UNIVERSITY OF SOUTHAMPTON
Faculty of Engineering, Science and Mathematics
School of Electronics and Computer Science

A progress report submitted for continuation towards a PhD

Supervisor: Professor Michael Luck
Dr. Nicholas Gibbins
Examiner: Dr. Terry Payne

Motivated Declarative Agents in Multiagent Domains: Open Issues

by Felipe Rech Meneguzzi

June 30, 2006

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

A progress report submitted for continuation towards a PhD

by Felipe Rech Meneguzzi

Despite the renewed interest in the declarative nature of goals in autonomous agents, a gap between declarative agent theory and practical architectures still exists. This can be attributed to the way in which architectures of autonomous agents were initially designed to rely on a pre-defined plan library that includes all possible agent behaviours encoded in procedural plans. In such architectures, the agent knows only that when a certain condition holds it should execute a plan in its entirety to achieve an implicit goal, and if some step of this plan fails, that implicit goal is viewed as impossible. An agent operating under this model has limited flexibility and no actual autonomy since its only concern is to carry out plans regardless of their implications to higher-level objectives. In this report we argue that agent autonomy is closely linked to the declarative nature of goals and the agent ability to reason about the importance of goal achievement as well as the implications of plan and goal failure. We review work on motivations and argue that a motivations model can be used to direct autonomous agent behaviour and drive the agent to adopt or to drop goals. We also review work on multiagent interactions aiming to investigate how social behaviour may be initiated by an agent following declarative goals. The literature thus reviewed allows us to point out a series of issues relating to the construction of practical architectures of declarative agents.

Contents

Acknowledgements **xi**

1 Introduction **1**

2 Declarative Agents **3**

2.1 BDI Agents . . . 3

2.1.1 BDI architectures . . . 5

2.1.1.1 IRMA: The embryonic BDI architecture . . . 5

2.1.1.2 PRS: Procedural Planning . . . 7

2.1.1.3 PRS Descendants . . . 11

2.2 Frameworks for Declarative Goals . . . 13

2.2.1 X-BDI . . . 13

2.2.2 GOAL . . . 15

2.2.3 Dribble . . . 16

2.2.4 Extensions to the GOAL/Dribble family of agents . . . 17

2.3 Motivated Agency . . . 18

2.3.1 Motivations . . . 19

2.3.2 Models of Motivated Agency . . . 20

2.3.2.1 Alarms . . . 20

2.3.2.2 $_3M$ Motivation Taxonomy . . . 21

2.3.2.3 SMART . . . 24

2.3.3 Motivation-driven Agent Architectures . . . 25

2.3.3.1 Motivated Blackboard . . . 25

2.3.3.2 Motivations in InteRRaP . . . 25

2.3.3.3 Creatures . . . 26

2.3.4 Motivation-based planning . . . 26

2.4 Discussion . . . 26

3 Agent Interaction **29**

3.1 Agreement Mechanisms . . . 30

3.1.1 An abstract negotiation framework . . . 30

3.1.2 Argumentation-based negotiation . . . 32

3.2 Coordination Mechanisms . . . 34

3.2.1 The Contract Net Protocol . . . 35

3.2.2 Commitments and Mental States . . . 36

3.2.3 Coalitions and Organizations . . . 37

3.2.3.1 Coalitions . . . 37

3.2.3.2 Payoff 38
3.2.3.3 Virtual Marketplaces 39
3.2.4 Multiagent Planning 39
3.2.4.1 Partial Global Planning 41
3.2.4.2 Generalised Partial Global Planning 41
3.3 Regulatory Mechanisms 42
3.3.1 Norms 43
3.3.1.1 Deontic logics 44
3.3.1.2 Norms for autonomous agents 45
3.3.2 Policies 46
3.3.3 Reputation and Trust 47
3.3.3.1 Trust mechanisms 47
3.3.3.2 Reputation of information sources 49
3.3.3.3 Reputation through social analysis 49
3.3.3.4 Automated relationship identification 50
3.3.3.5 Reputation as probability 51
3.4 Discussion 52

4 Issues regarding declarative agent architectures 55

5 Future Work 59
5.1 Activities and Deliverables 59
5.2 Work Plan 60
5.3 Contributions 60

Bibliography 61

List of Figures

2.1 IRMA architecture [Bratman et al., 1988]. 5
2.2 PRS architecture structure. 8
2.3 A PRS knowledge area. 9
2.4 X-BDI operation overview. 14
2.5 $_3M$ Motivation model. 23
2.6 A control loop driven by $_3M$ motivations. 23

3.1 Elements of an abstract negotiation framework. 31
3.2 Bartolini's abstract negotiation process. 32
3.3 Components of an ABN mechanism. 34
3.4 Coordination Mechanisms. 35
3.5 Steps in the contract net protocol. 36
3.6 Regulatory Mechanisms. 43
3.7 Dependencies among agent-interaction mechanisms. 53

4.1 Conceptual architecture for motivationally controlled delegation. 56

List of Tables

2.1 Human motivations from [Morignot and Hayes-Roth, 1996]. 21

Acknowledgements

Thanks to Mike Luck for giving me such a hard time in the writing of this report and allowing its content to be so much better than I would otherwise be able to write. I attribute to him any overly well-written passages, any mistakes are my own.

Chapter 1

Introduction

As computer systems become more complex, abstraction mechanisms have become more and more important. One abstraction mechanism that is increasingly becoming accepted is the notion of autonomous agents [Jennings, 2000], which embody a component independent from direct external control, and which are expected to operate unsupervised for an undetermined amount of time. Research into agent systems has yielded an extensive body of work in terms of both theoretical results and practical models and systems, and is still ongoing. In the particular area of *autonomous* agents, the theoretical work has not always been matched by the creation of practical models. Despite this difficulty, there have been many applications requiring autonomy; for example, control systems for space exploration vehicles, and for operation in hazardous environments. These applications are often specifically designed for their application domain and require regular human intervention.

For agent designers to expect an agent to act autonomously, they must be able to specify *what* the agent must accomplish and allow for the agent's reasoning process to select the best way of accomplishing it. In contrast, current practical agent models require the designer to provide detailed specifications of *how* an agent should achieve its goals (*e.g.* create a plan library), as well as precise descriptions of the conditions under which an agent should pursue their achievement. In this setting, the success or failure of a goal is implied by the successful execution of these plans, so that when the agent selects one of the available plans, it has no way of determining how appropriate that selection is until the plan has either succeeded or failed. Given the association of plan execution with goal achievement, the way in which an agent's plan library is defined might interfere with the agent's perception of which goals are possible or not. Recent work on declarative agent languages has partially addressed the problem by dissociating plan execution from goal achievement, and allowing the agent to try other plans to accomplish the same goal in case the currently chosen plan fails. However, agents still rely on a plan library and lack any kind of knowledge about the plan's suitability for a given situation until that plan has been achieved, so that an agent has no way of evaluating a course of action before trying to follow it. This lack of knowledge regarding the courses of action to follow applies also in the context of multiagent systems. That is, an agent never evaluates whether interacting with other agents is actually required, relying on a set of hard-coded rules specifying when it should go into *social mode*.

Normal living beings constantly evaluate their surroundings and their past experiences to decide their courses of action, and changes to ongoing courses of action are rarely sudden and arbitrary, reflecting a mechanism that is much more elaborate than simply obeying a set of rules or maximising rewards. Therefore, we believe that for autonomous behaviour to be possible, an agent has to be able to evaluate its available courses of action before investing any resources in pursuing them. For an agent to perform this kind of evaluation, it must be supplied not only with a set of design objectives, but also with a mechanism that allows it to evaluate how the importance of accomplishing individual objectives changes in response to events in the world.

This report reviews research efforts aimed at the development of autonomous agent systems, including research on autonomous and motivated agent models in Chapter 2, as well as agent interaction mechanisms in Chapter 3. In our review we identify outstanding issues on the integration of the individual results achieved by previous research in Chapter 4 in order to define a plan for future work in Chapter 5.

Chapter 2

Declarative Agents

2.1 BDI Agents

One commonly used way of informally describing autonomous behaviour is using the notions of beliefs, desires and intentions, in which beliefs describe one's knowledge about the world, while desires are states of affairs one seeks to achieve and intentions are one's commitment to achieving a particular subset of desires. This model was proposed for human practical reasoning by philosopher Michael Bratman [Bratman, 1984], to account for the way in which humans select a series of actions directed at the achievement of a larger goal while avoiding spending time considering less important ones. The BDI model serves as an architecture for intelligent agents [Bratman, 1987], using the same mental abstractions of beliefs desires and intentions to describe the operation of software programs. This BDI architecture has become one of most widely known and studied models of deliberative agents, and evolved from Bratman's seminal work [Bratman, 1987] into formalisations [Cohen and Levesque, 1990] and subsequently a more complete computational theory [Rao and Georgeff, 1995b; Wooldridge, 2000a].

The components that characterise the BDI model can be briefly described as follows [Müller, 1996].

- **Beliefs** are the agent's model of the current world, as perceived by its sensors, including the knowledge an agent has about how to modify the world.
- **Desires** are a (possibly inconsistent) set of preferences regarding world states.
- **Intentions** represent the agent's choice regarding alternative courses of action, constraining the consideration of new objectives to allow it to fulfill one subset of its desires at a time.

In essence BDI agents operate as follows.

1. The agent's beliefs are constantly updated by its sensors.
2. The agent chooses an internally and externally consistent subset of its desires. Internally consistent desires are those that do not conflict with each other, like being in two places at the same time. Externally consistent desires are those that are not deemed impossible given the agent's beliefs and are not already satisfied.
3. Considering the chosen desires, the agent commits itself to following a course of action (or intention) to fulfill them.

The BDI model has been the focus of agents research for a significant time [Georgeff et al., 1999], and is still ongoing. Examples of recent research include improving the model through the construction of new theories to underpin it as a unified system [der Hoek and Wooldridge, 2003; van Riemsdijk et al., 2005], and extending pre-existing BDI theories to allow ever more complex domains to handled by BDI agents [Bordini et al., 2003; Móra et al., 1999; Nair et al., 2003; Nide and Takata, 2002]. Among these efforts, many seek to address the fact that BDI architectures and models tended to avoid including many of the declarative aspects of desires/goals in support of practicality. More specifically, the first instances of complete BDI logics [Rao and Georgeff, 1995b] assumed an agent able to foresee all of the future ramifications of its actions as part of the process of deciding which courses of action to take [Schut and Wooldridge, 2001]. This assumption was clearly too strong if computationally-bounded BDI architectures were to be constructed. Therefore, when designing practical architectures based on specific BDI logics, modifications were necessary to avoid unbounded computations. Since the agent cannot look directly into future world states and then select the sequence of actions that leads to the desired future (as this would imply omniscience), the inverse approach was taken; that is, an agent would select from a set of known courses of action, the one that would lead to the desired future. In practice, this means that the agent no longer selects directly what he wanted to achieve, but rather what he wants to perform under the assumption that his actions would ultimately bring about the desired state of affairs. This way of selecting agent goals was later dubbed goals *to do* [Winikoff et al., 2002].

Concurrently with goals *to do* are what have been termed goals *to be* [Winikoff et al., 2002]; the difference being that here, an agent selects the desired state of affairs directly. As a consequence, the actions required by the agent to reach such a state of affairs are decoupled from the ultimate goal. In turn, this gives rise to the problem of discovering which actions will have to be taken by the agent to realise its goals, a problem which is sometimes referred to as the *agent design problem* [Wooldridge, 2000b]. The most widely known BDI agent implementations bypass this problem through the use of plan libraries where the courses of action for every possible objective an agent might have are stored [d'Inverno and Luck, 2004; Ingrand et al., 1992; Rao, 1996] (which we have seen are associated with *to do* agents). The near absence of pragmatic architectures that implement the notion of *to be* goals represents a gap that current research is trying to address.

2.1.1 BDI architectures

2.1.1.1 IRMA: The embryonic BDI architecture

The Intelligent Resource-bounded Machine Architecture (IRMA), was defined to demonstrate the applicability of Bratman's practical reasoning model [Bratman et al., 1988]. IRMA provides a reasoning mechanism for an agent taking into account its limited resources. IRMA was one of the first to incorporate intentions as a primary mental state, playing an important part in the means-end reasoning of the deliberative process. The role of intentions in IRMA is to keep track of the agent's progress in achieving a goal and constraining future deliberation to avoid new goals that conflict with the ones currently being pursued to be adopted.

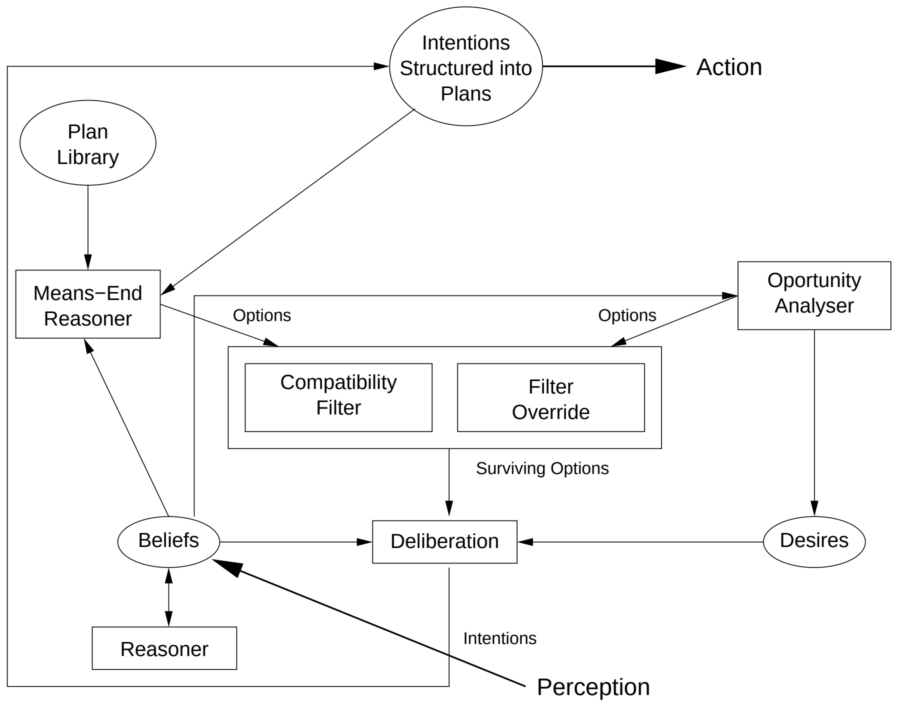


FIGURE 2.1: IRMA architecture [Bratman et al., 1988].

Figure 2.1 represents IRMA's architecture and contains two basic types of entities: processes (denoted by rectangles) and storage entities (denoted by ellipses). IRMA intentions are structured into high-level plans in which actual plans (or plans as recipes) are stored in a plan library. The *opportunity analyser* reacts to changes in the environment, creating action options based on events that were not predicted by conventional planning, which is performed by the *means-end reasoner*. Action options are alternative means through which the desired goals of an agent can be accomplished. The *means-end reasoner* has as its most obvious inputs the *beliefs* and the plans stored in the *plan library.* Also, according to Bratman's practical-reasoning model, the role of intentions in means-end reasoning is to narrow down the search space for its problems. The *means-end reasoner* and the *opportunity analyser* come up with options for the filtering process, represented by the *compatibility filter* and by the *filter override.* The *compatibility filter* checks if the generated options are consistent with the intentions currently adopted, and the surviving options are passed to the *deliberation* process, which assesses new options and incorporates them into plans. The *filter override* was included in the filtering process due to the possibly limited agent knowledge, which creates situations in which the consideration of certain options would be interesting despite an indication by the beliefs that these options are inconsistent. Therefore, even when a given option is eliminated by the *compatibility filter* it is possible that it might trigger a rule in the *filter override* causing it to survive.

Despite IRMA being a highly abstract architecture, it was useful to explore some of the problems that other BDI architectures might have to deal with [Bratman et al., 1988], such as the need for procedures to:

- propose new options when changes in the environment are detected;
- evaluate conflicting options; and
- override the compatibility filter.

Some of the concepts outlined in the original IRMA work were put to the test in the Tileworld system [Pollack and Ringuette, 1990], whose main objective was to provide an agent architecture testbed for meta-level reasoning strategies. Essentially, the Tileworld system consists of a simulated robot agent and a simulated environment, which is dynamic and unpredictable. Both environment and agent were designed to be highly parameterised so that various situations in which an agent might find itself could be tested, and the behaviour of these agent/environment pairs could be evaluated. The main components under scrutiny in the Tileworld agent were the *filtering mechanism*, comprising the *compatibility filter* and the *filter override*, which is responsible for deciding whether a change in the environment should cause a reconsideration of the agent's current intentions. Various deliberation strategies of increasing complexity were tested against different environment set-ups, which varied in several dimensions, in order to test their suitability. The experiments conducted over the Tileworld testbed demonstrated that more restrictive filtering mechanisms that allow only clear opportunities (*i.e.* tasks that can be accomplished with little planning and achieve an immediate benefit) to be passed down to a deliberation process are more desirable when the environment is more dynamic.

These results are often analysed using the notion of cautiousness and boldness [Bratman et al., 1988] where a cautious agent reconsiders its course of action frequently while a bold agent does not. Here a bold agent tends to perform better than a cautious one in a dynamic environment [Pollack et al., 1994], because the constant reconsideration triggered by changes in the environment prevents the agent from carrying out plans to their completion, diminishing the number of goals effectively achieved. Although these conclusions conform to the hypotheses outlined in earlier work [Bratman et al., 1988], Pollack et al. [Pollack et al., 1994] are careful not to assert their generality regarding real-world applications, given that the environment in which the agent was embedded was highly controlled.

2.1.1.2 PRS: Procedural Planning

The Procedural Reasoning System (PRS) [Georgeff and Lansky, 1987] was created as a BDI architecture that could be used in real-world applications, and was aimed at supporting both goal-directed and reactive reasoning. It was first used in the implementation of a task control system for a NASA spacecraft simulator.

A PRS agent or module consists of four components, shown in Figure 2.2:

- a *database* containing the current system's beliefs about the world;
- a set of current *goals*;
- a procedure or plan library (*knowledge area (KA) library*); and
- an *intention structure*.

KAs describe action and test sequences intended to achieve the proposed goals or to react to specific situations [Ingrand et al., 1992] while the *intention structure* maintains the set of plans chosen at runtime for execution. PRS integrates these components via an *interpreter*, which works as an inference mechanism that manipulates them and selects an adequate plan based on the system's beliefs and goals, putting this plan in the intention structure and executing it.

