ELSEVIER

Artificial Intelligence 73 (1995) 231-252

Artificial Intelligence

On social laws for artificial agent societies: off-line design [★]

Yoav Shoham, Moshe Tennenholtz [*]

Robotics Laboratory, Department of Computer Science, Stanford University, Stanford, CA 94305, USA

Received January 1992; revised November 1993

Abstract

We are concerned with the utility of social laws in a computational environment, laws which guarantee the successful coexistence of multiple programs and programmers. In this paper we are interested in the off-line design of social laws, where we as designers must decide ahead of time on useful social laws. In the first part of this paper we suggest the use of social laws in the domain of mobile robots, and prove analytic results about the usefulness of this approach in that setting. In the second part of this paper we present a general model of social law in a computational system, and investigate some of its properties. This includes a definition of the basic computational problem involved with the design of multi-agent systems, and an investigation of the automatic synthesis of useful social laws in the framework of a model which refers explicitly to social laws.

1. Introduction

This paper is concerned with the utility of social laws in a computational environment, laws which guarantee the successful coexistence of multiple programs and programmers. We imagine an environment in which multiple planners/actors are active; we will call these "agents" here, without attaching precise technical meaning to the term (but see the formal development in the next sections). These agents' actions, and perhaps also goals, may be either facilitated or hindered by those of others.[1]

[★] This work was supported in part by a grant from the US-Israel Binational Science Foundation.

[*] Corresponding author. Current address: Industrial Engineering and Management, Technion - Israel Institute of Technology, Haifa 32000, Israel. E-mail: moshet@ie.technion.ac.il.

[1] As a special case, we are interested in extending the framework of Agent Oriented Programming (AOP) [15]. This framework, which attaches a precise meaning to the term "agent" (briefly, it defines agents to have mental state, consisting of components such as beliefs and commitments), currently accounts for the design of individual agents; we are interested in a theory for the design of successful societies of such agents. However, this paper will not be specific to the AOP framework.

To illustrate the issues that come up when designing a society, consider the domain of mobile robots. Although still relatively simple, state-of-the-art mobile robots are able to perform several sorts of tasks. They can move from place to place, identify and grasp simple objects, follow moving objects, and so on. Each of these tasks involves sophisticated techniques, but is, broadly speaking, achievable with existing planning and control technology. However, when we consider gathering several robots in a shared environment, a host of new problems arises. The activities of the robots might interfere with one another: The planned space-time paths of robots might intersect, an object needed by one robot might be removed by another, bottlenecks might occur, and so on. Similarly, robots may require the assistance of other robots to carry out their task: It may take two robots to lift a particular object, one robot may lack a capability possessed by another, and so on.

How is one to deal with these phenomena? There are two extreme answers, neither of which is in general acceptable. One is to assume a single programmer, whose job it is to program all the robots. As such, he will need to worry about all possible interactions among them, in exactly the same way as he worries about the interactions among the different actions of a single robot (this is the approach usually taken in robotics; see for example [1,5]). This answer is unsatisfactory for several reasons. First, it is unreasonable to expect that a single individual will control all agents. For example, in the Gofer project [2] the goal is to populate the planned Information Sciences building at Stanford with 100 or so mobile robots, each programmed individually by members of the department in much the same way as workstations are used. Second, the set of agents can be expected to change over time, and one would hardly want to have to reprogram all agents upon each addition or deletion of an agent. Finally, even if a single programmer was given the task of programming all the agents, we have given him no guidelines as to how to achieve the task. To the extent that he will proceed by programming each agent individually, he will be forced to deal with the interactions among the different programs.

An alternative extreme answer is to admit that agents will be programmed individually in an unconstrained fashion, to acknowledge that as a result interactions will occur, and to equip the agents with the means for handling these interactions during execution. These means may take various forms. For example, one approach is to merely detect the interactions as they occur, and appeal to a central supervisor for resolution. An alternative method for handling interactions is to equip the agents with communication capabilities, and program them to engage in series of communications to resolve interactions. Again, using the domain of mobile robots for illustration, when two robots note that they are on a collision course with one another, they may either appeal to some central traffic controller for coordination advice, or alternatively they might engage in a negotiation resulting (say) in each robot moving slightly to its right. The use of negotiation to resolve conflicts is common in the distributed AI literature (see for example [3,7,14]).

Nonetheless, there are limitations to this second approach as well. By placing no constraints in advance, the number of interactions may be prohibitive; either the central arbiter may be deluged by pleas, or else agents will have to enter into negotiations at every step. While we certainly do not argue against the utility of either a central coordinator or a negotiation mechanism, we do argue that it is essential to add a mechanism that will minimize the need for either. Again, we draw on the domain of mobile robots for intuition. Suppose robots navigate along marked paths, much like cars do along streets. Why not adopt a convention, or, as we'd like to think of it, a social law, according to which each robot keeps to the right of the path? If each robot obeys the convention, we will have avoided all head-on collisions without any need for either a central arbiter or negotiation.

This then is the image, which is not original to us; it is implicit in many places in AI, and was made explicit already by Moses and Tennenholtz [10, 11]. The society will adopt a set of laws; each programmer will obey these laws, and will be able to assume that all others will as well. These laws will on the one hand constrain the plans available to the programmer (in the above example, plans which call for driving on the left are ruled out), but on the other hand will guarantee certain behaviors on the part of other agents. The two approaches discussed above simply mark the endpoints of this tradeoff continuum. The first, "single programmer" approach stretches the notion of a social law so as to completely dictate the behavior of each agent, leaving no freedom to the individual programmer. The second approach adopts an opposite, degenerate form of social law, the vacuous law. The goal of the intended theory of social laws will be to relax the restriction to these extreme solutions, and instead to strike a good balance between allowing freedom to the individual programmers on the one hand and ensuring the cooperative behavior among them on the other.

How is one to decide on appropriate social laws? One approach is to hand-craft laws for each domain of application. This is the approach we take in the first part of this work, where we present a number of traffic laws for a restricted domain of mobile robots; in this part we show social laws which ensure that no collisions or deadlocks occur, and agents are still allowed enough freedom to plan close to optimal paths.

In the second part of the paper we tackle a different problem; we are interested in a general model of social law in a computational system, and in general properties that can be proved in that model. We argue that the notion of social law, or constraints, is not epiphenomenal, but rather should be built into the action representation; we then offer such a representation. Second, we define the basic computational problem involved in the design of multi-agent systems in the framework of this model. In addition, we investigate the complexity of *automatically* deriving useful social laws in this model, given descriptions of the agents' capabilities, and the goals they might encounter over time. We show that in general the problem is NP-complete, and identify precise conditions under which it becomes polynomial.

