Computing and Informatics, Vol. 34, 2015, 4–22

STABILITY AND STRATEGIC TIME-DEPENDENT BEHAVIOUR IN MULTIAGENT SYSTEMS

Matei Popovici

Research Institute of University of Bucharest (ICUB)
University of Bucharest
Sector 5, 36-46 Mihail Kogalniceanu Blvd, 050107
Romania, Bucharest
e-mail: matei.popovici

Lorina Negreanu

Computer Science Department
POLITEHNICA University of Bucharest
Splaiul Independentei 313, Sector 6
060042 Bucharest, Romania
e-mail: lorina.negreanu@cs.pub.ro

Abstract. Temporal reasoning and strategic behaviour are important abilities of multiagent systems. We introduce a game-theoretic framework suitable for modelling selfish and rational agents which can store and reason about the evolution of an environment, and act according to their interests. Our aim is to identify *stable* interactions: those where no agent has a benefit from changing his behaviour to another. For this reason we deploy the game-theoretic concept of Nash equilibrium and strong Nash equilibrium. We show that not all agent interactions can be stable. Also, we investigate the computational complexity for verifying and checking the existence of stable agent interactions. This paves the way for developing agents which can take appropriate decisions in competitive and strategic situations.

Keywords: Temporal knowledge representation and reasoning, game theory, coordination

Mathematics Subject Classification 2010: 68T27

1 INTRODUCTION

Multiagent systems (MAS) have long been a successful means for modelling the interaction of software agents (or simply, programs) with themselves, other humans, and a given environment. In this context, there is an ever increasing need for MAS:

1. able to perform temporal inferences and
2. capable of strategic reasoning.

As concerns point 1. it implies that agents can store information related to the history of their environment as well as their own past and current state and they are able to infer some implicit knowledge based on such information. We believe this is critical, as it complements the agent's awareness of its own environment, with the awareness of time. The point 2. is justified by scenarios such as stock markets [8] or auctioning [9] where each agent (be it human or software) is assumed to be rational and self-interested, thus acts accordingly.

In this paper, we equip MAS with memory and temporal inference capabilities by deploying a method for temporal reasoning previously described in [4, 5, 13, 12], and we study the strategic behaviour of such systems. In our setting, such a behaviour is characterized by agents' ability to *deviate* whenever this is convenient, i.e. to change his action to another, given the actions of all other agents.

To illustrate such a situation, let us consider the following simplified stock-market scenario with three agents:

Agent 1 − the seller,

Agent 2 − the buyer,

Agent 3 − the supervisor.

On the market, stocks of type T can be sold or bought at a constant rate for a particular time interval. We assume T-stocks are already available for trading, initially[1]. Agent 1 possesses a number of stocks of type T and would like to sell, but before agent 2 is buying. Agent 2 would like to buy stocks of type T, but before a transaction tax is installed and after Agent 1 has finished selling. Agent 3 would like to set a transaction tax on stocks of type T after they are no longer on sale by Agent 1. If the tax is established sooner, it is not justified and deemed as unfair. Alternatively, he would like to monitor the stock market and then establish the tax. However, monitoring the market is a more expensive task.

The scenario is a particular case of a temporal strategic planning problem, in which agents must be coordinate so that all formulated restrictions are satisfied, if possible. More formally, we have four time-dependent properties *sell*, *buy*, *tax*, *monitor*, each controlled by the corresponding agent, and which stand for:

[1] This is achieved, possibly, by players other than 1, 2, 3.

1. *sell stocks on market*,
2. *buy stocks*,
3. *establish transaction tax* and
4. *monitor the stock market*, respectively.

The agent's *action* consists of validating one or more properties he controls at a particular time interval. Each agent has a particular *goal* which is dependent on the current and past state(s) of the environment. In our example, the goals $\mathbf{G}_i$ of agents $i \in \{1, 2, 3\}$ are informally described as: $\mathbf{G}_1$: *sell before buy*, $\mathbf{G}_2$: *buy before tax* and *buy after sell* and $\mathbf{G}_3$: *tax after sell* or *tax before monitor*. Formally, goals are expressed using the temporal language $L_{\mathcal{H}}$, introduced and described in [12].

Also, each action has associated a particular cost. We further assume that agents are *rational* and *self-interested*, meaning they will choose to execute that particular action which:

1. makes the goal satisfied,
2. minimizes costs.

We are interested in the situations where agents cannot individually satisfy their goals i.e. they may have (partially) conflicting and/or (partially) overlapping goals. In such situations, agents may deviate, whenever this is a means for achieving a better outcome. For instance, in our example, if agent 2 cannot buy stocks before the transaction tax is installed, he might change his action from buying stocks to no action. Our aim is to identify those *stable* actions, one for each agent, which ensure that no agent has no incentive to deviate.

With this objective in view, we use a *non-cooperative strategic* game to formally describe scenarios mentioned above. Thus, the actions of all agents are instantaneous[2]. Next, we introduce the game-theoretic concept of Nash equilibrium (NE) which describes MAS stability against individual deviations. Further on, we consider a much more restrictive concept, the strong Nash equilibrium (SNE), where stability is considered with respect to deviations from all possible coalitions of agents.

We further study the computational complexity of verifying and identifying (strong) Nash equilibria, using the $L_{\mathcal{H}}$-model checking procedure described in [12] and identify an upper complexity bound.

This paper extends our previous work from [14] which introduces a game-theoretic framework for studying agent stability, as well as the concept of Nash equilibrium, and finally, identifies the upper complexity bound for the verification and existence checking the Nash equilibria. In our current paper, we look at some limitations of the Nash equilibrium from [14], and propose the strong Nash equilibrium as an improved solution concept. We also examine situations where Nash

[2] In a strategic game, actions are performed just as in the *Rock-paper-scissors*-game [15]. Each agent has no prior knowledge about what the others might do, and all actions are simultaneous.

and strong Nash equilibria exist and extend our computational complexity results to the latter, new concept. Also, we complete our study with completeness results. The latter concept has suggested that polynomial algorithms for computing our solution concepts are not likely to exist. We also discuss alternatives for tackling this issue. Finally, we have made the game-theoretic setting independent of the goal specification language and slightly changed it in order to improve readability.

The paper is structured as follows. In Section 2 we introduce the formal concept of *environment*, which describes the structure of a possibly time-dependent domain. Also, we introduce temporal graphs which capture particular evolutions of such domain. In Section 3, we describe **t**-games which capture strategic scenarios, where agents can make decisions and influence the evolution of the domain. In Section 4, we introduce the Nash and strong Nash equilibrium, and study the **t**-games in which such equilibria exist. In Section 5 we introduce the language $L_{\mathcal{H}}$ [12], as formal means for expressing time-dependent agent goals. We use $L_{\mathcal{H}}$ to study the computational complexity of verifying and checking for the existence of Nash and strong Nash equilibria. Finally, in Sections 6 and 7 we discuss related and future work, and conclude.

