Logic for Automated Mechanism Design — A Progress Report

Michael Wooldridge* **Thomas Agotnes†** **Paul E. Dunne*** **Wiebe van der Hoek***

*Department of Computer Science
University of Liverpool
Liverpool L69 3BX, UK

†Department of Computer Engineering
Bergen University College
PB. 7030, N-5020 Bergen, Norway

Abstract

Over the past half decade, we have been exploring the use of logic in the specification and analysis of computational economic mechanisms. We believe that this approach has the potential to bring the same benefits to the design and analysis of computational economic mechanisms that the use of temporal logics and model checking have brought to the specification and analysis of reactive systems. In this paper, we give a survey of our work. We first discuss the use of *cooperation logics* such as Alternating-time Temporal Logic (ATL) for the specification and verification of mechanisms such as social choice procedures. We motivate the approach, and then discuss the work we have done on extensions to ATL to support incomplete information, preferences, and quantification over coalitions. We then discuss is the use of ATL-like cooperation logics in the development of social laws.

Introduction

In recent years, there has been a dramatic increase of interest in the study and application of *economic mechanisms* in computer science and artificial intelligence (Sandholm 1999; Nisan & Ronen 1999). For example, auctions are a well-known type of economic mechanism, used for resource allocation, which have achieved particular prominence in computer science (Krishna 2002; Cramton, Shoham, & Steinberg 2006). There are a number of reasons for this rapid growth of interest. The influence of multi-agent systems research is surely one (Bond & Gasser 1988; Weiß 1999; Wooldridge 2002), but perhaps more fundamentally, it is increasingly recognised that a truly deep understanding of many (perhaps most) distributed and networked systems can only come after acknowledging that they have the characteristics of economic systems, in the following sense. Consider an online auction system, such as eBay (EBAY 2001). At one level of analysis, this is simply a distributed system: it consists of various nodes, which interact with one-another by exchanging data, according to some protocols. Distributed systems have been very widely studied in computer science, and we have a variety of techniques for engineering and analysing them (see, e.g., (Ben-Ari 1990)). However, while this analysis is of course legitimate, and no doubt important, it is surely missing a big, and very important part of the picture. The participants in such online auctions are *self interested*. They are acting in the system *strategically*, in order to obtain the best outcome for themselves that they can. For example, the seller is typically trying to maximise selling price, while the buyer is trying to minimise it. Thus, if we only think of such a system as a distributed system, then our ability to predict and understand its behaviour is going to be rather limited. We also need to understand it from an *economic* perspective. In the area of multi-agent systems, we take these considerations one stage further, and start to think about the issues that arise when the participants in the system are themselves computer programs, acting on behalf of their users or owners (Wooldridge 2002).

A number of very natural issues arise if we start to consider the design of *computational mechanisms* (Rosenschein & Zlotkin 1994; Sandholm 1999; Kraus 2001). In this paper, we address ourselves to the following:

- How can we specify the desirable properties of computational mechanisms?
- How can we verify that these mechanisms behave as we intended?

The starting point for our research is that *logic* has proven to be an extremely successful and powerful tool in the specification and analysis of protocols in computer science. There is thus some reason for supposing that it might be of similar value in the specification and analysis of computational mechanisms. *Temporal logics* have been perhaps the most successful formalism in the specification and verification of conventional reactive and distributed systems (Emerson 1990), and the associated verification technology of *model checking* for temporal logics has proven to be enormously successful (Clarke, Grumberg, & Peled 2000; Holzmann 2003). However, conventional temporal logics are not well suited for expressing the properties of economic, game-like systems. They are intended for expressing liveness and safety properties, not for expressing *strategic* properties.

Our work over the past half decade has focused on the use of *cooperation logics* for automated mechanism design and analysis. Cooperation logics were developed independently and more-or-less simultaneously by several researchers in the late 1990s (Alur, Henzinger, & Kupferman 1997; Pauly 2002). As we shall see, although cooperation logics are in fact descended from conventional temporal logics, they are

ideal for expressing the strategic properties of systems. Our aims in this paper are, first, to motivate this research program in more detail, and second, to survey the progress we have made. Our work in this area is based around two directions, and the paper is structured accordingly. In the following section, we motivate and then introduce cooperation logics for *social choice mechanisms*. We then go on to consider how such logics can be used in the design of *social laws*. We then present some conclusions and future research issues.

