Preferences in Game Logics

Sieuwert van Otterloo
Department of Computer Science
University of Liverpool
Liverpool L69 7ZF, United Kingdom
sieuwert@csc.liv.ac.uk

Wiebe van der Hoek
wiebe@csc.liv.ac.uk

Michael Wooldridge
mjw@csc.liv.ac.uk

Abstract

We introduce a Game Logic with Preferences *(*GLP*), which makes it possible to reason about how information or assumptions about the preferences of other players can be used by agents in order to realize their own preferences.* GLP *can be applied to the analysis of social protocols such as voting or fair division problems; we illustrate this use of* GLP *with a number of worked examples. We then prove that the model checking problem for* GLP *is tractable, and describe an implemented model checker for the logic – by using the model checker, it is possible to automate the analysis and verification of social protocols.*

1. Introduction

We are interested in the formal specification, analysis, and verification of mechanisms for social interaction [15]. Examples of such mechanisms include voting and election procedures [5], auction and trading protocols, and solutions for fair division and dispute resolution [3]. As a simple example, consider the problem of constructing a voting procedure for Alice, Bob, and Caroline to decide between three holiday destinations: Xanadu, Yemen, and Zambia. An example requirement for this social choice mechanism might be that if two people prefer the same choice, then that choice should become the outcome.

One can model many mechanisms for social interaction as extensive games [14, 2], and then use game theoretic concepts to characterise properties of the protocol. Unfortunately, a game is not just a model for a protocol, it is a model for the protocol *and the preferences of the agents*. One would like to model procedures as complete and perfect information games, because this is a simple class of games; but then all agents have complete knowledge of each other's preferences. This is an unrealistic assumption. In fact, for the holiday example above, each agent can order the three countries in six possible ways, and therefore one would need to consider at least 216 different games, even excluding the possibility that an agent might value options equally. Models of social procedures that do not require one to completely specify the preferences of agents are thus highly desirable. Gal and Pfeffer [9, 10] have also recognized this problem, and use Networks of Influence Diagrams to model game situations in which agent may have different beliefs about which game they are playing. We take a different approach by starting with one common game form and use logical assumptions to represent different preference structures.

One can also model social procedures as distributed or interpreted systems [8]. These systems are models for the options and knowledge that agents have: important properties such as 'Alice, Bob and Caroline will eventually agree on an outcome' can be verified using temporal logics [4]; temporal epistemic logics can be used for knowledge based requirements [8], and coalition (epistemic) logics [17] can be used for properties such as 'Alice knows that on her own she has no strategy to block Zambia as an outcome'. Unfortunately, these approaches do not deal with preferences at all. The assumption in these systems is that agents know nothing about other agents' preferences – and of course this is also an unrealistic assumption. In short, this paper presents a logic that is intended to overcome these limitations. This logic allows us to reason about interactions between agents with only partial knowledge about their preferences. Specifically, we are interested in the question how the addition of information about other agents' preferences influences the strategies of agents and coalitions of agents.

The key assumption we make is that agents reason about the preferences of other agents, and then make certain strategic decisions. We do not specify how they know it (it might be by means of communication, espionage, or by studying previous histories of the game); the information about the preferences might even be hypothetically assumed. Reasoning about the preferences of (other) agents is interpreted as reasoning about their strategies, because for a rational agent, having a preference implies choosing a strategy that leads to this preference. The knowl-

edge that agents have, based upon their knowledge of other agents' preferences, is called *strategic knowledge* [7]. Although our aim is to study strategic knowledge in this paper, we do this *implicitly*, since we do not introduce strategies as first class citizens of our logic, but rather refer only to preferences. We develop a logic GLP (*Game Logic with Preferences*), which allows us to reason about the consequences of the announcement of information about preferences. Initially, agents know the protocol but do not know other agents' preferences. The logic enables us to make assumptions about their preferences, and subsequently to express the consequences of such assumptions. Note that some simplifying assumptions are made. We assume agents have perfect information about the current game state, so we do not consider imperfect information games. We also assume that all agents have the same information about preferences.

The remainder of the paper is structured as follows. Section 2 defines some game theoretic notions that are used throughout the paper. The logic GLP for expressing properties of protocols is defined in Section 3, together with its semantics. The use of GLP is illustrated through some example protocols in Section 4. In Section 5, an efficient algorithm for evaluating GLP properties is given, and an implementation of this algorithm in a Java model checker for GLP is described. We present some conclusions in Section 6.

2. Extensive Games and Game Forms

Extensive games are defined in several different but essentially equivalent ways in the literature: for example as a tree [13], or as a set of runs [14]. We use a definition as a tree, but stay close to the notation of [14], and we will simply refer to 'game' when we mean 'extensive game'.