2 MODELLING ENVIRONMENTS AND EVOLUTION

A strategic interaction of a MAS occurs in a particular environment. The description of the environment, $\mathcal{E}nv$, models time-dependent properties as labelled directed edges. We call such edges *labelled quality edges*. The label gives the semantics of the property at hand. Quality edges span *action nodes*. An action node models an instantaneous event, which changes the current state of the environment by initiating and terminating a *quality edge*. Each action node occurs at a precise moment of time. The latter are modelled by *hypernodes*.

Formally, we have:

Definition 2.0.1 (Environment)**.** Let $\mathcal{L}abels$ be a set of symbols. An environment is a structure $\mathcal{E}nv = \langle A, E, \mathcal{L} \rangle$ defined with respect to $\mathcal{L}abels$, as follows:

- A is a set whose elements are called *action nodes*
- $E \subseteq A \times A$ is a directed relation which models quality edges
- $\mathcal{L} : E \to \mathcal{L}abels$ is a surjective function which assigns a symbol (label) to each quality edge.

Given the quality edge $(u, v) \in E$, we say that action nodes u or v are the *constructor* or *destructor* nodes of (u, v), respectively. Also, we say that (u, v) is initiated (terminated) by u (v), respectively.

The following example formalizes the setting discussed in the introductory section.

Example 2.0.1 (Environment)**.** Let $\mathcal{L}abels_1 = \{sell, buy, tax, monitor\}$, $A_1 = \{u_1, u_2, u_3, u_4, u_5, u_6, v_1, v_2\}$, $E_1 = \{(u_1, u_2), (u_3, u_4), (u_5, u_6), (v_1, v_2)\}$, $\mathcal{L}_1((u_1, u_2)) = sell$,
$\mathcal{L}_1((u_3, u_4)) = buy$, $\mathcal{L}_1((u_5, a_6)) = tax$ and $\mathcal{L}_1((v_1, v_2)) = monitor$. The environment $\mathcal{E}nv_1 = \langle A_1, E_1, \mathcal{L}_1\rangle$ models a setting with four types of properties, namely *sell*, *buy*, *tax* and *monitor*. The properties are initiated by u_1, u_3, u_5, v_1 and terminated by u_2, u_4, u_6, v_2, respectively.

We note that $\mathcal{E}nv$ merely provides a taxonomy of the domain at hand, giving us information about the available events and what properties they introduce. It provides no information about the actual evolution of the domain at hand. The latter is captured by temporal graphs:

Definition 2.0.2 ((labelled) temporal graph)**.** Let $\mathcal{E}nv = \langle A, E, \mathcal{L}\rangle$ be an environment, H be a set whose elements are called *hypernodes* and which designate moments of time, and $\mathcal{T} : A \to H$ be a function which assigns an action node u, to the hypernode $\mathcal{T}(u)$, i.e. the moment when u takes place.

A labelled temporal graph (or **t**-graph) generated by $\mathcal{E}nv$ and $\mathcal{T}$ is a structure $\mathcal{H}^{\mathcal{T}}_{\mathcal{E}nv} = \langle A', H, \mathcal{T}, E', \mathcal{L}\rangle$ where $A' = \{u \in A :$ the function $\mathcal{T}$ is defined in $u\}$ and $E' = \{(u, v) : u, v \in A'\}$. When the environment and/or the $\mathcal{T}$ function are clear from context, we write $\mathcal{H}$ instead of $\mathcal{H}^{\mathcal{T}}_{\mathcal{E}nv}$. Whenever $\mathcal{T}(u) = \mathcal{T}(v)$ for two action nodes u, v, we say they are *simultaneous*. We denote by $\mathcal{H}set_{\mathcal{E}nv}$ the set of temporal graphs generated by $\mathcal{E}nv$.

Remark 2.0.1 (The interpretation of hypernode)**.** Each hypernode $h \in H$ interprets a *symbolic* moment of time, i.e. one for which no time-stamp is known. For this reason, we do not introduce an a-priori known total ordering of H. However, a (partial) pre-order can be inferred, as follows: $u \geq_{\mathcal{H}} v$ iff: (i) $\mathcal{T}(u) = \mathcal{T}(v)$ or (ii) $(u, v) \in E$ or (iii) there exists u' such that $u' \geq_{\mathcal{H}} v$ and $\mathcal{T}(u) = \mathcal{T}(u')$ or $(u, u') \in E$. We note that $\geq_{\mathcal{H}}$ is reflexive and transitive, however not total.

A temporal graph adds temporal information to an environment by assigning to action nodes u, temporal moments $\mathcal{T}(u)$ when they occur. The precise action nodes and quality edges of a temporal graph are those from the environment which correspond to *executed* action nodes, i.e. those for which the function $\mathcal{T}$ is defined.

Example 2.0.2 (temporal graph)**.** Let $H_1 = \{h_1, h_2, h_3, h_4\}$, and the function $\mathcal{T}_1$, defined as follows: $\mathcal{T}_1(u_1) = h_1$, $\mathcal{T}_1(u_2) = \mathcal{T}_1(u_3) = h_2$, $\mathcal{T}_1(u_5) = h_2$ and $\mathcal{T}_1(u_4) = \mathcal{T}_1(u_6) = h_4$. The temporal graph generated by the environment $\mathcal{E}nv_1$ from Example 2.0.1 and $\mathcal{T}_1$ is $\mathcal{H}_1 = \langle A'_1, H_1, \mathcal{T}_1, E'_1, \mathcal{L}_1\rangle$, where $\mathcal{L}_1$ are taken from Example 2.0.1 and A'_1, E'_1 are built according to Definition 2.0.2. $\mathcal{H}_1$ captures the evolution in which the buyer purchases the stocks precisely when they are no longer being sold and the supervisor inserts the tax while they are being bought. $\mathcal{H}_1$ is shown in Figure 1.

We note that quality edges have a dual representation. On one hand, they are edges of the form (u, v) which are explicit parts of the temporal graph. The edge

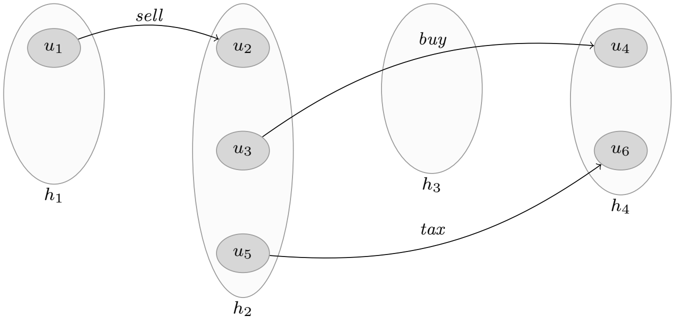

Figure 1. The temporal graph $\mathcal{H}_1$ for the evolution of the stock market

gives a temporal dimension to the modelled property. It is initiated by u, terminated by v and the interval when it is true is given by $[\mathcal{T}(u), \mathcal{T}(v)]$. On the other hand, the quality edge is labelled by a symbol, and the symbol encodes the semantics of the property. In [12], we also treat the more interesting case where labels can also be relational instances such as $P(a, b, c)$. Since the labels of temporal graphs do not affect in any way the formulation of our game-theoretic setting, in this paper, we adopt the restricted form of symbols only.