Logic for Social Choice Mechanisms

Social choice mechanisms are a very general class of economic mechanism (Arrow, Sen, & Suzumura 2002). Social choice mechanisms are concerned with selecting some particular outcome from a range of alternatives on behalf of a collection of participants, known as *agents*. Typically, the agents have different preferences over the possible outcomes, and the mechanism considers these preferences when choosing the outcome. *Voting procedures* are examples of social choice mechanisms (Brams & Fishburn 2002). Perhaps the best known voting procedure is the "first past the post" (FPTP) system, (also known as single winner plurality voting), which is used in the UK for electing political representatives. Here, the possible outcomes correspond to the possible candidates, only one of which can be elected; voters express their preferences by indicating their most preferred candidate, and the mechanism states that the selected outcome will be the one gaining the largest number of votes. While FPTP is simple to understand and implement, it has of course many well-documented drawbacks. For example, if there are more than two candidates, then the outcome selected may not in fact have an overall majority, meaning that a majority of voters would prefer some other outcome. Moreover, the mechanism is prone to *strategic manipulation*: for example, agents can sometimes benefit by voting against their true preferences if they believe their most preferred outcome is unlikely to win overall. Other social choice mechanisms have been proposed in an attempt to overcome the limitations of simple voting procedures such as FPTP – examples include the Borda count and single transferable vote. The study of such mechanisms has traditionally been the domain of *social choice theory* in economics (Arrow, Sen, & Suzumura 2002). Perhaps the most important result in social choice theory, due to Kenneth Arrow, is a negative one: any social choice mechanism involving more than two alternative outcomes must fail to satisfy one of three basic axioms for such protocols (Campbell & Kelly 2002)[1]. Another key negative result, the Gibbard-Satterthwaite impossibility theorem, says that in any non-dictatorial social choice mechanism (i.e., in any mechanism that is not "controlled" by a single agent), it is possible for an agent to benefit by voting strategically, i.e., voting against its preferences (Arrow, Sen, & Suzumura 2002). Although at first sight these results suggest that the further development of social choice mechanisms must be a quixotic enterprise, it turns out that useful mechanisms can in practice be developed for many settings, for example by modifying or relaxing some of the conditions of Arrow's theorem (Campbell & Kelly 2002, p.52).

Recently, there has been substantial interest in social choice mechanisms from within the computer science community. There are several reasons for this interest; for example:

- The multi-agent systems field is concerned with the problem of building software agents that can interact with one-another in order to achieve goals, typically on behalf of users (Wooldridge 2002). Such interaction frequently involves the agents *autonomously reaching agreements with one another*. This then raises the question of what protocols the agents will use to decide how to reach agreement with one another. The fact that the participants will be *software* agents (rather than humans) raises a rather different set of concerns to those that arise when considering the use of mechanisms by humans. For example, an obvious question is how computationally hard it is for an agent to determine how to vote so as to obtain the best possible outcome for itself, and the associated question of whether it is possible to design a social choice mechanism that is too computationally complex to manipulate in this way – see, e.g., (Conitzer 2006) for an example of such issues.
- Given the current international interest in e-government, and in particular the possibility of increased public involvement in the democratic process via the Internet, the design of appropriate social choice mechanisms for such scenarios has become of interest. A typical issue here is that of authentication: if a member of the public is registering their vote via the Internet, how can we ensure that the individual registering the vote really is who they purport to be? Moreover, how can an individual verify that her vote was indeed counted, without making public the votes of others?

It is common to refer to social choice mechanisms as "protocols", since they involve a number of parties exchanging messages in certain well-defined sequences. However, whereas protocol designers are typically concerned with such issues as deadlock-freeness, mutual exclusion over shared resources, and guaranteed receipt of messages, in an economic mechanism we are also, and primarily, concerned with a higher level set of issues, relating to the *strategic behaviour* of the participants. That is, we assume that the participants in the mechanism will always choose to act in their best interests, and ask what then follows. This implies the participants will take into account how *other* participants will act in the mechanism, under the assumption that they too are acting in their best interests. Ultimately, such strategising may lead to behaviours such as participants with similar interests colluding with one another, misrepresenting their actual preferences, or even being deliberately deceitful, if it seems this ultimately leads to some benefit for themselves. Thus, when designing mechanisms for software agents it is of course essential to consider protocol-level issues such as deadlock freeness; but the main issues one faces stem from strategic considerations. The fact that we must take account of strategic concerns, in addition to protocol-

[1] Formally, the criteria are: Pareto optimality, independence of irrelevant alternatives, and non-dictatorship.

level issues, makes social choice mechanisms particularly hard to design and analyse.

Although the mathematical foundations of social choice mechanisms have been studied within the game theory community for some time, their treatment as *computational objects*, and in particular, their formal specification and automated verification was not considered until recently. An important step forward in this regard came with the development of *cooperation logics* such as Alternating-time Temporal Logic (ATL) (Alur, Henzinger, & Kupferman 2002) for representing the properties of strategic interaction in multi-agent systems, and the realisation by Marc Pauly that such cooperation logics could be used to naturally capture the requirements of many social choice mechanisms.

ATL emerged from the use of Computation Tree Logic (CTL) for the specification and verification of reactive systems (Emerson 1990). CTL is a temporal logic that is interpreted over tree-like structures, in which nodes represent time points and arcs represent transitions between time points. In distributed/reactive systems applications, the set of all paths through a tree structure is interpreted as the set of all possible computations of a system. CTL combines *path quantifiers* "A" and "E" for expressing that a certain series of events will happen on all paths and on some path respectively, with *tense modalities* for expressing that something will happen eventually on some path ($\Diamond$), always on some path ($\Box$) and so on. Thus, for example, by using CTL-like logics, one may express properties such as "on all possible computations, the system never enters a fail state", which is represented by the CTL formula $\mathsf{A}\Box\neg fail$.

