not least, some photos of our official excursion: a wine tasting in Riol. As can be seen from the programme of presentations available on that web site, the seminar contained a wide variety of high-quality talks. Many participants commented on the excellent programme.

The working group idea generally worked well, with the groups often meeting outside their scheduled times. While the overall goal of the groups was perhaps too ambitious (after just two meetings), some interesting results have already emerged. We are currently trying to get the groups to continue their work (and, indeed, most seem keen) and hope that something useful and publishable will come out of it.

Following interactions during the seminar, it was decided to propose a new workshop on *Languages, Tools and Techniques for Programming Multi-Agent Systems* for AAMAS 2003 in Melbourne, Australia. This event is the most important conference on agent-based systems and is held annually. Over 12 seminar participants are now involved in the programme committee for this proposed workshop, and the time at Dagstuhl allowed us to work together on the application.

It has also been decided by several participants of the seminar to set up a steering committee for organising and continuing the CLIMA workshop series (*Computational Logic in Multi-Agent Systems*), which is closely related to the topic of the seminar.

Another important outcome of the seminar was to develop the details of a special issue of *Annals of Mathematics and Artificial Intelligence* on the topic of "Logic-Based Agent Implementation". Again, interactions at the seminar led to the publication of the *call for papers* for this initiative; see `http://www.csc.liv.ac.uk/~michael/LBAI03`.

4 Schedule

Monday 25th November	
08:45 - 09:00	*Juergen Dix* and *Michael Fisher* Welcome and Introduction
09:00 - 10:30 Chair: *Renate Schmidt*	*Joris Hulstijn* Multi-Agent Interaction Protocols and Dialogue Games *Paolo Torroni* A Logic-based Approach to Negotiation Dialogues
10:45 - 12:15 Chair: *John-Jules Meyer*	*João Alexandre Leite* Languages of Updates *Chiara Ghidini* Programming Individual Rational Agents
14:00 - 15:30 Chair: *Ulrich Hustadt*	*Amal El Fallah Seghrouchni* CLAIM: Computational Language for Autonomous, Intelligent and Mobile Agent *Michael Schroeder* Arguments and Misunderstandings: A Fuzzy Approach to Conflict Resolution in Open Systems
16:00 - 18:00	Working Group Discussion
Tuesday 26th November	
09:00 - 10:30 Chair: *Juergen Dix*	*Michael Fink* Answer Set Programming for Information Agents *Yingqian Zhang* Monitoring Agents
10:45 - 12:15 Chair: *Thomas Eiter*	*Wenjin Lue* Adversarial Planning in Multiagent Systems based on Graphplan *Cees Witteveen* A Resource Logic for Multi-Agent Plan Merging
14:00 - 15:30 Chair: *James Harland*	*Viviana Mascardi* A Survey and Discussion of Logic-Based Languages to Model and Program Intelligent Agents *Maurizio Martelli* CaseLP: A Prototyping Environment for Heterogeneous Agent Systems
16:00 - 18:00	Working Group Discussion

Wednesday 27th November	
09:00 - 10:30 Chair: *Cees Witteveen*	*Benjamin Hirsch* Organising Logic Based Agents *Renate A. Schmidt* Agent Dynamic Logic
10:45 - 12:15 Chair: *Chiara Ghidini*	*John-Jules Meyer* Agent programming in Dribble: from beliefs to goals with plans *Mehdi Dastani* Programming the Deliberation Cycle of Cognitive Agencies
14:00 15:30 - 22:00	Photo Session (in front of chapel) Excursion (Wine tasting)
Thursday 28th November	
09:00 - 10:30 Chair: *Katsumi Inoue*	*Ullrich Hustadt* Scientific Benchmarking for Agent-Method Logics *Ken Satoh* Speculative Computation in Multi-Agent System
10:45 - 12:15 Chair: *Chiaki Sakama*	*Frieder Stolzenburg* Specification and Analysis of Multi Agent Systems *Oliver Obst* Diagnosis in Simulated Soccer
14:00 - 15:30 Chair: *Ken Satoh*	*Katsumi Inoue* Speculative Computation by Consequence Finding *Chiaki Sakama* Default reasoning in multi-agent systems
16:00 - 18:00 Chairs: *M. Fisher* and *J. Dix*	Results of the working groups
Friday 29th November	
09:00 - 10:30 Chair: *João Leite*	*James Harland* Agents via Mixed Mode Computation in Linear Logic *Emil Weydert* Agents and Uncertainty - Induction of preferences
10:45 - 12:15 Chair: *Michael Fisher*	*Guido Boella* Normative Multi Agent Systems *Rafael Bordini* Progress in BDI Logic Programming with AgentSpeak(L)