Informally, a game consists of two components. The *game form* F specifies the available actions for each agent, whereas the *preference relation* $\succeq$ specifies an agent's preferences. For our formal approach we also need an *interpretation*, π, which allows us to use propositions for describing end states. We do not always need all these components at the same time, and therefore we distinguish the following structures.

F	A game form
$(F, \succeq)$	A game
$(F, (P, \pi))$	A game form interpretation
$(F, \succeq, (P, \pi))$	A game interpretation

Notation Throughout this paper we follow coalition logic conventions [17] by using Σ to denote a set of agents, Γ for a coalition of agents (so $\Gamma \subseteq \Sigma$), $i \in \Sigma$ for an arbitrary agent, P for a set of propositions and π as an interpretation function for propositions.

Trees The symbol N is used for a finite set of nodes, and $T \subseteq (N \times N)$ is a binary relation between them, yielding a tree $t = \langle N, T, n_0 \rangle$, with root $n_0 \in N$. Given a tree t, we define the successor function $Succ : N \to 2^N$, yielding for every $n \in N$ its set of successors $\{n' \in N \mid (n, n') \in T\}$. The function $Z(T) = \{n \in N | Succ(n) = \emptyset\}$ defines the set of all terminal nodes of the tree: we refer to them as *states*.

Game Form A *game form* is a tuple $F = \langle \Sigma, Ag, N, T, n_0 \rangle$, where Σ is a set of agents or players and $\langle N, T, n_0 \rangle$ is a tree. The function $Ag : N \setminus Z(T) \to \Sigma$ assigns an agent to each non-terminal node: this agent is thought of as the agent that decides the next action, or, equivalently, what the next node will be. Conversely, the function Ow returns, for every agent i, the set of nodes "owned" by that agent.

Game A *game* is a tuple $(F, \succeq)$ where F is a game form and $\succeq$: $\Sigma \to 2^{Z(T) \times Z(T)}$ is a function that assigns to each agent i a preference relation $\succeq_i$ over all terminal nodes. Preference relations are assumed to be transitive and symmetric. Informally, $n \succeq_i n'$ means that agent i thinks that n is at least as good as n'. We write $n \succ_i n'$ iff $n \succeq_i n'$ but not $n' \succeq_i n$.

Game Form Interpretation A *game form interpretation* is a tuple (F, P, π) where F is a game form $\langle \Sigma, Ag, N, T, n_0 \rangle$, P is a finite set of propositions and $\pi : Z(T) \to 2^P$ assigns to each non-terminal node the set of propositions that are considered true in that node.

Game Interpretation A *game interpretation* is a tuple $(F, \succeq, P, \pi)$ such that (F, P, π) is a game form interpretation and $(F, \succeq)$ is a game.

Preference We say that agent i has a *preference* for a set of outcomes $S \subseteq Z(T)$ in game $(F, \succeq)$ if for all $s \in S$ and $t \in Z(T) \setminus S$ we have that $s \succ_i t$. In this definition, a set S is thus preferred over its complement. Note that this is a strong notion of preferences. We can only say that a person prefers to have coffee if the worst situation in which it has coffee is better than the best situation in which it does not have coffee. This strong definition is convenient in a logical approach, because it allows one to draw definite conclusions. A notion of preference in which the average value of $s \in S$ is higher than the average value of an element $t \notin S$ might be applicable within a probabilistic framework. A coalition of agents Γ prefers a set S if each member $i \in \Gamma$ prefers S.

Strategies A *pure strategy* for agent i in a game form F is a function $PS_i : Ow(i) \to N$ such that $PS_i(n) \in Succ(n)$. This function prescribes a decision for agent i to each node owned by i. A *mixed strategy* is a function $MS_i : Ow(i) \to 2^N \setminus \emptyset$ such that $PS_i(n) \subseteq Succ(n)$. A mixed strategy prescribes a subset of alternatives to each node owned by i. A *strategy for a coalition* Γ, either mixed or pure, is a set of strategies, one for each agent in Γ. If S_Γ is a strategy for Γ, and $n \in N$ with $Ag(n) \in \Gamma$, then $S_\Gamma(n) = S_{Ag(n)}(n)$. We use both the notation $n' \in S_\Gamma(n)$ and $S_\Gamma(n) = n'$ when not confusing. For the grand coalition Σ, S_Σ is called a *strategy*

profile. Such a profile s represents a *Nash Equilibrium* if no agent in Σ can unilaterally deviate from s and improve his outcome. It is well-known that every two person zero-sum game has such an equilibrium, which can be obtained using backward induction [14].