System Database (Beliefs). The System Database can be viewed as the representation of the system's beliefs regarding the environment, represented as first-order predicates [Georgeff and Ingrand, 1989a]. These beliefs may initially contain constant properties regarding the world and application domain of a PRS system, though they can evolve through system execution to include the agent's observations about the world, or even conclusions derived by the system using the knowledge contained within the database. The system database is also the way through which the agent receives information

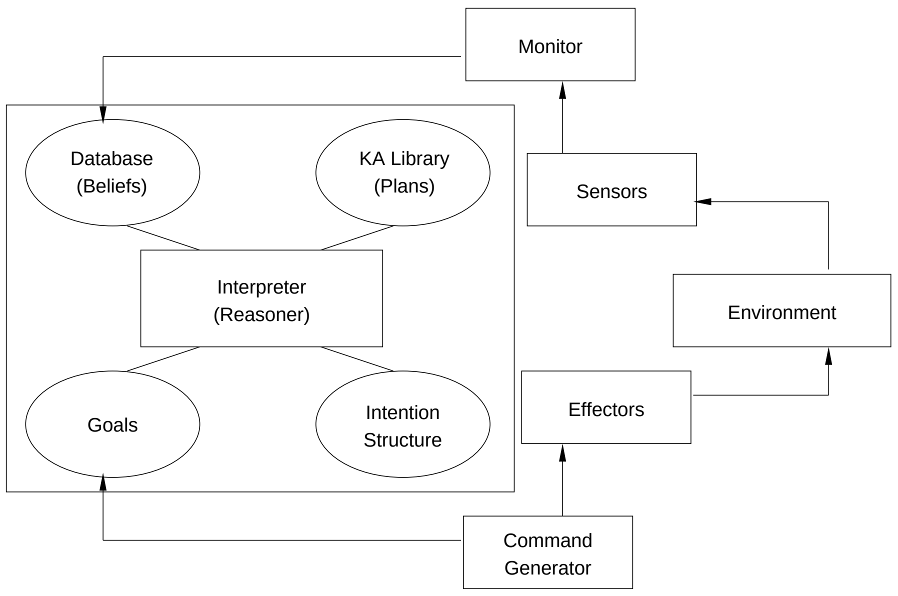


FIGURE 2.2: PRS architecture structure.

regarding the world [Ingrand and Coutance, 2001]; in PRS it is assumed that there is an automatic process that includes new facts coming from the environment. In addition to describing the world, the agent's beliefs may refer to the agent's structure and its internal state, including beliefs, goals and intentions. Such reflective beliefs are expressed as *meta-level* expressions, which describe properties concerning an agent's internal state, as well as operations affecting the agent's behaviour, as opposed to standard expressions, which only deal with the surrounding environment. These types of expression provide the agent with introspective capabilities.

Goals (Desires). PRS goals describe tasks and desired behaviours such as [Georgeff and Ingrand, 1989b; Ingrand et al., 1992]:

- reaching a given condition;
- testing a given condition;
- waiting until a given condition is true;
- maintaining a true condition;
- asserting a condition as being true ;
- retracting a condition; and
- concluding that a given condition is true.

Goals are divided into two types: *intrinsic* and *operational*. Intrinsic goals are adopted as the agent reacts to changes in the environment and become roots of intentions in the intention structure. Operational goals represent intermediate steps in the process of fulfilling an intrinsic goal, they can be seen as *means* to achieving intrinsic goals. As with beliefs, PRS allows goals to be defined as *meta-level*, thus allowing the specification of goals regarding internal system behaviour.

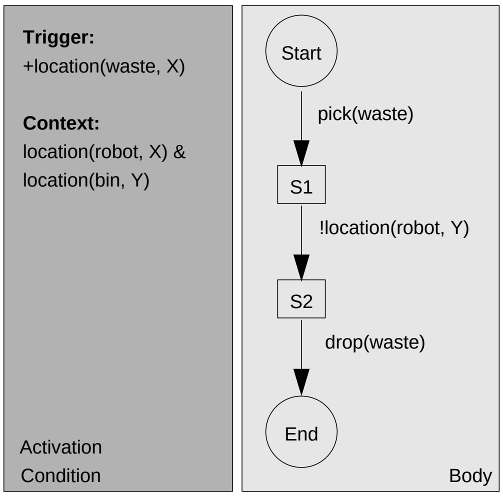


Figure 2.3: A PRS knowledge area.

Knowledge Area (Plans). The knowledge of how to achieve a given objective in PRS is described by declarative procedure specifications called *knowledge areas* (KAs) [Georgeff and Ingrand, 1989a,b], which use the notion of procedural knowledge [Georgeff and Lansky, 1986] in the sense that the agent's knowledge about the workings of its surrounding world is encoded in pre-defined plans used to achieve its goals. KAs are represented by a *body* and an activation condition, specifying a sequence of steps to achieve a given objective in a given situation. A KA body can be viewed as a plan or a plan schema, and is represented by a directed graph with a start node and one or more finish nodes (Figure 2.3). The graph's arcs are labelled with sub-goals to be achieved throughout plan execution, so that he execution of a KA is said to be successful when the arcs that connect a start goal to a finish goal are reached and, through this path, all the specified sub-goals are satisfied. This means that a graph path in a KA is actually multi-dimensional, given that satisfying sub-goals may require other KAs to be executed and whenever a subplan is instantiated differently a new dimension is added to the intention structure. It is possible that some KAs do not have a body, in which case they are called primitive KAs, because they have some kind of primitive action tied to them that is directly executable by the system. Note that the KA graph construction formalism allows the use of various constructs that control the execution flow, such as conditional branches, iterations and recursions.

The invocation condition of a KA is divided into two parts: the *triggering part* and the *context part* [Georgeff and Ingrand, 1989a].

- The triggering part of an invocation condition is a logic expression that describes the events that must take place in order for the KA to be executed. These events may consist in the acquisition of new goals, in which case the reasoning is goal oriented, or the modification of the agent's beliefs, in which case the reasoning is data-oriented or reactive.
- The context part of an invocation condition specifies the conditions that must be true regarding the current system state for the associated KA to be executed.

KAs are not limited to dealing with the environment that surrounds the agent, and can also be used to manipulate beliefs, desires and intentions of PRS itself. These KAs are therefore called *meta-level* KAs [Georgeff and Ingrand, 1989a; Ingrand et al., 1992]. One of the objectives of such KAs is the modification of standard PRS interpreter behaviour in dealing with reasoning. For example, this might include the modification of plans during execution, the establishment of new goals or even the modification of beliefs during the execution of a meta-level KA.

Intention Structure. The intention structure contains the tasks that the system has chosen for immediate or later execution; these tasks are called intentions [Georgeff and Ingrand, 1989b]. An intention is composed of a KA chosen to fulfill a goal, along with all the KAs needed for the initial KA to be completed. Intentions in the intention structure may be active, suspended or delayed (for example, waiting for a condition to become true). For every goal being pursued there is an intention, and that for each intention there is a stack of KAs to be executed in order to achieve the KA.

KAs in the intention structure are partially ordered, with possibly more than one KA as the root of the intention structure. The order established for the executing KAs is followed so that for a given KA to be executed it is necessary for all the preceding KAs to be either executed or dropped. Meta-level KAs are treated in the exact same way as regular KAs in the intention structure, ensuring a bounded response time regardless of the type of KA being processed. Once a given KA is chosen to fulfill a goal, no other KAs are selected to fulfill the same goal, even if their activation condition is satisfied [Georgeff and Ingrand, 1989a]; other plans will only be considered for achieving any given goals once the current plan fails.

System Interpreter. The interaction between the components of PRS is controlled by a simple interpreter in order to attain a minimum reaction time for PRS agents[Ingrand and Coutance, 2001]. Considering the existing desires and beliefs at a given moment, one or more KAs may become eligible for execution, and one or more of these KAs is chosen for inclusion in the intention structure (*i.e.* chosen for execution). In order for the interpreter to determine whether to execute the KAs, the interpreter only tries to unify the KA execution condition with the system's beliefs. If any other more complex inference process was used, it would not be possible to prove a bounded execution time for the KA selection process [Georgeff and Ingrand, 1989a]. Nevertheless, meta-level KAs can be used to create more complex inference processes [Georgeff and Ingrand, 1989a]. The use such of meta-level KAs does not violate the reactive capability of the system, since these are handled in the same way as any other KAs, allowing for new KAs to take precedence over the meta-level KAs used to make complex inferences.

2.1.1.3 PRS Descendants

PRS has been used to underpin a variety of implementations of the BDI agent model, and it has also evolved to a number of other systems aiming to solve some of its original shortcomings. Two of the most notable direct PRS descendants are dMARS and AgentSpeak(L), whose main objectives were respectively, to create an idealised C++ implementation of PRS, and to formally define an agent specification language and its semantics. These efforts are briefly described below, as well as more recent systems, such as JACK, Jason and JAM.

dMARS. The distributed Multi-Agent Reasoning System (dMARS) [d'Inverno et al., 1998] is a PRS implementation that uses the same representation of beliefs as PRS, consisting of Prolog-like ground literals of classical first-order logic. dMARS reduces the amount of possible goal types from the original seven in PRS to achievement goals, query (test) goals, and distinguishes the assertion and retraction goals present in PRS as *internal actions*, naming them, respectively, `add` and `remove` actions. It also formalises agent interaction with the environment through the notion of *external actions*, which are used to perform some arbitrary operation defined by the system programmer. In the same way as the original PRS, an agent's intentions in dMARS are represented by the currently adopted plans, though the events that trigger the adoption of a new intention have been expanded from the original addition and removal of beliefs and adoption of a new goal also to contain the receipt of a message. The definition of plans in dMARS is performed through a textual version of PRS plan graphs, and the two components of the PRS invocation condition were refined to the notions of relevance and applicability. One of the greatest contributions of the dMARS formalisation is the unambiguous definition of an interpreter algorithm.

JACK. JACK is a commercial agent framework aiming to provide a high-performance infrastructure designed to be a part of a larger legacy system [Busetta et al., 1999; Howden et al., 2001]. This framework itself is implemented in Java, while the agents are defined in an extended version of the Java language that includes new constructs for the specification of agent components. A special Java compiler is used to convert the extended subset of the base programming language into pure Java code that can be accessed by the rest of the system. Besides the resulting code describing agent behaviour, a deployed system contains a set of runtime support classes that provide an agent infrastructure management for the deployed agents.

JAM. JAM [Huber, 1999] is an implementation of PRS that incorporates features from several frameworks, like Structured Circuit Semantics (SCS) [Lee et al., 1994] and the Act plan interlingua, also present in independently implemented PRS descendants like UMPRS [Lee et al., 1994] and PRS-CL [Wilkins and Myers, 1995]. Its main purpose is to gradually integrate successful elements from other agent architectures as a means to further develop the JAM framework. In its current release it includes mobility capabilities; the next planned development step was to be the incorporation of generative planning capabilities. Such a capability will include a partial order planner for the generation of plans whenever the plan library fails to find an appropriate plan for a given situation.

AgentSpeak(L). The AgentSpeak language [Rao, 1996] was created in order to diminish the distance between BDI agent theory and practice. This gap is due to a number of factors [d'Inverno and Luck, 1998; Rao, 1996]: for example, implementations are generally conceived in a simplified manner, resulting in the weakening of its theoretical underpinning, and the logics used in the associated theories are usually weakly related to practical problems. The goal to diminish the gap was to be attained through a formal specification of the agents, later used to underpin their implementation. AgentSpeak(L) is an agent specification language whose operational model is formally defined to match that of a dMARS implementation. The agent programming language is based on a restricted first-order language with events and actions, in which BDI components such as beliefs, Desires and Intentions are not explicitly represented as modal formulas. Similarly to dMARS, AgentSpeak(L) is based on a number of simplifications over PRS, the most significant of which are the following.

- **Goal Types:** In PRS it is possible to specify the following goal types [Georgeff and Ingrand, 1989b; Ingrand et al., 1992]: to reach or test a condition, to wait for a condition to be true or to maintain true the condition, to assert or retract the truth value of a condition and to conclude that the condition is true. In AgentSpeak(L) it is possible to declare goals of achievement and testing of a given condition over the world, as well as to assert and retract conditions over the world through AgentSpeak(L) basic actions. This results in diminished expressivity at the language tied to AgentSpeak formal model.

- **Meta-level components:** In AgentSpeak(L) it is not possible to specify meta-level components such as those that are possible in PRS, thus limiting behaviour flexibility in the definition of a given agent.

Jason. Jason is a Java implementation of a modified version of an AgentSpeak(L) interpreter, AgentSpeak(XL) [Bordini et al., 2003]. AgentSpeak(XL) specifications are tailored to allow model checking to be used to verify a system prior to its deployment. The available Jason implementation includes features such as strong negation and distribution of processing using a multi-agent framework.

InteRRaP. Another architecture that follows the PRS philosophy of procedural goals is the InteRRaP (Integration of Reactive Behaviour and Rational Planning) architecture [Müller, 1996]. InteRRaP is not closely related to any formal theory but innovates through its layered architecture in which single-agent PRS-like reasoning operates concurrently with reasoning about communication and cooperation.

2.2 Frameworks for Declarative Goals

Implementations of BDI agents have generally focused on runtime efficiency in order to cope with deployment in systems with constrained computational resources. This has led the first generation of BDI systems (usually based on the PRS [Ingrand et al., 1992] model) to largely overlook the declarative nature of goals in BDI agent implementations in favour of procedural goals [Winikoff et al., 2002]. We believe that declarative goals were perceived to be computationally inefficient as a result of how a BDI decision procedure was initially formalised. In this formalisation, an agent was required to know all of the possible attainable world-states and the payoff of achieving them before deciding on a course of action [Rao and Georgeff, 1995b]. Constructing this kind of omniscient model of possible worlds incurs a high computational cost. Nevertheless, much work has recently been developed in order to bridge the gap between pragmatic BDI architectures and declarative goal semantics. These efforts focus on new formal agent models such as those by Hindricks *et al.* [Hindriks et al., 2000], Braubach *et al.* [Braubach et al., 2004] and van Riemsdijk *et al.* [van Riemsdijk et al., 2004, 2005]; as well as new agent systems and languages such as X-BDI [Móra et al., 1999], Dribble [van Riemsdijk et al., 2003] and its refinements [Dastani et al., 2003].

2.2.1 X-BDI

Executable-BDI or X-BDI is an agent model constructed using a non-monotonic form of Extended Logic Programming (ELP) [Alferes and Pereira, 1996]. This agent model was created to allow a formal agent specification to be directly executed [Móra et al., 1999], thus reducing the gap between theory and implementation. The implementation relies on ELP's reference implementation that includes a derivation procedure (SLX[Alferes and Pereira, 1996]), used to perform consistency maintenance of the agent's belief base. The same procedure that enforces consistency among the agent's beliefs underpins an abduction planning process [Shanahan, 2000] operating over an event calculus [Kowalski and Sergot, 1986] representation of the agent's actions that is responsible for the agent's means-ends reasoning.

An X-BDI agent is composed of a cognitive structure composed of a set of beliefs, a set of desires and a set of intentions as well as a set of time axioms. These components are used by the X-BDI kernel to define an agent's behaviour, an overview of which is shown in Figure 2.4.

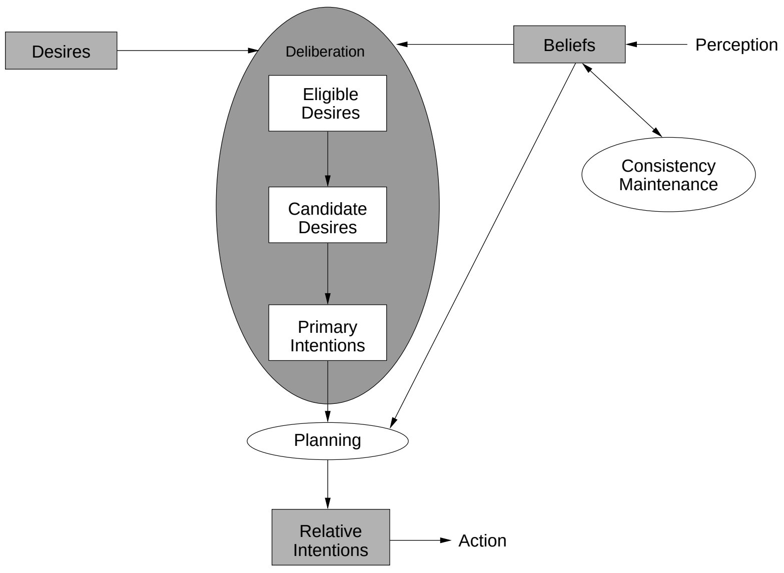


FIGURE 2.4: X-BDI operation overview.

Whenever sensor input is available to the agent, it is used to update its set of beliefs. When this set is changed in such a way that it affects the consistency of the agent's desires, the deliberation process commences. The deliberation process starts by selecting a set of desires deemed viable (called *eligible desires*) considering the set of beliefs. Mutually consistent subsets of eligible desires are then organised by some order relation. A search for the existence of a set of actions capable of fulfilling one of these subsets ensues. When one such set of actions is found, the desires associated with them are selected as the *candidate desires*, which the agent commits itself to achieving by generating a set of *primary intentions*. These intentions are refined into concrete actions (comprising the *relative intentions*) queued for execution, which will ultimately lead the agent to act.

2.2.2 GOAL

GOAL is a declarative programming language [Hindriks et al., 2000] developed in a similar spirit as X-BDI (*i.e.* to bridge the gap between theory and practice in agent programming), but unlike X-BDI, it was developed without recourse to an existing logical scheme. GOAL was inspired by the UNITY concurrent programming logic [Misra and Chandy, 1989], which expresses programs as actions that execute in parallel, assigning values to variables. These actions are invoked randomly until there is no possible action that will bring about a change in the state of the variables; unlike UNITY, agents in GOAL select their actions on the basis of their current mental states. A mental state is a pair containing the beliefs and the set of goals of the agent. These components are also subject to a set of constraints:

- an agent cannot have a goal to achieve something if it believes that such thing is already true; and
- no goal in the goal base can be inconsistent, *i.e.* no formula in the goal base is allowed to support contradiction.

Conditions on mental states are expressed by a language of mental state formulas, which expresses boolean combinations of the basic formulas over beliefs (formulas believed to be true) and goals (formulas that are goals). Besides beliefs and goals, a GOAL agent has a set of capabilities expressing actions that update the agent's belief base. A GOAL agent is therefore expressed as a triple, consisting of a specification of an initial mental state (beliefs and goals) along with a set of actions derived from the agent's capabilities. The process of action adoption is based on the evaluation of pre-conditions associated with the capabilities, and the actions are then executed in the process of fulfilling the agent's goals. The default commitment strategy of a GOAL agent is that of *blind commitment*, which can be manipulated by the use of two special actions to *adopt* and *drop* goals.

