A Markovian Model for Interaction among Behavior-Based Agents

Jörg P. Müller*

DFKI GmbH, Stuhlsatzenhausweg 3, D-66123 Saarbrücken

Abstract. This paper addresses the question of the effects of the local modeling of agents and their decision-making on the behavior and the performance of these agents in a multiagent environment. Using the example of a particular architecture, the INTERRAP agent model, and a specific application, the loading-dock domain of interacting robots, a general model is presented that allows to predict the problem-solving and interaction performance of a specific type of INTERRAP agents, namely those controlled by local behavior-based decision functions. The analysis of the model provides evidence for the necessity of explicit mechanisms for coordination based on communication in many cases.

1 Introduction

Agents that interact in dynamic multiagent environments need to be equipped with a set of vital capabilities, such as reactivity, the ability to act in a goal-directed manner (e.g., by planning their tasks), to efficiently perform routine tasks, to learn and to be adaptive, and to interact and to coordinate their activities with others. The task of building architectures that allow designers of agents to integrate these functionalities into one agent coherently has recently attracted a variety of research work, parts of which can be read about in this volume (e.g., [2, 9, 4, 5, 7, 20, 12, 17, 1]).

So far, most approaches have concentrated on the definition and the structure of the individual agent and have neglected the question how different design decisions at the layer of agent modeling influence their problem solving behavior in a multiagent environment[2]. On the other hand, researchers that have analyzed distributed systems (e.g., [8]) have not provided adequate operational models of the individual agents.

This paper addresses this question for the case of a specific agent architecture, namely the INTERRAP model (see [13, 14] and [6] in this volume). INTERRAP provides tools for modeling different types of agents within a layered framework. Previous research work primarily dealt with how to combine reactivity with local deliberation and with the ability to interact with other agents by negotiating and executing cooperative plans. In this work, we are concerned with questions of agent interaction that go beyond the scope of an individual agent. The goal of this work is to advance towards the problem of the *evaluation* of agent models and the agent behavior they bring about.

* email: jpm@dfki.uni-sb.de, phone: ++49 681 302 5331, fax: ++49 681 3025341

[2] This fact is not meant to be a criticism: rather it is a very natural phenomenon since research on agent architectures is still at an initial stage, and various problems occurring at the layer of internally describing one agent have not yet been resolved.

The application under consideration is FORKS, the simulation of an automated loading dock; the agents are robots which perform local transportation tasks in an automated loading dock using behavior-based or plan-based decision algorithms (see [15]). Interaction among robots originates from different conflict situations, such as potential collisions and blockings (see Figure 4 for an example). Similar to decision-making in local task planning, coordination can be achieved by applying different mechanisms, which may be either local (for example: trying to dodge the other robot!) or communicative (for instance: trying to find out about the other's goals and generate a joint plan for resolving the conflict!). Figure 1 classifies instances and techniques addressed by different combinations of the dimensions of decision-making and interaction.

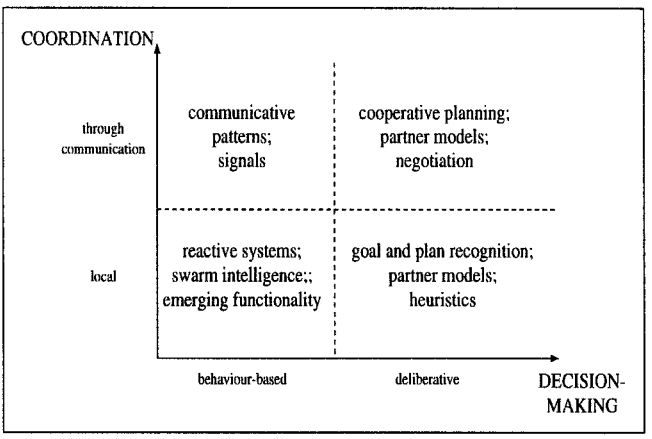

Fig. 1. Dimensions of interactive agent behavior

In the following, we present two probabilistic models based on finite Markov chains that allow to predict the performance of agents acting according to a specific class of local and behavior-based decision methods implemented for agents in the FORKS application. We show that the global implications of local agent design require careful analysis, as seemingly intuitive and reasonable local algorithms in some cases lead to pathological global behavior.

2 The Architectural Framework

The architecture which is under consideration for the local design of agents is INTERRAP. The main idea of INTERRAP is to define an agent by a set of hierarchical functional layers, linked by a activation-based control structure and a shared hierarchical knowledge base. It consists of three basic parts: the control unit, the world interface (WIF), and the knowledge base (KB). The WIF contains the agent's facilities for perception, action, and communication. The control unit consists of three layers: The behavior-based layer

(BBL), the local planning layer (LPL), and the cooperative planning layer (CPL). The BBL implements and controls the basic reactive behavior of the agent as well as its procedural knowledge (abstract actions). It is based on the concept of *patterns of behavior* (PoBs). These allow an agent to react flexibly to its environment, and to perform routine tasks efficiently without requiring explicit symbolic planning. The LPL controls a single-agent planning mechanism. Currently, a planner based on a plan library is provided where plans are hierarchical skeletal structures whose nodes may be either new subplans, or executable PoBs. The CPL extends the agent's planning abilities to joint plans to cope with special situations.

For a detailed description of the architecture, we refer to [6] in this volume and to [13]. In the following, we would like to clarify the scope of our work by providing an example of how a task is processed by an INTERRAP agent. The example is taken from the FORKS application (see Section 4). Assume a forklift agent receives an order `load_truck(`t_1, $b_2$3 to load the truck t_1 with box $b_2$3. In this case, control is shifted upward to the local planning layer, where a corresponding goal is created. Now, a hierarchical plan is created by dividing the goal into subgoals using the plan library. Figure 2 shows an example for a goal hierarchy. At a certain depth in the hierarchy,

