Reasoning about Games via Temporal Logic: A Model Checking Approach

Davi Romero de Vasconcelos[1] Edward Hermann Haeusler[1]
Marcus Poggi de Aragão[1] Mario Folhadela Benevides [2]

[1]Pontifícia Universidade Católica do Rio de Janeiro, Brazil
davirv,hermann,poggi@inf.puc-rio.br

[2]Programa de Sistemas, COPPE/UFRJ, Brazil
mario@cos.ufrj.br

PUC-RioInf.MCC47/04 December, 2004

Abstract

This article shows how to use a subset of first-order CTL, namely Game Analysis Logic (GAL), in order to reason about games. A model checking algorithm for GAL is presented. Standard games and solution concepts of Game Theory are described in this context. Taking into account the strong relationship between games and Multi-Agent systems(MAS), the approach described here seems to be completely suitable for MAS formal analysis. Strategic and Coalition games are modelled and formally analyzed as case studies in order to demonstrate this approach.

Keywords: Games, Model Checking, Multi-Agent Systems and Game Theory

Resumo

Este trabalho mostra como utilizar um subconjunto de CTL de primeira ordem, chamado Game Analysis Logic (GAL), para analisar jogos. Um algoritmo para verificação automática de modelos é também apresentado. Modelos e conceitos de soluções da Teoria dos Jogos são representados neste contexto. Considerando a forte relação entre jogos e sistemas Multi-Agentes, esta abordagem parece ser completamente adequada para análise formal de sistemas multi-agentes. Estudos de casos são também apresentados.

Palavras-chave: Jogos, Verificação de Modelos, Sistemas Multi-Agentes e Teoria dos Jogos

1 Introduction

Games are abstract models of decision-making in which decision-makers (players) interact in a shared environment to accomplish their goals. Several models have been proposed to analyze a wide variety of applications in many disciplines such as Mathematics, Computer Science and even political and social sciences among others.

Game Theory[14] has its roots in the work of Von Neumann and Morgenstern[13] and use mathematics is order to model and analyze games in which the decision-makers pursue well-defined exogenous (rational behavior) and take into account their knowledge or expectations of the other players' behavior. In the last few years, many works have used Game Theory for analyzing and implementing Multi-Agent systems[17, 8].

The technique of Model Checking[4] is frequently employed in computer science to accomplish formal validation of both software and hardware[2]. Model Checking consists of achieving automatic verification and other forms of formal analysis of a system behavior. A lot of implementations are available in the literature, such as Symbolic Model Verifier(SMV)[11] and SPIN[7], some other implementations also include specific features in the modelling, UPPAAL[1] works in real-time, HYTECH[6] with hybrid automata and PRISM[16] with stochastic automata. Recently, model checking has also been used to verify proprieties in multi-agent systems[19, 20, 9].

In this article we show how to use a subset of first-order CTL, called Game Analysis Logic (GAL), in order to reason about games in which a model of GAL is a game and a GAL formula is an analysis. We also provide a model checking algorithm for GAL. In this way, we model and analyze some standard games and solution concepts of Game Theory as well as multi-agent systems. Precisely, we specify Nash Equilibrium by means of a GAL formula, in general, and verify it using our model-checker. Core is a concept in Coalition Games as important as Nash Equilibrium is for Strategic Games. Core is also represented by means of a GAL formula and plays a central role in the formal analysis of studied Coalition Games.

This work is divided into six parts: Section 2 introduces Game Analysis Logic; A model checking algorithm for GAL is presented in Section 3. Standard concepts of Game Theory are expressed in GAL in Section 4. Section 5 presents some experimental results using our algorithm. Finally, Section 6 concludes this work.

2 Game Analysis Logic (GAL)

We model and analyze games using a many-sorted modal first-order logic language, called Game Analysis Logic, that is an extension of the standard Computation Tree Logic[3]. A game is a model of GAL, called game analysis structure, and an analysis is a GAL-formula.

The games that we model are represented by a set of states and an set of actions, where:

1. A state is defined by both first-order interpretation and set of players, where:
   - The first-order interpretation is used to represent the choices and the consequences of the players decisions. For example, we can use a list to represent the history of the players choices until certain state.
   - The set of players is a subset of the players' set of the game that can decide simultaneously at a state. The other players cannot make a choice at this state. For instance, we can model games such as strategic games and coalition games, where all players are in all states, or even games as Chess or turn-based synchronous game structure, where only a single player has to make a choice at each state. Notice that we may even have some states where none of players can make a decision.
2. An action is a relation between two states e_1 and e_2, where all players in the state e_1 made the decision to move to the state e_2.

Path is a sequence of states that could be reached through the set of actions from a given state. The game behavior is characterized by its paths that can be finite or infinite. Finite paths end in a state where the game is over, while infinite ones represent the game that will never end.

Below we present the formal syntax and semantics of GAL. As usual, we call the sets of sorts, predicate symbols, functional symbols and players as a non-logic language in contrast to the logic language that contains the quantifiers and the connectives. We use the notation Υ_s, where s is a sort, to denote the set of terms of sort s. We refer to (A_k) as a set of A_k with the index k.