The idea behind capabilities in GOAL is that they are *mental state transformers*, meaning that they map an agent's mental state into another mental state. Moreover, actions can be conditional upon the beliefs as well as the goals of an agent. Conditions on the beliefs are interpreted as pre-conditions for action execution, whereas conditions on the goals specify the applicability of a given action towards the accomplishment of a particular goal. Goal conditions are related to a special formula that states that a given subgoal partially fulfills a goal in a mental state. This is used to guide an agent in the action selection process that causes actions to be carried out, which in turn leads to computation steps, defined as transition relations in which conditional actions are executed causing a new agent state to be generated.

A GOAL agent is thus defined as a set of *traces*, in which a trace is an infinite computation sequence of consecutive mental states and actions performed in those mental states.

Action specification follows the formalism commonly used for planning and conditional assignments in concurrent programming. These actions are defined in advance as a plan library, thus making GOAL agents devoid of planning capabilities.

2.2.3 Dribble

The Dribble language was created to combine features of procedural and declarative goals [van Riemsdijk et al., 2003] by drawing on elements from 3APL [Hindriks et al., 1999] and GOAL [Hindriks et al., 2000]. The means for creating and modifying plans at runtime was inherited from 3APL while the use of declarative goals was inherited from GOAL. The basic operation of a Dribble agent consists of an agent selecting a plan to achieve a certain goal, using the beliefs and goals of an agent along with rules to select, create and modify plans. Plans in Dribble are defined to be a sequence of basic elements, as follows.

- **Basic Actions** are operators that effect changes in the agent's beliefs but have no effect in the real world.
- **If-then-else** are constructs that allow plans to have conditional branches encoded in their execution.
- **Abstract Plans** roughly correspond to the notion of procedural knowledge in PRS in that they resemble procedures in imperative programming, they contain both *basic actions* and *if-then-else* constructs.

A Dribble agent is a quadruple containing an initial mental state (with initial beliefs and goals), an empty plan, a set of goal rules, and a set of practical reasoning (PR) rules. Goal rules specify how to select a plan to satisfy a specific goal, while PR rules specify modifications to the current plan to ensure this plan remains viable. A selected plan is modified at run time as its actions are executed (and thus removed from the selected plan), or through PR rules that can be applied to modify such plan. Since Dribble maintains the blind commitment strategy used by GOAL, a goal rule is applied at the start of an agent's execution and whenever the currently pursued goal is achieved. This encoding of an agent's mental state entails that the agent is effectively single-minded in the sense that it will only be pursuing the one goal contained in the goal rule that used to select the current plan.

Since Dribble uses *propositional* logic in its formulas, the type of problem that can effectively be encoded using Dribble agents is severely limited. Moreover, if the interpreter was extended to handle first-order logic, Dribble agents might not be tractable.

A Dribble agent aims to select plans to further its goals rather than construct plans like a planner; it also has no knowledge about the plans it selects aside the fact that these plans are intended to accomplish a certain goal. This type of plan selection strategy is similar to that of PRS, so the planning component in Dribble cannot evaluate the plan prior to execution to determine if a given goal is possible or not (*i.e.* the agent applies plan modification rules until the plan succeeds or can no longer be modified or carried out).

2.2.4 Extensions to the GOAL/Dribble family of agents

The general model set forth in Dribble has been used as the basis for the analysis of the dynamics of adopting and dropping goal [van Riemsdijk et al., 2004], as well as the semantics of goal databases interaction [van Riemsdijk et al., 2005]. These were all carried out under the framework of the goal and PR rules defined for Dribble agents (Section 2.2.3) so as to override the default blind commitment initially adopted. But, since one of the main advantages of using a declarative semantics for agent goals is the decoupling of plan execution from goal achievement, multiple views of the dynamics of adopting and dropping goal were studied, as well as issues of goal persistency. The agent configuration from previous models was modified with the addition of the currently selected plan, and the rules collapsed into a single component.

Besides syntactic modifications, the transition system that describes system dynamics has been expanded to consider the expansion and contraction of beliefs. Moreover, the semantics of goal base transitions are studied in two distinct views, namely membership of a goal in the set goals and entailment of goals from the same set. The semantics of goal accomplishment was also investigated in terms of the adoption of subgoals that will somehow approximate the agent the associated top goal. Van Riemsdijk *et al.* remark subgoals as being either *parts* (decomposition) of the top goal or *landmarks* that should be reached on the way to accomplishing the main goal. The difference between these two however, does not seem to be clearly identified. The goal decomposition view states that the accomplishment of all subgoals entails the achievement of the top goal, while landmark goals are described as goals whose achievement is necessary for the final achievement of the top goal. Such a view of subgoals is supported in [van Riemsdijk et al., 2004] as a way of enabling an agent to cope with the shortage or absence of plans to fulfill the main goal by using alternative plans to achieve landmarks leading to that goal.

The extensions proposed in [van Riemsdijk et al., 2005] modify the basic agent configuration with the explicit notion of intention, thus approaching a classic BDI view of agent. An agent configuration in this extended model includes a component that encodes intentions as a set of pairs of goals and associated plans. The sets of rules are similar to those of previous extensions to GOAL/Dribble agents, but includes rules for intention generation, thus approaching a BDI-like definition of agents.

A semantics rule for intention generation was also defined so as to avoid the adoption of conflicting intentions. Such a semantics roughly approximates the idea of a screen of admissibility present in IRMA [Bratman et al., 1988]. The rules for intention adoption specify a belief (β) as well as a precondition goal (κ) that causes the adoption of a plan π that satisfies a goal ϕ. It is not clear, however, what is the relationship between ϕ and the precondition goal κ. Such a notion of intention relaxes the constraint set upon Dribble agents that they must be consistent. It also partially solves the issue of single-mindedness from which previous agent models from this series suffered. Ultimately, no rule for deciding what to do next was set forth, hence leaving the system non-determinism still in place and denying the agent the ability to prioritise the accomplishment of his goals.

2.3 Motivated Agency

In the agent architectures described in Section 2.2, an autonomous agent's reasoning is typically driven by the set of its designated goals. In these architectures the main criterion for selecting a given goal by an agent is a set of preconditions. This in turn results in more or less binary behaviour resulting from the fact that these preconditions are essentially triggers for the selection of goals with no indication of how valuable it is for the agent to accomplish the associated goals. Given the lack of a valuation for multiple goals, it can be very hard to predict which set of goals such an agent will pursue at any given time, particularly if the agent has a large goal base, and its goal activation preconditions can be complex. Moreover, the adopted goals have no indication of how relevant they are for the agent's larger purpose.

Simple precondition-based goal selection not only represents an oversimplification of autonomous behaviour specification but, according to Luck *et al.* [Luck et al., 2003], is also misleading about how to achieve autonomy in agent architectures. More specifically, most of the models described in Section 2.2 claim to be models of autonomous agents, despite the fact that among their components there is no identifiable source of autonomy. The absence of an explicit component responsible for autonomy implies that these architectures assume it to emerge from agents when they are sufficiently independent from user interaction. In this sense, autonomy is a consequence of an agent's independence of others in its actions. Instead, the notion of autonomy advocated by [Luck et al., 2003] is that autonomy derives from independence of choice rather than interaction, so that an agent can depend upon others to carry out its tasks without relinquishing its autonomy. This view is exemplified by the fact that it is very straightforward to construct a web agent whose sole responsibility is to deliver documents over the web, since it does not depend upon any others to fulfill such a task. Yet this activity does not involve any *choice* of alternate courses of action, and the agent is essentially carrying out an imperative *script* defined at design time. On the other hand, an agent operating in an electronic marketplace whose purpose is to buy and sell assets for profit may depend upon a multitude of agents to carry out its task. Its behaviour is directed by the motivation to profit from the marketplace and its members, yet the agents with whom it interacts in no way take over its behaviour. Though a mechanism for goal prioritisation is available in some architectures like X-BDI (Section 2.2.1), the mechanism is static in the sense that goal priorities are not allowed to change during agent execution.

2.3.1 Motivations

A psychology-inspired definition states that *a motivation represents an individual's orientation towards particular classes of goals* [Morignot and Hayes-Roth, 1996]. Such a definition, while broad, captures the fact that motivations are not necessarily tied to a specific set of goals, nor do they directly cause them to be adopted or dropped. That is to say that more than one specific motivation might be associated with a single or multiple goals, and they are not directly responsible for their adoption; rather, they create a setting in which adopting their associated goals is more likely.

From an ethological point of view, motivation is associated with *drives* and *incentives* [Balkenius, 1993; Luck and d'Inverno, 1998; Munroe et al., 2003]. In a simplistic description, *drives* are internally generated states resulting from the violation of an animal's homeostasis, such as the deprivation of food or the excess of a given hormone. *Incentives*, on the other hand, are externally generated stimuli that increase certain motivations within the animal, as in the presence of abundant food causing an animal to feed [Balkenius, 1993]. Motivations have also been described as giving rise to a continuum of appetitive behaviours leading to consummatory ones. This means that some behaviours result in the build up of strength of certain motivations related to appetitive behaviour, and when a motivation has reached a high enough level, consummatory behaviours for the mitigation of the motivation are triggered. Motivations can also be used to prevent certain undesirable behaviours from occurring simultaneously, *lateral inhibition*) [Grand and Cliff, 1998; Jennings et al., 2006], as in trying to look at a watch while holding a mug with the same hand.

The aspect of motivations most commonly sought to be captured by computational architectures is the continuous representation of priorities as a means to determine the *focus of attention* at any given time. This is important as it allows an agent with limited resources to concentrate its efforts on achieving goals that are relevant to it at specific moments, and to adapt such concentration of effort to the current reality. Contrasting with the traditional process of goal selection based solely on environmental state, real biological systems often generate different plans of action under the same environment. Hence, motivations provide a mechanism to model how internal *cues* explain goal generation in parallel with external factors [Jennings et al., 2006]. An internal cue can be seen as a trigger condition that, when activated, causes an agent to consider the adoption of a set of associated goals. It differs from the simple logical preconditions discussed previously in that internal cues are a result of the dynamics of motivation strength, rather than a simple binary condition over the current state of the world.

Based on the concept of explicitly represented autonomy, research on *motivational* states to guide the reasoning process has been conducted by an increasing number of efforts. These efforts range from agent architectures specifically underpinned by motivational states [Luck et al., 2003; Munroe et al., 2004; Norman and Long, 1995] to the adaptation of existing architectures to cope with motivated behaviour [Burt, 1998; Coddington, 2001; Morignot and Hayes-Roth, 1996].

2.3.2 Models of Motivated Agency

2.3.2.1 Alarms

The Alarms architecture allows agents to generate goals asynchronously to focus resources on the accomplishment of important goals [Norman and Long, 1995]. Asynchronous goal generation entails that new goals can be generated before current ones are accomplished, so that it is possible for an agent to adopt more goals than it can effectively work on at the same time. Adopted goals require processing resources for scheduling and planning, and since any agent has a limit on its processing resources regardless of its efficiency, there must be an upper bound for the number of goals it can pursue simultaneously. When this bound is exceeded, the agent will no longer function effectively. The Alarms architecture defines a motivational mechanism that limits the number of goals that can be pursued at the same time [Norman and Long, 1995]. An agent in this model has three basic components: *motives*, *motivated goals* and *motivations*. The choice of nomenclature for the components in this module is rather confusing, in particular with regard to the difference between motivations (which are evaluation functions) and motives (which are triggers for the generation of goals).

- **Motives** are responsible for causing an agent to act by monitoring the environment and the internal state of the agent, and generating goals to satisfy a particular interest represented by the motive. Formally a motive is a function that maps beliefs to a possibly empty list of motivated goals.
- **Motivated Goals** are generated by motives. Each motivated goal is a tuple consisting of a goal and an associated motivation.
- **Motivations** are heuristic functions whose purpose is to determine the motivational intensity for any given motivated goal. The heuristic nature of motivations aims to ensure that the computational effort required to calculate its intensity is

Class	Interpretation
Physiological	Preserve battery level
Safety	Avoid stairways and hostile agents
Affiliation	Achieve other agents' goals
Achievement	Achieve its own goals
Self-Actualisation	Cover unknown areas

Table 2.1: Human motivations from [Morignot and Hayes-Roth, 1996].

negligible compared to the other processes within an agent's deliberative process. Formally, a motivation is a function of beliefs into an intensity value.

A motivated agent performs two distinct functions, goal generation and goal activation, rather than simply responding to a conjunction of beliefs. Rather than using goals to direct agent behaviour, Alarm agents are guided by a number of motives that generate motivated goals as a result of changes in the environment. Goal generation is then a function of motives and beliefs that generates motivated goals.

Activated goals are considered by the agent in its deliberative process, *i.e.* spending computation resources on planning for its achievement and executing the actions resulting from this planning. A goal is activated if the motivation associated with it exceeds a certain threshold, and if the agent decides to act upon that goal. When the motivation threshold of a motivated goal is triggered, the goal associated with it is considered for planning by the agent. Hence, for a goal to be acted upon, it must have been activated, and once activated, it must have been selected by the deliberative process to be included in the planning. Motivation thresholds are dynamically modified to reflect the load of an agent, and therefore prevent an agent from adopting too many new goals when its capacity is nearing exhaustion.

2.3.2.2 $_3M$ Motivation Taxonomy

Considering that the main purpose of motivations is to orientate an individual towards particular classes of goals, it is important to identify different classes of goals. One possible classification of motivations from [Morignot and Hayes-Roth, 1996] is shown in Table 2.1.

While the taxonomy of Table 2.1 makes sense in a human setting, it contains redundancies that make its application to agent architectures unclear. More specifically, in order for classes of motivations to be more than an interesting philosophical proposition, there must be a well-defined role that each specific class of motivation plays within the reasoning of an agent. For instance, goals generated by safety motivations might take precedence over any other goals, but in this case, physiological motivations like preserving battery level are just as important, in which case, these two classes of motivation could be reduced to a single one. Moreover, if motivation classes are to perform a meaningful role in an agent's deliberation process, then their specification must also be included in the design of the agent they are to control. In this case, a simple but comprehensive motivation taxonomy is necessary for a viable design process.

More recent work proposes a taxonomy for motivations called $_3M$ that comprises three sets of motivations [Munroe and Luck, 2003], as follows.

- Domain motivations encode the roles an agent is to fulfill in a system. They should enable the agent to accomplish its purpose by stimulating the generation of relevant goals at relevant times.

- Constraint motivations impose restrictions on the use of resources and the importance placed on their use.

- Social motivations are used to encode regulations on the agent's interactions, for instance by evaluating requests for assistance and the importance of a relationship.

Besides the taxonomy, $_3M$ includes a motivational model that associates motivations to goals in order to influence the decision-making process. In this model, when an agent has to decide between competing sets of goals to be satisfied, it can refer to the motivational model to swiftly decide which set best serves the agent. The proposed model uses the concept of *motivational cues*, which are general characteristics of situations in the environment that affect the motivations. These motivations then steer the agent towards achieving particular goals in a control loop illustrated by Figure 2.6. In order to determine the effect of motivations upon their associated goals, motivations have a strength value that functions as an heuristic calculation of the worth of a given goal to the agent, as illustrated in Figure 2.5, which shows the inputs taken by this heuristic function.

The strength of a motivation not only serves to prioritise one goal over another, it also helps to determine when it is relevant for an agent to adopt a goal at all, so that an agent will not waste its efforts in accomplishing irrelevant goals, even when the agent would otherwise be idle. The influence a motivation exerts on the decision-making process is proportional to the number of cues associated with it since, if a large number of situations cause a motivation to be stimulated, it will in turn activate goals more often. After an agent adopts a goal as a result of an increase in a motivation's intensity, satisfying this goal causes a decrease in intensity of the associated motivation, also called *mitigation* of a motivation. Following its mitigation, a motivation exerts less influence upon the

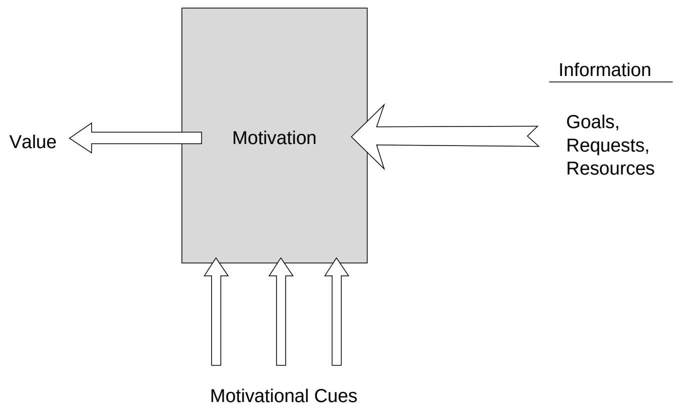


FIGURE 2.5: $_3M$ Motivation model.

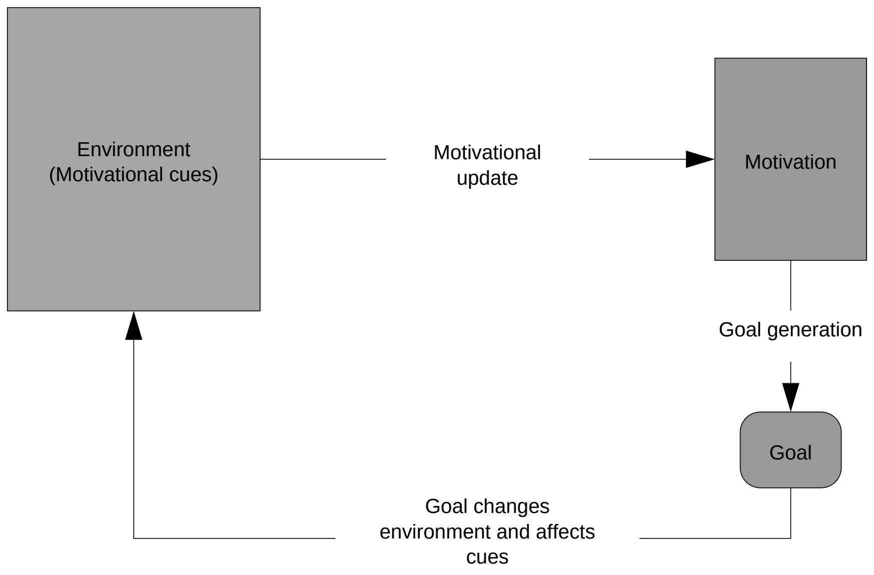


FIGURE 2.6: A control loop driven by $_3M$ motivations.

agent's decision process. Agent behaviour is then a cyclic process of becoming motivated by certain situations and then acting to mitigate them.

A goal is said to be *motivationally relevant* if the satisfaction of that goal affects one or more motivational cues. The relevance of a given goal is quantified by the amount of that this goal does for a motivation. The more a goal mitigates a motivation, the more work it does. The amount of work done by a goal is determined by the plan chosen to satisfy the goal, as well as the relation between the motivational cue and the motivational intensity, *i.e.* how much a cue increases or decreases motivational intensity. Since multiple cues might be associated with a single motivation, satisfying a goal might both mitigate a set of motivations and instigate another one.