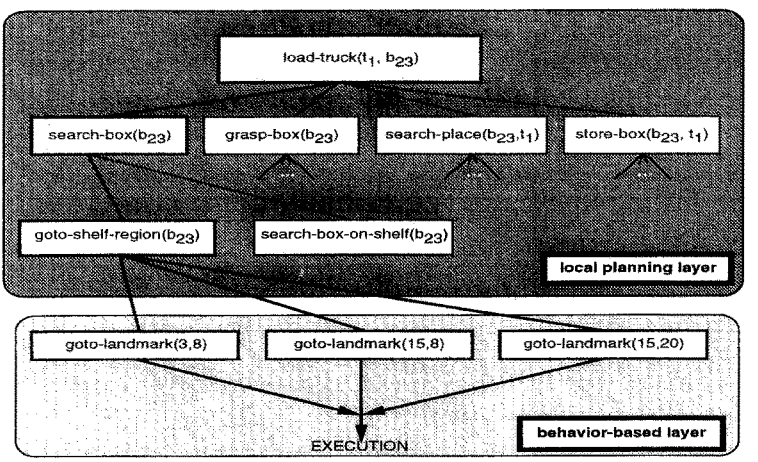

Fig. 2. Hierarchical planning and intelligent execution in INTERRAP

explicit planning stops, and procedure PoBs are activated in the behavior-based layer. In the case of robot navigation, an appropriate abstraction level to start procedure execution are *landmarks*. The advantage of using a procedure is that it provides a means for intelligent execution, i.e., for dealing with certain unforeseen events without explicitly causing replanning at the local planning layer.

A further advantage is that certain conflict resolution strategies may be built in into the procedures, thus often avoiding explicit conflict recognition effort and reducing the communication overhead. E.g., one source of conflicts in the loading dock are

symmetrical blocking situations (see Figure 4). One way to break this symmetry and to avoid conflicts at system design time is by locally programming the navigational PoBs such as `goto_landmark` as *probabilistic decision functions*:

Definition 1 Probabilistic Decision Function (PDF). Let $\mathcal{S}$ be a set of world states. Let $\mathcal{A}$ be a non-empty set of actions, $\mathcal{G} \subseteq \mathcal{S}$ a set of goals; let $f(s, \mathcal{A}, g) \in [0, 1]$ be a probability distribution on $\mathcal{A}$ given state $s \in \mathcal{S}$, and goal $g \in \mathcal{G}$. Then a PDF is a function $\mathcal{F}^f(s, \mathcal{A}, g) = a_i$ with probability $f(s, a_i, g)$ for each $a_i \in \mathcal{A}$. We omit the superscript f for $\mathcal{F}$ in cases it is obvious.

The decision function $\mathcal{F}$ is used in the control cycle of the behavior-based layer to determine the next action to perform in executing a PoB (see [6] in this book).

Now, given a set of agents locally selecting their navigational actions according to PDFs, and assuming that these agents operate in a shared environment, the question is how is the global performance of the system as a whole? In the following, we define two probabilistic models which allow us to predict the behavior of a multiagent system consisting of a set of agents each of which uses a PDF for its local low-level decision making. Note that the scope of this model are not the decisions made at the planning or the cooperation layer of INTERRAP, but rather those made at the procedural, behavior-based layer. This restriction is discussed in Section 5.

3 A Model for Behavior-based Decision-Making

The main idea we propose in this paper is to use the model of finite Markov chains to approximately predict the collective behavior of interacting robots whose individual decisions have the Markov property. This property can be assumed to hold for decision processes implemented at the behavior-based layer of the INTERRAP architecture. This holds true because taking into account only the current state of the world for making decisions is suitable for achieving reactive and situated behavior, which is the main purpose of the BBL.

3.1 Finite Markov chains

Finite Markov chains (see [10] for an introduction) have been used as a model for describing dynamic uncertain environments in the planning literature (see [3, 19]). In these approaches, encoding planning problems into an absorbing Markov chain with transient states and absorbing states (the goal states) allows to draw quantitative conclusions on the behavior of the system.

The way we are using Markov chains is not primarily to cover the uncertain *outcome* of actions (although this will play a role in coping with the presence of multiple agents) but to quantify properties of the class of decision algorithms defined above which describe the *choice* of an action as a stochastic process.

A *finite Markov process* is a finite stochastic process describing probability transitions trough a set of states whose conditional probability distribution satisfies the *Markov property*: the transition probability p_{ij} from state s_{i+1} held at time t_{i+1} depends only on the state s_i at time t_i, but not on previous states. A *finite Markov chain* is a finite

Markov process whose transition probability p_{ij} is independent from how often state s_i has been reached in the past. A Markov chain whose probability does not change over time is called *stationary*. An *absorbing chain* is defined by two disjoint sets of states: transient states and absorbing states. Absorbing states are states that are never left once entered.

The transition matrix M_{ij} of a finite absorbing Markov chain M has the form

$$\begin{pmatrix} Q & R \\ 0 & I \end{pmatrix} \tag{1}$$

where Q are the transitions among transient states, R are transitions from transient to absorbing states, 0 contains only zeros, and I is the identity. The following theorem recalls results from Markov chain theory.

Theorem 2. *Let* $M = \begin{pmatrix} Q & R \\ 0 & I \end{pmatrix}$ *be a finite stationary Markov chain. Then:*

- $N \stackrel{\text{def}}{=} (I - Q)^{-1} = \sum_{t=0}^{\infty} Q^t$ *always exists;* N_{ij} *is the average number of times state* s_j *(which is transient) is entered before an absorbing state is reached if the initial state is* s_i. N *is called the fundamental matrix.*
- $V = N \cdot (2 \cdot N^d - I) - N^2$ *describes the variance for* N, *i.e.,* V_{ij} *is the variance of the total number of times the process is in transient state* s_j *if the chain started in* s_i. N^2 *is computed by squaring each entry of* N; N^d *results from* N *by setting each off-diagonal entry of* N *to zero.*

3.2 A probabilistic model for agent interaction

In the following, we will procede in three steps; first, we provide the necessary formal framework by adopting the general Markov model to our settings; second, we show how this setting can be used to predict the behavior of an indidivual decision-making and acting agent; third, we discuss two ways of representing the effects of agent interaction into the model.

Formal framework We define a finite absorbing Markov chain as a tuple $M \stackrel{\text{def}}{=} (\mathcal{S}, \mathcal{A}, P)$ where $\mathcal{S} = \mathcal{S}_t \cup \mathcal{S}_a, \mathcal{S}_t \cap \mathcal{S}_a = \emptyset$ is a finite set of world states, $\mathcal{S}_t$ represent transient world states and $\mathcal{S}_a$ represent absorbing world states given by the set $\mathcal{G}$ of goal states (throughout the paper, we will mostly deal with the case where S_a is singleton; this, however, is not a general restriction); $\mathcal{A} = \{a_1, \ldots, a_k\}$ is a finite set of actions; $P : \mathcal{S} \times \mathcal{S} \mapsto [0; 1]$ is a probabilistic transition function of the environment: $P(s_i, s_j)$ denotes the probability of getting from state s_i to state s_j. In order to express that P depends on the set of goal states S_a, we write P^{S_a} or simply P^g if $S_a = \{g\}$.