Collected Abstracts

Multi-Agent Interaction Protocols and Dialogue Games

by *Joris Hulstijn*

In an open and dynamic environment like the Internet, interaction protocols need to be defined and reasoned about in a more flexible way. This paper proposes a methodology for the design and verification of flexible interaction protocols for agent communication. The method combines dialogue games, which constitute rules that indicate if a sequence of dialogue acts can be called *coherent*, with an extended version of Veltman's update semantics, in which the semantic content of each dialogue act is interpreted as an update of a particular information state. We define updates for the semantic content of questions and answers and of proposals, counterproposals and suggestions. Formal coherence relations are presented for dialogue games of cooperative information exchange and a version of negotiation.

A Logic-based Approach to Negotiation Dialogues

by *Paolo Torroni*

Negotiation for multi-agents is a way to reach an agreement on topics of common interest. Dialogue is a way to perform negotiation among two parties. We present a framework for agent dialogue and negotiation, based on Abductive Logic Programming (ALP).

The framework is based on an existing architecture for logic-based agents, and extends it by accommodating dialogues for negotiation. In particular, the agent knowledge representation is mapped into an abductive logic program. The framework comes together with an execution model, which includes an agent life cycle la Kowalski-Sadri, combining reactivity with rationality, and abductive reasoning capabilities in the agents. We present the framework in the context of resource reallocation.

In this presentation, we outline the agent knowledge representation, the negotiation language, the protocols followed by agents that engage in a dialogue, the concept of policies expressed by dialogue constraint, and we show a correspondence between this framework and ALP. We present some examples of agents, characterized by having different policies.

We define the notion of sequences of dialogues for solving a resource reallocation problem, and finally we introduce the idea of agents with multiple negotiation policies, associated with the notion of negotiation stages. As agents step through a sequence of stages they will have to do more 'thinking' and disclose more about themselves. If one stage fails to produce a deal amongst the agents, they may agree to move to the next stage where there is a better chance of a mutually agreeable deal.

We conclude by giving some formal results about the negotiation framework. Such results may apply in the general case of abductive agents or in the specific case of some classes of agent systems. They include: ability of the agents to produce dialogues, conformance of policies to protocols, termination, duration, and convergence of the negotiation process, ability to solve resource reallocation problems, classes of problems that can be solved by specific policies, and subsumption of policies.

Languages of Updates

by *João Alexandre Leite*

Logic programming has often been considered less than adequate for modelling the dynamics of knowledge changing over time. In this talk we take a guided tour of recent developments towards overcoming such drawback. Starting with Dynamic Logic Programming (DLP), we go through the Language of Dynamic Updates (LUPS), and the Knowledge And Behaviour Update Language (KABUL), to finally reach a simple though quite powerful approach to modelling the updates of knowledge bases expressed by generalized logic programs, by means of a new language, christened EVOLP (after EVOlving Logic Programs). The approach was first sparked by a critical analysis of previous efforts and results in this direction, and aims to provide a simpler, and at once more general, formulation of logic program updating, which runs closer to traditional logic programming (LP) doctrine. From the syntactical point of view, evolving programs are just generalized logic programs (i.e. normal LPs plus default negation also in rule heads), extended with (possibly nested) assertions, whether in heads or bodies of rules. From the semantics viewpoint, a model-theoretic characterization is offered of the possible evolutions of such programs. These evolutions arise both from self (or internal) updating, and from external updating too, originating in the environment. This formulation sets evolving programs on a firm basis in which to express, implement, and reason about dynamic knowledge bases, and opens up a number of interesting research topics with great applicability to Multi-Agent Systems.