2.3.2.3 SMART

The framework of Structured and Modular Agents and Relationship Types (SMART) [d'Inverno and Luck, 2004] is a formalisation of a general purpose model of agents. This framework provides a well defined hierarchy of entities that includes explicitly defined agents, both autonomous and non-autonomous, in order to allow rigorous specification of agents and agent systems as well as their interaction with entities that are not endowed with goal-driven behaviour. It consists of a four-tiered hierarchy of increasingly complex elements situated in an environment, in which:

- entities are just a grouping of attributes without functionality;
- objects are entities with abilities;
- agents are objects with goals that direct the agent into performing activities (on behalf of other agents); and
- autonomous agents are agents with motivations used to support goal generation.

Similar to most BDI-inspired agent models (e.g. Rao and Georgeff's [Rao and Georgeff, 1995b]), an agent operates by either adopting new goals from a base of known goals or by acting to fulfill existing ones. In this model, motivation underpins a process of goal selection based on evaluating new goals against competing or alternate ones, while trying to maximise utility. Such a process is analogous to what is expected of implementations of the BDI systems set forth by [Rao and Georgeff, 1995a,b], but without falling into the pitfall of unbounded computation. Unbounded computation and omniscience are side effects of the possibly infinite representation of the possible-worlds model used in these BDI logics. These limitations are circumvented in motivated systems by binding the utility of a given goal (and associated course of actions) to the motivation that generated it.

As autonomous agents are those that contain a set of motivations, one would expect a model of motivated reasoning associated with SMART agents. Nevertheless, SMART is an abstract framework for autonomous agents, and no specific process of motivation build up and mitigation is specified, nor are specific definitions of how motivations influence agent behaviour. Rather, definitions are provided for the elements that must be taken into account when an agent is to decide which goals to adopt and which actions to select. More specifically, an autonomous agent uses its motivations as the primary input of its action selection process; the component that distinguishes an autonomous agent from a simple one is its motivational state. Simple agents possess goals that are ascribed to them by other agents that can self-generate their goals. Conversely, autonomous agents might introduce new goals to be accomplished through a process underpinned by their set of motivations.

2.3.3 Motivation-driven Agent Architectures

Besides abstract models of motivation-generated behaviour, a number of systems have used the notion of motivation in implementations of autonomous agents, some of them in a more explicit way than others. In this section we describe a number of motivation-driven architectures. Some underpin real systems, whereas others are initial proposals for the modification of existing architectures and the foreseen repercussions of such modifications.

2.3.3.1 Motivated Blackboard

The architecture of Blackboard agents is composed of two levels running asynchronously: a *physical* level containing sensors and effectors responsible for interfacing the agent with the environment; and a *cognitive* level that contains reasoning components not directly related to perception or action [Morignot and Hayes-Roth, 1996]. Each level contains a set of *behaviours*, a *meta-controller* to sequence their activation and a *memory*. The memory component is a shared data structure analogous to a blackboard, which is manipulated by the meta-controller that tries, through forward chaining, to select the appropriate behaviours for execution. A modified version of the architecture at the meta-controller level includes a function for the stimulation and mitigation of motivations based on the current state of the environment. Different functions using a categorisation analogous to that of the $_3M$ taxonomy (Section 2.3.2.2) can be used to guide the agent into selecting relevant goals, and several types of relation are used to adjust the motivational dynamics and achieve a proper interplay of motivation and goal adoption. For example, the safety motivation is represented as an exponential function of the proximity of danger.

2.3.3.2 Motivations in InteRRaP

[Burt, 1998] proposes to extend the InteRRaP architecture to allow motivated reasoning. The InteRRaP architecture is a hybrid multi-layered architecture divided into three basic layers [Müller, 1996]: a *behaviour-based layer* controlling reactive behaviour, which has precedence over a *local-planning layer* that generates single-agent plans, which in turn has precedence over a *social layer* can control the other two layers and guide cooperative behaviour for the agent. The proposed extensions aim to expand InteRRaP with the means to determine the amount of resources to allocate to a process initially and the point at which to change it dynamically.

2.3.3.3 Creatures

While not an architecture in itself, the Creatures software implements a somewhat detailed simulation of interacting biological systems, which includes a biochemical model as well as a motivation-based reasoning mechanism [Grand and Cliff, 1998]. The system simulates the entire life cycle of the artificial creatures including genetic combination and mutation through sexual reproduction. Creatures are controlled by a neural network that includes a large amount of general purpose neurons, as well as dedicated lobes for meta-level tasks. One such task is the direction of the attention of the simulated organism, controlled by a so-called pair of lobes, which are essentially dedicated processing units. Stimuli arriving from objects in the environment are gathered by a set of neurons, and their intensity and frequency are accumulated over time. Simulated lateral-inhibition allows these cells to compete for control of the creature's attention, resulting in the creature's gaze (as well as most of its sensory apparatus) being fixed on this object. Such a mechanism serves to reduce sensory and neural processing to acceptable levels, since the underlying neural net need only consider one object at a time. Though not explicitly described as a motivation-based architecture, the process of directing the attention of the agent to relevant activities is certainly analogous to motivated action.

2.3.4 Motivation-based planning

Besides usage as a driver of goal generation and adoption by an autonomous agent, motivations have been proposed as a means to provide additional information to an agent's planning process [Coddington, 2001; Coddington and Luck, 2003, 2004] . Here, the agent architecture revolves around a continual planning process that operates over a data structure containing both goals to be achieved and actions to be executed. This data structure is constantly updated with actions and goals being added or removed either through satisfaction or dropping. The representation of an agent's capabilities differs from traditional action descriptions in that it contains a set of *pros* and *cons* specifying how motivations are affected by action execution. In turn, this information is used to guide both the planning process and the motivation update. The planning process is then able to use this information to decide which sets of actions (and therefore the goals intended to be accomplished by them) yield the higher motivational benefit.

2.4 Discussion

The concept of autonomous agent programming is inherently declarative in the sense that a designer should be able to specify *what* is to be achieved by an agent and leave the remaining decisions to the agent. Despite such a declarative nature, the assumptions made by logic systems initially developed to describe the process of autonomous decision making prevent them from being directly usable in a pragmatic agent architecture. As a consequence, initial efforts to create autonomous agent architectures have adopted a procedural approach for the description of the agents. Recent efforts focusing declarative agent frameworks have attempted to overcome these limitations by using different models of agent decision making, and can be roughly traced to a couple of endeavours, the widely known GOAL language [Hindriks et al., 2000] and the less well known X-BDI system [Móra et al., 1999].

These two systems take a logic-based approach to the description of agent semantics, but differ in that GOAL focuses on defining such logics and its decision procedures from the ground up, whereas X-BDI is defined on top of an extended logic programming model [Alferes and Pereira, 1996]. The Dribble language [van Riemsdijk et al., 2003] has been developed taking GOAL and 3APL [Hindriks et al., 1999] as the basis of a new language in which GOAL is used for reasoning about selecting declarative goals while 3APL is used to create and modify plans. X-BDI represents one of the first working declarative agent systems that we know of. Nevertheless, it inherits some of the limitations of the non-monotonic logic system upon which it is built. X-BDI also contains what could be seen as a crude prioritisation scheme based on static priorities assigned to the agent's goals at design time, which can be compared at some level to the motivational cues seen later on the chapter.

The declarative agent languages described in Sections 2.2.2, 2.2.3 and 2.2.4, though not providing many new insights into declarative agent semantics, contain a self-contained and evolving set of primitives for such agents. The most recent development of the declarative agent language from [van Riemsdijk et al., 2005] provides a formalisation of a language and (its possible semantics) for a declarative BDI agent system. Though van Riemsdijk's family of declarative languages deals extensively with the formalisation of how goals and beliefs are adopted or dropped, as well as the how to determine whether a declarative goal has been achieved or has failed, these efforts do no provide a clear indication of how a pragmatic implementation might be created. If one considers 3APL to be the basis for an implementation of these concepts [Hindriks et al., 1999], the ensuing system still relies on a predefined set of procedural plans to achieve a goal, in which case the introduction of a planning component such as the one in [Zorzo and Meneguzzi, 2005] becomes an interesting extension.

We believe that one of the main capabilities of a truly declarative agent semantics is the ability to calculate a course of action without recurring to a plan library, thus allowing agent descriptions to focus on *what* is to be achieved by the agent. In this setting, the autonomous calculation of a course of action is known to be a computationally demanding task, therefore an agent has to choose carefully which goals it is to pursue. We have seen that Rao and Georgeff's BDI logic [Rao and Georgeff, 1995b] is not practical in a real system, as it assumes the agent knows the costs and rewards for all possible courses of action (*i.e.* an omniscient agent). However, biological agents are capable of prioritising very effectively despite not being omniscient, and research efforts on motivations seem to explain how biological agents are able to focus their abilities and limited resources into achieving goals that are relevant to them, given the current circumstances [Balkenius, 1993]. Therefore, we have surveyed efforts on modeling such a mechanism within artificial agents. These efforts generally focus on describing heuristic functions that associate events in the world with increased or decreased level of motivations and how goals are activated and prioritised based on these levels. In the Alarms system described in Section 2.3.2.1, individual motivations are used to trigger and prioritise the adoption of individual goals, while in the $_3M$ model described in Section 2.3.2.2 motivations can be associated with more than one goal.

These mechanisms allow agents, even in face of a large number of options, to quickly determine its priorities and proceed to organise its reasoning and actions accordingly [Munroe and Luck, 2003]. Since motivations provide a quantification scheme associated with goals, they can be used as drivers and coordinators of external planning in agent systems, such as those used in $X^2 - BDI$. An example of how to apply the $_3M$ motivational system into $X^2 - BDI$, would use *domain* and *constraint* motivations used to: organise which sets of goals are to be investigated first (using domain motivations); and determine how much effort to put into the ensuing planning (using constraint motivations). This kind of approach would help an agent prioritise the goals which are important to a specific domain, as well as allowing an agent to discover and pursue easily achievable goals (opportunities). Finally, we have observed in the surveyed architectures that the motivation update function appears to be on of the trickiest parts of developing a motivations-driven agent. By looking at the approach taken by the Creatures system (Section 2.3.3.3) ones sees using a neural network to model such functions. In this setting, a network would be used to control the stimulation and mitigation of motivations. Such a network could be trained and evolved until the agent response is adequate, for instance using a criterion similar to that of [Morignot and Hayes-Roth, 1996]. Trained states could then be saved and reused in similar agent architectures.

Chapter 3

Agent Interaction

We have discussed agents as self-contained, autonomous, problem-solving entities (*i.e.* the *micro level*), but agents are also geared towards high-level interactions both with the world around them, and with other agents (*i.e.* the *macro level*).

There are several issues that are important to agent interactions relating to how an agent reasons about and influences the mental state of others. Because autonomous agents have control over their own mental state, agents cannot exercise direct control over other agents. Since direct control among agents is not possible, there must be other mechanisms to provide some form of control to enable interaction between agents. Autonomous agents might also share common goals, and their efficiency might increase if they coordinate their efforts. In this situation it is necessary for agents to use mechanisms to ensure that cooperation occurs in a predictable and reliable way. In order for multiagent systems to function coherently, agents must use their communication capability to attain some level of coordination. Here, coherence is a measure of how well a system works as a unit, and the degree of coordination in a system is the extent to which extraneous activity is minimised as a result of resource contention and deadlocks/livelocks.

Besides delegating tasks and coordinating joint pursuit of goals, self-interested agents sharing a common environment and resources require mechanisms for regulation and to prevent individual agents from taking advantage of others. Regulatory mechanisms might be completely transparent to the agents they regulate, or provide a reasoning framework to punish non-abiding agents. In this chapter we review mechanisms to allow agents to: agree on task/resource allocation (Section 3.1); regulate and direct actions (Section 3.3); and ensure coordination in joint tasks (Section 3.2).

3.1 Agreement Mechanisms

In systems containing self-interested agents, it is possible for some of these agents to have overlapping sets of objectives. Since agents are self-interested, cooperation may not arise naturally from the interactions between them, as the agents are effectively competing to further their own goals without regard for the goals of others. When such agents have a willingness to cooperate for the achievement of a common subset of goals (or alternatively to sell their capabilities in return for some kind of payment), they require a means with which to reach agreements that ensure that their contribution to any cooperative task will be rewarded adequately. There are several mechanisms for reaching agreements among competing agents, and automated negotiation is the most commonly used such method for agents with conflicting objectives and a desire to cooperate [Rahwan et al., 2003].

Negotiation is a process by which a unified decision is reached by two or more agents, each of which is trying to reach an individual objective [Huhns and Stephens, 1999]. After sharing their initial proposals, involved parties are expected to iteratively improve them until a mutually acceptable agreement is reached or no further improvements can be made, in which case no agreement is generated. Though there are many approaches to cooperation, researchers often characterise them as one of three categories [Kraus, 1997; Rahwan et al., 2003]:

- *game-theoretic* approaches in which the optimal strategy is determined by an analysis of the interaction between identical participants, seeking an equilibrium state;
- *heuristic-based* approaches in which approximations of the optimal strategy are discovered through processes relying on relaxed assumptions about the agents' rationality and resources; and
- *argumentation-based* approaches in which the agents exchange higher-level information (such as the reasons for turning down a proposal) aiming to influence each others' mental state in order to reach better agreements.

In this section we review two negotiation models: the first one (Section 3.1.1) is an abstract framework that could be used to implement cooperation using either a game-theoretic or a heuristic approach; while the second model (Section 3.1.2) deals with argumentation-based negotiation.

3.1.1 An abstract negotiation framework

Bartolini *et al.* [Bartolini et al., 2002] describe a generic framework for the description of negotiation mechanisms. The proposed approach is to explicitly specify negotiation rules, to be used in conjunction with a simple interaction protocol, and define negotiation mechanisms using these rules in a structured way. Since the rules of negotiation are explicitly represented, agents involved in negotiation can reason about the rules themselves.

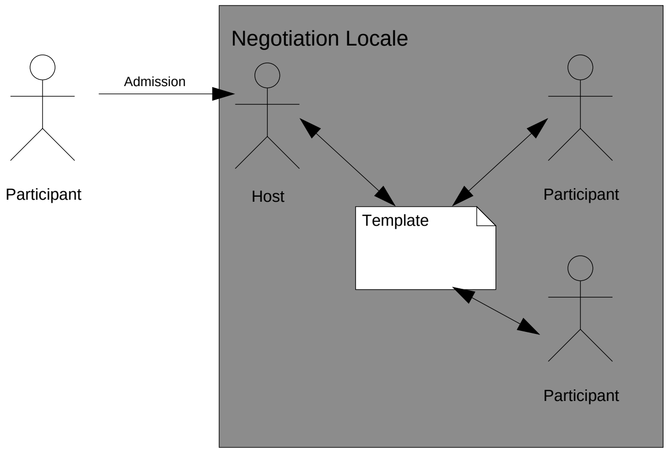


FIGURE 3.1: Elements of an abstract negotiation framework.

The proposed abstract negotiation process includes two negotiation roles: *negotiation participant* and *negotiation host*. The negotiation host is responsible for managing the *negotiation locale*, which is a blackboard where negotiation participants exchange information; Figure 3.1 illustrates this arrangement of host and participants in a negotiation locale. Information access in the locale is subject to the visibility rules of the protocol in question. Participants require *admission* to the negotiation before any other steps are taken, which involves checking their credentials. Once admitted to the negotiation, participants must share a *negotiation template*, which specifies the different parameters of the negotiation. A locale has an associated negotiation template that defines these parameters statically for negotiations taking place in that location, and that must be accepted by all participants wishing to engage in negotiation within that particular location.

Negotiation in this framework is the process of moving from a negotiation template to one or more acceptable agreements (multiple agreements are possible in case multiple parties are involved in negotiation in the same locale). In the process of reaching agreements (summarised in Figure 3.2), participants exchange proposals, which include constraints over the parameters specified in the negotiation template, representing the agreements currently acceptable to them. Proposals are submitted to the negotiation host, which validates them against two criteria:

- the restriction upon the negotiation parameters must be valid according to the parameter space defined in the template; and

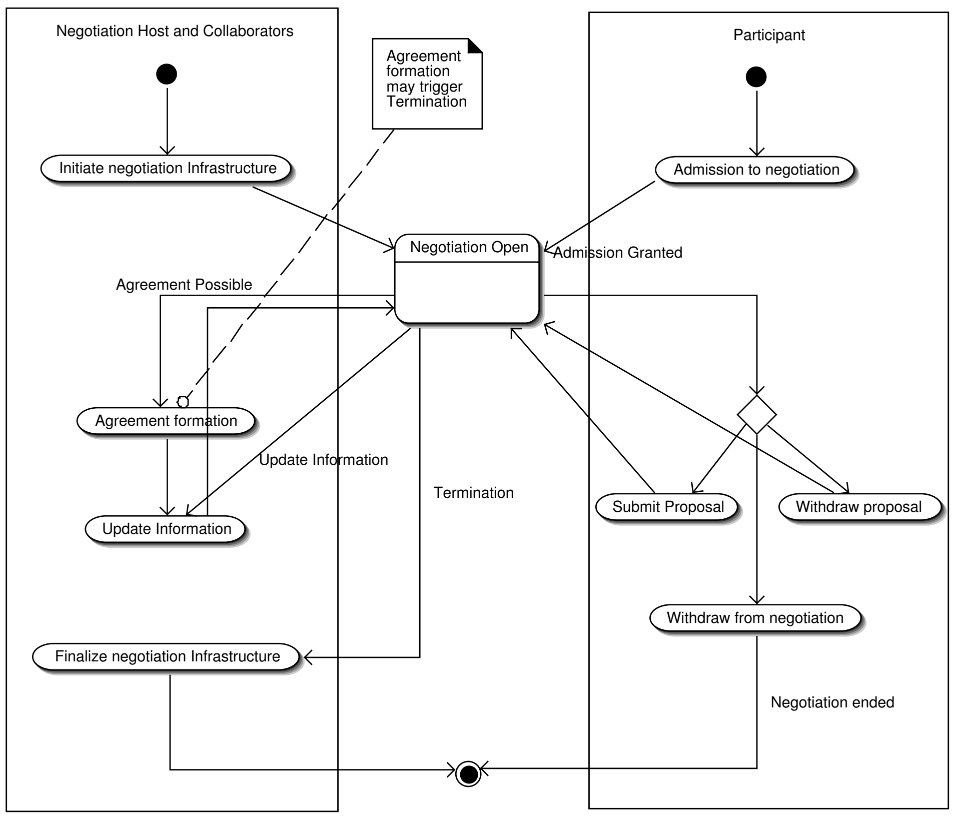


FIGURE 3.2: Bartolini's abstract negotiation process.