3 MODELLING A STRATEGIC INTERACTION

We use the term *strategic interaction* to describe any circumstance in which an individual's situation is influenced by the goals and interventions of other individuals.[3] In our scenario, the actors of a strategic interaction are agents. We assume they are selfish and rational in the game-theoretic sense [11]. Each agent i controls a limited set of action nodes $\mathcal{C}trl_i$. (S)he can perform an action $a_i = (A_i, \mathcal{T}_i)$. Agent actions are different from action nodes. The latter are merely events in a temporal graph. The former should be understood in the game-theoretic sense. They are interventions of the agent in the environment. The action has two components: $A_i \subset \mathcal{C}trl_i$ is a subset of the controlled action nodes of i, which i will execute. $\mathcal{T}_i$ is a function which assigns for each action node $u \in A_i$ the moment $\mathcal{T}_i(u)$ when it will be executed by the agent.

The outcome of a strategic interaction is a sequence of actions, one for each agent, i.e. an action profile. The latter can also be viewed as *the outcome* temporal graph, i.e. the one constructed from all agents' actions.

[3] We note that interaction should not be understood as a one-to-one relationship between pairs of agents. It should rather be viewed as means by which each agent can, by his own actions, affect the satisfaction of the goals of others. This is quite similar to the case of a game of poker. Here, players do not interact directly. They do so via the cards they choose to play. Each such action affects the outcome of the game, and hence, the winner.

Also, executing each action node u comes with a cost $c(u)$. We assume that agents have a limited number of resources which they can spend. This is captured by values g_i, one for each agent i. Finally, each agent has a goal which (s)he would like to enforce. The goal of agent i is defined as the set $\mathbf{G}_i$ of temporal graphs which the agent considers desirable. Later on, in Section 5, we shall provide and discuss a more compact representation for goals.

Formally, the strategic interaction is modelled by a temporal game (or $\mathbf{t}$-game), as follows:

Definition 3.0.3 ($\mathbf{t}$-game)**.** Let N be a set of agents, $\mathcal{E}nv$ be an environment and H be a set of hypernodes. A $\mathbf{t}$-game is a structure:

$$\mathcal{G} = \langle N, \mathcal{E}nv, H, (\mathcal{C}trl_i)_{i\in N}, c, (\mathbf{G}_i)_{i\in N}, (g_i)_{i\in N}\rangle$$

where each $\mathcal{C}trl_i \subseteq A$ is a subset of A (taken from $\mathcal{E}nv$) of action nodes controlled by agent i, $c : A \to \mathbb{R}^+$ is a cost function, each $\mathbf{G}_i \subset \mathcal{H}set_{\mathcal{E}nv}$ is the goal of agent i and each g_i is the amount of available resources of agent i. Also, $(\mathcal{C}trl_i)_{i\in N}$ is a proper partition of A, that is: $\mathcal{C}trl_i \cap \mathcal{C}trl_j = \emptyset$ for any $i \neq j \in N$ and $\cup_{i\in N}\mathcal{C}trl_i = A$.

Example 3.0.3 ($\mathbf{t}$-game)**.** The strategic interaction between the seller, buyer and supervisor, described informally in the introduction, is modelled by the $\mathbf{t}$-game $\mathcal{G}_1 = \langle N, \mathcal{E}nv_1, H_1, (\mathcal{C}trl_i)_{i\in N}, c, (\mathbf{G}_i)_{i\in N}, (g_i)_{i\in N}\rangle$ where H_1 is taken from Example 2.0.2, $N = \{1, 2, 3\}$ (1 denotes the seller, 2 – the buyer and 3 – the supervisor), $\mathcal{E}nv_1$ is taken from Example 2.0.1, $\mathcal{C}trl_1 = \{u_1, u_2\}$, $\mathcal{C}trl_2 = \{u_3, u_4\}$, $\mathcal{C}trl_3 = \{u_5, u_6, v_1, v_2\}$, $c(u) = 1$ for all $u \in A$, $\mathbf{G}_1 = \{\mathcal{H}_{\mathcal{E}nv_1} : \textit{sell}$ occurs before $\textit{buy}\}$, $\mathbf{G}_2 = \{\mathcal{H}_{\mathcal{E}nv_1} : \textit{buy}$ occurs before $\textit{tax}$ and after $\textit{sell}\}$, $\mathbf{G}_3 = \{\mathcal{H}_{\mathcal{E}nv_1} : \textit{tax}$ occurs after $\textit{sell}$ or before $\textit{monitor}\}$ and $g_1 = g_2 = g_3 = 5$. Thus, the seller (1), controls the actions which create/terminate the property *sell*, the buyer (2) controls those which create/terminate *buy*, while the supervisor (3) - those which create/terminate *tax* and *monitor*.

In what follows, we assume $\mathcal{G}$ is a $\mathbf{t}$-game.

Remark 3.0.2 (The interpretation of H)**.** We note that H i.e. the set of hypernodes, is given in advance, and without any specified ordering. H can be interpreted as a set of *slots*, where agents can perform their actions. In this context, those are the agents which establish a (partial) ordering of H, via their actions, more precisely, via the quality edges which they set, according to Remark 2.0.1.

Definition 3.0.4 (Action profile)**.** An action of agent i in the $\mathbf{t}$-game $\mathcal{G}$ is a pair $a_i = (A_i, \mathcal{T}_i)$ where $\mathcal{T}_i : A_i \to H$ is a surjective function, $A_i \subseteq \mathcal{C}trl_i$ and each $\mathcal{C}trl_i$ together with H are taken from $\mathcal{G}$. An action profile a in the $\mathbf{t}$-game $\mathcal{G}$ assigns to each agent $i \in N$, the action a_i. Formally, it is a sequence $a = (a_i)_{i\in N}$. We write a_{-i} to refer to the action profile a which does *not* contain a_i, a_C to refer to the sequence $(a_i)_{i\in C}$ and a_{-C} to refer to the action profile a which does *not* contain a_i for all $i \in C$.

An action profile provides a complete specification of what all agents in the MAS will do, and corresponds to a temporal graph which stores the evolution of the environment, according to each agents' action. The construction of the temporal graph simply takes all the action nodes executed by each agent, and introduces each of them at the moment when the agent desires to execute them. Formally, the temporal graph $\mathcal{H}(a)$ resulted from the action profile $a = ((A_i, \mathcal{T}_i))_{i \in N}$ in the $\mathbf{t}$-game $\mathcal{G}$, is the temporal graph generated by $\mathcal{E}nv$, H (which are taken from $\mathcal{G}$) and the function $\mathcal{T}$, which is built as follows: $\mathcal{T}(u_i) = \mathcal{T}_i(u_i)$ for each $u_i \in A_i$. We say that $\mathcal{H}(a)$ is the *outcome* of a.