Although they have proven to be enormously useful in the specification and verification of reactive systems (Clarke, Grumberg, & Peled 2000), logics such as CTL are of limited value for reasoning about systems in which strategic behaviour is of concern. The kinds of properties we wish to express of such systems typically relate to the *strategic powers* that system components have. For example, we might wish to express the fact that "agents 1 and 2 can cooperate to ensure that, no matter what agents 3 and 4 do, the system never enters a fail state". It is not possible to capture such statements using CTL-like logics. The best one can do is either state that something will inevitably happen, or else that it may possibly happen: CTL-like logics thus have no notion of agency. Alur, Henzinger, and Kupferman developed ATL in an attempt to remedy this deficiency. The key insight in ATL is that path quantifiers can be replaced by *cooperation modalities*: the ATL expression $\langle\langle C\rangle\rangle\varphi$, where C is a group of system components (agents), expresses the fact that C can cooperate to ensure that, no matter how other system components behave, φ will result. Thus $\langle\langle C\rangle\rangle\varphi$ captures the *strategic ability of C to bring about φ*. So, for example, the fact that "agents 1 and 2 can ensure that the system never enters a fail state, no matter what agents 3 and 4 do" may be captured in ATL by the following formula:

$$\langle\langle 1, 2\rangle\rangle\Box\neg fail.$$

Pauly's insight was that the ATL cooperation modality construct can be used to express the desirable properties of social choice mechanisms. To see how this works, consider the following informal requirements for a simple social choice mechanism (Pauly 2001):

> *Two individuals, A and B, must choose between two outcomes, p and q. We want a mechanism that will allow them to choose which will satisfy the following requirements: We want an outcome to be possible – that is, we want the two agents to choose, collectively, either p or q. We do not want them to be able to bring about both outcomes simultaneously. Finally, we do not want either agent to be able to unilaterally dictate an outcome – we want them both to have "equal power".*

These requirements may be formally and naturally represented using ATL, as follows:

$$\langle\langle A, B\rangle\rangle\bigcirc p \tag{1}$$
$$\langle\langle A, B\rangle\rangle\bigcirc q \tag{2}$$
$$\neg\langle\langle A, B\rangle\rangle\bigcirc (p \wedge q) \tag{3}$$
$$\neg\langle\langle A\rangle\rangle\bigcirc p \tag{4}$$
$$\neg\langle\langle B\rangle\rangle\bigcirc p \tag{5}$$
$$\neg\langle\langle A\rangle\rangle\bigcirc q \tag{6}$$
$$\neg\langle\langle B\rangle\rangle\bigcirc q \tag{7}$$

Property (1) states that A and B can collectively choose p, while (2) states that they can choose q; (3) states that they cannot choose p and q simultaneously; and properties (4)–(7) state that neither agent can dictate an outcome.

Now, once we have such a formal specification of the requirements of a mechanism in this way, we can start to apply the apparatus of automated reasoning developed within computer science and AI to *reason about* and *synthesise* mechanisms:

- The problem of *synthesising* a mechanism that satisfies properties φ reduces to a *constructive proof of satisfiability for φ*: given some requirements φ, again expressed using ATL, try to find some mechanism M such that $M \models_{\mathsf{ATL}} \varphi$; if we can exhibit such an M, then this will serve as our desired mechanism; if there is no such M, then announce that the no mechanism correctly implements the specification. The satisfiability problem for ATL is EXPTIME-complete (Drimmelen 2003; Walther *et al.* 2006), which means that synthesis in this way is going to be computationally costly.
- The problem of checking whether a mechanism M satisfies property φ, where φ is expressed using the language of ATL as in formulae (1)–(7), above, reduces to a *model checking problem*: check whether $M \models_{\mathsf{ATL}} \varphi$, cf. (Clarke, Grumberg, & Peled 2000). Alur and colleagues demonstrated that, for an *explicit state* representation of models, (i.e., where we "explicitly enumerate" the states of a model in the input), the model checking problem for ATL is PTIME-complete, and hence tractable (Alur, Henzinger, & Kupferman 2002); this is usually interpreted as a positive result. However, if we assume a representation of models such as those actually used by ATL model checkers (Alur *et al.* 1998), then the complexity of model checking rises dramatically – it is in fact just as hard as the satisfiability problem (Hoek, Lomuscio, & Wooldridge 2005).

This approach – specifying the desirable properties of a mechanism using such a logic – is the *Logic for Automated Mechanism Design and Analysis* paradigm, of which the first contours were sketched in (Pauly & Wooldridge 2003). In this paper, a number of social choice mechanisms were formally specified using ATL, and existing ATL model checking tools (Alur *et al.* 1998) were used to formally – and automatically – analyse properties of candidate mechanisms with respect to these specifications. For example, consider the following mechanism, intended to permit the agents to select between the outcomes in accordance with these requirements.

> *The two agents vote on the outcomes, i.e., they each choose either p or q. If there is a consensus, then the consensus outcome is selected; if there is no consensus, (i.e., if the two agents vote differently), then an outcome p or q is selected non-deterministically.*

Notice that, given this simple mechanism, the agents really can collectively choose the outcome, by cooperating. If they do not cooperate, however, then an outcome is chosen for them.