- the proposal must be submitted according to the rules of negotiation, which specify, for instance, which participants are allowed to submit new proposals, when they may do so, and how new proposals must improve upon previous ones.

3.1.2 Argumentation-based negotiation

According to Rahwan *et al.* [Rahwan et al., 2003], several researchers believe that traditional methods of negotiation are limited in the type of agreements that can be reached due to the restricted type of information exchanged during the negotiation process. In order to solve this problem, it has been advocated that the likelihood and quality of agreements can increase if agents exchange arguments to influence each others' mental state. The exchange of higher-level information in the negotiation process is known as *argumentation-based negotiation* (ABN).

In other models of automated negotiation, agents exchange proposals containing offers of monetary value or a similar concept, but are not allowed to exchange any additional information; moreover agents are also assumed to have a static mechanism to assess and compare any two proposals. In real environments, agents do not have complete information, so an important part of the negotiation process is to gather information about potential agreements and revise preferences accordingly. Game-theoretic and heuristic approaches assume the agents' utilities or preferences are fixed and, since a rational agent only modifies its preferences upon receiving new information (which these models do not facilitate), these approaches limit the number of possible agreements in a negotiation.

ABN attempts to improve on previous approaches by allowing agents to exchange non-monetary information. Moreover, by endowing an agent with the ability to reason about another agent's rejection of a given proposal, the former agent is in a better position to propose better offers.

Rahwan *et al.* [Rahwan et al., 2003] enumerate a series of components required by an abstract ABN framework. These are divided into external and internal components, which deal respectively with environmental constructs and agent reasoning. Required external components are as follows.

- A *communication language and a domain language* are required by ABN to allow for richer communication and the exchanged of argumentation information, as opposed to communication in other negotiation techniques, which generally deal with proposals alone.
- Similar to other negotiation techniques, a *negotiation protocol* is required for specifying rules for the interaction, exchange of proposals and negotiation termination. However, ABN-tailored protocols should include rules for avoiding disruption of the argumentation (by lack of clarity or fairness in an agent's argumentation), as well as rules to terminate the dialogue and avoid infinite arguments.
- ABN also requires *information stores* to keep externally accessible information during interaction and allow some kind of enforcement of protocol-specified behaviours.

Internal components of ABN agents must be able to explicitly exchange more sophisticated non-monetary information aiming to justify proposals and rejections, so they must be equipped with mechanisms for evaluating arguments and update their mental states accordingly. They must also be able to generate and select arguments and send them to other agents to carry on with a negotiation. In order to satisfy these requirements, Rahwan *et al.* [Rahwan et al., 2003] proposes components for the following activities.

- *Argument and proposal evaluation* to analyse incoming proposals aiming to influence local mental state and decide to which extent an external argument will affect the current preferences of an agent. This evaluation includes comparing arguments

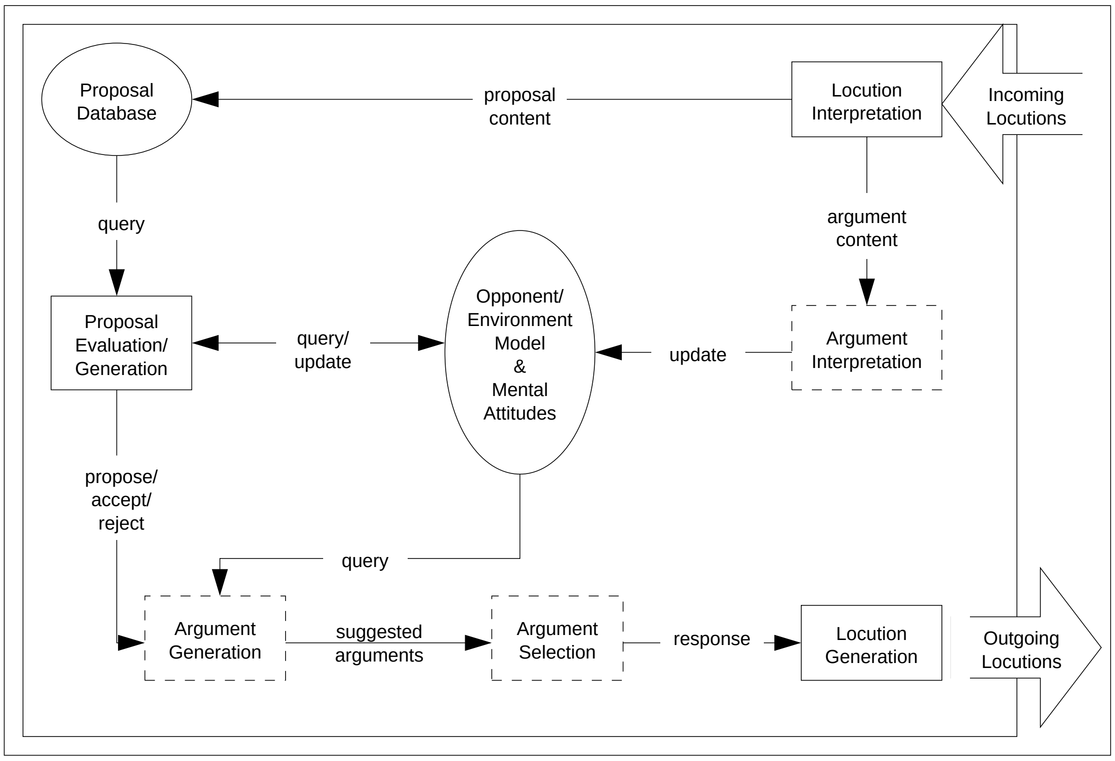


FIGURE 3.3: Components of an ABN mechanism.

so as to reject *weak* ones, which can be done, for example, by evaluating an argument's compatibility with the agent's preferences or the level of trust placed on the arguing agent [Ramchurn et al., 2003; Sadri et al., 2001].

- *Argument proposal generation*, used to create new arguments to further the negotiation process. Generation of new arguments is often the result of planning by the agent aiming to achieve a particular negotiation outcome.
- *Argument selection*, used by agent after generating new arguments to select the one that best furthers the agent's goals in the negotiation. The selection of an argumentation strategy remains an active area of research [Rahwan et al., 2004].

A summary the conceptual elements in an ABN agent is illustrated in Figure 3.3, the elements drawn with dashed lines represent additions to a classical negotiating agent.

3.2 Coordination Mechanisms

Research into distributed problem solving has developed several mechanisms for reasoning about delegation and commitment among agents. Here, agents strive to optimise their actions as a group while avoiding unnecessary conflicts between participants and reducing the amount of redundant work performed in the course of accomplishing a common goal.

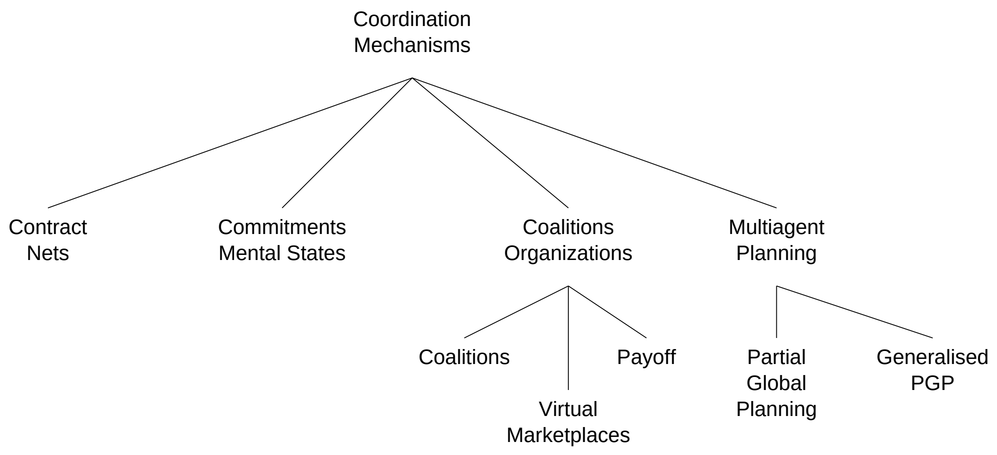
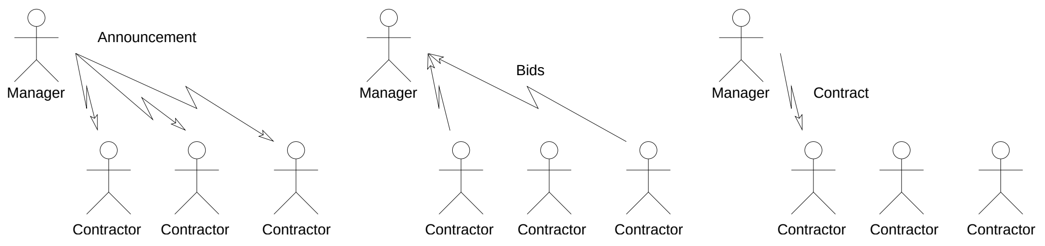


FIGURE 3.4: Coordination Mechanisms.

Many approaches to coordination have been proposed, each of which is based on different assumptions regarding the type of agent participating in the joint problem-solving and the type of problem being solved. In this section we focus on four approaches to this kind of coordination; these approaches and their subcomponents are schematised in Figure 3.4. The contract net protocol is one of the earliest attempts to coordinate agents in a highly decentralised manner, and is reviewed in Section 3.2.1. Section 3.2.2 reviews another possible strategy employed by agents whose representation is based on mental states, which is to use explicit commitments from and towards other agent's mental states in order to predict the future behaviour of participating agents. In systems where there is some kind of economy-based representation for task achievement, agents with coinciding goals or relevant capabilities may contribute to joint tasks aiming at monetary rewards proportional to their contribution (Section 3.2.3). Finally, agents might use planning in order to decide how to contribute towards the achievement of joint goals while interspersing coordination actions to avoid jeopardising the actions of other participants, this approach to coordination is reviewed in Section 3.2.4.

3.2.1 The Contract Net Protocol

In the context of distributed problem solving, one of the first approaches to distributed agent coordination was based on the formation of agreements among pairs of agents; when a problem is decomposed among multiple agents through a network of such agreements or contracts, a *contract net* is created [Davis and Smith, 1988]. Agents in this protocol can assume two possible roles: *manager* and *contractor*. Managers are agents bearing a problem to be solved while contractors are agents with the potential to carry out tasks delegated to them by manager agents. First, a manager typically decomposes a problem and announces tasks to be performed by other agents (as shown in Figure 3.5(a)). When potential contractors respond to the task announcement, the manager receives and evaluates the bids issued by prospective contractors (for example the two contractors shown in Figure 3.5(b)), and then awards contracts to the selected contractors (in Figure 3.5(c) only one of the responding contractors is awarded a contract). After the tasks are finished, the manager receives and synthesises the results. In turn, a contractor receives task announcements, evaluates its ability to perform the tasks, responds by either declining or bidding, performs the allotted task if its bid is accepted, and finally reports its results. Contractors can act as managers and further break down their allotted tasks, subcontracting to other agents. Since the only responsibility of an agent is to send an individual result to their higher-level manager, the fact that an agent subcontracts to others is irrelevant to higher-level managers, thus creating a scalable method of problem distribution.



(a) Announcement. (b) Bidding. (c) Contract.

FIGURE 3.5: Steps in the contract net protocol.

3.2.2 Commitments and Mental States

In his philosophical description of a model of practical reasoning, Bratman [Bratman, 1984] states that when an agent settles on what state of affairs to aim for, he commits to achieving it. The model for BDI defined by Cohen and Levesque [Cohen and Levesque, 1990] follows Bratman's notion and describes the relationship between mental states within an agent. These relations allow an agent to *commit* itself to achieving specific goals as a result of its own reasoning about what it can perform and what it wants to achieve. Commitment here refers to an agent selecting a set of viable goals, and devoting its reasoning to their achievement until the agent either believes it has achieved the selected goals, the goals are no longer viable, or the motivation underpinning the commitment no longer exists. Here, commitments specify *persistent* goals and the conditions for these goals to persist. By extrapolating these relations to the mental attitudes of neighbouring agents, an agent can reason about the commitments of other agents, as well as commit itself to performing actions in parallel to (or on behalf of) other agents as a result of its understanding of the actions others are about to perform. Under this expanded notion of commitment, Cohen and Levesque [Cohen and Levesque, 1991] define *joint commitments* stemming from *joint persistent goals*, which exist when all of the agents in a team have common individual goals (and commitments) to bring about a certain state of affairs. In this setting, agents need to uniformly believe their joint goal is useful and possible, which may not be the case in some situations, *e.g.* when a single agent discovers the goal to be impossible. To deal with such situations, joint persistent goals are themselves underpinned by an individual's *weak achievement goals*, which represent a weaker counterpart of individual persistent goals that allow for an agent to be in an intermediate state of commitment when expecting updates from other agents.

Cohen and Levesque's notion of commitment incorporates the rules that describe when these commitments are to be broken, which Jennings [Jennings, 1993] expands, arguing that effective coordination mechanisms require separate concepts of *commitments* and *conventions*, where:

- commitments are pledges to undertake a specified course of action; and
- conventions regulate how commitments persist, and what should be done when they are dropped.

Indeed, Jennings [Jennings, 1993] argues that all coordination mechanisms can ultimately be reduced to joint commitments and their associated social conventions.

3.2.3 Coalitions and Organizations

In an environment shared by multiple agents, it is possible for a subset to have coinciding goals. When agents share *common* goals, they might group together in order to solve the common problem jointly. Attempts to solve a common problem by a set of autonomous agents require them to agree upon individual responsibilities so that agents do their allotted tasks at the proper time, and refrain from replicating the work of others unnecessarily. Such coordination of tasks can be achieved by forming agent *coalitions* [Zlotkin and Rosenschein, 1994], as well as *virtual organisations* [Norman et al., 2004].

3.2.3.1 Coalitions

Coalitions are formed by agents when a set of participant agents determines they will benefit more from doing work jointly than alone. The process of creating a coalition consists of three processes [Sandholm, 1999]: i) forming the coalition structure (CS); ii) solving the joint problem; and iii) distributing payoff. Forming the coalition structure involves determining the relative contributions of the involved agents and determining the future payoff the agents receive after the problem is solved. This process aims to determine a stable payoff configuration so that agents will not decide to abandon the coalition once problem-solving has started. The process of reaching an appropriate distribution of payoffs is generally analysed in terms of *game theory* [von Neumann and Morgenstern, 1944].

These three processes are the focus of research aiming to generate coalitions whose participants are motivated to remain in them; that is, stable coalitions [Yokoo et al., 2005]. Among the methods proposed for determining stable coalition structures are:

- finding a CS that satisfies the conditions for Nash equilibrium to be achievable [Sandholm, 1999]; and
- considering CS formation as a Characteristic Function Game (CFGs) [Yokoo et al., 2005];

3.2.3.2 Payoff

Nash equilibria arise when an optimal collective strategy in a multiple player game is found such that no player has anything to gain by deviating from this strategy [Nash Jr., 1950]. By contrast, CFG-based methods for stable CS formation search for a payoff division that ensures agents are rewarded with a value that keeps them from wanting to leave their current coalition. There are several strategies for CS formation in CFGs [Shehory and Kraus, 1995], including methods that are able to ensure a bounded degree of optimality for the resulting CS [Sandholm et al., 1999]. Since the conditions for Nash equilibria to exist are very strict and such equilibria can be broken by coordinated deviation from the strategy in equilibrium, CFG-based methods for ensuring coalitional stability are often preferred [Sandholm, 1999].

Some CFG-based approaches to the formation of coalition structures assume the cost of communication and any possible overhead for agent coordination to be negligible. Under such an assumption, the outcome of merging every pair of coalitions in a system is as good as, or better than the outcome of individual coalitions, resulting in *super additive* games. Thus, in super-additive games, merging every single coalition in the system into an all encompassing one (the *grand coalition*) is a natural evolution as the system progresses. However, super additivity overlooks the fact that in most real world scenarios, there is an associated cost with communication and coordination, as well as possible anti-trust penalties [Sandholm, 1999]. In addition, solving the common problem of the grand coalition might be more complex than solving the smaller, and possibly unrelated, problems of its subgroups. This increase in complexity means that the computational overhead of solving the larger problem might also undermine the timely achievement of the coalition's objectives.

In CFGs that are not super additive, finding the appropriate distribution of payoffs that ensures stability is a more complicated problem, which involves finding a CS in which the reward to agents is maximised while the costs associated with communication and coordination are minimised. Finding an optimal CS under this criterion, where agents are guaranteed to receive the best payoff in a given coalition, is known as *welfare maximisation*. Two of the most notable welfare maximisation strategies are:

- finding a distribution that seeks to maximise social welfare by generating a set of possible payoffs in which no agent is motivated to depart from the coalition, such a distribution is said to be within the *core* of a coalition [Zlotkin and Rosenschein, 1994]; and
- calculating the marginal contribution of an agent to the coalition averaging it by the order in which the agents joined the coalition, this contribution value is called the *Shapley Value* [Shehory and Kraus, 1995].

3.2.3.3 Virtual Marketplaces

Aside from situations in which agents *happen* to have coinciding goals, some agent systems operate as *virtual marketplaces* [Norman et al., 2004], in which agents supply their problem-solving capabilities in exchange for some kind of compensation. In this kind of environment, consumer agents announce the need for a particular service or resource, to which other agents acting as suppliers respond. In order to meet the requirements announced by consumer agents, supplying agents might team up to provide a service or resource resulting from the union of their individual capabilities. These agents are represented as a single entity, termed *virtual organisation* [Sandholm, 1999], making transparent to the consumer agent the fact that multiple agents are involved in the supply of the requested capability.

3.2.4 Multiagent Planning

In order to solve problems in a distributed manner, multiple agents need both group coherence and competence [Durfee, 2001]. Coherence relates to agents wanting to work together (addressed in previous sections), while competence relates to agents knowing how to work together well. Perhaps the best way of solving distributed problems is through planning, though planning in a distributed fashion requires another problem to be solved, that of planning on how to work together. This activity of *planning to plan* involves decomposing problems into subproblems, allocating these subproblems, exchanging the obtained solutions and synthesising overall solutions.

Distributed planning is categorised according to whether distribution occurs at the execution or planning stages, or both [Durfee and Lesser, 1991]. When *centralised planning* is employed to produce *distributed plans* for parallel execution, the problem is for the coordinator agent to break a plan into separate threads, possibly adding synchronisation actions throughout the plan in order to ensure that dependency constraints are observed during plan execution. In order to distribute execution of a centralised plan, Durfee [Durfee, 2001] proposes the following steps:

- create a plan using traditional planning, possibly biasing the planner to favour parallel actions;
- decompose the ensuing plans trying to minimise ordering constraints between subplans;
- insert synchronisation actions;
- allocate subplans to executing agents; and
- execute the subplans.