The remainder of the paper is structured as follows. Section 2 is devoted to the case study of mobile robots. Section 3 is devoted to the more general model and computational problem. In Section 2.1 we present the multi-robot grid system. In Section 2.2 we introduce two sets of traffic laws for that system. The first is trivial, restrictive, and produces inefficient behavior on the part of the robots. The second is more flexible, and allows very efficient behaviors while still avoiding collisions among the robots. Section 2.3 briefly discusses a number of extensions to the basic framework. In Section 3.1 we set up the formal (and, we believe, natural) model of multi-agent actions. In Section 3.2 we state the computational problem precisely, and show that in its full generality the problem is NP-complete. We also show a similar result for one natural restriction on the general model. In Section 3.3 we formulate additional natural restrictions on the general model; we show that these conditions are both necessary and sufficient for the problem to become polynomial. Finally, in Section 3.4 we look at a special case in which the (possibly many) states of agents happen to be encodable concisely, and show that there too the problem is polynomial. We conclude with a summary, discussion of related work, and planned future work. The proofs of the theorems concerning the traffic laws appearing in Section 2 follow from the details of these laws. Proof sketches of the theorems stated in Section 3 appear in an enclosed appendix.

2. A case study: mobile robots

Building efficient multi-robot systems is an attractive task for the relatively near future. Although it might be the case that building a sophisticated robot is a very complex task, gathering many simple state of the art robots in a shared environment can be attacked at this point and might be simpler than the task of building a single sophisticated robot. In this section we investigate the idea of social laws for artificial agent societies in the context of mobile robots. Our aim is to prove analytic results about the usefulness of our approach in that framework. The specific domain we look at is an idealized one. Hence, although the results presented in this section can be used for practical systems, the reader should treat them mainly as a nontrivial demonstration of the artificial social systems approach in a particular domain of application.

2.1. The multi-robot grid system

The domain we investigate will be called the *multi-robot grid system*. It consists of an $n \times n$ grid, and a set of m robots. It might be useful for the reader to keep in mind that the rows and columns of the grid can refer to lanes in a supermarket, or to paths in a warehouse, etc. When mobile robots have to move along these lanes and paths we get a multi-robot grid system structure. At each point in time each robot is located at some grid coordinate. The fact that several robots occupy the same coordinate denotes a collision among them; part of our goal as designers of the system will be to ensure that collisions are avoided. We assume that each robot has the capability at each stage to move to one of its neighbor coordinates or stay in place. Each such operation lasts a certain amount of time and consumes a certain amount of energy. We assume that the system is synchronous and that movement to a neighbor coordinate takes one time unit and consumes one energy unit, unless this motion follows a stop (that is, taken from a resting position). In the latter case we assume that movement to a neighbor coordinate takes c time units and consumes c energy units, where $c > 1$ (identical constants are used for time and energy only for ease of exposition). Remaining motionless wastes no energy, although of course it does waste time.

This is the bare-bones framework, on which we concentrate in this paper, but it can be extended in a number of ways. In order to reason about assembly tasks, we will extend the system to contain (in the spaces between coordinates) various items and substances that the robots might need for their tasks. Or, we may allow robots to travel at different velocities. Such extensions are discussed briefly in Section 2.3.

We have so far discussed *how* robots might travel, but not *why*. For the latter we assume that at random times robots are given target locations, and their goal is to reach those destinations, minimizing either time, or energy consumption, or both (they are not required to stay at the location for any period of time, just get there). In the simplest case, at any time each robot has at most one target destination; in the general case robots may accumulate several destinations. In addition, robots are required to avoid collisions.

The job of the programmer of an individual robot will be to implement an efficient path-planning program. In the absence of other robots, the task is conceptually simple: In the case in which at all times each robot has at most one destination it is trivial, and in the general case it is some variation on the familiar Traveling Salesman Problem. However, the existence of other robots may render useless an otherwise perfect plan. There has been some work in robotics on the coordinated operation (and, in particular, motion) of multiple robots (see for example [1,5]). However, much of this work has assumed a centralized controller which has full information about the system. As we have argued in the introduction, this assumption, while reasonable in some applications, is in general untenable, as would be a decentralized solution which relied only on on-line negotiations among robots. Although there exist local techniques for collision avoidance, such as artificial potential fields (see [6]), the reader should not confuse them with a global mechanism of coordination such as social laws.

Again, without detracting from the merit of the above mechanisms, we argue that it is necessary to design social (or, in this special case, traffic) laws to minimize the need for on-line conflict resolution. In fact, we will be interested in eliminating this need altogether. In other words, our job as designers of this small social system will be to impose on it traffic laws which will guarantee that no collisions will occur—provided that all programmers abide by the law. However, in our zeal to prevent collision we must be careful not to preclude the possibility of robots reaching their destinations (so, for example, forbidding all motion would be inappropriate). In fact, we will be interested in traffic laws that not only allow robots to reach their destinations without collision, but allow them to do so reasonably efficiently.

2.2. *First two traffic systems*

Any system of laws must assume certain capabilities on the part of agents. For example, a law against disposal of garbage in public places assumes the ability to identify garbage as well as public places. The situation regarding traffic laws is similar, and our laws will differ depending on assumptions we can make about the robots. The assumptions that will be relevant here regard sensory capabilities: Can robots detect other robots? If so, at what distance? In this section we will consider two cases: In the first the robots lack all sensory capabilities, and in the second they can detect "close" robots.

From this point on we will assume that n is even. Rows and columns in the grid will be numbered from bottom to top, and from left to right, respectively. We assume that $m \leqslant n$, but that m is still large. In the rest of this paper with the exception of Section 2.3.4 we will assume that $m = O(\sqrt{n})$.[2]

We start by assuming that robots cannot sense each other's presence. The question is whether there exists a traffic law which will allow the robots to reach their destinations guaranteeing that no collisions occur. The answer is positive, although perhaps in a somewhat disappointing way. The first traffic law requires that all robots snake their way around the grid in a regular pattern, passing sequentially through all nodes. More explicitly, we have the following:

Traffic Law 1. Each robot is required to move constantly. The direction of motion is fixed as follows. On even rows each robot must move left, while in odd rows it must move right. It is required to move up when it is on the rightmost column. Finally, it is required to move down when it is on either the leftmost column of even rows or on the second rightmost column of odd rows. The movement is therefore in a "snake-like" structure, and defines an Hamiltonian cycle on the grid.

It is trivial to show that the following holds, even when robots are given new goals while still on the way to an existing one:

Theorem 2.1. *Traffic Law 1 guarantees that that no collisions will occur, and that each agent will reach each goal it is given in* $O(n^2)$ *steps from the time it is given that goal.*

Clearly, Traffic Law 1 is an example of the extreme case discussed in the introduction, in which the laws completely dictate the behavior of agents, leaving no room for individual planning. Intuitively, such rigid laws lead to suboptimal behavior, and indeed, while the Traffic Law 1 does guarantee a certain minimal performance, this performance is quite suboptimal: Robots might have to snake around the entire grid just to get to a nearby node. It is possible to do much better, but for that we must make further assumptions about the perceptual capabilities of the robots. In the following we assume that each robot in the multi-robot grid system is able to perceive robots that are at a distance of one coordinate away from it (including robots that are at a distance of one on both rows and columns).