In our application, the function P is composed by two probability distributions: the probability distribution f over possible action selections in s a situation, and the distriution e over the possible outcomes of executing an action. $f : \mathcal{S} \times \mathcal{A} \times \mathcal{S}_a \mapsto [0; 1]$ is the function underlying the actual decision function $\mathcal{F}$ of an agent (see Definition 1). For $s \in \mathcal{S}$, $a \in \mathcal{A}, g \in \mathcal{S}_a, f(s, a, g)$ denotes the probability a is selected for execution

by an agent in state s given the agent's current goal is g. $e : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0; 1]$ describes a probability distribution over the possible outcomes of actions: for $s_1, s_2 \in \mathcal{S}$, $a \in \mathcal{A}$, $e(s_1, a, s_2)$ is the probability of reaching s_2 by executing a given the initial state s_1.

Behavior-based decision-Making Given an agent that makes its decisions using a PDF $\mathcal{F}^f$, the above framework can be used as a model to predict the agent's behavior. For this purpose, we define the transition function P:

Definition 3. Given the initial world state s_0 and a goal state g, the transition function $P : \mathcal{S} \times \mathcal{S} \mapsto [0; 1]$ is given by the transition matrix P_{ij} as the transition probability from state s_i to state s_j with $P^g_{ij} = \sum_{a_k \in \mathcal{A}}(e(s_i, a_k, s_j)|a_k$ has been selected in $s_i) = \sum_{a_k \in \mathcal{A}} f(s_i, a_k, g) \cdot e(s_i, a_k, s_j)$.

By dividing the set of states into two subsets $\mathcal{S}_t$ and $\mathcal{S}_a$, where $\mathcal{S}_a = \mathcal{G}$ and $\mathcal{S}_t = \mathcal{S} - \mathcal{G}$, P^g_{ij} can be written in the form according to Definition 1. By computing the fundamental matrix N and its variance matrix V in accordance with Definition 2, we can determine e.g., the expected number of actions n_i the agent will perform before she reaches a state $g \in \mathcal{G}$ starting from state s_i, which is $n_i = \sum_{j=1}^{n-1} N_{ij}$ if S_a is singleton. See the example in Section 4.

Interaction by local behavior So far, we can draw quantitative conclusions about the performance of a problem-solving agent that is alone in the world. Since we are interested in the multiagent case, we will extend our framework to be able to deal with the presence of multiple agents. In the following, we will discuss two ways of integrating the presence of other agents into our framework: a probabilistic one and an exact one taking into account the current situation.

A naive model: For the probabilistic model, we make use of the observation that at the level of local behavior-based decision-making, the main effect of the presence of others is that it increases the likelihood of failure of domain actions. A probabilistic extension of the above model is proposed that allows to predict the approximate performance of an agent given the size of its environment and the number of other agents being around.

Given a Markov chain model $M = (\mathcal{S}, \mathcal{A}, P(f, e))$ as defined above, the main idea is to incorporate knowledge about the effects of the presence of other agents into the transition function P. For the scope of this paper, we restrict ourselves to a basic type of interaction which occurs in the domain we are looking at: the probability of an action to fail increases linearly with the number of other agents being present: for k agents, the probability for an action a by an individual agent to fail in state s is $p(fails(a, s)) = \frac{k-1}{c}$ for a constant $c \geq k - 1$. In order to cover the notion of execution failure which occurs if an action is carried out whose precondition is not satisfied, we make the following assumption[3]:

Definition 4 Execution failure. Let $s \in \mathcal{S}$ be a state of the world, $a \in \mathcal{A}$ be an action. Let $prec(a)$, $exec(a, s)$ be formulae denoting the preconditions of a, and the execution

[3] This restriction is made for simplicity. it could be weakened by assuming a probability distribution over different possible states.

of a in state s, respectively; let $\mathit{eff}(a, s)$ the state resulting from executing a in state s, and let $holds(p, s)$ be a metapredicate evaluating to true if predicate p holds in state s. Then, $\neg holds(prec(a), s) \wedge exec(a, s) \Rightarrow \mathit{eff}(a, s) = s$.

The implication for the transition function P is that the failure of the execution of an action will result in an unchanged state, i.e., a transition $s \rightarrow s$. This is covered by the following definition:

Definition 5. Given the local $n \times n$ transition matrix $P = P^1$ for one agent. Then the *k-agent transition matrix* P^k, $k \geq 2$, is defined by

$$P_{ij}^k = \begin{cases} \frac{k-1}{c} \cdot \sum_{\substack{m=1 \\ m \neq i}}^{n} P_{im} & \text{if } i = j \\ (1 - \frac{k-1}{c}) \cdot P_{ij} & \text{otherwise} \end{cases}$$

Now, we are able to define the naive model as a Markov chain.

Definition 6 Naive model. Let $M = (\mathcal{S}, \mathcal{A}, P)$ be a finite absorbing Markov chain for an individual agent; let k be the number of agents. Then $M^k = (\mathcal{S}, \mathcal{A}, P^k)$ is the Markov chain denoting the naive model of interaction.

Theorem 7. *Let M be a finite absorbing Markov chain. Then M^k is a finite absorbing Markov chain.*

In Section 4, we will provide an example how the naive model can be used to derive quantities of the process.

An exact model: The main problem of the approach based on approximation is that it does not allow good predictions of the performance of agents given specific initial situations, such as conflicts (see Figure 4). Therefore, we provide an alternative approach, which models the system consisting of several agents by one Markov chain. The Markov transition matrix for the system can be computed from the transition matrices of the individual agents. It is important to note that in constructing the exact model we raise our view from the perspective of the individual agent to a global perspective of the system. We start from a multiagent system with k agents, in the sequel denoted by indices $1 \leq i \leq k$, where the local behavior of each agent is described by a finite absorbing Markov chain.