Programming Individual Rational Agents

by *Chiara Ghidini*

In this talk we introduce a logical model of rational agency incorporating the key notions of ability, (bounded) belief, and confidence, the last of these capturing a flexible motivational attitude. Since the logical basis we propose is relatively simple, formal descriptions are amenable to both direct execution and formal verification. In addition, we provide two examples characterising aspects of a rational agent that uses the variability of resource bounds, and the motivational attitude of confidence in order to act in a sophisticated way.

CLAIM: A Computational Language for Autonomous, Intelligent and Mobile Agents

by *Amal El Fallah-Seghrouchni* and *Alexandru Suna*

This talks proposes a language called CLAIM as a Computational Language for Autonomous Intelligent and Mobile agents.

CLAIM allows to design Multi-Agent Systems (MAS) that support both stationary and mobile agents. Agents designed thanks to CLAIM are endowed with cognitive capabilities (e.g. reasoning), are able to communicate with others (send and receive several kinds of messages) and are mobile.

The primitives of mobility are inspired from the ambient calculus.

The language CLAIM is supported by a multi-platform system (SyMPA) compliant with the standard MASIF (OMG specifications); i.e. agents can be distributed over several platforms and can move from one to another with respect to MASIF specifications.

This talks presents the main features of our language CLAIM and describes the most significant aspects of SyMPA implementation (e.g. the protocols of migration). It goes on to highlight the expressiveness of our language and discusses its main properties (completeness in particular).

Arguments and Misunderstandings: A Fuzzy Approach to Conflict Resolution in Open Systems

by *Michael Schroeder and Ralf Schweimeier*

Expressive knowledge representation will be an important feature of the semantic web. Facts as represented in a table of a relational database or by RDF can be extended along three axes:

1. Rules & amp; deduction can be added: the very core of RuleML.
2. Negation can be added to express positive and negative knowledge.
3. Fuzziness can be added to express uncertainty.

How can we reason with these extensions? We will use argumentation as an elegant mechanism to define the semantics of rules, negation, and fuzziness. Arguments attack each other and an argument is acceptable if it can be defended against any attack. Depending on which kinds of attack and defence we allow a different notion of justified arguments results. We will present a framework to relate these notions and show which ones are identical and which ones relate to other approaches such as put forward by Dung, Prakken and Sartor, and WFSX by Alferes, Damasio, and Pereira. Next we show how to interpret fuzziness in the light of negation and define an appropriate semantics.

With such expressive knowledge representation at hand, can we apply and use it on the semantic web? The semantic web is open and the use of different ontologies will lead to misunderstandings, since concept names and predicate arity etc. may mismatch. To this end, we use our approach and embed fuzzy unification into it. Fuzzy unification unifies any atoms, but scores their similarity. This enables us to reason in the light of missing and mismatching terms and predicate names.

Answer Set Programming for Information Agents

by *Thomas Eiter* and *Michael Fink*

Today, the search for specific information on the World Wide Web faces several problems, which arise on the one hand from the vast number of information sources available, and on the other hand from their intrinsic heterogeneity, since standards are missing. A promising approach for solving the complex problems emerging in this context is the use of multi-agent systems of information agents, which cooperatively solve advanced information-retrieval problems. This requests for providing advanced capabilities, such as search and assessment of information sources, query planning, information merging and fusion, dealing with incomplete information, handling of inconsistency, and so on.