Definition 1 *Non-logic language of GAL:*
A non-logic language of GAL is given by $\langle S, F, P, N\rangle$, where:

1. *S is a set, called set of sorts.*
2. *F is a set, called function symbols, such that, to each functional symbol is associated a signature: $w \rightarrow s$, where $w \in S^*$ and $s \in S$.*

   A functional symbol f with signature $w \rightarrow s$ is usually denoted by: $f : w \rightarrow s$. If w is empty, then f is called constant symbol, or constant only.
3. *P is a set, called predicate symbols, such that, to each predicate symbol is associated a signature: w, where $w \in S^*$. $p : w$ denotes a predicate symbol p with signature w. If w is empty, then p is called by propositional symbol, or proposition only.*
4. *N is a set, called set of players.*

Definition 2 *Syntax of GAL:*
Given $t_1, ..., t_n$ terms of sort $s_1, ..., s_n$, t'_1 term of sort s_1, $P : s_1...s_n$ a predicate symbol, i a player, x a variable of a sort, and $\approx$ a logic symbol, the language of the GAL is generated by the following BNF definition:
$\Phi :== i \mid P(t_1, ..., t_n) \mid (t_1 \approx t'_1) \mid (\neg\Phi_1) \mid (\Phi_1 \rightarrow \Phi_2) \mid \exists x \Phi_1 \mid [\forall\bigcirc]\Phi_1 \mid \exists(\Phi_1 \mathcal{U}\ \Phi_2) \mid \forall(\Phi_1 \mathcal{U}\ \Phi_2)$.

The modalities are similar to the branching-time temporal logic CTL with the following meaning.

1. $[\forall\bigcirc]\Phi_1$ means "for all paths in the next state Φ_1".
2. $\exists(\Phi_1 \mathcal{U} \Phi_2)$ means "there is path Φ_1 until Φ_2".
3. $\forall(\Phi_1 \mathcal{U} \Phi_2)$ means "for all path Φ_1 until Φ_2".

The logical constants $\vee, \wedge, \forall x, [\forall\Diamond], [\exists\Diamond], [\forall\Box], [\exists\Box]$ and $[\exists\bigcirc]$ are given by following usual abbreviations.

- $\alpha \wedge \beta \iff \neg(\alpha \rightarrow \neg\beta)$
- $\alpha \vee \beta \iff (\neg\alpha \rightarrow \beta)$
- $\forall x \alpha(x) \iff \neg\exists x \neg\alpha(x)$
- $[\exists\bigcirc]\alpha \iff \neg[\forall\bigcirc]\neg\alpha$
- $[\forall\Diamond]\alpha \iff \forall(\top\ \mathcal{U}\ \alpha)$
- $[\exists\Diamond]\alpha \iff \exists(\top\ \mathcal{U}\ \alpha)$
- $[\forall\Box]\alpha \iff \neg\exists(\top\ \mathcal{U}\ \neg\alpha)$
- $[\exists\Box]\alpha \iff \neg\forall(\top\ \mathcal{U}\ \neg\alpha)$

Definition 3 *Game Analysis Logic Structure (GAL-Structure):*
Let $\langle S, F, P, N\rangle$ be a non-logic language of GAL. A Game Analysis Logic Structure for this language is a tuple $\mathcal{G} = \langle \mathcal{SE}, \mathcal{SE}_o, \mathcal{CA}, (\mathcal{D}_s), (\mathcal{F}_{f,e}), (\mathcal{P}_{p,e}), (N_e)\rangle$ such that:

- *$\mathcal{SE}$ is a set, called set of states.*
- *$\mathcal{SE}_o$ is a set of initial states, where $\mathcal{SE}_o \subseteq \mathcal{SE}$.*
- *$\mathcal{CA} \subseteq \mathcal{SE} \times \mathcal{SE}$, called set of actions.*
- *For each sort $s \in S$, $\mathcal{D}_s$ is a nonempty set, called domain of sort s[1].*

[1]In algebraic terminology $\mathcal{D}_s$ is a carrier for the sort s.

- *For each function symbol $f : w \to s$ of F and each state $e \in \mathcal{S}E$, $\mathcal{F}_{f,e}$ is a function such that $\mathcal{F}_{f,e} : \left(\prod_{s_k \in w} \mathcal{D}_{s_k}\right) \to \mathcal{D}_s$.*
- *For each predicate symbol $p : w$ of P and state $e \in \mathcal{S}E$, $\mathcal{P}_{p,e}$ is a relation such that $\mathcal{P}_{p,e} \subseteq \left(\prod_{s_k \in w} \mathcal{D}_{s_k}\right)$.*
- *For each state $e \in \mathcal{S}E$, N_e is a subset of N.*

A GAL-structure is finite if the set of states $\mathcal{S}E$ and the set of domains (D_s) are finite. Otherwise, it is infinite.

In order to provide the semantics of GAL, we define a valuation function for each free variable of each sort.

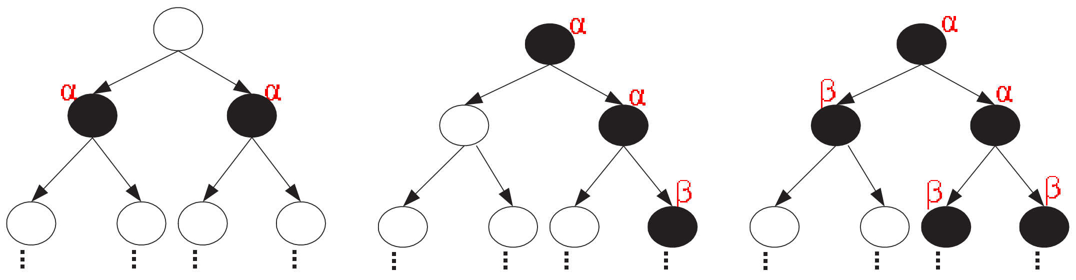
Definition 4 *Valuation:*
Let $\mathcal{G} = \langle \mathcal{S}E, \mathcal{S}E_o, \mathcal{CA}, (\mathcal{D}_s), (\mathcal{F}_{f,e}), (\mathcal{P}_{p,e}), (N_e)\rangle$ be a GAL-structure, where $s \in S, f \in F$ and $e \in \mathcal{S}E$. A valuation for a sort s in this model $\mathcal{G}$ is a mapping σ_s that assigns to each free variable x of sort s and each state e some member $\sigma_s(e, x)$ of domain $\mathcal{D}_s$ of this model.