Example 3.0.4 (Action profile)**.** We continue Example 3.0.3. The action profile a^1 in $\mathcal{G}_1$ which produces the temporal graph $\mathcal{H}_1$ from Example 2.0.2 is:

$$a^1 = ((\{u_1, u_2\}, \mathcal{T}_1), (\{u_3, u_4\}, \mathcal{T}_2), (\{u_5, u_6\}, \mathcal{T}_3))$$

where $\mathcal{T}_1(u_1) = h_1$, $\mathcal{T}_1(u_2) = h_2$, $\mathcal{T}_2(u_3) = h_2$, $\mathcal{T}_2(u_4) = h_4$, $\mathcal{T}_3(u_5) = h_2$ and $\mathcal{T}_3(u_6) = h_4$.

Each outcome of the interaction produces a certain benefit or disadvantage for each agent. Naturally, this is dependent on:

1. the costs which each agent has to support and
2. whether or not the agent's goal is satisfied.

Definition 3.0.5 (Cost, goal satisfaction, utility)**.** Given a temporal graph $\mathcal{H}(a)$ which is the outcome the action profile $a = ((A_i, \mathcal{T}_i))_{i \in N}$, we define the cost of agent i as $cost_i(\mathcal{H}(a)) = \sum_{u \in A_i} c(i)$. The cost is computed as the sum of costs of all action nodes executed by i. We say the goal of agent i is satisfied, if $\mathcal{H}(a) \in \mathbf{G}_i$. Finally, the utility of agent i in $\mathcal{H}(a)$ is given by:

$$u_i(\mathcal{H}(a)) = \begin{cases} g_i - cost_i(\mathcal{H}(a)) & \text{if } \mathcal{H}(a) \in \mathbf{G}_i \\ -cost_i(\mathcal{H}(a)) & \text{otherwise} \end{cases}$$

Example 3.0.5 (Cost, utility)**.** The cost supported by each agent in $\mathcal{H}_1$ (Example 2.0.2) is $cost_i(\mathcal{H}_1) = 1 + 1 = 2$ for $i \in \{1, 2, 3\}$. The utility of the seller in $\mathcal{H}_1$ is $u_1(\mathcal{H}_1) = g_1 - cost_1(\mathcal{H}_1) = 5 - 2 = 3$, since the goal of the seller is satisfied. However, the goal of the buyer is not satisfied, since *buy* does not occur after *tax*, therefore his/her utility is $u_2(\mathcal{H}_1) = 0 - cost_2(\mathcal{H}_1) = -2$.

Utility values provide a quantitative measure of an agent's welfare in a given outcome (temporal graph). Beside costs, it also incorporates a qualitative component, namely goal satisfaction. If the goal is satisfied, the utility of an agent is computed as the amount of resources he is able to save (initial resources minus costs), while satisfying his goal. If the goal is not satisfied, the agent is penalized: his/hers utility is given by the negative value of his costs.

The choice for defining utility as above follows two assumptions:

1. agents must be incentivised to satisfy their goals, in the cheapest way possible and
2. the agents are resource-bounded, thus they cannot spend infinitely much for satisfying their goal.

Naturally, an agent will prefer a temporal graph over another, depending on the amount of utility which (s)he obtains in either of the two:

Definition 3.0.6 (Preference)**.** We say an agent i (strictly) prefers a temporal graph $\mathcal{H}$ over $\mathcal{H}'$ and write $\mathcal{H} \succ_n \mathcal{H}'$, if $u_i(\mathcal{H}) > u_i(\mathcal{H}')$. We also extend the preference relation over action profiles. Given two action profiles a, a', we write $a \succ_i a'$ iff $\mathcal{H}(a) \succ_i \mathcal{H}'(a)$. The non-strict preference relation $\succeq_i$ is defined with respect to $\geq$ in a similar way. Also, we lift the preference relations to coalitions $C \subseteq N$ of agents: we write $a \succ_C a'$ when $a \succ_i a'$ for all $i \in C$.

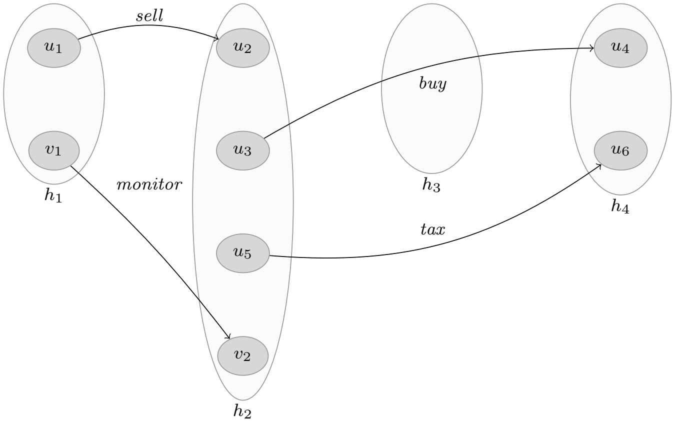

Figure 2. $\mathcal{H}_2$ – A better temporal graph for the supervisor

Example 3.0.6 (Preference)**.** Consider the temporal graph $\mathcal{H}_2$ from Figure 2, also generated by the environment $\mathcal{E}nv_1$[4]. First, note that $u_3(\mathcal{H}_2) = 5-(1+1+1+1) = 1$, since, in this scenario, the supervisor monitors the market before setting the tax and $u_3(\mathcal{H}_1) = 5 - (1 + 1)$ since in $\mathcal{H}_1$ the tax is introduced after *sell*. Thus $\mathcal{H}_1 \succ_3 \mathcal{H}_2$. Similarly $\mathcal{H}_2 \succeq_1 \mathcal{H}_1$ but $\mathcal{H}_2 \not\succ_1 \mathcal{H}_1$ and $\mathcal{H}_1 \not\succ_1 \mathcal{H}_2$: the seller does not strictly prefer a temporal graph over another, since he has the same utility in both of them.

[4] together with H_1 and the function $\mathcal{T}_2$, defined as follows: $\mathcal{T}_2(u_1) = \mathcal{T}_2(v_1) = h_1$, $\mathcal{T}_2(u_2) = \mathcal{T}_2(u_3) = \mathcal{T}_2(u_5) = \mathcal{T}_2(v_2) = h_2$ and $\mathcal{T}_2(u_4) = \mathcal{T}_2(u_6) = h_4$

4 SOLUTION CONCEPTS

In the previous section we have formally described strategic settings by **t**-games. They consist of agents, their ability to execute cost-dependent action nodes, their goals and available resources. An interaction in such a setting consists of an action profile, in which each agent performs an action. The outcome of an interaction is a temporal graph. Also, we have seen that agents prefer certain temporal graphs over others, depending on costs and goal satisfaction.

In this section, we are interested in looking at those action profiles which are *stable*, more precisely, those where agents have no incentive to deviate.

Definition 4.0.7 (Nash equilibrium)**.** A *Nash equilibrium* (NE) of a **t**-game is an action profile a^* such that, for each agent $i \in N$ and each action profile $a \neq a^*$ we have that $(a^*_{-i}, a^*_i) \succeq_i (a^*_{-i}, a_i)$. A *Nash-stable* temporal graph is $\mathcal{H} = \mathcal{H}(a)$ where a is a Nash equilibrium.

