EPTCS 112

Proceedings of the

1st International Workshop on

Strategic Reasoning

Rome, Italy, March 16-17, 2013

Edited by: Fabio Mogavero, Aniello Murano and Moshe Y. Vardi

Published: 1st March 2013
DOI: 10.4204/EPTCS.112
ISSN: 2075-2180
Open Publishing Association

Preface

This volume contains the proceedings of the First International Workshop on Strategic Reasoning 2013 (SR 2013), held in Rome (Italy), March 16-17, 2013.

The SR workshop aims to bring together researchers, possibly with different backgrounds, working on various aspects of strategic reasoning in computer science, both from a theoretical and a practical point of view.

Strategic reasoning is one of the most active research area in multi-agent system domain. The literature in this field is extensive and provides a plethora of logics for modeling strategic reasoning. Theoretical results are now being used in many exciting domains, including software tools for information system security, robot teams with sophisticated adaptive strategies, and automatic players capable of beating expert human adversary, just to cite a few. All these examples share the challenge of developing novel theories and tools for agent-based reasoning that takes into account the behavior of adversaries.

This year SR has hosted four invited talks:

- Breaking the O(n*m) Barrier for Büchi Games and Probabilistic Verification
  Krishnendu Chatterjee (IST Austria)
- Model Checking Systems against Epistemic Specifications
  Alessio R. Lomuscio (Imperial College London)
- Looking at Mean-Payoff and Total-Payoff through Windows
  Jean-Francois Raskin (Université Libre de Bruxelles)
- Bad Equilibria (and what to do about them)
  Michael Wooldridge (University of Oxford)

The program committee also selected 13 papers among the 23 contributions submitted. Contributions were selected according to quality and relevance to the topics of the workshop.

We would like to acknowledge the people and institutions, which contributed to the success of this edition of SR. We thank the organizers of the European Joint Conferences on Theory and Practice of Software (ETAPS 2013) for giving us the opportunity to host SR 2013. Many thanks go to all the Program Committee members and the additional reviewers for their excellent work, the fruitful discussions and the active participation during the reviewing process. We also thank Giuseppe Perelli and Loredana Sorrentino for their great work as members of the Organizing Committee. We would like to acknowledge the EasyChair organization for supporting all tasks related to the selection of contributions, and both EPTCS and arXiv for hosting the proceedings. We gratefully acknowledge the financial support to SR 2013 by ExCAPE - an NSF-funded Expeditions Project in Computer Augmented Program Engineering. Finally, we acknowledge the patronage from the Department of Electrical Engineering and Information Technology of the Università degli Studi di Napoli Federico II.

Rome, March 2013
Fabio Mogavero, Aniello Murano, and Moshe Y. Vardi

F. Mogavero, A. Murano, and M.Y. Vardi (Eds.):
1st Workshop on Strategic Reasoning 2013 (SR'13)
EPTCS 112, 2013, pp. i–ii, doi:10.4204/EPTCS.112.0

Program Co-Chair

- Fabio Mogavero, Università degli Studi di Napoli Federico II, Napoli, Italy
- Aniello Murano, Università degli Studi di Napoli Federico II, Napoli, Italy
- Moshe Y. Vardi, Rice University, Huston, Texas, USA

Program Committee

- Thomas Ågotnes, University of Bergen, Bergen, Norway
- Dietmar Berwanger, CNRS de Cachan, Cachan, France
- Valentin Goranko, Technical University of Denmark, Lyngby, Denmark
- Joseph Y. Halpern, Cornell University, Ithaca, New York, USA
- Wojtek Jamroga, University of Luxembourg, Luxembourg City, Luxembourg
- Orna Kupferman, Hebrew University, Jerusalem, Israel
- Nicolas Markey, CNRS de Cachan, Cachan, France
- Fabio Mogavero, Università degli Studi di Napoli Federico II, Napoli, Italy
- Aniello Murano, Università degli Studi di Napoli Federico II, Napoli, Italy
- Ramaswamy Ramanujam, Institute of Mathematical Sciences, Chennai, India
- Wolfgang Thomas, RWTH Aachen, Aachen, Germany
- Moshe Y. Vardi, Rice University, Huston, Texas, USA

Organizing Committee

- Fabio Mogavero, Università degli Studi di Napoli Federico II, Napoli, Italy
- Aniello Murano, Università degli Studi di Napoli Federico II, Napoli, Italy
- Giuseppe Perelli, Università degli Studi di Napoli Federico II, Napoli, Italy
- Loredana Sorrentino, Università degli Studi di Napoli Federico II, Napoli, Italy

Additional Referees

Benjamin Aminof, Massimo Benerecetti, Thomas Brihaye, Sjur Kristoffer Dyrkolbotn, Marco Faella, Lukasz Kaiser, Prateek Karandikar, Martin Lange, Erik Parmann, Madhusudan Parthasarathy, Soumya Paul, Truls Pedersen, Giuseppe Perelli, Nir Piterman, Luigi Sauro, Olivier Serre, Nicolas Troquard, Yi Wang.

Table of Contents

Breaking the $O(n \cdot m)$ Barrier for Büchi Games and Probabilistic Verification

Invited Talk

Krishnendu Chatterjee
IST Austria

Turn-based Büchi games and maximal end-component decomposition are two classic graph theoretic problems that are core algorithmic problems in synthesis and verification of probabilistic systems. Moreover, many other problems on graph games reduce to them, and as an example we will first describe how analysis of reachability objectives in concurrent games reduces to Büchi games. We will present recent results that break the O(n*m) barrier for Büchi games, and show how the same techniques break the barrier for maximal end-component decomposition.

F. Mogavero, A. Murano, and M.Y. Vardi (Eds.):
1st Workshop on Strategic Reasoning 2013 (SR'13)
EPTCS 112, 2013, pp. 1–1, doi:10.4204/EPTCS.112.1

Model Checking Systems against Epistemic Specifications

Invited Talk

Alessio R. Lomuscio
Imperial College London

Twenty years after the publication of the influential article ”Model checking vs theorem proving: a manifesto” by Halpern and Vardi, the area of model checking systems against agent-based specifications is flourishing. In this talk I will present some of the approaches I have developed with collaborators. I will begin by discussing BDD-based model checking for epistemic logic combined with ATL operators and then move to abstraction techniques including symmetry reduction. I will then highlight how, in our experience, bounded model checking can also successfully be used in this context, particularly in combination with BDDs, and how synthesis problems can be formulated and solved in an epistemic setting. The talk will include examples in the context of security protocols and a brief demo of MCMAS, an open-source model checker implementing some of these techniques.

F. Mogavero, A. Murano, and M.Y. Vardi (Eds.):
1st Workshop on Strategic Reasoning 2013 (SR’13)
EPTCS 112, 2013, pp. 3–3, doi:10.4204/EPTCS.112.2

Looking at Mean-Payoff and Total-Payoff through Windows

Invited Talk

Jean-Francois Raskin
Université Libre de Bruxelles

We consider two-player games played on weighted directed graphs with mean-payoff and total-payoff objectives, which are two classical quantitative objectives. While for single dimensional objectives all results for mean-payoff and total-payoff coincide, we show that in contrast to multi-dimensional mean-payoff games that are known to be coNP-complete, multi-dimensional total-payoff games are undecidable. We introduce conservative approximations of these objectives, where the payoff is considered over a local finite window sliding along a play, instead of the whole play. For single dimension, we show that (i) if the window size is polynomial, then the problem can be solved in polynomial time, and (ii) the existence of a bounded window can be decided in NP and in coNP, and is at least as hard as solving mean-payoff games. For multiple dimensions, we show that (i) the problem with fixed window size is ExpTime-complete, and (ii) there is no primitive-recursive algorithm to decide the existence of a bounded window.

F. Mogavero, A. Murano, and M.Y. Vardi (Eds.):
1st Workshop on Strategic Reasoning 2013 (SR'13)
EPTCS 112, 2013, pp. 5–5, doi:10.4204/EPTCS.112.3

Bad Equilibria (and what to do about them)

Invited Talk

Michael Wooldridge
University of Oxford

In economics, an equilibrium is a steady-state situation, which obtains because no participant has any rational incentive to deviate from it. Equilibrium concepts are arguably the most important and widely used analytical weapons in the game theory arsenal. The concept of Nash equilibrium in particular has found a huge range of applications, in areas as diverse and seemingly unrelated as biology and moral philosophy. However, there remain fundamental problems associated with Nash equilibria and their application. First, there may be multiple Nash equilibria, in which case, how should we choose between them? Second, some equilibria may be undesirable, in which case, how can we avoid them? In this presentation, I will introduce work that we have done addressing these problems from a computational/AI perspective. Assuming no prior knowledge of game theory or economic solution concepts, I will discuss various ways in which we can try to engineer a game so that desirable equilibria result, or else engineer out undesirable equilibria. In particular, I will consider thee possible devices for the management of equilibria: taxation, communication, and law-making. While all of these devices are regularly used in human societies, in this work, we consider these as computational problems.

F. Mogavero, A. Murano, and M.Y. Vardi (Eds.):
1st Workshop on Strategic Reasoning 2013 (SR'13)
EPTCS 112, 2013, pp. 7–7, doi:10.4204/EPTCS.112.4

Functional Dependence in Strategic Games (extended abstract)

Kristine Harjes and Pavel Naumov

Department of Mathematics and Computer Science
McDaniel College, Westminster, Maryland, USA
{keh013,pnaumov}@mcdaniel.edu

The paper studies properties of functional dependencies between strategies of players in Nash equilibria of multi-player strategic games. The main focus is on the properties of functional dependencies in the context of a fixed dependency graph for pay-off functions. A logical system describing properties of functional dependence for any given graph is proposed and is proven to be complete.

1 Introduction

Functional Dependence. In this paper we study dependency between players' strategies in Nash equilibria. For example, the coordination game described by Table 1 has two Nash equilibria: (a_1,b_1) and (a_2,b_2). Knowing the strategy of player a in a Nash equilibrium of this game, one can predict the strategy of player b. We say that player a functionally determines player b and denote this by $a \rhd b$.

	b_1	b_2
a_1	1,1	0,0
a_2	0,0	1,1

Table 1: Coordination Game

Note that in the case of the coordination game, we also have $b \rhd a$. However, for the game described by Table 2 statement $a \rhd b$ is true, but $b \rhd a$ is false.

The main focus of this paper is functional dependence in multi-player games. For example, consider a "parity" game with three players a, b, c. Each of the players picks 0 or 1, and all players are rewarded if the sum of all three numbers is even. This game has four different Nash equilibria: $(0,0,0)$, $(0,1,1)$, $(1,0,1)$, and $(1,1,0)$. It is easy to see that knowledge of any two players' strategies in a Nash equilibrium reveals the third. Thus, using our notation, for example $a,b \rhd c$. At the same time, $\neg(a \rhd c)$.

As another example, consider a game between three players in which each player picks 0 or 1 and all players are rewarded if they have chosen the same strategy. This game has only two Nash equilibria: $(0,0,0)$ and $(1,1,1)$. Thus, knowledge of the strategy of player a in a Nash equilibrium reveals the strategies of the two other players. We write this as $a \rhd b,c$.

	b_1	b_2
a_1	1,1	0,0
a_2	0,0	1,1
a_3	1,1	0,0

Table 2: Strategic Game

Functional dependence as a relation has been studied previously, especially in the context of database theory. Armstrong [1] presented the following sound and complete axiomatization of this relation:

1. *Reflexivity*: $A \rhd B$, if $B \subseteq A$,
2. *Augmentation*: $A \rhd B \to A,C \rhd B,C$,

F. Mogavero, A. Murano, and M.Y. Vardi (Eds.):
1st Workshop on Strategic Reasoning 2013 (SR'13)
EPTCS 112, 2013, pp. 9–15, doi:10.4204/EPTCS.112.5

3. *Transitivity*: $A \rhd B \to (B \rhd C \to A \rhd C)$,

where here and everywhere below A,B denotes the union of sets A and B. The above axioms are known in database literature as Armstrong's axioms [5]. Beeri, Fagin, and Howard [2] suggested a variation of Armstrong's axioms that describe properties of multi-valued dependence.

Dependency Graphs. As a side result, we will show that the logical system formed by the Armstrong axioms is sound and complete with respect to the strategic game semantics. Our main result, however, is a sound and complete axiomatic system for the relation $\rhd$ in games with a given dependency graph.

Dependency graphs [7, 8, 4, 3] put restrictions on the pay-off functions that can be used in the game. For example, dependency graph Γ_1 depicted in Figure 1, specifies that the pay-off function of player a only can depend on the strategy of player b in addition to the strategy of player a himself. The pay-off function for player b can only depend on the strategies of players a and c in addition to the strategy of player b himself, etc.

Figure 1: Dependency Graph Γ_1

An example of a game over graph Γ_1 is a game between players a, b, c, and d in which these players choose real numbers as their strategies. The pay-off function of players a and d is the constant 0. Player b is rewarded if his value is equal to the mean of the values of players a and c. Player c is rewarded if his value is equal to the mean of the values of players b and d. Thus, Nash equilibria of this game are all quadruples (a,b,c,d) such that $2b = a+c$ and $2c = b+d$. Hence, in this game $a,b \rhd c,d$ and $a,c \rhd b,d$, but $\neg(a \rhd b)$.

Note that although the statement $a,b \rhd c,d$ is true for the game described above, it is not true for many other games with the same dependency graph Γ_1. In this paper we study properties of functional dependence that are common to all games with the same dependency graph. An example of such statement for the graph Γ_1, as we will show in Proposition 1, is $a \rhd d \to b,c \rhd d$.

Informally, this property is true for any game over graph Γ_1 because any dependencies between players a and d must be established through players b and c. This intuitive approach, however, does not always lead to the right conclusion. For example, in graph Γ_2 depicted in Figure 2, players b and c also separate players a and d. Thus, according to the same intuition, the statement $a \rhd d \to b,c \rhd d$ must also be true for any game over graph Γ_2. This, however, is not true. Consider, for example, a game in which all four players have three strategies: *rock*, *paper*, and *scissors*. The pay-off function of players a and d is the constant 0. If a and d pick the same strategy, then neither b nor c is paid. If players a and d pick different strategies, then players b and c are paid according to the rules of the standard rock-paper-scissors game. In this game Nash equilibrium is only possible if a and d pick the same strategy. Hence, $a \rhd d$. At the same time, in any such equilibria b and c can have any possible combination of values. Thus, $\neg(b,c \rhd d)$. Therefore, the statement $a \rhd d \to b,c \rhd d$ is not true for this game.

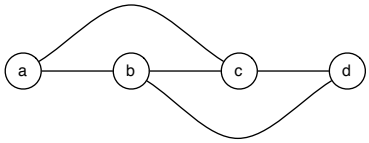


Figure 2: Dependency Graph Γ_2

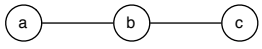


Figure 3: Dependency Graph Γ_3

As our final example, consider the graph Γ_3 depicted in Figure 3. We will show that $a \rhd c \to b \rhd c$ is not true for at least one game over graph Γ_3. Indeed, consider the game in which players a,b, and c use real numbers as possible strategies. Players a and c have a constant pay-off of 0. The pay-off of the player b is equal to 0 if players a and c choose the same real number. Otherwise, it is equal to the number chosen by the player b himself. Note that in any Nash equilibrium of this game, the strategies of players a and c are equal. Therefore, $a \rhd c$, but $\neg(b \rhd c)$.

The main result of this paper is a sound and complete axiomatization of all properties of functional

dependence for any given dependency graph. This result is closely related to work by More and Naumov on functional dependence of secrets over hypergraphs [9]. However, the logical system presented in this paper is significantly different from theirs. A similar relation of "rational" functional dependence without any connection to dependency graphs has been axiomatized by Naumov and Nicholls [10].

The counterexample that we have constructed for the game in Figure 3 significantly relies on the fact that player b has infinitely many strategies. However, in this paper we show completeness with respect to the semantics of finite games, making the result stronger.

2 Syntax and Semantics

The graphs that we consider in this paper contain no loops, multiple edges, or directed edges.

Definition 1 *For any set of vertices U of a graph (V,E), border $\mathscr{B}(U)$ is the set*

$$\{v \in U \mid (v,w) \in E \text{ for some } w \in V \setminus U\}.$$

A cut (U,W) of a graph (V,E) is a partition $U \sqcup W$ of the set V. For any vertex v in a graph, by $Adj(v)$ we mean the set of all vertices adjacent to v. By $Adj^+(v)$ we mean the set $Adj(v) \cup \{v\}$.

Definition 2 *For any graph $\Gamma = (V,E)$, by $\Phi(\Gamma)$ we mean the minimal set of formulas such that (i) $\bot \in \Phi(\Gamma)$, (ii) $A \rhd B \in \Phi(\Gamma)$ for each $A \subseteq V$ and $B \subseteq V$, (iii) $\phi \to \psi \in \Phi(\Gamma)$ for each $\phi, \psi \in \Phi(\Gamma)$.*

Definition 3 *By game over graph $\Gamma = (V,E)$ we mean any strategic game $G = (V, \{S_v\}_{v\in V}, \{u_v\}_{v\in V})$ such that (i) The finite set of players in the game is the set of vertices V, (ii) The finite set of strategies S_v of any player v is an arbitrary set, (iii) The pay-off function u_v of any player v only depends on the strategies of the players in $Adj^+(v)$.*

By $NE(G)$ we denote the set of all Nash equilibria in the game G. The next definition is the core definition of this paper. The second item in the list below gives a precise meaning of the functional dependence predicate $A \rhd B$.

Definition 4 *For any game G over graph Γ and any $\phi \in \Phi(\Gamma)$, we define binary relation $G \vDash \phi$ as follows (i) $G \nvDash \bot$, (ii) $G \vDash A \rhd B$ if $\mathbf{s} =_A \mathbf{t}$ implies $\mathbf{s} =_B \mathbf{t}$ for each $\mathbf{s}, \mathbf{t} \in NE(G)$, (iii) $G \vDash \psi_1 \to \psi_2$ if $G \nvDash \psi_1$ or $G \vDash \psi_2$, where here and everywhere below $\langle s_v \rangle_{v\in V} =_X \langle t_v \rangle_{v\in V}$ means that $s_x = t_x$ for each $x \in X$.*

3 Axioms

The following is the set of axioms of our logical system. It consists of the original Armstrong axioms and an additional Contiguity axiom that captures properties of functional dependence specific to a given graph Γ.

1. Reflexivity: $A \rhd B$, where $B \subseteq A$
2. Augmentation: $A \rhd B \to A, C \rhd B, C$
3. Transitivity: $A \rhd B \to (B \rhd C \to A \rhd C)$
4. Contiguity: $A, B \rhd C \to \mathscr{B}(U), \mathscr{B}(W), B \rhd C$, where (U,W) is a cut of the graph such that $A \subseteq U$ and $C \subseteq W$.

Note that the Contiguity axiom, unlike the Gateway axiom [9], effectively requires "double layer" divider $\mathscr{B}(U), \mathscr{B}(W)$ between sets A and C. This is because in our setting values are assigned to the vertices and not to the edges of the graph.

We write $\vdash_\Gamma \phi$ if $\phi \in \Phi(\Gamma)$ is provable from the combination of the axioms above and propositional tautologies in the language $\Phi(\Gamma)$ using the Modus Ponens inference rule. We write $X \vdash_\Gamma \phi$ if ϕ is provable using the additional set of axioms X. We often omit the parameter Γ when its value is clear from the context.

Lemma 1 $\vdash A \rhd C \to A, B \rhd C$.

Proof. Assume $A \rhd C$. By the Reflexivity axiom, $A, B \rhd A$. Thus, by the Transitivity axiom, $A, B \rhd C$. ⊠

4 Examples

In this section we give examples of proofs in our formal system. The soundness and the completeness of this system will be shown in the appendix.

Proposition 1 $\vdash_{\Gamma_1} a \rhd d \to b, c \rhd d$, *where* Γ_1 *is the graph depicted in Figure 1.*

Proof. Consider cut (U, W) of the graph Γ_1 such that $U = \{a, b\}$ and $W = \{c, d\}$. Thus, $\mathscr{B}(U) = \{b\}$ and $\mathscr{B}(W) = \{c\}$. Therefore, by the Contiguity axiom, $a \rhd d \to b, c \rhd d$. ⊠

Proposition 2 $\vdash_{\Gamma_1} a, c \rhd d \to (d, b \rhd a \to b, c \rhd a, d)$, *where* Γ_1 *is the graph depicted in Figure 1.*

Proof. Assume that $a, c \rhd d$ and $d, b \rhd a$. Consider cut (U, W) of the graph Γ_1 such that $U = \{a, b\}$ and $W = \{c, d\}$. Thus, $\mathscr{B}(U) = \{b\}$ and $\mathscr{B}(W) = \{c\}$. Therefore, by the Contiguity axiom with $A = \{a\}$, $B = \{c\}$, and $C = \{d\}$, $a, c \rhd d \to b, c \rhd d$. Thus,

$$b, c \rhd d. \tag{1}$$

by the first assumption. Similarly, using the second assumption, $b, c \rhd a$. Hence, by the Augmentation axiom,

$$b, c \rhd a, b, c. \tag{2}$$

Thus, from statement (1) by the Augmentation axiom, $a, b, c \rhd a, d$. Finally, using statement (2) and the Transitivity axiom, $b, c \rhd a, d$. ⊠

Proposition 3 $\vdash_{\Gamma_4} a, c \rhd e \to b, c, d \rhd e$, *where* Γ_4 *is the graph depicted in Figure 4.*

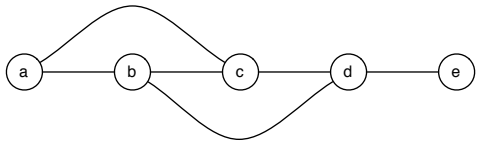


Figure 4: Dependency Graph Γ_4

Proof. Consider cut (U, W) of the graph Γ_4 such that $U = \{a, b, c\}$ and $W = \{d, e\}$. Thus, $\mathscr{B}(U) = \{b, c\}$ and $\mathscr{B}(W) = \{d\}$. Therefore, $a, c \rhd e \to b, c, d \rhd e$ by the Contiguity axiom with $A = \{a\}$, $B = \{c\}$, and $C = \{e\}$. ⊠

Proposition 4 *$\vdash_{\Gamma_5} a \rhd b \to (b \rhd c \to (c \rhd a \to d,e,f \rhd a,b,c))$, where Γ_5 is depicted in Figure 5.*

Proof. Assume $a \rhd b$, $b \rhd c$, and $c \rhd a$. Consider cut (U,W) of the graph Γ_5 such that $U = \{c,f\}$ and $W = \{a,b,d,e\}$. Thus, $\mathscr{B}(U) = \{f\}$ and $\mathscr{B}(W) = \{d,e\}$. Therefore, by the Contiguity axiom with $A = \{c\}$, $B = \varnothing$, and $C = \{a\}$, $c \rhd a \to d,e,f \rhd a$. Hence, $d,e,f \rhd a$ by the third assumption. Similarly, one can show $d,e,f \rhd b$, and $d,e,f \rhd c$. By applying the Augmentation axiom to the last three statements, $d,e,f \rhd a,d,e,f$, and $a,d,e,f \rhd a,b,d,e,f$, and $a,b,d,e,f \rhd a,b,c$. Therefore, $d,e,f \rhd a,b,c$ by the Transitivity axiom applied twice. ⊠

Proposition 2 and Proposition 4 are special cases of a more general principle. We will say that a subset of vertices is *sparse* if the shortest path between any two vertices in this subset contains at least three edges. The general principle states that if W is a sparse subset of vertices in the graph (V,E) and each vertex $w \in W$ is functionally determined by the set $V \setminus \{w\}$, then the subset $V \setminus W$ functionally determines the subset W:

$$\bigwedge_{w \in W} ((V \setminus \{w\})) \rhd w \to (V \setminus W) \rhd W.$$

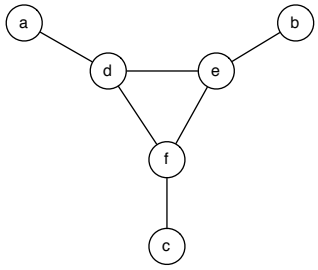


Figure 5: Dependency Graph Γ_5

For example, the set $\{a,d\}$ in the graph Γ_1 depicted in Figure 1 is sparse. Due to the general principle, $a,b,c \rhd d \to (d,c,b \rhd a \to b,c \rhd a,d)$. Thus, by Lemma 1, $a,c \rhd d \to (d,b \rhd a \to b,c \rhd a,d)$, which is the statement of Proposition 2. In the case of Proposition 4, the sparse set is $\{a,b,c\}$. The proof of the general principle is similar to the proof of Proposition 4.

5 Soundness

In this section, we prove soundness of our logical system by proving soundness of each of our four axioms. The proof of completeness can be found in [6].

Lemma 2 (reflexivity) *$G \vDash A \rhd B$ for each game G over a graph $\Gamma = (V,E)$ and each $B \subseteq A \subseteq V$.*

Proof. For any $\mathbf{s},\mathbf{t} \in NE(G)$, if $\mathbf{s} =_A \mathbf{t}$, then $\mathbf{s} =_B \mathbf{t}$ because $A \subseteq B$. ⊠

Lemma 3 (augmentation) *If $G \vDash A \rhd B$, then $G \vDash A,C \rhd B,C$ for each game G over a graph $\Gamma = (V,E)$ and each $A,B,C \subseteq V$.*

Proof. Suppose that $G \vDash A \rhd B$ and consider any $\mathbf{s},\mathbf{t} \in NE(G)$ such that $\mathbf{s} =_{A,C} \mathbf{t}$. We will show that $\mathbf{s} =_{B,C} \mathbf{t}$. Indeed, $\mathbf{s} =_{A,C} \mathbf{t}$ implies that $\mathbf{s} =_A \mathbf{t}$ and $\mathbf{s} =_C \mathbf{t}$. Thus, $\mathbf{s} =_B \mathbf{t}$ by the assumption $G \vDash A \rhd B$. Therefore, $\mathbf{s} =_{B,C} \mathbf{t}$. ⊠

Lemma 4 (transitivity) *If $G \vDash A \rhd B$ and $G \vDash B \rhd C$, then $G \vDash A \rhd C$ for each game G over a graph $\Gamma = (V,E)$ and each $A,B,C \subseteq V$.*

Proof. Suppose that $G \vDash A \rhd B$ and $G \vDash B \rhd C$. Consider any $\mathbf{s},\mathbf{t} \in NE(G)$ such that $\mathbf{s} =_A \mathbf{t}$. We will show that $\mathbf{s} =_C \mathbf{t}$. Indeed, $\mathbf{s} =_B \mathbf{t}$ due to the first assumption. Hence, by the second assumption, $\mathbf{s} =_C \mathbf{t}$. ⊠

Lemma 5 (contiguity) *If $G \vDash A, B \rhd C$, then $G \vDash \mathscr{B}(S), \mathscr{B}(T), B \rhd C$, for each game $G = (V,E)$ over a graph Γ, each cut (U,W) of Γ, and each $A \subseteq U$, $B \subseteq V$, and $C \subseteq W$.*

Proof. Suppose that $G \vDash A, B \rhd C$. Consider any $\mathbf{s} = \langle s_v \rangle_{v \in V} \in NE(G)$ and $\mathbf{t} = \langle t_v \rangle_{v \in V} \in NE(G)$ such that $\mathbf{s} =_{\mathscr{B}(U), \mathscr{B}(W), B} \mathbf{t}$. We will prove that $\mathbf{s} =_C \mathbf{t}$. Indeed, consider strategy profile $\mathbf{e} = \langle e_v \rangle_{v \in V}$ such that

$$e_v = \begin{cases} s_v & \text{if } v \in U, \\ t_v & \text{if } v \in W. \end{cases}$$

We will first prove that $\mathbf{e} \in NE(G)$. Assuming the opposite, let $v \in V$ be a player in the game G that can increase his pay-off by changing strategy in profile $\mathbf{e}$. Without loss of generality, let $v \in U$. Then, $\mathbf{e} =_{Adj(v) \cup \{v\}} \mathbf{s}$. Thus, player v can also increase his pay-off by changing strategy in profile $\mathbf{s}$, which is a contradiction with the choice of $\mathbf{s} \in NE(G)$.

Note that $\mathbf{e} =_{U,B} \mathbf{s}$ and $\mathbf{e} =_{W,B} \mathbf{t}$. Thus, $\mathbf{e} =_{A,B} \mathbf{s}$ and $\mathbf{e} =_C \mathbf{s}$. Hence, $\mathbf{e} =_C \mathbf{s}$ by the assumption $G \vDash A, B \rhd C$. Therefore, $\mathbf{s} =_C \mathbf{e} =_C \mathbf{t}$. ⊠

6 Conclusion

In this paper, we have described a sound and complete logical system for functional dependence in strategic games over a fixed dependency graph. The dependency graph puts restrictions on the type of pay-off functions that can be used in the game. If no such restrictions are imposed, then the logical system for functional dependence in strategic games is just the set of original Armstrong axioms. This statement follows from our results since the absence of restrictions corresponds to the case of a complete (in the graph theory sense) dependency graph. In the case of a complete graph, the Contiguity axiom follows from the Armstrong axioms because for any cut (U,W), the set $\mathscr{B}(U) \cup \mathscr{B}(W)$ is the set of all vertices in the graph.

References

[1] W. W. Armstrong (1974): *Dependency structures of data base relationships*. In: *Information processing 74 (Proc. IFIP Congress, Stockholm, 1974)*, North-Holland, Amsterdam, pp. 580–583.

[2] Catriel Beeri, Ronald Fagin & John H. Howard (1977): *A complete axiomatization for functional and multivalued dependencies in database relations*. In: *SIGMOD '77: Proceedings of the 1977 ACM SIGMOD international conference on Management of data*, ACM, New York, NY, USA, pp. 47–61, doi:10.1145/509404.509414.

[3] Edith Elkind, Leslie Ann Goldberg & Paul W. Goldberg (2006): *Nash Equilibria in Graphical Games on Trees Revisited*. *Electronic Colloquium on Computational Complexity (ECCC)* (005).

[4] Edith Elkind, Leslie Ann Goldberg & Paul W. Goldberg (2007): *Computing good Nash equilibria in graphical games*. In Jeffrey K. MacKie-Mason, David C. Parkes & Paul Resnick, editors: *ACM Conference on Electronic Commerce*, ACM, pp. 162–171, doi:10.1145/1250910.1250935.

[5] Hector Garcia-Molina, Jeffrey Ullman & Jennifer Widom (2009): *Database Systems: The Complete Book*, second edition. Prentice-Hall.

[6] Kristine Harjes & Pavel Naumov (2013): *Functional Dependence in Strategic Games*. *CoRR* arXiv:1302.0447 [math.LO].

[7] Michael J. Kearns, Michael L. Littman & Satinder P. Singh (2001): *Graphical Models for Game Theory*. In Jack S. Breese & Daphne Koller, editors: *UAI*, Morgan Kaufmann, pp. 253–260.

[8] Michael L. Littman, Michael J. Kearns & Satinder P. Singh (2001): *An Efficient, Exact Algorithm for Solving Tree-Structured Graphical Games*. In Thomas G. Dietterich, Suzanna Becker & Zoubin Ghahramani, editors: *NIPS*, MIT Press, pp. 817–823.

[9] Sara Miner More & Pavel Naumov (2011): *The Functional Dependence Relation on Hypergraphs of Secrets*. In João Leite, Paolo Torroni, Thomas Ågotnes, Guido Boella & Leon van der Torre, editors: *CLIMA*, *Lecture Notes in Computer Science* 6814, Springer, pp. 29–40, doi:10.1007/978-3-642-22359-4_3.

[10] Pavel Naumov & Brittany Nicholls (2012): *Rationally Functional Dependence*. In: *10th Conference on Logic and the Foundations of Game and Decision Theory (LOFT)*.

Restricted Manipulation in Iterative Voting: Convergence and Condorcet Efficiency

Umberto Grandi
University of Padova
umberto.uni@gmail.com

Andrea Loreggia
University of Padova
andrea.loreggia@gmail.com

Francesca Rossi
University of Padova
frossi@math.unipd.it

Kristen Brent Venable
Tulane University and IHMC
kvenabl@tulane.edu

Toby Walsh
NICTA and UNSW
toby.walsh@nicta.com.au

In collective decision making, where a voting rule is used to take a collective decision among a group of agents, manipulation by one or more agents is usually considered negative behavior to be avoided, or at least to be made computationally difficult for the agents to perform. However, there are scenarios in which a restricted form of manipulation can instead be beneficial. In this paper we consider the iterative version of several voting rules, where at each step one agent is allowed to manipulate by modifying his ballot according to a set of restricted manipulation moves which are computationally easy and require little information to be performed. We prove convergence of iterative voting rules when restricted manipulation is allowed, and we present experiments showing that most iterative voting rules have a higher Condorcet efficiency than their non-iterative version.

1 Introduction

In multi-agent systems, often agents need to take a collective decision. A voting rule can be used to decide which decision to take, mapping the agents' preferences over the possible candidate decisions into a winning decision for the collection of agents. In this kind of scenarios, it seems desirable that agents do not have any incentive to manipulate, that is, to misreport their preferences in order to influence the result of the voting rule in their favor.