One use of the perceptual capability could be to assign each robot a unique number, and rely on time-slicing to avoid collisions (each time slice would be devoted to one of the robots, which would be able to move to an unoccupied neighboring node). This class of solutions has a number of drawbacks. Chief among them are the facts that robots might be blocked and that in such solutions each robot ends up spending either $O(mn)$ or $O(mt)$ time units, where t is the complexity of the best solution in the absence of other robots (recall that m is the number of robots and $n \times n$ is the grid size); we would

[2] We recall the standard big-O and little-o notation. Given a pair of functions $f(n)$ and $g(n)$, $g(n) = O(f(n))$ means that $g(n)/f(n)$ converges to a constant when n converges to infinity. Similarly, $g(n) = o(f(n))$ means that $g(n)/f(n)$ converges to 0 when n converges to infinity.

like a solution that is truly linear in n or, yet better, in t. An additional drawback is that we envision an open system in which robots may be added and removed without having to reprogram the entire system.

We opt for a different use of the sensory capabilities. The capability to perceive the immediate environment is similar to our experience in everyday life, and our second traffic law too derives from that experience. The basic idea is to adopt familiar traffic laws such as giving way to a robot arriving from the right. However, naive laws of this kind quickly give rise to deadlocks and related problems, and so we will craft a somewhat more sophisticated law. The following law might seem strange at first. However, its importance follows from the results we are able to prove concerning it. In fact, all parts of this law excluding the sixth restriction of it are quite natural and can be related to our everyday experience. However, this sixth restriction is crucial for the particular results we got. Moreover, this restriction seems to be extremely useful when we consider modifications of the basic framework (see Section 2.3.4).

We now briefly discuss the idea of the following law, and then present its details. The basic idea is to superimpose on the original grid a coarser one. The key point is that when agents move on the coarse grid several simple traffic laws will prevent them from getting stuck, and will guarantee that each robot will have to wait only a small number of times for any other robot before entering an internal grid (leaving the coarse grid). In an internal grid we ensure that each robot is able to reach any possible coordinate of that grid. This requires a somewhat complex behavior presented by the sixth requirement of the following law.

Traffic Law 2.

(1) Superimpose on the original grid a coarser one. The squares of the coarser grid will be subgrids of size $2m \times 2m$ of the original grid. A sketch of the "multi-grid" created appears in Fig. 1.[3]

(2) In odd rows of the coarse grid robots will be required to move only right, while in even rows of it they will be required to move only left.

(3) In odd columns of the coarse grid robots will be required to move only down while in even columns of it they will be required to move only up.

(4) In junctions of the coarse grid, priority will be given on the basis of "first in first out". More specifically, when a robot is in a distance of 1 from entering one of the coarse grid's junctions, then it should not enter that junction if there is another robot which is in a distance of 1 from that coarse grid's junction and arrived there first. The default priority (in the case where robots arrive to the corresponding locations at the same time) will always be given to robots moving on columns.

(5) While a robot is moving along the coarse grid, it is allowed to change its movement direction (from "along a row" to "along a column" and vice versa)

[3] Our results hold also for smaller coefficients of m, but although this would make our results even stronger, we will not discuss these cases here. Similarly, for ease of exposition we assume that the coarse grid "fits nicely" on top of the original fine grid. Our results don't depend on this assumption.

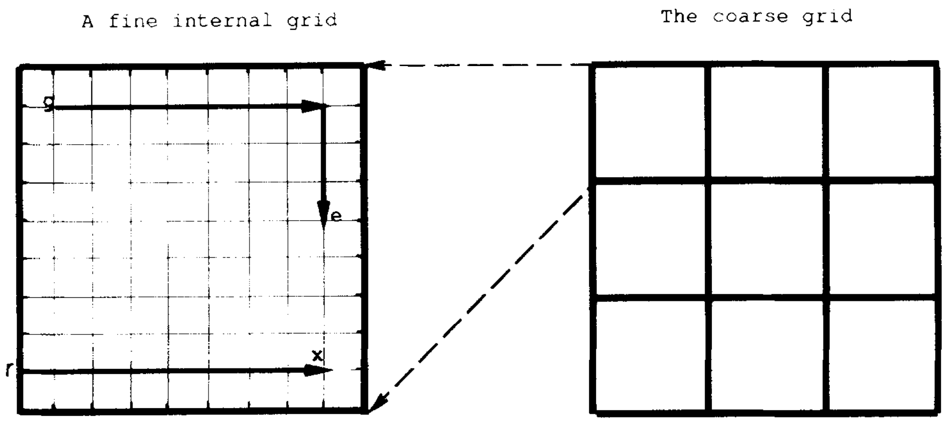

Fig. 1. A coarse grid imposed on the fine grid.

only k times before leaving the coarse grid (and entering an internal grid), where k is a small constant.

(6) Inside each $2m \times 2m$ grid each robot will move in the following way (see Fig. 1): The entrance point to the grid will be from the leftmost coordinate of the row denoted by r (from the leftmost coordinate of that grid's "lowest internal row"). The robot will then be required to move along the row r to the coordinate denoted by x (the rightmost internal coordinate of row r). During the movement along r, if the robot perceives the existence of a robot in the coordinate in front it or finds that it reached x, then it has to stop and wait for $2n+cmk$ time units. Then, it has to continue (if still necessary) along r to x. After reaching that coordinate (x) and waiting there (if still necessary) for the above mentioned period, the robot has to reach the coordinate denoted by g (which is in distance $4m-2$ from it) in exactly $4m-2$ steps without entering any of the areas appearing in thick lines (both in the internal grid and on the coarse grid) in Fig. 1. Afterwards, the robot will be required to move from g to the coordinate denoted by e along the path appearing in thick lines in Fig. 1, and leave that grid from there.

(7) Robots are not allowed to stop moving unless the robot in front of them does not move or when they are required otherwise by the social law.

(8) The points of return from an $2m \times 2m$ grid to the coarse grid will be considered as additional junctions of the coarse grid (i.e. the traffic laws of the coarse grid will have to be obeyed there).

The following two theorems show that Traffic Law 2 has attractive properties; they both assume that there are $m = O(\sqrt{n})$ robots in the $n \times n$ multi-robot grid system, and that the robots can perceive their neighbors (that is, those robots that are at distance of at most one coordinate away along rows and columns).