Example 4.0.7 (Nash equilibrium)**.** Let $\mathcal{H}_3$ which is obtained from $\mathcal{H}_1$ by removing action nodes u_3 and u_4 and the quality edge (u_3, u_4) and $\mathcal{H}_1 = \mathcal{H}(a^1)$, where a^1 is the action profile from Example 3.0.4. Note that $\mathcal{H}_1 \succ_3 \mathcal{H}_2$ since in $\mathcal{H}_1$ the goal of the supervisor is satisfied in a cheaper way than in $\mathcal{H}_2$. Thus, $\mathcal{H}_2$ is not a Nash equilibrium. Also $\mathcal{H}_3 \succ_2 \mathcal{H}_2$. The goal of the buyer is not satisfied in either temporal graph, however, since in $\mathcal{H}_3$ the buyer executes no action node, his/her utility is 0, whereas in $\mathcal{H}_2$ the utility is -2.

Finally, $\mathcal{H}_3$ is a Nash equilibrium since no player has the ability to individually come up with a better action.

The following proposition settles the question regarding stability of arbitrary **t**-games.

Proposition 4.0.1 (Nash equilibrium existence)**.** There are **t**-games where no Nash equilibrium exists.

Proof. [Sketch] Let $\mathcal{G}_2$ be the **t**-game obtained from $\mathcal{G}_1$ (from Example 3.0.3), where the goals of the buyer are modified as: $\mathbf{G}_2 = \{\mathcal{H}_{\mathcal{E}nv_1} : \mathit{tax} \text{ does not occur during } \mathit{buy}\}$ and $\mathbf{G}_3 = \{\mathcal{H}_{\mathcal{E}nv_1} : \mathit{tax} \text{ occurs during } \mathit{buy}\}$. There exists no Nash equilibrium of $\mathcal{G}_2$. This is motivated by the following reasoning: let us assume that the buyer starts to buy stocks. If the supervisor takes no action, the goal of the buyer is satisfied since there is no *tax* property in the temporal graph. However, the goal of the supervisor is not satisfied, and he is incentivised to set the tax on the interval when *buy* is true. This action would satisfy his goal, but invalidate that of the buyer. Thus, now the buyer has an incentive to deviate, either to set *buy* in another interval, or, if this is not possible, not to set *buy* at all. This deviation increases the buyer's utility from a strictly negative value, to either a positive one, or to zero. As a response to the buyer's deviation, the supervisor is again incentivised to deviate, and the whole process repeats itself. The conclusion is that there exists no Nash-stable temporal graph in $\mathcal{G}_2$. $\square$

As illustrated by Example 4.0.7, the Nash equilibrium captures only individual deviations. Such deviations do not necessarily produce a globally desirable temporal graph. This is the case with $\mathcal{H}_3$ from Example 4.0.7. $\mathcal{H}_3$ is Nash-stable, however the goal of the buyer is not satisfied. Also, it might be the case that some goal is satisfied, but with a high cost. For these reasons, we introduce the strong Nash equilibrium which captures a much stronger notion of stability.

Definition 4.0.8 (strong Nash equilibrium)**.** A *strong Nash equilibrium* (SNE) of a $\mathbf{t}$-game is an action profile a^* such that, for each coalition of agents $C \subseteq N$ and each action profile $a \neq a^*$ we have that $(a^*_{-C}, a^*_C) \succeq_C (a^*_{-C}, a_C)$. A *strong Nash-stable* temporal graph is $\mathcal{H} = \mathcal{H}(a)$ where a is a strong Nash equilibrium.

Example 4.0.8 (strong Nash equilibrium)**.** The temporal graph $\mathcal{H}_3$ from Example 4.0.7 is not a strong Nash equilibrium. Coalition $\{1, 2, 3\}$ can deviate and implement a better temporal graph, namely $\mathcal{H}_4$ from Figure 3. Note that, in $\mathcal{H}_4$, the utility of the buyer is $u_2(\mathcal{H}_4) = u_3(\mathcal{H}_4) = 5 - (1 + 1) = 2$, which is strictly higher than the one obtained in $\mathcal{H}_3$. However, $\mathcal{H}_4$ is a strong Nash equilibrium. $\mathcal{H}_4$ shows that, by cooperating, the buyer and the supervisor are mutually better of: the buyer can acquire his stocks before the tax is installed, and thus satisfy his goal.

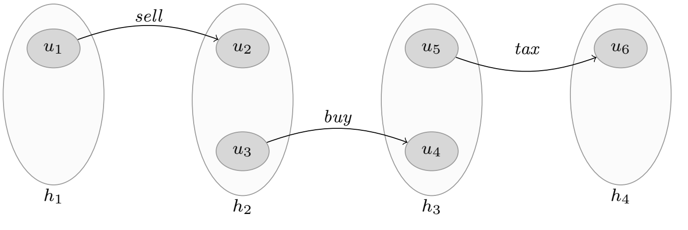

Figure 3. The strong-Nash stable temporal graph $\mathcal{H}_4$

Proposition 4.0.2. There are $\mathbf{t}$-games where no strong Nash equilibrium exists.

Proof. [Sketch] The reasoning is similar to that from the proof of Proposition 4.0.1. In the same $\mathbf{t}$-game $\mathcal{G}_2$ there exists no strong Nash equilibrium. For each possible temporal graph in $\mathcal{G}_2$, there exists a coalition, which contains either the buyer or the supervisor, which has an incentive to deviate. □

The behaviour of the strong Nash equilibrium simulates a form of coordination. In temporal graphs which are strong Nash stable, groups of agents seek the best available joint action.

Remark 4.0.3 (Discussion on $\mathbf{t}$-games)**.** As mentioned in the introduction, in a $\mathbf{t}$-game, the actions of each agent are simultaneous, as in the *Rock-paper-scissors*-game [15]. Moreover, agents are not aware of the actions taken by others. The only disclosed information about other agents is that given by the **t**-game itself: available actions, costs, goals, goal values.

Although, in most practical scenarios, actions are not simultaneous, it is reasonable to assume they are not publicly advertised, thus they remain unknown to other agents. This assumption is also justified by feasibility. Normal-form games, the ones in which the interaction is not necessarily simultaneous, are more complex, and require more computationally expensive solution concepts.

5 EXPRESSING GOALS

So far, we have expressed goals as sets of temporal graphs, which satisfy certain conditions, expressed in natural language. In this section, we take on a more formal approach, and use formulae from the temporal language $L_{\mathcal{H}}$ [12] for expressing such conditions. The motivation for our approach is twofold: on the one hand, expressing goals in such a way is more compact and straightforward. On the other hand, it offers means for computing Nash and strong Nash equilibria based on the $L_{\mathcal{H}}$ model checking procedure [12].