Having formally set out the desirable properties that we wish a mechanism to satisfy, and having described a mechanism that we believe satisfies these properties, our next step is to formally verify that the mechanism does indeed satisfy them. We do this via model checking: we express the mechanism as a model suitable for the ATL model checking system MOCHA, and then, using MOCHA, we check whether the requirements are realised in this model.

A MOCHA model of the mechanism is given in Figure 1. While space restrictions preclude a detailed introduction to the modelling language of MOCHA, it is nevertheless worth briefly describing the key features of this representation. We model the scenario via three agents, which in MOCHA terminology are called `modules`:

- `AgentA` and `AgentB` correspond to the A and B in our scenario. Each agent controls (i.e., has exclusive write access to) a variable that is used to record their vote. Thus `voteA` records the vote of `AgentA`, where a value of `false` in this variable means voting for outcome `P`, while `true` implies voting for `Q`. The "program" of each agent is made up of two remaining guarded commands, which simply present the agent with a choice of voting either way.
- The `Environment` module is used to model the mechanism itself. This module simply looks at the two votes, and if they are the same, sets the variable `outcome` to be the consensus outcome; if the two votes are different, then the guarded commands defining `Environment`'s behaviour say that an outcome will be selected non-deterministically.

Notice that in translating this simple mechanism in a form suitable for MOCHA, it has not been possible to remain entirely neutral with respect to all issues. For example, the way we have coded the mechanism means that it is in principle possible for one agent to see another agent's vote (i.e., votes are common knowledge), even though, in the implementation given here, agents do not make any use of this information. The informal description of the mechanism – and indeed, the original requirements – said nothing about whether votes (and hence preferences) should remain hidden or should be common knowledge, and in fact, we could have coded the scenario in such a way that an agent's vote was visible only to the `Environment` module. But the point is that we have been forced to make a commitment one way or the other by the need to code the scenario. It is of course likely that in more sophisticated (and realistic) scenarios, we would desire votes to remain private. We discuss this issue in more detail below.

```
-- voteA == false ...  agent A votes for outcome P
-- voteA == true ...  agent A votes for outcome Q
module AgentA
   interface voteA : bool
   atom controls voteA
   init update
      [] true -> voteA' := false
      [] true -> voteA' := true
   endatom
endmodule

-- voteB == false ...  agent B votes for outcome P
-- voteB == true ...  agent B votes for outcome Q
module AgentB
   interface voteB : bool
   atom controls voteB
   init update
      [] true -> voteB' := false
      [] true -> voteB' := true
   endatom
endmodule

-- outcome == false ...  P is selected
-- outcome == true ...  Q is selected
module Environment
   interface outcome :  bool
   external voteA, voteB : bool
   atom controls outcome awaits voteA, voteB
   init update
   -- if votes are the same, go with selected outcome
      [] (voteA' = voteB') -> outcome' := (voteA' & voteB')
   -- otherwise select outcome non-deterministically
      [] ~(voteA' = voteB') -> outcome' := true
      [] ~(voteA' = voteB') -> outcome' := false
   endatom
endmodule -- Environment

System := (AgentA || AgentB || Environment)
```

Figure 1: A simple social choice mechanism, defined in the ReactiveModules language of the MOCHA model checker.

Having captured the mechanism in the modelling language of MOCHA, we can use a model checker to check that the desired properties do actually hold. And indeed they do.

It should be clear to readers familiar with social choice

theory that we are not too far away from the kinds of properties that Arrow and Gibbard-Satterthwaite deal with in their famous theorems. However, we can only *explicitly* capture properties such as dictatorship using "vanilla" ATL. In the following sub-sections, we shall see some of the extensions to ATL that we have been developing to allow other properties to be naturally represented.

Incomplete Information

Incomplete information plays a role in most mechanisms. For example, in a sealed bid auction, the fact that I do not know what you are bidding (and you do not know what I am bidding) is an essential aspect of the mechanism. It is therefore very natural to consider *epistemic* extensions to ATL. Based on the type of epistemic logic popularised in AI by Fagin-Halpern-Moses-Vardi (Fagin *et al.* 1995), we developed and investigated epistemic extensions to ATL (Hoek & Wooldridge 2002; 2003b; 2003a). The first line of attack we followed was to simply add epistemic modalities K_i for each agent i to ATL: a formula $K_i\varphi$ is intended to express the fact that agent i knows φ. The resulting language, ATEL, is extremely powerful and very natural for expressing the properties of communicating systems. For example, the following formula expresses that a can communicate its knowledge of φ to b:

$$K_a\varphi \rightarrow \langle\langle a \rangle\rangle \bigcirc K_b\varphi$$

As another example, consider a security protocol, in which agents a and b share some common secret (a key S_{ab} for instance), what one typically wants is the following, which expresses that a can send private information to b, without revealing the message to another agent c:

$$K_a\varphi \wedge \neg K_b\varphi \wedge \neg K_c\varphi \wedge \langle\langle a, b \rangle\rangle \bigcirc (K_a\varphi \wedge K_b\varphi \wedge \neg K_c\varphi)$$

Knowledge pre-conditions, of the type introduced into the theoretical foundations of AI planning by Moore (Moore 1990), are also very naturally expressed in ATEL. The fact that knowledge of ψ is a necessary pre-condition to be able to achieve φ is represented by the following.

$$\langle\langle a \rangle\rangle \bigcirc \varphi \rightarrow K_a\psi$$

Of course, as Moore's seminal analysis shows, the interaction between knowledge and ability is rather complex and subtle, and some of the issues raised by Moore are reviewed in the context of ATEL in (Jamroga & van der Hoek 2004).