Definition 8. Let $Ag = \{1, \ldots, k\}$ be a set of agents. For each $i \in Ag$, let $M^i = (\mathcal{S}^i, \mathcal{A}^i, P^i)$ be the Markov chain describing the local behavior of i. Let $\mathcal{D}$ be a set of first-order formulae denoting a domain theory. Then, $M = (\mathcal{S}, \mathcal{A}, P)$ is the Markov chain denoting the exact model, with $\mathcal{S} = \mathcal{S}^1 \times \ldots \times \mathcal{S}^k$ $\mathcal{A} = \mathcal{A}^1 \times \ldots \times \mathcal{A}^k$ $P = \gamma(\mathcal{D}, P^1, \ldots, P^n)$

States are defined by the Cartesian product of the states of the individual agents; states change due to simultaneous actions of the agents. The transition probabilities are computed by the probabilities of the individual agents using a function γ. A naive way to derive the probability of a transition among two states $s, s' \in \mathcal{S}$, $s = (s_1 \ldots s_k)$ and $s' = (s'_1 \ldots s'_k)$ is to define $P(s, s') = \prod_{i=1}^{k} P_i(s_i, s'_i)$. However, we have to exclude

inconsistent state transitions using a domain theory $\mathcal{D}$ representing a set of domain constraints. $\mathcal{D}$ partitions the set $\mathcal{S}$ of states into two disjoint subsets, the admissible and the inconsistent state transitions:

Definition 9. Let $M = (\mathcal{S}, \mathcal{A}, P)$ be a finite absorbing Markov chain for k agents. Let $\mathcal{D}$ be a domain theory. Let $s, s' \in \mathcal{S}$, $s = (s_1 \ldots s_k), s' = (s'_1 \ldots s'_k)$. Then the set of admissible state transitions AS is defined as $\{s \rightarrow s' \in \mathcal{S} | P^i(s_i \rightarrow s'_i) > 0 \wedge \mathit{consistent}(\mathcal{D}, s \rightarrow s')\}$. The set of inconsistent state transitions is $IS = \mathcal{S} - AS$.

I.e., for a state transition to be admissible we require both the feasibility of the local state transitions and consistency of the global state transition with the domain constraints. In case the global transition achieved by the set of simultaneous local transitions is inconsistent, the system as a whole is still required to end up in a well-defined state. Therefore, we define a set CS of *compromise state transitions* whose purpose is to replace inconsistent state transitions. CS characterizes the set of alternative states which can be taken by the k-agent system whenever the simultaneous local decisions of the agents would lead to a globally inconsistent state $\hat{s} \in IS$. CS is required to be consistent with $\mathcal{D}$; however, for $s \rightarrow s' \in CS$ we do not require that $P^i(s_i, s'_i) > 0$. I.e., the result of a compromise state transition for an individual agent can be one the agent herself would never have selected if it was the only agent in the world.

Algorithm 1 *Algorithm for computing* $P = \gamma(P^1, \ldots, P^k)$*:*

1. *Compute the Cartesian product of all possible state transitions:* $S' := S^1 \times \ldots \times S^k$.
2. $\forall s, s' \in S', s = (s_1, \ldots, s_k), s' = (s'_1, \ldots, s'_k) : p'(s, s') := \prod_{i=1}^{k} P^i(s, s')$.
3. *Compute* AS *and* IS *from* S'. *Set* $CS := \emptyset$.
4. $\forall (s, s') \in AS$: $P_{AS}(s, s') := p'(s, s')$.
5. $\forall (s, s') \in IS$: $CS := CS \cup \tau(s, s')$, *where* $\tau(s, s')$ *is the set of all consistent subsequent states of* s *when trying to reach state* s'.
6. *For all* $(s, s') \in IS$ *and for all* $\hat{s} \in \tau(s, s') : P_{CS}(s, \hat{s}) := p'(s, \hat{s}) + \frac{p'(s, s')}{|\tau(s, s')|}$.
7. *For all* $(s, s') \in IS$ *set* $P_{IS}(s, s') := 0$.
8. $P = P_{AS} \cup P_{CS} \cup P_{IS}$

Theorem 10. *Let* Ag, M^i, $\mathcal{D}$, *and* $M = (\mathcal{S}, \mathcal{A}, P)$ *be as in Definition 8 and Algorithm 1. Let* M^i *describe a finite absorbing Markov chain for each* $i \in Ag$. *Then* M *describes a finite absorbing Markov chain.*

Theorem 10 guarantees us that, having computed P, the model $M = (\mathcal{S}, \mathcal{A}, P)$ can be used in the standard way by applying the tools of Markov chains to derive quantitative properties of the MAS. For an example, we refer to Section 4.

Whilst the exact model allows to make detailed predictions on the future behavior of the system, this analysis is not for free. Whereas space and time complexity of the naive model are polynomial (space complexity in $\mathcal{O}(n^2)$, time complexity in $\mathcal{O}(n^3)$), the space requirement for the exact model for n agents with m local states is in $\mathcal{O}(n^{2 \cdot m})$. The complexity of computing the fundamental matrix is in $\mathcal{O}(n^{3 \cdot m})$ and thus exponential in the number of agents.

However, in many domains (in particular in the one investigated here), many interesting situations (such as the conflicts described in Section 4) can be described by rather

simple settings and do not involve a prohibitive number of states. In these cases, as we will see in Section 4, the exact model is tractable and useful to analyze global effects of local decision algorithms in specific situations. Thus, it can help the system designer to avoid certain dangerous pitfalls.

4 Behavior-based Interaction among Interacting Robots

4.1 FORKS: the loading dock application

FORKS describes autonomous forklifts as agents in a multiagent system that have to carry out transportation tasks in a loading dock. FORKS has been implemented both as a computer simulation running and in a real robot scenario using KHEPERA miniature robots.

Figure 3 illustrates the structure of the loading dock. It is represented as a grid; each square $((x, y), t, r)$ has coordinates (x, y), a type $t \in \{grnd, truck, shelf\}$, and can be within a region $r \in \{parking\_zone, hallway, truck\_region, shelf\_region\}$. Squares of types *truck* and *shelf* can additionally contain at most one box. The forklifts themselves

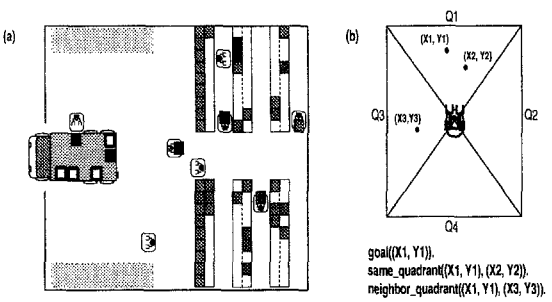

Fig. 3. (a) The loading dock (b) Quadrants

occupy one square at a time; they have a certain range of perception (one square in front is realistic), can communicate with other agents and can perform actions that change the world. The actions a forklift can perform are to move to a neighbor field, to turn to a certain direction, to grasp a box standing in front of it, and to put a box it currently holds.