Theorem 2.2. *Traffic Law 2 guarantees that no collisions occur.*

Theorem 2.3. *Traffic Law 2 guarantees that, given a goal which can be achieved in isolation (that is, assuming no other robots are present) in t time and energy units,*[4] *there exists a plan that achieves it in $t + 2n + o(n)$ time units, and (for t satisfying $m = o(t)$), in $t + o(t)$ energy units.*

The proofs of these theorems follow from the details of Traffic Law 2. Here we only make the following remarks. The fact that we have a coarser grid enables the agents to not get stuck at junctions. By having complex movements only in the $2m \times 2m$ grids, the robots will move freely most of the time (while keeping distance from others). The law enables each robot to achieve its goal while waiting at most k times for each other robot. This is achieved by the traffic arrangements in the coarse grid, and by the fact that each robot waits for the other robots to reach their goals when it reaches an internal grid. The above need to wait for others might be sometimes relaxed, and then we are able to get even better results (see Section 2.3.1). The traffic law enables to reach an internal grid, in which the goal is, in a simple way. This is due to the rules of behavior regarding movement on the coarse grid. Inside an internal grid each coordinate is reachable (i.e., liveness is guaranteed). Whenever two different robots enter an internal grid and move towards the coordinate g, the one that enters the internal grid earlier is always closer to g than the other one (i.e., safety is guaranteed; it is easy to see that after a robot reaches g it cannot collide with others given the above traffic law).

Notice that although the robots are assumed to obey the traffic law, there is still much freedom (and local planning to be done) for the robots. In a typical problem an agent might need to visit several coordinates. Planning the shortest path will be part of the planning stage. This decision might depend on many possible criteria that are *not* known to the designer(s) and to other robots. However, when a robot decides that its next destination is a certain coordinate then it can reach it efficiently (in a reasonable amount of time and with close to optimal consumption of energy), *regardless of the goals and plans of other agents*—as long as all agents play by the rules of the social system.

Notice that the average time which is required for a robot working in isolation to reach a coordinate is linear in n. Therefore, we get a relatively good coordination. For a goal which can be achieved in isolation in n time units, our social laws will guarantee that we will lose only a factor of 3 (in many cases we can do even better and get close to optimal solutions; see Section 2.3.1). If we are interested in the units of energy that are spent, and we consider destinations that are reasonably far, then the existence of other agents causes us to lose almost no energy at all.

There are a number of ways in which to improve, refine and extend the system; some of these are explored in the next section. Notice for example that the above traffic laws might require a robot which is assigned no goal to move constantly. In order to treat this issue we will have to specify locations where robots can "rest". This can be easily handled if we assume that there is enough space for two robots in each coordinate of the grid. However, in this paper we concentrate mainly in extensions where the original format of the grid will be preserved.

[4] Notice that for a robot working in isolation, the optimal path will be without stops. As a result, the time consumption and the energy consumption of the robot moving along that path will be identical.

2.3. *Various extensions*

In this section we briefly discuss various extensions to the basic framework presented in the previous sections.

2.3.1. *On a single robot ability and social law efficiency*

Recall that c is the amount of time and energy spent on the first movement after a stop. Recall also the qualitative difference between the energy and time consumptions guaranteed by Traffic Law 2. However, if we assume that c is small (say $c = 1.5$), then we can achieve excellent results concerning time consumption also. In this case, we will drop the requirement of waiting for $2n + ckm$ time units in an $2m \times 2m$ grid (this requirement is part of the sixth requirement of Traffic Law 2). If we denote the traffic law generated by dropping the above requirement from Traffic Law 2 by Traffic Law 3, then we can show that Traffic Law 3 guarantees that instead of spending t time units (when working in isolation) each robot will need to spend only about ct time units in the multi-agent case.[5] This result sheds light on some interesting practical issues. For example, making robots that can accelerate without wasting much energy and in a relatively fast fashion might be (as is shown here) more important than building a sophisticated communication scheme for coordinating activity. On the other hand, Traffic Law 2 shows that if our main interest is to minimize the energy spent by the robots (while still requiring that their time consumption will be reasonable), then the appropriate way for achieving it might not necessarily be the option in which they will finish each task in the fastest way. In our traffic law, when a robot has to wait for $2n + ckm$ time units, it can shut its engine and batteries and spend no energy during that period (no perception or movement is needed). As a result, each robot will be able to achieve close to optimal energy consumption even for large c.

2.3.2. *Allowing multiple velocities*

Other interesting issues arise when we consider robots moving in different velocities (we capture the notion of velocity by enabling robots to move along several coordinates in one time unit). In this subsection, we assume that moving in a certain velocity is a matter of decision. Each robot might have a different optimal velocity, and we assume that the cost of deviating from this velocity is proportional to the size of that deviation.[6] Therefore, we would like to enable the robots to move most of the time in a velocity which is close to their optimal velocity. We treat the above "velocities problem" by devoting different lanes on the coarse grid for different velocities while still requiring moving in a particular constant velocity inside the $2m \times 2m$ grids. If the robots' optimal velocities can range between v_1 to v_j, we can decide on s velocities, $v_i = v_1 + i \cdot (v_j - v_1)/s$, and associate with column and row l of the coarse grid the velocity v_i iff $l = i(\mathrm{mod}(s))$ (robots moving there will be required to move in that velocity). The velocity of movement inside each $2m \times 2m$ grid will be v_1. Now, we

[5] In fact this holds also for a large c, but then Traffic Law 2 will be unsatisfactory for distant goals.

[6] The case when a certain robot cannot move in a certain velocity which is optimal to another is an extreme case that (although interesting) will not be discussed in this paper.

augment Traffic Law 2 by the above mentioned requirements, and refer to the augmented law as Traffic Law 4. It is easy to show that Traffic Law 4 guarantees that each robot will be able to reach any goal it receives while moving in a velocity that differs by at most $(v_j - v_1)/s$ from its optimal velocity, with the exception of $o(n)$ coordinates along which it will have to move in a different velocity. However, before using this result, some cautions should be taken. Because of the fact that the $o(n)$ obtained is proportional to the size of s, then when considering an n which is not large we will still have to optimize the size of s for getting satisfying behavior (whenever n is larger, s will be chosen to be larger).

2.3.3. A few words on workstations

The next problems to be discussed are concerned with assembly issues. When considering assembly issues, there might be a need to change some of the traffic rules, because robots might need to stop at certain workstations in order to carry out assembly tasks. For example, if a robot decides to assemble a certain unit with/from the elements it gathered, then it might need to stop in order to perform the task. This might of course disturb others. We can treat the above problem using a similar idea to the idea used in the "velocities problem". This will be achieved by designating "workstation" areas, in which the assembly activity will be performed. The frequency of such locations should be calculated carefully in order to minimize the length of paths the robots will have to follow in order to achieve their goals. Different assembly locations will be devoted to different robots and to different expected assembly times in order to decrease the amount of unnecessary stops. Notice that all of the above is still concerned with traffic laws. However, the general framework suggested does not restrict itself to these laws. When we consider assembly issues, we might look at totally different social laws: return public tools to their places, notify other robots about the place of a certain tool you used, etc.