5.1 The Language $L_{\mathcal{H}}$

$L_{\mathcal{H}}$ allows the expression of temporal constraints between the occurrence of quality edges in a temporal graph. In what follows, we briefly introduce and explain the syntax and semantics of $L_{\mathcal{H}}$. For an extended discussion on $L_{\mathcal{H}}$, we defer the interested reader to [12]. Again, we consider $\mathcal{L}abels$ is a set of symbols.

Definition 5.1.1 ($L_{\mathcal{H}}$ syntax)**.** Let $p, q \in \mathcal{L}abels$. The syntax of $L_{\mathcal{H}}$ is recursively as follows:

$$\varphi ::= p \mid p \propto \varphi_1 \mid \neg\varphi_1 \mid p \propto \varphi_1 \wedge p \propto \varphi_2$$

where $\propto\, \in \{\mathbf{b}, \mathbf{a}, \mathbf{o}\}$[5] is a temporal connective, and $\mathbf{b}, \mathbf{a}, \mathbf{o}$ stand for *before*, *after* and *overlaps*, respectively. Two quality edges overlap if they have exactly the same life-span (start and end in action nodes from the same hypernode). Also, the logical connective $\vee$ is defined with respect to $\neg$ and $\wedge$ in the standard way.

Example 5.1.1 ($L_{\mathcal{H}}$ syntax)**.** We consider the same set $\mathcal{L}abels$ as the one introduced in Example 2.0.1. The following:

- $\varphi_1 = sell\mathbf{b}buy$
- $\varphi_2 = buy\mathbf{b}tax \wedge buy\mathbf{a}sell$
- $\varphi_3 = tax\mathbf{a}buy \vee tax\mathbf{a}monitor$

are valid $L_{\mathcal{H}}$-formulae.

[5] In this paper, for the brevity of our exposition, we use only a limited number of temporal connectives, which naturally limits the expressive power of $L_{\mathcal{H}}$. The full necessary set of connectives is discussed in [12].

In order to identify the quality edges in a temporal graph which satisfies a $L_{\mathcal{H}}$-formula, the concept of *path* is important. A path is a finite sequence $u_1, \ldots, u_n$ of action nodes such that each adjacent u_i and u_j:

1. share the same hypernode ($\mathcal{T}(u_i) = \mathcal{T}(u_j)$) or
2. are the constructor and destructor of a quality edge, respectively ($(u_i, u_j) \in E$).

Intuitively, paths are used to identify the precendence (or temporal order) of quality edges. For instance, the quality edge (u, v) occurs before (u', v') in $\mathcal{H}$ if there is a path from v to u in $\mathcal{H}$.

Definition 5.1.2 ($L_{\mathcal{H}}$ semantics)**.** The evaluation of a $L_{\mathcal{H}}$-formula done with respect to a temporal graph $\mathcal{H}$ and is given by the mapping $\| \cdot \|_{\mathcal{H}} : L_{\mathcal{H}} \to 2^E$ which assigns for each formula $\varphi \in L_{\mathcal{H}}$ a set of quality edges $\|\varphi\|_{\mathcal{H}}$ which satisfy it. $\| \cdot \|_{\mathcal{H}}$ is defined as follows:

- $\|p\|_{\mathcal{H}} = \{(u, v) \in E : \mathcal{L}((u, v)) = p\}$
- $\|\neg p\|_{\mathcal{H}} = \{(u, v) \in E : \mathcal{L}((u, v)) \neq p\}$
- $\|p\mathbf{b}\varphi\|_{\mathcal{H}} = \{(u, v) \in \|p\|_{\mathcal{H}}$: there exists $(u', v') \in \|\varphi\|_{\mathcal{H}}$ such that there is a path from v to $u\}$
- $\|\neg(p\mathbf{b}\varphi)\|_{\mathcal{H}} = \{(u, v) \in \|p\|_{\mathcal{H}}$: there exists **no** $(u', v') \in \|\varphi\|_{\mathcal{H}}$ such that there is a path from v to $u\}$
- $\|p \propto \varphi \wedge p \propto \varphi'\|_{\mathcal{H}} = \|p \propto \varphi\|_{\mathcal{H}} \cap \|p \propto \varphi'\|_{\mathcal{H}}$
- $\|\neg(\varphi \wedge \varphi')\|_{\mathcal{H}} = \|\neg\varphi \vee \neg\varphi'\|_{\mathcal{H}}$

The cases corresponding to other temporal connectives are defined in a similar way. For details, see [12].

Example 5.1.2 (Semantics)**.** We consider the formulae φ_i, $i \in \{1, 2, 3\}$ from Example 5.1.1. $\|\varphi_1\|_{\mathcal{H}_1} = \{(u_1, u_2)\}$, since the action node u_2 is directly connected to u_3 (by hypernode h_2) and (u_3, u_4) is labelled with *buy*. $\|\varphi_2\|_{\mathcal{H}_1} = \emptyset$ since there is no quality edge labelled with *buy* which occurs after one labelled with *tax*. However, $\|\varphi_2\|_{\mathcal{H}_4} = \{(u_3, u_4)\}$, since $\|buy\mathbf{a}sell\|_{\mathcal{H}_4} = \{(u_3, u_4)\}$, $\|buy\mathbf{b}tax\|_{\mathcal{H}_4} = \{(u_3, u_4)\}$, and $\|\varphi_2\|_{\mathcal{H}_4} = \|buy\mathbf{a}sell\|_{\mathcal{H}_4} \cap \|buy\mathbf{a}sell\|_{\mathcal{H}_4}$. Similarly, $\|\varphi_3\|_{\mathcal{H}_4} = \{u_5, u_6\}$.

Now, we can use $L_{\mathcal{H}}$-formule to express goals. Given an environment $\mathcal{E}nv$ and a goal formula ϕ_i of agent i, the goal set $\mathbf{G}_i$ can be expressed as $\mathbf{G}_i = \{\mathcal{H}_{\mathcal{E}nv} : \|\phi_i\|_{\mathcal{H}_{\mathcal{E}nv}} \neq \emptyset\}$. In other words, a temporal graph $\mathcal{H}_{\mathcal{E}nv}$ generated by $\mathcal{E}nv$ satisfies the goal formula ϕ_i iff the evaluation of ϕ_i in $\mathcal{H}_{\mathcal{E}nv}$ yields at least one quality edge.

Proposition 5.1.1 ([12])**.** Given a formula $\phi \in L_{\mathcal{H}}$ and a quality edge (u, v), the decision problem $(u, v) \in \|\phi\|_{\mathcal{H}}$ is in PTIME. Also, computing the set $\|\phi\|_{\mathcal{H}}$ can be done in polynomial time with respect to the size of the formula ϕ and that of the temporal graph where ϕ is evaluated.

5.2 Computing NE and SNE

In this section, we look at the computational complexity of two well-established types of problems, namely:

1. *verifying* if an action profile is a (strong) Nash equilibrium of a given $\mathbf{t}$-game and
2. establishing whether or not *there exist* (strong) Nash equilibria in a given $\mathbf{t}$-game.

Finally, we discuss the impact of our results to the development of algorithms which solve these two problems.