Strategy Updates A game is often thought of as a set of possible computations or runs. Each path on the game tree then corresponds to a run. Formally, a *run* of the game form $\langle \Sigma, Ag, N, T, n_0 \rangle$ is a sequence $n_0, n_1, \ldots, n_z$ where n_0 is the root of T and $n_z \in Z(T)$ is a terminal node, and for each k it is the case that $(n_k, n_{k+1}) \in T$. If an agent or a coalition of agents is committed to a certain strategy S, certain runs can be excluded from consideration because these runs are not compatible with that strategy. A run $n_0, n_1, \ldots, n_z$ is compatible with a mixed strategy S_Γ if for any pair (n_k, n_{k+1}) with $Ag(n_k) \in \Gamma$ it is the case that $n_{k+1} \in S_\Gamma(n_k)$. This means that every choice made by an agent from coalition Γ must be made according to S_Γ.

We define an *update function* u for any game form $F = \langle \Sigma, Ag, N, T, n_0 \rangle$ and strategy S_Γ, such that $u(F, S_\Gamma) = \langle \Sigma, Ag', N', T', n_0 \rangle$ is the game consisting of all runs compatible with S_Γ. To achieve this, we stipulate that Ag' is the restriction of Ag to $N' \subseteq N$ which is generated from N by the new T' relation, which is defined as

$$T' = \{(n, n') \in T \mid Ag(n) \in \Gamma \;\Rightarrow\; n' \in S_\Gamma(n)\}$$

Note that in this updated model, agents not in Γ still have all of their options open. Only agents that are part of Γ are now limited to making choices that are compatible with S_Γ.

3. A Logic for Preferences

In this section we define the syntax and semantics of the logic GLP.

3.1. Syntax of GLP

We find it convenient to define the syntax of GLP in several stages, beginning with (classical) propositional logic.

Definition 1 *The language PL of propositional logic over a set of propositions P is the smallest set $L \supseteq P$ such that for any $\varphi \in L$ and $\psi \in L$ we have that $\varphi \vee \varphi \in L$ and $\neg\varphi \in L$.*

We use propositional logic to express properties of the outcomes or results of games. An example inspired by football is the statement $\neg win_A \wedge \neg win_B$, which expresses that neither team A wins nor team B wins. It is important to realize that propositional logic is *only* used for properties of terminal states, not for intermediate states of a game or protocol.

The logic GLP contains all connectives of propositional logic and two additional operators. One can be used to introduce formulas of propositional logic in GLP, and the other one is used to express consequences of preferences.

Definition 2 *Let P be a set of propositions, and Σ a group of agents. Let PL be the language of propositional logic over P. The language for* GLP *is the smallest language L such that for any formula $\varphi_0 \in PL$ and $\varphi, \psi \in L$, it is the case that:*

$$\begin{aligned} \Box\varphi_0 &\in L \\ \varphi \vee \psi &\in L \\ \neg\varphi &\in L \\ [\Gamma : \varphi_0]\psi &\in L \end{aligned}$$

The formula $[\Gamma : \varphi_0]\psi$ has the intended reading 'In any game in which coalition Γ prefers φ_0, ψ will hold'. The box operator, $\Box\varphi_0$, takes a propositional logic formula φ_0, which can be interpreted in a specific state, and converts it into a GLP formula: it means that φ_0 holds for every possible outcome. A useful shorthand is the sometimes operator $\Diamond\varphi_0$, which means that φ_0 holds for *some* outcome. It is defined as $\Diamond\varphi_0 = \neg\Box\neg\varphi_0$. Where no confusion is possible, we omit brackets and commas in the notation of a set Γ of agents, for example writing $[ABC : \varphi_0]\psi$ instead of $[\{A, B, C\} : \varphi_0]\psi$.

Sample formulas of this logic, with their reading in the holiday example from Section 1 are given in the next table. The outcomes Xanadu, Yemen or Zambia are indicated by propositions x, y and z, while Alice, Bob and Caroline are called A, B and C, respectively.

$\Diamond x$ Xanadu is a possible outcome.

$[AB : y]\Box y$ If Alice and Bob prefer Yemen, then Yemen is selected.

$[A : x \vee y][B : x]\Box\neg z$ If Alice prefers Xanadu or Yemen, and Bob prefers Yemen, then they will not go to Zambia.