The overhead incurred by the communication required for subplan synchronisation must be taken into account when deciding whether or not to distribute a centralised plan; that is, there exists a minimum subplan size below which parallelisation is not worthwhile.

Distributed planning for centralised plans is associated with cooperative planning, in which a complex planning problem is distributed among different specialised agents which contribute parts of the solution to an overarching plan, in a very similar fashion to that of result sharing.

Distributed planning for distributed plans is the mode of planning most representative of problem solving in multi-agent societies [Durfee, 2001]. In this case, a plan representation for the whole plan is not required to exist at all, as long as the participant agents are not in conflict during the planning and execution tasks.

Besides the planning process, distributed execution must be carried out in a coordinated manner. That is, agents must somehow make sure that: agents execute their assigned actions at the appropriate times; the actions of one agent do not jeopardise the actions of another agent working for the same goal; and agents do not compete for control of critical resources. In turn, coordination might be interleaved between continuous planning and execution, such as in *partial global planning* [Cox et al., 2005; Durfee, 1988], or might be guaranteed either before or after the planning process is carried out. Moreover, when dealing with self-interested agents, agents may be required to negotiate during distributed planning in order to resolve conflicts. Failure to do so can lead to system collapse, so negotiation mechanisms that facilitate the resolution of critical conflicts are an important component of a distributed planning strategy.

One way of ensuring coordination after plans are created is through *contingency planning* [Meuleau and Smith, 2003]. In this approach, an agent not only plans to satisfy the specified problem, it also creates alternative plans to be resorted to in response to contingencies occurring at execution time. Clearly, this entails more complex plans, as well as an overhead to the execution and coordination process, which must now consider the possible threads of plan execution. Aside from contingency planning, agents can monitor execution progress and replan if problems arise. Nevertheless, too much replanning can become a liability, in which case plan repair could be an advantageous approach.

Pre-planning coordination, on the other hand, involves defining a set of constraints that are enforced by the agents in the society during their individual planning processes. If the appropriate set of constraints is defined, agents can theoretically work on any part of the problem, since conflicts can be avoided by carefully abiding by these constraints. Another way of viewing these constraints is as *social laws*, which encode prohibitions against particular choices of actions in particular contexts. This in turn implies the design of combinations of laws that curtail undesirable states, yet are flexible enough to allow for the desired states to arise from the agents.

3.2.4.1 Partial Global Planning

Partial Global Planning (PGP) [Durfee and Lesser, 1990] is a distributed planning framework that adopts a strategy where coordination is a matter of explicitly planning cooperative interactions [Durfee and Lesser, 1991]. In this approach all agents maintain a partial representation of the global plan, and no agent is assumed to be able to see the entirety of the global plan. Here plans detail a node's problem-solving strategy and its expectation about the actions of neighbouring nodes, and although nodes attempt to follow their partial plan as closely as possible, they can also make changes to their plan or propose changes to the plans of other nodes. The PGP framework integrates organisational principles by introducing two types of organisation: the first specifies the long-term problem-solving roles and responsibilities of nodes (*i.e.* a plan of actions); the second, or metalevel organisation, gives nodes a framework for deciding how to solve coordination problems (*i.e.* a plan of communication). Nodes are expected to exchange information about the state of their plans to a certain extent, sharing only high-level information deemed relevant to the nodes being informed. They can also perform task-sharing by proposing (and counterproposing) the transfer of a part of their local plans to other nodes that might be underburdened.

3.2.4.2 Generalised Partial Global Planning

Generalised Partial Global Planning (GPGP) [Lesser et al., 1998] shares with PGP the idea that agents construct their own local view of the tasks they intend to pursue and the relationships among them; this local view can be augmented with information from other agents, allowing agents to create a partial view of the global plan. The generalised framework extends PGP by including individual *coordination* mechanisms used in the creation of such partial views, detecting relations between task structures and ensuring coherent and coordinated behaviour by making commitments to other agents. In turn, these commitments are used by a domain-independent scheduler included in GPGP to create a schedule of activities for the agent to follow.

GPGP also incorporates a representation of task structures from the TAEMS [Wagner et al., 1997] framework to drive the coordination mechanisms. This representation includes information about:

- top-level goals an agent intends to achieve;
- one or more of the ways in which these goals could be achieved;
- a precise quantitative definition of the degree of achievement of goals; and
- task relationships indicating how tasks contribute to the achievement of goals.

GPGP uses the basic TAEMS task structure representation and adds the partial representation of the task structures held by other agents as well as local and non-local commitments to task achievement. Moreover, the quantitative definition of the degree of achievement for goals and tasks indicate that GPGP deals with worth-oriented domains rather than the boolean representation of achievement often used by planning algorithms.

3.3 Regulatory Mechanisms

Research in multiagent interactions has generally focused on the coordination of agents under the assumption that agents do not try to exploit or purposely disrupt the activities of others for individual gain. Moreover, these methods assume that either all of the agents in a society take part in the coordination effort, or the actions of those not included are of no consequence to the ones trying to coordinate endeavours [Durfee, 2001]. The methods used in such efforts often require agents to directly communicate with their interaction partners to ensure coordination (*e.g.* multiagent planning and agent coalitions).

However, in open systems agents are exposed to completely unknown agents, making it unrealistic for an agent to assume that any other agent is completely trustworthy, or will be willing to either participate in coordinated activities or refrain from interfering with them. As a result, coordination mechanisms for open agent systems must deal with uncertainty regarding the willingness of the involved agents to accomplish their allotted tasks. These mechanisms should be able to specify and enforce a *standard of behaviour* in order for the participating agents to predict and to provide assurance about the behaviour of others [Lopez y Lopez, 2003], or allow agents to reason about the reliability of other agents in order to minimise interactions with untrustworthy partners. Ultimately, regulatory mechanisms are necessary for open systems to ensure a level of

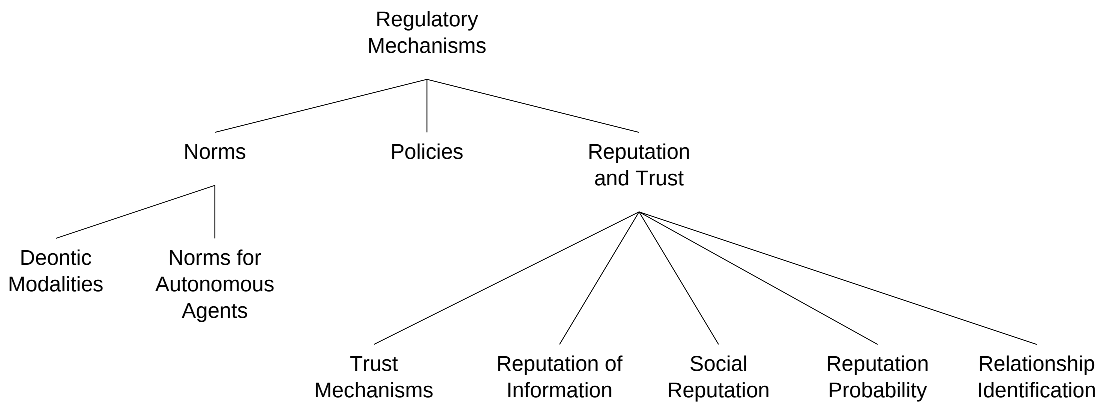


FIGURE 3.6: Regulatory Mechanisms.

reliability so that agents may operate under a minimum degree of certainty. In this section we survey some of these mechanisms, outlined in Figure 3.6.

3.3.1 Norms

In open dynamic societies, agents are required to work with others that do not necessarily have the same set of objectives. If left unchecked, self-interested agents will try to accomplish their individual goals without regard for other agents. In order to minimise conflict between self-interested agents, systems of prescriptive *norms* can be used to specify permissions, prohibitions and responsibilities within a system. These norms are explicitly represented and reasoned about by agents subordinated by them. Though the precise semantics of norms varies throughout different research efforts, Lopez y Lopez [Lopez y Lopez, 2003] identifies six different perspectives.

- *regulation of human societies*, where research focuses mainly on the sociological aspects of human normative bodies;
- *patterns of behaviour*, used to foster coherent group behaviour without the need for explicit planning of coordination actions;
- *constraints on actions*, used to specify permitted and forbidden actions [Shoham and Tennenholtz, 1995];
- *social commitments*, where norms express obligations among agents;
- *mental states*, focusing on the influence exerted by norms upon the adoption of goals by an agent; and
- *norm modelling*, focusing on the definition of the concept of norms and the specification of models of norms.

Here we focus on models of norms which at some level deal with mental states allowing for the construction of normative systems on top of a declarative agent architecture.

3.3.1.1 Deontic logics

Dignum [Dignum, 1999] argues that even agents said to be autonomous are assumed to obey standard protocols, so are predictable in some ways, implying some level of knowledge of the internal mechanisms of these agents. Here, predictability is the result of a set of conventions hard-wired into an agent, undermining the actual autonomy of the agent and consequently its ability to react to a dynamic environment. Dignum states that in order for an agent to be truly autonomous, it must be able to reason about the norms to which it should abide, and occasionally violate them if they are in conflict among themselves or with the agent's private goals. Dignum's [Dignum, 1999] research focuses on defining logical modalities for obligations and permissions (*i.e.* a deontic logic), which distinguish between three levels at which agent behaviour is influenced:

- conventions level;
- contract level; and
- private level.

The division of this framework into levels allows the definition of rules for the different social interactions of an agent. The conventions level represents obligations that constitute a *default background* against which agents interact. These obligations hold under normal circumstances unless higher priority concerns intervene. Norms in the conventions level are generally fixed when the system is initialised and represent general rules for agents in a system to follow (termed *prima facie* norms), such as **agents should not overprice their goods**. Modalities at this level specify obligations, prohibitions and permissions that hold between a given agent in relation to an undefined or abstract counterpart (*i.e.* the agent society). Since there is no specific counterpart towards which the norm is directed, it is assumed that agents follow the rule either due to a commonsense benefit, or that there are agents in charge of enforcing conventions.

The contract level represents commitments between agents, in the form of either *directed obligations* or *authorisations*. Contracts express the expectation of one agent towards another as well as the conditions for these contracts to hold and the consequences of failing to fulfill them. Directed obligations express a commitment from one agent to another that either a world state will hold or an action will be executed. Authorisations express the justification of an agent to perform an action involving another agent; for example, if an agent is to demand payment from another (implying that the latter agent is obliged to pay), it must be authorised to do so.

The private level is used to translate the influences received from the other levels into something that directs the agent's future behaviour. For example, in a BDI setting, external influences and their conditions can be translated into conditional desires for the agent.

3.3.1.2 Norms for autonomous agents

With the same stance as Dignum [Dignum, 1999] with regard to the requirements of norms for autonomous agents, Lopez y Lopez and Luck [Lopez y Lopez and Luck, 2003; Lopez y Lopez et al., 2004] define a formal model of norms whose constructs are reasoned about by autonomous agents. In this model, norms are *prescriptive* in that they specify how agents should behave, and *social* as they are used in situations where multiple agents might come into conflict. Moreover, given the possibility that norms might conflict with an agent's individual goals and that punishments are defined for non-compliance, norms also represent a form of *social pressure* upon the agent.

Depending on their purpose, norms are classified as obligations, prohibitions, social commitments and social codes [Lopez y Lopez and Luck, 2003], where:

- obligations and prohibitions are norms aimed at ensuring coordination among agents in a society, non-compliance of obligations entails punishment, and the manifestation of behaviours targeted by prohibitions leads to punishment;
- social commitments are norms created as a result of agreements or negotiations between a group of agents in order to force them to comply with the agreement or settlement; and
- social codes are norms whose compliance is seen as an end in itself, as it is understood that these are principles accepted in a given society.

Because norms in a given system are rarely isolated from each other, systems of norms are created to ensure that agents comply with whole sets of norms rather than choosing individual norms with which to comply. Systems of norms can also be used to maintain consistency among constituent norms. The association of multiple norms can be attained by relating the activation of a given norm to the violation (or fulfilment) of another through *activation triggers*. Such triggers can be based on agents failing to comply with a norm (*i.e. non-compliance*), in which case a secondary norm is activated to punish the non-compliant agent. Alternatively, agents can be encouraged to comply with certain norms if other norms are created to trigger rewards to compliant agents. These triggers may serve the purpose of either punishing norm violators or rewarding norm followers. In case a violator requires punishment for a transgression, an *enforcer* norm might be activated following the transgression. Alternatively, achievement of a prescriptive goal might trigger a *reward* norm so the compliant agent will be rewarded. Finally, norms may be used to provide for the evolution of the normative system itself. In this context, *legislation* norms are used to permit actions to issue new norms or abolish existing ones.

Since normative systems are maintained within the society employing them through delegation of punishment, reward and legislative goals, the effect of these systems upon prospective members of these societies should be reasoned about by the autonomous agents. When deciding whether to voluntarily join or leave a society regulated by norms, Lopez y Lopez *et al.* [Lopez y Lopez et al., 2004] advocates that an autonomous agent must have an additional set of characteristics to include ways of reasoning about the advantages and disadvantages of abiding by the norms, thus leading to the possibility of norm infringement. Transgression of norms might occur for three main reasons [Lopez y Lopez et al., 2004]:

- individual goals can conflict with society norms;
- norms might conflict among themselves; and
- agents might be members of more than one society.

In light of the possibility of norm infringement and the need for autonomous agents to reason about normative societies, Lopez y Lopez *et al.* [Lopez y Lopez et al., 2004] also define reasoning mechanisms over the effects of norm compliance and violation, as well as rewards and punishments. This model proposes methods for evaluating the benefits of joining a society as well as methods for evaluating whether to stay in a society or to leave it. An agent is seen as staying in a society for two main reasons: due to *unfulfilled goals* within the society or *social obligations.* A social obligation might be that of complying with agreed upon norms, to reciprocate or help a fellow agent or even coercion from another member of the society. The autonomy advocated by this model also includes mechanisms for an agent to voluntarily adopt norms; that is, an agent recognises itself as an addressee and starts following the appropriate norms. This mechanism is important, for instance in situations in which societal laws change dynamically. Finally, the model defines processes through which an agent complies with the norms by adopting or refraining from adopting intentions to achieve normative goals.

3.3.2 Policies

We have seen that norms are rules explicitly adopted and maintained by agents within a society in order to preserve stability and predictability. Hence norms can be encoded within the mental state of an agent and used explicitly by an agent's reasoning process. On the other hand, policies tend to be regarded as prescriptive externally imposed rules that do not emerge from group conventions or patterns of interaction, but are consciously designed and enforced arbitrarily by a centralised authority [Bertino et al., 2005; Xuan and Lesser, 2002]. Policy-based approaches have three main characteristics [Bradshaw et al., 2003]:

- they support dynamic runtime changes;
- agents have no choice about the adoption of policies and might not even be aware that a given set of policies is in force; and
- they are enforced preemptively so as to prevent faulty or malicious agents from jeopardising the system in the first place, rather than punishing violations or rewarding compliance after the fact.

Policies have seen more wide adoption by distributed systems devoid of inherent autonomy so that compliance does not necessitate any degree of reasoning capability [Bandara et al., 2004; Bertino et al., 2005; Graham et al., 2004].

3.3.3 Reputation and Trust

In most of the interaction schemes proposed so far, it is often assumed that the agents involved are inherently trustworthy, *i.e.* there is no chance that a committed agent would not fulfill its commitments. But in the real world, as well as in open multi-agent systems, this assumption quickly fades, and trust implies risk [Castelfranchi and Falcone, 1998]. Trust is present in any form of delegation of tasks, and an agent must trust the target of delegation. According to Falconi and Castelfranchi trust has two dimensions: predictability and belief in competence [Falcone and Castelfranchi, 2001].

Since interaction partners have varying degrees of reliability, agents must have a way of quantifying the trustworthiness of potential partners in order to avoid making arbitrary decisions when creating partnerships [Teacy et al., 2005]. Moreover it is necessary to ensure that agents do not take systematic advantage of others, by ensuring that exploitation-prone agents are identifiable in a system. Mechanisms for evaluating the trustworthiness of an agent within electronic marketplaces generally depend either on using a history of interactions or on recommendations from other agents [Ashri et al., 2005; Teacy et al., 2005] and key to these mechanisms are the concepts of *trust* and *reputation*. Trust quantifies how reliable one agent perceives another to be, whereas the acquisition and processing of trust information from third parties regarding an individual yields that individual's reputation.

3.3.3.1 Trust mechanisms

Trust refers to the belief one has that another party will *do what it says it will* or *reciprocate*, when faced with the prospects of higher payoffs through *defection* [Ramchurn et al., 2004]. A high degree of trust in an agent implies that this agent is likely to be selected for interaction and reciprocated over time, whereas a low degree of trust implies the agent would be neglected and not reciprocated over time. According to Ramchurn *et al.* , trust can be divided into two concepts [Ramchurn et al., 2004]: individual-level and system-level trust. Individual-level trust can be narrowed down to an agent's beliefs about other agents, while system-level trust entails a global mechanism that forces agents to be trustworthy by the protocols within the system.

An individual agent trying to select the most reliable interaction partner can do one of the following.

- interact with each agent in the system to learn their behaviour over several encounters;
- ask other agents about their perception of the potential partners; or
- characterise known motivations of the other agents.

Individual-level trust can be further classified as: learning and evolution-based, reputation based; and socio-cognitive based [Ramchurn et al., 2004]. Evolution-based strategies often rely on game theory to determine the best course of action in the long run for an agent to interact with others, and the metrics for evolving this trust model are usually encoded as a set of equations over the interaction history. An agent using reputation-based strategies builds a model of another agent's reputation based on the results of interactions of the target agent with others; this involves gathering and aggregating ratings while coping with inaccuracies or absence of information. Finally, socio-cognitive strategies involve reasoning about the subjective perception of the candidate for interaction (*e.g.* by evaluating its motivations), rather than analysing the outcomes of interactions. Such a strategy has been proposed by encoding specific beliefs in BDI agents, such as competence, willingness, persistence and motivation [Falcone and Castelfranchi, 2001].

System-level trust, on the other hand is mostly concerned with creating tamper-proof methods of agent interaction, such as:

- devising truth-eliciting interaction protocols;
- developing reputation mechanisms to foster trustworthy behaviour; and
- developing security mechanisms that ensure new entrants can be trusted.

Several open-issues are still being addressed regarding trust between agents, such as ways to detect strategic lying and thus uncover consistent liars, as well as collusion-detection to avoid multi-agent scams. Trust models should also take into consideration the context in which agents fail to fulfill their obligations in order to avoid misjudging them due to environment changes.