4.2 The model

In this Section, we show how the domain is represented using the framework developed in Section 3. A Markov chain is built by defining states, actions, and state transitions. Due to space limitations, we will restrict the model to a specific portion of the domain, namely to *coordinated movement* of the agents through the loading dock. Conflicts are

investigated by the example of blocking situations (see Figure 4). Other parts of the domain, e.g., those describing the search and manipulation of boxes can be modeled as independent Markov subchains in a way similar to the one we describe here; they can be composed to describe the overall behavior of the agent (see [3] for a discussion of this subject in the context of restricting the search space for planning). In the framework of INTERRAP this assumption makes sense, since the higher-level behavior of an agent is maintained by a planner that represents the goals of an agent in a hierarchical tree structure (See Section 2). Only the leaf nodes of this tree are achieved by patterns of behavior such as *goto_landmark* or *search_box* which are described by the model presented in this paper. Hence, the independence assumption between the different patterns of behavior is justified.

States: The state of a forklift is defined as a 3-tuple $((x, y), o, b)$ where (x, y) denotes the current location of the forklift, $o \in \{s, n, e, w\}$ its direction, and $b \in \{boxdescription, nil\}$ denotes whether the forklift is currently holding a box or not. For the movement task considered in this paper, the only relevant variable identifying the state of a forklift is its location. Therefore, we will restrict the description of the local state of a forklift by constructing an *envelope* [3] consisting of the possible tuples (x, y).

Actions: The action which is relevant for studying coordination of movement is the *moveto* action which makes the forklift move to a direction specified as an argument to the action. The operational semantics is as follows:

Name: *moveto(Dir)*
Effects: $s = ((x, y), o, b) \mapsto ((x', y'), o', b)$ with
$o' = Dir$
$(x', y') = switch(Dir)$ `/* case distinction */`
{
case $'n'$: $(x, y + 1)$;
case $'e'$: $(x + 1, y)$;
case $'s'$: $(x, y - 1)$;
case $'w'$: $(x - 1, y)$
}
Precond: $type((x', y')) \notin \{truck, shelf\} \wedge$
$status((x', y')) \neq occupied.$

Here, *status* is a unary function that receives as argument a square and returns *occupied* if the square is occupied by an agent. $Type$ is the accessor function to the type t of a square $((x, y), t, r)$. Note that this definition describes the intended effects of the action rather than the actual effects at the time of execution.

Transitions: Transitions among local states that correspond to movements of the forklifts through the scenario are achieved by the execution of actions. The incentive to act is given by goals. In our example, a goal is given by a location the robot is to reach. Thus, there is a natural mapping from goals to states.

The local transition function P^i of agent i is given by instantiations of the functions f^i and e^i: The function e^i that represents the outcome of an action is assumed to be deterministic. I.e., for the one-agent case we assume that actions bring about the expected effects. f^i defines the probabilities of the robot to move to some direction depending on the relative position of the goal wrt. to the robot's current position. The underlying decision function $\mathcal{F}^f$ is based on the notion of quadrants (see Figure (3.b)). Given the current position s and a goal g, the world is divided into four quadrants q_1 – q_4. q_1 is the quadrant where the goal field is located. q_4 is the quadrant which is directed away from the goal; q_2 and q_3 are orthogonal to the goal. Given a state s and a goal state g, $q_i(s,g)$ returns the state (square) which is directly reachable from s and which is in quadrant q_i wrt. g. Thus, the square $q_i(s,g)$ is selected with a probability of $f(s,q_i,g)$. In the following, let q_i denote the action which will result in the state of the agent being $q_i(s,g)$. $f(s,q_i,g)$ can be equal to zero if the agent cannot move to this direction because there is an obstacle there. For this case, we include a dummy operator $q_0(s,g) = s$. Function P^i can now be defined as follows:

Definition 11. Let $\mathcal{F}^f$ be a PDF; let $c_1, c_2 > 1$. Let $\mathcal{A}^i = \{q_j | 0 \leq j \leq 4\}$. Let f be constrained by the following conditions: For all s, g, we have:

(1) $\sum_{i=0}^{4} f(s,q_i,g) = 1$
(2) $f(s,q_1,g) > 0 \wedge f(s,q_2,g) > 0 \Rightarrow (s,q_1,g) = c_1 \cdot f(s,q_2,g)$
(3) $f(s,q_1,g) > 0 \wedge f(s,q_3,g) > 0 \Rightarrow f(s,q_1,g) = c_1 \cdot f(s,q_3,g)$
(4) $f(s,q_2,g) > 0 \wedge f(s,q_3,g) > 0 \Rightarrow f(s,q_2,g) = f(s,q_3,g)$
(5) $f(s,q_2,g) > 0 \wedge f(s,q_4,g) > 0 \Rightarrow f(s,q_2,g) = c_2 \cdot f(s,q_4,g)$
(6) $f(s,q_3,g) > 0 \wedge f(s,q_4,g) > 0 \Rightarrow f(s,q_3,g) = c_2 \cdot f(s,q_4,g)$
(7) $f(s,q_0,g) > 0 \Leftrightarrow \sum_{i=1}^{4} f(s,q_i,g) = 0$

Furthermore, let $e^i : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0;1]$, $e_i(s,q_i,q_i(s,g)) = 1$. Then, $P^i : \mathcal{S} \times \mathcal{S} \mapsto [0;1]$ is defined by $P^{i,g}(s,s') = \sum_{q \in \mathcal{A}} f_i(s,q,g) \cdot e_i(s,q,s') = \sum_{q \in \mathcal{A}} f_i(s,q,g)$.

If $c_1 = c_2 = 1$, we obtain a random walk strategy, where the agent selects each direction with the same equality. By choosing $c_1, c_2 > 1$ we obtain a weighted strategy that makes the agent move towards its goal with a higher probability.