3.2. A Game-theoretic Semantics

We now give the formal semantics of GLP. We begin by showing how propositional logic can be used for describing the outcomes of game form interpretations. We then show how GLP can be evaluated over game form interpretations, by using an update function for preferences, and to complete our definition we define the update function. The first and easiest step is defining the interpretation of propositional logic over end-states.

$$\begin{array}{lll} \pi, s \models p & \text{iff} & p \in \pi(s) \\ \pi, s \models \varphi \vee \psi & \text{iff} & \pi, s \models \varphi \text{ or } \pi, s \models \psi \\ \pi, s \models \neg\varphi & \text{iff} & \text{not } \pi, s \models \varphi \end{array}$$

In the interpretation of GLP, the logical connectives are defined similarly. The operator $\Box\varphi_0$ is defined as 'truth in all end-states', and the assumption operator $[\Gamma : \varphi_0]\psi$ is defined on game form interpretations M as follows. We then calculate a restricted game form interpretation $M' = Up(M, \Gamma, \varphi_0)$, in which Γ acts as to guarantee φ_0, and we evaluate whether M' fulfills ψ. Let $M = \langle \Sigma, Ag, N, T, n_0, P, \pi \rangle$ be a game form interpretation and φ_0 and φ as before.

$$
\begin{array}{lll}
M \models \Box\varphi_0 & \text{iff} & \text{for all } s \in Z(T)\ \pi, s \models \varphi_0 \\
M \models \varphi \vee \psi & \text{iff} & M \models \varphi \text{ or } M \models \psi \\
M \models \neg\varphi & \text{iff} & \text{not } M \models \varphi \\
M \models [\Gamma : \varphi_0]\psi & \text{iff} & Up(M, \Gamma, \varphi_0) \models \psi
\end{array}
$$

In Section 2, we defined what it means for a coalition to prefer a set of outcomes S. We now extend this definition to propositional logic formulas. We say that Γ has a preference for φ in game interpretation $(F, \succeq, P, \pi)$ if Γ prefers the set $S = \{s | \pi, s \models \varphi\}$. Thus, a propositional formula is preferred if the set of end nodes in which it is true is preferred. The update function $Up(M, \Gamma, \varphi_0)$ returns a model in which the actions of agents in Γ are restricted to actions that help them achieve a preferred outcome. We use the notion of a subgame perfect strategy [16]. The behaviour of agents should be rational in any subgame-form F' of the complete game form F. Moreover, the agents should not assume any helpful behaviour from agents outside Γ. Finally, the model $Up(M, \Gamma, \varphi_0)$ should not be unnecessarily restricted: any actions that are rational should be available to agents in Γ. Below we have formalized when a strategy guarantees something, when it is subgame perfect, and when a strategy is more general than another strategy. These notions are use to define the update function.

A mixed strategy MS_Γ for coalition Γ contains a pure strategy PS_Γ of coalition Γ iff for all nodes n with $Ag(n) \in \Gamma$ we have that $PS_\Gamma(n) \in MS(n)$. A mixed strategy is completely described by the pure strategies it contains. We can think of a mixed strategy as giving some information about which pure strategy is used. A mixed strategy seen in this way is a description of the information other agents may have about the strategy used [1]. One can compare mixed strategies using an *inclusion* operator. For two mixed strategies S_1 and S_2, both for coalition Γ, we say that S_1 is contained in S_2, or that S_2 is more general than S_1 and write $S_1 \subseteq S_2$, if and only if for all nodes n with $Ag(n) \in \Gamma$ we have that $S_1(n) \subseteq S_2(n)$. This provides a partial ordering on all mixed strategies.

Definition 3 *A strategy S_Γ guarantees φ_0 in game form F iff every outcome s of game form $u(F, S_\Gamma)$ fulfills $\pi, s \models \varphi_0$. We say that Γ can guarantee φ_0 in F if there exists a strategy S_Γ such that S_Γ guarantees φ_0. We say that S_Γ is* subgame perfect *for φ_0 in F if for every subgame F' of F such that Γ can guarantee φ_0 in F' it is the case that S_Γ guarantees φ_0 in F'. The strategy $S^* = S^*(F, \Gamma, \varphi_0)$ is the most general sub game perfect strategy for Γ and φ_0 in F. The update function Up is defined as*

$$Up(M, \Gamma, \varphi_0) = \langle u(F, S^*(F, \Gamma, \varphi_0)), P, \pi' \rangle$$

where π' is the restriction of π to the new set of nodes.

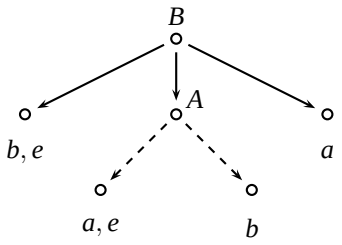

Figure 1. Alice and Bob eat cake

4. Examples

This section provides some examples that illustrate how GLP can be used in the specification and analysis of social interaction mechanisms.