Manipulation is indeed usually seen as bad behavior from agents, to be avoided or at least to be made computationally difficult to accomplish. While we know that every voting rule is manipulable when no domain restriction is imposed on the agents' preferences (such as single-peakedness), we can try to make sure that a voting rule is computationally difficult to manipulate for single agents or coalitions of agents.

In this paper we consider a different setting, in which instead manipulation is allowed in a fair way. As in the usual case, we start with agents expressing their preferences over a set of candidates and the voting rule selecting the current winner. However, this is just a temporary winner, since at this point a single agent may decide to manipulate, that is, to change her preference if by doing so the result changes in her favor. The process repeats with a new agent manipulating until we eventually reach a convergence state, i.e., a profile where no single agent can get a better result by manipulating. We call such a process *iterative voting*. In this scenario, manipulation can be seen as a way to achieve consensus, to give every agent a chance to vote strategically (a sort of fairness), and to account for inter-agent influence over time.

A practical example of this process is Doodle,[1] a very popular on-line system to select a time slot for a meeting by considering the preferences of the participants. In Doodle, each participant can approve as

[1] http://doodle.com/

F. Mogavero, A. Murano, and M.Y. Vardi (Eds.):
1st Workshop on Strategic Reasoning 2013 (SR'13)
EPTCS 112, 2013, pp. 17–24, doi:10.4204/EPTCS.112.6

many time slots as she wants, and the winning time slot is the one with the largest number of approvals. At any point, each participant can modify her vote in order to get a better result, and this can go on for several steps. Depending on the voting rule, on the tie-breaking rule (to be used when there are several tied winners), and on the possible manipulation moves (that is, how agents are allowed to change their preferences in a single step), we may get convergence or not. We will say that the iterative version of a voting rule converges if it gets to a stable state no matter the initial profile.

Iterative voting has been the subject of numerous publications in recent years. Previous work has focused on iterating the plurality rule [6] and on the problem of convergence for several voting rules [5]. Lev and Rosenschein [5] showed that, if we allow agents to manipulate in any way they want (i.e., to provide their best response to the current profile), then the iterative version of most voting rules do not converge. Therefore, an interesting problem is to seek restrictions on the manipulation moves to guarantee convergence of the associated iterative rule. Restricted manipulation moves are good not only for convergence, but also because they can be easier to accomplish for the manipulating agent. In fact, contrarily to what we aim for in classical voting scenarios, here we do not want manipulation to be computationally difficult to achieve. It is actually desirable that the manipulation move be easy to compute while not requiring too much information to be computed.

An example of a restricted manipulation move is the one for agents called k-pragmatists by Reijngoud and Endriss [8]: a k-pragmatist just needs to know the top k candidates in the collective candidate order, and will move the most preferred of those candidates to the top position of her preference. To accomplish this move, a k-pragmatist needs very little information and it is computationally easy to perform the move. This move assures convergence with a number of voting rules.

In this paper we introduce two restricted manipulation moves within the scenario of iterative voting and we analyze some of their theoretical and practical properties. Both manipulation moves we consider are polynomial to compute and require little information to be used. We show that convergence is guaranteed under both moves, except for STV for which we only have experimental evidence of convergence. Moreover, we show that if a voting rule satisfies some axiomatic properties, such as Condorcet consistency or unanimity, then its iterative version will also satisfy the same properties as well. For voting rules that are not Condorcet consistent, we tested experimentally whether their Condorcet efficiency (that is, the probability to elect the Condorcet winner) improves by adopting the iterative version. Our experiments show that the Condorcet efficiency improves when restricted manipulation moves are used.

The paper is organized as follows. In Section 2 we introduce the basic definitions of iterative voting and we define two new restricted manipulation moves. Section 3 contains theoretical results on convergence and preservation of axiomatic properties, and in Section 4 we present our experimental evaluation of restricted iterative voting. Section 5 contains our conclusions and directions for future research.

2 Background Notions

In this section we recall the basic notions of voting theory that we shall use in this paper, we present the setting of iterative voting, and we define a number of restrictions on the manipulation moves that agents can perform.

2.1 Voting Rules

Let $\mathscr{X}$ be a finite set of m candidates and $\mathscr{I}$ be a finite set of n individuals. We assume individuals have preferences p_i over candidates in $\mathscr{X}$ in the form of *strict linear orders*, i.e., transitive, anti-symmetric and complete binary relations. Individuals express their preferences in form of a ballot b_i (e.g., the top candidate, a set of approved candidates, or the full linear order) and we call the choice of a ballot for each individual a profile $\mathbf{b} = (b_1, \ldots, b_n)$. In this paper, we assume that individuals submit as a ballot for the

election their full linear order, and we thus use the two notions of ballot and preference interchangeably.

A (non-resolute) *voting rule* F associates with every profile $\mathbf{b} = (b_1, \ldots, b_n)$ a non-empty subset of winning candidates $F(\mathbf{b}) \in 2^{\mathscr{X}} \setminus \emptyset$. There is a wide collection of voting rules that have been defined in the literature [2] and here we focus on the following ones:

Positional scoring rules (PSR): Let $(s_1, \ldots, s_m)$ be a scoring vector such that $s_1 \geq \cdots \geq s_m$ and $s_1 > s_m$. If a voter ranks candidate c at j-th position in her ballot, this gives s_j points to the candidate. The candidates with the highest score win. We focus on four particular PSR: *Plurality* with scoring vector $(1,0,\ldots,0)$, *veto* with vector $(1,\ldots,1,0)$, 2-*approval* with vector $(1,1,0,\ldots,0)$, 3-*approval* with vector $(1,1,1,0,\ldots,0)$, and *Borda* with vector $(m-1, m-2, \ldots, 0)$.

Copeland: The score of candidate c is the number of pairwise comparisons she wins (i.e., contests between c and another candidate a such that there is a majority of voters preferring c to a) minus the number of pairwise comparisons she loses. The candidates with the highest score win.

Maximin: The score of a candidate c is the smallest number of voters preferring it in any pairwise comparison. The candidates with the highest score win.

Single Transferable Vote (STV): At the first round the candidate that is ranked first by the fewest number of voters gets eliminated (ties are broken following a predetermined order of candidates). Votes initially given to the eliminated candidate are then transferred to the candidate that comes immediately after in the individual preferences. This process is iterated until one alternative is ranked first by a majority of voters.

All rules considered thus far are non-resolute, i.e., they associate a set of winning candidates with every profile of preferences. To eliminate ties in the outcome we assume that the set $\mathscr{X}$ of candidates is ordered by $\prec_{\mathscr{X}}$, and in case of ties the alternative ranked highest by $\prec_{\mathscr{X}}$ is chosen as the unique outcome.

2.2 Iterative Voting

A classical problem studied in voting theory is that of manipulation: do individuals have incentive to misreport their preferences, in order to force a candidate they prefer as winner of the election? The Gibbard-Satterthwaite Theorem [4, 9] showed that under natural conditions all voting rules can be manipulated. Following this finding, a considerable amount of work has been spent on devising conditions to avoid manipulation, e.g., in form of restrictive conditions on individual preferences, or in form of computational barriers that make the calculation of manipulation strategies too hard for agents [1, 3].

In this paper, we take a different stance on manipulation: we consider the fact that individuals are allowed to change their preferences as a positive aspect of the voting process, that may eventually lead to a better result after a sufficient number of steps. Thus, we consider a sequence of repeated elections in which at each step one of the individuals is allowed to manipulate, i.e., to modify her ballot in order to change the outcome of the election in her favor. The iteration process starts at $\mathbf{b}^0$ (which we shall refer to as the truthful profile) and continues to $\mathbf{b}^1, \ldots, \mathbf{b}^k$. At each step only one individual $\tau(k)$ is allowed to manipulate, following a turn function τ (e.g., τ follows the order in which individuals are given), while all other individual ballots remain unchanged.

The setting of iterative voting was first introduced and studied by [6] for the case of the plurality rule, and expanded by [5]. In their work, the authors describe the iterated election process as a *voting game*, in which convergence of the iterative process corresponds to reaching a Nash equilibria of the game. They show that convergence is rarely guaranteed with most voting rules under consideration: for instance, the iterative version of PSRs and Maximin do not always converge, even with deterministic tie-breaking (i.e., not randomized). On the other hand, plurality always converges with any tie-breaking rule, as well as veto with linear tie-breaking.

2.3 Restricted Manipulation Moves

The convergence of the iterated version of a voting rule can be obtained by restricting the set of manipulation strategies available to the agents. We now list a number of restrictions that have been studied in the literature, and we add two new definitions to this list. Let $\mathbf{b}^k$ be the current profile at step k, $\mathbf{b}^0$ be the initial (truthful) profile, and F be a voting rule. Assume that $\tau(k) = i$.

Best response (no restriction): the manipulator i changes her full ballot by selecting the linear order which results in the best possible outcome for her truthful preference b_i^0 [5].

k-pragmatist: the manipulator i moves to the top of her reported ballot the most preferred candidate following b_i^0 among those that scored in the top k positions [8].

M1: the manipulator i moves to the top of her reported ballot the second-best candidate in b_i^0, unless the current winner $w = F(\mathbf{b}^k)$ is already her best or second-best candidate in b_i^0.

M2: the manipulator i moves to the top of her reported ballot the most preferred candidate in b_i^0 which is above $w = F(\mathbf{b}^k)$ in b_i^k, among those that can become the new winner of the election.

Different restrictions on manipulation moves induce different iterated versions of a voting rule:

Definition 1. *Let F be a voting rule and M a restriction on manipulation moves. $F^{M,\tau}$ associates with every profile* $\mathbf{b}$ *the outcome of the iteration of F using turn function τ and manipulation moves in M if this converges, and $\uparrow$ otherwise.*

In the sequel we shall omit the superscript τ from the notation when this will be clear from the context. Observe that if M is the set of best responses, then $F^M = F^*$.

Restrictions on the set of manipulation moves can be evaluated following three parameters: (i) the convergence of the iterated voting rule associated with the restriction, (ii) the information to be provided to voters for computing their strategy[2], and (iii) the computational complexity of computing the manipulation move at every step. An ideal restriction always guarantees convergence, requires as little information as possible, and is computationally easy to compute.

As we pointed out at the end of the previous section, convergence is not guaranteed in most cases if the set of manipulation moves is not restricted (i.e., using best responses). Reijngoud and Endriss [8] show convergence for PSRs using the k-pragmatist restriction, and we shall investigate convergence results for M1 and M2 in the following section. Let us move to the other two parameters: on the one hand, M1 requires as little information as possible to be computed, i.e., only the winner of the current election, and is also very easy to compute. On the other hand, computing the best response requires an agent to have full knowledge of a profile, and may be computationally very hard to compute [1]. The k-pragmatist restriction has good properties: it is easy to compute, and the information required to compute the best strategy is just the set of the candidates ranked in the top k positions. M2 also requires little information for the agents: the scoring vector of candidates in case of scoring rules, the majority graph for Copeland and Maximin. In the case of STV the full profile is instead required. Moreover, from the point of view of the manipulator, M2 is computationally easy (i.e., polynomial) to perform.

3 Convergence and Axiomatic Properties

In this section we prove that the iterated version of PSR, Maximin and Copeland converge when using our two new restrictions on the manipulation moves. We also analyze, for a number of axiomatic properties, the behavior of the iterated version of a voting rule.

[2] This parameter is called *poll information function* by Reijngoud and Endriss [8].

Theorem 1. *F^{M1} converges for every voting rule F.*

Proof. The proof of this statement is straightforward from our definitions. The iteration process starts at the truthful profile $\mathbf{b}_0$, and each agent is then allowed to switch the top candidate with the one in second position. Thus, the iteration process stops after at most n steps. □

Theorem 2. *F^{M2} converges if F is a PSR, the Copeland rule or the Maximin rule.*

Proof. The winner of an election using a PSR, Copeland or Maximin is defined as the candidate maximizing a certain score (or with maximal score and higher rank in the tie-breaking order). Since the maximal score of a candidate is bounded, it is sufficient to show that the score of the winner increases at every iteration step (or, in case the score remains constant, that the position of the winner in the tie-breaking order increases) to show that the iterative process converges.

Let us start with PSR. Recall that the score of a candidate c under PSR is $\sum_i s_i$ where s_i is the score given by the position of c in ballot b_i. Using M2, the manipulator moves to the top a candidate which lies above the current winner c. Thus, the position – and hence the score – of c remains unchanged, and the new winner must have a strictly higher score (or a better position in the tie-breaking order) than the previous one. The case of Copeland and Maximin can be solved in a similar fashion: it is sufficient to observe that the relative position of the current winner c with all other candidates (and thus also its score) remains unchanged when ballots are manipulated using M2. Thus, the Copeland score and the Maximin score of a new winner must by higher than that of c (or the new winner must be placed higher in the tie-breaking order). □

While currently we do not have a proof of convergence for STV, we observed experimentally that its iteration always terminates on profiles with a Condorcet winner when a suitable turn function described in the following section is used.

Voting rules are traditionally studied using axiomatic properties, and we can inquire whether these properties extend from a voting rule to its iterated version. We refer to the literature for an explanation of these properties [10]. Let us call F_t^M the iterated version of voting rule F after t iteration steps. We say that a restricted manipulation move *M preserves* a given axiom if whenever a voting rule F satisfies the axiom then also F_t^M does satisfy it for all t.

Theorem 3. *M1 and M2 preserve unanimity.*

Proof. Assume that the iteration process starts at a unanimous profile $\mathbf{b}$ in which candidate c is at top position of all individual preferences. If F is unanimous, then $F(\mathbf{b}) = c$, and no individual has incentives to manipulate either using M1 or M2. Thus, iteration stops at step 1 and $F_t^{\mathrm{M1}}(\mathbf{b}) = c$ and $F_t^{\mathrm{M2}}(\mathbf{b}) = c$, satisfying the axiom of unanimity. □

Theorem 4. *M1 and M2 preserve Condorcet consistency.*

Proof. Let c be the Condorcet winner of a profile $\mathbf{b}$. If F is Condorcet-consistent then $F(\mathbf{b}) = c$. As previously observed, when individuals manipulate using either M1 or M2 the relative position of the current winner with all other candidates does not change, since the manipulation only involves candidates that lie above the current winner in the individual preferences. Thus c remains the Condorcet winner in all iteration steps $\mathbf{b}^k$. Since $F_k^{\mathrm{M1}}(\mathbf{b}) = F(\mathbf{b}^k)$ and F is Condorcet-consistent, we have that $F_k^{\mathrm{M1}}(\mathbf{b}) = c$ and thus F_k^{M1} is Condorcet consistent. Similarly for M2. □

Other properties that transfer from a voting rule to its iterative version are neutrality and anonymity (supposing the turn function satisfies an appropriate version of neutrality and anonymity). The Pareto-condition does not transfer to the iterated version, as can be shown by adapting an example by Reijngoud and Endriss [8].

4 Experimental Evaluation of Restricted Manipulation Moves

In this section we evaluate our two restricted manipulation moves M1 and M2 under one important aspect: we measure whether the restricted iterative version of a voting rule has a higher Condorcet efficiency than the initial voting rule, i.e., whether the probability that a Condorcet winner (if it exists) gets elected is higher for the iterative rather than non-iterative rule. We show that in most cases the Condorcet efficiency of a voting rule increases if iterated manipulation is allowed using M1 or M2 (except for Copeland and Maximin which are already Condorcet-consistent rules). We also compare our findings with the *k-pragmatist* restriction for $k = 2, 3$.

Our results are obtained using a program implemented in Java ver.1.6.0. The software generates profiles with uniform distribution (i.e., impartial culture assumption). The impartial culture assumption has received criticism in recent years [7]. However, it remains the most common assumption used in social choice theory, and thus represents the obvious starting point for our empirical evaluation. Our test set contains 10.000 profiles with Condorcet winner. We set the number of candidates to 5 and varied the number of voters from 20 to 100.

The turn function used in our experiments associates to each voter i a dissatisfaction index $d_i(k)$, which increases of one point for each iteration step in which the individual has an incentive to manipulate but is not allowed to do so by the turn function. At iteration step k the individual that has the highest dissatisfaction index is allowed to move (in the first step, and in case of ties, the turn follows the initial order in which voters are given). We were always able to compute the outcome of the iterative voting rules after convergence in reasonable time.

4.1 Condorcet efficiency of restricted iterative voting

Figure 1 compares, for several voting rules, the Condorcet efficiency of the respective iterative version using restricted manipulation moves M1, M2, 2-*pragmatist* and 3-*pragmatist*. The number of voters is set to $n = 50$.

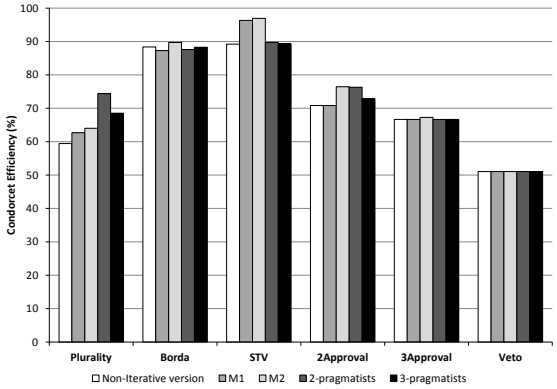


Figure 1: Iterated Condorcet efficiency.

Except for the case of the Borda rule, the Condorcet efficiency of the iterated version of a voting rule

improves significantly with respect to the non-iterated version, and the growth is significantly higher when voters manipulate the election using M2 rather than M1. A plausible reason for this behavior is the difference in range of candidates that can be helped by the two manipulation moves. While M1 may help a Condorcet winner being elected only if it was ranked second by some of the individuals, M2 may help a candidate even if it was ranked lower. Let us also stress that while the increase in Condorcet efficiency using M1 is minimal, it is still surprising that such a simple move can result in a better performance than the original version of the voting rule. The 2-*pragmatist* and 3-*pragmatist* restriction perform quite well with the plurality rule, while for all other rules our restriction M2 results in a better performance.

In the case of Plurality a significant increase can be obtained with both M1 and M2. In Figure 2 we show the trend in Condorcet efficiency when voters vary from $n = 20$ to $n = 100$. It can be observed that the increase is higher for smaller numbers of voters and stabilizes at around $n = 60$. The same behavior can be observed in Figure 3 for the case of STV.

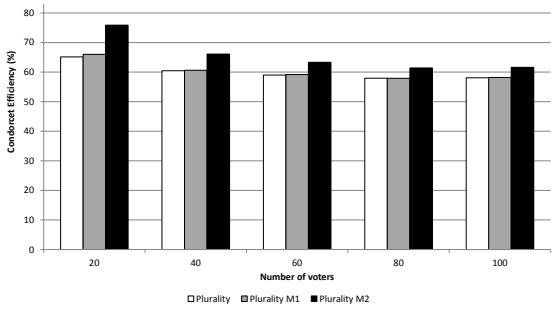


Figure 2: Condorcet efficiency of plurality.

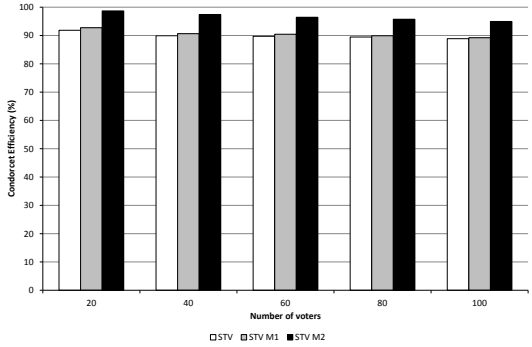


Figure 3: Condorcet efficiency for iterated STV.

STV has the highest performance of all voting rules considered thus far. STV has already a high Condorcet efficiency, but this is amplified by the use of manipulation moves, in particular M2. In Figure 3 we show that its Condorcet-efficiency can be augmented to more than 95 percent. As remarked earlier, we observed convergence in all profiles considered.

The absence of any increase in Condorcet efficiency for veto (as well as 2-approval and 3-approval using M1) is a consequence of the fact that our restricted moves do not change the candidates' score with these particular scoring vectors.

5 Conclusions and Future Work

This paper studies the iteration of classical voting rules allowing individuals to manipulate the outcome of the election using a restricted set of manipulation moves.

We provided two new definitions of manipulation moves M1 and M2 and showed that they lead to convergence for all voting rules considered (cf. Theorem 1 and 2). We also showed that most axiomatic properties, such as unanimity and Condorcet consistency, are preserved in the iteration process. We evaluated the performance of our restricted manipulation moves with respect to the Condorcet efficiency of the iterated version of a voting rule as well as the average position of the winner in the initial truthful profile. Our experiments showed that allowing restricted manipulation in iterative voting yields a positive increase in Condorcet efficiency, and that, predictably, the best performance is obtained when more information is given to agents (cf. the case of STV with M2).

This work gives rise to a number of interesting directions to be explored in future research. First, different restrictions on manipulation moves may be considered, and their performance should be compared with that of existing definitions. We tested a move similar to M2, which did not restrict the choice of a candidate to those who become the new winners of the iterated election, obtaining a performance comparable to that of M2. Restricted manipulation moves may also be evaluated using other parameters, and could be tested on more realistic distributions of profiles of preferences, for instance by exploiting data extracted from Internet-based polling services like Doodle.

References

[1] J. J. Bartholdi & J. B. Orlin (1991): *Single transferable vote resists strategic voting*. Social Choice and Welfare 8, pp. 341–354, doi:10.1007/BF00183045.

[2] Steven J. Brams & Peter C. Fishburn (2002): *Voting Procedures*. In Kenneth Arrow, Amartya Sen & Kotaro Suzumura, editors: Handbook of Social Choice and Welfare, Elsevier, doi:10.1016/S1574-0110(02)80008-X.

[3] P. Faliszewski & A. D. Procaccia (2010): *AI's War on Manipulation: Are We Winning?* AI Magazine 31(4), pp. 53–64.

[4] A. Gibbard (1973): *Manipulation of Voting Schemes: A General Result*. Econometrica 41(4), pp. 587–601, doi:10.2307/1914083.

[5] O. Lev & J. S. Rosenschein (2012): *Convergence of iterative voting*. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2012).

[6] R. Meir, M. Polukarov, J. S. Rosenschein & N. R. Jennings (2010): *Convergence to Equilibria in Plurality Voting*. In: Proceedings of the Twenty-fourth conference on Artificial Intelligence (AAAI-2010).

[7] M. Regenwetter, B. Grofman, A. Marley & I. Tsetlin (2006): *Behavioral social choice*. Cambridge University Press.

[8] A. Reijngoud & U. Endriss (2012): *Voter Response to Iterated Poll Information*. In: Proceedings of the 11th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2012).

[9] M. A. Satterthwaite (1975): *Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions*. Journal of Economic Theory 10(2), pp. 187 – 217, doi:10.1016/0022-0531(75)90050-2.

[10] A. D. Taylor (2005): *Social choice and the mathematics of manipulation*. Cambridge University Press, doi:10.1017/CBO9780511614316.

Infinite games with uncertain moves

Nicholas Asher and Soumya Paul
IRIT, Université Paul Sabatier, 118 Route de Narbonne, 31062 Toulouse, France.
{nicholas.asher,soumya.paul}@irit.fr *

We study infinite two-player games where one of the players is unsure about the set of moves available to the other player. In particular, the set of moves of the other player is a strict superset of what she assumes it to be. We explore what happens to sets in various levels of the Borel hierarchy under such a situation. We show that the sets at every alternate level of the hierarchy jump to the next higher level.

1 Introduction

Infinte two-player games have attracted a lot of attention and found numerous applications in the fields of topology, descriptive set-theory, computer science etc. Examples of such types of games are: Banach-Mazur games, Gale-Stewart games, Wadge games, Lipschitz games, etc. [7, 6, 11, 3], and they each characterize different concepts in descriptive set theory.

These games are typically played between two players, Player 0 and Player 1, who take turns in choosing finite sequences of elements (possibly singletons) from a fixed set A (finite or infinite) which is called the alphabet. This process goes on infinitely and hence defines an infinite sequence $u_0u_1u_2\ldots$ of finite strings which in itself is an infinite string over the set A. In addition, the game has a winning condition *Win* which is a subset of the set of infinite strings over A, A^ω. Player 0 is said to win the game if the sequence $u_0u_1u_2\ldots$ is in *Win*. Player 1 wins otherwise.

In addition to their applications in descriptive set-theory and topology, such games have also been used in computer science in the fields of verification and synthesis of reactive systems [4]. The verification problem is modeled as a game between two players: the system player and the environment player. The winning set *Win* is specified using formulas in some logic, LTL, CTL, μ-calculus etc. The goal of the system player is to meet the specification along every play and that of the environment player is to exhibit a play which does not meet it. To verify the system then amounts to show that the system player has a winning strategy in the underlying game and to find this strategy.

When *Win* is specified using the usual logics, it corresponds to sets in the low levels of the Borel hierarchy. It is known that the complexity of the winning strategy increases with the increase in the level of the Borel hierarchy to which *Win* belongs [10]. For instance, in Gale-Stewart games, reachability, safety and Muller are winning conditions in the Σ^0_1, Π^0_1 and Σ^0_2 levels of the Borel hierarchy respectively and a player has positional winning strategies for reachability and safety but needs memory to win for the Muller condition. However it was shown in [5, 8] that a finite amount of memory suffices. The notion of Wadge reductions also formalises this increase in complexity of the sets along the Borel hierarchy.

Such games (esp. Banach-Mazur and Gale-Stewart games) also find applications in linguistics. [2] shows that conversations have a topological structure similar to that of Banach-Mazur games and explores how the different types of objectives of conversations correspond to different levels in the Borel hierarchy

*We thank ERC grant 269427 for research support.

F. Mogavero, A. Murano, and M.Y. Vardi (Eds.):
1st Workshop on Strategic Reasoning 2013 (SR'13)
EPTCS 112, 2013, pp. 25–32, doi:10.4204/EPTCS.112.7

depending on their complexity. [2] also applied of the classical results from the literature of Banach-Mazur games to the conversational setting. [1] applies Gale-Stewart games to the study of politeness.

In this paper, we look at what happens to sets in the Borel hierarchy when the underlying alphabet is expanded. That is, the alphabet is changed from A to B such that B is a strict superset of A. We show that sets at every alternate level of the Borel hierarchy undergo a jump to the next higher level. More precisely, a set at level n of the hierarchy with alphabet A moves to level $n+1$ when the alphabet is expanded to B. This process goes on for all countable levels and stabilises at ω.

Our result has consequences for both formal verification and linguistic applications some of which we elucidate in the concluding section.

The rest of the paper is organised as follows. In Section 2 we formally introduce the necessary concepts and give the required background for the paper. Then in Section 3 we state and prove the main results of the paper. Finally we conclude with some interesting consequences in Section 4.

2 Preliminaries

In this section we present the necessary background required for the paper. Although we define most of the concepts used in the paper, we assume some familiarity with the basic notions of topology and set-theory.

2.1 Open and closed sets

Let A be a non-empty set. We sometimes refer to A as the alphabet. For any subset X of A, as usual, we denote by X^* the set of finite strings over X and by X^ω, the set of countably infinite strings over X. For any string $u \in A^* \cup A^\omega$ we denote the ith element of u by $u(i)$. The set of prefixes of u are all strings $v \in A^*$ such that $u = vv'$ for $v' \in A^* \cup A^\omega$.

We define a topology on A^ω, the standard topology (also known as the Cantor topology) on the set of infinite strings over A. This topology can be defined in at least three equivalent ways. The first way is to define the discrete topology on A and then assign A^ω the product topology. The second way is to explicitly define the open sets of the topology. The open sets are given by sets of the form XA^ω where X is a subset of A^*. Thus an open set is a set of finite strings over X followed by their all possible continuations. For a set $X \subseteq A^*$, we denote the open set XA^ω by $O_A(X)$ or simply by $O(X)$ when the underlying alphabet A is clear from the context. When X is a singleton $\{u\}$, we abuse notation to denote the open set uA^ω by $O_A(u)$. Example 1 illustrates these concepts.

Example 1. Let $A = \{a,b,c\}$. Then $abcA^\omega$ is an open set and so is $abA^\omega \cup baA^\omega$. The complement of the set $abcA^\omega$ is the set X of all strings that do not have abc as their prefix. This is a closed set.

Yet another equivalent way to define the topology is to give an explicit metric for it. Given two strings, $u_1, u_2 \in A^\omega$, the distance between them $d(u_1,u_2)$ is defined to be $1/2^{n(u_1,u_2)}$, where $n(u_1,u_2)$ is the first index where u_1 and u_2 differ from each other. Thus the above topology is metrisable. Henceforth, when we use the term 'set' we shall mean a subset of A^ω.

Note that the set $(\overline{abcA^\omega})$ in the above example is also open. That is because it is a union of the open sets $O(aa), O(ac), O(b)$ and $O(c)$. Such sets, which are both open and closed are called clopen sets. So what is a set which is open but not closed (and vice versa)?

Proposition 1 ([9]) *If A is a finite alphabet, a subset of A^ω is clopen if and only if it is of the form XA^ω where X is a finite subset of A^*.*

Thus if A is finite then a set of the form XA^ω where X is an infinite subset of A^* is open but not closed. If A is infinite, the subsets of A^ω of the form XA^ω, where X is a set of words of bounded length of A^* are clopen. However there might exist clopen sets which are not of this form.

2.2 The Borel hierarchy

A set of subsets of A^ω is called a σ-algebra if it is closed under countable unions and complements. Given a set X, the smallest σ-algebra containing X is called the σ-algebra generated by X. It is equivalent to the intersection of all the σ-algebras containing X. The sigma algebra generated by the open sets of a topological space is called the Borel σ-algebra and its sets are called the Borel sets.

The Borel sets can also be defined inductively. This gives a natural hierarchy of classes Σ^0_α and Π^0_α for $1 \leq \alpha < \omega_1$. Let Σ^0_1 be the set of all open sets. $\Pi_1 = \overline{\Sigma^0_1}$ is the set of all closed sets. Then for any $\alpha > 1$ where α is a successor ordinal, define Σ^0_α to be the countable union of all $\Pi^0_{\alpha-1}$ sets and define Π^0_α to be the complement of Σ^0_α. For a limit ordinal η, $1 < \eta < \omega_1$, Σ^0_η is defined as $\Sigma^0_\eta = \bigcup_{\alpha<\eta} \Sigma^0_\alpha$ and $\Pi^0_\eta = \overline{\Sigma^0_\eta}$. The infinite hierarchy thus generated is called the Borel hierarchy and they together form the Borel algebra. It is known [9] that if the space is metrisable and the underlying alphabet contains at least two elements, then the hierarchy is indeed infinite, that is, the containments, $\Sigma^0_\alpha \subset \Sigma^0_{\alpha+1}$ and $\Pi^0_\alpha \subset \Pi^0_{\alpha+1}$ are strict.

2.3 Wadge reductions and complete sets

Let A and B be two alphabets. A function $f : A^\omega \to B^\omega$ is said to be continuous if for every open subset $Y \subseteq B^\omega$, $f^{-1}(Y)$ is also open.

A set $X \subseteq A^\omega$ is said to Wadge reduce to another set $Y \subseteq B^\omega$, denoted $X \leq_W Y$, if there exists a continuous function $f : A^\omega \to B^\omega$ such that $f^{-1}(Y) = X$.

Let A be an alphabet. A set $X \subseteq A^\omega$ is said to be Σ^0_α (resp. Π^0_α) complete if $X \in \Sigma^0_\alpha$ (resp. $X \in \Pi^0_\alpha$) and for any other alphabet B and for any Σ^0_α (resp. Π^0_α) set $Y \subseteq B^\omega$, $Y \leq_W X$. Intuitively, given a class of sets Γ, the complete sets of that class represent the sets which are structurally the most complex in that class.

For the Borel hierarchy, completeness can be characterised in the following simple way:

Proposition 2 ([9]) *Let $X \subseteq A^\omega$. Then X is Π^0_α (resp. Σ^0_α) complete if and only if $X \in \Pi^0_\alpha \setminus \Sigma^0_\alpha$ (resp. $\Sigma^0_\alpha \setminus \Pi^0_{\alpha-1}$).*

2.4 Infinite games

Let A be an alphabet. An infinite game on A is played between two players, Player 0 and Player 1, who take turns in choosing finite sequences of elements (possibly singletons) from a fixed set A (finite or infinite) which is called the alphabet. This process goes on infinitely and hence defines an infinite sequence $u_0u_1u_2\ldots$ of finite strings which in itself is an infinite string over the set A. In addition, the game has a winning condition *Win* which is a subset of the set of infinite strings over A, A^ω. Player 0 is said to win the game if the sequence $u_0u_1u_2\ldots$ is in *Win*. Player 1 wins otherwise.

In a Banach-Mazur game, each player at her turn chooses a finite non-empty sequence of elements from A while in a Gale-Stewart game the players are restricted to choosing just single elements from A. An infinite game can also be imagined to be played on a graph $G = (V, E)$ where the set of vertices V is partitioned into V_0 and V_1 which represent the Player 0 and 1 vertices respectively. The game starts at an initial vertex $v_0 \in V$ and the players take turns in moving a token along the edges of the graph depending

on whose vertex it is currently. This process is continued ad infinitum and thus generates an infinite path p in the graph G. Player 0 wins if and only if $p \in Win$ where Win is a pre-specified set of infinite paths.