2.3.4. The $m = O(n)$ case

In the previous sections we showed how close to optimal coordination can be achieved when we are interested in the total energy consumption, and how such results are obtained concerning time consumption when the constant c is small. However, the optimality of these results relies on the assumption that $m = o(n)$. Therefore, when we consider a system of many robots we have to assume that our grid is quite large. An interesting complementary problem is concerned with our ability to devise social laws that will be good for the case where $m \leqslant n$ but still $m = O(n)$. Of course we can use the $O(n^2)$ result of Theorem 2.1, but the question is whether we can do better. The following traffic law relies on Traffic Law 2 and points to the fact that, when we are interested in reaching relatively distant goals, then we can achieve close to optimal coordination even in the $m = O(n)$ case.

Traffic Law 5. This traffic law will be similar to the sixth requirement of Traffic Law 2 when we replace the $2m \times 2m$ grid by the whole grid, and use the following modifications. Row r will be the bottom row of the whole grid, and x will be the second rightmost coordinate of that row. The coordinate g will be the leftmost coordinate of the top row of the grid, and the column of e will be the rightmost column of the grid. However, instead of stopping at e, each robot will have to move along the rightmost column until the bottom of the grid and then return back to x. When a robot is in x, it will have to reach g in one of the shortest paths possible without entering the top row. We assume that the robots start operating from the lowest row (but will have to return only to x and continue from there). In the first iteration (when first arriving at x) the robots must move from x to g without entering r.

We can show:

Theorem 2.4. *Given that there are $m = O(n)$, $m \leqslant n$, robots in the multi-robot grid system, Traffic Law 5 guarantees that each goal can be achieved in $4n$ time units.*

Notice that, if we are interested in reaching coordinates at a distance of about n, then this result can be considered as a very good performance. Notice also that no stops or perception are needed in order to achieve goals according to this law. In the case that robots might be working towards several goals, meaningful local planning will still be needed in order to try achieving several goals before reaching g. However, this traffic law is more restrictive than Traffic Law 2. This has a significant drawback: a robot might not be able to achieve all its goals before reaching g, and then it will have to achieve some of them in later iterations. This of course might badly affect the efficiency of the robots in cases where robots have several goals at the same point in time. The above situation points to the fact that although the social law must restrict the behavior of agents in order to enable individual achievement of goals, we have to be careful not to be too restrictive in order to enable the agents to devise efficient plans for achieving their goals.

3. The general framework

This section introduces a more general treatment of social laws in a computational setting. We see this part as complementary to the previous one. In fact, we believe that both of them have to be understood in order to have an initial theory of artificial social systems.

3.1. The model

The term "agent" is common in AI, and used in diverse ways. Here we will take an agent to have state, and to engage in actions which move it among states. Although this basic definition is consistent with most uses of the term in AI,[7] it is perhaps somewhat too impoverished to be worthy of the lofty name. However, our goal here is not to contribute to the theory of individual agents but to the theory of social law, and so it will be illuminating to initially adopt a model in which all the complexity arises from the existence of multiple agents, and not from the behavior of individual agents.

[7] For example, it is consistent with its definition in Agent Oriented Programming, mentioned earlier.

We adopt a synchronous model: Agents repeatedly and simultaneously take action, which leads them from their previous state to a new one. The actions of an agent are taken from a given repertoire. The problem in defining the transition functions of agents is due to the fact that the state in which the agent ends up after taking a particular action at a particular state depends also on actions and states of other agents. Thus, in principle, we could think of the transition function of the entire system, a mapping from the states and actions of all agents to the new states of all agents. In this view, for example, in order to determine the effect of one car turning left at an intersection, we would have to specify the states and actions of all other cars. An alternative view would be to define the transition function of each agent independently, and account for the effects of other agents by making the function nondeterministic. In this view, for example, the effect of turning left at an intersection would be either a new position and direction of the car, or a collision.

These are two extreme approaches to modelling concurrent actions. In the first approach, *all* information about other actions must be supplied, and the transition function produces the most specific prediction. In the second approach, *no* information about other agents is supplied, and the transition function produces the most general prediction. Instead of adopting either extreme view, we propose an intermediate approach. We propose adding the concept of *social law*, or constraints on the behavior of agents. The constraints specify which of the actions that are in general available are in fact allowed in a given state; they do so by a predicate over that state. The transition function now takes as an argument not only the initial state and the action taken by the agent, but also the constraints in force; it produces a prediction about the set of possible next states of the agent. For example, it might predict that as a result of turning left at the intersection, *given certain traffic rules*, the car will successfully complete the turn.

We claim that this is a natural representation of actions, and an advantageous one. Generally speaking, the prediction based on constraints will be more general than the prediction that is based on the precise states and actions of all other agents, and more specific than the prediction based on *no* information on the other agents.

A final comment, before we present the formal model. We make an assumption of *homogeneity*; specifically, we assume that the sets of states and the available actions are common to all agents. We do not assume that the agents will necessarily be in the same state at any time, nor that they will take the same action when in the same state; only that they "have the same hardware". Similarly, we assume an egalitarian society, in which the same constraints apply to all agents. None of these assumptions are crucial to our approach or results, but they simplify the discussion.

The formal model

Definition 3.1. Given a set of states S, a first-order language $\mathcal{L}$ (with an entailment relation $\models$), and a set of actions A, a *constraint* is a pair (a, φ) where $a \in A$ and $\varphi \in \mathcal{L}$ is a sentence. A *social law* is a set of constraints (a_i, φ_i), at most one for each $a_i \in A$.

The language $\mathcal{L}$ will be used to describe what is true and false in different states. Given a state $s \in S$ and a sentence $\varphi \in \mathcal{L}$, s might satisfy or not satisfy φ. We denote the fact that s satisfies φ by $s \models \varphi$. The intuitive meaning of (a_i, φ_i) will be that φ_i is the most general condition about states which *prohibits* taking action a_i.

In the sequel, we will use the following notation. Given a pair of social laws, sl_1 and sl_2, we denote by $sl_2 < sl_1$ the fact that for every $(a_i, \varphi_i) \in sl_2$ there exists $(a_i, \varphi_j) \in sl_1$ such that $\varphi_j \models \varphi_i$. Intuitively, it will mean that sl_1 is more restrictive than sl_2.

Definition 3.2. A *social agent* is a tuple $(S, \mathcal{L}, A, SL, T)$ where S, $\mathcal{L}$, A are as above, SL a set of social laws, and T is a total transition function $T : S \times A \times SL \rightarrow 2^S$ such that:

- For every $s \in S$, $a \in A$, $sl \in SL$, if $s \models \varphi$ holds and $(a, \varphi) \in sl$ then $T(s, a, sl) = \emptyset$, the empty set.
- For every $s \in S$, $a \in A$, $sl_1 \in SL$, $sl_2 \in SL$, if $sl_2 < sl_1$ then $T(s, a, sl_1) \subseteq T(s, a, sl_2)$.