Alice and Bob eat cake

Alice and Bob have a cake, and they have agreed to divide it by means of a "cut-and-choose" protocol [3]. Alice has cut the cake and unfortunately one of the pieces is bigger than the other. Bob can now choose from three options: he can select the big piece, select the small piece, or he can say to Alice 'No, you choose'. If he lets Alice choose, she can either choose the big piece or the small piece. Both agents have common knowledge of this protocol, and they know that they neither know the other agent's preferences (see Figure 1): proposition *a* means that Alice gets the biggest piece, *b* that Bob gets the biggest piece, and *e* means that something embarrassing has happened, namely that either Alice or Bob has chosen the biggest piece. In many cultures this is not polite and in several of the example formulas we assume that everyone is embarrassed if this happens. Using GLP one can express relevant properties of this protocol. First we will provide several GLP formulas (*A* stands for Alice, *B* stands for Bob).

- $[B : \neg e]\Box a$ If *B* does not want either of them being impolite, he must take the smallest piece. Our semantics takes a pessimistic view, so Bob cannot take the risk of letting *A* choose. Figure 2 shows the model $Up(M, \{B\}, \neg e)$.
- $[B : \neg e][A : \neg e]\Box a$ This formula is a consequence of the

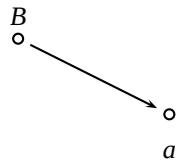

Figure 2. Model $Up(M, \{B\}, \neg e)$

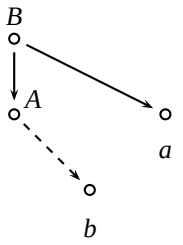

Figure 3. Model $Up(M, \{A, B\}, \neg e)$

previous example. It expresses that if B does not want embarrassment and that A does not want embarrassment, then A gets the biggest piece. This may seem strange, since there is an outcome in which $\neg e$ and b are true. However the order of assumptions is important. The formula expresses that B wishes to guarantee the absence of embarrassment, independently of what A does. Two possible readings of the formula are that he commits himself to his strategy before he learns that A has the same preference, or that he thinks that this goal is so important that he does not wish to rely on A for this property.

• $[AB : \neg e][B : b]\Box b$ In this example, A and B commonly want to avoid embarrassment, and B also prefers b. If this is the case, B can let A choose and then A will take the smallest piece. Figure 3 shows the updated model $Up(M, \{A, B\}, \neg e)$. Note that, strictly speaking, B cannot prefer $\neg e$ and b simultaneously, since that would mean that the situation with $\neg e \wedge \neg b$ is preferred over the situation with $e \wedge b$ and vice versa. However the preference for b appears in the scope of a preference for $\neg e$, so one could say that the agent prefers first $\neg e \wedge b$, then $\neg e$ and then $\neg e \vee b$. This does correspond to a single preference relation.

• $[A : \neg e][B : \neg e][B : b]\Box b$ This formula expresses that if A does not want embarrassment, B does not want embarrassment, and B prefers the biggest piece then B gets the biggest piece. The behaviour of B is influenced by the fact that he knows that A prefers to avoid embarrassment. In this scenario A should try to hide the fact that she has good manners, because it is not in her advantage if B knows this.

These examples illustrate that, by using GLP, one can express consequences of ordering goals in a certain way. Similar to the concept of common knowledge in epistemic logic [8], we have the idea of a common goal, which is different from both having a goal.

Voting Protocol

In the introduction, we already mentioned the problem of defining a fair voting protocol. The problem is to design a voting protocol for three agents (A, B, and C) over three alternatives (x, y, and z). We use this problem to show how GLP can be used for expressing the requirements of such protocols. Next, we present two example solutions and show how GLP can express the differences between them.

We require a voting protocol P in which exactly one option is chosen and any group of two agents can decide on any outcome. This is formalized in the next formulas. Let $\Gamma \subseteq \{A, B, C\}$ and u, v variables over the options $O = \{x, y, z\}$.

$\Box(x \vee y \vee z)$	at least one alternative
$\bigwedge_{u,v \in O, u \neq v} \Box\neg(u \wedge v)$	at most one alternative
$\bigwedge_{u \in O}[\Gamma : u]\Box u$	for all $\lvert\Gamma\rvert > 1$. Majority decides

In Figure 4, a protocol P_1 is depicted which satisfies these requirements – it is in fact a smallest protocol that satisfies the requirements, as one can verify by testing all smaller trees. It works in two steps. First, A can say whether B or C can make a choice. Then, either B or C can indicate which of the three destinations is chosen. The protocol may seem strange because A cannot directly support a certain outcome. What is the best action for A depends on what B and C will do. In this protocol it is thus important for A to know what the others do, while C and B need no information about the other agents. The next GLP formulas illustrate this peculiarity of protocol P_1.

$$P \models [AB : x]\Box x$$
$$P \models [B : x][A : x]\Box x$$
$$P \not\models [A : x][B : x]\Box x$$