As we use terms, we extend every function σ_s to a function $\bar{\sigma}_s$ from state and term to element of sort s that is done in a standard way. When the valuation functions are not necessary, we will omit them.

Definition 5 *GAL Semantics:*
Let $\mathcal{G} = \langle \mathcal{S}E, \mathcal{S}E_o, \mathcal{CA}, (\mathcal{D}_s), (\mathcal{F}_{f,e}), (\mathcal{P}_{p,e}), (N_e)\rangle$ be a GAL-structure and (σ_s) be valuation functions, where $s \in S, f \in F$ and $e \in \mathcal{S}E$. We write $\mathcal{G}, (\sigma_s) \models_e \alpha$ to indicate that the state e satisfies the formula α in the structure $\mathcal{G}$ with valuation functions (σ_s). The formal definition of satisfaction ($\models$) proceeds as follows:

- $\mathcal{G}, (\sigma_s) \models_e i \iff i \in N_e$.
- $\mathcal{G}, (\sigma_s) \models_e p(t_1^{s_1}, ..., t_n^{s_n}) \iff \langle \bar{\sigma}_{s_1}(e, t_1^{s_1}), ..., \bar{\sigma}_{s_n}(e, t_n^{s_n})\rangle \in \mathcal{P}_{p,e}$
- $\mathcal{G}, (\sigma_s) \models_e (t_1^s \approx t_2^s) \iff \bar{\sigma}_s(e, t_1^s) = \bar{\sigma}_s(e, t_2^s)$
- $\mathcal{G}, (\sigma_s) \models_e \neg\alpha \iff$ *NOT* $\mathcal{G}, (\sigma_s) \models_e \alpha$
- $\mathcal{G}, (\sigma_s) \models_e (\alpha \to \beta) \iff$ *IF* $\mathcal{G}, (\sigma_s) \models_e \alpha$ *THEN* $\mathcal{G}, (\sigma_s) \models_e \beta$
- $\mathcal{G}, (\sigma_s) \models_e [\forall\bigcirc]\alpha \iff \forall e' \in \mathcal{S}E$ *such that* $\langle e, e'\rangle \in \mathcal{CA}$, $\mathcal{G}, (\sigma_s) \models_{e'} \alpha$ *(see figure 1.a).*
- $\mathcal{G}, (\sigma_s) \models_e \exists(\alpha\ \mathcal{U}\ \beta) \iff \exists\pi(e) = (e_0 e_1 e_2 ... e_i)$, $i \in I$ *and* $I \subseteq \mathbb{N}$ *such that* $\exists k[k \geq 0$, $\mathcal{G}, (\sigma_s) \models_{e_k} \beta$, $\forall j[0 \leq j < k$, $\mathcal{G}, (\sigma_s) \models_{e_j} \alpha]]$ *(see figure 1.b).*
- $\mathcal{G}, (\sigma_s) \models_e \forall(\alpha\ \mathcal{U}\ \beta) \iff \forall\pi(e) = (e_0 e_1 e_2 ... e_i)$, $i \in I$ *and* $I \subseteq \mathbb{N}$ *such that* $\exists k[k \geq 0, \mathcal{G}, (\sigma_s) \models_{e_k} \beta, \forall j[0 \leq j < k, \mathcal{G}, (\sigma_s) \models_{e_j} \alpha]]$ *(see figure 1.c);*
- $\mathcal{G}, (\sigma_s, \sigma_{s_k}) \models_e \forall x^{s_k}\alpha \iff$ *for every* $d \in \mathcal{D}_{s_k}$, *we have* $\mathcal{G}, (\sigma_s, \sigma_{s_k}(x|d)) \models_e \alpha$, *where* $\sigma_{s_k}(x|d)$ *is the function which is exactly like* σ_{s_k} *except for one thing: At the variable x it assumes the value d. This can be expressed by the equation:*

$$\sigma_s(x|d)(e, y) = \begin{cases} \sigma_s(e, y), & \textit{if } y \neq x \\ d, & \textit{if } y = x \end{cases}$$



(a) - $[\forall\bigcirc]\alpha$ (b) - $\exists(\alpha\,\mathcal{U}\beta)$ (c) - $\forall(\alpha\,\mathcal{U}\beta)$

Figure 1: Modal Connectives of *GAL*

3 GAL Model Checking

Let $\mathcal{G} = \langle \mathcal{SE}, \mathcal{SE}_o, \mathcal{CA}, (\mathcal{D}_s), (\mathcal{F}_{f,e}), (\mathcal{P}_{p,e}), (N_e)\rangle$ be a finite GAL-structure with the non-logic language $\langle S, F, P, N\rangle$, (σ_s) be valuation functions and α be a GAL-formula. The GAL model checking problem is to find the set of states that satisfies the formula α.

$$\{e \in \mathcal{SE} \mid \mathcal{G}, (\sigma_s) \models_e \alpha\}$$

The algorithm for solving the GAL model checking problem uses an explicit representation of the GAL-structure as a labelled, directed graph. The nodes represent the states $\mathcal{SE}$, the arcs in the graph provide the set of actions $\mathcal{CA}$ and the labels associated with the nodes describe both the set of players N_e and the first-order interpretation (the set of the interpreted functions $(\mathcal{F}_{f,e})$ and the set of the interpreted predicates $(\mathcal{P}_{p,e})$). The algorithm also uses the functions $\mathcal{D} : S \rightarrow \mathcal{D}_s$, $\mathcal{N} : \mathcal{SE} \rightarrow N_e$, $\mathcal{F} : F \times \mathcal{SE} \rightarrow \mathcal{F}_{f,e}$ and $\mathcal{P} : P \times \mathcal{SE} \rightarrow \mathcal{P}_{p,e}$ in order to provide an implicity representation of the set of domains $(\mathcal{D}_s)$, the set of players N_e,the functions $(\mathcal{F}_{f,e})$ and the relations $(\mathcal{P}_{p,e})$, respectively. Thus, we only evaluate them on demand.