In this talk, our focus is on the role which answer set programming techniques can play in the realization of reasoning capabilities for information agents. In particular, we are interested to see in how they can be used, extended, and further developed for the needs of this domain of application. As an example task of information agents we consider the selection of relevant information sources in automated query answering and we present an approach for information-site selection, which is based on the answer set programming paradigm. We report experimental results obtained in a particular application domain, discuss advantages and drawbacks, and point out possible extensions and open issues.

Monitoring Agents

by *Yingqian Zhang, Juergen Dix*
joint work with *Thomas Eiter, Michael Fink* and *Axel Polleres*

There are many challenges in multi-agent system. Given a task, it is difficult for agent designers to model/specify a ”proper” multi-agent system. Moreover, there are lots of uncertainties in MAS due to the dynamic environment, coordination and communication among agents. If a single agent fails to act on a predefined action, it can bring the whole system disruption. Thus, it is a critical task to monitor the real-time behavior of MAS. In our approach, we build a MAS from a viewpoint of system domain, that is, to convert the modelling problem to the planning problem. We use a planning system to, first, specify the collaboration of agents in a MAS; then, verify the actual behaviors of the agents.

The planning system we are using is "action language K", which can formulate a planning problem based on the given task. We consider monitoring a Gofish MAS as an example task. According to the plan formulation, a Gofish Post Office MAS has been developed to simulate the package delivery service. Furthermore, a monitor agent has been modelled, which exploits DLV^K system to obtain a set of plan. Checking the action consistency, this monitor agent can detect the communication states between Gofish agents.

Adversarial Planning in Multiagent Systems based on Graphplan

by *Wenjin Lue*

We study the problem of adversarial planning in the context of multi-agent system environments. In this context we present GamePlan, an algorithm that takes into account the open and unpredictable nature of such environments. Unpredictability manifests itself here as follows: what an agent assumes to hold at one stage of its interaction with the environment may change as a consequence of the unpredictable, and possibly adversarial, actions of other agents in that environment; such actions take place at later stages of the interaction. Gameplan is based on Graphplan, a general purpose and efficient planner for STRIPS domains, where a plan is a compact structure represented as a graph specifying the flow of properties holding as a result of various actions taking place in the environment. Like Graphplan, Gameplan has the property that useful information for constraining search can quickly be propagated through the graph as it is being built. Unlike Graphplan, however, Gameplan constructs the planning graph by labelling each action level with the agent who will eventually execute this action, while the solution extraction is adapted to one that can be handled by a conditional planer to deal with the adversarial and uncertain behaviours of other agents in the environment.

A Resource Logic for Multi-Agent Plan Merging

by *Mathijs de Weerdt and Cees Witteveen*

We discuss a logic-based framework for investigating multi agent plan merging. Using a multi-sorted language, we distinguish resource predicates and constraint predicates. Fully instantiated resource predicates are used to express simple resource facts. Constraint predicates are used to connect resource predicates and to state conditions that have to be fulfilled in order to use resource facts. Goals are also expressed using resource and constraint predicates. An agent has to produce certain resources satisfying goal expressions, given some set of initial resources. To this end an agent is able to use a set of primitive actions? Actions are schemata that consume and produce resource facts. Plan schemes are partially ordered sets of actions. Plans are plan schemes where the resources consumed (inputs) and resources produced (outputs) are fully instantiated. Using a transition systems semantics we are able to prove that plans behave like actions. We define

a plan reduction process as a process where actions are removed from an existing plan without affecting goal realizability. Plan reduction can be modelled as a serial refinement process that removes actions without affecting the partial order relation between the remaining set of actions. Plan merging in our framework comes down to serial refinement while taking into account the privacy of the plans of the parties involved. An application area where this multi-agent plan merging idea can be applied is the improvement of ride plans in transportation.