There is of course more than one protocol that meets the given requirements: Figure 5 shows another solution. In this protocol, A can vote for any alternative and then B votes for one of the alternatives. If they vote for the same option, then this option is chosen. If they disagree, agent C can choose between the two options that A and B voted for. So agent C has only two options. Nine extra propositions are introduced, to indicate which agent has cast which vote: proposition a_x means that agent a has chosen option x, and simi-

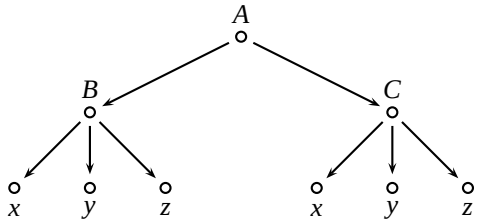

Figure 4. A voting protocol P_1

larly for $a_y, a_z, b_x, b_y, b_z, c_x, c_y, c_z$. In the figure only the left branch is fully shown.

This protocol also satisfies the requirements. A nice property of this protocol is that A can directly indicate a preference. The extra propositions $(a_x, b_x, \ldots)$ show how one can use propositions in end states to indicate that a certain action was chosen.

This demonstrates that in finite trees, one can use propositions in end states for events that one would describe using intermediate propositions or propositions on transitions in other models.

The extra propositions allow one to formulate questions about strategic voting. An agent votes strategically if it does not vote for the options it prefers most [11]. In an ideal voting protocol, agents do not gain anything by strategic voting. Unfortunately the Gibbard-Satterthwaite Theorem [11] states that in any voting scheme not completely determined by a single agent it is possible to benefit from strategic voting. With GLP, one can determine for any protocol in which circumstances strategic voting might occur.

First, we list some related properties about the options. We see that agent A is completely free to vote for x if she wants that, while C is not. On the other hand, C's vote is always decisive, which is not true for A's vote.

$$P_2 \models [A : a_x]\Box a_x$$
$$P_2 \not\models [C : c_x]\Box c_x$$
$$P_2 \models \Box((c_x \to x) \wedge (c_y \to y) \wedge (c_z \to z))$$
$$P_2 \not\models \Box((a_x \to x) \wedge (a_y \to y) \wedge (a_z \to z))$$

An interesting result is that C, under our rationality condition, should not vote for an option it does not prefer. Unfortunately, this result cannot be shown for A. If A prefers x, but A knows that B and C prefer y, then it is equally rational for A not to vote for x, since A knows it will not be successful.

$$P_2 \models [C : \neg x]\Box\neg c_x$$
$$P \models [B : x][A : x]\Box x$$
$$P \not\models [A : x][B : x]\Box x$$

Suppose we know what the preferences of our agents are: can one then use GLP to derive which outcome will

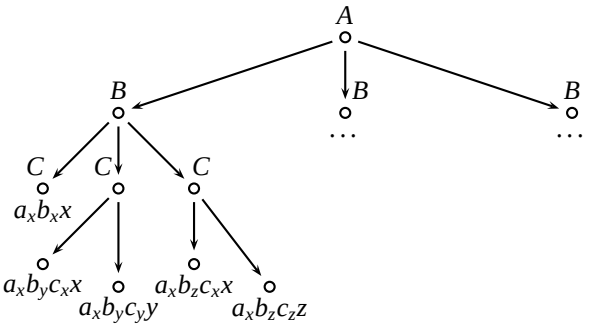

Figure 5. A second voting protocol P_2

be reached? In the next table the preference relation of each agent over the outcomes is given.

$$x \succ_A y \succ_A z$$
$$z \succ_B x \succ_B y$$
$$y \succ_C x \succ_C z$$

All formulas that are preferred by some coalition are listed in the next table.

Coalition	Preferred formulas
A	$x, x \vee y$
B	$z, x \vee z$
C	$y, x \vee y$
AC	$x \vee y$

One can see that there are not many coalition preferences for this preference relation. The reason is that our definition of a preference is quite strong. One can show that Yemen is a likely outcome of the holiday election according to protocol P_1 by checking the following:

$$P_1 \models [A : x][C : y][AC : x \vee y]\Box y$$

This formula expresses that given the preferences of A and C and the protocol, A will say that C may decide and C will choose for option y. This outcome is a sub-game perfect Nash equilibrium of the game $(P, \succeq)$. For the other protocol, the same outcome can be rationalized by:

$$P_2 \models [A : x][C : y][AC : x \vee y]\Box y$$

However, other outcomes can be rationalized by making different assumptions.

$$P_2 \models [B : x \vee z][C : x \vee y][A : x]\Box x$$

In this case, A votes for x. From the assumption that B prefers x or z, everyone can conclude that B will not vote y. If B votes x, x is the outcome. Otherwise c will chooses x over z, and x is also the outcome.