3.3.3.2 Reputation of information sources

One way of applying reputation-based trust is in the belief revision process of an agent [Barber and Kim, 2001]. Here, trust refers to the agent's confidence in the ability and intention of an information source to deliver correct information, while reputation refers to the amount of correct information delivered by an agent while interacting with other agents. Since agents often do not have complete knowledge or ground truth about the problem domain, different perspectives of the same system state arise, making it very difficult to ascertain if an information source is unreliable due to maliciousness or just incompetence. In the absence of ground truth in a system trust can be used to characterise confidence in information possessed by agents. Since a rational agent is assumed to desire to have the most reliable information regarding the actual system state, a good reputation is viewed as an asset stimulating an agent to behave properly in order to maintain the flow of reliable information. Agents with consistently low reputations will be isolated from the society at some point, as others will rarely accept justifications or arguments from agents with low reputation. Barber and Kim term the isolation of unreliable partners from the society as *soft security*, as opposed to a system-level enforced *hard security* common in normative mechanisms.

3.3.3.3 Reputation through social analysis

An alternative way of handling reputation in a multi-agent system is to analyse the relationships between individuals in a society, or *social network analysis* [Sabater and Sierra, 2002b]. This is the approach taken by the social ReGreT system [Sabater and Sierra, 2002a]. The intuition behind transferring this method of analysis from human relations to agent systems is that, since multi-agent systems are much simpler in social terms, it allows for even better results than those achievable in a human society. The downside is that the relational data upon which this method depends, which is usually collected through opinion polls from humans [Sabater and Sierra, 2002a], is much scarcer. Social ReGreT assumes the majority of agents in a system to be rational and to behave according to their goals and the type of relationship they hold with other agents. Relations between agents in Social ReGreT are classified into three types:

- **competition** arises when two agents pursue the same goals and need the same (limited) resources, as an agent is assumed to used any mechanism to gain advantage over the competitor, including deception;
- **cooperation** arises when there is a significant exchange of sincere information between agents and predisposition to help each other if possible; and
- **trade** arises when a commercial transaction is under way (trading agents can be seen as partially cooperating and competing).

ReGreT is based on three dimensions of reputation, comprising an *individual dimension*, when considering only direct interactions with others to estimate one's reputation: a *social dimension* used by the agent if it employs information from other members of the society and the social relations; and an *ontological dimension*, which is a concept used to represent different types of reputation among individual agents, and how they are combined to obtain new types of reputation. To support the individual reputation, an agent maintains database of *outcomes* that is moderated by the number of stored entries to assess trust for another agent in a given scenario, and comprises the outcome trust reputation.

The social dimension is used when direct interactions are not available. This might be due to an agent being a newcomer to a system, or simply due to the system being too large for these first hand accounts to be prolific. Depending on the information source, the social dimension uses three types of reputation: witness reputation, neighbourhood reputation and system reputation. Witness reputation is based on information coming from other agents about the target agent. This information is moderated by the reputation of the witnesses themselves, as it might be false, inaccurate or incomplete; the relationship between witnesses and the target agent is also taken into account (the *social trust* [Sabater and Sierra, 2002a]). For example, agents engaged in a cooperative relation imply some degree of complicity, and their information is probably biased. Neighbourhood reputation refers to the type of partner an agent associates with, and is calculated based on the reputation of the *social* neighbours of the target and its relations to them. For example, an agent that commonly associates with low-reputation agents is likely to be untrustworthy. System reputation uses common knowledge about the role played by an agent for that institutional structure in order to assign a default reputation to the agent. The ontological dimension determines the roles of an agent within an organisation.

The system uses these three dimensions, as well as the reputation types associated with them, in a hierarchical model, where the outcome reputation takes precedence over the witness and the neighbourhood reputations followed by the system reputation. An agent uses a lower priority reputation source when its confidence in the higher level ones is not high enough. A shortcoming in this model is that the specific levels of trust are assumed for specific social relations, and therefore cannot be actually detected when it occurs outside of the defined social patterns.

3.3.3.4 Automated relationship identification

An extended version of the ReGreT reputation model builds on the social ReGreT [Sabater and Sierra, 2002a] mechanism utilising the SMART agent framework [d'Inverno and Luck, 2004] to propose a process for agents to dynamically identify relationships in a dynamic marketplace [Ashri et al., 2005]. Like ReGreT, the model describes general types of relationships that ought to be considered regarding trust, as well as the types of reasoning that could be used with that information. Moreover, the ontology framework used in ReGreT is leveraged in order to analyse relationships. Here relationship analysis includes:

- **relationship identification**, which identifies relationships between agents by analysing their individual capabilities and inferring how an agent may come to rely on another one;
- **relationship characterisation**, which distinguishes the type of relationships that are most relevant with regard to trust; and
- **relationship interpretation**, which interprets the obtained relationship patterns to derive trust valuations.

The relationship identification model is underpinned by the concept of agent capabilities from the SMART agent framework, ultimately using this notion to determine what relations may exist within agents in a given context. After relationships are identified, they are characterised. Since there is a large number of potential relationships and combinations of them, along with a myriad of interpretations for them, it is necessary to narrow down the amount of processing required to interpret the meaning of the identified relationships. In order to accomplish this, the model includes basic types of relationships to be used as patterns for evaluation against the dynamically detected ones. Basic relationship types include trade, dependency, competition, collaboration, and tripartite relationships. The addition of an automatic model of relationship identification aims to overcome the limitation of Social ReGreT of relying on predefined relationship patterns.

3.3.3.5 Reputation as probability

Another view of reputation and trust characterises them explicitly in terms of probability theory. This is the stance taken by the TRAVOS model [Teacy et al., 2005], in which trust is defined to be the level of *subjective* probability with which an agent assesses that another agent will perform a particular action [Teacy et al., 2006]. Similarly to other models of reputation, an agent uses trust to determine how reliable another agent is to interact with, and in case there is no history of direct interactions for this assessment to be made, reputation is inferred by gathering information from third parties. This information must be assumed to be potentially incomplete, flawed or unreliable as a result of the attempts of self-interested agents to gain advantage over others. Under these assumptions, this model identifies three requirements for a trust and reputation model:

- it must provide a trust metric that represents the level of trust in an agent;

- it must reflect an individual's confidence in its level of trust for another agent; and
- it must not assume that the opinions of others are accurate or based on actual experience.

In TRAVOS, the level of trust a *truster* agent places in a *trustee* is defined as the probability that the trustee will fulfill its obligation to the truster. Since complete information cannot be assumed, the actual probability value represents the expected probability of a successful interaction. This value is calculated using the experience of the truster, which in turn comprises the set of interaction outcomes observed by the agent. The expected outcome of an interaction has an associated *confidence* metric to account for the uncertainty of the information, as well as its incompleteness. The confidence metric underpins a decision process that prompts an agent to seek more evidence when required and gather reputation information from surrounding agents when this confidence is below a predefined threshold.

In an ideal setting, the experiences gathered from other agents would be as valuable as the ones witnessed by the agent itself. However, autonomous agents exercising preferences over a selection of interaction partners might assign a higher value to interactions with a specific set of agents, overestimating the likelihood of them succeeding. This entails that the experiences of third parties may be biased, inaccurate, or completely fabricated, and thus not applicable to the context of the requesting agent. In order to overcome these hazards, TRAVOS includes a technique to filter inaccurate reputation. The approach taken differs from several endogenous techniques that assume dishonest reputation providers will comprise a minority within a society, so ratings that deviate from mainstream opinion are assumed to be in error and can be filtered out from reputation estimates. TRAVOS uses an exogenous technique that calculates the probability of an opinion being accurate by comparing the history of previously provided opinions and the outcomes of the interactions for which these opinions were requested by the truster. Once the probabilities for all relevant opinions are gathered, the collected opinions are adjusted so that overly optimistic or pessimistic opinions are only considered entirely truthful if its originator is very likely to be accurate. Finally, whatever first hand experiences an agent has about a potential interaction partner are added to the adjusted experiences gathered from neighbouring agents.

3.4 Discussion

In this chapter we have surveyed what we believe to be the main building blocks for coherent multi-agent interactions. These building blocks consist of agreement mechanisms to allow agents to decide whether or not to cooperate with others, coordination mechanisms to ensure that cooperating agents will work together as harmoniously as possible, and finally regulatory mechanisms to keep the effects of malicious behaviour to a minimum within an agent society.

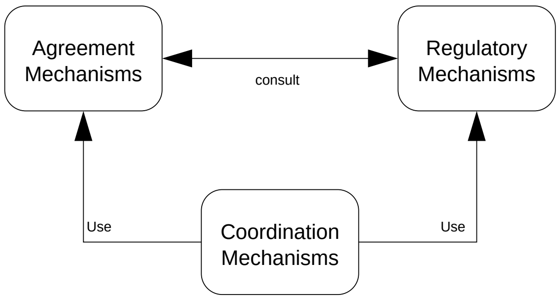


Figure 3.7: Dependencies among agent-interaction mechanisms.

These building blocks were not defined in a vacuum, and some interaction mechanisms rely on others to address possible problems arising from the interaction of autonomous and possibly self-interested agents. More specifically, we have seen that agent cooperation may require negotiation steps to ensure that agents agree on individual tasks and responsibilities to avoid conflicts during joint action. Agents that rely on negotiation to find suitable partners for interaction might keep track of the outcomes of previous interactions and use this information to favour dealing with agents that are deemed as trustworthy. Besides serving as a way to optimise an individual agent's negotiation strategy, trust information can also be used by a society as a whole to isolate overly disruptive individuals. These dependencies are summarised in Figure 3.7.

Research on multiagent interaction has created a series of mechanisms for handling interaction among agents. The interplay between these mechanisms have also been studied to a certain extent. However, the specific reasons for using one technique over another, or even to use multiagent techniques at all rather than a centralised approach have scarcely been investigated. In situations where explicit notions of payoff are used (*e.g.* coalitions), the reason to join a group can be defined to be that of payoff maximisation. For most other mechanisms, their usage and potential benefits are often evaluated by the system designer in an ad-hoc manner. The disconnection between a technique and the circumstance for its application is greater regarding regulatory mechanisms. More specifically, most trust mechanisms do not detail when one approach is more advantageous than other, or to which extent an agent should spend resources (*e.g.* by storing information, or requesting it from others) before deciding whether to interact with another agent. Probabilistic approaches to trust and reputation seem to provide some insights into how to quantify this kind of decision.

Chapter 4

Issues regarding declarative agent architectures

We have seen that the efforts in developing agent architectures yielded two main models of agent interpreter: procedural and declarative. Procedural agent architectures are those in which the achievement of the agent's *design objectives* occurs after certain *procedures* are executed. In this view goal achievement is *implied*, as the goal is assumed to have been accomplished by these procedures. Declarative agent architectures are those in which the achievement of the agent's design objectives is brought about after the agent has explicitly *declared* a set of properties that should be perceived as true by the agent. The agent then selects a course of action through some reasoning mechanism in order to achieve such goals.

Each of these two models of agent architecture is not necessarily better than the other. In some applications where predictability is more important than autonomy, a procedural approach is more desirable than a declarative one. Autonomy on the other hand is facilitated by a declarative approach. Though a large number of procedural agent architectures exist, there are few efforts towards specifying practical declarative agent architectures. The most recent efforts focus on important logical properties and semantic possibilities for mental states, but their solution to how an agent chooses a course of action tend to fall back into either a procedural approach or delegating the definition of plan libraries to the programmer. In these efforts, the declarative nature of goals is explored only to the extent of verifying goal achievement after the agent has executed a plan, and upon goal failure, deciding whether or not to try other plans in an agent's plan library.

Such an approach to the implementation of declarative goals merely provides a higher level method for organising *procedure calls* since the agent is not truly reasoning about the steps it is taking to achieve its declarative goals. This is also true for multi-agent interactions in the sense that agents are often bound by their design to participate in a joint problem-solving effort without any consideration of the reasons for doing so or the actual benefit of joining other agents rather than remaining on its own. In these situations the analysis of when to perform a particular plan or when to enter multiagent *mode* is done by the designer prior to the agent being deployed, so when such an agent is operating without supervision the reasons for its behaviour amounts more to the *dogmas* imposed by the designer than to actual autonomy. If an agent is to behave outside the boundaries of hard-coded rules it must be able to:

- assess when to forgo established sets of behaviours and construct new plans; and
- assess when to delegate tasks to, and when to accept tasks from, other agents.

Assessing when to use one strategy over another is not a simple task, as it involves weighing the effort required by these strategies against their perceived benefits. Attempts to model such an assessment as utility maximisation has lead to decision procedures that assume an omniscient agent [Rao and Georgeff, 1995b], resulting in models that are unsuitable for practical applications. On the other hand, research on motivational states focuses on the utility that an individual agent *expects*, as opposed to absolute knowledge about utility, allowing the agent to determine a goals outcome based only on present and past world-states. Clearly, an agent designer wants an agent to satisfy its design objectives in a predictable way under ideal circumstances, but he also wants the agent to be able to fend for itself in unforeseen situations. As we have seen in Section 2.3, truly autonomous agents are able to generate their own goals, and a model of motivations can underpin the process of goal generation [Luck et al., 2003]. Using this strategy we propose *motivation* as a *control* mechanism for autonomous declarative agents.

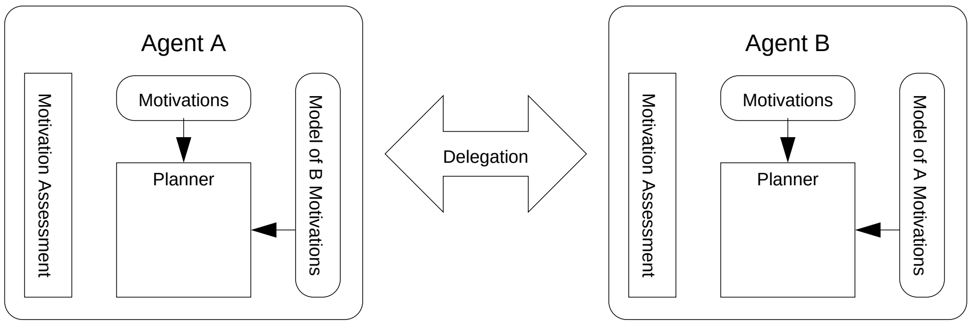


FIGURE 4.1: Conceptual architecture for motivationally controlled delegation.

By using a quantitative model of motivations along with a compatible representation of the cost of an agent's capabilities and resources, an agent should be able to quickly assess the reward of a given strategic choice. Moreover, if we assume that other agents within an environment operate using a similar model of motivational control, an agent should also be capable of querying its neighbours and discover their respective motivational level towards specific goals. Given such an assessment of the motivations of others, an agent should be able to decide when to delegate the achievement of certain goals to (as well as when to accept tasks delegated by) others. For such a model of control to work, a number of issues must first be addressed, these are related to:

- modelling motivations versus the cost of resources and planning;
- modelling the motivations of others;
- assessing the motivations of others;
- evaluating the reasons for interacting with others; and
- assessing when delegation should be attempted.

If motivations are to be used to tune the operation of a planning component, then it is necessary to define which parameters of the planning process are to be affected by motivational intensity. We currently envision a model in which the intensity of a motivation will determine the amount of processing time an agent should dedicate to the planning of the goal associated to that motivation. When such processing time is consumed, the agent stops the planning process and assumes that the goal is not *worthy* of being achieved at that time. We consider this to be a weaker failure mode for a given goal, since the agent is not proving that such goal is impossible. We believe that this kind of reason for a goal to be dropped constitutes one of the reasons for an agent to attempt cooperation with other agents (that may be motivated to spend more processing power planning for the achievement of a joint goal).

Chapter 5

Future Work

Considering the open issues identified in Chapter 4, in this chapter we identify the main set of issues we intend to focus our research for the next 9-month period. Moreover, we outline a strategy to carry out this research, including the activities we intend to execute as well as their associated deliverables; these activities are then organised in a work plan.

5.1 Activities and Deliverables

- **Activity:** Study of planning in declarative agents.
  Deliverables: A prototype of a planning declarative agent and a conference paper reporting the results.

- **Activity:** Study quantitative models of motivations and their integration to agent resource management and prioritisation.
  Deliverables: A modified version of the previous prototype and a conference paper reporting the results.

- **Activity:** Study of multiagent planning and its relation to the previous activities.
  Deliverables: A report or paper outlining a simple agent architecture using the studied concepts.

- **Activity:** Study reasons for agent interaction and task delegation.
  Deliverables: A report or paper describing a decision procedure for interaction/-task delegation.

- **Activity:** Write mini-thesis.
  Deliverables: Mini-thesis and possibly a journal paper with similar content.

5.2 Work Plan

Period	Activity
2006	
May	Study of planning in declarative agents under way. Implementation of planning in a declarative agent architecture under way.
June	Write and submit paper to either AAMAS 2007 or IJCAI 2007.
July	Study of quantitative models of motivations.
August	Study of the integration of motivation models into a planning architecture. Start work on the prototype integrating motivations.
September	Write conference paper (conference TBD.)
October	Study of motivated multiagent planning
November	Write report/paper
December	Study reasons for agent interaction and task delegation.
2007	
January	Write report/paper
February March	Write mini-thesis

5.3 Contributions

Despite its recognised importance in the development of autonomous agents, architectures of declarative agents are scarce, and declarative architectures suitable for application in real domains are non-existent. Existing architectures implement only parts of what we believe to be truly declarative operation. Moreover, motivation models are mostly used in toy-examples rather than applied to fully-fledged agent architectures. Therefore, our main contribution is an investigation of how these concepts can be integrated to create a practical agent architecture. Underlying the integration of motivations with a declarative semantics are the various issues of the correlation of motivational states with planning processes and resource allocation, as well as issues regarding the assessment of potential interaction partners in a multiagent system. We believe that the analysis of these underlying issues constitutes another important contribution to be achieved by our work.

Bibliography

J. J. Alferes and L. M. Pereira. *Reasoning with Logic Programming.* Springer Verlag, 1996.

R. Ashri, S. D. Ramchurn, J. Sabater, M. Luck, and N. R. Jennings. Trust evaluation through relationship analysis. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 1005–1011, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-093-0.

C. Balkenius. The roots of motivation. In *Proceedings of the 2nd International Conference on Simulation of Adaptive Behavior.* MIT Press, 1993.

A. Bandara, E. Lupu, J. Moffett, and A. Russo. A goal-based approach to policy refinement. In *Policies for Distributed Systems and Networks, 2004. POLICY 2004. Proceedings. Fifth IEEE International Workshop on*, pages 229–239, 2004.

K. S. Barber and J. Kim. Belief revision process based on trust: Agents evaluating reputation of information sources. In *Proceedings of the workshop on Deception, Fraud, and Trust in Agent Societies held during the Autonomous Agents Conference*, pages 73–82, London, UK, 2001. Springer-Verlag. ISBN 3-540-43069-5.

C. Bartolini, C. Preist, and N. R. Jennings. Architecting for reuse: A software framework for automated negotiation. In *AOSE*, pages 88–100, 2002.