The algorithm is similar to CTL model-checking algorithm[3] that operates by labelling each state $e \in \mathcal{SE}$ with the set of *labels*(e) of subformulas of α which are true in e. The algorithm starts with the set *labels*(e) as the empty set[2] and then goes by a series of steps (the number of operators in α). At each step k, subformulas with $k-1$ nested GAL operators are processed. When a formula is processed, it is added to the labelling of the state in which it is true. Thus, $\mathcal{G}, (\sigma_s) \models_e \alpha \iff \alpha \in labels(e)$.

As GAL-formulas are represented in terms of i, $P(t_1, ..., t_n)$, $(t_1 \approx t'_1)$, $(\neg\Phi_1)$, $(\Phi_1 \rightarrow \Phi_2)$, $\exists x\Phi_1$, $[\forall\bigcirc]\Phi_1$, $\exists(\Phi_1\mathcal{U}\ \Phi_2)$, $\forall(\Phi_1\mathcal{U}\ \Phi_2)$, it is sufficient to handle these cases. The cases $(\neg\Phi_1)$, $(\Phi_1 \rightarrow \Phi_2)$, $[\forall\bigcirc]\Phi_1$, $\exists(\Phi_1\mathcal{U}\ \Phi_2)$ and $\forall(\Phi_1\mathcal{U}\ \Phi_2)$ are similar to CTL model-checking algorithm and we do not present here (see [11] for more details). Below we present and give the complexity time of the other procedures.

- Case i: The procedure *verifyPlayer* (see below) labels all states $e \in \mathcal{SE}$ with the player i if the player i belongs to the set of players in e. This procedure requires time $O(|\mathcal{SE}|)$.

```
procedure verifyPlayer(i)
    for all $e \in \mathcal{SE}$
        if $i \in \mathcal{N}(e)$
            $labels(e) := labels(e) \cup \{i\}$
```

- Case $p(t_1^{s_1}, ..., t_n^{s_n})$: The procedure *verifyInterpretedPredicate* (see below) labels all states $e \in \mathcal{SE}$ in which the interpretation of the predicate p with the interpretation of terms $t_1^{s_1}, ..., t_n^{s_n}$ is true in e. We use the notation $\bar{\sigma}_{s_1}(e, t_1^{s_1})$ as the evaluation function that

[2]The CTL model-checking algorithm starts the set of labels(e) as the set of propositions in e. In our algorithm we just evaluate the predicates and functions on demand.

interprets the term $t_1^{s_1}$ in the state e. This procedure requires time $O((|\bar{\sigma}_{s_1}(e,t_1^{s_1})| + ... + |\bar{\sigma}_{s_n}(e,t_n^{s_n})|) \times |\mathcal{S}E|)$. Notice that the evaluation of the predicate and the terms are done in all states and the complexity time of them could not be polynomial.

```
procedure verifyInterpretedPredicate($p(t_1^{s_1},...,t_n^{s_n})$)
    for all $e \in \mathcal{S}E$
        if $\langle \bar{\sigma}_{s_1}(e,t_1^{s_1}),...,\bar{\sigma}_{s_n}(e,t_n^{s_n})\rangle \in \mathcal{P}(p,e)$
            $labels(e) := labels(e) \cup \{p(t_1^{s_1},...,t_n^{s_n})\}$
```

- Case $t_1^{s_1} \approx t_2^{s_1}$: The procedure *verifyEquality* (see below) labels all state $e \in \mathcal{S}E$ in which the interpretation of the terms $t_1^{s_1}$ and $t_2^{s_1}$ are equal. We also use the notation $\bar{\sigma}_{s_1}(e,t_1^{s_1})$ and the complexity time is $O((|\bar{\sigma}_{s_1}(e,t_1^{s_1})| + |\bar{\sigma}_{s_n}(e,t_2^{s_1})|) \times |\mathcal{S}E|)$

```
procedure verifyEquality($t_1^{s_1} \approx t_2^{s_1}$)
    for all $e \in \mathcal{S}E$
        if $\bar{\sigma}_{s_1}(e,t_1^{s_1}) = \bar{\sigma}_{s_1}(e,t_2^{s_1})$
            $labels(e) := labels(e) \cup \{t_1^{s_1} \approx t_2^{s_1}\}$
```

- The procedure *verifyExists* (see below) labels all states $e \in$. We use the notation $\alpha[x \leftarrow d]$ as a function that substitutes all occurrence of x by d in α. This procedure requires $O(|\mathcal{D}_{s_k}| \times |\mathcal{S}E|)$

```
procedure verifyExists($\exists x^{s_k}\alpha$)
    for all $d \in \mathcal{D}(s_k)$
        $T := \{e \mid \alpha[x \leftarrow d] \in Labels(e)\}$
        for all $e \in T$
            if $\exists x^{s_k}\alpha \notin labels(e)$
                $labels(e) := labels(e) \cup \{\exists x^{s_k}\alpha\}$
```

4 Game Theory in Game Analysis Logic

We can model both the standard models and the standard solution concepts of Game Theory using GAL. In this section we show that the standard models are related to as GAL-structures and the standard solution concepts are related to as GAL-formulas.