In practice, the transition function T will be only partially specified. If $T(s, a, sl)$ is not explicitly defined for a particular sl, then $T(s, a, sl)$ is assumed to be the conjunction of all explicitly defined $T(s, a, sl_i)$ satisfying that $sl_i < sl$. If this conjunction is over an empty set, then $T(s, a, sl) = \emptyset$, the empty set.

Definition 3.3. A *social multi-agent system* is a collection of social agents which share the set of states, the language for describing states, the set of potential actions, the set of potential social laws, and the transition function.[8]

3.2. The computational problem

The multi-agent system defined in the previous section provides one degree of freedom in addition to those present in standard multi-agent models: the social law in effect. Once we fix the social law, the social multi-agent system reduces to a standard one, since all transitions which are incompatible with this law are now ignored. (Note, however, that the remaining transitions may still be nondeterministic.) Thus, a social multi-agent system and a particular social law together induce a standard multi-agent system, which makes no reference to social laws. Loosely speaking, the computational problem will be to select from among the candidate social laws one that, given the social multi-agent system, will induce a "good" standard system. But what makes for a "good" system?

For this purpose we identify a subset of the set of states, which we call *focal states*. We will be interested in a social law which will ensure that each agent, given two focal states, is able to construct a plan guaranteed to move him from one state to the other—*no matter what actions are taken by other agents*. Note the existence of a certain tradeoff: The more restrictive the law, the more can the planner rely on the effects of his actions, but, on the other hand, the more restrictive the law, the fewer actions are available to the planner in the first place. In selecting a social law we wish to strike a

[8] Recall again that this only means that the agents "have the same hardware", not that they will in fact be in the same state or take the same actions in a given state.

good balance, allowing each agent enough freedom to achieve its goals but not enough freedom to foil those of others.

Before proceeding with the formal development, we note two (probably obvious) points regarding the complexity of deriving useful social laws:

- Unlike planning, which occurs frequently over time, social laws need only be computed once (assuming that the characteristics of the system do not change; see discussion in last section). We might therefore not mind spending more time in this off-line stage of deriving the social laws. Still, if we show (say) an exponential lower bound on the problem of deriving these laws, it will limit the sizes of the problem for which we can expect a solution, even off-line.
- It might at first seem that deriving social laws is inherently harder than planning, since in the former "one must simulate the planning of agents anyway". This is clearly false; for example, the utility of the law requiring keeping to the right of the road can be demonstrated without simulating the route-planning agents might perform.

We now turn to the formal development. Intuitively, an agent's *legal plan* is a decision on how to act in each state, in a way which is consistent with the law in force:

Definition 3.4. Given a social agent $(S, \mathcal{L}, A, SL, T)$ and a social law $sl \in SL$, a *legal plan* is a total function $DO : S \longrightarrow A$ such that if $(a, \varphi) \in sl$ and $s \models \varphi$ holds, then $DO(s) \neq a$. An *execution* of the plan from a state s_0 is a sequence of states $s_0, s_1, s_2, \ldots$ such that $s_{i+1} \in T(s_i, DO(s_i), sl)$.

Note that a plan forces the agent to take action at every step, but we certainly allow the user to include the null action, which might leave states unchanged. Also note that even if the null action is included, some social laws may prohibit it in certain states!

Definition 3.5. Given a social multi-agent system and a subset F of the set of states (the focal states), a *useful law* is a law for which, given any $s_1, s_2 \in F$, there exists a legal plan such that *every* execution of that plan in s_1 includes s_2.

Note that, strictly speaking, we cannot speak of a plan "leading" to the goal, since the plan is by definition infinite. Also note that this definition does *not* mean that there necessarily exists one *fixed* sequence of actions that will connect two focal states; an action of the agent may have nondeterministic effects, and the following action in the execution of the plan may depend on the state in which the agent landed.

We are now in a position to phrase a precise computational problem:

Definition 3.6 (*The Useful Social Law Problem* (*USLP*)). Given a social multi-agent system and a set of focal states, find a useful law if one exists, or, if no such law exists, announce that this is the case.

The technical results of this section concern the computational complexity of the USLP. In order to present quantitative results we have to be more precise about some of the details of our model. In the following we will assume that the number of states in the representation of an agent is finite and is denoted by n, and we will measure the computational complexity as a function of n. We assume that the total size of an agent's representation is polynomial in n. We also assume that each property of the form $s \models \varphi$, and of the form $sl_1 < sl_2$ can be efficiently verified.

The following theorem shows that the general USLP is intractable, although its complexity is lower than the complexity of many other problems discussed in multi-agent activity.

Theorem 3.7. *The USLP is NP-complete.*

We point again that, since the USLP is computed off-line, this result is not entirely negative. Still, it would be satisfying to be able to achieve lower complexity by imposing various restrictions on the structure of agents. This is what we do in the next two sections.

3.3. Several restrictions on the general model

We start by imposing the following restriction: For each state s, the number of transitions which might change s is bounded by $O(\log(n))$.[9] This is a straightforward generalization of the natural "bounded fan-out" restriction which is common for classical automata. Intuitively, this restriction says that the number of actions an agent might perform at a given state is small relative to the total number of states, while the quality of the information about constraints which might be relevant to the effects of a particular action in a particular state is relatively high. Our logarithmic bound enables us to treat the case in which the number of transitions which are applicable in any given state is small relative to the total number of states, while it is still a function of that number. Notice that the total number of actions and social laws appearing in the representation might still be much more than logarithmic.

The computational problem related to the above restriction is defined as follows:

Definition 3.8 (*The Bounded Useful Social Law Problem* (*BUSLP*)). Given a social multi-agent system where the number of transitions which might change a particular state is bounded by $O(\log(n))$, and a set of focal states, find a useful law if one exists, or, if no such law exists, announce that this is the case.

On its own, this natural restriction does not buy us a whole lot as far as the computational complexity goes:

Theorem 3.9. *The BUSLP is NP-complete.*

However, we have not presented this restriction as a mere curiosity; we now present precise conditions under which the BUSLP become tractable. Consider the following three restrictions:

[9] If $T(s, a, sl_1) = T(s, a, sl_2)$ for $sl_2 < sl_1$, then we do not count $T(s, a, sl_1)$ as one of the transitions.