Proposition 5.2.1 ((S)NE verification)**.** Checking whether an action profile is a Nash equilibrium or a strong Nash equilibrium of a $\mathbf{t}$-game with goals formulated in the language $L_{\mathcal{H}}$, is *coNP*-complete.

Proof. Consider the complement of the problem: given a $\mathbf{t}$-game $\mathcal{G}$ and a temporal graph $\mathcal{H}$, is $\mathcal{H}$ *not* a (strong) Nash equilibrium.

Nash Equilibrium Membership in NP (sketch): Let a be the action profile which produces $\mathcal{H}$. We non-deterministically build all possible action profiles a^* and agents i, in non-deterministic polynomial time (in the number of agents and quality edges from the environment). We then check $(a_{-i}, a_i) \succeq_i (a_{-i}, a_i^*)$ which can be done in deterministic polynomial time. The preference relation is computed by:

1. summing up all costs,
2. determining whether the goal of i is satisfied (which can be done in polynomial time, according to Proposition 5.1.1)
3. if it is the case, subtracting the resources from the total cost.

NP hardness: We build a reduction from SAT to our problem. Consider a propositional formula $\varphi = T_1 \wedge T_2 \wedge \ldots \wedge T_n$, where each T_i is of the form $T_i = \alpha_{i1} \vee \alpha_{i2} \vee \ldots \vee \alpha_{ik}$, where each α_{ij} is of the form x or $\neg x$, and x is a variable. We model an *interpretation* (assignment of a truth value to each variable) as a temporal graph $\mathcal{H}$ of an environment $\mathcal{E}nv$, containing two hypernodes. $\mathcal{E}nv$ is given by: $H = \{h_1, h_2\}$. For each variable x of φ, we build action nodes u_x^c, u_x^d and quality edge (u_x^c, u_x^d), labelled as p_x. Thus, each variable has a corresponding quality edge. We further add a distinguished quality edge (u_*^c, u_*^d) labelled as p_*. The temporal graph $\mathcal{H}$ generated by $\mathcal{E}nv$ is built as follows: We fix the span of p_* from h_1 to h_2 ($\mathcal{T}(u_*^c) = h_1$ and $\mathcal{T}(u_*^d) = h_2$). Whenever a variable x is true, we fix $\mathcal{T}(u_x^c) = h_1$ and $\mathcal{T}(u_x^d) = h_2$. Thus p_x holds in $\mathcal{H}$ between h_1 and h_2. Whenever x is false, we do not set p_x in $\mathcal{H}$ at all. Thus, the atomic propositional formula x corresponds to the $L_{\mathcal{H}}$ formula $p_* \,\mathbf{o}\, p_x$, which means that p_x must overlap with p_*.

The formula $\neg x$ corresponds to $\neg(p_* \textbf{ o } p_x)$, which means that it cannot be the case that p_* overlaps with p_x.

We build a goal formula $\varphi_{L_\mathcal{H}}$ from φ by replacing each atomic positive and negated literal by the $L_\mathcal{H}$-formulae described above. Now φ is satisfiable iff there exists a temporal graph $\mathcal{H}$ generated by $\mathcal{E}nv$ such that $\|\varphi_{L_\mathcal{H}}\| = \{(u_*^c, u_*^d)\}$.

Next, we built a $\mathbf{t}$-game $\mathcal{G}$ with $N = 2$. The first agent, 1, controls the property p_* and has as goal, the formula p_*. The second agent, 2, controls all variable properties p_x and has as goal $\varphi_{L_\mathcal{H}}$. The costs of setting all variables are 0. The goal values are 1 for both agents. We build an action profile a in which the first agent sets p_* and the other does nothing.

$\Longrightarrow$ **(sketch).** If φ is satisfiable, then agent 2 has an incentive to deviate and execute those action nodes which will make $\|\varphi_{L_\mathcal{H}}\| \neq \emptyset$. Thus a is not a Nash equilibrium.

$\Longleftarrow$ **(sketch).** If a is not a Nash equilibrium, then there exists an agent which can deviate to increase his utility. The first agent has the maximal utility of 1. However, agent 2 has utility 0 and could achieve 1 if his goal is satisfied. Thus, he is the only one that can deviate. If a deviation exists, this must make $\|\varphi_{L_\mathcal{H}}\| \neq \emptyset$. Thus φ is satisfiable.

Strong Nash equilibrium. The proof is similar to that of the Nash equilibrium case, except that we consider groups of agents instead of individual agents. The same reduction can be used for showing $coNP$-completeness.

□

Proposition 5.2.2 ((S)NE existence)**.** Checking if there exists an action profile is a Nash equilibrium or strong Nash equilibrium of a $\mathbf{t}$-game with goals formulated in the language $L_\mathcal{H}$, is in Σ_2.

Proof.

Nash equilibrium. We show the complement of the problem is in $\Sigma_2 = NP^{coNP}$. We non-deterministically build all action profiles if the $\mathbf{t}$-game at hand, as illustrated in the proof of Proposition 5.2.1. This can be done in non-deterministic polynomial time. Hence, the building process is in NP. For each such built action profile, we apply the procedure from Proposition 5.2.1, to establish if the profile is a Nash equilibrium. Thus, the entire existence checking process is in NP, if we use an oracle from $coNP$ (precisely the algorithm described in Proposition 5.2.1. Hence, the existence checking is in NP^{coNP}.

Strong Nash equilibrium. The proof is similar to the above case.

□

We conjecture that the existence checking is also complete for Σ_2.

The above results are, at least as first sight, quite pessimistic. They show that verification is as hard a problem as the hardest problems from NP, while existence checking is in Σ_2 and possibly complete for this class. However:

1. the complexity results coincide with those associated to similar problems. For instance, computing Nash equilibria [7] in propositional logic games are (co-)NP complete. If we take into account the fact that goals formulated in $L_{\mathcal{H}}$ are more expressive that the ones expressed in propositional logic, our results are quite optimistic. They show that we can *add temporal reasoning for free* to a strategic interaction, in terms of computational complexity;
2. although NP-completeness implies the exploration of a solution space of exponential size, this can be feasible for any MAS with a small number of agents. Thus, SAT solvers [10], which are known to be very fast, can be employed for determining Nash and strong Nash equilibria.

6 RELATED WORK

We consider Boolean (or propositional logic) Games [7] to be one of the first frameworks which use logic for describing goal-based strategic interactions between agents. We also note [3] which generalises the former setting from 2 to n-player scenarios. We believe that our approach is conceptually similar. However, instead of propositional logic, we use the more expressive language $L_{\mathcal{H}}$, which also allows expressing domain-dependent temporal properties. Thus, the outcome of a game, which in the former setting is an interpretation of variables, is replaced here by a temporal graph.

The choice of $L_{\mathcal{H}}$ over well-known temporal logics such as LTL [16] or CTL [2], is motivated, on one hand, by the increased computational complexity of model checking in the case of LTL [16] and by the reduced expressive power of CTL, with respect to the extended version of $L_{\mathcal{H}}$ presented in [12]. We also note that, unlike temporal graphs, Kripke Structures – the models of CTL and LTL – may become inefficient in storing the evolution of large domains. Consider a property P which holds over a number n of states. In a Kripke Structure, storing the property P involves labelling each of the n states. As we have previously seen, in our approach, storing a property P does not depend in any way on the number of states (hypernodes) which it covers. For a more detailed comparison between $L_{\mathcal{H}}$ and standard temporal logics we defer the reader to [12].