3 Results

In this section we present the main results of this paper. Given a subset B of an alphabet A the topology of B^ω where the open sets are given by $O \cap B^\omega$ for every open set O of A^ω is called the relative topology of B^ω with respect to A^ω. However we are interested in the opposite question. What happens when the alphabet expands? In particular, we show that when the alphabet set changes from A to B (say) such that B is a strict superset of A then the sets in the alternative levels of the Borel hierarchy undergo a jump in levels.

Lemma 1 *Let A and B be two alphabets such that $A \subsetneq B$. An open set O in the space A^ω jumps to Σ_2^0 in the space B^ω. A closed set C in the space A^ω remains closed in B^ω.*

Proof The proof is by carried out by coding the open set O in the space B^ω and demonstrating a complete set for B^ω.

Let O be an open set in A^ω. Then O is of the form XA^ω where $X \subseteq A^*$. Let $\mathscr{X}_\beta$ be an indexing of the set X.

Each element u of X gives the open set $O_A(u)$ which is a subset of A^ω. Now, when we move to the alphabet B, the set $O_B(u)$ is the set of strings which have u as a prefix and all possible continuations using letters of B. Thus $O_B(u)$ is a strict superset of $O_A(u)$. Hence, we need to restrict $O_B(u)$ in B^ω such that we obtain a set which is equal to $O_A(u)$ in A^ω. One way to do do so is as follows. Consider all the finite continuations of u in letters from A. Let $\mathscr{U}_\gamma$ be an indexed set of all these continuations. Then $O_A(u)$ is the set

$$O_A(u) = \bigcap O_B(u'),\ u' \in \mathscr{U}_\gamma \tag{1}$$

which is a closed set, being an arbitrary intersection of closed sets.

Thus the set O can be represented in B^ω as

$$O = \bigcup O_A(u),\ u \in \mathscr{X}_\beta$$

each of which by (1) is a closed set. Hence $O \in \Sigma_2^0$ in the space B^ω.

Next we demonstrate a Σ_1^0 set O in a space A^ω which is complete for Σ_2^0 in a space B^ω where $A \subsetneq B$. Let $A = \{a,b\}$ and $B = \{a,b,c\}$. Let $X = \{ab, abab, ababab, \ldots\} \subset A^*$ and let $O = XA^\omega$. Then O is open. Each subset $O_A(u),\ u \in X$ is represented in B^ω as

$$O_A(u) = O_B(u) \cap O_B(ua) \cap O_B(ub) \cap O_B(uaa) \cap O_B(uab) \cap O_B(uba) \cap O_B(ubb) \cap \ldots$$

and

$$O = O_A(u_1) \cup O_A(u_2) \cup \ldots,\ u_i \in X$$

Hence O is a Σ_2^0 set in B^ω.

To show that O is Σ_2^0 complete for B^ω we use Proposition 2. O is not open in B^ω. Indeed, because otherwise, there exists a finite string u whose all possible continuations with letters from B are in O and that is a contradiction. O is also not closed in B^ω. To see this, note that the complement of O, $\overline{O}$ in A^ω is the set XA^ω where $X \subseteq A^*$ is given as $X = \{b, aa, abb, abaa, \ldots\}$. For O to be closed in B^ω, $\overline{O}$ should

be open in B^ω. This means that there should exist a finite string v whose all possible continuations with letters from B are in $\overline{O}$ which is again a contradiction.

Thus $O \notin \Sigma^0_1$ and $O \notin \Pi^0_1$ in B^ω and hence it is complete for Σ^0_2 in B^ω.

Next suppose C is a closed set in A^ω. We show how to represent C in B^ω. Let $\mathscr{U}_\beta$ be the indexed set of prefixes of C. Then C can be represented in B^ω as

$$C = \bigcap O_B(v),\ v \in \mathscr{U}_\beta$$

Each $O_B(v)$ is a closed set in B^ω and hence C being an arbitrary intersection of closed sets in B^ω is closed. Thus $C \in \Pi^0_1$ in A^ω remains Π^0_1 in B^ω. ∎

We generalise the above Lemma to the entire Borel hierarchy in the following theorem.

Theorem 1 *Let A and B be two alphabets such that $A \subsetneq B$. We have the following in the Borel hierarchy:*

1. *For $1 \leq \alpha < \omega$ and α odd,*
   - *(a) a set $X \in \Sigma^0_\alpha$ in the space A^ω jumps to $\Sigma^0_{\alpha+1}$ in the space B^ω*
   - *(b) a set $X \in \Pi^0_\alpha$ in the space A^ω remains Π^0_α in the space B^ω.*
2. *For $1 \leq \alpha < \omega$ and α even,*
   - *(a) a set $X \in \Sigma^0_\alpha$ in the space A^ω remains Σ^0_α in the space B^ω*
   - *(b) a set $X \in \Pi^0_\alpha$ in the space A^ω jumps to $\Pi^0_{\alpha+1}$ in the space B^ω.*
3. *For $\alpha \geq \omega$, a Σ^0_α (resp. Π^0_α) set remains Σ^0_α (resp. Π^0_α) on going from the space A^ω to B^ω. That is, the sets stabilise.*

Proof The proof is by induction on α. For the base case, $\alpha = 1$, the result follows from Lemma 1.

The inductive case is relatively straightforward, given the inductive structure of the Borel hierarchy. For convenience, we subscript the sets with A or B to denote whether they are sets in A^ω or B^ω respectively.

Suppose $1 < \alpha < \omega$ and α is odd. Then

$$\begin{aligned}\Sigma^0_{\alpha,X} =& \bigcup \Pi^0_{\alpha-1,X} \text{ [by definition]}\\ =& \bigcup \Pi^0_{\alpha,Y} \text{ [by induction hypothesis]}\\ =& \Sigma^0_{\alpha+1,Y}\end{aligned}$$

$$\begin{aligned}\Pi^0_{\alpha,X} =& \overline{\Sigma}^0_{\alpha,X} = \overline{\bigcup \Pi^0_{\alpha-1,X}} = \bigcap \overline{\Pi}^0_{\alpha-1,X} = \bigcap \Sigma^0_{\alpha-1,X} \text{ [by definition]}\\ =& \bigcap \Sigma^0_{\alpha-1,Y} \text{ [by induction hypothetis]}\\ =& \Pi^0_{\alpha,Y}\end{aligned}$$

Now, suppose $1 < \alpha < \omega$ and α is even. Then

$$\begin{aligned}\Sigma^0_{\alpha,X} =& \bigcup \Pi^0_{\alpha-1,X} \text{ [by definition]}\\ =& \bigcup \Pi^0_{\alpha-1,Y} \text{ [by induction hypothesis]}\\ =& \Sigma^0_{\alpha,Y}\end{aligned}$$

$$\begin{aligned}\Pi^0_{\alpha,X} &= \overline{\Sigma}^0_{\alpha,X} = \overline{\bigcup \Pi^0_{\alpha-1,X}} = \bigcap \overline{\Pi}^0_{\alpha-1,X} = \bigcap \Sigma^0_{\alpha-1,X} \text{ [by definition]} \\ &= \bigcap \Sigma^0_{\alpha,Y} \text{ [by induction hypothetis]} \\ &= \Pi^0_{\alpha+1,Y}\end{aligned}$$

Finally,

$$\Sigma^0_{\omega,X} = \bigcup_{n<\omega} \Sigma^0_{n,X} = \bigcup_{n<\omega} \Sigma^0_{n,Y} = \Sigma^0_{\omega,Y}$$

and

$$\Pi^0_{\omega,Y} = \overline{\Sigma}^0_{\omega,Y} = \Pi^0_{\omega,X}$$

■

The above result can be concisely summarised by Figure 1.

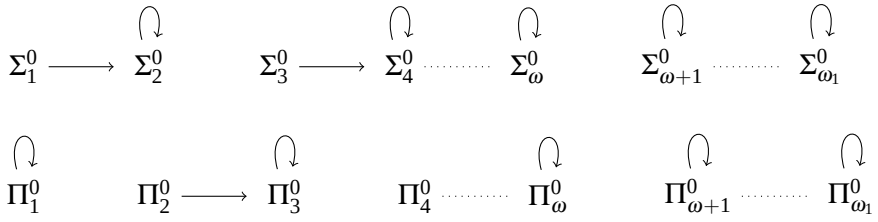


Figure 1: Jumps in the Borel hierarchy

4 Applications

The result we showed has interesting consequences in the fields of both formal verification and linguistics.

4.1 Formal verification

As we mentioned in the introduction, to formally verify a reactive system M (a piece of hardware or software which interacts with users/environment), we often model the system as a finite graph $G(M)$. Two players, the system player and the environment player then play an infinite game on $G(M)$. The goal of the system player is to meet a certain specification on all plays on $G(M)$ and that of the environment player is to exibit a play which does not meet it.

The result stated in this paper represents situations where the system player is unsure about the exact moves of the environment player. This shows that in such a situation, the system player might have to strategise at a higher level of the hierarchy in order to account for this uncertainty.

It can also be used to represent situations where the underlying model might change (expand). Let M be the original system and M' be the expanded system (which is generated from M by the addition of a module say). If the objective of the system player in $G(M)$ was to reach one of the states in some subset R of $G(M)$ (reachability) then it is enough for her to play positionally. However, in the bigger

graph $G(M')$ she not only has to reach R but also has to stay within the states of the original graph $G(M)$ in order to achieve the same objective. This is the Muller objective which is a level higher.

Example 2. Consider the example shown in Figure 2. Player 0 nodes have been depicted as ◯ and Player 1 nodes as □. Suppose initially the system is M and the objective of Player 1 in $G(M)$ is to reach v_3. Then the winning set is the set of all sequences in $V = \{v_0, v_1, v_2, v_3\}$ in which v_3 occurs in some position. That is, $Win = \{u \mid \exists i,\ u(i) = v_3\}$. This is a reachability condition where the reachability set $R = \{v_3\}$. To win, Player 0 can either play v_1 or v_2 from v_0 and hence both these strategies are winning strategies for her. Now suppose the system expands to M' where, in $G(M')$, it is possible for Player 1 to go to the new node v_4 from v_1. Also suppose *Win* remains the same. Then *Win* is no longer a reachability condition because then it would also include sequences involving the vertex v_4. It is rather a Muller condition where the Muller set $\mathscr{F} = \{\{v_0, v_1, v_2, v_3\}\}$. However, note that Player 0 does not have a winning strategy in this game. That is because to win, she has to visit vertex v_1 infinitely often from which Player 1 can force the play through v_4 infinitely often.

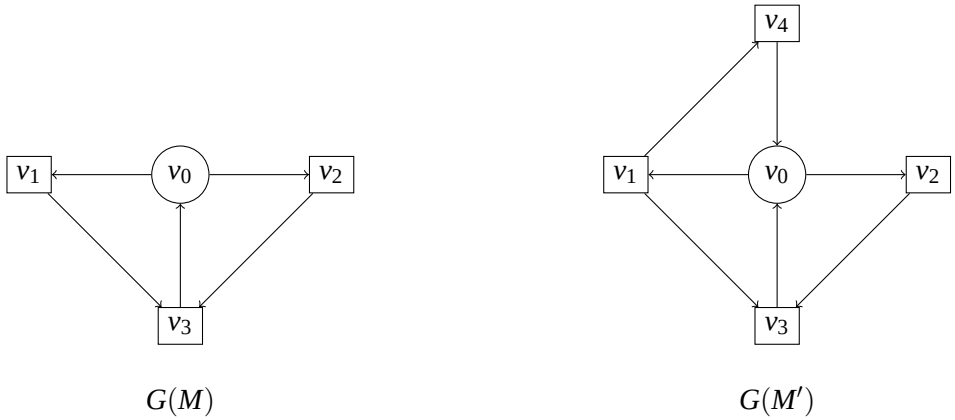


Figure 2: Jump from reachability to Muller

4.2 Linguistics

In [2] we demonstrated what seems to be a compelling similarity between human conversations and Banach-Mazur games. We showed how various conversational objectives correspond to various levels of the Borel hierarchy and how strategies of increasing complexity are called for to attain such objectives. Our result shows that when Player 1 is unsure about what Player 2 might say, it might be wise for her to strategise at a higher level to account for this uncertainty. She engages in a conversation, believing she is equipped with a strategy for all the situations the other player might put her into when suddenly the other player says something and she is left dumbfounded.

An example which still sticks in the memory of one of the authors after almost 20 years is the memorable line by Senator Lloyd Bentsen in his Vice-Presidential debate with Dan Quale in 1984. Quayle's strategy in the debate was to counter the perception that he was too inexperienced to have the job, and he did this by drawing similarities between his political career and former President John Kennedy's. Quayle seemed to be doing a good job in achieving his objective or winning condition, when Bentsen interrupted and said:

> Sir, I knew Jack Kennedy. I knew Jack Kennedy. And you, sir, are no Jack Kennedy.

Quayle's strategy at that point fell apart. He had no effective come back and by all accounts lost the debate handily.

The way we model this as follows. Building on [2], we take each move in a game to be a discourse which may be composed of several, even many clauses. Abstractly, we consider such discourses as sequences of basic moves, which we will be the alphabet. In a situation of incomplete information about the discourse moves, the set of moves (or the alphabet) of the Banach Mazur game being played by the players is different for the two players. Player 0 has an alphabet A (say) while Player 1 has an alphabet B such that $A \subsetneq B$. Player 0 may or may not be aware of this fact.

Thus, from the point of view of Player 0, if she is playing a Banach-Mazur game where she is unsure of the set of moves available to Player 1, it is better for her to strategise in such a way so as to account for this jump in the winning set. In other words, if Player 0's winning condition is at a level n (say) of the hierarchy, she is better off strategising for level $n+1$ given that she is unsure of Player 1's moves and given that a set at level n might undergo a jump to level $n+1$. Thus Quayle might have even won the debate had he strategiesed at a higher level expecting the unexpected.

References

[1] N. Asher, E. McReady & S. Paul (2012): *Strategic Politeness.* In: *LENLS 9.*

[2] N. Asher & S. Paul (2012): *Conversations as Banach-Mazur Games. Dialogue and Discourse (submitted).*

[3] D. Gale & F. M. Stewart (1953): *Infinite Games with Perfect Information. Annals of Mathematical Studies* 28, pp. 245–266.

[4] Erich Grädel, Wolfgang Thomas & Thomas Wilke, editors (2002): *Automata, Logics, and Infinite Games: A Guide to Current Research. Lecture Notes in Computer Science* 2500, Springer, doi:10.1007/3-540-36387-4.

[5] Y. Gurevich & L. Harrington (1982): *Trees, automata and games. Proceedings of the 14th Annual Symposium on Theory of Computing*, pp. 60–65, doi:10.1145/800070.802177.

[6] Akihiro Kanamori (2003): *The higher infinite : large cardinals in set theory from their beginnings*. Springer.

[7] A Kechris (1995): *Classical descriptive set theory*. Springer-Verlag, New York, doi:10.1007/978-1-4612-4190-4.

[8] A. W. Mostowski (1991): *Games with forbidden positions*. Technical Report, Instytut Matematyki, Universytet Gdanski, Poland.

[9] D. Perrin & J. E. Pin (1995): *Infinite Words - Automata, Semigroups, Logic and Games*. Elsevier, doi:10.1007/978-94-011-0149-3_3.

[10] Olivier Serre (2004): *Games with Winning Conditions of High Borel Complexity*. In: *ICALP*, pp. 1150–1162, doi:10.1016/j.tcs.2005.10.024.

[11] William W. Wadge (1983): *Reducibility and determinateness on the Baire space*. Ph.D. thesis, UC, Berkeley.

How to Be Both Rich and Happy: Combining Quantitative and Qualitative Strategic Reasoning about Multi-Player Games (Extended Abstract)

Nils Bulling
Clausthal University of Technology, Germany
bulling@in.tu-clausthal.de

Valentin Goranko
Technical University of Denmark, Denmark
vfgo@imm.dtu.dk

We propose a logical framework combining a game-theoretic study of abilities of agents to achieve *quantitative objectives* in multi-player games by optimizing payoffs or preferences on outcomes with a logical analysis of the abilities of players for achieving *qualitative objectives* of players, i.e., reaching or maintaining game states with desired properties. We enrich concurrent game models with payoffs for the normal form games associated with the states of the model and propose a quantitative extension of the logic ATL* enabling the combination of quantitative and qualitative reasoning.

1 Introduction

There are two rich traditions in studying strategic abilities of agents in multi-player games:

Game theory has been studying rational behavior of players, relevant for their achievement of *quantitative objectives*: optimizing payoffs (e.g., maximizing rewards or minimizing cost) or, more generally, preferences on outcomes. Usually, the types of games studied in game theory are one-shot normal form games, their (finitely or infinitely) repeated versions, and extensive form games.

Logic has been mostly dealing with strategic abilities of players for achieving *qualitative objectives*: reaching or maintaining outcome states with desired properties, e.g., winning states, or safe states, etc.

Among the most studied models in the logic tradition are *concurrent game models* [5, 21]. On the one hand they are richer than normal form games, as they incorporate a whole family of such games, each associated with a state of a transition system; but on the other hand, they are somewhat poorer because the outcomes of each of these normal form games, associated with a given state, are simply the successor states with their associated games, etc. whereas no payoffs, or even preferences on outcomes, are assigned. Thus, plays in concurrent game models involve a sequence of possibly different one-shot normal form games played in succession, and all that is taken into account in the purely logical framework are the properties – expressed by formulae of a logical language – of the states occurring in the play. Concurrent game models can also be viewed as generalization of (possibly infinite) extensive form games where cycles and simultaneous moves of different players are allowed, but no payoffs are assigned.

Put as a slogan, the game theory tradition is concerned with *how a player can become maximally rich, or how to pay as little cost as possible*, while the logic tradition – with *how a player can achieve a state of 'happiness', e.g. winning, or to avoid reaching a state of 'unhappiness' (losing) in the game*.

The most essential technical difference between qualitative and quantitative players' objectives is that the former typically refer to (a temporal pattern over) Boolean properties of game states on a given play and can be monitored locally whereas the latter are determined by the entire history of the play (accumulated payoffs) or even the whole play (its value, being a limit of average payoffs, or of discounted

F. Mogavero, A. Murano, and M.Y. Vardi (Eds.):
1st Workshop on Strategic Reasoning 2013 (SR'13)
EPTCS 112, 2013, pp. 33–41, doi:10.4204/EPTCS.112.8

accumulated payoffs). It is therefore generally computationally more demanding and costly to design strategies satisfying quantitative objectives or to verify their satisfaction under a given strategy of a player or coalition.

These two traditions have followed rather separate developments, with generally quite different agendas, methods and results, including, inter alia:

- on the purely qualitative side, *logics of games and multiagent systems*, such as the Coalition logic CL [21], the Alternating time temporal logic ATL [5], and variations of it, see e.g. [15], [18], etc., formalizing and studying qualitative reasoning in concurrent game models;
- some *single-agent and multi-agent bounded resource logics* [9, 3, 19] extending or modifying concurrent game models with some quantitative aspects by considering cost of agents' actions and reasoning about what players with bounded resources can achieve.
- extensions of qualitative reasoning (e.g., reachability and Büchi objectives) in multi-player concurrent games with 'semi-quantitative' aspects by considering a preference preorder on the set of qualitative objectives, see e.g., [6], [7], thereby adding payoff-maximizing objectives and thus creating a setting where traditional game-theoretic issues such as game value problems and Nash equlibria become relevant.
- deterministic or stochastic *infinite games on graphs*, with qualitative objectives: typically, reachability, and more generally – specified as ω-regular languages over the set of plays, see e.g. [4], [10], [12].
- on the purely quantitative side, first to mention *repeated games*, extensively studied in game theory (see e.g., [20]), which can be naturally treated as simple, one-state concurrent game models with accumulating payoffs paid to each player after every round and no qualitative objectives;
- from a more computational perspective, stochastic games with quantitative objectives on discounted, mean or total payoffs, in particular energy objectives, see e.g. [11].
- the conceptually different but technically quite relevant study of *counter automata, Petri nets, vector addition systems*, etc. – essentially a study of the purely quantitative single-agent case of concurrent game models (see e.g. [14]), where only accumulated payoffs but no qualitative objectives are taken into account and a typical problem is to decide reachability of payoff configurations satisfying formally specified arithmetic constraints from a given initial payoff configuration.

A number of other relevant references discuss the interaction between qualitative and quantitative reasoning in multi-player games, e.g. [22], [16], which we cannot discuss here due to space limitations.

This project purports to combine the two agendas in a common logical framework, by enriching concurrent game models with payoffs for the one-shot normal form games associated with the states, and thus enabling the combination of quantitative game-theoretic reasoning with the qualitative logical reasoning. Again, put as a slogan, our framework allows reasoning about whether/how a player can reach or maintain a state of 'happiness' while becoming, or remaining, as rich as (rationally) possible, or paying the least possible price on the way. The purpose of this extended abstract is to introduce and discuss a general framework of models and logics for combined quantitative and qualitative reasoning that would naturally cover each of the topics listed above, and to initiate a long term study on it.

2 Preliminaries

A **concurrent game model** [5] (CGM) $\mathscr{S} = (\mathsf{Ag}, \mathsf{St}, \{\mathsf{Act}_\mathsf{a}\}_{\mathsf{a}\in\mathsf{Ag}}, \{\mathsf{act}_\mathsf{a}\}_{\mathsf{a}\in\mathsf{Ag}}, \mathsf{out}, \mathsf{Prop}, \mathsf{L})$ comprises:

- a non-empty, fixed set of players $\mathsf{Ag} = \{1, \ldots, k\}$ and a set of actions $\mathsf{Act_a} \neq \emptyset$ for each $\mathsf{a} \in \mathsf{Ag}$. For any $A \subseteq \mathsf{Ag}$ we will denote $\mathsf{Act}_A := \prod_{\mathsf{a} \in A} \mathsf{Act_a}$ and will use $\overrightarrow{\alpha}_A$ to denote a tuple from Act_A. In particular, $\mathsf{Act_{Ag}}$ is the set of all possible action profiles in $\mathscr{S}$.
- a non-empty set of game states St.
- for each $\mathsf{a} \in \mathsf{Ag}$ a map $\mathsf{act_a} : \mathsf{St} \to \mathscr{P}(\mathsf{Act_a})$ setting for each state s the actions available to a at s.
- a transition function $\mathsf{out} : \mathsf{St} \times \mathsf{Act_{Ag}} \to \mathsf{St}$ that assigns the (deterministic) *successor (outcome) state* $\mathsf{out}(q, \overrightarrow{\alpha}_{\mathsf{Ag}})$ to every state q and action profile $\overrightarrow{\alpha}_{\mathsf{Ag}} = \langle \alpha_1, \ldots, \alpha_k \rangle$ such that $\alpha_{\mathsf{a}} \in \mathsf{act_a}(q)$ for every $\mathsf{a} \in \mathsf{Ag}$ (i.e., every α_{a} that can be executed by player a in state q).
- a set of atomic propositions Prop and a labelling function $\mathsf{L} : \mathsf{St} \to \mathscr{P}(\mathsf{Prop})$.

Thus, all players in a CGM execute their actions synchronously and the combination of these actions, together with the current state, determines the transition to a (unique) successor state in the CGM.

The **logic of strategic abilities** ATL^* (*Alternating-Time Temporal Logic*), introduced and studied in [5], is a logical system, suitable for specifying and verifying qualitative objectives of players and coalitions in concurrent game models. The main syntactic construct of ATL^* is a formula of type $\langle\langle C \rangle\rangle \gamma$, intuitively meaning: *"The coalition C has a collective strategy to guarantee the satisfaction of the objective γ on every play enabled by that strategy."* Formally, ATL^* is a multi-agent extension of the branching time logic CTL*, i.e., multimodal logic extending the linear-time temporal logic LTL– comprising the temporal operators **X** ("at the next state"), **G** ("always from now on") and **U** ("until") – with *strategic path quantifiers* $\langle\langle C \rangle\rangle$ indexed with coalitions C of players. There are two types of formulae of ATL^*, *state formulae*, which constitute the logic and that are evaluated at game states, and *path formulae*, that are evaluated on game plays. These are defined by mutual recursion with the following grammars, where $C \subseteq \mathsf{Ag}$, $p \in \mathsf{Prop}$: state formulae are defined by $\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle C \rangle\rangle \gamma$, and path formulae by $\gamma ::= \varphi \mid \neg\gamma \mid \gamma \wedge \gamma \mid \mathbf{X}\gamma \mid \mathbf{G}\gamma \mid \gamma \mathbf{U} \gamma$.

The logic ATL^* is very expressive and that comes at a high computational price: satisfiability and model checking are **2ExpTime**-complete. A computationally better behaved fragment is the logic ATL, which is the multi-agent analogue of CTL, only involving state formulae defined by the following grammar, for $C \subseteq \mathsf{Ag}$, $p \in \mathsf{Prop}$: $\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle C \rangle\rangle \mathbf{X}\varphi \mid \langle\langle C \rangle\rangle \mathbf{G}\varphi \mid \langle\langle C \rangle\rangle (\varphi \mathbf{U} \varphi)$. For this logic satisfiability and model checking are **ExpTime**-complete and **P**-complete, respectively. We will, however, build our extended logical formalism on the richer ATL^* because we will essentially need the path-based semantics for it.

Arithmetic Constraints. We define a simple language of arithmetic constraints to express conditions about the accumulated payoffs of players on a given play. For this purpose, we use a set $V_{\mathsf{Ag}} = \{v_{\mathsf{a}} \mid \mathsf{a} \in \mathsf{Ag}\}$ of special variables to refer to the accumulated payoffs of the players at a given state and denote by V_A the restriction of V_{Ag} to any group $A \subseteq \mathsf{Ag}$. The payoffs can be integers, rationals[1], or any reals. We denote the domain of possible values of the payoffs, assumed to be a subset of the reals $\mathbb{R}$, by $\mathbb{D}$ and use a set of constants symbols X, with $0 \in X$, for names of special real values (see further) to which we want to refer in the logical language.

For fixed sets X and $A \subseteq \mathsf{Ag}$ we build the set $T(X,A)$ of *terms over X and A* from $X \cup V_A$ by applying addition, e.g. $v_a + v_b$. An evaluation of a term $t \in T(X,A)$ is a mapping $\eta : X \cup V_A \to \mathbb{D}$. We write $\eta \models t$ to denote that t is *satisfied* under the evaluation η. Moreover, if some order of the elements $X \cup V_A$ is clear from context, we also represent an evaluation as a tuple from $\mathbb{D}^{|A|+|V_A|}$ and often assume that elements from X have their canonic interpretation. The set $\mathsf{AC}(X,A)$ of *arithmetic constraints* over X and A consists of all expressions of the form $t_1 * t_2$ where $* \in \{<, \leq, =, \geq, >\}$ and $t_1, t_2 \in T(X,A)$. We use $\mathsf{ACF}(X,A)$ to refer to the set of Boolean formulae over $\mathsf{AC}(X,A)$; e.g. $(t_1 < t_2) \wedge (t_2 \geq t_3) \in \mathsf{ACF}(X,A)$

[1] Note that models with rational payoffs behave essentially like models with integer payoffs, after once-off initial re-scaling.

for $t_1, t_2, t_3 \in T(X,A)$. We note that the language $\mathsf{ACF}(X,A)$ is strictly weaker than Presburger arithmetic, as it involves neither quantifiers nor congruence relations.

We also consider the set $\mathsf{APC}(X,A)$ of *arithmetic path constraints* being expressions of the type $w_a * c$ where $a \in \mathsf{Ag}$, $* \in \{<, \leq, =, \geq, >\}$ and $c \in X$. The meaning of w_a is to represent the value of the current play for the player a. That value can be defined differently, typically by computing the accumulated payoff over the entire play, by using a future discounting factor, or by taking the limit – if it exists – of the mean (average) accumulated payoff (cf. [20]). We note that the discounted, accumulated, mean or limit payoffs may take real values beyond the original domain of payoffs $\mathbb{D}$; so, we consider the domain for X to be a suitable closure of $\mathbb{D}$.

3 Concurrent Game Models with Payoffs and Guards

We now extend concurrent game models with utility values for every action profile applied at every state and with guards that determine which actions are available to a player at a given configuration, consisting of a state and a utility vector, in terms of arithmetic constraints on the utility of that player.

Definition 1 *A **guarded CGM with payoffs (GCGMP)** is a tuple* $\mathfrak{M} = (\mathscr{S}, \mathsf{payoff}, \{g_\mathsf{a}\}_{\mathsf{a}\in\mathsf{Ag}}, \{d_\mathsf{a}\}_{\mathsf{a}\in\mathsf{Ag}})$ *where* $\mathscr{S} = (\mathsf{Ag}, \mathsf{St}, \{\mathsf{Act}_\mathsf{a}\}_{\mathsf{a}\in\mathsf{Ag}}, \{\mathsf{act}_\mathsf{a}\}_{\mathsf{a}\in\mathsf{Ag}}, \mathsf{out}, \mathsf{Prop}, \mathsf{L})$ *is a CGM and:*

- $\mathsf{payoff} : \mathsf{Ag} \times \mathsf{St} \times \mathsf{Act}_{\mathsf{Ag}} \to \mathbb{D}$ *is a payoff function assigning at every state s and action profile applied at s a payoff to every agent. We write* $\mathsf{payoff}_\mathsf{a}(s, \overrightarrow{\alpha})$ *for* $\mathsf{payoff}(\mathsf{a}, s, \overrightarrow{\alpha})$.
- $g_\mathsf{a} : \mathsf{St} \times \mathsf{Act}_\mathsf{a} \to \mathsf{ACF}(X, \{a\})$*, for each player* $\mathsf{a} \in \mathsf{Ag}$*, is a guard function that assigns for each state* $s \in \mathsf{St}$ *and action* $\alpha \in \mathsf{Act}_\mathsf{a}$ *an arithmetic constraint formula* $g_\mathsf{a}(s,\alpha)$ *that determines whether* α *is available to* a *at the state s given the current value of* a*'s accumulated payoff. The guard must enable at least one action for* a *at s. Formally, for each state* $s \in \mathsf{St}$*, the formula* $\bigvee_{\alpha\in\mathsf{Act}_\mathsf{a}} g_\mathsf{a}(s,\alpha)$ *must be valid. Moreover, a guard* $g_\mathsf{a}(s,\alpha)$ *is called* state-based *if* $g_\mathsf{a}(s,\alpha) \in \mathsf{ACF}(X)$.
- $d_\mathsf{a} \in [0,1]$ *is a discount factor, for each* $\mathsf{a} \in \mathsf{Ag}$*, used in order to define realistically values of infinite plays for players or to reason about the asymptotic behavior of players' accumulated payoffs.*

The guard g_a refines the function act_a from the definition of a CGM, which can be regarded as a guard function assigning to every state and action a constant arithmetic constraint `true` or `false`. In our definition the guards assigned by g_a only depend on the current state and the current accumulated payoff of a. The idea is that when the payoffs are interpreted as costs, penalties or, more generally, consumption of resources the possible actions of a player would depend on her current availability of utility/resources.

Example 1 *Consider the GCGMP shown in Figure 1 with 2 players, I and II, and 3 states, where in every state each player has 2 possible actions, C (cooperate) and D (defect). The transition function is depicted in the figure. The normal form games associated with the states are respectively versions of the Prisoners Dilemma at state s_1, Battle of the Sexes at state s_2 and Coordination Game at state s_3.*

The guards for both players are defined at each state so that the player can apply any action if she has a positive current accumulated payoff, may only apply action C if she has accumulated payoff 0; and must play an action maximizing her minimum payoff in the current game if she has a negative accumulated payoff. The discounting factors are 1 and the initial payoffs of both players are 0.

Configurations, plays, and histories. Let $\mathfrak{M}$ be a GCGMP defined as above. A *configuration* (in $\mathfrak{M}$) is a pair $(s, \overrightarrow{u})$ consisting of a state s and a vector $\overrightarrow{u} = (u_1, \ldots, u_k)$ of currently accumulated payoffs, one for each agent, at that state. Hereafter we refer to *accumulated payoffs* as *utility*, at a given state. We define the set of possible configurations as $\mathsf{Con}(\mathfrak{M}) = \mathsf{St} \times \mathbb{D}^{|\mathsf{Ag}|}$. The *partial configuration transition function* is defined as $\widehat{\mathsf{out}} : \mathsf{Con}(\mathfrak{M}) \times \mathsf{Act}_{\mathsf{Ag}} \times \mathbb{N} \to \mathsf{Con}(\mathfrak{M})$ such that $\widehat{\mathsf{out}}((s, \overrightarrow{u}), \overrightarrow{\alpha}, l) = (s', \overrightarrow{u'})$ iff:

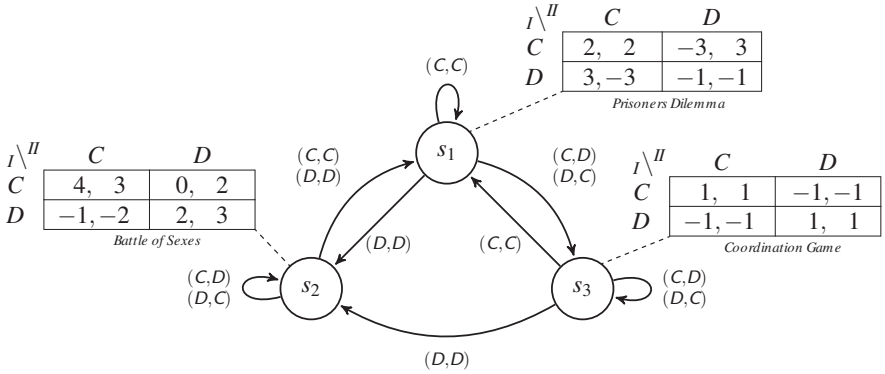


Figure 1: A simple GCGMP.