A Survey and Discussion of Logic-Based Specification Languages for Intelligent Software Agents

by *Viviana Mascardi*

Agent-Oriented Software Engineering (AOSE) is the research field aiming at finding abstractions, languages, methodologies and toolkits for modeling, verifying, validating and prototyping complex applications conceptualized as Multi-Agent Systems (MASs). A very lively research area studies how formal methods can be used for AOSE; the ARPEGGIO open framework belongs to this area and aims at developing an open framework where logic-based formal specification languages and logic-based executable languages can be integrated to provide the means for specifying and prototyping a MAS choosing the most suitable language for each feature to model and implement. This talk presents a survey of six logic-based executable agent specification languages that have been chosen for their possibility to be integrated in the ARPEGGIO framework: Congolog, AGENT-0, the IMPACT agent programming language, DyLog, Concurrent METATEM and Ehhf. The dimensions along which the languages have been compared are: - Time: is time dealt with explicitly in the language? Are there operators for defining complex timed expressions? - Sensing: does the language provide the constructs for sensing actions (namely, actions which sense the environment)? - Communication: are communication primitives provided by the language? Is it necessary for an agent to know details of another agent's implementation in order to communicate with it, or communication takes place on a more abstract level? - Modularity: Does the language provide constructs for defining modules, macros and/or procedures? - Concurrency: Does the language allow to model concurrency of actions within the same agent? Does it support concurrency among executing agents? - Nondeterminism: Does the language support nondeterminism? In which way? - Semantics: Is a formal semantics of the language defined? If yes, which?

CaseLP: a Prototyping Environment for Heterogeneous Multi-Agent Systems

by *Maurizio Martelli*

This talk describes CaseLP, an environment for the rapid prototyping of Multi-Agent Systems (MASs). CaseLP provides the developer with heterogeneous languages for the

specification, design and implementation of the MAS components and their interactions, and an engineering methodology for the realization of the prototype following a sequence of clear steps. CaseLP agents are characterized by their architecture, the roles they play in the MAS, the ontologies they understand, their internal program, and their state. CaseLP offers semi-automatic compilers for implementing executable code starting from the heterogeneous specifications of the agents' components and tools for running and debugging the resulting prototype. The high-level specification languages range from rule-based ones to graphical ones to object-oriented ones. One specification language provided by CaseLP is executable, allowing the early verification and testing of the specification. The prototype implementation language is based on Prolog, and a graphical user interface is provided to help the MAS developer to iteratively test, debug, and refine design and implementation choices. CaseLP has been proven useful for engineering MAS prototypes in application domains ranging from vehicle monitoring to remote video-encoding of mail pieces, from distributed health-care management to decision support systems. Many of these applications were developed for industrial partners.

Organising Rational Agents

by *Benjamin Hirsch*

In this talk we address the task of organising multi-agent systems in order to collectively solve problems. We base our approach on a logical model of rational agency comprising a few simple, but powerful, concepts. While many other researchers have tackled this problem using formal logic, the important aspect of the work described here is that the logical descriptions of the agents are directly executable using the Concurrent MetateM framework, allowing the execution of agents described in a combination of temporal, belief and ability logics. Here, we are particularly concerned with exploring some of the possible logical constraints that may be imposed upon these agents, and how these constraints affect the ability of the agents to come together to collectively solve problems.

This is joint work with Michael Fisher and Chiara Ghidini.

Proof Methods for Multi-Agent Systems

by *Renate Schmidt and Dmitry Tishkovsky*

We propose families of logics which provide formal means for specifying and reasoning about dynamic aspects (actions), informational aspects (knowledge and belief) and motivational aspects (wishes, goals, commitments) of agents. One family of logics, called agent dynamic logics (ADL logics), are based on the combination of propositional dynamic logic and propositional modal logics. We consider a number of extensions of ADL with axiom schemata formalising interactions between knowledge and commitment (expressing an agent's awareness of its commitments), and interactions between knowledge

and actions (expressing no learning and persistence of knowledge after actions). The deductive systems are proved sound and complete with respect to a Kripke-style semantics. Moreover, ADL logics have the small model property and are decidable.