Example: Let us consider example (4.a). Assume that agent a who is at location $(1,2)$ has the goal to move to location $(2,2)$. Then, $q_1((1,2),(2,2)) = (2,2)$, $q_2((1,2),(2,2)) = (1,1)$, $q_3((1,2),(2,2)) = (3,1)$; q_4 is not possible in the example. By choosing $c_1 = 4$, $c_2 = 1.5$, we obtain the probabilities $p((1,2),(2,2)) = \frac{2}{3}$, $p((1,2),(1,3)) = p((1,2),(1,1)) = \frac{1}{6}$. Thus, the local transition matrices P^a and P^b for agents a and b can be written as

$$P^a = \begin{array}{c|c} & \begin{array}{cccccc} 11 & 12 & 13 & 21 & 23 & 22 \end{array} \\ \hline \begin{array}{c} 11 \\ 12 \\ 13 \\ 21 \\ 23 \\ 22 \end{array} & \begin{pmatrix} 0 & \frac{1}{5} & 0 & \frac{4}{5} & 0 & 0 \\ \frac{1}{6} & 0 & \frac{1}{6} & 0 & 0 & \frac{2}{3} \\ 0 & \frac{1}{5} & 0 & 0 & \frac{4}{5} & 0 \\ \frac{1}{5} & 0 & 0 & 0 & 0 & \frac{4}{5} \\ 0 & 0 & \frac{1}{5} & 0 & 0 & \frac{4}{5} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \qquad P^b = \begin{array}{c|c} & \begin{array}{cccccc} 11 & 13 & 21 & 23 & 22 & 12 \end{array} \\ \hline \begin{array}{c} 11 \\ 13 \\ 21 \\ 23 \\ 22 \\ 12 \end{array} & \begin{pmatrix} 0 & 0 & \frac{1}{5} & 0 & 0 & \frac{4}{5} \\ 0 & 0 & 0 & 0 & \frac{1}{5} & \frac{4}{5} \\ \frac{4}{5} & 0 & 0 & \frac{1}{5} & 0 & 0 \\ \frac{1}{6} & 0 & \frac{1}{6} & 0 & 0 & \frac{2}{3} \\ 0 & \frac{4}{5} & 0 & \frac{4}{5} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Based on the single-agent Markov chains, in the next Section, we will provide examples for the naive model and the exact model defined in Section 3.

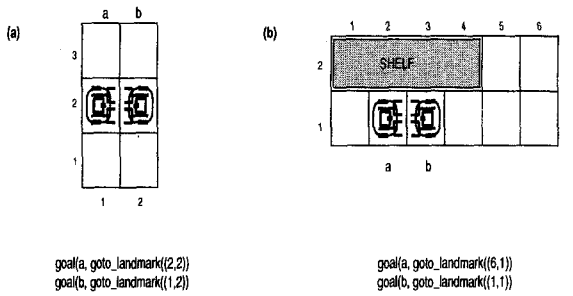

Fig. 4. Conflict situations in the loading dock

4.3 Analysis of system behavior

The naive model The transition matrix P^a for agent a in the example shown in Figure (4.a) can be used to derive some quantities as regards the local problem-solving behavior of agent a. The fundamental matrix N^a for P^a has the form

$$N^a = \begin{array}{c} \\ 11 \\ 12 \\ 13 \\ 21 \\ 23 \end{array} \begin{array}{c} \begin{array}{ccccc} 11 & 12 & 13 & 21 & 23 \end{array} \\ \left(\begin{array}{ccccc|c} 1.24 & 0.26 & 0.05 & 0.99 & 0.04 & \mathbf{2.59} \\ 0.22 & 1.09 & 0.22 & 0.18 & 0.18 & \mathbf{1.88} \\ 0.05 & 0.26 & 1.24 & 0.04 & 0.99 & \mathbf{2.60} \\ 0.25 & 0.05 & 0.01 & 1.20 & 0.01 & \mathbf{1.52} \\ 0.01 & 0.05 & 0.25 & 0.01 & 1.20 & \mathbf{1.52} \end{array}\right) \end{array} \quad \begin{array}{c} \\ \\ \\ \\ \\ \textit{total} \end{array}$$

Thus, for $c_1 = 4, c_2 = 1.5$, the expected number of steps needed to reach the goal state $(2,2)$ when starting from state $(1,1)$ is equal to $\bar{n} = 2.59$, the variance is equal to $v = 1.08$. The optimal solution in this case is 2. Thus, the performance of agent behavior can be improved by making it more goal-directed, i.e., by selecting other values for c_1, c_2. For $c_1 - c_2 \to \infty$, the solution asymptotically approximates the optimum; the variance decreases. E.g., for $c_1 = 20, c_2 = 4.5$, we have $\bar{n} = 2.11$ and $v = 0.21$. However, as we will see, if we introduce multiple agents which all behave according to this strategy, this is no longer the case.

Yet, first we provide an instantiation of the naive model defined above. For k agents and a number of grids of n, we compute the k-agent matrix P^k according to Definition 5, setting $c = n - 1$ and $p(fails(a,s)) = \frac{k-1}{n-1}$. For Example (4.a) and $k = 5$, $c_1 = 4$, $c_2 = 1.5$, $P^{k,a}$ and $N^{k,a}$ can be computed as:

$$P^{k,a} = \begin{pmatrix} 0.8 & 0.04 & 0 & 0.16 & 0 & 0 \\ 0.03 & 0.8 & 0.03 & 0 & 0 & 0.13 \\ 0 & 0.04 & 0.8 & 0 & 0.16 & 0 \\ 0.04 & 0 & 0 & 0.8 & 0 & 0.16 \\ 0 & 0 & 0.04 & 0 & 0.8 & 0.16 \end{pmatrix}; N^{k,a} = \begin{pmatrix} 6.18 & 1.28 & 0.23 & 4.95 & 0.18 & \mathbf{12.82} \\ 0.96 & 5.38 & 0.96 & 0.77 & 0.77 & \mathbf{8.84} \\ 0.23 & 1.28 & 6.18 & 0.18 & 4.95 & \mathbf{12.82} \\ 1.24 & 0.26 & 0.05 & 5.99 & 0.04 & \mathbf{7.58} \\ 0.05 & 0.26 & 1.24 & 0.04 & 5.99 & \mathbf{7.58} \end{pmatrix}$$

For getting from $(1,1)$ to $(2,2)$ in the above example, we have $\bar{n} = 12.28$; the variation $v = 77.22$. For $c_1 = 20, c_2 = 4.5$, we obtain $\bar{n} = 10.55$ and $v = 47.4$. Thus,

in the naive model, goal-directed behavior of individuals improves the system efficiency also in the multiagent case. The reason for this is the underlying assumption that other agents will behave randomly. Using the exact model allows us to drop this assumption.