¿From these examples, it should be clear that it is not only important who has which preferences, but also which preferences one considers. If different preferences are announced, signaled, or otherwise communicated in some way, different outcomes can arise.

5. Model Checking GLP

Model checking is the problem of deciding whether a given formula is true on a given model [4]. A model checking algorithm for GLP can be used for the automatic verification of multi agent interaction protocols, such as verifying that the protocols from Section 4 satisfy the stated requirements. In this section, we prove that the model checking problem for GLP is tractable. Our proof is constructive: Figures 6 and 7 give a polynomial time model checking algorithm for GLP. Formally, we assume that the size of Σ is bounded by some constant. The size of a model ($|M|$) is defined as the number of nodes it contains, while the size of a formula ($|\varphi|$) is the number of connectives, operators, and propositions. Let $c(X)$ denote the number of operations needed to compute the value of expression X. The notation $c(X) = \mathcal{O}(f(|X|))$ means that there is a size s and a constant g such that, for any input X with $|X| > s$, $c(x) < g \cdot f(|X|)$ [6]. Now:

Theorem 1 *The problem of determining whether a* GLP *formula holds on a game form interpretation can be solved in time polynomial in the size of the game form and the size of the formula. Formally,*

$$c(M \models \varphi) = \mathcal{O}(|M| \cdot |\varphi|)$$

Outline proof: In order to see that this theorem is true, we need to establish both that the algorithms in Figure 6 and 7 are correct and that the algorithms have the stated efficiency. The eval algorithm is basically an algorithmic representation of the semantics of GLP, and its correctness proof follows from induction on the structure of formulae. The update2 function of Figure 7 is a modified version of Zermelo's algorithm or backward induction [2]. (Instead of only returning the value of the game (either -1 or 1) it also returns a set of transitions that one should avoid in order to realize the value). The correctness of Zermelo's algorithm, we claim, carries over to our algorithm. For the complexity bound, we use that checking propositional formulas in end nodes takes time proportional to the size of the formula: $c(\pi, s \models \varphi_0) = \mathcal{O}(|\varphi_0|)$. In order to prove the main theorem we claim the following about the update function: $c(Up(M, \Gamma, \varphi_0)) = \mathcal{O}(|M| \cdot |\varphi_0|)$. Zermelo's algorithm traverses the whole game tree without visiting a node twice. This takes time proportional to the size of M. In each end node it evaluates propositional logic formulas, which takes time proportional to the size of φ_0. Since the number of end nodes must be less than $|M|$, we obtain our intermediate claim.

```
eval(((Σ, N, T, I), P, π), φ)={
    if φ = □φ0
        for z ∈ Z(T)
            if π, z ⊭ φ0
                return false
        return true
    if φ = ¬ψ
        return ¬eval(((Σ, N, T, I), P, π), ψ)
    if φ = ψ ∨ χ
        return eval(((Σ, N, T, I), P, π), ψ) ∨
               eval(((Σ, N, T, I), P, π), χ)
    if φ = [Γ : φ0]ψ
        let Mu =update(((Σ, N, T, I), P, π), Γ, φ0)
        return eval(Mu, ψ)
}
```

Figure 6. The GLP model checking algorithm

To complete the proof of the main theorem, we note that the evaluation function either evaluates its subformulas φ_{sub} (which takes $\mathcal{O}(|M| \cdot |\varphi_{sub}|)$), or propositional logic formulas φ_{prop} in end nodes ($\mathcal{O}(|M| \cdot |\varphi_{prop}|)$), or it updates with a subformula φ_{sub} ($\mathcal{O}(|M| \cdot |\varphi_{sub}|)$). Finally, note that every part of the formula φ is processed only once.

In the update algorithm we use $c_1(T, x) = \{x\} \cup \{c_1(T, y) | (x, y) \in T\}$ and $c_2(N, T) = T \cap N^2$. These function remove spurious transitions and nodes, such that $\langle c_1(T, x), c_2(c_1(T, x), T), x\rangle$ is always a proper tree.

We have implemented a model checking system for GLP which uses this algorithm. The three example protocols described in this paper are part of the distribution of the model checker. The program can be found at `www.csc.liv.ac.uk/~sieuwert/glp`.

6. Conclusion

What agents do not only depends on their own preferences, but also on their knowledge of other agents' preferences. The logic GLP allows one to analyze multi agent protocols making various assumptions on what is known about agents' preferences. Besides defining this logic and a suitable interpretation over extensive game forms, we show that the model checking problem for this logic has a low computational complexity. We demonstrated how one can specify interesting social problems and protocols as an extensive game form, and how GLP is useful for understanding these protocols.