The models of Game Theory are mainly divided into two groups: noncooperative models in which the sets of possible actions of players are primitives; and cooperative models in which the sets of possible joint actions are the primitives. Below we model both the Strategic Game and Coalition Game with Transferable Payoff[14] that are the standard models for cooperative and noncooperative games respectively, as GAL-structures. We also express the standard solution concepts for both, that are Nash Equilibrium[12] and Core[13], as GAL-formulas. An example for each group is also provided.

A strategic game, that is also called game in normal form, is a model that consists of a set of players N, for each player $i \in N$, a set of available actions A_i and a preference relation on a set of action profiles. We refer to an action profile $a = (a_i)$, for each player i, as an outcome, and denote the set $\prod_{i \in N} A_i$ of outcomes by A. We also refer to all players other than some given player i as $-i$. We write (a_i, a_{-i}) for the profile $(a_1, \ldots, a_i, \ldots, a_n)$, where n is the number of players.

Definition 6 *Strategic Game:*
A strategic game is a tuple $\langle N, (A_i), (\succeq_i)\rangle$, where

- *N is a finite set, called the set of players.*
- *for each player $i \in N$,*

– *A_i is a nonempty set, called the set of actions available to player i.*

– *$\succeq_i$ is a preference relation on $A = \prod_{j \in N} A_j$, called the preference relation of player i.*

Sometimes the preference relation $\succeq_i$ of player i in a strategic game can be represented by a payoff function $u_i : A \to \mathbb{R}$ (also called a utility function), in the sense that $u_i(a) \geq u_i(b)$ whenever $a \succeq_i b$. In such case we denote the game by $\langle N, (A_i), (u_i)\rangle$ rather than $\langle N, (A_i), (\succeq_i)\rangle$.

Each strategic game can be represented as a GAL-structure in which each outcome of a strategic game is a state of GAL-structure and as this game has perfect information each state knows all states. A constant for each player is used to model the player's choice and its interpretation changes depending on the outcome. The utility functions are defined as their owns. The relation $\geq$ is defined as constant for every state and has the usual meaning. All players are in all states. To summarize, a strategic game as a GAL-structure is the tuple $\langle A, A, A \times A, (A_i, \mathbb{R}), (a_{i,a}, u_i)), \geq, N\rangle$ with non-logic language $\langle (A_i, \mathbb{R})\ , (a_i :\to A_i, u_i : A \to \mathbb{R})\ , \geq, N\rangle$, where A is the set of outcomes, i is a player and for each $a = \langle a_1, \ldots, a_i, \ldots, a_n\rangle \in A$, $a_{i,a} = a_i$.

A Nash equilibrium is a profile of strategies such that each player's strategy is an optimal response to the other players' strategies.

Definition 7 *Nash Equilibrium*

A Nash Equilibrium of a strategic game $\langle N, (A_i), (u_i)\rangle$ is a profile $a^* \in A$ of actions such that for each player $i \in N$ we have

$$u_i(a^*_{-i}, a^*_i) \geq u_i(a^*_{-i}, a_i) \text{ for all } a_i \in A_i$$

We express Nash Equilibrium as a GAL-formula that holds for each state in which its outcome representation (interpretations of constants) is a Nash Equilibrium.

Let $\langle A, A, A \times A, (A_i, \mathbb{R}),\ (a_{i,a}, u_i)), \geq, N\rangle$ be a GAL-structure with non-logic language $\langle (A_i, \mathbb{R})\ , (a_i :\to A_i, u_i : A \to \mathbb{R})\ , \geq, N\rangle$ and (v_{A_i}) be variables of sort A_i, where $i \in N$, $a \in A$ and n is the number of players of N. A Nash Equilibrium formula is defined as follows.

$$\exists v_{A_1} \ldots \exists v_{A_n} ((\bigwedge_{i \in N} v_{A_i} = a_i) \wedge$$

$$[\forall \bigcirc] \bigwedge_{i \in N} ((\bigwedge_{j \in N, i \neq j} v_{A_j} = a_j) \to u_i(v_{A_1}, ..., v_{A_n}) \geq u_i(v_{A_1}, ..., a_i, ..., v_{A_n})))$$

The complexity time of finding the Nash Equilibria of a strategic game $\langle N, (A_i), (u_i)\rangle$ is $O(\prod_{i \in N} |A_i|^2)$, where $|A_i|$ is the number of actions for the player i.

Exemple 1 *(Bach or Stravinsky[3]) Two people want to go out together to a concert of music by either Bach or Stravinsky. Their main concern is to go out together, but one person prefers Bach and the other person prefers Stravinsky. The formal definition for this game is $\langle\{1, 2\}, (A_1, A_2), (u_1, u_2)\rangle$ where*

- *$A_1 = A_2 = \{B, S\}$*
- *$u_1(B, B) = 2$, $u_1(B, S) = 0$, $u_1(S, B) = 0$, $u_1(S, S) = 1$, $u_2(\langle B, B\rangle) = 1$, $u_2(B, S) = 0$, $u_2(S, B) = 0$, $u_2(S, S) = 2$.*

This game has two Nash Equilibria: Both players choose Bach (B,B); Both players choose Stravinsky (S,S). We represent it in the figure 2.a. The actions for player 1 and 2 are identified with the rows and columns respectively. The players' payoffs are represented by the two numbers in the box. For example, when player 1 and 2 choose B and B, the payoffs are 2 and 1 respectively.

Below we present this game and the Nash Equilibrium using GAL. Figure 2.b represents a GAL-structure for this game, in which each outcome is a state and the constants are interpreted over the outcomes of this game. For instance, the outcome (B, B), the constants a_1 and a_2 are identified as the state BB, B and B respectively. And the Nash Equilibrium formula is true just for the states BB and SS that are the Nash Equilibria.