A detailed case study, in which we show how epistemic-ability properties may be model checked, is given in (Hoek & Wooldridge 2003a).

Preferences

We don't get very far in the study of mechanisms without some way of dealing with *preferences*. Of course, it *is* possible to represent preferences in "vanilla" ATL, but not very elegantly. We have to make use of the propositional logic machinery available in the language, for example by introducing propositions of the form $u_{i,x}$, with the intended interpretation that in the current state, agent i gets utility x. This is not a very attractive approach (Benthem 2002). Unfortunately, the logical representation of preferences is a ongoing research area, and there is no universally accepted approach: we have been investigating a number of alternatives. In (van Otterloo, Hoek, & Wooldridge 2004), we considered an operator $[C : \varphi]\psi$, with the intended reading "if the agents C prefer φ, and act accordingly, then ψ follows". It was shown how this preference operator could be used to naturally capture properties of mechanisms such as "any coalition of size greater than n which prefers φ can bring about φ". However, this assumes that the preferences of C are made public, while we might want to consider cases where an agent does not publically disclose its preferences, or falsely announces them. In (Agotnes, van der Hoek, & Wooldridge 2006a), we developed a logic intended for reasoning about coalitional games without transferable utility, which combined an ATL-style ability operator with a direct representation of preferences over outcomes, of the form $(\omega_1 \succ_i \omega_2)$, meaning agent i prefers outcome ω_1 over ω_2; it was shown how these constructs were sufficient to characterise properties of coalitional games such as core non-emptiness (cf. (Osborne & Rubinstein 1994)). Finally, in (Agotnes, van der Hoek, & Wooldridge 2007c), we developed a formalism explicitly intended to support reasoning about Arrovian properties of social choice mechanisms, and Arrow's theorem has a direct and succinct syntactic characterisation as an axiom of the logic. The logic provides for (modal) quantification over alternatives and preference profiles, although it is arguably not a "natural" formalism for humans to read. We should emphasise that, although a lot of research has been done in this area, there is still as yet no entirely satisfactory way of representing preferences within an ATL-like formalism, and this topic remains the subject of ongoing research.

Quantification

Expressing many interesting properties of mechanisms requires *quantification* over coalitions. For example, consider the following property: "agent i is a member of every coalition that can achieve φ". We *can* represent this in ATL, as follows:

$$\bigwedge_C (\langle\langle C \rangle\rangle \bigcirc \varphi) \rightarrow \neg\langle\langle C \setminus \{i\} \rangle\rangle \bigcirc \varphi$$

We thus use conjunction as a universal quantifier. The problem with this formulation is that it results in a formula that is exponentially long in the number of agents in the system. An obvious solution would be to extend ATL with a first-order-style apparatus for quantifying over coalitions. In such a quantified ATL, one might express the above by the following formula:

$$\forall C : \langle\langle C \rangle\rangle \Diamond \varphi \rightarrow (i \in C)$$

However, adding quantification in such a naive way leads to undecidability over infinite domains (using basic quantificational set theory we can define arithmetic), and very high computational complexity even over finite domains. The question therefore arises whether we can add quantification

to cooperation logics in such a way that we can express useful properties of cooperation in games *without* making the resulting logic too computationally complex to be of practical interest. In (Agotnes, van der Hoek, & Wooldridge 2007b), we answered this question in the affirmative. We introduced *Quantified Coalition Logic*, which allows a useful but restricted form of quantification over coalitions. In QCL, we replace cooperation modalities $\langle\langle C\rangle\rangle$ with expressions $\langle P\rangle\varphi$ and $[P]\varphi$; here, P is a *predicate over coalitions*, and the two sentences express the fact that *there exists a coalition C satisfying property P such that C can achieve φ* and *all coalitions satisfying property P can achieve φ*, respectively. Thus we add a limited form of quantification *without* the apparatus of quantificational set theory. The resulting logic, QCL, is exponentially more succinct than the corresponding fragment of ATL, while being computationally no worse with respect to the key problem of model checking.