In the second part we introduce a family of so-called BDL logics which are based on a new formalisation and semantics of the test operator of propositional dynamic logic and a representation of actions which distinguishes abstract actions from concrete actions. The new test operator, called informational test, can be used to formalise the beliefs and knowledge of particular agents as dynamic modalities. This approach is consistent with the formalisation of the agents' beliefs and knowledge as K(D)45 and S5 modalities. It also avoids an unwanted side-effect of the interaction of knowledge operators with the classic test operator encountered in ADL logics. Properties concerning the preservation of informativeness, truthfulness and belief are proved for a derivative of the informational test operator. It is shown that common belief and common knowledge can be expressed in BDL logics. As a consequence, these logics are more expressive than propositional dynamic logic with an extra modality for belief or knowledge. However, the logics are still decidable and in 2EXPTIME. Versions of the considered logics express natural additional properties of beliefs or knowledge and interaction of beliefs or knowledge with actions. A simulation of PDL can be constructed in one of these extensions.

Agent Programming in Dribble: from Beliefs to Goals Using Plans

by *John-Jules Meyer (joint work with B. van Riemsdijk and W. van der Hoek)*

To support the practical development of intelligent agents, several programming languages have been introduced that incorporate concepts from agent logics: on the one hand, we have languages that incorporate beliefs and plans (i.e., procedural goals), and on the other hand, languages that implement the concepts of beliefs and (declarative) goals. We propose the agent programming language Dribble, in which these features of procedural and declarative goals are combined. The language Dribble thus incorporates beliefs and goals as well as planning features. The idea is, that a Dribble agent should be able to select a plan to reach a goal from where it is at a certain point in time. In order to do that, the agent has beliefs, goals and rules to select plans and to create and modify plans. Dribble comes with a formally defined operational semantics and, on top of this semantics, a dynamic logic is constructed that can be used to specify and verify properties of Dribble agents. The correspondence between the logic and the operational semantics is established.

Programming the Deliberation Cycle of Cognitive Agencies

by *Mehdi Dastani*

This talk presents the specification of a programming language for implementing the deliberation cycle of cognitive agents. The mental attitudes of cognitive agents are assumed to be represented in an object language. The implementation language for the deliberation cycle is considered as a meta-language the terms of which denote formulae from the object language. Without losing generality, we use the agent programming language 3APL as the object language. Using the meta-deliberation language, one can program the deliberation process of a cognitive agent. We discuss a set of programming constructs that can be used to program various aspects of the deliberation cycle including the planning constructs.

Scientific Benchmarking for Agent-Method Logics

by *Ullrich Hustadt*

In this talk I will first discuss the lack of automated theorem provers for agent logics and consider for which component logics of well-known agent logics working automated theorem provers are publicly available. I will then discuss the problem of how we can assess such theorem provers. I propose a hypothesis-driven design of the empirical analysis of different decision procedures which we refer to as scientific benchmarking. The approach is to start by choosing the benchmark problems for which, on the basis of analytical considerations, we expect a particular decision procedure to exhibit a behaviour different from another decision procedure. Then empirical tests are performed in order to verify the particular hypothesis concerning the decision procedures under consideration. As a case study, I apply this methodology to compare different decision procedures for propositional temporal logic. We define two classes of randomly generated temporal logic formulae which we use to investigate the behaviour of two tableaux-based temporal logic approaches using the Logics Workbench, a third tableaux-based approach using the STeP system, and temporal resolution using two provers called TRP and TRP++.

Speculative Computation in Multi-Agent System

by *Ken Satoh*

In this talk, we present a method of problem solving in multi-agent systems when communication between agents is not guaranteed. Under incomplete communication environments such as the Internet, the communication might fail and a reply might be significantly delayed. Therefore, research of problem solving under incomplete communication is very important.

To solve the problem, we propose a method using abduction. Abduction is a way of reasoning where some hypothesis will be used to complement unknown information. The

idea is (1) When communication is delayed or failed, then we use a default hypothesis as a tentative answer and continue computation (2) When some response is obtained, we check consistency of the response and the current computation. If the response is consistent with the current used hypothesis, we continue the current computation; otherwise, we discard the current computation and seek another alternatives.