In GLP one can do semantic updates of a specific kind: we assign a preference to a coalition, and since agents act

```
update(((Σ, N, T, I), P, π), Γ, φ)={
    let (R, v) :=update2(T, π, n_0, x, Γ, φ)
    return ((Σ, c_1(T \ R, n_0), c_2(c(T \ R, n_0), T), I), P, π)
}
update2(T, π, x, Γ, φ)={
    if x ∈ Z(T)
        if eval(π, x, φ)
            return (∅,1)
        return (∅,-1)
    let N := {(x, n)|(x, n) ∈ T}
    let R := ∅ , R2 := ∅ , v := 1
    if I(x) ∈ Γ
        for (x, n) ∈ N
            (Rn, vn) := update2(T, π, n, Γ, φ)
            R := R ∪ Rn
            if vn = 1
                R2 := R2 ∪ {(x, n)}
        if R2 ≠ N
            return (R ∪ R2, 1)
        return (R, −1)
    else
        for (x, n) ∈ N
            (Rn, vn) := update2(T, π, n, Γ, φ)
            R := R ∪ Rn
            if vn = −1
                v := −1
        return (R, v)
}
```

Figure 7. The GLP update function

rational with respect to the known preferences, this constraints the coalition's behaviour. These updates are similar to public announcements in dynamic epistemic logic [18], in the sense that all agents learn commonly from the assumption. We believe the assumption that agents have partial knowledge about each other's preferences is an important one. The issue how the strategic knowledge that arises from this assumption influences agents needs more investigation, which this paper starts by looking at strategic knowledge in finite extensive games of perfect information.

The difference between GLP and a coalition logic like ATEL is that GLP assumes that once a coalition adopts a certain strategy, this strategy is known to all agents. This assumption makes sense from a game theoretic and security viewpoint. We believe this difference makes GLP suitable for the analysis of adversarial multi-agent systems. Harrenstein et al ([12]) also uses preferences over extensive game trees, in this case to find modal formulas over such trees, conceived of as Kripke models, that characterize game theoretic notions, like that of Nash equilibrium. Unlike GLP, [12] does not provide a notion of hypothetical reasoning would players play such an equilibrium.

An extension of GLP would be a framework in which coalitions not only have preferences over outcomes, but over *complete games*. Another extension would be to have parallel updates. Another direction is to develop a similar logic for extensive games with imperfect information. This would allow us to study a wider class of protocols and problems in which information is important.

References

[1] R. Aumann and A. Brandenburger. Epistemic conditions for a nash equilibrium. *Econometrica*, 63:1161–1180, 1995.

[2] K. Binmore. *Fun and Games: A Text on Game Theory*. D. C. Heath and Company: Lexington, MA, 1992.

[3] S. Brahms and D. Taylor. *Fair division: from cake cutting to dispute resolution*. Cambridge University Press, 1996.

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

[5] V. Conitzer and T. Sandholm. Complexity of mechanism design. In *Proceedings of the Uncertainty in Artifi cial Intelligence Conference (UAI), Edmonton, Canada.*, 2002.

[6] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press: Cambridge, MA, 1990.

[7] S. Druiven. Knowledge development in games of imperfect information, 2002. University Maastricht Master Thesis.

[8] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about knowledge*. The MIT Press: Cambridge, MA, 1995.

[9] Y. Gal and A. Pfeffer. A language for modeling agents' decision making processes in games. In *Autonomous Agents and Multi Agent Systems 2003*, Melbourne, July 2003.

[10] Y. Gal and A. Pfeffer. A language for opponent modeling in repeated games. In *AAMAS Workshop on Decision-Theoretic and Game-Theoretic Agents*, Melbourne, July 2003.

[11] A. Gibbard. Manipulation of voting schemes: a general result. *Econometrica*, 41(4), 1973.

[12] B. Harrenstein, W. van der Hoek, J.-J. C. Meyer, and C. Witteveen. On modal logic interpretations of games. In *Procs ECAI 2002*, volume 77, pages 28–32, Amsterdam, July 2002.

[13] H. Kuhn. Extensive games and the problem of information. *Contributions to the Theory of Games*, II:193–216, 1953.

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

[15] M. Pauly. *Logic for Social Software*. PhD thesis, University of Amsterdam, 2001. ILLC Dissertation Series 2001-10.

[16] R. Selten. Reexamination of the perfectness concept for equilibrium points in extensive games. *International journal of game theory*, 4:25–55, 1975.

[17] W. van der Hoek and M. Wooldridge. Cooperation, knowledge, and time: Alternating-time temporal epistemic logic and its applications. *Studia Logica*, 75(4):125–157, 2003.

[18] H. P. van Ditmarsch. *Knowledge Games*. PhD thesis, University of Groningen, Groningen, 2000.