The exact model Assume now that agent a in Figure (4.a) has the goal to move to square $(2,2)$, whereas agent b's goal is to move to square $(1,2)$. We will analyze how well this type of conflict can be resolved by using the local behavior-based decision algorithms described above. The same question will be studied by the example shown in Figure (4.b) which shows a conflict situation in a shelf corridor that seems (intuitively) harder to resolve.

First, we will define the exact model according to Definition 8. In our example, $\mathcal{S}$ is defined by tuples (s_a, s_b). E.g., the initial state is encoded as $s = ((1,2),(2,2) \equiv 1222$[4], the goal state is $g = 2212$. Actions in $\mathcal{A}$ are pairs (q_a, q_b) denoting simultaneous moves through the grid. Consequently, the transition matrix P defines the probabilities of reaching one (compound) state from another one. The domain theory $\mathcal{D}$ is given by

$$\begin{aligned}\mathcal{D} = \{\ & \forall i,j \in Ag \forall t \in T. loc(i,t) \neq loc(j,t), \\ & \forall i,j \in Ag \forall t_i \in T. \mathit{neighbor}(loc(i,t_i), loc(j,t_i)) \\ & \Rightarrow (loc(i,t_{i+1}) \neq loc(j,t_i) \vee loc(j,t_{i+1}) \neq loc(i,t_i))\}\end{aligned}$$

where T is a set of time instants, $Ag = \{1,\ldots,k\}$ a set of agents, $loc : T \times Ag \mapsto \mathcal{S}$, $loc(i,t) = s$ if the location of agent i at time t is s, and $\mathit{neighbor} : \mathcal{S} \times \mathcal{S} \mapsto [0;1]$, $\mathit{neighbor}(s_1, s_2) = 1$ iff s_1 and s_2 are neighbors. $\mathcal{D}$ expresses two domain constraints: first, two agents are not allowed to be at the same location at a time; second, agents must not move through each other. Hence, the direct transition from $s = 1222$ to $g = 2221$ is not admissible.

In the following, we will compute P using Algorithm 1 based on the individual transition matrices P^a and P^b and domain theory $\mathcal{D}$. After computing the Cartesian product S' of possible state transitions, the probabilities p'_{ij} for S' are initialized. At this stage, we can restrict our attention to entries (s,s') with $p'(s,s') > 0$. For $s = 1222$, we obtain

$$\begin{array}{lll} p'(1222,1112) = \frac{1}{9} & p'(1222,1121) = \frac{1}{36} & p'(1222,1123) = \frac{1}{36} \\ p'(1222,1312) = \frac{1}{9} & p'(1222,1321) = \frac{1}{36} & p'(1222,1323) = \frac{1}{36} \\ p'(1222,2212) = \frac{4}{9} & p'(1222,2221) = \frac{1}{9} & p'(1222,2223) = \frac{1}{9} \end{array}$$

Now, S' is divided in admissible and inconsistent states; the only successor of 1222 which is inconsistent with $\mathcal{D}$ is 2221: $IS = \{2221\}$ and $AS = S' - IS$. Next, the compromise states CS are computed from IS. In this example, the only consistent subsequent state when trying to reach 2221 is the initial state 1222, i.e., the local state transitions fail for both a and b. Thus, $\tau(1222,2212) = \{1222\}$, and $CS = \{1222\}$. $P_{CS}(1222,1222) = \frac{p'(1222,2212)}{1} = \frac{4}{9}$. Now, the set of direct successors for 1222 and the corresponding probabilities P are computed for all states. The resulting matrix is of size 36×36. The fundamental matrix N is computed as usual; it can be used to derive the expected number of steps needed to end in an absorbing state given any transient

[4] We will use this compact notation throughout the example.

state. Table 1 shows the mean number of steps to get from the initial state 1222 to the goal state 2122 for three different initial values for c_1, c_2 in comparison with the optimal solution[5].

		Example (3.a)		Example (3.b)		
c_1	c_2	mean # of state transitions	opt. # of state trans.	mean # of state trans.	opt. # of state trans.	strategy
1	1	7.8	3	122.99	8	random
4	1.5	4.86	3	57.4	8	moderately goal-directed
20	4.5	7.91	3	631.48	8	strongly goal-directed

Table 1. Mean number of states needed to resolve the conflicts in Figure 3

Surprisingly, the highly goal-directed strategy performs worst for conflict resolution; it is even worse than a random strategy. The moderately goal-directed behavior performs considerably better; it might still be acceptable if we take into account the cost of synchronization in the optimal solution.

Next we will analyze the conflict situation shown in Figure (4.b). Assuming the goal of agent a is to move from square $(2, 1)$ to $(6, 1)$ and the goal of agent b is to move from $(3, 1)$ to $(1, 1)$, constructing the k-agent Markov chain according to Algorithm 1 yields the results shown in Table 1. None of the strategies yields acceptable results[6]. Again, the moderately goal-directed strategy performs best; however, agents the highly goal-directed strategy needs much longer to resolve the conflict than the random strategy. The basic reason for this is that both agents follow a hill-climbing strategy. However, resolving the conflict situation in Figure (4.b) requires agent b to make a number of steps in the "wrong" direction, which becomes the more unlikely the more goal-directed b acts.

The main result of this analysis is that there are dangerous pitfalls in the local modeling of agents using even very intuitive strategies if these strategies are used in a multiagent environment. One rash conclusion from this result would be to abolish these strategies and to employ optimal strategies instead, e.g., by coordinating the behavior of the agents globally. However, as Latombe [11] has shown for multi-robot path planning, global modeling is not tractable in general. For this reason, it makes sense to program the behavior-based layer of robots using simple and general algorithms, and then to provide a control architecture that extends these local strategies with the ability to recognize conflict situations as they occur and to employ other means of conflict resolution in situations where necessary. For this purpose, our architecture allows the agent designer to enable agents to resolve conflicts by synchronized plans for multiple agents where synchronization is achieved by communication.