[3]This game is often referred as Battle of Sex when we consider the two people as a man and a woman

- *Bach or Stravinsky as GAL-Structure.*

  The non-language logic is $\langle\{A_1, A_2, \mathbb{R}\}, \{a_1 :\rightarrow A_1, a_2 :\rightarrow A_2, u_1 : A_1 \times A_2 \rightarrow \mathbb{R}, u_2 : A_1 \times A_2 \rightarrow \mathbb{R}\}, \geq, \{1,2\}\rangle$ *and the Bach or Stravinsky game is* $\mathcal{G} = \langle A, A, \mathcal{CA}, (A_1, A_2, \mathbb{R}), (a_{1,e}, a_{2,e}, u_1, u_2), \geq, (N_e)\rangle$*, where*

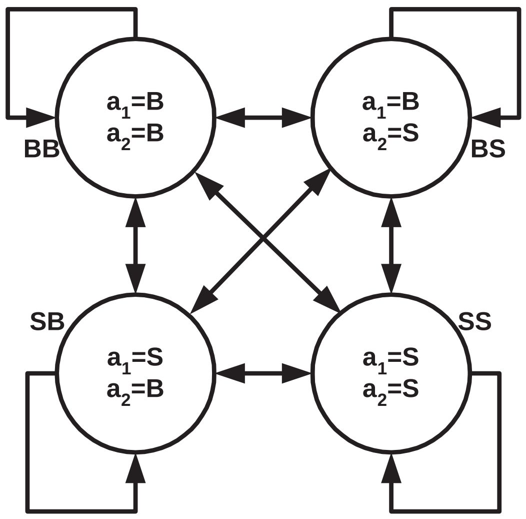
  - $A = \{BB, BS, SB, SS\}$
  - $\mathcal{CA} = \{\langle BB, BB\rangle, \langle BB, BS\rangle, \langle BB, SB\rangle, \langle BB, SS\rangle, \langle BS, BB\rangle, \langle BS, BS\rangle, \langle BS, SB\rangle, \langle BS, SS\rangle, \langle SB, BB\rangle, \langle SB, BS\rangle, \langle SB, SB\rangle, \langle SB, SS\rangle, \langle SS, BB\rangle, \langle SS, BS\rangle, \langle SS, SB\rangle, \langle SS, SS\rangle\}$
  - $A_1 = A_2 = \{B, S\}$
  - $u_1(B,B) = 2$, $u_1(B,S) = 0$, $u_1(S,B) = 0$, $u_1(S,S) = 1$, $u_2(\langle B,B\rangle) = 1$, $u_2(B,S) = 0$, $u_2(S,B) = 0$, $u_2(S,S) = 2$.
  - $a_{1,BB} = B, a_{2,BB} = B$, $a_{1,BS} = B, a_{2,BS} = S$, $a_{1,SB} = S, a_{2,SB} = B$, $a_{1,SS} = S$ *and* $a_{2,SS} = SS$.
  - $N_{BB}, N_{BS}, N_{SB}, N_{SS} = \{1,2\}$.

- *Nash Equilibrium formula*

$$\exists v_{A_1} \exists v_{A_2}((v_{A_1} = a_1 \wedge v_{A_2} = a_2)$$
$$\wedge[\forall\bigcirc]((v_{A_2} = a_2 \rightarrow (u_1(v_{A_1}, v_{A_2}) \geq u_1(a_1, v_{A_2}))$$
$$\wedge(v_{A_1} = a_1 \rightarrow (u_2(v_{A_1}, v_{A_2}) \geq u_2(v_{A_1}, a_2)))))$$

		Player 2	
		B	S
Player 1	B	2 , 1	0 , 0
	S	0 , 0	1 , 2

(a) - Strategic game



(b) - GAL-structure

Figure 2: Bach or Stravinsky.

A coalition game with transferable payoff is a model that consists of a set of players N and for each possible coalition (a subset of N) associates a real number, interpreted as the payoff that is available to the coalition; There are no restriction on how this payoff may be divided among the members of the coalition.

Definition 8 *Coalition Game with Transferable Payoff*
A coalition game with transferable payoff consists of $\langle N, v\rangle$*, where*

- *a set of players* N

- *a function v that assigns a real number for each nonempty subset $S \subseteq N$ (coalition)*

Let $\langle N, v\rangle$ be a coalition game with transferable payoff. For any profile $x \in \mathbb{R}^N$ and a coalition S we let[4] $c(x,S) = \sum_{i \in S} x_i$, where x_i is the real number for the player i in x. A vector $(x_i)_{i\in S}$ is a **S-feasible payoff vector** if $c(x,S) = v(S)$. We refer to a set S_f as the set of S-feasible payoff vector.

A coalition game with transferable payoff can be represented as a GAL-structure in which each N-feasible payoff vector is a state of the GAL-structure. A constant x is used to model the N-feasible vector and its interpretation changes depending on the N-feasible vector. We use 3 sorts $N_f, \mathit{Coalition}, \mathbb{R}$ and interpret them respectively as: the set N_f that is the set of N-feasible vectors; the set 2^N that is the set of coalitions; and the set of real numbers. The function v and c are defined as their owns. To summarize, a coalition game as a GAL-structure is the tuple $\langle N_f, N_f, \varnothing, (\mathbb{R}, N_f, 2^N), (x_{n_f}, v, c), \geq, N\rangle$ with non-logic language $\langle(\mathbb{R}, N_f, \mathit{Coalition}), (x :\rightarrow N_f, v : \mathit{Coalition} \rightarrow \mathbb{R}, c : N_f \times \mathit{Coalition} \rightarrow \mathbb{R}), \geq\rangle$

The main solution concept for a coalition game with transferable payoff is the core that requires that no set of players is able to break away and take a joint action that makes all of them better off.