(1) The number of focal states is bounded by a constant c.
(2) For any pair of focal states $s_1, s_2 \in F$, there exists a legal plan all of whose executions reach s_2 starting at s_1 *while visiting the same sequence of states*. Intuitively, this requirement states that it is not enough that there be a plan, as required in the definition of a useful law; the plan must be deterministic.
(3) For any pair of focal states $s_1, s_2 \in F$, there exists a legal plan all of whose executions reach s_2 starting at s_1 *in no more than* $O(\log(n)/\log(\log(n)))$ *steps*. Intuitively, this requirement states that it is not enough that there be a plan; the plan must be short.[10]

These three restrictions may or may not be acceptable; although we expect that they are reasonable in some interesting applications, we take no stance on this here. The significance of these restrictions is that they allow us to state precise conditions under which the BUSLP becomes polynomial:

Theorem 3.10. *The BUSLP is polynomial if restrictions* (1), (2), *and* (3) *hold; if we drop any of these restrictions (and do not add additional ones), then the BUSLP is NP-complete.*

3.4. Succinct representation of agents

In the previous section we provided necessary and sufficient conditions under which the BUSLP becomes polynomial; note that this result does not provide necessary conditions for the USLP to become polynomial (it provides only sufficient conditions for that). Indeed, in this section we mention a different restriction of the USLP which guarantees polynomial time complexity.

The general framework imposes no structure on the set of states, and places no restrictions on the number of laws which affect the results of any particular action. In practice, there is much more structure in the makeup of agents. For example, while the set of states might be very large, it is usually possible to identify components of the agent, such that the set of states of the agent consists of the Cartesian product of the sets of states of the individual components. If we consider for example the representation of a robot, then one component may relate to its location, another to its orientation, and another to its arm position.

This modularity of states is the first restriction we consider here; specifically, we assume that there exist $O(\log(n))$ components, each of which can be in one of a constant number of states. Thus the state of an agent consists of the Cartesian product of these states. The total number of an agent's states is still n; note that this is no contradiction.

The modularity of state gives rise to a modularity of action; usually, only a small number of social laws will be relevant to determining the change in the state of a particular component as a result of a particular action. In the robotic setting, for example,

[10] Notice that the $\log(\log(n))$ factor can be treated as a small constant for any social agent model of a manageable size, and thus, practically speaking, the restriction is to plans of a length that is logarithmic in the number of states. However, the theorem below requires the precise term given here.

a change in the particular location of the robot might depend on particular traffic laws, but not on laws requiring one to shut windows after opening them, on laws prohibiting one from taking the last cookie without announcing the fact, and perhaps not even on many of the possible traffic laws. We capture this intuition by requiring that the change of a particular state of a particular component can depend on only a constant number of social laws.

We will use the term *modular social agent* to denote an social agent whose structure has these two properties of modularity, and *modular social multi-agent system* to denote a collection of such agents. Notice that these restrictions are independent from those discussed in the previous section. Modular systems have the following property:

Theorem 3.11. *Given a modular social multi-agent system, the USLP is polynomial.*

4. Conclusions

A basic approach to coordinating multiple agents is to restrict their activities in a way which enables them to achieve their dynamically-acquired goals while not interfering with other agents. Although this is, we believe, a commonsensical approach, it has not received much attention so far within AI; exceptions include [10, 11], where the authors investigate several basic principles and formal aspects of this approach.

The contribution of the case study presented in the first part of this paper is twofold: We have offered a new approach to designing multi-robot systems, and have demonstrated the general approach of artificial social system in a concrete domain. In the second part of this paper we treated the more general computational theory of social laws. We have presented model of multi-agent action whose novel feature is the explicit mention of social laws in the definition of agents. We then defined and investigated the fundamental computational problem involved with finding useful social laws. We showed that the general problem is intractable, and then identified restrictions which achieve tractability. We believe that the formal model, computational problem definition, and complexity results constitute an essential contribution to our understanding of the role of social law in artificial intelligence. Some work had been concerned with issues related to topics discussed in this paper. This includes work in the areas of organization theory (see [8]), team theory [9], and DAI (see for example [4]). The related work in these areas of research is especially concerned with the design of agents' roles and communication structures which enable a cooperative achievement of a common goal. The work on artificial social systems concentrates on a somewhat complementary issue: the off-line design and computation of laws which enable each agent to work individually and successfully towards its own goals during the on-line activity given that all the agents obey these laws. Additional related work includes the synthesis of multi-agent programs [12] and work on cooperative discrete event systems (DES) [13]. A main difference between these lines of research and the work in the framework of artificial social systems stems from the fact that the latter gives more structure for the design of multi-agent systems (first find a social law and then enable agents to tend to work individually) while concentrating on the fundamental problems of social design (we have to restrict the behavior of agents but not restrict them too much). A crucial difference between our setting and that of DES is that in our case the effects of actions taken locally by an agent depend on actions taken by other agents; the result of taking action A by one agent depends on whether another agent took action B. In DES the supervisor restricts the actions available to machines so that the sequence of actions in the systems ends up belonging to a given language; however the effect of each individual action is unaffected by the supervisor. This is also true of so-called "product automata"; there too the job of the coordinator is to ensure global properties such as mutual exclusion by restricting actions of agents, but the effect of any allowed action is not altered. [11] It can be shown that various extended notions, such as "shared events" in DESs still operate within the basic DES framework, and thus address quite separate issues that in our work.

Much remains to be done. For example, as was mentioned in Section 3.1, we have assumed a homogeneous society, both in term of the makeup of agents and in terms of the applicability of social laws. In future work we will relax this assumption, made here only for convenience. Harder assumptions to relax include the assumption of law-abiding citizenship; here we have assumed that all agents obey the social laws, but what happens if some don't? How vulnerable is the society to rogue agents? Similarly, we have assumed that the useful social law is computed off-line. Sometimes this is infeasible, however, either because the makeup of the agents is not known in advance, or because it changes over time. In such cases, can we devise methods by which, through trial and error, the agents will over time converge on a social law? These are some of the questions in which we are currently interested; experimental results on these matters appear in [16].

Appendix A. Proof sketches

Proof of Theorem 3.7 (*Sketch*). It is easy to check that the problem is in NP. We prove NP-hardness by a reduction from 3-SAT. Clause i in the 3-SAT instance will correspond to transitions which change from state i to state $i+1$. For each literal appearing in the 3-SAT instance there will be a corresponding action. The transition which corresponds to performing l (i.e., the action which corresponds to the literal l) in state i (l should appear as a literal in clause i) will change from state i to state $i+1$ if and only if the social law which says that $\neg l$ is disallowed will hold. For example, if clause i is $x \wedge y \wedge \neg z$, then we will have actions which correspond to x, y, $\neg z$ that will lead from state i to state $i+1$ given that the actions which correspond to $\neg x$, $\neg y$, z will be disallowed respectively. We assume the existence of an auxiliary action that leads with no conditions from any state to state 1. We assume that the focal states include state 1 and the "final" state (whose number is determined according to the number of clauses). Our result can be easily verified now. □

[11] To quote from Ramadge and Wonham's survey article [13], "... we assumed that the components of the open loop product system were independent, and that any interaction between the components could be modeled as part of a control constraint".