To see how QCL works, consider specifying *majority voting*:

> *An electorate of n voters wishes to select one of two outcomes ω_1 and ω_2. They want to use a simple majority voting protocol, so that outcome ω_i will be selected iff a majority of the n voters state a preference for it. No coalition of less than majority size should be able to select an outcome, and* any *majority should be able to choose the outcome (i.e., the selection procedure is not influenced by the "names" of the agents in a coalition).*

Let $maj(n)$ be a predicate over coalitions that is satisfied if the coalition against which it is evaluated contains a majority of n agents. For example, if $n = 3$, then coalition $\{1, 3\}$ would satisfy the predicate, as would coalitions $\{2, 3\}$ and $\{1, 2\}$, but coalitions $\{1\}$, $\{2\}$, and $\{3\}$ would not. We can express the majority voting requirements above as follows. First: *every majority should be able to select an outcome*.

$$([maj(n)]\omega_1) \wedge ([maj(n)]\omega_2)$$

Second: no coalition that is not a majority can select an outcome.

$$(\neg\langle\neg maj(n)\rangle\omega_1) \wedge (\neg\langle\neg maj(n)\rangle\omega_2)$$

Simple though this example is, it is worth bearing in mind that its expression in ATL is exponentially long in n.

Succinct Representations

ATL does not have much to say about the *origins* of an agent's powers. If we consider specific models for where an agent's powers come from, then we end up with systems closely related to ATL, but with some rather different properties. We considered one such variation in (Hoek & Wooldridge 2005b), where we modelled a system by supposing that each agent in the system controlled a set of propositions. The powers of an agent, and the coalitions of which it is a member, derive from the possible assignments of truth or falsity that it can give to the propositions under its control. The resulting logic was shown to be much simpler than ATL. In (Hoek & Wooldridge 2005a), we considered a variation of this in which it is possible for agents to *transfer control* of the propositions they control to other agents. In (Wooldridge & Dunne 2004; Agotnes, van der Hoek, & Wooldridge 2006b; Wooldridge & Dunne 2006), we considered the issue of how to represent the semantic structures underpinning logics such as ATL, and in particular, we developed a representation for them based on propositional logic.

Logic for Social Laws

It is often implicitly assumed that, when we come to construct a mechanism, we have complete freedom to design the mechanism, starting with a blank slate. In practice, of course this is rarely the case: we have to deal with *legacy* systems. In this section, we review our work on the design of mechanisms for use in settings where we are *given a pre-existing system in which the mechanism must operate*. In AI, this idea was introduced in the *social laws* paradigm of Shoham, Tennenholtz, and Moses (Shoham & Tennenholtz 1992; Moses & Tennenholtz 1995; Shoham & Tennenholtz 1997). A social law can be understood as a set of rules imposed upon a multiagent system with the goal of ensuring that some desirable behaviour will result. Social laws work by *constraining* the behaviour of the agents in the system – by *forbidding* agents from performing certain actions in certain circumstances.

In (Hoek, Roberts, & Wooldridge 2007), we investigated the use of ATL for specifying the desirable properties of social laws. The idea is that the designer of a social law will have some objective in mind, which they desire the social law to achieve. We explored extensions to the Shoham, Tennenholtz, and Moses social law model in which this objective was expressed in ATL. In so doing, we could explicitly define social laws in which the objective was to ensure that agents in the system had "rights" which would be preserved by the social law. We showed how, in some cases, it was possible to view the social law synthesis problem as one of ATL model checking. We considered social laws with epistemic ATL objectives in (Hoek, Roberts, & Wooldridge 2005).

In (Wooldridge & van der Hoek 2005), we introduced a variant of ATL called *Normative* ATL, which was intended to directly support reasoning about social laws. Normative ATL replaces cooperation modalities $\langle\langle C\rangle\rangle$ with expressions $\langle\langle \eta : C\rangle\rangle\varphi$, where η is a social law, or *normative system*, C is a coalition, and φ is a sentence of the logic. The intended interpretation of $\langle\langle \eta : C\rangle\rangle\varphi$ is that operating within the context of the normative system η, coalition C have the ability to bring about φ; more precisely, that C have a winning strategy for φ, where this strategy conforms to the strictures of the normative system η. We showed how this logic could be used to reason about normative systems, and how it could be used in the logical analysis of social contracts. Crudely, the term "social contract" refers to the collection of norms or conventions that a society abides by. These norms serve to regulate and restrict the behaviour of citizens within a society. The benefit of a social contract is that it prevents mutually destructive behaviours. However, there are many apparent paradoxes associated with the social contract, not the least being that of why a rational, self-interested

agent should choose to conform to the social contract, when choosing to do otherwise might lead to a better individual outcome; the problem being that if everyone reasons this way (and as rational agents, they should), then nobody conforms to the social contract, and its benefits are lost. There have been several game theoretic accounts of the social contract, which attempt to understand how a social contract can work in a society of self-interested agents (Binmore 1994; 1998); our work was an attempt to give a logical account. We further developed these ideas in (Agotnes *et al.* 2007).

In (Agotnes, van der Hoek, & Wooldridge 2007a), we combined ideas from our logic-based social law design approach with ideas from game theoretic mechanism design. For example, we showed that the problem of designing a social law such that everybody participating in the social law represents a Nash equilibrium is NP-complete.

Conclusions

We believe that the use of logic for automated mechanism design and analysis has the potential to bring the same benefits to the design and analysis of computational economic mechanisms that the use of temporal logics and model checking have brought to the specification and analysis of reactive systems. In this paper, we have surveyed some of our work in this area over the past five years. We are still in the early stages of this research; trying to identify the issues, and tentatively proposing solutions to overcome the hurdles we encounter. By analogy with AI planning, we are probably still living in the blocks world. Nevertheless, we believe there is every reason to be optimistic about this research direction, and we hope that, after reading this paper, you will be as excited about it as we are.