(i) $\mathsf{out}(s, \overrightarrow{\alpha}) = s'$ (s' is a successor of s if $\overrightarrow{\alpha}$ is executed).

(ii) assigning the value u_a to v_a satisfies the guard $g_{\mathsf{a}}(s, \alpha_{\mathsf{a}})$ for each $\mathsf{a} \in \mathsf{Ag}$, i.e. $u_a \models g_{\mathsf{a}}(s, \alpha_{\mathsf{a}})$ (each agent's move α_{a} is enabled at s by the respective guard g_{a} applied to the current accumulated utility value u_a).

(iii) $u'_{\mathsf{a}} = u_{\mathsf{a}} + d^l_{\mathsf{a}} \cdot \mathsf{payoff}_{\mathsf{a}}(s, \overrightarrow{\alpha})$ for all $\mathsf{a} \in \mathsf{Ag}$ (i.e., the utility values change according to the utility function and the discounting rate where l denotes the number of steps that took place).

A GCGMP $\mathfrak{M}$ with a designated initial configuration $(s_0, \overrightarrow{u_0})$ gives rise to a *configuration graph on* $\mathfrak{M}$ consisting of all configurations in $\mathfrak{M}$ reachable from $(s_0, \overrightarrow{u_0})$ by the configuration transition function. A *play* in a GCGMP $\mathfrak{M}$ is an infinite sequence $\pi = c_0\overrightarrow{\alpha_0}, c_1\overrightarrow{\alpha_1}, \ldots$ from $(\mathsf{Con}(\mathfrak{M}) \times \mathsf{Act})^\omega$ such that $c_n \in \widehat{\mathsf{out}}(c_{n-1}, \overrightarrow{\alpha}_{n-1})$ for all $n > 0$. The set of all plays in $\mathfrak{M}$ is denoted by $\mathsf{Plays}_{\mathfrak{M}}$. Given a play π we use $\pi[i]$ and $\pi[i,\infty]$ to refer to the ith element and to the subplay starting in position i of π, respectively. A *history* is any finite initial sequence $h = c_0\overrightarrow{\alpha_0}, c_1\alpha_1, \ldots, c_n \in (\mathsf{Con}(\mathfrak{M}) \times \mathsf{Act})^*\mathsf{Con}(\mathfrak{M})$ of a play in $\mathsf{Plays}_{\mathfrak{M}}$. The set of all histories is denoted by $\mathsf{Hist}_{\mathfrak{M}}$. For any history h we also define $h[i]$ as for plays and additionally $h[last]$ and $h[i,j]$ to refer to the last state on h and to the sub-history between i and j, respectively. Finally, we introduce functions $\cdot^c$, $\cdot^u$, and $\cdot^s$ which denote the projection of a given play or history to the sequence of its configurations, utility vectors, and states, respectively. For illustration, let us consider the play $\pi = c_0\overrightarrow{\alpha_0}, c_1\overrightarrow{\alpha_1}, \ldots$. We have that $\pi[i,\infty] = c_i\overrightarrow{\alpha_i}, c_{i+1}\overrightarrow{\alpha_{i+1}}, \ldots$; $\pi[i] = c_i\overrightarrow{\alpha}_i$; $\pi^c[i,\infty] = c_i, c_{i+1}, \ldots$; $\pi^c[i] = c_i$; $\pi^a[i] = \overrightarrow{\alpha_i}$; $\pi^u[i] = v_i$; and $\pi^s[i] = s_i$ where $c_i = (s_i, \overrightarrow{u_i})$.

Example 2 *Some possible plays starting from s_1 in Example 1 are given in the following where we assume that the initial accumulated payoff is 0 for both agents. We note that this implies that the first action taken by any agent is always C.*

1. *Both players cooperate forever:* $(s_1,0,0),(s_1,2,2),(s_1,4,4),\ldots$
2. *After the first round both players defect and the play moves to s_2, where player I chooses to defect whereas II cooperates. Then I must cooperate while II must defect but at the next round can choose any action, so a possible play is:* $(s_1,0,0),(s_1,2,2),(s_2,1,1),(s_2,0,-1),(s_2,0,1),(s_2,0,3),(s_2,0,5),\ldots$
3. *After the first round player I defects while II cooperates and the play moves to s_3, where they can get stuck indefinitely, until – if ever – they happen to coordinate, so a possible play is:* $(s_1,0,0),(s_1,2,2),(s_3,5,-2),(s_3,4,-3),(s_3,3,-4),\ldots(s_3,0,-7),(s_3,-1,-8),\ldots$.
   Note, however, that once player I reaches accumulated payoff 0 he may only apply C at that round, so if player II has enough memory or can observe the accumulated payoffs of I he can use the

opportunity to coordinate with I at that round by cooperating, thus escaping the trap at s_3 and making a sure transition to s_2.

4. *If, however, the guards did not force the players to play C when reaching accumulated payoffs 0, then both players could plunge into an endless misery if the play reaches s_3.*

Strategies. A *strategy* of a player a is a function $s_\mathsf{a} : \mathsf{Hist} \to \mathsf{Act}$ such that if $s_\mathsf{a}(h) = \alpha$ then $h^u[last]_\mathsf{a} \models g_\mathsf{a}(h^s[last], \alpha)$; that is, actions prescribed by a strategy must be enabled by the guard. Our definition of strategy is based on histories of configurations and actions, so it extends the notion of strategy from [5] where it is defined on histories of states, and includes strategies, typically considered e.g. in the study of repeated games, where often strategies prescribe to the player an action dependent on the previous action, or history of actions, of the other player(s). Such are, for instance, TIT-FOR-TAT or GRIM-TRIGGER in repeated Prisoners Dillemma; likewise for various card games, etc. Since our notion of strategy is very general, it easily leads to undecidable model checking problems. So, we also consider some natural restrictions, such as: *state-based*, *action-based* or *configuration-based*, *memoryless*, *bounded memory*, of *perfect recall* strategies [2]. Here we adopt a generic approach and assume that two classes of strategies $\mathscr{S}^p$ and $\mathscr{S}^o$ are fixed as parameters, with respect to which the proponents and opponents select their strategies, respectively. The proponent coalition A selects a $\mathscr{S}^p$-strategy s_A (i.e. one agreeing with the class $\mathscr{S}^p$) while the opponent coalition $\mathsf{Ag}\backslash A$ selects a $\mathscr{S}^o$-strategy $s_{\mathsf{Ag}\backslash A}$. The outcome play $\mathsf{outcome\_play}_\mathfrak{M}(c,(s_A,s_{\mathsf{Ag}\backslash A}),l)$ in a given GCGMP $\mathfrak{M}$ determines the play emerging from the execution of the (complete) strategy profile $(s_A,s_{\mathsf{Ag}\backslash A})$ from configuration c in $\mathfrak{M}$.

4 The Logic: Quantitative ATL*

We now extend the logic ATL^* to the logic QATL^* with atomic quantitative objectives being state or path arithmetic constraints over the players' accumulated payoffs. The semantics of QATL^* naturally extends the semantics of ATL^* over GCGMPs, but parameterised with the two classes of strategies $\mathscr{S}^p$ and $\mathscr{S}^o$.

Definition 2 (The logic QATL^*) *The language of* QATL^* *consists of* state formulae φ*, which constitute the logic, and* path formulae γ*, generated as follows, where $A \subseteq \mathsf{Ag}$, $\mathsf{ac} \in \mathsf{AC}$, $\mathsf{apc} \in \mathsf{APC}$, and $p \in \mathsf{Prop}$:*

$$\varphi ::= p \mid \mathsf{ac} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle A\rangle\rangle\gamma \text{ and } \gamma ::= \varphi \mid \mathsf{apc} \mid \neg\gamma \mid \gamma \wedge \gamma \mid \mathbf{X}\gamma \mid \mathbf{G}\gamma \mid \gamma\mathbf{U}\gamma.$$

Let $\mathfrak{M}$ be a GCGMP, c a configuration, $\varphi, \varphi_1, \varphi_2$ state-formulae, $\gamma, \gamma_1, \gamma_2$ path formulae, and $l \in \mathbb{N}$. Further, let $\mathscr{S}^p$ and $\mathscr{S}^o$ be two classes of strategies as described above. The semantics of the path constraints is specified according to the limit-averaging or discounting mechanism adopted for computing the value of a play for a player. Then the truth of a QATL^* *formula at a position of a configuration in $\mathfrak{M}$ is defined by mutual recursion on state and path formulae as follows:*

$\mathfrak{M}, c, l \models p$ *for* $p \in \mathsf{Prop}$ *iff* $p \in \mathsf{L}(c^s)$; $\mathfrak{M}, c, l \models \mathsf{ac}$ *for* $\mathsf{ac} \in \mathsf{AC}$ *iff* $c^u \models \mathsf{ac}$,

$\mathfrak{M}, c, l \models \langle\langle A\rangle\rangle\gamma$ *iff there is a collective $\mathscr{S}^p$-strategy s_A for A such that for all collective $\mathscr{S}^o$-strategies $s_{\mathsf{Ag}\backslash A}$ for $\mathsf{Ag}\backslash A$ we have that $\mathfrak{M}, \mathsf{outcome\_play}^\mathfrak{M}(c,(s_A,s_{\mathsf{Ag}\backslash A}),l), l \models \gamma$.*

$\mathfrak{M}, \pi, l \models \varphi$ *iff* $\mathfrak{M}, \pi[0], l \models \varphi$; $\mathfrak{M}, \pi, l \models \mathsf{apc}$ *iff* $\pi^u, l \models \mathsf{apc}$ *for* $\mathsf{apc} \in \mathsf{APC}$.

$\mathfrak{M}, \pi, l \models \mathbf{G}\gamma$ *iff* $\mathfrak{M}, \pi[i], l+i \models \gamma$ *for all* $i \in \mathbb{N}_0$,

$\mathfrak{M}, \pi, l \models \mathbf{X}\gamma$ *iff* $\mathfrak{M}, \pi[1], l+1 \models \gamma$,

$\mathfrak{M}, \pi, l \models \gamma_1\mathbf{U}\gamma_2$ *iff there is* $j \in \mathbb{N}_0$ *such that* $\mathfrak{M}, \pi[j], l+j \models \gamma_2$ *and* $\mathfrak{M}, \pi[i], l+i \models \gamma_1$ *for all* $0 \leq i < j$.

Ultimately, we define $\mathfrak{M}, c \models \varphi$ as $\mathfrak{M}, c, 1 \models \varphi$. Moreover, if not clear from context, we also write $\models_{(\mathscr{S}^p, \mathscr{S}^o)}$ for $\models$.

[2] We note that all strategies need to be consistent with the guards, so state-based strategies are only applicable in models where the guards only take into account the current state, but not the accumulated payoffs.

The semantics presented above extends the standard semantics for ATL* and is amenable to various refinements and restrictions, to be studied further. For instance, if appropriate, an alternative semantics can be adopted, based on irrevocable strategies [1] or, more generally, on strategy contexts [8] or other mechanisms for strategy commitment and release [2]. Also, the nested operators as defined here access the accumulated utility values and require plays to be infinite. Similarly to [9], one can consider variants of these settings which may yield decidable model checking and better complexity results.

As the logic QATL* extends ATL*, it allows expressing all purely qualitative ATL* properties. It can also express purely quantitative properties, e.g.: $\langle\langle\{\mathsf{a}\}\rangle\rangle\mathbf{G}(v_{\mathsf{a}} > 0)$ meaning "Player a has a strategy to maintain his accumulated payoff to be always positive", or $\langle\langle A\rangle\rangle(w_a \geq 3)$ meaning "The coalition A has a strategy that guarantees the value of the play for player a to be at least 3". Moreover, QATL* can naturally express combined qualitative and quantitative properties, e.g. $\langle\langle\{\mathsf{a},\mathsf{b}\}\rangle\rangle((v_{\mathsf{a}}+v_{\mathsf{b}} \geq 1)\mathbf{U}p))$, etc.

Example 3 *The following* QATL* *state formulae are true at state s_1 of the GCGMP in Example 1, where p_i is an atomic proposition true only at state s_i, for each $i = 1,2,3$:*
(i) $\langle\langle\{I,II\}\rangle\rangle\mathbf{F}(p_1 \wedge v_I > 100 \wedge v_{II} > 100) \wedge \langle\langle\{I,II\}\rangle\rangle\mathbf{XX}\langle\langle\{II\}\rangle\rangle(\mathbf{G}(p_2 \wedge v_I = 0) \wedge \mathbf{F}\, v_{II} > 100)$.
(ii) $\neg\langle\langle\{I\}\rangle\rangle\mathbf{G}(p_1 \vee v_I > 0) \wedge \neg\langle\langle\{I,II\}\rangle\rangle\mathbf{F}(p_3 \wedge \mathbf{G}(p_3 \wedge (v_I + v_{II} > 0)))$.

5 (Un)Decidability: Related Work and Some Preliminary Results

Generally, the GCGMP models are too rich and the language of QATL* is too expressive to expect computational efficiency, or even decidability, of either model checking or satisfiability testing. Some preliminary results and related work show that model checking of QATL* in GCGMPs is undecidable under rather weak assumptions, e.g. if the proponents or the opponents can use memory-based strategies. These undecidability results are not surprising as GCGMPs are closely related to Petri nets and vector addition systems and it is known that model checking over them is generally undecidable. In [13], for example, this is shown for fragments of CTL and (state-based) LTL over Petri nets. Essentially, the reason is that the logics allow to encode a "test for zero"; for Petri nets this means to check whether a place contains a token or not. In our setting undecidability follows for the same reason, and we will sketch some results below.

Undecidability results. The logic QATL restricts QATL* in the same way as ATL restricts ATL*, due to lack of space we skip the formal definition. As a first result we show that model checking QATL is undecidable even if only the proponents are permitted to use perfect recall strategies and the opponents are bound to memoryless strategies. More formally, let S^{pr} denote the class of perfect recall state-based strategies and S^{m} the class of memoryless state-based strategies. That is, strategies of the former class are functions of type $\mathsf{St}^* \to \mathsf{Act}$ and of the latter class functions of type $\mathsf{St} \to \mathsf{Act}$.

Undecidability can be shown using ideas from e.g. [9, 13]. Here, we make use of the construction of [9] to illustrate the undecidability by simulating a two-counter machine (TCM). A TCM [17] can be considered as a transition system equipped with two integer counters that enable/disable transitions. Each step of the machine depends on the current state, symbol on the tape, and the counters, whether they are zero or not. After each step the counters can be incremented $(+1)$, or decremented (-1) , the latter only if the respective counter is not zero. A TCM is essentially a (nondeterministic) push-down automaton with two stacks and exactly two stack symbols (one of them is the initial stack symbol) and has the same computation power as a Turing machine (cf. [17]). A *configuration* is a triple (s,w_1,w_2) describing the current state (s), the value of counter 1 (w_1) and of counter 2 (w_2). A *computation* δ is a sequence of subsequent configurations effected by transitions.

For the simulation, we associate each counter with a player. The player's accumulated payoff encodes the counter value; actions model the increment/decrement of the counters; guards ensure that the actions respect the state of the counters. The accepting states of the two-counter machine are encoded by a special proposition *halt*. Now, the following lemma stating the soundness of the simulation can be proved:

Lemma 1 (Reduction) *For any two-counter machine A we can construct a finite GCGMP $\mathfrak{M}^A$ with two players and proposition* halt *such that the following holds: A halts on the empty input iff $\mathfrak{M}^A$ contains a play π with $\pi^c = (s^0,(v_1^0,v_2^0))(s^1,(v_1^1,v_2^1))\ldots$ such that there exists $j \in \mathbb{N}$ with* $\mathsf{halt} \in \mathsf{L}(s^j)$.

The next theorem gives two cases for which the model checking problem is undecidable. By the previous Lemma we have to ensure that the halting state is reached which can be expressed by $\langle\langle 1\rangle\rangle \mathbf{F}\mathsf{halt}$. We can also use purely state-based guards and encode the consistency checks in the formula as follows: $\langle\langle 1\rangle\rangle(v_1 \geq 0 \wedge v_2 \geq 0 \wedge e_1 \rightarrow v_a = 0 \wedge e_2 \rightarrow v_2 = 0)\mathbf{U}\mathsf{halt}$ where the proposition e_i is added to the model to indicate that the value of counter i is zero. Not that this information is static and obtained from the transition relation of the automaton.

Proposition 1 *Model checking the logic* QATL *is undecidable, even for the 2 agent case and no nested cooperation modalities, where $\mathscr{S}^p = \mathsf{S}^{pr}$ and $\mathscr{S}^o = \mathsf{S}^m$. This does even hold either for formulae not involving arithmetic constraints, or for state-based guards.*

Restoring decidability. There are some natural semantic and syntactic restrictions of QATL* where decidability may be restored; these include for instance, the enabling of only memoryless strategies, imposing non-negative payoffs, constraints on the transition graph of the model, bounds on players utilities etc. For instance, the main reason for the undecidability result above is the possibility for negative payoffs that allow for decrementing the accumulated payoffs and thus simulating the TCM operations. Therefore, a natural restriction in the quest for restoring decidability is to consider only GCGMP models with non-negative payoffs. In this case the accumulated payoffs increase monotonically over every play of the game, and therefore the truth values of every arithmetic constraint occurring in the guards and in the formula eventually stabilize in a computable way, which in the long run reduces the model checking of any QATL-formula in an GCGMP to a model checking of an ATL-formula in a CGM. One can thus obtain decidability of the model checking of the logic QATL in finite GCGMP with non-negative payoffs and perfect information. We will discuss these and other decidability results in a future work, where we will also consider restrictions similar to [9].

6 Concluding Remarks

This paper proposes a long-term research agenda bringing together issues, techniques and results from several research fields. It aims at bridging the two important aspects of reasoning about objectives and abilities of players in multi-player games: quantitative and qualitative, and eventually providing a uniform framework for strategic reasoning in multi-agent systems.

Acknowledgements: We thank the anonymous referees for detailed and helpful comments and additional references.

References

[1] T. Ågotnes, V. Goranko & W. Jamroga (2007): *Alternating-time Temporal Logics with Irrevocable Strategies*. In D. Samet, editor: *Proceedings of TARK XI*, pp. 15–24, doi:10.1145/1324249.1324256.

[2] T. Ågotnes, V. Goranko & W. Jamroga (2008): *Strategic Commitment and Release in Logics for Multi-Agent Systems (Extended abstract)*. Technical Report IfI-08-01, Clausthal University of Technology.

[3] Natasha Alechina, Brian Logan, Nguyen Hoang Nga & Abdur Rakib (2011): *Logic for coalitions with bounded resources*. *J. Log. Comput.* 21(6), pp. 907–937, doi:10.1093/logcom/exq032.

[4] Luca de Alfaro, Thomas A. Henzinger & Orna Kupferman (2007): *Concurrent reachability games*. *Theor. Comput. Sci.* 386(3), pp. 188–217, doi:10.1016/j.tcs.2007.07.008.

[5] R. Alur, T. A. Henzinger & O. Kupferman (2002): *Alternating-Time Temporal Logic*. *Journal of the ACM* 49, pp. 672–713, doi:10.1145/585265.585270.

[6] Patricia Bouyer, Romain Brenguier, Nicolas Markey & Michael Ummels (2011): *Nash Equilibria in Concurrent Games with Büchi Objectives*. In S. Chakraborty & A. Kumar, editors: *FSTTCS'2011 LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, pp. 375–386, doi:10.4230/LIPIcs.FSTTCS.2011.375.

[7] Patricia Bouyer, Romain Brenguier, Nicolas Markey & Michael Ummels (2012): *Concurrent Games with Ordered Objectives*. In L. Birkedal, editor: *Proc. of FoSSaCS'2012, Springer LNCS, vol. 7213*, pp. 301–315, doi:10.1007/978-3-642-28729-9_20.

[8] T. Brihaye, A. Da Costa, F. Laroussinie & N. Markey (2008): *ATL with Strategy Contexts and Bounded Memory*. Technical Report LSV-08-14, ENS Cachan, doi:10.1007/978-3-540-92687-0_7.

[9] Nils Bulling & Berndt Farwer (2010): *On the (Un-)Decidability of Model-Checking Resource-Bounded Agents*. In H. Coelho & M. Wooldridge, editors: *Proc. of ECAI 2010*, IOS Press, Amsterdam, pp. 567–572, doi:10.3233/978-1-60750-606-5-567.

[10] Krishnendu Chatterjee, Luca de Alfaro & Thomas A. Henzinger (2011): *Qualitative concurrent parity games*. *ACM Trans. Comput. Log.* 12(4), p. 28, doi:10.1145/1970398.1970404.

[11] Krishnendu Chatterjee & Laurent Doyen (2012): *Energy parity games*. *Theor. Comput. Sci.* 458, pp. 49–60, doi:10.1016/j.tcs.2012.07.038.

[12] Krishnendu Chatterjee & Thomas A. Henzinger (2012): *A survey of stochastic ω-regular games*. *J. Comput. Syst. Sci.* 78(2), pp. 394–413. Available at `http://dx.doi.org/10.1016/j.jcss.2011.05.002`.

[13] Javier Esparza: *Decidability of Model Checking for Infinite-State Concurrent Systems*. *Acta Informatica* 34, pp. 85–107, doi:10.1007/s002360050074.

[14] Javier Esparza (1998): *Decidability and complexity of Petri net problems - an Introduction*. In: *In Lectures on Petri Nets I: Basic Models*, Springer-Verlag, pp. 374–428, doi:10.1007/3-540-65306-6_20.

[15] Valentin Goranko & Wojciech Jamroga (2004): *Comparing Semantics of Logics for Multi-agent Systems*. *Synthese* 139(2), pp. 241–280, doi:10.1023/B:SYNT.0000024915.66183.d1.

[16] Erich Grädel & Michael Ummels (2008): *Solution Concepts and Algorithms for Infinite Multiplayer Games*. In Krzysztof Apt & Robert van Rooij, editors: *New Perspectives on Games and Interaction*, *Texts in Logic and Games* 4, Amsterdam University Press, pp. 151–178. Available at `http://www.logic.rwth-aachen.de/~ummels/knaw07.pdf`.

[17] JE Hopcroft & JD Ullman (1979): *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, Massachusetts, doi:10.1145/568438.568455.

[18] W. Jamroga & T. Ågotnes (2007): *Constructive Knowledge: What Agents Can Achieve under Incomplete Information*. *Journal of Applied Non-Classical Logics* 17(4), pp. 423–475, doi:10.3166/jancl.17.423-475.

[19] Dario Della Monica, Margherita Napoli & Mimmo Parente (2011): *On a Logic for Coalitional Games with Priced-Resource Agents*. *Electr. Notes Theor. Comput. Sci.* 278, pp. 215–228, doi:10.1016/j.entcs.2011.10.017.

[20] M. Osborne & A. Rubinstein (1994): *A Course in Game Theory*. MIT Press.

[21] M. Pauly (2002): *A Modal Logic for Coalitional Power in Games*. *J. of Logic and Computation* 12(1), pp. 149–166, doi:10.1093/logcom/12.1.149.

[22] Sophie Pinchinat (2007): *A Generic Constructive Solution for Concurrent Games with Expressive Constraints on Strategies*. In K. Namjoshi et al, editor: *Proc. of ATVA'2007, Springer LNCS, vol. 4762*, pp. 253–267, doi:10.1007/978-3-540-75596-8_19.

Lossy Channel Games under Incomplete Information

Rayna Dimitrova
Saarland University, Germany
dimitrova@cs.uni-saarland.de

Bernd Finkbeiner
Saarland University, Germany
finkbeiner@cs.uni-saarland.de

In this paper we investigate lossy channel games under incomplete information, where two players operate on a finite set of unbounded FIFO channels and one player, representing a system component under consideration operates under incomplete information, while the other player, representing the component's environment is allowed to lose messages from the channels. We argue that these games are a suitable model for synthesis of communication protocols where processes communicate over unreliable channels. We show that in the case of finite message alphabets, games with safety and reachability winning conditions are decidable and finite-state observation-based strategies for the component can be effectively computed. Undecidability for (weak) parity objectives follows from the undecidability of (weak) parity perfect information games where only one player can lose messages.

1 Introduction

Lossy channel systems (LCSs), which are finite systems communicating via unbounded lossy FIFO channels, are used to model communication protocols such as link protocols, a canonical example of which is the Alternating Bit Protocol. The decidability of verification problems for LCSs has been well studied and a large number of works have been devoted to developing automatic analysis techniques. In the control and synthesis setting, where games are the natural computational model, this class of systems has not yet been so well investigated. In [1], Abdulla et al. establish decidability of two-player safety and reachability games where one (or both) player has downward-closed behavior (e.g., can lose messages), which subsumes games with lossy channels where one player (i.e., the environment) can lose messages. They, however, assume that the game is played under perfect information, which assumption disregards the fact that a process has no access to the local states of other processes or that it has only limited information about the contents of the channels. To the best of our knowledge, games under incomplete information where the players operate on unbounded unreliable channels have not been studied so far.

We define *lossy channel games under incomplete information* and show that in the case of finite message alphabets, games with safety and reachability winning conditions are decidable and finite-state observation-based strategies for the player who has incomplete information can be effectively computed.

Algorithms for games under incomplete information carrying out an explicit knowledge based subset construction [9] are not directly applicable to infinite-state games. Symbolic approaches [4] are effective for restricted classes of infinite-state games like discrete games on rectangular automata [5]. The symbolic algorithms that we present in this paper rely on the monotonicity of lossy channel systems w.r.t. the subword ordering, which is a well-quasi ordering (WQO). It is well known that upward and downward-closed sets of words used in the analysis of lossy channel systems can be effectively represented by finite sets of minimal elements and simple regular expressions [2], respectively. Unsurprisingly, the procedures for solving lossy channel games under incomplete information that we develop manipulate sets of sets of states. Thus, our termination arguments rely on the fact that the subword ordering is in fact a better-quasi ordering (BQO) [7, 8], a stronger notion than WQO that is preserved by the powerset operation [6].

F. Mogavero, A. Murano, and M.Y. Vardi (Eds.):
1st Workshop on Strategic Reasoning 2013 (SR'13)
EPTCS 112, 2013, pp. 43–51, doi:10.4204/EPTCS.112.9

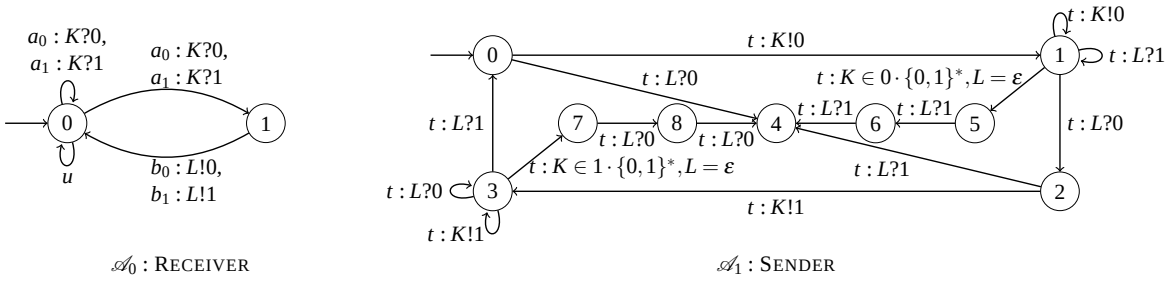


Figure 1: A communication protocol with partially specified RECEIVER process. For process RECEIVER we have $\Sigma_0 = \{a_0, a_1, b_0, b_1, u\}$ and $\Sigma_\exists = \{b_0, b_1\}$. The property that the implementation must satisfy is that location 4 in SENDER is not reachable, i.e., the receiver does not acknowledge messages that have not been sent, and once all messages and acknowledgements from previous phases have been consumed, the receiver can only send one delayed acknowledgement. Note that by using an extra channel and an extra location in process RECEIVER we can ensure that the error location is in process RECEIVER.

2 Lossy Channel Games under Incomplete Information

Lossy channel systems are asynchronous distributed systems composed of finitely many finite-state processes communicating through a finite set of unbounded FIFO channels that can nondeterministically lose messages. We consider *partially specified lossy channel systems*, where the term partially specified refers to the fact that we consider a second ("friendly") type of nondeterminism, in addition to the ("hostile") one due to the model. More specifically, this second type of nondeterminism models *unresolved implementation decisions* that can be resolved in a favorable way. We consider the case when these decisions are within a single process, and thus we can w.l.o.g. assume that the system consist of only two processes: the process under consideration and the parallel composition of the remaining processes.

Definition 1. A *partially specified lossy channel system (LCS)* is a tuple $\mathscr{L} = (\mathscr{A}_0, \mathscr{A}_1, C, M, \Sigma_0, \Sigma_1, \Sigma_\exists)$, where for each *process identifier* $p \in \{0,1\}$, $\mathscr{A}_p$ is a finite automaton describing the behavior of process p, C is a finite set of *channels*, M is a finite set of *messages*, $\Sigma = \Sigma_0 \dot\cup \Sigma_1$ is the union of the disjoint finite sets of *transition labels* for the two processes, and $\Sigma_\exists \subseteq \Sigma_0$ is a subset of the labels of the *partially specified process* $\mathscr{A}_0$. The automaton $\mathscr{A}_p = (Q_p, q_p^0, \delta_p)$ for a process p consists of a finite set Q_p of *control locations*, an *initial location* q_p^0 and a finite set δ of *transitions* of the form (q, a, Gr, Op, q'), where $q, q' \in Q_p$, $a \in \Sigma_p$, $Gr : C \to \{true, (= \varepsilon), \in (m \cdot M^*) \mid m \in M\}$ and $Op : C \to \{!m, ?m, nop \mid m \in M\}$. Intuitively, the function Gr maps each channel to a guard, which can be an emptiness test, a test of the letter at the head of the channel or the trivial guard *true*. The function Op gives the update operation for the respective channel, which is either a write, a read or *nop*, which leaves the channel unchanged.

Example. Fig.1 depicts a partially specified protocol consisting of two processes, SENDER and RECEIVER, communicating over the unreliable channels K and L. Process SENDER sends messages to RECEIVER over channel K and RECEIVER acknowledges the receipt of a message using channel L. Note that we use guards that test channels for emptiness or test the first letter of their contents.

The two processes are represented as nondeterministic finite-state automata. Process SENDER essentially runs the Alternating Bit Protocol. Process RECEIVER, however, is only *partially specified*: its alphabet of transition labels $\Sigma_0 = \{a_0, a_1, b_0, b_1, u\}$ is partitioned according to the unresolved decisions in the process specification: The subset $\Sigma_\exists = \{b_0, b_1\}$ of controllable transition labels specifies the unresolved implementation decisions, namely what bit to be sent on channel L at location 1.

The property that the protocol must satisfy is encoded as the unreachability of location 4 in process SENDER. However, the automata can easily be augmented (with an extra channel and an error location in process RECEIVER) in a way that the error location is in process RECEIVER. The property states that:

1. the receiver does not acknowledge messages that have not been sent, that is in location 2 in SENDER the language of L is 0^* and in location 0 in SENDER the language of L is 1^*,
2. once all messages and acknowledgements trailing from previous phases have been consumed (or lost), the number of delayed acknowledgements the receiver can send is bounded by one.

A *configuration* $\gamma = (q_0, q_1, w)$ of $\mathscr{L}$ is a tuple of the locations of the two processes and a function $w : C \to M^*$ that maps each channel to its contents. The *initial configuration* of $\mathscr{L}$ is $\gamma^0 = (q_0^0, q_1^0, \varepsilon)$, where $\varepsilon(c) = \varepsilon$ for each $c \in C$. The set of possible channel valuations is $W = \{w \mid w : C \to M^*\}$.

The *strong labeled transition relation* $\to \subseteq (Q_0 \times Q_1 \times W) \times \Sigma \times (Q_0 \times Q_1 \times W)$ of $\mathscr{L}$ consists of all tuples $((q_0, q_1, w), a, (q_0', q_1', w'))$ (denoted $(q_0, q_1, w) \xrightarrow{a} (q_0', q_1', w')$) such that if $a \in \Sigma_p$, then $q_{1-p}' = q_{1-p}$ and there is a transition $(q_p, a, Gr, Op, q_p') \in \delta$ such that for each $c \in C$ all of the following conditions hold: (1) if $Gr(c) = (\in m \cdot M^*)$ then $w(c) \in m \cdot M^*$, (2) if $Gr(c) = (= \varepsilon)$ then $w(c) = \varepsilon$, (3) if $Op(c) = !m$, then $w'(c) = w(c) \cdot m$, (4) if $Op(c) = ?m$, then $m \cdot w'(c) = w(c)$, and (5) if $Op(c) = nop$, then $w'(c) = w(c)$.

Let $\preceq$ denote the (not necessarily contiguous) subword relation on M^* and let us define its extension to elements of W as follows: $w_1 \preceq w_2$ for $w_1, w_2 \in W$ iff $w_1(c) \preceq w_2(c)$ for every $c \in C$.