Definition 9 *Core of Coalition Game with Transferable Payoff*
The core for a coalition game with transferable payoff $\langle N, v\rangle$ is the following set:

$$\{x |\ x \text{ is a N-feasible vector and } c(x,S) \geq v(S) \text{ for all } S \subset N\}$$

We express the core as a GAL-formula such that it holds for each state in which its N-feasible vector (interpretation of x) is a core.

Let $\langle N_f, N_f, \varnothing, (\mathbb{R}, N_f, 2^N), (x_{n_f}, v, c), \geq, N\rangle$ be a with non-logic language $\langle(\mathbb{R}, N_f, \mathit{Coalition}), (x :\rightarrow N_f, v : \mathit{Coalition} \rightarrow \mathbb{R}, c : N_f \times \mathit{Coalition} \rightarrow \mathbb{R}), \geq\rangle$, and S be a variable of sort $\mathit{Coalition}$. A core formula is defined as follows.

$$\forall S(c(x,S) \geq v(S))$$

The complexity time of finding the core of the coalition game with transferable payoff $< N, v >$ is $O(|N_f| \times |2^N|)$, where $|N_f|$ is the set of N-feasible vectors and $|2^N| - 1$ is the number of possible coalitions. If we only consider integer solutions to the core, then the number of elements of N_f is $\frac{(p+n-1)!}{p! * (n-1)!}$, where n is the number of players and p is the value of $v(N)$.

Exemple 2 *Let $\langle N, v\rangle$ be a coalition game with transferable payoff in which*

- $N = \{A, B, C, D\}$
- $v(\{A, B, C, D\}) = 35$,
- $v(\{A\}) = 10, v(\{B\}) = 8, v(\{C\}) = 5, v(\{D\}) = 7$
- $v(\{A, B\}) = 20, v(\{A, C\}) = 17, v(\{A, D\}) = 18, v(\{B, C\}) = 15, v(\{B, D\}) = 12, v(\{C, D\}) = 13$
- $v(\{A, B, C\}) = 25, v(\{A, B, D\}) = 25, v(\{A, C, D\}) = 23, v(\{B, C, D\}) = 22$

Both n-feasible vector $x = (13, 9, 6, 7)$ and $y = (10, 10, 8, 7)$ belong to the core solution.

As the core may be empty or even may be a huge set of N-feasible vector, other criteria should be defined to provide a solution. Below we list some of other criteria that may be adopt as well as GAL formulas to represent these criteria.

[4]In order to avoid confusing when we refer to a profile x or a function x, we use the notation c(x,S) instead of the standard notation x(S) in the literature.

- If the core was empty, then how much (in percentage) we have to subside the coalition to have non-empty core?

$$(\neg\forall S((C(x,S)) \geq v(S))) \rightarrow (\exists m\forall S(((1+m)*C(x,S)) \geq v(S)))$$

- The agent A (in the example above) only accepts to be a member of the coalition if his worth was at least 13.

$$(\forall S((C(x,S)) \geq v(S))) \wedge (C(x,\{A\}) \geq 13)$$

  Thus, the n-feasible vector $y = (10, 10, 8, 7)$ does not belong to this solution.

We can consider a coalition game regarding a sequence of the decision problems by the coalition game with transferable payoff. We refer to this game as evolving coalition game. An evolving coalition game consists of a set of players N and a sequence of functions $(v_0, ..., v_k)$ in which each function v_i associates a real number for each possible coalition. We refer N_{f_i} to as the set of N-feasible vectors of the function v_i.

This game has the same non-logic language of coalition game and the interpretation is also similar. The sort N_f is interpreted as the set of all possible N-feasible vectors for all functions v_i and the set of action regards the sequential of the functions v_i. To summarize, the formal definition as follows.

Definition 10 *An evolving coalition game is a tuple* $\langle N_f, N_{f_0}, A, (\mathbb{R}, N_f, 2^N), (x_{n_{f_i}}, v_i, c), \geq, N\rangle$ *with non-logic language* $\langle(\mathbb{R}, N_f, Coalition), (x :\rightarrow N_f, v : Coalition \rightarrow \mathbb{R}, c : N_f \times Coalition \rightarrow \mathbb{R}), \geq\rangle$*, such that*

- $N_f = \bigcup N_{f_i}$*, where* N_{f_i} *is the set of N-feasible vectors of the function* v_i*.*
- *A is the set of actions such that for all* $0 \leq i < k$ $\langle n_i, n_{i+1}\rangle \in A$ *and* $n_i \in N_i$*.*
- $x_{n_{f_i}}$ *is the interpretation of the constant x on the state* $n_{f_i} \in N_{f_i}$*.*

Is the solution concept of the core for this game enough? As this game regards about sequence of functions, it seems clear that the concept of core should be regarding of this sequence. So, a solution should be a sequence of n-feasible vectors. The following GAL-formula represents this meaning.

$$[\exists\Box]\forall S(c(x,S) \geq v(S))$$

We can also consider an extensive coalition game that regards to a tree of functions instead of sequence. The definition is similar to an evolving coalition games. And the solution concept should regard all tree branches.

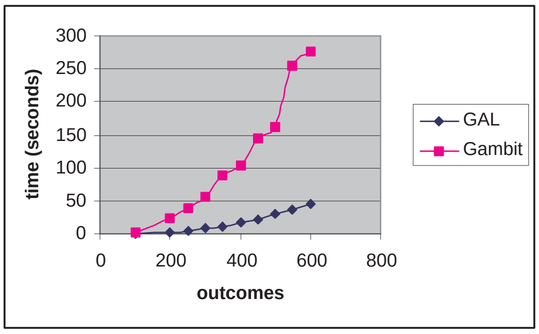
$$[\forall\Box][\exists\Box]\forall S(c(x,S) \geq v(S))$$

5 Experimental Results

In this section we show the performance of the GAL model-checking algorithm against other algorithms. The algorithm was written in Java and the experiments were executed on a 2.4GHz Celeron with 512 MBytes of RAM, running Windows XP Home Edition.