We call this process ”speculative computation”. In this talk, we give a method for speculative computation and prove correctness of this method.

Specification and Analysis of Multi Agent Systems

by *Frieder Stolzenburg*

In this talk, a framework is introduced, which allows us to express declarative aspects of multiagent systems by means of (classical) propositional logic and procedural aspects of these systems by means of state machines (statecharts). Nowadays statecharts are a well accepted means to specify dynamic behavior of software systems. They are a part of the Unified Modelling Language (UML). We describe in a rigorously formal manner, how the specification of multiagent systems in general and its verification by model checking can be done, integrating different methods from the field of artificial intelligence. As example application domain, we will consider robotic soccer.

Using model-based diagnosis in multi-agent systems to make assumptions about spatial properties

by *Oliver Obst*

In this talk we present a method to build a hypothesis on the condition of the environment in which a robotic multi-agent team moves. Initially the robots have a default assumption about the conditions of the floor and on how moving under these condition works. For certain parts of the environment however, the default assumption may be wrong and moving around does not work in the expected way. Now the robotic team builds a hypothesis on the conditions of the yet unvisited part of the environment, so resources can be saved by avoiding areas that possibly also contain obstacles.

For a description of the environment and of the observations of the robots, we use propositional formulae in a way similar to computing a diagnosis for electrical circuits. To actually compute the hypothesis, we need to compute models of the given set of clauses, where the extension of the ab-literal is minimal. The description of the environment can be generated automatically, and the proposed method is flexible so that different kinds of topologies can be covered.

Speculative Computation by Consequence Finding

by *Katsumi Inoue*
Joint work with *Koji Iwanuma*

This work is concerned with a multi-agent system which performs speculative computation under incomplete communication environments. In a master-slave style multi-agent system with speculative computation, a master agent asks queries to slave agents in problem solving, and proceeds computation with default answers when answers from slave agents are delayed. In this work, we first provide a semantics for speculative computation using default logic. Speculative computation is considered in which reply messages from slave agents to a master are tentative and may change from time to time. In this system, default values used in speculative computation are only partially determined in advance. Next, we propose a procedure to compute speculative computation using a first-order consequence-finding procedure SOL with the answer literal method. The use of a consequence-finding procedure is convenient for updating agents' beliefs according to situation changes in the world. Then, we further refine the SOL calculus using conditional answer computation and skip-preference in SOL. The conditional answer format has a great advantage of explicitly representing how a conclusion depends on tentative replies and defaults. This dependency representation is important to avoid unnecessary recomputation of tentative conclusions. On the other hand, the skip-preference method has the great ability of preventing irrational/redundant derivations. Finally, we implemented a mechanism of process maintenance to avoid duplicate computation when slave agents change their answers. As long as actual answers from slave agents do not conflict with any previously encountered situation, the obtained conclusions are never recomputed. We applied the proposed system to the meeting-room reservation problem to see the usefulness of the framework.

Default and Cooperative Reasoning in Multi-Agent Systems

by *Chiaki Sakama*

A multi-agent system (MAS) consists of agents which generally have incomplete information as individuals. In artificial intelligence a single agent performs default reasoning in face of incomplete information. In a multi-agent setting, however, the situation is bit different from the case of a single agent. There are two different sources of incomplete information in a multi-agent environment. One is due to incomplete belief with respect to an agent's internal world, and the other is due to incomplete knowledge about its external world. In the former case an agent can perform default reasoning by itself, while in the latter case an agent should perform cooperative reasoning with other agents. To realize default and cooperative reasoning in MASs, it is necessary to distinguish different sources of incompleteness which may arise in a knowledge base. In this study, we represent an agent's knowledge base by logic programming, and introduce a framework of default and cooperative reasoning for MASs. We first provide a declarative semantics

of an MAS using the notion of belief sets. Belief sets characterize the mental state of an agent and distinguish different types of incompleteness which may arise in an MAS. Next we provide a proof procedure for MASs, which solves a given query by default and cooperative reasoning in a top-down manner. The procedure is sound under the belief set semantics when an agent's knowledge base is given as a stratified normal logic program. We discuss further extensions and optimization issues.