The *weak labeled transition relation* $\Rightarrow \subseteq (Q_0 \times Q_1 \times W) \times \Sigma \times (Q_0 \times Q_1 \times W)$ for $\mathscr{L}$ is defined as follows: $(q_0, q_1, w) \stackrel{a}{\Rightarrow} (q_0', q_1', w')$ iff there exist w_1 and w_2 such that $w_1 \preceq w$ and $w' \preceq w_2$ and $(q_0, q_1, w_1) \xrightarrow{a} (q_0', q_1', w_2)$, i.e., the channels can lose messages before and after the actual transition.

Definition 2 (LC-game structure with incomplete information)**.** Let $\mathscr{L} = (\mathscr{A}_0, \mathscr{A}_1, C, M, \Sigma_0, \Sigma_1, \Sigma_\exists)$ be a partially specified LCS, and $C_{obs} \subseteq C$ be a set of *observable channels* that includes the set of all channels occurring in guards or read operations in $\mathscr{A}_0$. The *lossy channel game structure with incomplete information* for $\mathscr{L}$ and C_{obs} is $\mathscr{G}(\mathscr{L}, C_{obs}) = (S, I, \to_g, C, M, \Sigma_0, \Sigma_1, \Sigma_\exists, C_{obs})$, where:

- The set of *states* of $\mathscr{G}$ is $S = \{0,1\} \times Q_0 \times Q_1 \times W$. The first component p of a state (p, q_0, q_1, w) identifies the process to be executed and the remaining ones encode the current configuration of $\mathscr{L}$. The set of initial states of $\mathscr{G}$ is $I = \{(p, q_0, q_1, w) \mid p \in \{0,1\},\ q_0 = q_0^0,\ q_1 = q_1^0,\ w = \varepsilon\}$.
- The labeled transition relations $\to_g \subseteq S \times \Sigma \times S$ and $\Rightarrow_g \subseteq S \times \Sigma \times S$ of $\mathscr{G}$ are defined as follows: for states $s = (p, q_0, q_1, w)$ and $s' = (p', q_0', q_1', w')$ and $a \in \Sigma$ we have $s \xrightarrow{a}_g s'$ iff $a \in \Sigma_p$ and $(q_0, q_1, w) \xrightarrow{a} (q_0', q_1', w')$, and we have $s \stackrel{a}{\Rightarrow}_g s'$ iff $a \in \Sigma_p$ and $(q_0, q_1, w) \stackrel{a}{\Rightarrow} (q_0', q_1', w')$.

Remark. The first component of states in S is used to model the interleaving semantics and is updated nondeterministically in the transition relation $\to_g$ (and $\Rightarrow_g$). For simplicity, in Definition 2 we do not make any assumptions about the nondeterministic choice of which process to be executed. One natural assumption one might want to make is that the selected process must have at least one transition enabled in the current state. This and other restrictions can be easily imposed in the above model.

For the rest of the paper, let $\mathscr{G} = \mathscr{G}(\mathscr{L}, C_{obs}) = (S, I, \to_g, C, M, \Sigma_0, \Sigma_1, \Sigma_\exists, C_{obs})$ be the LC-game structure with incomplete information for a partially specified LCS $\mathscr{L}$ and observable channels C_{obs}.

Player$_\exists$ plays the game under incomplete information, observing only certain components of the current state of the game. Let $H_{obs} = C_{obs} \to (M \cup \{\varepsilon\})$ and $Obs = \{0,1\} \times Q_0 \times H_{obs}$. The *observation function* $obs : S \to Obs$ maps each state $s = (p, q_0, q_1, w)$ in $\mathscr{G}$ to the tuple $obs(s) = (p, q_0, h)$ of state components observed by *Player*$_\exists$, where for each $c \in C_{obs}$, if $p = 1$, then $h(c) = \varepsilon$ and otherwise if $w(c) = \varepsilon$, then $h(c) = \varepsilon$ and if $w(c) = m \cdot w'$ for some $m \in M$ and $w' \in M^*$, then $h(c) = m$. That is, when

$p = 0$ we have for $c \in C_{obs}$ that $h(c)$ is the letter at the head of $w(c)$, when c is not empty. For $o \in Obs$, we denote with $States(o) = \{s \in S \mid obs(s) = o\}$ the set of states whose observation is o.

Let $S_0 = \{(p, q_0, q_1, w) \in S \mid p = 0\}$ be the states where process 0 is to be executed and $S_1 = S \setminus S_0$.

The game $\mathscr{G}$ is played by $Player_\exists$ and $Player_\forall$ who build up a play $s_0 a_0^\exists a_0 s_1 a_1^\exists a_1 \ldots$, which is sequence of alternating states in S, labels in $\Sigma_\exists^\perp = \Sigma_\exists \cup \{\perp\}$ and labels in Σ, starting with a state $s_0 \in I$. Each time the current state is in S_0, $Player_\exists$ has to choose a label from the set $\Sigma_\exists \cup \{\perp\}$, that is either a label from $\Sigma_\exists$ of a transition enabled in the current state, or can be the special element $\perp$ in case no transition with label in $\Sigma_\exists$ is enabled or if there exists an enabled transition with label from $\Sigma_0 \setminus \Sigma_\exists$.

Let $\mathsf{Enabled}(s) = \{a \in \Sigma_0 \mid \exists s'.\ s \xrightarrow{a}_g s'\}$. Note that for states $s_1, s_2 \in S_0$ with $obs(s_1) = obs(s_2) = o$ it holds that $\mathsf{Enabled}(s_1) = \mathsf{Enabled}(s_2)$, and, abusing notation, we denote this set with $\mathsf{Enabled}(o)$.

For an observation $o = (0, q_0, h)$, the set $Act_\exists(o) = (\mathsf{Enabled}(o) \cap \Sigma_\exists) \cup \{\perp \mid \mathsf{Enabled}(o) \cap \Sigma_\exists = \emptyset \text{ or } \mathsf{Enabled}(o) \cap (\Sigma_0 \setminus \Sigma_\exists) \neq \emptyset\}$ consists of the transition labels that $Player_\exists$ can choose in a set $s \in S_0$ with $obs(s) = o$. For a label $a^\exists \in \Sigma_\exists^\perp$, the set $Act_\forall(o, a^\exists) = (\{a^\exists\} \cap \Sigma_\exists) \cup (\mathsf{Enabled}(o) \setminus \Sigma_\exists)$ consists of the transition labels which $Player_\forall$ can choose when the current choice of $Player_\exists$ is $a^\exists$.

The play is built by $Player_\forall$ respecting the choices of $Player_\exists$ and the transition relation $\Rightarrow_g$. When $s_i \in S_0$, then $a_i^\exists \in Act_\exists(obs(s_i))$ is the transition label chosen by $Player_\exists$ after the play prefix $s_0 a_0^\exists a_0 s_1 a_1^\exists a_1 \ldots a_{i-1}^\exists a_{i-1} s_i$ and $a_i \in Act_\forall(obs(s_i), a_i^\exists)$. After $Player_\exists$ has made his choice, $Player_\forall$ resolves the remaining nondeterminism by choosing a_i and the successor state s_{i+1} to extend the play.

A *play* in $\mathscr{G}$ is a sequence $\pi = s_0 a_0^\exists a_0 s_1 a_1^\exists a_1 s_1 \ldots \in (S \cdot (\Sigma_\exists^\perp \cdot \Sigma \cdot S)^* \cup S \cdot (\Sigma_\exists^\perp \cdot \Sigma \cdot S)^\omega)$ such that $s_0 \in I$, for every $i \geq 0$ it holds that $s_i \overset{a_i}{\Rightarrow}_g s_{i+1}$, and if $s_i \in S_1$, then $a^\exists = \perp$, and if $s_i \in S_0$ then $a_i^\exists \in Act_\exists(obs(s_i))$ and $a_i \in Act_\forall(obs(s_i), a_i^\exists)$. A play π is finite iff $\mathsf{last}(\pi)$ has no successor in $\mathscr{G}$, where $\mathsf{last}(\pi) \in S$ is the last element of π. The set $\mathsf{Prefs}(\mathscr{G}) \subseteq S \cdot (\Sigma_\exists^\perp \cdot \Sigma \cdot S)^*$ consists of the finite prefixes of plays in $\mathscr{G}$, and we denote with $\mathsf{Prefs}_\exists(\mathscr{G}) = \{\pi \in \mathsf{Prefs}(\mathscr{G}) \mid \mathsf{last}(\pi) \in S_0\}$ the set of prefixes ending in S_0.

A *strategy for* $Player_\exists$ is a total function $f_\exists : \mathsf{Prefs}_\exists(\mathscr{G}) \to \Sigma_\exists^\perp$ such that $f_\exists(\pi) \in Act_\exists(obs(\mathsf{last}(\pi)))$.

The *outcome of a strategy* $f_\exists$ is the set of plays $\mathsf{Outcome}(f_\exists)$ such that $\pi = s_0 a_0^\exists a_0 s_1 a_1^\exists a_1 \ldots \in \mathsf{Outcome}(f_\exists)$ iff for every $i \geq 0$ with $s_i \in S_0$ it holds that $a_i^\exists = f_\exists(s_0 a_0^\exists a_0 s_1 a_1^\exists a_1 \ldots s_i)$.

We define a function $obs^+ : \mathsf{Prefs}_\exists(\mathscr{G}) \to (Obs \cdot \Sigma_0)^* \cdot Obs$ that maps a prefix in $\mathsf{Prefs}_\exists(\mathscr{G})$ to the sequence of state and action observations made by $Player_\exists$: $obs^+(s_0 a_0^\exists a_0 s_1 a_1^\exists a_1 \ldots s_n) = obs'(s_0) \cdot obs'(a_0) \cdot obs'(s_1) \cdot obs'(a_1) \ldots \cdot obs'(s_n)$, where for $s \in S$, we define $obs'(s) = obs(s)$ if $s \in S_0$ and $obs'(s) = \varepsilon$ otherwise, and for $a \in \Sigma$ we define $obs'(a) = a$ if $a \in \Sigma_0$ and $obs'(a) = \varepsilon$ otherwise.

We call a strategy $f_\exists$ for $Player_\exists$ obs^+*-consistent* if for every pair of prefixes π_1 and π_2 in $\mathsf{Prefs}_\exists(\mathscr{G})$ for which $obs^+(\pi_1) = obs^+(\pi_2)$ holds, it also holds that $f_\exists(\pi_1) = f_\exists(\pi_2)$.

We are interested in *finite-state* strategies for $Player_\exists$, that is, strategies that can be implemented as finite automata. A finite state obs^+-consistent strategy for $Player_\exists$ in $\mathscr{G}$ is one that can be represented as a finite automaton $\mathscr{M}_s = (Q_s, q_s^0, (Q_0 \times H_{obs}) \times (\Sigma_\exists^\perp \times \Sigma_0), \rho)$ with alphabet $(Q_0 \times H_{obs}) \times (\Sigma_\exists^\perp \times \Sigma_0)$, whose transition relation $\rho \subseteq (Q_s \times ((Q_0 \times H_{obs}) \times (\Sigma_\exists^\perp \times \Sigma_0)) \times Q_s)$ has the following properties:

- (i) for each $q \in Q_s$, $o \in Q_0 \times H_{obs}$, $a^\exists \in \Sigma_\exists^\perp$, $a \in \Sigma_0$, and $q_1', q_2' \in Q_s$, it holds that if $(q, (o, (a^\exists, a)), q_1') \in \rho$ and $(q, (o, (a^\exists, a)), q_2') \in \rho$, then $q_1' = q_2'$ (i.e., the transition relation ρ is deterministic),
- (ii) for each $q \in Q_s$ and $o \in Q_0 \times H_{obs}$ there exist $a^\exists \in \Sigma_\exists^\perp$, $a \in \Sigma_0$, $q' \in Q_s$ with $(q, (o, (a^\exists, a)), q') \in \rho$,
- (iii) if $(q, (o, (a^\exists, a_1)), q_1') \in \rho$ and $a_2 \in Act_\forall((0, o), a^\exists)$, then $(q, (o, (a^\exists, a_2)), q_2') \in \rho$ for some $q_2' \in Q_s$,
- (iv) if $(q, (o, (a_1^\exists, a_1)), q_1') \in \rho$ and $(q, (o, (a_2^\exists, a_2)), q_2') \in \rho$, then $a_1^\exists = a_2^\exists$.

The automaton $\mathscr{M}_s$ defines an obs^+-consistent strategy $f_\exists$ for $Player_\exists$. According to the properties of $\mathscr{M}_s$, for each $\pi \in \mathsf{Prefs}_\exists(\mathscr{G})$ with $obs^+(\pi) = o_0 a_0 o_1 a_1 \ldots o_{n-1} a_{n-1} o_n$ there exists a unique sequence $a_0^\exists a_1^\exists a_{n-1}^\exists \in {\Sigma_\exists^\perp}^n$ such that there is a run of $\mathscr{M}_s$ (also unique) on the word $o_0 a_0^\exists a_0 o_1 a_1^\exists a_1 \ldots o_{n-1} a_{n-1}^\exists a_{n-1}$.

Let q be the last state of this run. We then define $f_\exists(\pi) = a^\exists$, where $a^\exists \in \Sigma_\exists^\perp$ is the unique label that exists by conditions (ii) and (iv) such that there are $a \in \Sigma_0$ and $q \in Q_s$ such that $(q,(o_n,(a^\exists,a)),q') \in \rho$.

We now turn to the definition of winning conditions in LC-games under incomplete information. We consider *safety* and *reachability* winning conditions for $Player_\exists$ defined by visible sets of states in $\mathscr{G}$. A set $T \subseteq S$ is *visible* iff for every $s \in T$ and every $s' \in S$ with $obs(s') = obs(s)$ it holds that $s' \in T$.

A *safety LC-game under incomplete information* $\mathsf{Safety}(\mathscr{G},Err)$ is defined by a LC-game structure with incomplete information $\mathscr{G}$ and a visible set Err of error states that $Player_\exists$ must avoid. A strategy $f_\exists$ for $Player_\exists$ is *winning* in $\mathsf{Safety}(\mathscr{G},Err)$ iff no play in $\mathsf{Outcome}(f_\exists)$ visits a state in Err.

Note that according to this definition, $Player_\exists$ wins finite plays that do not reach an error state. If we want to ensure that plays reaching a state in $\mathscr{G}$ that corresponds to a deadlock in $\mathscr{L}$ are not winning for $Player_\exists$, we can easily achieve this by appropriately instrumenting $\mathscr{L}$ and Err.

A *reachability LC-game under incomplete information* $\mathsf{Reach}(\mathscr{G},Goal)$ is defined by a LC-game structure with incomplete information $\mathscr{G}$ and a visible set $Goal$ of goal states that $Player_\exists$ must reach. A strategy $f_\exists$ for $Player_\exists$ is *winning* in $\mathsf{Reach}(\mathscr{G},Goal)$ iff each play in $\mathsf{Outcome}(f_\exists)$ visits a state in $Goal$.

Remark. The definition of visible sets allows that $Err \cap S_1 \neq \emptyset$ and $Goal \cap S_1 \neq \emptyset$. Thus, our definition of visible objectives does not require that for each pair of plays π_1 and π_2 with $obs^+(\pi_1) = obs^+(\pi_2)$ (where obs^+ is defined for plays analogously to prefixes) it holds that $Player_\exists$ wins π_1 iff he wins π_2. For the algorithms, which we present in the next section, for solving safety and reachability LC-games under incomplete information, the objective for $Player_\exists$ does not have to satisfy this condition.

3 Algorithms for Solving Safety and Reachability Games

Better-Quasi Orderings. The subword ordering $\preceq$ on M^* is a WQO (and so is the ordering $\preceq$ on W defined earlier). That means, it is a reflexive and transitive relation such that for every infinite sequence $w_0, w_1, \ldots$ of elements of M^* there exist indices $0 \leq i < j$ such that $w_i \preceq w_j$.

The subword ordering (as well as other WQOs commonly used in verification) is in fact also a BQO, and so is the ordering on W. Hence they are preserved by the powerset operation. Here we omit the precise definition of BQOs since it is rather technical and it is not necessary for the presentation of our results. When needed, we recall its properties relevant for our arguments.

We extend $\preceq$ to a BQO $\preceq$ on the set S of states in $\mathscr{G}$ in the following way: for $s = (p,q_0,q_1,w) \in S$ and $s' = (p',q_0',q_1',w') \in S$, we have $s \preceq s'$ iff $p = p'$, $q_0 = q_0'$, $q_1 = q_1'$, $obs(s) = obs(s')$ and $w \preceq w'$.

A set $T \subseteq S$ is *upward-closed* (respectively *downward-closed*) iff for every $s \in T$ and every $s' \in S$ with $s \preceq s'$ (respectively $s' \preceq s$) it holds that $s' \in T$. The upward-closure of a set $T \subseteq S$ is $T\uparrow = \{s' \in S \mid \exists s.\ s \in T \text{ and } s \preceq s'\}$. For each upward (respectively downward) closed set $T \subseteq S$ and $o \in Obs$, the set $T' = \{s \in T \mid obs(s) = o\}$ is also upward (respectively downward) closed. We let $\mathscr{U}_{obs}(S) = \{u \subseteq S \mid u \neq \emptyset,\ u = u\uparrow \text{ and } \exists o \in Obs. \forall s \in u.\ obs(s) = o\}$ and for $u \in \mathscr{U}_{obs}(S)$ we define $obs(u)$ in the obvious way. The set $\mathscr{D}_{obs}(S)$ and $obs : \mathscr{D}_{obs}(S) \to Obs$ are defined analogously, requiring that the elements are downward-closed instead of upward-closed. $\mathscr{D}_{obs}^{\mathsf{fin}}(S)$ is the set of finite sets in $\mathscr{D}_{obs}(S)$.

The transition relation $\Rightarrow_g$ enjoys the following property: if $s \stackrel{a}{\Rightarrow}_g s'$ and $s \preceq s''$, then $s'' \stackrel{a}{\Rightarrow}_g s'$. Thus, the set of predecessors w.r.t. some $a \in \Sigma$ of any set of states is upward-closed. For LCSs the set of successors w.r.t. some $a \in \Sigma$ of any set of states is a downward-closed set.

Let $\mathsf{Pre} : \mathscr{P}(S) \times \Sigma \to \mathscr{P}(S)$ be the function defined as $\mathsf{Pre}(T,a) = \{s \in S \mid \exists s' \in T.\ s \stackrel{a}{\Rightarrow}_g s'\}$ and let $\mathsf{Post} : \mathscr{P}(S) \times \Sigma \to \mathscr{P}(S)$ be the function defined as $\mathsf{Post}(T,a) = \{s \in S \mid \exists s' \in T.\ s' \stackrel{a}{\Rightarrow}_g s\}$. As recalled above, for each $T \subseteq S$ and each $a \in \Sigma$, $\mathsf{Pre}(T,a)$ is upward-closed and $\mathsf{Post}(T,a)$ is downward-closed.

We define the functions $\mathsf{Pre}_0 : \mathscr{U}_{obs}(S) \times \Sigma_0 \to \mathscr{P}_{\mathsf{fin}}(\mathscr{U}_{obs}(S))$ and $\mathsf{Pre}_1 : \mathscr{U}_{obs}(S) \to \mathscr{P}_{\mathsf{fin}}(\mathscr{U}_{obs}(S))$ that map a set $u \in \mathscr{U}_{obs}(S)$ to a finite set of upward-closed sets that partition the respective set of predecessors of u according to the observations *Player*$_\exists$ makes. Formally, $\mathsf{Pre}_0(u,a) = \{u' \in \mathscr{U}_{obs}(S) \mid \exists o \in Obs.\ u' = \mathsf{Pre}(u,a) \cap States(o)\}$ and $\mathsf{Pre}_1(u) = \{u' \in \mathscr{U}_{obs}(S) \mid \exists o \in Obs.\ u' = (\bigcup_{a \in \Sigma_1} \mathsf{Pre}(u,a)) \cap States(o)\}$. Similarly, using the function Post above, we can define the successor functions $\mathsf{Post}_0 : \mathscr{D}_{obs}(S) \times \Sigma_0 \to \mathscr{P}_{\mathsf{fin}}(\mathscr{D}_{obs}(S))$ and $\mathsf{Post}_1 : \mathscr{D}_{obs}(S) \to \mathscr{P}_{\mathsf{fin}}(\mathscr{D}_{obs}(S))$. Since the transition relation of $\mathscr{G}$ has finite branching, if $d \in \mathscr{D}^{\mathsf{fin}}_{obs}(S)$ then $d' \in \mathscr{D}^{\mathsf{fin}}_{obs}(S)$ for $d' \in \mathsf{Post}_0(d,a)$ or $d' \in \mathsf{Post}_1(d)$.

When analyzing LCSs, upward-closed sets are typically represented by their *finite sets of minimal elements*, and downward-closed sets are represented by *simple regular expressions*. These representations can be extended to obtain finite representations of elements of $\mathscr{U}_{obs}(S)$ and $\mathscr{D}_{obs}(S)$. By the definition of $\preceq$ on S, each visible set of states is upward-closed, and hence, the sets Err and $Goal$ in safety and reachability games are finitely representable. In the rest, we assume that they are represented such a way.

Our termination arguments rely on the following property: For every BQO $\preceq$ on a set X, the superset relation $\supseteq$ is a BQO on the set of upward-closed sets in $\mathscr{P}(X)$ and the subset relation $\subseteq$ is a BQO on the set of downward-closed sets. This implies that $\supseteq$ is a BQO on $\mathscr{U}_{obs}(S)$ and that $\subseteq$ is a BQO on $\mathscr{D}_{obs}(S)$.

LC-games under incomplete information with safety objectives. We describe a decision procedure for safety LC-games under incomplete information which is based on a backward fixpoint computation.

Each step in the fixpoint computation corresponds to a step in the game, which is not necessarily observable by *Player*$_\exists$. Thus, this construction is correct w.r.t. *Player*$_\exists$ strategies that are $\widetilde{obs}$-consistent, where, intuitively, the function $\widetilde{obs}$ maps a prefix to a sequence that includes also the (trivial) observations of S_1 states, and $\widetilde{obs}$-consistency is defined analogously to obs^+-consistency. To avoid this problem, our algorithm performs the fixpoint computation on a LC-game structure with incomplete information $\widetilde{\mathscr{G}}$ obtained from $\mathscr{G}$ by adding an *idle transition* for process 1. This game structure has the following property: *Player*$_\exists$ has an obs^+-consistent winning strategy in the game $\mathsf{Safety}(\mathscr{G}, Err)$ iff *Player*$_\exists$ has an $\widetilde{obs}$-consistent winning strategy in $\mathsf{Safety}(\widetilde{\mathscr{G}}, Err)$, which yields correctness of the algorithm.

Formally, the function $\widetilde{obs} : \mathsf{Prefs}_\exists(\mathscr{G}) \to (Obs^* \cdot \Sigma_0)^* \cdot Obs$ is defined as: $\widetilde{obs}(s_0 a_0^\exists a_0 \dots s_n) = obs(s_0) \cdot obs'(a_0) \cdot \ldots \cdot obs(s_n)$. The game structure $\widetilde{\mathscr{G}}$ is the tuple $\widetilde{\mathscr{G}} = (S, I, \widetilde{\to_g}, C, M, \Sigma_0, \widetilde{\Sigma_1}, \Sigma_\exists, C_{obs})$ where $\widetilde{\Sigma_1} = \Sigma_1 \cup \{idle\}$ and $idle \notin \Sigma$, and $\widetilde{\to_g} = \to_g \cup \{((1, q_0, q_1, w), idle, (p', q_0, q_1, w)) \mid p' \in \{0,1\}\}$.

We define the set $\mathscr{L}(S)$ for S as $\mathscr{L}(S) = \{l \in \mathscr{P}_{\mathsf{fin}}(\mathscr{U}_{obs}(S)) \mid l \neq \emptyset \text{ and } \exists o \in Obs. \forall u \in l.\ obs(u) = o\}$ and define $obs(l)$ for each $l \in \mathscr{L}(S)$ in the obvious way. We provide a fixpoint-based algorithm that computes a set $B \subseteq \mathscr{L}(S)$ such that each $l \in B$ has the following property: if $K \subseteq S$ is the set of states that the game can be currently in according to *Player*$_\exists$'s knowledge and $K \cap u \neq \emptyset$ for every $u \in l$, then *Player*$_\exists$ cannot win when his knowledge is K. Considering the set I of initial states, if for some $l \in B$ it holds that $I \cap u \neq \emptyset$ for all $u \in l$, then *Player*$_\exists$ has no obs^+-consistent winning strategy in $\mathsf{Safety}(\mathscr{G}, Err)$.

Our procedure computes a sequence $B_0 \subseteq B_1 \subseteq B_2 \dots$ of finite subsets of $\mathscr{L}(S)$. The computation starts with the set $B_0 = \{\{Err \cap States(o)\} \mid o \in Obs\}$. For $i \geq 0$, we let $B_{i+1} = B_i \cup N_{i+1}$, where the set N_{i+1} of new elements is computed based on B_i and is the smallest set that contains each $l \in \mathscr{L}(S)$ which is such that $l \subseteq \bigcup_{l' \in B_i, u' \in l'} ((\bigcup_{a \in \Sigma_0} \mathsf{Pre}_0(u', a)) \cup \mathsf{Pre}_1(u'))$ and:

- if $l \in \mathscr{P}(\mathscr{P}(S_0))$ then for every possible choice $a^\exists \in Act_\exists(obs(l))$ of *Player*$_\exists$, there exist an action $a \in Act_\forall(obs(l), a^\exists)$ and $l' \in B_i$ such that for every $u' \in l'$ it holds that $\mathsf{Pre}_0(u', a) \cap l \neq \emptyset$,
- if $l \in \mathscr{P}(\mathscr{P}(S_1))$ then there exists $l' \in B_i$ such that for every $u' \in l'$ it holds that $\mathsf{Pre}_1(u') \cap l \neq \emptyset$.

The ordering $\sqsubseteq$ on $\mathscr{L}(S)$ is defined such that for $l, l' \in \mathscr{L}(S)$, we have $l \sqsubseteq l'$ iff for every $u \in l$ there exists a $u' \in l'$ such that $u \supseteq u'$. The ordering $\sqsubseteq$ is a BQO, since $\supseteq$ is a BQO on $\mathscr{U}_{obs}(S)$. Intuitively, if l belongs to the set of elements of $\mathscr{L}(S)$ in which *Player*$_\exists$ cannot win, so does every l' with $l \sqsubseteq l'$.

We say that the sequence $B_0, B_1, B_2 \ldots$ *converges at k* if $\mathsf{Min}(B_{k+1}) \subseteq \mathsf{Min}(B_k)$, where $\mathsf{Min}(B_i)$ is the set of minimal elements of B_i w.r.t. $\sqsubseteq$. This condition can be effectively checked, since each B_i is finite. We argue that there exists a $k \geq 0$ such that the sequence computed by the procedure described above converges at k (and hence the procedure will terminate).

Let $F_0, F_1, F_2, \ldots$ be the sequence of upward-closed elements of $\mathscr{P}(\mathscr{L}(S))$ where $F_i = B_i \uparrow$ for each $i \geq 0$. As $F_0, F_1, F_2 \ldots$ is a monotonically increasing sequence of upward-closed sets of elements of $\mathscr{L}(S)$, it must eventually stabilize, i.e., there is a $k \geq 0$ such that $F_{k+1} \subseteq F_k$. Thus, since $F_{i+1} \subseteq F_i$ if and only if $\mathsf{Min}(B_{i+1}) \subseteq \mathsf{Min}(B_i)$, the sequence $B_0, B_1, B_2 \ldots$ is guaranteed to converge at some $k \geq 0$.

Proposition 1. *Let $B = B_k$, where the sequence $B_0, B_1, B_2 \ldots$ converges at k. Then, Player$_\exists$ has an $\widetilde{obs}$-consistent winning strategy in $\mathsf{Safety}(\widetilde{\mathscr{G}}, Err)$ iff for every $l \in B$ there exists $u \in l$ with $u \cap I = \emptyset$.*

If Player$_\exists$ has an $\widetilde{obs}$-consistent winning strategy in $\mathsf{Safety}(\widetilde{\mathscr{G}}, Err)$, then Player$_\exists$ has a finite-state obs^+-consistent winning strategy in the original game $\mathsf{Safety}(\mathscr{G}, Err)$.

Proof Idea. A counterexample tree for $\mathsf{Safety}(\widetilde{\mathscr{G}}, Err)$ represents a witness for the fact that *Player*$_\exists$ does not have an $\widetilde{obs}$-consistent winning strategy in $\mathsf{Safety}(\widetilde{\mathscr{G}}, Err)$. It is a finite tree with nodes labeled with elements of $\mathscr{D}_{obs}(S)$. If there is a $l \in B$ such that $u \cap I \neq \emptyset$ for every $u \in l$, a counterexample tree can be constructed in a top-down manner. For the other direction we can show by induction on the depth of the existing counterexample trees that there exists a $l \in B$ such that $u \cap I \neq \emptyset$ for every $u \in l$.

For the case when *Player*$_\exists$ wins the game $\mathsf{Safety}(\widetilde{\mathscr{G}}, Err)$ we can construct a finite-state obs^+-consistent winning strategy for *Player*$_\exists$ in the game $\mathsf{Safety}(\mathscr{G}, Err)$ by using as states for the strategy automaton functions from observations to a finite set $\mathscr{V} \subseteq \mathscr{P}_{\mathsf{fin}}(\mathscr{P}(S))$ each of whose elements V preserves the invariant that for every $l \in B$ there exists a $u \in l$ such that $u \cap \bigcup_{v \in V} v = \emptyset$. □

LC-Games under incomplete information with reachability objectives. For reachability games we give a procedure based on forward exploration of the sets of states representing the knowledge of *Player*$_\exists$ about the current state of the game. Since *Player*$_\exists$ can only observe the heads the observable channels, his knowledge at each point of the play is a finite downward-closed set, element of $\mathscr{D}^{\mathsf{fin}}_{obs}(S)$. To update this knowledge we define functions $\mathsf{Post}^{obs}_0 : \mathscr{D}^{\mathsf{fin}}_{obs}(S) \times \Sigma_0 \to \mathscr{P}_{\mathsf{fin}}(\mathscr{D}^{\mathsf{fin}}_{obs}(S))$ and $\mathsf{Post}^{obs}_1 : \mathscr{D}^{\mathsf{fin}}_{obs}(S) \to \mathscr{P}_{\mathsf{fin}}(\mathscr{D}^{\mathsf{fin}}_{obs}(S))$ that map a set $d \in \mathscr{D}^{\mathsf{fin}}_{obs}(S)$ to a finite set of elements of $\mathscr{D}^{\mathsf{fin}}_{obs}(S)$, each of which is a set of states that *Player*$_\exists$ knows, according to his current observation, the game may be in after (a transition from Σ_0 and) a sequence of transitions from Σ_1. For each $d \in \mathscr{D}^{\mathsf{fin}}_{obs}(S)$ we have $d' \in \mathsf{Post}^{obs}_0(d,a)$ (respectively $d' \in \mathsf{Post}^{obs}_1(d)$) iff there exists a sequence $d_0, d_1, \ldots, d_n \in \mathscr{D}^{\mathsf{fin}}_{obs}(S)$ such that $d_0 \in \mathsf{Post}_0(d,a)$ (respectively $d_0 = d$), for every $1 \leq i \leq n$ it holds that $d_{i-1} \subseteq S_1$ and $d_i \in \mathsf{Post}_1(d_{i-1})$, and for every $0 \leq i < j < n$ it holds that $d_i \not\subseteq d_j$ and one of the following conditions is satisfied: (1) $d' \subseteq \mathit{Goal}$, $d' = d_0$ and $n = 0$ (i.e., $d' \subseteq \mathit{Goal} \cap S_1$), or (2) there exists a $1 \leq i < n$ such that $d_i \subseteq d_n$ and $d' = d_n$ (i.e., $d' \subseteq S_1$), or (3) $d' = \{(0, q'_0, q'_1, w') \mid (1, q'_0, q'_1, w') \in \bigcup_{i=0}^{n} d_i\}$ (i.e., $d' \subseteq S_0$).

We construct a finite set of trees rooted at the different possible knowledge sets for *Player*$_\exists$ at location q^0_o. The nodes of the trees are labeled with knowledge sets, i.e., with elements of $\mathscr{D}^{\mathsf{fin}}_{obs}(S)$. The edges are labeled wit pairs of transition labels, i.e., elements of $\Sigma^{\perp}_{\exists} \times \Sigma_0$, where the first element of a pair is a possible choice of *Player*$_\exists$ and the second element is a corresponding choice of *Player*$_\forall$.

Formally, the forward exploration procedure constructs a forest $\mathscr{T}$ in which the roots are labeled with the sets $\{(0, q^0_0, q^0_1, \varepsilon)\}$ and all the sets $d \in \mathsf{Post}^{obs}_1(\{(1, q^0_0, q^0_1, \varepsilon)\} \setminus \mathit{Goal})$. At each step of the construction an open leaf node n with label d is processed in the following way:

- If $d \subseteq \mathit{Goal}$, we close the node and do not expand further from this node.
- If $d \not\subseteq \mathit{Goal}$ and either $d \subseteq S_0$ and there exists an ancestor of n that is labeled with d' and such that $d' \subseteq d$, or $d \subseteq S_1$, we close the node and do not expand further from this node.