There is a considerable amount of work aimed at expressing the strategic abilities of agents, such as ATL [1] and (Quantified) Coalition Logics [6]. These languages are aimed at expressing stability conditions, such as the ones captured by the Nash and strong Nash equilibrium. The final purpose is to analyze general properties of games. Our approach is considerably different. We consider the stability conditions as fixed (more precisely, they are NE and SNE) and analyze the possible domain (instead of game) evolutions which satisfy our conditions.

7 CONCLUSION AND FUTURE WORK

While the scope of our paper is rather formal, we believe our results can be easily put into practice. Our complexity results show that implementations are possible, if proper tools, such as SAT solvers, are deployed. Also, by introducing *limited memory*, that is, by truncating temporal graphs to a fixed number of hypernodes, the computational effort may be further controlled. As suggested by all the examples above, our modelling method can be used for identifying stable behaviour of agents in time-dependent environments. For clarity, we have employed a very simple fictitious stock-market example. However, we believe that our approach has a wide range of applicability, in areas such as intelligent environments and ambient intelligence, where both humans and devices interact in order to achieve individual interests. We are currently working in deploying our approach in this setting. This would naturally require efficient algorithms and appropriate techniques to tame the complexity of the problems at hand. We leave these issues for the future work.

Acknowledgement

The authors wish to acknowledge the help and support from Prof. Cristian Giumale, which was the first to think about temporal graphs, and whose guidance patroned the work presented in this paper.

The work has been funded by Project 264207, ERRIC-Empowering Romanian Research on Intelligent Information Technologies/FP7-REGPOT-2010-1.

This paper was supported by the project "Research scholarships for young researchers, Competition 2015" financed by the University of Bucharest via the Research Institute of University of Bucharest (ICUB).

REFERENCES

[1] Alur, R.—Henzinger, T. A.—Kupferman, O.: Alternating-Time Temporal Logic. In Revised Lectures from the International Symposium on Compositionality: The Significant Difference (COMPOS '97), Springer-Verlag, London, UK, 1998, pp. 23–60.

[2] Arnold, A.—Crubille, P.: A Linear Algorithm to Solve Fixed-Point Equations on Transition Systems. Inf. Process. Lett., Vol. 29, 1988, No. 2, pp. 57–66.

[3] Bonzon, E.—Lagasquie-Schiex, M.-Ch.—Lang, J.—Zanuttini, B.: Boolean Games Revisited. In ECAI, 2006, pp. 265–269.

[4] Giumale, C.—Negreanu, L.: Reasoning with Fluid Qualities. 17^{th} International Conference on Control Systems and Computer Science (CSCS 17), December 2009, Vol. 2, pp. 197–203.

[5] Giumale, C.—Negreanu, L.—Muraru, M.—Popovici, M.: Modeling Ontologies for Time-Dependent Applications. Symbolic and Numeric Algorithms for Scientific Computing, International Symposium, 2010, pp. 202–208.

[6] Ågotnes, T.—van der Hoek, W.—Wooldridge, M.: Quantified Coalition Logic. Synthese, Vol. 165, 2008, No. 2, pp. 269–294.

[7] Harrenstein, P.—van der Hoek, W.—Meyer, J.-J.—Witteveen, C.: Boolean Games. Proceedings of the 8^{th} Conference on Theoretical Aspects of Rationality and Knowledge (TARK '01), San Francisco, CA, USA, 2001, Morgan Kaufmann Publishers Inc., pp. 287–298.

[8] Luo, Y.—Liu, K.—Davis, D. N.: A Multi-Agent Decision Support System for Stock Trading. Network, IEEE, Vol. 16, 2002, No. 1, pp. 20–27.

[9] Ma, Y.-L.—Li, B.-Q.: An Auction-Based Multi-Agent System Model with Variable Use-Factor. 2008 International Conference on Machine Learning and Cybernetics, July 2008, Vol. 2, pp. 899–903.

[10] Moskewicz, M. W.—Madigan, C. F.—Zhao, Y.—Zhang, L.—Malik, S.: Chaff: Engineering an Efficient Sat Solver. In Annual ACM IEEE Design Automation Conference, ACM, 2001, pp. 530–535.

[11] Osborne, M.J.—Rubinstein, A.: A Course in Game Theory. MIT Press, 1994.

[12] Popovici, M.: Using Evolution Graphs for Describing Topology-Aware Prediction Models in Large Clusters. In: Fisher, M., van der Torre, L., Dastani, M. and Governatori, G. (Eds.): Computational Logic in Multi-Agent Systems, Springer Berlin/Heidelberg, 2012, Vol. 7486, pp. 94–109.

[13] Popovici, M.—Muraru, M.—Agache, A.—Giumale, C.—Negreanu, L.—Dobre, C.: A Modeling Method and Declarative Language for Temporal Reasoning Based on Fluid Qualities. Proceedings of the 19^{th} International Conference on Conceptual Structures for Discovering Knowledge (ICCS '11), Berlin, Heidelberg, Springer-Verlag, 2011, pp. 215–228.

[14] Popovici, M.—Negreanu, L.: Strategic Behaviour in Multi-Agent Systems Able to Perform Temporal Reasoning. In IDC, 2013, pp. 211–216.

[15] Shoham, Y.—Leyton-Brown, K.: Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations. Cambridge University Press, 2009.

[16] Sistla, A. P.—Clarke, E. M.: The Complexity of Propositional Linear Temporal Logics. J. ACM, Vol. 32, 1985, No. 3, pp. 733–749.

Matei Popovici is a postdoctoral researcher at the Research Institute of University of Bucharest (ICUB) and Lecturer at the Computer Science Department of POLITEHNICA University of Bucharest, where he teaches Programming Paradigms and Algorithms & Complexity Theory courses. He received his Ph.D. degree from the same institution in 2012, with a thesis on temporal knowledge representation and reasoning. His main research interests include algorithmic game theory, logics for multiagent system verification, network verification, and artificial intelligence.

Lorina Negreanu is currently Associate Professor with the Department of Computer Science, University POLITEHNICA of Bucharest (UPB). She graduated in 1987 from the Faculty of Control and Computers, Computer Science Department of UPB and she received her Ph.D. degree from the same university in 1999. Her main research areas include software and data modelling, model checking and software specification methods. She teaches courses on object oriented programming, formal languages (B.Sc. degree in CS), software verification and validation, advanced database systems (M.Sc. degree in CS). She is coordinating the research group "Software Modelling and Formal Verification". She co-authored more than 50 research papers published in international journals and conference proceedings, 4 books, and 3 university textbooks. She was Visiting Professor at University of Maryland University College, US, and was/is member of Program Committees of several international conferences. She participated as a research team member in several national and international R & D grants, and international R & D projects.