In [21, 22] is proposed a metalanguage to describe games, called *RollGame*, as well as a translation into the input language of the well-known model-checker SMV[4] in order to reason about games. Tic-Tac-Toe game is used to illustrate how the actions of one of the players follows a certain strategy while the actions of the remaining players spread all possible moves. It is also shown that this strategy never reaches a losing position in the game. We also model this game with the same strategy using our algorithm and the performance of verifying the strategy was much better in our algorithm (0.001 seconds) than using SMV model-checker (54.719 seconds).

Several algorithms for the problem of finding a Nash Equilibrium are proposed in the literature(see [10] for a survey). Most of them compute a randomized Nash Equilibrium. However, the complexity time is still unknown[15]. Gambit[18] is the best-known game theory software that implements most of all algorithms. We use both Gambit (with its *EnumPureSolve* method) and our algorithm in order to compute the pure Nash Equilibria of some games (see figure below).



As the problem of finding a Nash Equilibrium, the problem of finding the core is also a hard task. In [5] is shown that even checking if a N-feasible vector is in the core is Co-NP Complete. We use our algorithm in order to find the core when we just consider integer solutions. The table below presents the integer solution of the core for the example in the previous section.

Agent A	Agent B	Agent C	Agent D
10	10	7	8
11	9	6	9
11	9	7	8
11	10	6	8
11	10	7	7
11	11	6	7
12	8	7	8
12	8	8	7
12	9	6	8
12	9	7	7
12	10	5	8
12	8	7	8
13	8	7	7
13	9	6	7

The time required for finding the core was 2.144 seconds. We also executed finding the core with the restriction for the agent A (his worth is at least 13) and the time required was 2.274 seconds. And only the vectors $(13, 8, 7, 7)$ and $(13, 9, 6, 7)$ are in this solution.

6 Conclusion

In this work, we have presented a first-order modal logic (GAL) to model and analyze games. We have also provided a model checking algorithm for GAL to achieve automatic verification. We specified in GAL the main concepts of Strategic and Coalition games, namely, Nash Equilibrium and Core. Using the implementation of this algorithm, we performed case studies that seem to demonstrate that this approach to formal analysis of Multi-Agent systems (MAS) by means of a game model is promising. It is worth mention that if any property of MAS can be regarded as a game property, then GAL is completely suitable for MAS formal analysis. Besides the standard Core concept, as section 4 describes, many of its variations can be taken into account.

References

[1] J. Bengtsson, K. G. Larsen, F. Larsson, P. Pettersson, and Wang Yi. UPPAAL - a tool suite for automatic verification of real-time systems. In *Hybrid Systems*, pages 232–243, 1995.

[2] J. R. Burck, E. M. Clarke, and D. E. Long. Symbolic model checking with partitioned transition relations. *In A. Halaas and P. B. Denyer, eds., proceedings of the 1991 International Conference on VLSI*, pages 49–58, August 1991.

[3] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching-time logic. In *Workshop on Logic of Programs*, pages 52–71. Springer, may 1981.

[4] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking.* MIT Press, 1999.

[5] V. Conitzer and T. Sandholm. Computing shapley values, manipulating value division schemes, and checking core membership in multi-issue domains. In *AAAI*, pages 219–225, 2004.

[6] T. A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. A user guide to hytech. In *Tools and Algorithms for Construction and Analysis of Systems*, pages 41–71, 1995.

[7] G. J. Holzmann. The SPIN model checker. *IEEE Transaction on Software Engineering*, 23(5):279–295, May 1997.

[8] S. Kraus. Negotiation and cooperation in multi-agent environments. *Artificial Intelligence*, 94(1-2):79–97, 1997.

[9] A. Lomuscio M. Kacprzak and W. Penczek. Verification of multiagent systems via unbounded model checking. In *the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS04)*, page 749 753, 2004.

[10] R. McKelvey and A. McLennan. Computation of equilibria in finite games. In *Handbook of Computational Economics*, 1996.

[11] K. McMillan. *Symbolic Model Checking.* PhD thesis, Kluwer Academic Publishers, 1993.

[12] J. F. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences of the United States of American 36*, pages 48–49, 1950.

[13] J. Von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior.* John Wiley and Sons, 1944.

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

[15] C. Papadimitriou. Algorithms, games, and the internet. In *STOC-01*, page 749 753, 2001.

[16] D. Parker. *Implementation of Symbolic Model Checking for Probabilistic Systems.* PhD thesis, University of Birmingham, August 2002.

[17] S. Parsons and M. Wooldridge. Game theory and decision theory in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 5(3):243–254, 2002.

[18] A. McLennan R. McKelvey and T. Turocy. *Gambit: Software tools for game theory, version 0.97.0.5*, 2003.

[19] F. Raimondi and A. Lomuscio. Verification of multiagent systems via ordered binary decision diagrams: an algorithm and its implementation. In *the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS04)*, page 749 753, 2004.

[20] W. van der Hoek and M. Wooldridge. Tractable multiagent planning for epistemic goals. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 1167–1174. ACM Press, 2002.

[21] D. R. Vasconcelos. Análise de estratégia utilizando verificação formal de modelos. Master's thesis, PUC-Rio, 2003.

[22] D. R. Vasconcelos and E. H. Haeusler. A logic view of playing games. In *4th Congress of Logic Applied to Technology LAPTEC2003*, volume 101 Frontiers in Artificial Intelligence and Applications, pages 67–80. IOS Press, 2003.