- Otherwise, we add the set of successors of n: for each $a^{\exists} \in Act_{\exists}(obs(d))$, each $a \in Act_{\forall}(obs(d), a^{\exists})$ and each $d' \in \mathsf{Post}_0^{obs}(d,a)$ we add exactly one successor n' labeled with d' and label the edge (n,n') with $(a^{\exists},a)$. The set of successors for $(a^{\exists},a)$ is denoted with $Children(n,a^{\exists},a)$.

The finite branching of the transition relation of $\mathscr{G}$ and the fact that $\subseteq$ is a BQO on $\mathscr{D}_{obs}^{\mathsf{fin}}(S)$ imply that each of the sets $\mathsf{Post}_0^{obs}(d,a)$ and $\mathsf{Post}_1^{obs}(d)$ can be effectively computed, the set of roots and the out-degree of each node are finite, and the above procedure terminates constructing a finite forest $\mathscr{T}$.

We label each node n in $\mathscr{T}$ with a boolean value $win(n)$. For a leaf node n with $d(n) \subseteq Goal$, we define $win(n) = true$ and for any other other leaf node n we define $win(n) = false$. The value of a non-leaf node is computed based on those of its children by interpreting the choices of $Player_{\exists}$ disjunctively and the choices of $Player_{\forall}$ conjunctively. Formally, for every non-leaf node n we define $win(n) = \bigvee_{a^{\exists} \in Act_{\exists}(obs(d(n)))} \bigwedge_{a \in Act_{\forall}(obs(d(n)),a^{\exists})} \bigwedge_{n' \in Children(n,a^{\exists},a)} win(n')$, where $d(n)$ is the set of states labeling n.

Proposition 2. *$Player_{\exists}$ has an obs^{+}-consistent winning strategy in $\mathsf{Reach}(\mathscr{G}, Goal)$ iff for every root n in $\mathscr{T}$ it holds that $win(n) = true$. If $Player_{\exists}$ has an obs^{+}-consistent winning strategy in $\mathsf{Reach}(\mathscr{G}, Goal)$, then he also has a finite state obs^{+}-consistent winning strategy in $\mathsf{Reach}(\mathscr{G}, Goal)$.*

Proof Idea. If all the roots are labeled with *true* we can construct a finite-state obs^{+}-consistent strategy winning for $Player_{\exists}$ in $\mathsf{Reach}(\mathscr{G}, Goal)$, by mapping each prefix in $\mathsf{Prefs}_{\exists}(\mathscr{G})$ to a label in $\Sigma_{\exists}^{\perp}$, determined by a corresponding path in $\mathscr{T}$ and a fixed successful choice at its last node, if such path and choice exist, or given an appropriate default value otherwise. For the other direction we suppose that some root is labeled with *false* and show that for any obs^{+}-consistent strategy $f_{\exists}$ for $Player_{\exists}$, we can use the tree to construct a play $\pi \in \mathsf{Outcome}(f_{\exists})$ that never visits a state in $Goal$. □

LC-games under incomplete information with parity objectives. We now turn to more general ω-regular visible objectives for $Player_{\exists}$ where the undecidability results established in [1] for perfect information lossy channel games in which only one player can lose messages, carry on to our setting.

A *visible priority function* $pr : Obs \to \{0,1,\ldots,n\}$ for natural number $n \in \mathbb{N}$ maps each observation to a non-negative integer priority. For an infinite play $\pi = s_0 a_0^{\exists} a_0 s_1 a_1^{\exists} a_1 \ldots$ we define $pr(\pi) = \min\{pr(o) \mid o \in InfObs(\pi)\}$, where $InfObs(\pi)$ is the set of observations that occur infinitely often in π, and define $wpr(\pi) = \min\{pr(obs(s_0)), pr(obs(s_1)), \ldots\}$. A *parity* (respectively *weak parity*) LC-game under incomplete information $Parity(\mathscr{G}, pr)$ (respectively $WeakParity(\mathscr{G}, pr)$) is defined by a LC-game structure with incomplete information $\mathscr{G}$ and a visible priority function pr. A strategy $f_{\exists}$ for $Player_{\exists}$ is *winning* in the parity game $Parity(\mathscr{G}, pr)$ (weak parity game $WeakParity(\mathscr{G}, pr)$) iff for every infinite play $\pi \in \mathsf{Outcome}(f_{\exists})$ it holds that $pr(\pi)$ is even (respectively $wpr(\pi)$ is even).

Proposition 3. *The weak parity game solving problem for LC-games under incomplete information, that is, given a weak parity LC-game under incomplete information $WeakParity(\mathscr{G}, pr)$ to determine whether there exists an obs^{+}-consistent winning strategy for $Player_{\exists}$ in $WeakParity(\mathscr{G}, pr)$, is undecidable.*

Proof Idea. In [1] it was shown that in the perfect information setting the weak parity problem for B-LCS games, which are games played on a finite set of channels in which player A has a weak parity objective and only player B is allowed to lose messages, is undecidable. Their proof (given for A-LCS games but easily transferable into a proof for B-LCS games) is based on a reduction from the infinite computation problem for transition systems based on lossy channel systems, which is undecidable [3].

We argue that this reduction can be adapted for our framework, with $Player_{\exists}$ in the role of player A and $Player_{\forall}$ in the role of player B. The fact that here $Player_{\exists}$ choses only transition labels and plays under incomplete information does not affect the proof for B-LCS games, since there player A just

follows passively, while player B simulates the original system. The values of the priority function used in [1] do not depend on the contents of the channels. Thus, we can define a visible priority function. □

As a consequence, the parity game solving problem for LC-games under incomplete information is undecidable as well. As noted in [1], the construction from the proposition above can be used to show undecidability of A-LCS and B-LCS games with Büchi and co-Büchi objectives.

Summary of the results. The results of the paper are summarized in the following theorem.

Theorem 1. *For lossy channel game structures with incomplete information*

- *games with visible safety or reachability objectives for Player$_\exists$ are decidable, and when Player$_\exists$ has an observation-based winning strategy, a finite-state such strategy can be effectively computed,*
- *games with visible weak parity objectives for Player$_\exists$ are undecidable.*

4 Conclusion

We showed that the game solving problem for LC-games under incomplete information with safety or reachability objective for *Player*$_\exists$ is decidable. LC-games under incomplete information with more general winning conditions, such as weak parity (as well as Büchi and co-Büchi) condition can easily be shown to be undecidable, using a reduction similar to the one described in [1] for A-LCS games (which are perfect information games defined on LCSs in which only one player can lose channel messages). An orthogonal extension that is also clearly undecidable is decentralized control. This implies that suitable abstraction techniques are needed to address the synthesis problem within these undecidable settings.

Acknowledgements This work is partially supported by the DFG as part of SFB/TR 14 AVACS.

References

[1] Parosh Aziz Abdulla, Ahmed Bouajjani & Julien d'Orso (2008): *Monotonic and Downward Closed Games*. *J. Log. Comput.* 18(1), pp. 153–169, doi:10.1093/logcom/exm062.

[2] Parosh Aziz Abdulla, Aurore Collomb-Annichini, Ahmed Bouajjani & Bengt Jonsson (2004): *Using Forward Reachability Analysis for Verification of Lossy Channel Systems*. *FMSD* 25(1), pp. 39–65, doi:10.1023/B:FORM.0000033962.51898.1a.

[3] Parosh Aziz Abdulla & Bengt Jonsson (1996): *Undecidable Verification Problems for Programs with Unreliable Channels*. *Inf. Comput.* 130(1), pp. 71–90, doi:10.1006/inco.1996.0083.

[4] K. Chatterjee, L. Doyen, T. A. Henzinger & J.-F. Raskin (2006): *Algorithms for Omega-Regular Games with Imperfect Information*. In: *Proc. CSL'06, LNCS* 4207, Springer, pp. 287–302, doi:10.1007/11874683_19.

[5] M. De Wulf, L. Doyen & J.-F. Raskin (2006): *A Lattice Theory for Solving Games of Imperfect Information*. In: *Proc. HSCC'06, LNCS* 3927, Springer, pp. 153–168, doi:10.1007/11730637_14.

[6] Alberto Marcone (2001): *Fine Analysis of the Quasi-Orderings on the Power Set*. *Order* 18(4), pp. 339–347, doi:10.1023/A:1013952225669.

[7] E. C. Milner (1985): *Basic Wqo- and Bqo-Theory*. In: *Graphs and order*, pp. 487–502, doi:10.1007/978-94-009-5315-4_14.

[8] C. Nash-Williams (1965): *On well-quasi ordering infinite trees*. *Proceedings of the Cambridge Philosophical Society* 61, pp. 697–720, doi:10.1017/S0305004100039062.

[9] John H. Reif (1984): *The Complexity of Two-Player Games of Incomplete Information*. *J. Comput. Syst. Sci.* 29(2), pp. 274–301, doi:10.1016/0022-0000(84)90034-5.

Strategic Analysis of Trust Models for User-Centric Networks

Marta Kwiatkowska
Department of Computer Science
University of Oxford

David Parker
School of Computer Science
University of Birmingham

Aistis Simaitis
Department of Computer Science
University of Oxford

We present a strategic analysis of a trust model that has recently been proposed for promoting cooperative behaviour in user-centric networks. The mechanism for cooperation is based on a combination of reputation and virtual currency schemes in which service providers reward paying customers and punish non-paying ones by adjusting their reputation, and hence the price they pay for services. We model and analyse this system using PRISM-games, a tool that performs automated verification and strategy synthesis for stochastic multi-player games using the probabilistic alternating-time temporal logic with rewards (rPATL). We construct optimal strategies for both service users and providers, which expose potential risks of the cooperation mechanism and which we use to devise improvements that counteract these risks.

1 Introduction

User-centric networks are designed to encourage users to act cooperatively, sharing resources or services between themselves, for example in order to provide connectivity in a mobile ad-hoc network. The effectiveness of such networks is heavily dependent on their cooperation mechanisms, which are often based on the use of incentives to behave unselfishly. In this paper, we present an analysis of a cooperation mechanism for user-centric networks [3], which combines a *reputation*-based incentive, used to establish a measure of *trust* between users, and a *virtual currency* mechanism used to "buy" and "sell" services.

The cooperation model proposed in [3] was analysed formally by the authors using probabilistic model checking [1, 2]. They verified several performance properties, specified in the probabilistic temporal logics PCTL and CSL, on discrete- and continuous-time Markov chains models and, in [1], also used Markov decision processes to assess the worst-case performance of service providers.

In this paper, we take a different approach and study the cooperation mechanism using *strategy-based analysis*. The system is modelled as a *stochastic multi-player game*, in which service providers and customers are modelled as players with objectives, expressed in the logic probabilistic alternating-time temporal logic with rewards (rPATL) [4]. We model and analyse the cooperation mechanism using PRISM-games [5], a probabilistic model checker for stochastic multi-player games. We use rPATL model checking to identify weaknesses in the cooperation mechanism and then perform *strategy synthesis* to discover important insights into the model: firstly, we construct and visualise potential attacks or undesirable behaviour; secondly, we develop improvements to the system that alleviate these problems and check their correctness.

Related work. Game-theoretic techniques have been applied to a wide variety of problems in the context of computer networks, from network security [8] to self-organisation in ad-hoc networks [6]. Of particular relevance to this paper is the work in [7], which gives a game-theoretic analysis of cooperative incentive schemes in mobile ad-hoc networks and proposes the combination of trust and currency mechanisms used in [3]. Its effectiveness is analysed using a combination of theoretical and simulation results. By contrast, we adopt a semi-automatic approach where the strategies are synthesised automatically by the tool from rPATL specifications, and are then analysed to understand and improve the cooperation

F. Mogavero, A. Murano, and M.Y. Vardi (Eds.):
1st Workshop on Strategic Reasoning 2013 (SR'13)
EPTCS 112, 2013, pp. 53–59, doi:10.4204/EPTCS.112.10

mechanism. The logic rPATL has been previously used to analyse cooperation incentives in micro-grid energy management and decentralised agreement in sensor networks [4], but a detailed strategy-based analysis was not performed.

2 Modelling the Cooperation Mechanism

2.1 The cooperation mechanism

The basic ideas behind the cooperation mechanism of [3] can be summarised as follows. We assume a general model of *providers* offering *services* to *requesters*. Cooperation between *users* of the network (requesters and providers) is managed through a combination of reputation and virtual currency.

Reputation is captured by a discrete *trust measure*, denoted $trust_{ij}$, representing the extent to which user i trusts user j, based on previous interactions between them and the recommendations provided by the other users in the network. This is used to determine whether a service request from j is accepted by i. A *trust level* T_{ij} is computed as a weighted sum $T_{ij} = \alpha \cdot trust_{ij} + (1-\alpha) \cdot recs_{ij}$, where $recs_{ij}$ is an "indirect" trust measure, taken as the the average value of $trust_{kj}$ for other users k (whereas $trust_{ij}$ is called a "direct" measure of trust). By default, i will decide to accept j's request if T_{ij} is not below a pre-specified *service trust level*, denoted st_i. The parameter $\alpha \in [0,1]$ controls the relative influence that the direct and indirect measures of trust have on this decision.

The reputation scheme is then integrated with a virtual currency system, where services are bought and sold between users, and the cost paid to i by j for a service is a function of $trust_{ij}$. Assuming model parameters for minimum and maximum costs C_{min}, C_{max} and threshold T', the cost is defined as

$$C(trust_{ij}) = \begin{cases} C_{min} + \frac{C_{max}-C_{min}}{T'} \cdot (T' - trust_{ij}) & \text{if } trust_{ij} < T' \\ C_{min} & \text{if } trust_{ij} \geq T' \end{cases}$$

Procurement of a service proceeds in several phases. First, a requester j chooses a provider i and makes a request. If $T_{ij} \geq st_i$, the request is accepted. In this case, the two users then "negotiate" the service cost, using the function of $trust_{ij}$ given above. The negotiation may, however, fail: with probability c_i, user i cancels the accepted request; this represents the "cooperative attitude" [1] of the provider i. If not cancelled, the service is delivered and the requester chooses whether or not to pay the negotiated price to the provider. If payment is made, the provider increases the trust measure of the requester by one unit. If not, the measure is decreased by td_i units. On encountering a requester for the first time, a provider shares the trust measure with the other providers.

2.2 A stochastic game model

We build a model of the cooperation mechanism of [3] as a (turn-based) stochastic multi-player game (SMG). An SMG comprises a finite set of *players* and a finite set of *states*. In each state, exactly one player chooses (possibly randomly) from a set of available *actions*. When an action is taken, the result is a *probabilistic transition*, i.e. a successor state is chosen according to a discrete probability distribution. The choices for each player are made by a *strategy*, which selects an action (or distribution over actions) based on the history of the SMG so far. The strategies needed in this paper are *memoryless* (i.e. history independent) and *deterministic* (i.e. do not use randomisation).

We developed the SMG model using the PRISM-games model checker, taking the PRISM model of [1] as a starting point.[1]. The SMG model has one player for each user in the network. The choices

[1] All model/property files are available at: `http://www.prismmodelchecker.org/files/sr13trust/`

made by a "requester" player model the decision of which provider is selected at each point in the system execution. In the basic model, the "provider" players do not have any choices to make; later (in Sections 3.2 and 3.3), we will add choices for these players in order to synthesise strategies that can be used to improve the cooperation mechanism. The stochastic aspects of the SMG model are primarily required to model the fact that negotiations fail probabilistically.

We adopt the same basic network configuration as used in the original analysis of the protocol [1], which comprises 3 providers and 1 requester. Even though this network is relatively small, it still captures the fundamental aspects of the protocol. For instance, observe that the decision whether to provide a service to a requester does not depend on the trust level of other requesters in the network, so incorporating more requesters does not offer any more information about the dynamics of trust and provided services. On the other hand, as we will show, using three service providers already allows us to identify malicious strategies for the requester that can be generalised to an arbitrary number of providers (see, e.g., the discussion about the number of unpaid requests in Section 3.1).

The parameters of the cooperation mechanism are also taken from [1] and are as follows. The trust measure is an integer in the range 0 to 10 and is initially 5. We use α=0.8 to compute trust levels, unless stated otherwise, and the service trust threshold st_i is set to 5 for all providers. We use a negotiation failure probability of c_i=0.05 for all providers i, and the parameters used to compute prices are fixed at C_{min}=2, C_{max}=10 and T'=8.

3 A Strategy-based Analysis

We now analyse the cooperation model described above, showing how the interplay between the two key components of the protocol, trust and virtual currency, affects the cooperation dynamics. Our analysis is based on strategy synthesis for properties in the temporal logic rPATL [4]. The logic combines features of the multi-agent logic ATL, the probabilistic logic PCTL, and operators to reason about expected reward or cost measures. A simple example of an rPATL formula is $\langle\langle\{1,2\}\rangle\rangle \mathsf{P}_{\geq 0.75}[\mathsf{F}^{\leq 5} goal]$, which asks "do players 1 and 2 have a (combined) strategy to ensure that the probability of reaching a '*goal*' state within 5 steps is at least 0.75, regardless of the strategies of other players in the game?". Alternatively, we can use a numerical query such as $\langle\langle\{1,2\}\rangle\rangle \mathsf{P}_{\max=?}[\mathsf{F}^{\leq 5} goal]$: "what is the maximum probability of reaching a '*goal*' state within 5 steps that can be ensured by players 1 and 2?". An example of property to reason about rewards (or costs) is $\langle\langle\{3\}\rangle\rangle \mathsf{R}^{r}_{\leq 10}[\mathsf{F}^{\star} goal]$, which asks "does player 3 have a strategy to ensure that the expected amount of reward r cumulated before reaching a '*goal*' state is at most 10?". The $\star$ parameter lets us specify what the total reward should be if a '*goal*' state is *not* reached: we can assign zero reward ($\star$=0), infinite reward ($\star$=∞) or allow reward to accumulate indefinitely ($\star$=c). For precise details of the logic rPATL and its semantics, we refer the reader to [4].

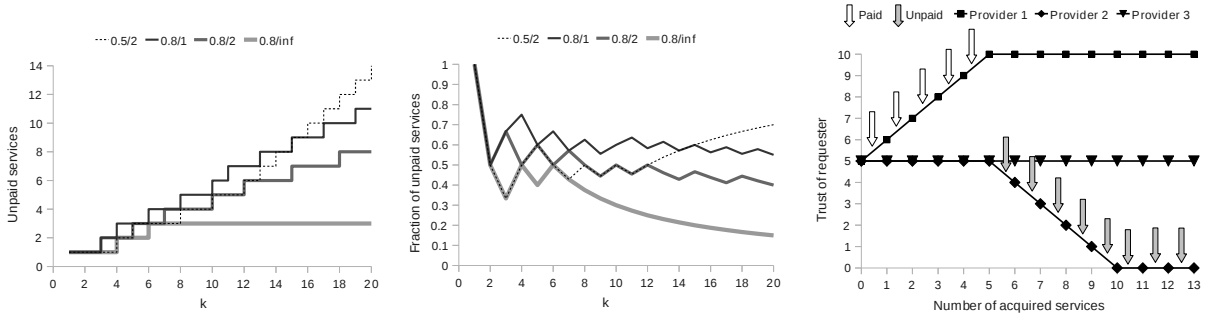
3.1 Unpaid requests

First, we consider the extent to which the requester can obtain services without paying for them. We analyse the maximum (expected) number of unpaid services that the requester can obtain if its goal is to get k services in total. This is expressed in rPATL as:

$$\langle\langle\{requester\}\rangle\rangle \mathsf{R}^{unpaid}_{\max=?}[\mathsf{F}^{c} services = k],$$

where *unpaid* denotes a *reward structure* assigning 1 to every unpaid request. The results for various combinations of model parameters α and td_i are shown in Figure 1 (we use 0.5/2 to indicate that $\alpha = 0.5$

and trust is decreased by $td_i = 2$ units upon an unpaid service; $td_i = \text{inf}$ means that trust is reset to 0 upon an unpaid service).



(a) *Number* of unpaid services. (b) *Fraction* of unpaid services. (c) Strategy for 0.5/2 and $k = 13$.

Figure 1: Maximum unpaid services the requester can achieve in obtaining k services.

Figures 1a and 1b show the number and fraction, respectively, of services that are unpaid, for a range of k. From Figure 1b, in particular, we see that, for parameters 0.5/2 and 0.8/*inf*, the behaviour is fundamentally different from the other two - the portion of requests converges to 1 and 0, respectively. For 0.8/*inf*, this behaviour is expected, because the trust measure is decreased to 0 upon non-payment; however, the behaviour of 0.5/2 represents an attack on the trust model allowing the requester to receive an unlimited number of unpaid services for a fixed cost. We synthesise an attacker (requester) strategy for our model with 3 providers, for the case of acquiring $k = 13$ services: for a cost of 5 services, the requester can get an unlimited number of unpaid services. We depict the strategy in Figure 1c. Arrows represent "request-and-pay" (white arrow) and "request-and-do-not-pay" (grey arrow) actions of the optimal requester strategy, depending on the number of services acquired so far.

This attack is possible if $st_i \leq (1-\alpha) \cdot T_{\max}$ for some provider i, where $T_{\max}$ is the maximum trust level among all providers. We note that it is only viable if the network is sufficiently small since the fixed cost increases with the number of providers sharing the trust information: to achieve the required indirect trust measure $recs_{ij} \geq \frac{st_i}{1-\alpha}$, the requester must pay for a number of services proportional to the number of providers. However, in order to work, this requires that all providers share their initial direct trust measure even though they have not encountered the requester.

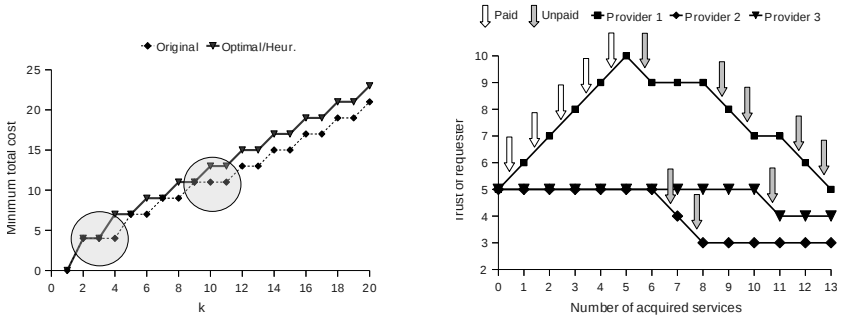
3.2 Cost of obtaining services

We now turn our attention to the virtual currency system, and study the minimum price at which the requester can buy k services. For this, we use rPATL formula:

$$\langle\langle \{requester\} \rangle\rangle \mathsf{R}^{cost}_{\min=?}[\mathsf{F}^{\infty} services = k].$$

Intuitively, the requester has a strategy to get one unpaid service for each paid service by executing the following sequence: pay, not pay, pay, not pay, etc. However, a plot of the above property (see highlighted sections of line 'Original' in Figure 2a), shows deviations from this pattern, where the requester can get 4 services for the price of 2 and, similarly, 11 services for the price of 9.

We synthesise a strategy achieving this and depict it in Figure 2b. We can see that all paid requests are directed to one provider and the others only receive unpaid requests. In fact, by exploiting the reputation system, the requester is even able to obtain 2 unpaid requests from provider 2.



(a) Minimum cost to obtain k services. (b) Example strategy for $k = 13$.

Figure 2: Cost of k services for requester and a strategy example.

Next, we devise a fix by changing the model to allow providers to manage the way they share trust information between themselves: they can choose whether to share trust information after interaction with the requester. We synthesise the optimal trust information sharing strategy for cooperating providers, whose behaviour is shown as 'Optimal/Heur.' in Figure 2a and can be seen to avoid the above shortfall. Manual examination of the synthesised strategy reveals a suitable heuristic whereby providers share trust information only when its direct trust of the requester is smaller than that of the others. We implement this heuristic in the model and find that it yields the same model checking results as the optimal strategy.

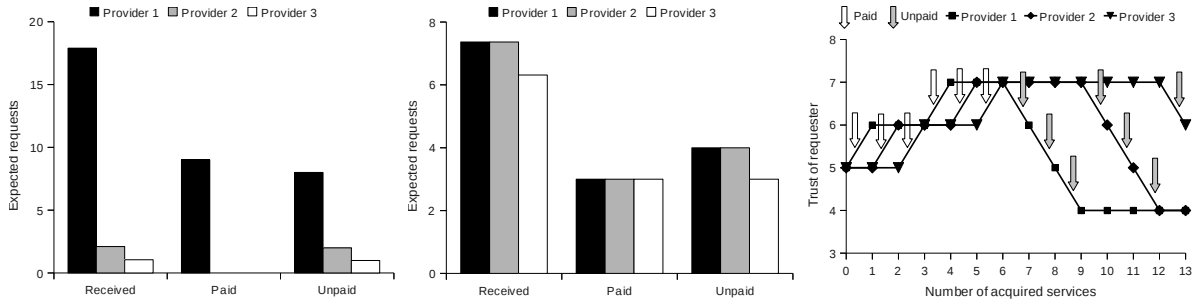
3.3 Provider selection incentives

Another interesting feature revealed by the analysis of the strategy in the previous section is that the proposed virtual currency system provides an incentive for the requester to only ever pay for services from one provider (see Figure 3a). This is in fact optimal behaviour because, in the computation of the service cost, only the direct trust measure is used. This may or may not be a desired feature for the mechanism. We can show that a simple change that incorporates the maximum difference between trust into the pricing model (i.e., cost is now computed as $original\_cost + \max_k |trust_{ij} - trust_{kj}|$, where $original\_cost$ is the cost assigned by the pricing scheme from Section 2.1) incentivises the requester to disperse its requests between service providers.

Figure 3b shows the distribution of requests between providers and Figure 3c depicts the actions of the optimal strategy in the new pricing scheme. This strategy contrasts with the strategy for the original mechanism from Figure 2b because paid requests are now distributed uniformly across all the service providers. This analysis of strategies has been performed using the "strategy implementation" feature of PRISM-games, which allows the user to synthesise an optimal player strategy for some rPATL formula, and then evaluate a second rPATL property on the modified SMG in which one coalition's strategy is fixed using the previously synthesised one. In this instance, we used the following rPATL formulae:

$$\langle\langle \{requester\} \rangle\rangle \mathsf{R}^{cost}_{\min=?}[\mathsf{F}^{\infty} services = k] \quad \text{and} \quad \langle\langle \emptyset \rangle\rangle \mathsf{R}^{r}_{\min=?}[\mathsf{F}^{c} services = k],$$

where the first formula was used to synthesise the strategy and the second formula is the one used to analyse it (r represents reward structures for Received, Paid, and Unpaid).



(a) Original pricing scheme. (b) Modified pricing scheme. (c) Optimal strategy for 13 services.

Figure 3: Distribution of requests among providers.

4 Discussion and future work

We have presented a strategy-based analysis of a cooperation mechanism for user-centric networks, using automated verification of stochastic multiplayer games. We have identified several undesirable properties of the model, including attacks on the reputation system and inefficiencies of the virtual currency mechanism. These would have been difficult to discover using conventional model checking. Furthermore, we have shown that an analysis of optimal strategies for the model can help us understand the incentives that the model introduces to the system and to devise and verify improvements.

Our approach, which is based on probabilistic model checking, builds and analyses a more detailed system model than other game-theoretic analysis techniques, such as [7]. On the one hand, this may impose limitations on the scalability of our approach. On the other hand, we are able to look at the protocol in fine detail and, as we have shown in this paper, identify subtle problems that arise even with a small number of system components, but which may also generalise to larger models.

There are many interesting directions for future work. We plan to further develop our probabilistic model checker PRISM-games to provide a wider range of analysis techniques. For example, we plan to incorporate additional reward operators dealing with limit averages and discounted sums. We would also like to investigate extensions of our techniques to incorporate partial-information strategies or more complex solution concepts such as Nash and subgame-perfect equilibria.

Acknowledgments. The authors are part supported by ERC Advanced Grant VERIWARE, the Institute for the Future of Computing at the Oxford Martin School and EPSRC grant EP/F001096/1.

References

[1] A. Aldini & A. Bogliolo (2012): *Model Checking of Trust-Based User-Centric Cooperative Networks*. In: *Proc. 4th International Conference on Advances in Future Internet (AFIN'12)*, pp. 32–41.

[2] A. Aldini & A. Bogliolo (2012): *Trading Performance and Cooperation Incentives in User-Centric Networks*. In: *Proc. International Workshop on Quantitative Aspects in Security Assurance (QASA'12)*.

[3] A. Bogliolo, P. Polidori, A. Aldini, W. Moreira, P. Mendes, M. Yildiz, C. Ballester & J. Seigneur (2012): *Virtual Currency and Reputation-based Cooperation Incentives in User-Centric Networks*. In: *Proc. 8th International Wireless Communications and Mobile Computing Conference (IWCMC'12)*, pp. 895 –900, doi:10.1109/IWCMC.2012.6314323.

[4] T. Chen, V. Forejt, M. Kwiatkowska, D. Parker & A. Simaitis (2013): *Automatic Verification of Competitive Stochastic Systems*. Formal Methods in System Design. To appear.

[5] T. Chen, V. Forejt, M. Kwiatkowska, D. Parker & A. Simaitis (2013): *PRISM-games: A Model Checker for Stochastic Multi-Player Games*. In: Proc. 19th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'13), LNCS 7795, Springer, pp. 187–193.

[6] M. Felegyhazi, J.-P. Hubaux & L. Buttyan (2006): *Nash equilibria of packet forwarding strategies in wireless ad hoc networks*. Mobile Computing, IEEE Transactions on 5(5), pp. 463 – 476, doi:10.1109/TMC.2006.68.

[7] Z. Li & H. Shen (2012): *Game-Theoretic Analysis of Cooperation Incentive Strategies in Mobile Ad Hoc Networks*. IEEE Transactions on Mobile Computing 11(8), pp. 1287 –1303, doi:10.1109/TMC.2011.151.

[8] S. Roy, C. Ellis, S. Shiva, D. Dasgupta, V. Shandilya & Q. Wu (2010): *A Survey of Game Theory as Applied to Network Security*. In: Proc. 43rd Hawaii International Conference on System Sciences (HICSS'10), pp. 1 –10, doi:10.1109/HICSS.2010.35.

Concurrent Game Structures with Roles*

Truls Pedersen† Sjur Dyrkolbotn‡ Piotr Kaźmierczak§ Erik Parmann¶

In the following paper we present a new semantics for the well-known strategic logic ATL. It is based on adding *roles* to concurrent game structures, that is at every state, each agent belongs to exactly one role, and the role specifies what actions are available to him at that state. We show advantages of the new semantics, provide motivating examples based on sensor networks, and analyze model checking complexity.

1 Introduction

ATL [1] is not only a highly-expressive and powerful strategic logic, but also has a relatively low (polynomial) model checking complexity. However, as investigated by Jamroga and Dix [5], in order for the complexity to be polynomial, the number of agents must be *fixed*. If the number of agents is taken as a parameter, model checking ATL is Δ_2^{P}-complete or Δ_3^{P}-complete depending on model representation [6]. Also, van der Hoek, Lomuscio and Wooldridge show in [3] that the complexity of model checking is polynomial only if an *explicit* enumeration of *all* components of the model is assumed. For models represented in *reactive modules language* (RML) complexity of model checking for ATL becomes as hard as the satisfiability problem for this logic, namely EXPTIME [3].

We present an alternative semantics that interprets formulas of ordinary ATL over concurrent game structures with *roles*. Such structures introduce an extra element – a set R of roles and associates each agent with exactly one role which are considered *homogeneous* in the sense that all consequences of the actions of the agents belonging to the topical role is captured by considering only the number of "votes" an action gets (one vote per agent).

We present the revised formalism for ATL in Section 2, discuss model checking results in Section 3 and conclude in Section 4.

2 Role-based semantics for ATL

In this section we will introduce *concurrent game structures with roles* (RCGS), illustrate them with an example and show in Theorem 1 that treating RCGS or CGS as the semantics of ATL are equivalent.

We will very often refer to sets of natural numbers from 1 to some number $n \geq 1$. To simplify the reference to such sets we introduce the notation $[n] = \{1, \ldots, n\}$. Furthermore we will let A^B denote the set of functions from B to A. We will often also work with tuples $v = \langle v_1, \ldots, v_n \rangle$ and view v as a function with domain $[n]$ and write $v(i)$ for v_i. Given a function $f : A \times B \to C$ and $a \in A$, we will use f_a to denote the function $B \to C$ defined by $f_a(b) = f(a,b)$ for all $b \in B$.

*A preliminary version of this paper was presented during LAMAS workshop held at AAMAS on June 4th 2012, and a talk based on that version was given at LBP workshop during ESSLLI summer school, August 2012. It is available on arXiv [2].