Acknowledgments

We gratefully acknowledge the support of the EU through their IST research grant programme, and the support of several UK EPSRC grants. We would also like to thank Marc Pauly for encouragement and support, and most of all, for introducing us to Coalition Logic.

References

Agotnes, T.; van der Hoek, W.; Rodriguez-Aguilar, J. A.; Sierra, C.; and Wooldridge, M. 2007. On the logic of normative systems. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*.

Agotnes, T.; van der Hoek, W.; and Wooldridge, M. 2006a. On the logic of coalitional games. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2006)*.

Agotnes, T.; van der Hoek, W.; and Wooldridge, M. 2006b. Temporal qualitative coalitional games. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2006)*.

Agotnes, T.; van der Hoek, W.; and Wooldridge, M. 2007a. Normative system games. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2007)*.

Agotnes, T.; van der Hoek, W.; and Wooldridge, M. 2007b. Quantified coalition logic. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*.

Agotnes, T.; van der Hoek, W.; and Wooldridge, M. 2007c. Reasoning about judgment and preference aggregation. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2007)*.

Alur, R.; Henzinger, T. A.; Mang, F. Y. C.; Qadeer, S.; Rajamani, S. K.; and Taşiran, S. 1998. Mocha: Modularity in model checking. In *CAV 1998: Tenth International Conference on Computer-aided Verification, (LNCS Volume 1427)*, 521–525. Springer-Verlag: Berlin, Germany.

Alur, R.; Henzinger, T. A.; and Kupferman, O. 1997. Alternating-time temporal logic. In *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science*, 100–109.

Alur, R.; Henzinger, T. A.; and Kupferman, O. 2002. Alternating-time temporal logic. *Journal of the ACM* 49(5):672–713.

Arrow, K. J.; Sen, A. K.; and Suzumura, K., eds. 2002. *Handbook of Social Choice and Welfare Volume 1*. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands.

Ben-Ari, M. 1990. *Principles of Concurrent and Distributed Programming*. Prentice Hall.

Benthem, J. 2002. Extensive games as process models. *Journal of Logic, Language, and Information* 11(3):289–313.

Binmore, K. 1994. *Game Theory and the Social Contract Volume 1: Playing Fair*. The MIT Press: Cambridge, MA.

Binmore, K. 1998. *Game Theory and the Social Contract Volume 2: Just Playing*. The MIT Press: Cambridge, MA.

Bond, A. H., and Gasser, L., eds. 1988. *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann Publishers: San Mateo, CA.

Brams, S. J., and Fishburn, P. C. 2002. Voting procedures. In Arrow, K. J.; Sen, A. K.; and Suzumura, K., eds., *Handbook of Social Choice and Welfare Volume 1*. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands. chapter 4.

Campbell, D. E., and Kelly, J. S. 2002. Impossibility theorems in the arrovian framework. In Arrow, K. J.; Sen, A. K.; and Suzumura, K., eds., *Handbook of Social Choice and Welfare Volume 1*. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands. chapter 1.

Clarke, E. M.; Grumberg, O.; and Peled, D. A. 2000. *Model Checking*. The MIT Press: Cambridge, MA.

Conitzer, V. 2006. Computing slater rankings using similarities among candidates. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06)*.

Cramton, P.; Shoham, Y.; and Steinberg, R., eds. 2006. *Combinatorial Auctions*. The MIT Press: Cambridge, MA.

Drimmelen, G. 2003. Satisfiability in alternating-time tem-

poral logic. In *Eighteenth Annual IEEE Symposium on Logic in Computer Science (LICS 2003)*, 208–217.

EBAY. 2001. The eBay online marketplace. See `http://www.ebay.com/`.

Emerson, E. A. 1990. Temporal and modal logic. In van Leeuwen, J., ed., *Handbook of Theoretical Computer Science Volume B: Formal Models and Semantics*. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands. 996–1072.

Fagin, R.; Halpern, J. Y.; Moses, Y.; and Vardi, M. Y. 1995. *Reasoning About Knowledge*. The MIT Press: Cambridge, MA.

Hoek, W., and Wooldridge, M. 2002. Tractable multiagent planning for epistemic goals. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002)*, 1167–1174.

Hoek, W., and Wooldridge, M. 2003a. Model checking cooperation, knowledge, and time — a case study. *Research in Economics* 57(3):235–265.

Hoek, W., and Wooldridge, M. 2003b. Time, knowledge, and cooperation: Alternating-time temporal epistemic logic and its applications. *Studia Logica* 75(1):125–157.

Hoek, W., and Wooldridge, M. 2005a. On the dynamics of delegation, cooperation, and contol: A logical account. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2005)*, 701–708.

Hoek, W., and Wooldridge, M. 2005b. On the logic of cooperation and propositional control. *Artificial Intelligence* 164(1-2):81–119.

Hoek, W.; Lomuscio, A.; and Wooldridge, M. 2005. On the complexity of practical ATL model checking. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2006)*.