E. Bertino, A. Mileo, and A. Provetti. Pdl with preferences. In *Policies for Distributed Systems and Networks, 2005. Sixth IEEE International Workshop on*, pages 213–222, 2005.

R. H. Bordini, M. Fisher, C. Pardavila, and M. Wooldridge. Model checking AgentSpeak. In *Proceedings of the 2nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS-03)*, pages 409–416, Melbourne, Australia, July 2003. ACM Press.

J. Bradshaw, A. Uszok, R. Jeffers, N. Suri, P. Hayes, M. Burstein, A. Acquisti, B. Benyo, M. Breedy, M. Carvalho, D. Diller, M. Johnson, S. Kulkarni, J. Lott, M. Sierhuis, and R. V. Hoof. Representation and reasoning for daml-based policy and domain services in kaos and nomads. In *AAMAS '03: Proceedings of the second international joint* *conference on Autonomous agents and multiagent systems*, pages 835–842, New York, NY, USA, 2003. ACM Press. ISBN 1-58113-683-8.

M. E. Bratman. Two faces of intention. *Philosophical Review*, 93:375–405, 1984.

M. E. Bratman. *Intention, Plans and Practical Reason.* Harvard University Press, Cambridge, MA, 1987.

M. E. Bratman, D. J. Israel, and M. E. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4(4):349–355, 1988.

L. Braubach, A. Pokahr, W. Lamersdorf, and D. Moldt. Goal representation for BDI agent systems. In R. H. Bordini, M. Dastani, J. Dix, and A. E. Fallah-Seghrouchni, editors, *Proceedings of the Second International Workshop on Programming Multiagent Systems Languages and tools (PROMAS 2004)*, pages 7–9, 2004.

A. Burt. Modelling motivational behaviour in intelligent agents in virtual worlds. In *Proceedings of the 1998 Conference on Virtual Worlds and Simulation*, 1998.

P. Busetta, R. Rönnquist, A. Hodgson, and A. Lucas. Jack intelligent agents - components for intelligent agents in java. AgentLink Newsletter, January 1999. White paper, http://www.agent-software.com.au.

C. Castelfranchi and R. Falcone. Principles of trust for mas: Cognitive anatomy, social importance, and quantification. In *ICMAS*, pages 72–79, 1998.

A. M. Coddington. *Self-Motivated Planning in Autonomous Agents.* PhD thesis, University College London, 2001.

A. M. Coddington and M. Luck. Towards motivation-based plan evaluation. In I. Russell and S. Haller, editors, *Proceedings of Sixteenth International FLAIRS Conference*, pages 298–302, Florida, USA, 2003.

A. M. Coddington and M. Luck. A motivation-based planning and execution framework. *International Journal on Artificial Intelligence Tools.*, 10(1):5–25, 2004.

P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(2-3):213–261, 1990.

P. R. Cohen and H. J. Levesque. Teamwork. *Noûs*, 25(4):487–512, 1991. Special Issue on Cognitive Science and Artificial Intelligence.

J. S. Cox, E. H. Durfee, and T. Bartold. A distributed framework for solving the multiagent plan coordination problem. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 821–827, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-093-0.

M. Dastani, B. van Riemsdijk, F. Dignum, and J.-J. C. Meyer. A programming language for cognitive agents goal directed 3apl. In *PROMAS*, pages 111–130, 2003.

R. Davis and R. G. Smith. Negotiation as a metaphor for distributed problem solving. pages 333–356, 1988.

W. V. der Hoek and M. Wooldridge. Towards a logic of rational agency. *Logic Journal of the IGPL*, 11(2):133–157, March 2003.

F. Dignum. Autonomous agents with norms. *Artificial Intelligence and Law*, 7(1):69–79, Mar. 1999.

M. d'Inverno and M. Luck. *Understanding Agent Systems*. Springer Series on Agent Technology. Springer Verlag, Berlin, 2nd edition, 2004.

M. d'Inverno and M. Luck. Engineering AgentSpeak(L): A formal computational model. *Journal of Logic and Computation*, 8(3):233–260, 1998.

M. d'Inverno, D. Kinny, M. Luck, and M. Wooldridge. A formal specification of dMARS. In M. P. Singh, A. S. Rao, and M. Wooldridge, editors, *Agent Theories, Architectures, and Languages*, volume 1365 of *Lecture Notes in Computer Science*, pages 155–176. Springer-Verlag, 1998.

E. Durfee and V. Lesser. Partial global planning: a coordination framework for distributed hypothesis formation. *Systems, Man and Cybernetics, IEEE Transactions on*, 21(5):1167–1183, 1991. ISSN 0018-9472.

E. H. Durfee. Distributed problem solving and planning. pages 118–149, 2001.

E. H. Durfee. *Coordination of Distributed Problem Solvers*. Springer, 1988.

E. H. Durfee and V. R. Lesser. Predictability versus responsiveness: coordinating problem solvers in dynamic domains. pages 198–203, 1990.

R. Falcone and C. Castelfranchi. Social trust: a cognitive approach. pages 55–90, 2001.

M. Georgeff, B. Pell, M. E. Pollack, M. Tambe, and M. Wooldridge. The belief-desire-intention model of agency. In J. Müller, M. P. Singh, and A. S. Rao, editors, *Proceedings of the 5th International Workshop on Intelligent Agents V : Agent Theories, Architectures, and Languages (ATAL-98)*, volume 1555, pages 1–10. Springer-Verlag: Heidelberg, Germany, 1999.

M. P. Georgeff and F. F. Ingrand. Decision-making in an embedded reasoning system. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI'89)*, pages 972–978, Detroit, MI, 1989a. Morgan Kaufmann.

M. P. Georgeff and F. F. Ingrand. Monitoring and control of spacecraft systems using procedural reasoning. In *Proceedings of the Space Operations and Robotics Workshop*, page n/a, Houston, TX, July 1989b.

M. P. Georgeff and A. L. Lansky. Procedural knowledge. *Proceedings of the IEEE, Special Issue on Knowledge Representation*, 74(10):1383–1898, 1986.

M. P. Georgeff and A. L. Lansky. Reactive reasoning and planning. In *Proceedings of the American Association for Artificial Intelligence (AAAI)*, pages 677–682, Seattle, WA, 1987. Morgan Kaufmann Publishers.

A. Graham, T. Radhakrishnan, and C. Grossner. Incremental validation of policy-based systems. In *Policies for Distributed Systems and Networks, 2004. POLICY 2004. Proceedings. Fifth IEEE International Workshop on*, pages 240–249, 2004.

S. Grand and D. Cliff. Creatures: Entertainment software agents with artificial life. *Autonomous Agents and Multi-Agent Systems*, 1(1):39–57, 1998.

K. V. Hindriks, F. S. de Boer, W. van der Hoek, and J.-J. C. Meyer. Agent programming in 3apl. *Autonomous Agents and Multi-Agent Systems*, 2(4):357–401, 1999.

K. V. Hindriks, F. S. de Boer, W. van der Hoek, and J.-J. C. Meyer. Agent programming with declarative goals. In *ATAL*, pages 228–243, 2000.

N. Howden, R. Rönnquist, A. Hodgson, and A. Lucas. Jack ? summary of an agent infrastructure. In *Proceedings of the 5th International Conference on Autonomous Agents*, 2001.

M. J. Huber. JAM: a BDI-theoretic mobile agent architecture. In *Proceedings of the 3rd Annual Conference on Autonomous Agents*, pages 236–243. ACM Press, 1999. ISBN 1-58113-066-X.

M. N. Huhns and L. M. Stephens. Multiagent systems and societies of agents. pages 79–120, 1999.

F. F. Ingrand and V. Coutance. Real-time reasoning using procedural reasoning. Technical Report 93104, LAAS/CNRS, LAAS/CNRS, France, January 2001. Technical Report.

F. F. Ingrand, M. P. Georgeff, and A. S. Rao. An architecture for real-time reasoning and system control. *IEEE Expert, Knowledge-Based Diagnosis in Process Engineering*, 7 (6):33–44, 1992.

N. R. Jennings. On agent-based software engineering. *Artificial Intelligence*, 117(2): 277–296, 2000.

N. R. Jennings. Commitments and conventions: The foundation of coordination in multi-agent systems. *The Knowledge Engineering Review*, 8(3):223–250, 1993.

N. R. Jennings, A. G. Cohn, M. Fox, D. Long, M. Luck, D. T. Michaelides, S. Munroe, and M. J. Weal. *Cognitive Systems: Information Processing Meets Brain Science*, chapter 8. Motivation, Planning and Interaction, pages 163–188. Queen's Printer and Controller of HMSO, 2006.

R. A. Kowalski and M. J. Sergot. A logic-based calculus of events. *New Generation Computing*, 4(1):67–95, 1986.

S. Kraus. Negotiation and cooperation in multi-agent environments. *Artif. Intell.*, 94 (1-2):79–97, 1997. ISSN 0004-3702.

J. Lee, M. J. Huber, P. G. Kenny, and E. H. Durfee. UM-PRS: An Implementation of the Procedural Reasoning System for Multirobot Applications. In *Conference on Intelligent Robotics in Field, Factory, Service, and Space (CIRFFSS)*, pages 842–849, Houston, Texas, 1994.

V. Lesser, K. Decker, N. Carver, D. Neiman, M. N. Prasad, and T. Wagner. Evolution of the gpgp domain-independent coordination framework. Technical report, Amherst, MA, USA, 1998.

F. Lopez y Lopez. *Social Power and Norms: Impact on agent behaviour.* PhD thesis, University of Southampton, 2003.

F. Lopez y Lopez and M. Luck. Modelling norms for autonomous agents. In *Computer Science, 2003. ENC 2003. Proceedings of the Fourth Mexican International Conference on*, pages 238–245, 2003.

F. Lopez y Lopez, M. Luck, and M. d'Inverno. Normative agent reasoning in dynamic societies. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 732–739, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 1-58113-864-4.

M. Luck and M. d'Inverno. Motivated behavior for goal adoption. In *DAI*, pages 58–73, 1998.

M. Luck, S. Munroe, and M. d'Inverno. *Autonomy: Variable and Generative*, chapter Chapter 2, pages 9–22. Kluwer, 2003.

N. Meuleau and D. E. Smith. Optimal limited contingency planning. In *UAI*, pages 417–426, 2003.

J. Misra and K. M. Chandy. *Parallel Program Design: A Foundation.* Addison-Wesley, 1989.

M. C. Móra, J. G. Lopes, R. M. Viccari, and H. Coelho. BDI models and systems: Reducing the gap. In J. P. Müller, M. P. Singh, and A. S. Rao, editors, *Proceedings of the 5th International Workshop on Intelligent Agents V : Agent Theories, Architectures, and Languages (ATAL-98)*, volume 1555 of *Lecture Notes in Computer Science.* Springer Verlag, Germany, 1999.

P. Morignot and B. Hayes-Roth. Motivated agents. Technical report, Knowledge Systems Laboratory – Stanford University, 1996.

J. P. Müller. The design of intelligent agents: A layered approach. In *The Design of Intelligent Agents: A Layered Approach*, volume 1177 of *Lecture Notes in Computer Science*. Springer Verlag, Germany, 1996.

S. Munroe, M. Luck, and M. d'Inverno. Motivation-based selection of negotiation partners. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1520–1521, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 1-58113-864-4.

S. J. Munroe and M. Luck. 3m motivational taxonomy. In *Agents and Computational Autonomy*, pages 55–67, 2003.

S. J. Munroe, M. Luck, and M. d'Inverno. Towards motivation-based decisions for worth goals. In *Proceedings of Multi-Agent Systems and Applications III, Proceedings of the 3rd International Central and European Conference on Multi-Agent Systems*, pages 17–28, 2003.

R. Nair, M. Tambe, and S. Marsella. Integrating belief-desire-intention approaches with POMDPs: The case of team-oriented programs. In P. Doherty, J. McCarthy, and M.-A. Williams, editors, *Logical Formalization of Commonsense Reasoning 2003 AAAI Spring Symposium*, pages 107–115. AAAI Press, 2003.

J. F. Nash Jr. Equilibrium points in n-person games. *Proceedings of the Nationall Academy of Sciences*, 36:48–49, 1950.

N. Nide and S. Takata. Deduction systems for BDI logics using sequent calculus. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 928–935. ACM Press, 2002. ISBN 1-58113-480-0.

T. J. Norman and D. Long. Alarms: An implementation of motivated agency. In *ATAL*, pages 219–234, 1995.

T. J. Norman, A. Preece, S. Chalmers, N. R. Jennings, M. Luck, V. D. Dang, T. D. Nguyen, V. Deora, J. Shao, W. A. Gray, and N. J. Fiddian. Agent-based formation of virtual organisations. *Knowledge-Based Systems*, 17(2-4):103–111, May 2004.

M. E. Pollack and M. Ringuette. Introducing the tileworld: experimentally evaluating agent architectures. In T. Dietterich and W. Swartout, editors, *Proceedings of the 8th National Conference on Artificial Intelligence*, pages 183–189, Menlo Park, CA, 1990. AAAI Press.

M. E. Pollack, D. Joslin, A. Nunes, S. Ur, and E. Ephrati. Experimental investigation of an agent commitment strategy. Technical Report 94–31, University of Pittsburgh, Pittsburgh, PA 15260, 1994.

I. Rahwan, S. D. Ramchurn, N. R. Jennings, P. Mcburney, S. Parsons, and L. Sonenberg. Argumentation-based negotiation. *Knowl. Eng. Rev.*, 18(4):343–375, 2003. ISSN 0269-8889.

I. Rahwan, L. Sonenberg, and F. P. Dignum. *On Interest-Based Negotiation*, volume 2922. Springer Verlag, Jan. 2004.

S. D. Ramchurn, N. R. Jennings, and C. Sierra. Persuasive negotiation for autonomous agents: A rhetorical approach. In *Proceedings of the IJCAI Workshop on Computational Models of Natural Arguments*, pages 9–17. AAAI Press, 2003.

S. D. Ramchurn, D. Huynh, and N. R. Jennings. Trust in multi-agent systems. *Knowl. Eng. Rev.*, 19(1):1–25, 2004. ISSN 0269-8889.

A. S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In W. V. de Velde and J. W. Perram, editors, *7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, volume 1038 of *Lecture Notes on Computer Science*, pages 42–55. Springer Verlag, Eindhoven, Netherlands, 1996.

A. S. Rao and M. P. Georgeff. BDI-agents: from theory to practice. In *Proceedings of the First International Conference on Multiagent Systems ICMAS-95*, pages 312–319, San Francisco, 1995a.

A. S. Rao and M. P. Georgeff. Formal models and decision procedures for multi-agent systems. Technical Report 61, Australian Artificial Intelligence Institute, 171 La Trobe Street, Melbourne, Australia, 1995b. Technical Note.

J. Sabater and C. Sierra. Social regret, a reputation model based on social relations. *SIGecom Exch.*, 3(1):44–56, 2002a. ISSN 1551-9031.

J. Sabater and C. Sierra. Reputation and social network analysis in multi-agent systems. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 475–482, New York, NY, USA, 2002b. ACM Press. ISBN 1-58113-480-0.

F. Sadri, F. Toni, and P. Torroni. Dialogues for negotiation: Agent varieties and dialogue sequences. In *ATAL*, pages 405–421, 2001.

T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohmé. Coalition structure generation with worst case guarantees. *Artif. Intell.*, 111(1-2):209–238, 1999. ISSN 0004-3702.

T. W. Sandholm. Distributed rational decision making. pages 201–258, 1999.

M. Schut and M. Wooldridge. The control of reasoning in resource-bounded agents. *The Knowledge Engineering Review*, 16(3), 2001.

M. Shanahan. An abductive event calculus planner. *The Journal of Logic Programming*, 2000.

O. Shehory and S. Kraus. Coalition formation among autonomous agents: Strategies and complexity. In *From Reaction to Cognition*, number 957, pages 57–72, 1995.

Y. Shoham and M. Tennenholtz. On Social Laws for Artificial Agent Societies: Off-Line Design. *Artif. Intell.*, 73(1-2):231–252, 1995.

W. T. L. Teacy, J. Patel, N. R. Jennings, and M. Luck. Coping with inaccurate reputation sources: experimental analysis of a probabilistic trust model. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 997–1004, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-093-0.

W. T. L. Teacy, J. Patel, N. R. Jennings, and M. Luck. Travos: Trust and reputation in the context of inaccurate information sources. *Journal of Autonomous Agents and Multi-Agent Systems*, 12(2), 2006.

B. van Riemsdijk, W. van der Hoek, and J.-J. C. Meyer. Agent programming in dribble: from beliefs to goals using plans. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 393–400, New York, NY, USA, 2003. ACM Press. ISBN 1-58113-683-8.

B. van Riemsdijk, M. Dastani, F. Dignum, and J.-J. C. Meyer. Dynamics of declarative goals in agent programming. In *DALT*, pages 1–18, 2004.

M. B. van Riemsdijk, M. Dastani, and J.-J. C. Meyer. Semantics of declarative goals in agent programming. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 133–140, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-093-0.

J. von Neumann and O. Morgenstern. *Theory of games and economic behavior*. Princeton University Press, 1944.

T. Wagner, A. Garvey, and V. Lesser. Complex goal criteria and its application in design-to-criteria scheduling. Technical report, Amherst, MA, USA, 1997.

D. E. Wilkins and K. L. Myers. A Common Knowledge Representation for Plan Generation and Reactive Execution. volume 5, pages 731–761, 1995.

M. Winikoff, L. Padgham, J. Harland, and J. Thangarajah. Declarative & Procedural Goals in Intelligent Agent Systems. In D. Fensel, F. Giunchiglia, D. L. McGuinness, and M.-A. Williams, editors, *Proceedings of the Eights International Conference on Principles and Knowledge Representation and Reasoning (KR-02)*, pages 470–481, Toulouse, France, April 2002. Morgan Kaufmann.

M. Wooldridge. *Reasoning about Rational Agents*. The MIT Press, 2000a.

M. Wooldridge. The Computational Complexity of Agent Design Problems. In E. Durfee, editor, *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS 2000)*, pages 341–348. IEEE Press, 2000b.

P. Xuan and V. Lesser. Multi-agent policies: from centralized ones to decentralized ones. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 1098–1105, New York, NY, USA, 2002. ACM Press.

M. Yokoo, V. Conitzer, T. Sandholm, N. Ohta, and A. Iwasaki. Coalitional games in open anonymous environments. In *AAAI*, pages 509–515, 2005.

G. Zlotkin and J. S. Rosenschein. Coalition, cryptography and stability: Mechanisms for coalition formation in task oriented domains. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 432–437, 1994.

A. F. Zorzo and F. R. Meneguzzi. An agent model for fault-tolerant systems. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 60–65, New York, NY, USA, 2005. ACM Press. ISBN 1-58113-964-0.