†Dept. of Information Science and Media Studies, University of Bergen, Norway. truls.pedersen@infomedia.uib.no

‡Durham Law School, Durham University, UK. s.k.dyrkolbotn@durham.ac.uk

§Dept. of Computing, Mathematics and Physics, Bergen University College, Norway. phk@hib.no

¶Dept. of Informatics, University Bergen, Norway. erik.parmann@ii.uib.no

F. Mogavero, A. Murano, and M.Y. Vardi (Eds.):
1st Workshop on Strategic Reasoning 2013 (SR'13)
EPTCS 112, 2013, pp. 61–69, doi:10.4204/EPTCS.112.11

Definition 1. *An* RCGS *is a tuple* $H = \langle \mathscr{A}, R, \mathscr{R}, Q, \Pi, \pi, \mathbb{A}, \delta \rangle$ *where:*

- $\mathscr{A}$ *is a non-empty set of players. In this text we assume* $\mathscr{A} = [n]$ *for some* $n \in \mathbb{N}$*, and we reserve* n *to mean the number of agents.*
- Q *is the non-empty set of states.*
- R *is a non-empty set of roles. In this text we assume* $R = [i]$ *for some* $i \in \mathbb{N}$*.*
- $\mathscr{R} : Q \times \mathscr{A} \to R$*. For a coalition* A *we write* $A_{r,q}$ *to denote the agents in* A *which belong to role* r *at* q*, and notably* $\mathscr{A}_{r,q}$ *are* all *the agents in role* r *at* q*.*
- Π *is a set of propositional letters and* $\pi : Q \to \wp(\Pi)$ *maps each state to the set of propositions true in it.*
- $\mathbb{A} : Q \times R \to \mathbb{N}^+$ *is the number of available actions in a given state for a given role.*
- *For* $\mathscr{A} = [n]$*, we say that the set of* complete *votes for a role* r *in a state* q *is* $V_r(q) = \{v_{r,q} \in [n]^{[\mathbb{A}(q,r)]} \mid \sum_{1 \le a \le \mathbb{A}(q,r)} v_{r,q}(a) = |\mathscr{A}_{r,q}|\}$*, the set of functions from the available actions to the number of agents performing the action. The functions in this set account for the actions of* all *the agents. The set of* complete *profiles at* q *is* $P(q) = \prod_{r \in R} V_r(q)$*. For each* $q \in Q$ *we have a transition function at* q*,* $\delta_q : P(q) \to Q$ *defining a partial function* $\delta : Q \times \bigcup_{q \in Q} P(q) \to Q$ *such that for all* $q \in Q$*,* $P \in P(q)$*,* $\delta(q, P) = \delta_q(P)$*.*

The following example illustrates how RCGS differs from an ordinary concurrent game structure:

Example 1 (Sensor networks)**.** *A wireless sensor network is a system composed of a number of (homogeneous) sensors that can be triggered by various stimuli. In Figure 1 we show a* 1-tier *(i.e., completely homogeneous) sensor network with n sensors. There are two states in the system with labels corresponding to an* indicator *of the network.* $\neg p$ *stands for* idle *state of the network, while* p *indicates that the network detected a stimulus. In this very simple example we say that* k *is our* threshold*, i.e. if at least* k *number of sensors detect something, then* p*. Since all the sensors behave in the same way we say the role of sensors is homogeneous. Hence the system can be modeled using only a single role. This gives us the model depicted in Figure 1. One can easily add another role to the model if needed, for example in a scenario with a "controller" who processes the reported signals, or in a* 2*-tier network with several types of sensors.*

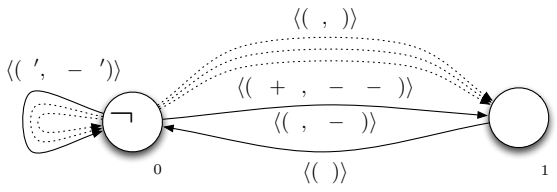


Figure 1: A depiction of H_1 – a simple 1-tier sensor network.

A more complex example is presented in Figure 2, where we add another role *to our structure, that of a* supervisor *or* controller*. The supervisor can act upon sensors' actions, i.e. if the sensors report that* p*, the supervisor can perform* q*. As illustrated by the drawing, the supervisor has three actions available: he can wait, he can reject the message or he can accept the message and proceed to state* q_2 *performing* q *(e.g., call the police in an intrusion detection scenario). Finally, in Figure 3 we sketch*

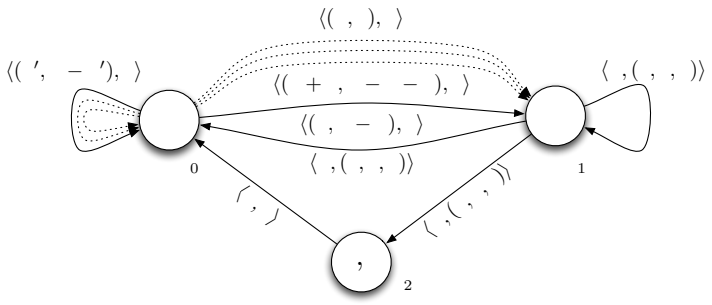


Figure 2: A sketch of structure H_2: a 1-tier sensor network with a supervisor.

a multi-tier *example, with two different types of sensors, n_1 and n_2, each type with its corresponding role. The transition function with the addition of a new role looks like this:*

$$\delta_{q_0}(\langle(x_1, n_1 - x_1), (x_2, n_2 - x_2)\rangle) = \begin{cases} q_1, & x_1 \geq t_1 \wedge x_2 \geq t_2 \\ q_0, & \textit{otherwise} \end{cases}$$

$$\delta_{q_1}(\langle n_1, n_2\rangle) = q_0$$

where t_1,t_2 are thresholds set according to significance *of the sensors.*

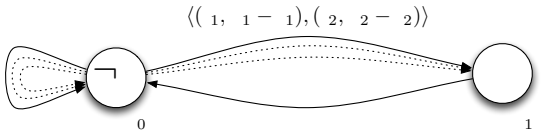


Figure 3: A sketch of structure H_3: a multi-tier sensor network example.

These simple structures show the benefit of using roles when modelling scenarios which involve a high degree of homogeneity among agents. In this simplified sensor setting a sensor either signals that he has made a relevant observation or he does not – a binary choice. If modelled using concurrent game structures without *roles, models would have 2^n number of possible action profiles in state q_0, since the identity of the agents signaling that they have made an observation has to be accounted for. This, however, is irrelevant for the high-level protocol – all that matters is how many sensors of a given type signal that they have made an observation. With roles we can exploit this, and we only need to account for the genuinely different scenarios that can occur – corresponding to the number of sensors of each type that decide to signal that they have made an observation. In the case of just a single role, this means that we get n as opposed to 2^n number of different profiles, and the size of the model goes from exponential to linear in the number of sensors. In general, as we will show in Section 3, we shift the exponential dependence in the size of models from the number of agents to the number of roles.*

Given a role r, a state q and a coalition A, the set of A-votes for r at q is $V_r(q,A)$, defined as:

$$V_r(q,A) = \left\{ v \in [|A_{r,q}|]^{[\mathbb{A}(q,r)]} \;\middle|\; \sum_{a \in [\mathbb{A}(q,r)]} v(a) = |A_{r,q}| \right\}.$$

The A-votes for r at q give the possible ways agents in A that are in role r at q can vote. Given a state q and a coalition A, we define the set of A-profiles at q:

$$P(q,A) = \{\langle v_1, \ldots, v_{|R|}\rangle \mid 1 \leq i \leq |R| : v_i \in V_r(q,A)\}.$$

For any $v \in V_r(q,A)$ and $w \in V_r(q,B)$ we write $v \leq w$ iff for all $i \in [\mathbb{A}(q,r)]$ we have $v(i) \leq w(i)$. If $v \leq w$, we say that w *extends* v. If $F = \langle v_1, \ldots, v_R\rangle \in P(q,A)$ and $F' = \langle v'_1, \ldots, v'_R\rangle \in P(q,B)$ with $v_i \leq v'_i$ for every $1 \leq i \leq |R|$, we say that $F \leq F'$ and that F extends F'. Given a (partial) profile F' at a state q we write $ext(q,F)$ for the set of all complete profiles that extend F'.

Given two states $q, q' \in Q$, we say that q' is a *successor* of q if there is some $F \in P(q)$ such that $\delta(q,F) = q'$. A *computation* is an infinite sequence $\lambda = q_0 q_1 \ldots$ of states such that for all positions $i \geq 0$, q_{i+1} is a successor of q_i. We follow standard abbreviations, hence a q-computation denotes a computation starting at q, and $\lambda[i]$, $\lambda[0,i]$ and $\lambda[i,\infty]$ denote the i-th state, the finite prefix $q_0 q_1 \ldots q_i$ and the infinite suffix $q_i q_{i+1} \ldots$ of λ for any computation λ and its position $i \geq 0$, respectively. An A-*strategy* for $A \subseteq \mathscr{A}$ is a function $s_A : Q \to \bigcup_{q \in Q} P(q,A)$ such that $s_A(q) \in P(q,A)$ for all $q \in Q$. That is, s_A maps states to A-profiles at that state. The set of all A-strategies is denoted by $strat(A)$. When needed to distinguish between different structures we write $strat(S,A)$ to indicate that we are talking about the set of strategies for A in structure S. If s is an $\mathscr{A}$-strategy and we apply δ_q to $s(q)$, we obtain a unique new state $q' = \delta_q(s(q))$. Iterating, we get the *induced* computation $\lambda_{s,q} = q_0 q_1 \ldots$ such that $q = q_0$ and $\forall i \geq 0 : \delta_{q_i}(s(q_i)) = q_{i+1}$. Given two strategies s and s', we say that $s \leq s'$ iff $\forall q \in Q : s(q) \leq s'(q)$. Given an A-strategy s_A and a state q we get an associated *set* of computations $out(s_A, q)$. This is the set of all computations that can result when at any state, the players in A are voting/acting in the way specified by s_A, that is $out(s_A,q) = \{\lambda_{s,q} \mid s \text{ is an } \mathscr{A}\text{-strategy and } s \geq s_A\}$.

Given the definitions above, we can interpret ATL formulas in the following manner, leaving out the propositional cases and abbreviations:

Definition 2. *Given a* RCGS S *and a state* q *in* S*, we define the satisfaction relation* $\models$ *inductively:*

- $S, q \models \langle\langle A\rangle\rangle \bigcirc \phi$ *iff there is* $s_A \in strat(A)$ *such that for all* $\lambda \in out(s_A, q)$*, we have* $S, \lambda[1] \models \phi$
- $S, q \models \langle\langle A\rangle\rangle \phi \,\mathscr{U}\, \phi'$ *iff there is* $s_A \in strat(A)$ *such that for all* $\lambda \in out(s_A, q)$ *we have* $S, \lambda[i] \models \phi'$ *and* $S, \lambda[j] \models \phi$ *for some* $i \geq 0$ *and for all* $0 \leq j < i$

Towards the statement that interpreting formulas over CGS and RCGS is equivalent (Theorem 1) we will describe a surjective translation function f translating each RCGS to a CGS. The following two lemmas will be useful in formulating the proof of Theorem 1.

The translation function f from RCGS to CGS is defined as follows:

$$f\langle \mathscr{A}, R, \mathscr{R}, Q, \Pi, \pi, \mathbb{A}, \delta\rangle = \langle \mathscr{A}, Q, \Pi, \pi, d, \delta'\rangle$$

where:

$$d_a(q) = \mathbb{A}(q,r) \qquad \text{where } a \in \mathscr{R}(q,r)$$
$$\delta'(q, \alpha_1, \ldots, \alpha_n) = \delta(q, v_1, \ldots, v_{|R|}) \qquad \text{where for each role } r$$
$$v_r = \langle |\{i \in \mathscr{R}(q,r) \mid \alpha_i = 1\}|, \ldots, |\{i \in \mathscr{R}(q,r) \mid \alpha_i = \mathbb{A}(q,r)\}|\rangle$$

We describe a surjective function $m : strat(f(S)) \to strat(S)$ mapping action tuples and strategies of $f(S)$ to profiles and strategies of S respectively. For all $A \subseteq \mathscr{A}$ and any action tuple for A at q,

$t_q = \langle \alpha_{a_1}, \alpha_{a_2}, ..., \alpha_{a_{|A|}} \rangle$ with $1 \leq \alpha_{a_i} \leq d_{a_i}(q)$ for all $1 \leq i \leq |A|$, the A-profile $m(t_q)$ is defined in the following way:

$$m(t_q) = \langle v(t_q,1), \ldots, v(t_q,|R|) \rangle \text{ where for all } 1 \leq r \leq |R| \text{ we have}$$
$$v(t_q,r) = \langle |\{a \in A_{r,q} \mid \alpha_a = 1\}|, \ldots, |\{a \in A_{r,q} \mid \alpha_a = \mathbb{A}(q,r)\}| \rangle$$

Lemma 1. *For any* RCGS S *and any* $A \subseteq \mathscr{A}$*, the function* $m : strat(f(S),A) \to strat(S,A)$ *is surjective.*

Proof. Let p_A be some strategy for A in S. We must show there is a strategy s_A in $f(S)$ such that $m(s_A) = p_A$. For all $q \in Q$, we must define $s_A(q)$ appropriately. Consider the profile $p_A(q) = \langle v_1, \ldots, v_{|R|} \rangle$ and note that by definition of a profile, all v_r for $1 \leq r \leq |R|$ are A-votes for r and that by definition of an A-vote, we have $\sum_{1 \leq i \leq \mathbb{A}(q,r)} v_r(i) = |A_{r,q}|$. Also, for all agents $a, a' \in A_{r,q}$ we know, by definition of f, that $d_a(q) = d_{a'}(q) = \mathbb{A}(q,r)$.

It follows that there are functions $\alpha : A \to \mathbb{N}^+$ such that for all $a \in A$, $\alpha(a) \in [d_a(q)]$ and $|\{a \in A_{r,q} \mid \alpha(a) = i\}| = v_r(i)$ for all $1 \leq i \leq \mathbb{A}(q,r)$, i.e.

$$v_r = \langle |\{a \in A_{r,q} | \alpha(a) = 1\}|, \ldots, |\{a \in A_{r,q} | \alpha(a) = \mathbb{A}(q,r)\}| \rangle$$

We choose some such α and $s_A = \langle \alpha(a_1), \ldots, \alpha(a_{|A|}) \rangle$. Having defined s_A in this way, it is clear that $m(s_A) = p_A$. □

It will be useful to have access to the set of states that can result in the next step when $A \subseteq \mathscr{A}$ follows strategy s_A at state q, $succ(q,s_A) = \{q' \in Q \mid \exists F \in ext(q,s_A) : \delta(q,F) = q'\}$. Given either a CGS or an RCGS S, we define the set of sets of states that a coalition A can *enforce* in the next state of the game:

$$force(S,q,A) = \{succ(q,s_A) \mid s_A \text{ is a strategy for } A \text{ in } S\}.$$

Using the surjective function m we can prove the following lemma, showing that the "next time" strength of any coalition A is the same in S as it is in $f(S)$.

Lemma 2. *For any* RCGS S*, and state* $q \in Q$ *and any coalition* $A \subseteq \mathscr{A}$*, we have* $force(S,A,q) = force(f(S),A,q)$.

Proof. By definition of $force$ and Lemma 1 it is sufficient to show that for all $s_A \in strat(f(S),A)$, we have $succ(S,m(s_A),q) = succ(f(S),s_A,q)$. We show $\subseteq$ as follows: Assume that $q' \in force(S,m(s_A),q)$. Then there is some complete profile $P = \langle v_1, \ldots, v_{|R|} \rangle$, extending $m(s_A)(q)$, such that $\delta(q,P) = q'$. Let $m(s_A)(q) = \langle w_1, \ldots, w_{|R|} \rangle$ and form $P' = \langle v'_1, \ldots, v'_{|R|} \rangle$ defined by $v'_i = v_i - w_i$ for all $1 \leq i \leq |R|$. Then each v'_i is an $(\mathscr{A} \setminus A)$-vote for role i, meaning that the sum of entries in the tuple v'_i is $|(\mathscr{A} \setminus A)_{r,q}|$. This means that we can define a function $\alpha : \mathscr{A} \to \mathbb{N}^+$ such that for all $a \in \mathscr{A}$, $\alpha(a) \in [d_a(q)]$ and for all $a \in A$, $\alpha(a) = s_a(q)$ and for every $r \in R$ and every $a \in (\mathscr{A} \setminus A)$, and every $1 \leq j \leq \mathbb{A}(q,r)$, $|\{a \in (\mathscr{A} \setminus A)_{r,q} \mid \alpha(a) = j\}| = v'_r(j)$. Having defined α like this it follows by definition of m that for all $1 \leq j \leq \mathbb{A}(q,r)$, $|\{a \in A_{r,q} \mid \alpha(a) = j\}| = w_r(j)$. Then for all $r \in R$ and all $1 \leq j \leq \mathbb{A}(q,r)$ we have $|\{a \in \mathscr{A}_{q,r} \mid \alpha(a) = j\}| = v_r(j)$. By definition of $f(S)$ it follows that $q' = \delta(q,P) = \delta'(q,\alpha)$ so that $q' \in force(f(S),s_A,q)$. We conclude that $force(S,f(s_A),q) \subseteq force(f(S),s_A,q)$. The direction $\supseteq$ follows easily from the definitions of m and f. □

We now state and prove the equivalence.

Theorem 1. *For any* RCGS S*, any* ϕ *and any* $q \in Q$*, we have* $S,q \models \phi$ *iff* $f(S),q \models_{\text{CGS}} \phi$*, where* f *is the surjective model-translation function.*

Proof. Given a structure S, and a formula ϕ, we define $true(S,\phi) = \{q \in Q \mid S,q \models \phi\}$. Equivalence of models S and $f(S)$ is now demonstrated by showing that the equivalence in next time strength established in Lemma 2 suffices to conclude that $true(S,\phi) = true(f(S),\phi)$ for all ϕ.

We prove the theorem by showing that for all ϕ, we have $true(S,\phi) = true(f(S),\phi)$. We use induction on complexity of ϕ. The base case for atomic formulas and the inductive steps for Boolean connectives are trivial, while the case of $\langle\langle A\rangle\rangle \bigcirc \phi$ is a straightforward application of Lemma 2. For the cases of $\langle\langle A\rangle\rangle \Box \phi$ and $\langle\langle A\rangle\rangle \phi \mathscr{U} \psi$ we rely on the following fixed point characterizations, which are well-known to hold for ATL, see for instance [4], and are also easily verified against definition 2:

$$\begin{aligned} \langle\langle A\rangle\rangle \Box \phi &\leftrightarrow \phi \wedge \langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle \Box \phi \\ \langle\langle A\rangle\rangle \phi_1 \mathscr{U} \phi_2 &\leftrightarrow \phi_2 \vee (\phi_1 \wedge \langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle \phi_1 \mathscr{U} \phi_2 \end{aligned} \tag{1}$$

We show the induction step for $\langle\langle A\rangle\rangle \Box \phi$, taking as induction hypothesis $true(S,\phi) = true(f(S),\phi)$. The first equivalence above identifies $Q' = true(S, \langle\langle A\rangle\rangle \Box \phi)$ as the maximal subset of Q such that ϕ is true at every state in Q' and such that A can enforce a state in Q' from every state in Q', i.e. such that $\forall q \in Q' : \exists Q'' \in force(q,A) : Q'' \subseteq Q'$. Notice that a unique such set always exists. This is clear since the union of two sets satisfying the two requirements will itself satisfy them (possibly the empty set). The first requirement, namely that ϕ is true at all states in Q', holds for S iff if holds for $f(S)$ by induction hypothesis. Lemma 2 states $force(S,q,A) = force(f(S),q,A)$, and this implies that also the second requirement holds in S iff it holds in $f(S)$. From this we conclude $true(S, \langle\langle A\rangle\rangle \Box \phi) = true(f(S), \langle\langle A\rangle\rangle \Box \phi)$ as desired. The case for $\langle\langle A\rangle\rangle \phi \mathscr{U} \psi$ is similar, using the second equivalence. □

Example 2 (Sensor networks contd.)**.** *To further illustrate the use of* ATL *interpreted over* RCGS*, we provide example formulas that are related to the structures shown in Example 1.*

In the structure depicted in Figure 1, if at least k sensors signal something, p becomes true (e.g. the alarm is raised). This is expressed by formula $\langle\langle A\rangle\rangle \bigcirc p$ which is satisfied in the structure from Figure 1, i.e. $H_1, q_0 \vDash \langle\langle A\rangle\rangle \bigcirc p$ whenever $|A \cap \mathscr{R}(q_0,1)| \geq k$. In Figure 2, the supervisor decides whether signals that indicate p are strong enough in order for him *to signal q, e.g. raise the alarm. In this scenario, the sensors alone cannot raise the alarm, hence $H_2, q_0 \not\models \langle\langle A\rangle\rangle \Diamond q$ whenever $A \cap \mathscr{R}(q_1,2) = \emptyset$ (which means that whenever the coalition A does not include the supervisor, q cannot be enforced). On the other hand, $H_2, q_0 \models \langle\langle A\rangle\rangle \bigcirc \langle\langle B\rangle\rangle \bigcirc q$ whenever $|A \cap \mathscr{R}(q_0,1)| \geq k$ and $B \cap \mathscr{R}(q_1,2) \neq \emptyset$ (which means that the coalition of agents without a supervisor can enable the supervisor to take action).*

3 Model checking and the size of models

In this section we will see how using roles can lead to a dramatic decrease in the size of ATL models. We first investigate the size of models in terms of the number of roles, players and actions, and then we analyze model checking of ATL over concurrent game structures with roles.

Given a set of numbers $[a]$ and a number n, it is a well-known combinatorial fact that the number of ways in which to choose n elements from $[a]$, allowing repetitions, is $\frac{(n+(a-1))!}{n!(a-1)!}$. Furthermore, this number satisfies the following two inequalities:[1]

$$\frac{(n+(a-1))!}{n!(a-1)!} \leq a^n \quad \text{and} \quad \frac{(n+(a-1))!}{n!(a-1)!} \leq n^a. \tag{2}$$

[1] If this is not clear, remember that n^a and a^n are the number of functions $[n]^{[a]}$ and $[a]^{[n]}$ respectively. It should not be hard to see that all ways in which to choose n elements from a induce non-intersecting sets of functions of both types.

These two inequalities provide us with an upper bound on the *size* of RCGS models that makes it easy to compare their sizes to that of CGS models. Typically, the size of concurrent game structures is dominated by the size of the domain of the transition function. For an RCGS and a given state $q \in Q$ this is the number of complete profiles at q. To measure it, remember that every complete profile is an $|R|$-tuple of votes v_r, one for each role $r \in R$. Also remember that a vote v_r for $r \in R$ is an $\mathbb{A}(q,r)$-tuple such that the sum of entries is $|\mathscr{A}_{q,r}|$. Equivalently, the vote v_r can be seen as the number of ways in which we can make $|\mathscr{A}_{q,r}|$ choices, allowing repetitions, from a set of $\mathbb{A}(q,r)$ alternatives. Looking at it this way, we obtain:

$$|P(q)| = \prod_{r \in R} \frac{(|\mathscr{A}_{q,r}| + (\mathbb{A}(q,r)-1))!}{|\mathscr{A}_{q,r}|!(\mathbb{A}(q,r)-1))!}.$$

We sum over all $q \in Q$ to obtain what we consider to be the size of an RCGS S. In light of Equation 2, it follows that the size of S is upper bounded by both of the following expressions.

$$\mathscr{O}(\textstyle\sum_{q \in Q} \prod_{r \in R} |\mathscr{A}_{q,r}|^{\mathbb{A}(q,r)}) \quad \text{and} \quad \mathscr{O}(\textstyle\sum_{q \in Q} \prod_{r \in R} \mathbb{A}(q,r)^{|\mathscr{A}_{q,r}|}). \tag{3}$$

We observe that growth in the size of models is polynomial in $a = max_{q \in Q, r \in R} \mathbb{A}(r,q)$ if $n = |\mathscr{A}|$ and $|R|$ is fixed, and polynomial in $p = max_{q \in Q, r \in R} |\mathscr{A}_{q,r}|$ if a and $|R|$ are fixed. This identifies a significant potential advantage arising from introducing roles to the semantics of ATL. The size of a CGS M, when measured in the same way, replacing complete profiles at q by complete action tuples at q, grows exponentially in the players whenever the players have more than one action. We stress that we are *not* just counting the number of transitions in our models differently. We do have an additional parameter, the roles, but this is a new semantic construct that gives rise to genuinely different semantic structures. We have established that it is possible to use them to give the semantics of ATL, but this does not mean that there is not more to be said about them. Particularly crucial is the question of model checking over RCGS models.

3.1 Model checking using roles

For ATL there is a well known model checking algorithm [1]. It does model checking in time linear in the length of the formula and the size of the model. Given a structure S, and a formula ϕ, the standard model checking algorithm $mcheck(S,\phi)$ returns the set of states of S where ϕ holds.

The algorithm depends on a function $enforce(S,A,q,Q')$, which given a structure S, a coalition A, a state $q \in Q$ and a set of states Q' answers true or false depending on whether or not A can enforce Q' from q. This is the only part of the standard algorithm that needs to be modified to accommodate roles.

for $F \in P(q,A)$ **do**
  $p \leftarrow true$
  for $F' \in ext(q,F)$ **do**
    if $\delta(q,F') \notin Q'$ **then**
      $p \leftarrow false$
  if $p = true$ **then**
    return $true$
return $false$

Figure 4: $enforce(S,A,q,Q')$

For all profiles $F \in P(q,A)$ the $enforce$ algorithm runs through all complete profiles $F' \in P(q)$ that extend F. It is polynomial in the number of complete profiles, since for any coalition A and state q we have $|P(q,A)| \leq |P(q)|$, meaning that the complexity of $enforce$ is upper bounded by $|P(q)|^2$. Given a fixed length formula and a fixed number of states, $enforce$ dominates the running time of $mcheck$. It follows that model checking of ATL over concurrent game structures with roles is polynomial in the size of the model. We summarize this result.

Proposition 1. *Given a* CGS *S and a formula ϕ, $mcheck(S,\phi)$ takes time $\mathscr{O}(le^2)$ where l is the length of ϕ and $e = \sum_{q \in Q} |P(q)|$ is the total number of transitions in S*

Since model checking ATL over CGSs takes only linear time, $\mathscr{O}(le)$, adding roles apparently makes model checking harder. On the other hand, the *size* of CGS models can be bigger by an exponential factor, making model checking much easier after adding roles. In light of the bounds we have on the size of models, c.f. Equation 3, we find that as long as the roles and the actions remain fixed, complexity of model checking is only polynomial in the number of agents. This is a potentially significant argument in favor of including roles in the semantics.

Roles should be used at the modeling stage, as they give the modeler an opportunity for exploiting homogeneity of the system under consideration. We think that it is reasonable to hypothesize that in practice, most large scale multi-agent systems that lend themselves well to modeling by ATL exhibit significant homogeneity.

The question arises as to whether or not using an RCGS is *always* the best choice, or if there are situations when the losses incurred in the complexity of model checking outweigh the gains we make in terms of the size of models. We conclude with the following proposition, also shown in [2], which states that as long we use the standard algorithm, model checking any CGS M can be done at least as quickly by model checking an *arbitrary* $S \in f^{-}(M)$.

Proposition 2. *Given any* CGS*-model M and any formula ϕ, let $c(mcheck(M,\phi))$ denote the complexity of running $mcheck(M,\phi)$. We have, for all $S \in f^{-}(M)$, that complexity of running $mcheck(S,\phi)$ is $\mathscr{O}(c(mcheck(M,\phi)))$*

Proof. It is clear that for any $S \in f^{-}(M)$, running $mcheck(S,\phi)$ and $mcheck(M,\phi)$, a difference in overall complexity can arise only from a difference in the complexity of $enforce$. So we compare the complexity of $enforce(S,A,q,Q'')$ and $enforce(M,A,q,Q'')$ for some arbitrary $q \in Q$, $Q'' \subseteq Q$. The complexity in both cases involves passing through all complete extensions of all strategies for A at q. The sizes of these sets can be compared as follows, the first inequality is an instance of Equation 2 and the equalities follow from definition of f and the fact that $M = f(S)$.

$$\begin{aligned}
\prod_{r\in R}\left(\frac{(|A_{r,q}|+(\mathbb{A}(r,q)-1))!}{|A_{r,q}|!(\mathbb{A}(r,q)-1)!}\right)\times\prod_{r\in R}\left(\frac{((|\mathscr{A}_{q,r}|-|A_{r,q}|)+(\mathbb{A}(r,q)-1))!}{(|\mathscr{A}_{q,r}|-|A_{r,q}|)!(\mathbb{A}(r,q)-1)!}\right)& \\
\leq\left(\prod_{r\in R}\mathbb{A}(r,q)^{|A_{r,q}|}\times\prod_{r\in R}\mathbb{A}(r,q)^{|\mathscr{A}_{q,r}|-|A_{r,q}|}\right)& \\
=\prod_{r\in R}\left(\prod_{a\in A_{r,q}}\mathbb{A}(r,q)\right)\times\prod_{r\in R}\left(\prod_{a\in\mathscr{A}_{a,r}\setminus A_{r,q}}\mathbb{A}(r,q)\right)& \\
=\left(\prod_{a\in A}d_a(q)\times\prod_{a\in\mathscr{A}\setminus A}d_a(q)\right)=\prod_{a\in\mathscr{A}}d_a(q)&
\end{aligned}$$

We started with the number of profiles (transitions) we need to inspect when running $enforce$ on S at q, and ended with the number of action tuples (transitions) we need to inspect when running $enforce$ on $M = f(S)$. Since we showed the first to be smaller or equal to the latter and the execution of all other elements of *mcheck* are identical between S and M, the claim follows. □

4 Conclusions, related and future work

In this paper we have described a new type of semantics for the strategic logic ATL. We have provided illustrating examples and argued that although in principle model checking ATL interpreted over concurrent game structures with roles is harder than the standard approach, it is still polynomial and can generate exponentially smaller models. We believe this provides evidence that concurrent game structures with roles are an interesting semantics for ATL, and should be investigated further.

Relating our work to ideas already present in the literature we find it somewhat similar to the idea of exploiting symmetry in model checking, as investigated by Sistla and Godefroid [7]. However, our approach is different, since we look at agent symmetries in ATL as the basis of a new semantics. When it comes to work related directly to strategic logics, we find no similar ideas present, hence concluding that our approach is indeed novel.

For future work we plan on investigating the homogeneous aspect of our 'roles' in more depth. We are currently working on a derivative of ATL with a different language that will fully exploit the role based semantics.

Acknowledgments: We thank Pål Grønås Drange, Valentin Goranko and Alessio Lomuscio for helpful comments. Piotr Kaźmierczak's work was supported by the Research Council of Norway project 194521 (FORMGRID).

References

[1] Rajeev Alur, Thomas A. Henzinger & Orna Kupferman (2002): *Alternating-time temporal logic*. *Journal of the ACM (JACM)* 49(5), pp. 672–713, doi:10.1145/585265.585270.

[2] Sjur Dyrkolbotn, Piotr Kaźmierczak, Erik Parmann & Truls Pedersen (2012): *No big deal: introducing roles to reduce the size of ATL models*. Available at `http://arxiv.org/abs/1204.3495`.

[3] Wiebe van der Hoek, Alessio Lomuscio & Michael Wooldridge (2006): *On the complexity of practical ATL model checking*. In: *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, ACM, pp. 201–208, doi:10.1145/1160633.1160665.

[4] Wojciech Jamroga (2009): *Easy Yet Hard: Model Checking Strategies of Agents*. In Michael Fisher, Fariba Sadri & Michael Thielscher, editors: *Computational Logic in Multi-Agent Systems*, Springer-Verlag, pp. 1–12, doi:10.1007/978-3-642-02734-5_1.

[5] Wojciech Jamroga & Jürgen Dix (2005): *Do agents make model checking explode (computationally)?* In M. Pechoucek, P. Petta & L. Z. Varga, editors: *Multi-Agent Systems and Applications IV (LNAI Volume 3690)*, doi:10.1007/11559221_40.

[6] François Laroussinie, Nicolas Markey & Ghassan Oreiby (2007): *On the expressiveness and complexity of ATL*. In: *Proceedings of the 10th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'07), volume 4423 of Lecture Notes in Computer Science*, Springer, pp. 243–257, doi:10.1007/978-3-540-71389-0_18.

[7] A. Prasad Sistla & Patrice Godefroid (2004): *Symmetry and reduced symmetry in model checking*. *ACM Trans. Program. Lang. Syst.* 26(4), pp. 702–734, doi:10.1145/1011508.1011511.

Reasoning about Strategies under Partial Observability and Fairness Constraints

Simon Busard, Charles Pecheur*
ICTEAM Institute,
Université catholique de Louvain,
Louvain-la-Neuve, Belgium
simon.busard@uclouvain.be
charles.pecheur@uclouvain.be

Hongyang Qu
Dept. of Computer Science,
University of Oxford,
Oxford, United Kingdom
Hongyang.Qu@cs.ox.ac.uk

Franco Raimondi
Dept. of Computer Science,
Middlesex University,
London, United Kingdom
f.raimondi@mdx.ac.uk

A number of extensions exist for Alternating-time Temporal Logic; some of these mix strategies and partial observability but, to the best of our knowledge, no work provides a unified framework for strategies, partial observability and fairness constraints. In this paper we propose $ATLK^F_{po}$, a logic mixing strategies under partial observability and epistemic properties of agents in a system with fairness constraints on states, and we provide a model checking algorithm for it.