Hoek, W.; Roberts, M.; and Wooldridge, M. 2005. Knowledge and social laws. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2005)*.

Hoek, W.; Roberts, M.; and Wooldridge, M. 2007. Social laws in alternating time: Effectiveness, feasibility, and synthesis. *Synthese* 156(1):1–19.

Holzmann, G. J. 2003. *The SPIN Model Checker: Primer and Reference Manual*. Addison-Wesley: Reading, MA.

Jamroga, W., and van der Hoek, W. 2004. Agents that know how to play. *Fundamenta Informaticae* 63(2-3):185–219.

Kraus, S. 2001. *Strategic Negotiation in Multiagent Environments*. The MIT Press: Cambridge, MA.

Krishna, V. 2002. *Auction Theory*. The Academic Press: London, England.

Moore, R. C. 1990. A formal theory of knowledge and action. In Allen, J. F.; Hendler, J.; and Tate, A., eds., *Readings in Planning*. Morgan Kaufmann Publishers: San Mateo, CA. 480–519.

Moses, Y., and Tennenholtz, M. 1995. Artificial social systems. *Computers and AI* 14(6):533–562.

Nisan, N., and Ronen, A. 1999. Algorithmic mechanism design. In *Proceedings of the Thirty-first Annual ACM Symposium on the Theory of Computing (STOC-99)*, 129–140.

Osborne, M. J., and Rubinstein, A. 1994. *A Course in Game Theory*. The MIT Press: Cambridge, MA.

Pauly, M., and Wooldridge, M. 2003. Logic for mechanism design — a manifesto. In *Proceedings of the 2003 Workshop on Game Theory and Decision Theory in Agent Systems (GTDT-2003)*.

Pauly, M. 2001. *Logic for Social Software*. Ph.D. Dissertation, University of Amsterdam. ILLC Dissertation Series 2001-10.

Pauly, M. 2002. A modal logic for coalitional power in games. *Journal of Logic and Computation* 12(1):149–166.

Rosenschein, J. S., and Zlotkin, G. 1994. *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*. The MIT Press: Cambridge, MA.

Sandholm, T. 1999. Distributed rational decision making. In Weiß, G., ed., *Multiagent Systems*. The MIT Press: Cambridge, MA. 201–258.

Shoham, Y., and Tennenholtz, M. 1992. On the synthesis of useful social laws for artificial agent societies. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*.

Shoham, Y., and Tennenholtz, M. 1997. On the emergence of social conventions: Modelling, analysis, and simulations. *Artificial Intelligence* 94(1-2):139–166.

van Otterloo, S.; Hoek, W.; and Wooldridge, M. 2004. Preferences in game logics. In *Proceedings of the Third International Conference on Autonomous Agents and Multiagent Systems (AAMAS-04)*, 152–158.

Walther, D.; Lutz, C.; Wolter, F.; and Wooldridge, M. 2006. ATL satisfiability is indeed ExpTime-complete. *Journal of Logic and Computation* 16:765–787.

Weiß, G., ed. 1999. *Multi-Agent Systems*. The MIT Press: Cambridge, MA.

Wooldridge, M., and Dunne, P. E. 2004. On the computational complexity of qualitative coalitional games. *Artificial Intelligence* 158(1):27–73.

Wooldridge, M., and Dunne, P. E. 2006. On the computational complexity of coalitional resource games. *Artificial Intelligence* 170(10):853–871.

Wooldridge, M., and van der Hoek, W. 2005. On obligations and normative ability. *Journal of Applied Logic* 3:396–420.

Wooldridge, M. 2002. *An Introduction to Multiagent Systems*. John Wiley & Sons.

Fixed automation, also known as â€œhard automation,â€ refers to an automated production facility in which the sequence of processing operations is fixed by the equipment configuration. In effect, the programmed commands are contained in the machines in the form of cams, gears, wiringÂ One of the most important application areas for automation technology is manufacturing. To many people, automation means manufacturing automation. We begin by reviewing the mechanism design problem, and discussing the issue of how mechanisms might be specified and verified. We then discuss a logic (Alternating-time Temporal Logic â€” ATL), which is well suited to representing and reasoning about strategic encounters. We show how ATL can be used to formally specify two social choice mechanisms, and we demonstrate how current model checkers for ATL can be used to verify properties of these mechanisms. We conclude by discussing issues for future research.Â Logic for Automated Mechanism Design - A Progress Report. M. Wooldridge, Thomas Ã…gotnes, â€¦ W. HoekAAAI2007. Algorithmic mechanism design, and the recent emergence of computational social choice theory are two examples of this growth of interest. If we take seriously the idea that computational agents will participate in economically inspired mechanisms, then it is natural to consider the questions of knowledge representation and reasoning for them.Â The talk will report joint work with Thomas Agotnes (Bergen), Wiebe van der Hoek (Liverpool), Marc Pauly (Stanford), and Paul E. Dunne (Liverpool). Keywords.Â Cite this paper as: Wooldridge M. (2008) Logic for Automated Mechanism Design and Analysis. In: Bergmann R., Lindemann G., Kirn S., PÄ›chouÄek M. (eds) Multiagent System Technologies. MATES 2008.