[5] An optimal solution for the example is e.g., $1222 \rightarrow 2221 \rightarrow 2211 \rightarrow 2212$.

[6] The state transitions $2131 \rightarrow 3141 \rightarrow 4151 \rightarrow 5152 \rightarrow 6151 \rightarrow 6141 \rightarrow 6131 \rightarrow 6121 \rightarrow 6111$ describe an optimal solution.

The results provided by this paper are complemented by empirical results (see [6] inside this volume) comparing the performance of different agent types in the FORKS system. Especially, the simple behavior-based decision strategies presented in this paper were compared conflict resolution methods based on communication. The essence of these experiments is that communication can yield a considerable gain in performance, as is implied by the theoretical results developed in the paper at hand. However, the actual benefit depends strongly on several factors such as on the number of agents. If there are many agents and a rapidly changing environment, (cooperative) planning tends to become obsolete before it comes to a result. Therefore, local modeling and decision-making mechanisms have a justification for practical systems.

5 Discussion

We have addressed the effects of the local modeling of agents on the behavior of these agents in a multiagent environment. For a specific agent architecture and a class of applications, two views of a formal model have been provided that support the analysis of agents, which use a particular class of behavior-based decision algorithms. The model allows us to make quantitative predictions of the performance of different decision algorithms using a strict mathematical framework and that it takes into account both uncertainty in action selection and uncertainty in execution. Maybe the main question concerns the adequateness of the model: Is it reasonable to assume the Markov property for the decision-making of interacting agents? If asked in this generality, the answer is certainly "no". Changes in the environment will force the agent to learn and to change its behavior. However, the agent's decision behavior does not change arbitrarily often over time. Thus, the model is still useful to explain the behavior of the agent in between these changes. A second limitation of the model is that it is not able to cope with these forms of agent interaction studied in DAI (see e.g., [18, 16]) that deal with negotiation and require an agent to maintain a history of interactions with other agents. The point is, however, that the model is not intended to cover these forms of interaction, since they are modeled at the cooperation layer of INTERRAP. Future work will investigate models and tools for evaluating agent interaction based on planning and communication.

Acknowledgements

I thank Michael Beetz and Klaus Fischer for helpful discussions on earlier drafts of the paper. The work presented in this paper has been supported by the German Ministry of Education and Research under grant ITW9104.

References

1. R. P. Bonasso, D. Kortenkamp, D. P. Miller, and M. Slack. Experiences with an architecture for intelligent, reactive agents. In M. Wooldridge, J. P. Müller, and M. Tambe, editors, *Intelligent Agents — Proceedings of the 1995 Workshop on Agent Theories, Architectures, and Languages (ATAL-95)*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 1996. (In this volume).

2. Rodney A. Brooks. A robust layered control system for a mobile robot. In *IEEE Journal of Robotics and Automation*, volume RA-2 (1), pages 14–23, April 1986.
3. T. Dean, L. P. Kaelbling, L. P. Kirman, and A. Nicholson. Planning with deadlines in stochastic domains. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 574–579, 1993.
4. I. A. Ferguson. *TouringMachines: An Architecture for Dynamic, Rational, Mobile Agents*. PhD thesis, Computer Laboratory, University of Cambridge, UK,, 1992.
5. R. James Firby. Building symbolic primitives with continuous control routines. In J. Hendler, editor, *Proceedings of the 1st International Conference on Artificial Intelligence Planning Systems (AIPS-92)*. Morgan Kaufmann Publishers, San Mateo, CA, 1992.
6. K. Fischer, J. P. Müller, and M. Pischel. A pragmatic BDI architecture. In M. Wooldridge, J. P. Müller, and M. Tambe, editors, *Intelligent Agents — Proceedings of the 1995 Workshop on Agent Theories, Architectures, and Languages (ATAL-95)*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 1996. (In this volume).
7. E. Gat. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In *Proceedings of AAAI'92*, pages 809–815, 1992.
8. J. Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, 1990.
9. L. P. Kaelbling. An architecture for intelligent reactive systems. In J. Allen, J. Hendler, and A. Tate, editors, *Readings in Planning*, pages 713–728. Morgan Kaufmann, 1990.
10. J. Kemeny and L. Snell. *Finite Markov Chains*. van Nostrand, Princeton, NJ, 1960.
11. J. P. Latombe. How to move (physically speaking) in a multi-agent world. In Y. Demazeau and E. Werner, editors, *Decentralized A.I.*, volume 3. North-Holland, 1992.
12. D. M. Lyons and A. J. Hendriks. A practical approach to integrating reaction and deliberation. In *Proceedings of the 1st International Conference on AI Planning Systems (AIPS)*, pages 153–162, San Mateo, CA, June 1992. Morgan Kaufmann.
13. J. P. Müller and M. Pischel. An architecture for dynamically interacting agents. *International Journal of Intelligent and Cooperative Information Systems (IJICIS)*, 3(1):25–45, 1994.
14. J. P. Müller and M. Pischel. Integrating agent interaction into a planner-reactor architecture. In M. Klein, editor, *Proceedings of the 13th International Workshop on Distributed Artificial Intelligence*, Seattle, WA, USA, July 1994.
15. J. P. Müller, M. Pischel, and M. Thiel. Modeling reactive behaviour in vertically layered agent architectures. In M. J. Wooldridge and N. R. Jennings, editors, *Intelligent Agents — Theories, Architectures, and Languages*, volume 890 of *Lecture Notes in AI*. Springer, January 1995.
16. J. S. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*. MIT Press, 1994.
17. A. Sloman and R. Poli. SIM_AGENT: A toolkit for exploring agent designs. In M. Wooldridge, J. P. Müller, and M. Tambe, editors, *Intelligent Agents — Proceedings of the 1995 Workshop on Agent Theories, Architectures, and Languages (ATAL-95)*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 1996. (In this volume).
18. K. P. Sycara. *Resolving Adversarial Conflicts: An approach integrating case-based and analytic methods*. PhD thesis, Georgia Institute of Technology, Atlanta, Georgia, June 1987.
19. S. Thiébaux, J. Hertzberg, W. Shoaff, and M. Schneider. A stochastic model for actions and plans for anytime planning under uncertainty. *Intl. Journal of Intelligent Systems*, 10(2):155–183, February 1995.
20. M. J. Wooldridge and N. R. Jennings, editors. *Intelligent Agents – Theories, Architectures, and Languages*, volume 890 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1995.