Proof of Theorem 3.9 (*Sketch*). The proof follows from the fact that the reduction presented in the proof outline of Theorem 3.7 is useful for the BUSLP case as well (in fact, we get an instance of the BUSLP). □

Proof of Theorem 3.10 (*Sketch*). The polynomial result is achieved by applying a DFS (depth-first search algorithm) starting from each possible focal state where the actions which usually appear in classical DFS are replaced by transitions containing also social laws, and where the depth of the search is bounded by $O(\log(n)/\log(\log(n)))$. An additional difference is that along each such DFS we have to check whether we use an action which is already disallowed (by a previous transition in the appropriate path). The above algorithm is polynomial and finds all the possible paths of the appropriate length (where each path includes both actions and social laws) between a given pair of focal states. The number of such possible paths is polynomial. Because of the fact that there is only a constant number of focal states, we remain only with a polynomial number of alternatives that we have to check (there is a polynomial number of relevant paths for each pair of focal states, and there is a constant number of such pairs).

The NP-complete results are achieved again by a reduction from 3-SAT. The idea is to associate with each literal of each clause of the 3-SAT instance a distinguished action, and to build an agent that needs to be able to perform for each clause at least one action from among the actions that correspond to that clause. However, in order that an action which corresponds to a literal l will have desired effects, all the actions that correspond to $\neg l$ should be disallowed. Let us illustrate this idea in the case where we allow any number of focal states (where restriction (1) is dropped). In that case the structure of our agent (the reduction's outcome) will be of an "almost" acyclic graph (where the states are the nodes and the transitions are the edges of that graph). In fact, it will be "almost" a tree where we also allow several edges (with the same direction) between adjacent nodes. The depth of that tree will be $O(\log(n)/\log(\log(n)))$ where the degree of each internal node is $O(\log(n))$ (where n is the number of clauses in the 3-SAT instance). We associate the set of edges connecting a particular pair of adjacent nodes with a particular clause of the 3-SAT (there will be a 1-1 correspondence between clauses and pairs of adjacent nodes in that graph). We will associate several actions with the pair of states which corresponds to clause i: a unique action for each literal appearing in that clause. However, an agent which performs in a given state a particular action that corresponds to the literal l will reach the adjacent state if and only if the actions which correspond to $\neg l$ will be disallowed. We also assume that there is an auxiliary action that leads with no condition from any leaf state to the root state (and therefore the graph is only "almost" acyclic). If we assume that the focal states are the root state and the leaves states, we get the appropriate result.

A similar idea is used for proving the case where we drop the second restriction. In that case the reduction will be quite similar to the one mentioned before, but we will change a little bit the structure of the above tree. We will assume that the only focal state is the root state (and therefore there is only a constant number of focal states now), and will force the need to pass from each vertex in the previous graph to its son in that graph using the nondeterminism (in the previous case this was forced by the fact that we had to reach any focal state and therefore any leaf). In order to do so, we will replace the direct connection between each given non-leaf vertex x of the previous graph and its sons by a connection from x to $O(\log(n))$ new vertices each of which will be connected to a different and unique vertex which previously was a son of x. The connection from a new vertex n_y to the vertex y which was previously a son of x will be by edges mentioning the same actions that were mentioned by the connection from x to y. However, in x itself the agent will be able to perform now only one action: a new distinguished action which leads nondeterministically to all the previously mentioned new vertices. It is easy to verify that this construction will force us to select for each clause a particular appropriate action (in order to reach the focal state given all the possible nondeterministic effects). Given that an action which corresponds to a literal l will have the desired effects only if the actions which correspond to $\neg l$ are disallowed, we get the desired reduction.

The case where we drop restriction (3) follows immediately from the construction used in the proof of Theorem 3.7. □

Proof of Theorem 3.11 (*Sketch*). The proof of this theorem follows immediately from the fact that on that natural case the number of possible alternatives for useful social laws which we have to consider is polynomial. More precisely, there is a logarithmic number of components each of which might relate to a constant number of focal states, and therefore the total number of potential social laws is polynomial and can be enumerated in a polynomial time. For each such law we can verify in polynomial time whether each particular focal state is reachable from all the other focal states. This is based on the facts that the number of focal states as well as the problem of verifying reachability between a pair of focal states are polynomial. As a result, we get a full polynomial procedure for determining a useful social law. □

References

[1] S.J. Buckley, Fast motion planning for multiple moving robots, in: *Proceedings 1989 IEEE International Conference on Robotics and Automation*, Scottsdale, AZ (1989) 322–326.

[2] P. Caloud, W. Choi, J.C. Latombe, C.L. Pape and M. Yim, Indoor automation with many mobile robots, in: *Proceedings IEEE International Workshop on Intelligent Robots and Systems*, Tsuchiura, Japan (1990).

[3] R. Davis and R.G. Smith, Negotiation as a metaphor for distributed problem solving, *Artif. Intell.* **20** (1) (1983) 63–109.

[4] E.H. Durfee, V.R. Lesser and D.D. Corkill, Coherent cooperation among communicating problem solvers, *IEEE Trans. Comput.* **36** (1987) 1275–1291.

[5] M. Erdmann and T. Lozano-Perez, On multiple moving robots, *Algorithmica* **2** (4) (1987) 477–521.

[6] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, *Int. J. Rob. Res.* **5** (1) (1986).

[7] S. Kraus and J. Wilkenfeld, The function of time in cooperative negotiations, in: *Proceedings AAAI-91*, Anaheim, CA (1991) 179–184.

[8] T.W. Malone, Modeling coordination in organizations and markets, in: A.H. Bond and L. Gasser, eds., *Readings in Distributed Artificial Intelligence* (Morgan Kaufmann, San Mateo, CA, 1988).

[9] J. Marschak and R. Radner, *Economic Theory of Teams* (Yale University Press, New Haven, CT, 1972).

[10] Y. Moses and M. Tennenholtz, Artificial social systems, Part I: basic principles, Tech. Rept. CS90-12, Weizmann Institute, Rehovot, Israel (1990).

[11] Y. Moses and M. Tennenholtz, On computational aspects of artificial social systems, in: *Proceedings DAI-92* (1992).
[12] A. Pnueli and R. Rosner, Distributed reactive systems are hard to synthesize, in: *Proceedings 31th IEEE Symposium on Foundations of Computer Science* (1990).
[13] P. Ramadge and W. Wonham, The control of discrete event systems, *Proc. IEEE* 77 (1) (1989) 81-98.
[14] J.S. Rosenschein and M.R. Genesereth, Deals among rational agents, in: *Proceedings IJCAI-85*, Los Angeles, CA (1985) 91-99.
[15] Y. Shoham, Agent oriented programming, Tech. Rept. STAN-CS-1335-90, Department of Computer Science, Stanford University, Stanford, CA (1990).
[16] Y. Shoham and M. Tennenholtz, Emergent conventions in multi-agent systems: initial experimental results and observations, in: *Proceedings KR-92*, Cambridge, MA (1992).