Agents via Mixed Mode Computation in Linear Logic

by *James Harland*

Agent systems based on the Belief, Desire and Intention model of Rao and Georgeff have been used for a number of successful applications. However, it is often difficult to learn how to apply such systems, due to the complexity of both the semantics of the system and the computational model. In addition, there is a gap between the semantics and the concepts that are presented to the programmer. In this paper we address these issues by re-casting the foundations of such systems into a logic programming framework. In particular we show how the integration of backward- and forward-chaining techniques for linear logic provides a natural starting point for this investigation. We discuss how the integrated system provides for the interaction between the proactive and reactive parts of the system, and we discuss several aspects of this interaction. In particular, one perhaps surprising outcome is that goals and plans may be thought of as declarative and procedural aspects of the same concept. We also discuss the language design issues for such a system, and particularly the way in which the potential choices for rule evaluation in a forward-chaining manner is crucial to the behaviour of the system.

Agents and Uncertainty - Induction of preferences

by *Emil Weydert*

Uncertainty is a major issue for real-world agents, especially in a multi-agent context. However, the classical decision-theoretic paradigm - maximum expected utility - is not directly applicable because the requirement of a unique probability and utility distribution is quite demanding. In fact, we cannot expect to summarize complex mental states, confronted to highly heterogeneous incoming information, by simple valuations. That's why we propose a two-stage process, where the complex informational and motivational structures are first mapped to a simplified partial representation to which we may then apply a suitable kind of decision-theoretic strategy.

We illustrate this approach with a logical framework based on a logic of histories, of conditional beliefs (with a quasi-probabilistic ranking semantics) and of conditional desires (with a standard utility semantics). In the cognitive deliberation step, we start with nonmonotonic conditional completion procedures, before we apply a further defeasible

inference notion to extract the preferences, which specify the actions (in our framework). Canonical ranking construction strategies have turned out to be reasonable and useful in this context. A major advantage of this approach is that it allows us to combine the quantitative and the qualitative perspective in a flexible way.

Normative Multi Agent Systems

by *Guido Boella*

In this presentation paper sanction-based obligations are formalized in a qualitative decision theory. In particular, we formalize an agent who attributes mental attitudes such as beliefs, goals and desires to the normative e.g. legal, moral system which creates and controls its obligations. The wishes (goals, desires) of the normative system are the commands (obligations) of the agent. The agent reasons about what counts as a violation and when and which sanctions are applied. The agent behavior is determined by its interpretation of the obligations, as well as by its agent type. Since the agent is able to reason about the normative system behavior, our model accounts for many ways a norm can be violated without the risk of being sanctioned.

Progress in BDI Logic Programming with AgentSpeak(L)

by *Rafael H. Bordini*

In this talk, I overview recent work that has been done on AgentSpeak(L), a BDI agent-oriented logic programming language. The first strand of this work is on extensions of AgentSpeak(L) aimed at turning it into a more practical programming language; we call the extended language AgentSpeak(XL). In particular, it allows the use of decision-theoretic task scheduling for the automatic generation of efficient intention selection functions. An interpreter for this language has been implemented. We then overview our framework for proving BDI properties of AgentSpeak(L) agents based on its operational semantics. This framework has been used to show which of the asymmetry thesis principles are satisfied by AgentSpeak(L) agents. The most recent strand of work is on algorithmic verification for AgentSpeak(F) agents, a restricted version of AgentSpeak(L) to ensure that finite state models of such agents can be generated. We can then automatically translate AgentSpeak(F) agents into the model specification languages of existing LTL model checkers. We also convert specifications written in a simplified form of BDI logic into LTL, so that we can use model checking for the verification of AgentSpeak(F) multi-agent systems according to BDI specifications. I conclude the talk by mentioning ongoing and future work.