1 Introduction

A number of extensions exist for Alternating-time Temporal Logic; starting from [7], partial observability has been investigated by many authors, see for instance [8] and references therein. But, to the best of our knowledge, no work provides a unified framework for strategies, partial observability and fairness constraints. For example, Jamroga and van der Hoek proposed, among other logics, ATOL, mixing partial observability with strategies of agents [10]. Along the same lines, Schobbens studied ATL_{ir}[14], seen as the minimal ATL-based logic for strategies under partial observability [9]. On the other hand, some efforts have been made on bringing fairness to ATL. For instance the work of Alur et al. [1], or the work of Klüppelholz and Baier [11] introduce the notion of fairness constraints on actions, asking for an infinitely often enabled action to be taken infinitely often. For temporal and epistemic logics, however, fairness conditions are normally provided on *states*. Furthermore, it has been shown that (weak, strong or unconditional) fairness constraints on actions, can be reduced to (weak, strong or unconditional, respectively) fairness constraints on states (see [2], for instance). In this paper we propose $ATLK^F_{po}$, a logic mixing strategies under partial observability and epistemic properties of agents in a system with unconditional fairness constraints *on states*, and we provide a model checking algorithm for it.

To motivate the need for fairness constraints in ATL under partial observability, consider the simple card game example in [10]. The game is played between a player and a dealer. It uses three cards, *A*, *K* and *Q*; *A* wins over *K*, *K* wins over *Q* and *Q* wins over *A*. First, the dealer gives one card to the player, keeps one and leaves the last one on table. Then the player can keep his card or swap it with the one on the table. The player wins if his card wins over the dealer's card. Under ATL_{ir} semantics, the player cannot win the game: he cannot distinguish between, for example, $< A,K >$ and $< A,Q >$ (where $< a,b >$ means "player has card *a*, dealer has card *b*") and thus has to make the same action in both states, with a different result in each case. Consider now a variation of this game: the game does not terminate after the first round. Instead, if the player does not win, cards are redistributed. In this case, too, the player cannot win the game: for instance, he will have to choose between keeping or swapping cards in $< A,K >$ and $< A,Q >$, so he won't be able to enforce a win because the dealer (that chooses

*This work is supported by the European Fund for Regional Development and by the Walloon Region.

F. Mogavero, A. Murano, and M.Y. Vardi (Eds.):
1st Workshop on Strategic Reasoning 2013 (SR'13)
EPTCS 112, 2013, pp. 71–79, doi:10.4204/EPTCS.112.12

the given cards) can be unfair and always give the losing pair. But if we add one fairness constraint per intermediate state—i.e. the states in which the player has to choose between swapping or keeping—the player has a strategy to finally win the game. In this case, we only consider paths along which all fairness constraints are met infinitely often: this situation corresponds to a fair dealer, giving the cards randomly. The player can thus finally win because $< A,K >$ will eventually happen—even if he cannot distinguish it from $< A,Q >$—, so he knows a strategy to win at least a round: keeping his card.

Another example of application of fairness constraints in ATL is Multi-Agent Programs [5]. These programs are composed of interleaved agent programs and fairness constraints are used to avoid unfair interleaving. Dastani and Jamroga express fairness as formulae of the logic ATL* [5]; in this paper, instead, we deal only with ATL and therefore fairness constraints cannot be expressed as formulae of the logic. The situation is similar to the case of LTL versus CTL model checking: in the first case model checking fairness is reduced to model checking a more complex formula using the same verification algorithms; in the second case fairness is incorporated into bespoke verification algorithms. In our work we chose ATL over ATL* because of complexity considerations (see Section 3).

The rest of the paper is structured as follows: Section 2 presents the syntax and semantics of $ATLK^F_{po}$ and Section 3 presents two model checking algorithms for the logic. Finally, Section 4 summarizes the contribution and draws some future work.

2 Syntax and Semantics

This section presents the syntax and semantics of $ATLK^F_{po}$, an extension of ATL with partial observability under fairness constraints on states. An extension with full observability under the same fairness constraints $ATLK^F_{fo}$ is also presented because the model checking algorithm for $ATLK^F_{po}$ relies on the one for $ATLK^F_{fo}$.

Syntax Both logics share the same syntax, composed of the standard Boolean connectors ($\vee$, $\wedge$, $\neg$, etc.), CTL operators (EX, EU, EG, etc.) [4], knowledge operators (K_{ag}, E_Γ, D_Γ, C_Γ) [6] and strategic operators ($\langle\Gamma\rangle X$, $\langle\Gamma\rangle G$, $\langle\Gamma\rangle U$, $\langle\Gamma\rangle W$ and their $[\Gamma]$ counterparts) [1].

Models and notation $ATLK^F_{fo}$ and $ATLK^F_{po}$ formulae are interpreted over models $M = \langle Ag, S, Act, T, I, \{\sim_i\}, V, F\rangle$ where (1) Ag is a set of n agents; (2) $S = S_1 \times ... \times S_n$ is a set of global states, each of which is composed of n local states, one for each agent; (3) $Act = Act_1 \times ... \times Act_n$ is a set of joint actions, each of which is composed of n actions, one for each agent; (4) $T \subseteq S \times Act \times S$ is a transition relation between states in S and labelled with joint actions (we write $s \xrightarrow{a} s'$ if $(s,a,s') \in T$); (5) $I \subseteq S$ is the a set of initial states; (6) $\{\sim_i\}$ is a set of equivalence relations between states, and $\sim_i$ partitions the set of states in terms of knowledge of agent i—$s \sim_i s'$ iff $s_i = s'_i$, i.e two states are indistinguishable for agent i if they share the same local state for i; (7) $V : S \to 2^{AP}$ labels states with atomic propositions of AP; (8) $F \subseteq 2^S$ is a set of fairness constraints, each of which is a subset of states.

A joint action $a = (a_1, ..., a_n)$ *completes* a partially joint action $a_\Gamma = (a'_i, ..., a'_j)$ composed of actions of agents in $\Gamma \subseteq Ag$—written $a_\Gamma \sqsubseteq a$—if actions in a for agents in Γ correspond to actions in a_Γ. Furthermore, we define the function $img : S \times Act \to 2^S$ as $img(s,a) = \{s' \in S | s \xrightarrow{a} s'\}$, i.e. $img(s,a)$ is the set of states reachable in one step from s through a.

A model M represents a non-deterministic system where each agent has an imperfect information about the current global state. One restriction is made on T: $\forall s,s' \in S, s \sim_i s' \implies enabled(s,i) = enabled(s',i)$ where $enabled(s,i) = \{a_i \in Act_i | \exists s' \in S, a \in Act$ s.t. $(a_i) \sqsubseteq a \wedge s \xrightarrow{a} s'\}$. This means that

the actions an agent can perform in two epistemically equivalent states are the same. The *enabled* function is straightforwardly extended to groups of agents.

A *path* in a model M is a sequence $\pi = s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} ...$ of elements of T. We use $\pi(d)$ for s_d. A state s is *reachable* in M if there exist a path π and $d \geq 0$ such that $\pi(0) \in I$ and $\pi(d) = s$. A path π is *fair* according to a set of fairness conditions $F = \{f_1, ..., f_k\}$ if for each fairness condition f, there exist infinitely many positions $d \geq 0$ such that $\pi(d) \in f$. A state s is *fair* if there exists a fair path starting at s.

A *strategy* for agent i is a function $f_i : S \to Act_i$ where, for any state s, $f_i(s) \in enabled(s,i)$; a strategy maps each state to an enabled action. We call these strategies *global strategies*. A *uniform strategy* for agent i is a global strategy f_i where $\forall s, s' \in S, s' \sim_i s \implies f_i(s) = f_i(s')$, i.e. agent i cannot choose two different actions for two indistinguishable states. The *strategy outcomes* from a state s for a strategy f_i, denoted with $out(s, f_i)$, is the set of paths a strategy can enforce, i.e. $out(s, f_i) = \{\pi = s_0 \xrightarrow{a_1} s_1 ... | s_0 = s \wedge \forall d \geq 0, s_{d+1} \in img(s_d, a_{d+1}) \wedge (f_i(s_d)) \sqsubseteq a_{d+1}\}$. The definition of outcomes is naturally extended to sets of strategies for a subset of agents.

Semantics The semantics of both logics are defined over states of a model M by defining the relations $M, s \models^F_{fo} \phi$ and $M, s \models^F_{po} \phi$, for $ATLK^F_{fo}$ and $ATLK^F_{po}$, respectively. M can be omitted when clear from the context. Both relations share a part of their semantics; we write $s \models^F \phi$ if $s \models^F_{fo} \phi$ and $s \models^F_{po} \phi$. The $s \models^F_{fo} \phi$ and $s \models^F_{po} \phi$ relations are recursively defined over the structure of ϕ and follow the standard interpretation for most of the operators. $s \models^F p$ if $p \in V(s)$; $\vee$ and $\neg$ are interpreted in the natural way. $s \models^F K_i \phi$ if ϕ is true in all fair reachable states indistinguishable from s for agent i, $s \models^F E_\Gamma \phi$ if all agents in Γ know ϕ, $s \models^F D_\Gamma \phi$ if, by putting all their knowledge in common, agents of Γ would know ϕ, and $s \models^F C_\Gamma \phi$ if ϕ is common knowledge among agents of Γ [6]. $s \models^F E\psi$ if there is a path π starting at s satisfying ψ, $\pi \models^F X\phi$ if $\pi(1)$ satisfies ϕ, $\pi \models^F \phi_1 U \phi_2$ if ϕ_1 is true along the path until ϕ_2 is true, $\pi \models G\phi$ if ϕ is always true along π, and $\pi \models \phi_1 W \phi_2$ if $\pi \models (\phi_1 U \phi_2) \vee G\phi_1$ [4].

The meaning of the $\langle\Gamma\rangle$ operator is different in the two semantics:
(i) $s \models^F_{fo} \langle\Gamma\rangle\psi$ iff there exists a set of **global strategies** f_Γ, one for each agent in Γ, such that for all **fair paths** $\pi \in out(s, f_\Gamma), \pi \models^F \psi$;
(ii) $s \models^F_{po} \langle\Gamma\rangle\psi$ iff there exists a set of **uniform strategies** f_Γ, one for each agent in Γ, such that for all $s' \sim_\Gamma s$, for all **fair paths** $\pi \in out(s', f_\Gamma), \pi \models^F \psi$.

The $[\Gamma]$ operator is the dual of $\langle\Gamma\rangle$: $s \models^F [\Gamma]\psi$ iff $s \models^F \neg\langle\Gamma\rangle\neg\psi$.

3 Model Checking $ATLK^F_{fo}$ and $ATLK^F_{po}$

Model checking $ATLK^F_{fo}$ The model checking algorithm for $ATLK^F_{fo}$ is defined by the function $[\![.]\!]^F_{fo} : ATLK^F_{fo} \to 2^S$ returning the set of states of a given model M satisfying a given $ATLK^F_{fo}$ property. This function is defined in the standard way for Boolean connectors, CTL and knowledge operators [4, 13]. The $[\Gamma]$ operators are evaluated as follows:

$$[\![[\Gamma]X\phi]\!]^F_{fo} = Pre_{[\Gamma]}([\![\phi]\!]^F_{fo} \cap Fair_{[\Gamma]})$$

$$[\![[\Gamma]\phi_1 U \phi_2]\!]^F_{fo} = \mu Z.([\![\phi_2]\!]^F_{fo} \cap Fair_{[\Gamma]}) \cup ([\![\phi_1]\!]^F_{fo} \cap Pre_{[\Gamma]}(Z))$$

$$[\![[\Gamma]G\phi]\!]^F_{fo} = \nu Z.[\![\phi]\!]^F_{fo} \cap \bigcap_{f \in F} Pre_{[\Gamma]}(\mu Y.(Z \cap f) \cup ([\![\phi]\!]^F_{fo} \cap Pre_{[\Gamma]}(Y)))$$

$$[\![[\Gamma]\phi_1 W \phi_2]\!]^F_{fo} = \begin{array}{l} \nu Z.([\![\phi_2]\!]^F_{fo} \cap Fair_{[\Gamma]}) \\ \quad \cup ([\![\phi_1]\!]^F_{fo} \cap \bigcap_{f \in F} Pre_{[\Gamma]}(\mu Y.([\![\phi_2]\!]^F_{fo} \cap Fair_{[\Gamma]}) \cup (Z \cap f) \cup ([\![\phi_1]\!]^F_{fo} \cap Pre_{[\Gamma]}(Y)))) \end{array}$$

where $Pre_{[\Gamma]}(Z) = \{s | \forall a_\Gamma \in enabled(s,\Gamma), \exists a \text{ s.t. } a_\Gamma \sqsubseteq a \wedge img(s,a) \cap Z \neq \emptyset\}$ and $Fair_{[\Gamma]} = [\![[\Gamma] G\, true]\!]^F_{fo}$. $\mu Z.\tau(Z)$ and $\nu Z.\tau(Z)$ are the least and greatest fix points of function $\tau(Z)$. Intuitively, the $Pre_{[\Gamma]}(Z)$ operator returns the set of states in which Γ cannot avoid to reach a state of Z. Thus, $[\![[\Gamma] G\phi]\!]^F_{fo}$ returns the set of states in which Γ cannot avoid a path of states of $[\![\phi]\!]^F_{fo}$ going through all fairness constraints infinitely often; $Fair_{[\Gamma]}$ is the set of states in which Γ cannot avoid a fair path. Note that the $\langle\Gamma\rangle$ operators can be computed using the $[\Gamma]$ and $\neg$ operators, but can also be computed directly using the dual forms from the ones above. For example $[\![\langle\Gamma\rangle G\phi]\!]^F_{fo} = \nu Z.([\![\phi]\!]^F_{fo} \cup \overline{Fair_{[\Gamma]}}) \cap Pre_{\langle\Gamma\rangle}(Z)$, where $Pre_{\langle\Gamma\rangle}(Z) = \overline{Pre_{[\Gamma]}(\overline{Z})} = \{s | \exists a_\Gamma \in enabled(s,\Gamma)$ such that $\forall a, a_\Gamma \sqsubseteq a \implies img(s,a) \subseteq Z\}$. $\overline{Z} \subseteq S$ is the complement of the set $Z \subseteq S$.

The correctness of the model checking algorithm for $ATLK^F_{fo}$ follows from Theorem 1.

Theorem 1. *For all states $s \in S$, $s \models^F_{fo} \phi$ if and only if $s \in [\![\phi]\!]^F_{fo}$.*

Proof sketch. First, $Reach_{[\Gamma]}(P_1,P_2) = \mu Y.P_2 \cup (P_1 \cap Pre_{[\Gamma]}(Y))$ computes the set of states in which Γ cannot avoid a finite path of states of P_1 to a state of P_2. We can prove it by induction over the computation of the least fix point. It is true by definition of the least fix point and the $Pre_{[\Gamma]}$ operation.

Then, for the $[\Gamma]G\phi$ operator, $[\![[\Gamma]G\phi]\!]^F_{fo} = \nu Z.[\![\phi]\!]^F_{fo} \cap \bigcap_{f\in F} Pre_{[\Gamma]}(\mu Y.(Z \cap f) \cup ([\![\phi]\!]^F_{fo} \cap Pre_{[\Gamma]}(Y)))$ $= \nu Z.[\![\phi]\!]^F_{fo} \cap \bigcap_{f\in F} Pre_{[\Gamma]}(Reach_{[\Gamma]}([\![\phi]\!]^F_{fo}, Z \cap f))$ computes the set of states in which Γ cannot avoid a fair path (i.e. going through each $f \in F$ infinitely often) that satisfies $G\phi$. We prove it by induction over the computation of the greatest fix point and by using what has been proved just above.

Thanks to this, we can easily prove that $Fair_{[\Gamma]} = [\![[\Gamma] G true]\!]^F_{fo}$ computes the set of states in which Γ cannot avoid a fair path (it is just a particular case of the $[\Gamma]G$ operator).

Then, $[\Gamma]X$ and $[\Gamma]U$ operators compute the set of states in which Γ cannot avoid a successor in $[\![\phi]\!]^F_{fo}$ in which Γ cannot avoid a fair path, respectively in which Γ cannot avoid a finite path through states of $[\![\phi_1]\!]^F_{fo}$ to a state of $[\![\phi_2]\!]^F_{fo}$, in which Γ cannot avoid a fair path. In particular, the proof for $[\Gamma]U$ directly follows from the proof for $Reach_{[\Gamma]}$.

Finally, the proof for the $[\Gamma]W$ operator is similar to the one for $[\Gamma]G$ operator. The proof of correctness of the algorithms for $\langle\Gamma\rangle$ operators follows from the proof for $[\Gamma]$ operators, the duality of these operators and standard fix point properties. □

Model checking $ATLK^F_{po}$ – basic algorithm A basic algorithm is presented in Algorithm 1. It relies on the model checking algorithm for $ATLK^F_{fo}$. It uses two sub-algorithms: *Split* and $[\![.]\!]^F_{fo}|_{strat}$, where *strat* is a strategy represented as a set of state/action pairs. The latter is a modified version of the algorithm described in the previous section with $Pre_{\langle\Gamma\rangle}|_{strat}$ replacing $Pre_{\langle\Gamma\rangle}$ where $Pre_{\langle\Gamma\rangle}|_{strat}(Z) = \{s | \exists a_\Gamma \in enabled(s,\Gamma)$ such that $\langle s, a_\Gamma\rangle \in strat \wedge \forall a, a_\Gamma \sqsubseteq a \implies img(s,a) \subseteq Z\}$, i.e., $Pre_{\langle\Gamma\rangle}|_{strat}(Z)$ is $Pre_{\langle\Gamma\rangle}(Z)$ restricted to states and actions allowed by *strat*. Furthermore, $[\![.]\!]^F_{fo}|_{strat}$ recursively calls $[\![.]\!]^F_{po}$ on sub-formulae, instead of $[\![.]\!]^F_{fo}$.

The *Split* algorithm is given in Algorithm 2. $Split(S \times Act_\Gamma)$ returns the set of uniform strategies of the system (a uniform strategy is represented by the action for group Γ allowed in each state, and this action needs to be the same for each state in the same equivalence class).

Intuitively, Algorithm 1 computes, for each possible uniform strategy *strat*, the set of states for which the strategy is winning, and then keeps only the states s for which the strategy is winning for all states equivalent to s.

Before proving the correctness of the basic algorithm, let's prove the correctness of the *Split* algorithm.

Algorithm 1: $[\![\langle\Gamma\rangle\psi]\!]^F_{po}$

Data: M a given (implicit) model, Γ a subset of agents of M, ψ an $ATLK^F_{po}$ path formula.
Result: The set of states of M satisfying $\langle\Gamma\rangle\psi$.

$sat = \{\}$
for $strat \in Split(S \times Act_\Gamma)$ **do**
  $winning = [\![\langle\Gamma\rangle\psi]\!]^F_{fo}|_{strat}$
  $sat = sat \cup \{s \in winning | \forall s' \sim_\Gamma s, s' \in winning\}$
return sat

Algorithm 2: $Split(Strats)$

Data: $Strats \subseteq S \times Act_\Gamma$.
Result: The set of all the largest subsets SA of $Strats \subseteq S \times Act_\Gamma$ such that no conflicts appear in SA.

$C = \{\langle s, a_\Gamma\rangle \in Strats | \exists \langle s', a'_\Gamma\rangle \in Strats \; s.t. \; s' \sim_\Gamma s \wedge a_\Gamma \neq a'_\Gamma\}$
if $C = \emptyset$ **then return** $\{Strats\}$
else
  $\langle s, a_\Gamma\rangle =$ pick one in C
  $E = \{\langle s', a'_\Gamma\rangle \in Strats | s' \sim_\Gamma s\}$
  $A = \{a_\Gamma \in Act_\Gamma | \exists \langle s, a_\Gamma\rangle \in E\}$
  $strats = \{\}$
  for $a_\Gamma \in A$ **do**
    $S = \{\langle s', a_\Gamma\rangle \in E | a'_\Gamma = a_\Gamma\}$
    $strats = strats \cup Split(S \cup (Strats \backslash E))$
  return $strats$

Theorem 2. *$Split(Strats)$ computes the set of all the largest subsets SA of $Strats \subseteq S \times Act_\Gamma$ such that no conflicts appear in SA.*

Remark 1. *A conflict appears in $SA \subseteq S \times Act_\Gamma$ if there exist two elements $\langle s, a_\Gamma\rangle$ and $\langle s', a'_\Gamma\rangle$ in SA such that $s' \sim_\Gamma s$ and $a_\Gamma \neq a'_\Gamma$, i.e. there is a conflict if SA proposes two different actions in two equivalent states.*

Proof sketch of Theorem 2. *Split* gets all the conflicting elements of *Strats*. If there are no such elements, then *Strats* is its own largest non-conflicting subset; otherwise, *Split* takes one conflicting equivalence class E and, for each of its largest non-conflicting subsets S—i.e. subsets of states using the same action—it calls *Split* on the rest of *Strats* augmented with the non-conflicting subset S.

We can prove the correctness of *Split* by induction over the number of conflicting equivalence classes of *Strats*. If *Strats* does not contain any conflicting equivalence classes, *Strats* is its own single largest subset in which no conflicts appear. Otherwise, let's assume that $Split(Starts \backslash E)$ with E a conflicting equivalence class of *Strats* returns the set of all the largest non-conflicting subsets of $Strats \backslash E$; then, by what has been explained above, *Split* returns the cartesian product between all the largest non-conflicting subsets of E and all the largest non-conflicting subsets of $Strats \backslash E$. Because these cannot be conflicting as they belong to different equivalence classes, we can conclude that *Split* returns the set of the largest non-conflicting subsets of *Strats*. □

The correctness of Algorithm 1 is then given by the following theorem.

Theorem 3. *$[\![\langle\Gamma\rangle\psi]\!]^F_{po}$ computes the set of states of M satisfying $\langle\Gamma\rangle\psi$, i.e.*

$$\forall s \in S, s \in [\![\langle\Gamma\rangle\psi]\!]^F_{po} \textit{ iff } s \models^F_{po} \langle\Gamma\rangle\psi.$$

Proof sketch. First, $Split(S \times Act_\Gamma)$ returns all the possible uniform strategies of the system, where a uniform strategy is represented by the only action allowed in each equivalence class of states—states equivalent in terms of the knowledge of Γ—, this action being the same for every state of the class.

Indeed, the set of the largest non-conflicting subsets of $S \times Act_\Gamma$ is the set of possible uniform strategies. A non-conflicting subset of $S \times Act_\Gamma$ provides at most one action for each equivalence class of states, otherwise it would not be non-conflicting; second, a largest non-conflicting subset of $S \times Act_\Gamma$ provides exactly one action for each equivalence class of states, otherwise there would be a larger subset giving one action for the missing equivalence classes and this subset would not be conflicting. Finally, a largest non-conflicting subset of $S \times Act_\Gamma$ is a uniform strategy because it is exactly the definition of a uniform strategy: giving one possible action for each equivalence class. This thus ends the proof that $Split$ returns the set of all possible uniform strategies.

Second, $winning = [\![\Gamma]\!]\psi]\!]^F_{fo}\psi|_{strat}$ returns the set of states for which the strategy *strat* is winning. Indeed, it uses $ATLK^F_{fo}$ model checking algorithm, restricted to actions in *strat*. It thus returns the set of states for which there is a (global) winning strategy in *strat*. As *strat* is, by construction, a uniform strategy, *winning* is the set of states for which there exists a uniform winning strategy—in fact, it is *strat* itself.

Finally, the set $\{s \in winning | \forall s' \sim_\Gamma s, s' \in winning\}$ is the set of states s for which *strat* is a winning strategy for all $s' \sim_\Gamma s$. *sat* thus accumulates all the states s for which there is a winning strategy for all states indistinguishable from s. As this is exactly the semantics of the property, i.e. *sat* is exactly the set of states of the system satisfying the property, the proof is done. □

Improving the basic algorithm The first improvement proposed for the basic algorithm is the pre-filtering of states to the ones satisfying the property under $ATLK^F_{fo}$; we can filter them because if a state s does not satisfy $\langle\Gamma\rangle\psi$ under $ATLK^F_{fo}$, s cannot satisfy $\langle\Gamma\rangle\psi$ under $ATLK^F_{po}$. The second one is the alternation between filtering and splitting the strategies. Both improvements are aimed at reducing the number of uniform strategies to consider. The improved algorithm is presented in Algorithm 3. Using this algorithm, we can compute $[\![\langle\Gamma\rangle\psi]\!]^F_{po}$ as $Improved[\![\langle\Gamma\rangle\psi]\!]^F_{po}|_{S\times Act_\Gamma}$. The intuition behind Algorithm 3 is to start by computing the set of states satisfying the property and the associated actions (line 3), then get all conflicts (line 3) and, if there are conflicts, choose one conflicting equivalence class of states and possible actions (lines 3 to 3) and for each possible action a_Γ, recursively call the algorithm with the strategies following a_Γ (lines 3 and 3)—i.e. split the class into uniform strategies for this class and recursively call the algorithm on each strategy.

More in detail, Algorithm 3 returns the set of states satisfying the property in *Strats*. So, to get the final result, we have to take all the states satisfying the property in $S \times Act_\Gamma$. Algorithm 3 uses the function $[\![.]\!]^{F,ac}_{fo}|_{strats}$. This function is a modification of the $[\![.]\!]^F_{fo}|_{strats}$ function where actions are linked to states. More precisely, every sub-call to $[\![.]\!]^F_{po}$ or $\overline{Fair_{[\Gamma]}}$ is enclosed by $StatesActions_\Gamma|_{strats}$ to get all enabled actions in these states, restricted to *strats*—$StatesActions_\Gamma|_{strats}(Z) = \{\langle s, a_\Gamma\rangle \in strats | s \in Z \wedge a_\Gamma \in enabled(s,\Gamma)\}$—, and $Pre_{\langle\Gamma\rangle}|_{strats}$ is replaced by $Pre^{ac}_{\langle\Gamma\rangle}|_{strats}(Z) = \{\langle s, a_\Gamma\rangle \in strats | a_\Gamma \in enabled(s,\Gamma) \wedge \forall a, a_\Gamma \sqsubseteq a \implies img(s,a) \subseteq Z\}$. For example, $[\![[\Gamma]G\phi]\!]^{F,ac}_{fo}|_{Strats} = \nu Z.(StatesActions_\Gamma|_{Strats}([\![\phi]\!]^F_{po} \cup \overline{Fair_{[\Gamma]}})) \cap Pre^{ac}_{\langle\Gamma\rangle}|_{Strats}(Z)$.

Algorithm 3: $Improved[\![\langle\Gamma\rangle\psi]\!]^F_{po}|_{Strats}$

Data: M a given (implicit) model, Γ a subset of agents of M, ψ an $ATLK^F_{po}$ path formula, $Strats \subseteq S \times Act_\Gamma$.
Result: The set of states of M satisfying $\langle\Gamma\rangle\psi$ in $Strats$.

1 $Z = [\![\langle\Gamma\rangle\psi]\!]^{F,ac}_{fo}|_{Strats}$
2 $C = \{\langle s, a_\Gamma\rangle \in Z | \exists \langle s', a'_\Gamma\rangle \in Z$ such that $s \sim_\Gamma s' \wedge a_\Gamma \neq a'_\Gamma\}$
if $C = \emptyset$ **then**
4 **return** $\{s \in S | \exists a_\Gamma \in Act_\Gamma$ s.t. $\forall s' \sim_\Gamma s, \langle s', a_\Gamma\rangle \in Z\}$
else
6 $\langle s, a_\Gamma\rangle =$ pick one in C
7 $E = \{\langle s', a'_\Gamma\rangle \in Z | s \sim_\Gamma s'\}$
8 $A = \{a_\Gamma \in Act_\Gamma | \exists \langle s, a_\Gamma\rangle \in E\}$
$sat = \{\}$
for $a_\Gamma \in A$ **do**
11 $strat = \{\langle s', a'_\Gamma\rangle \in E | a'_\Gamma = a_\Gamma\} \cup (Z \backslash E)$
12 $sat = sat \cup Improved[\![\langle\Gamma\rangle\psi]\!]^F_{po}|_{strat}$
return sat

Intuitively, $StatesActions_\Gamma|_{strats}(Z)$ returns all the states of Z with their enabled actions allowed by *strats* and $Pre^{ac}_{\langle\Gamma\rangle}|_{strats}(Z)$ returns the states that can enforce to reach Z in one step, and the actions that allow them to do so, restricted to actions in *strats*. $[\![\langle\Gamma\rangle\psi]\!]^{F,ac}_{fo}|_{strats}$ thus returns the states satisfying $\langle\Gamma\rangle\psi$ associated to the actions of *strats* that allow them to do so.

The correctness of Algorithm 3 is given by the following theorem.

Theorem 4. *$Improved[\![\langle\Gamma\rangle\psi]\!]^F_{po}|_{S\times Act_\Gamma}$ computes the set of states of M satisfying $\langle\Gamma\rangle\psi$, i.e.*

$$\forall s \in S, s \in Improved[\![\langle\Gamma\rangle\psi]\!]^F_{po}|_{S\times Act_\Gamma} \text{ iff } s \models^F_{po} \langle\Gamma\rangle\psi.$$

Proof sketch. First, $[\![\langle\Gamma\rangle\psi]\!]^{F,ac}_{fo}|_{Strats}$ returns the set of states s (and associated actions) such that there exists a global strategy in *Strats* allowing Γ to enforce the property in s. This means that if a state/action pair is not returned, Γ has no global strategy to enforce the property from the given state by using the action given in the pair. By extension, there is no uniform strategy to enforce the property neither. Thus, only state/action pairs returned by $[\![\langle\Gamma\rangle\psi]\!]^{F,ac}_{fo}|_{Strats}$ have to be considered when searching for a uniform strategy in *Strats*. This also means that $[\![\langle\Gamma\rangle\psi]\!]^{F,ac}_{fo}|_{Strats}$ filters *Strats* to winning global strategies; if the result is also a uniform strategy, all the states in the returned set have a uniform strategy to enforce the property.

Second, $Improved[\![\langle\Gamma\rangle\psi]\!]^F_{po}|_{Strats}$ returns the set of states satisfying the property in *Strats*. We can prove this by induction on the number of conflicting equivalence classes of *Strats*: this is true if there are no conflicting classes because Line 3 computes a winning uniform strategy—as discussed above—and Line 3 returns the set of states for which the strategy is winning for all indistinguishable states. This is also true in the inductive case because (1) filtering with $[\![\langle\Gamma\rangle\psi]\!]^{F,ac}_{fo}|_{Strats}$ doesn't lose potential state/action pairs and (2) the algorithm takes one conflicting class and tries all the possibilities for this class.

The final result thus is correct since it returns the set of states s for which there is a uniform strategy in $S \times Act_\Gamma$ that is winning for all states equivalent to s. □

Complexity considerations Model checking ATL with perfect recall and partial observability is an undecidable problem [14], while model checking ATL_{ir} is a Δ_2^P-complete problem [9]. $ATLK_{po}^F$ subsumes ATL_{ir} and its model checking problem is therefore Δ_2^P-hard. Algorithm 1 performs a call to $[[.]]_{fo}^F$ for each uniform strategy: $[[.]]_{fo}^F$ is in **P**, but in the worst case there could be exponentially many calls to this procedure, as there could be up to $\prod_{i\in\Gamma} |Act_i|^{|S_i|}$ uniform strategies to consider.

4 Conclusion

A number of studies in the past have investigated the problem of model checking strategies under partial observability and, separately, some work has provided algorithms for including fairness constraints on *actions* in the case of full observability. To the best of our knowledge, the issue of fairness constraints and partial observability have never been addressed together.

In this paper we presented $ATLK_{po}^F$, a logic combining partial observability and fairness constraints on *states* (which is the standard approach for temporal and epistemic logics), and we have provided a model checking algorithm.The proposed algorithm is similar to the one of Calta et al. [3]. They also split possible actions into uniform strategies, but they do not provide a way to deal with fairness constraints.

Finally, the structure of our algorithm is compatible with symbolic model checking using OBDDs, and we are working on its implementation in the model checker MCMAS [12], where fairness constraints are only supported for temporal and epistemic operators.

References

[1] Rajeev Alur, Thomas A. Henzinger & Orna Kupferman (2002): *Alternating-time temporal logic*. *J. ACM* 49(5), pp. 672–713, doi:10.1145/585265.585270.

[2] Christel Baier & Joost-Pieter Katoen (2008): *Principles of Model Checking*. The MIT Press.

[3] Jan Calta, Dmitry Shkatov & Holger Schlingloff (2010): *Finding Uniform Strategies for Multi-agent Systems*. In Jürgen Dix, João Leite, Guido Governatori & Wojtek Jamroga, editors: *Computational Logic in Multi-Agent Systems, Lecture Notes in Computer Science* 6245, Springer Berlin / Heidelberg, pp. 135–152, doi:10.1007/978-3-642-14977-1_12.

[4] E. M. Clarke, O. Grumberg & D. Peled (1999): *Model Checking*. MIT Press.

[5] Mehdi Dastani & Wojciech Jamroga (2010): *Reasoning about strategies of multi-agent programs*. In: *Proceedings of AAMAS 10*, pp. 997–1004.

[6] Ronald Fagin, Joseph Y. Halpern, Yoram Moses & Moshe Y. Vardi (1995): *Reasoning about Knowledge*. MIT Press, Cambridge.

[7] Wiebe van der Hoek & Michael Wooldridge (2003): *Cooperation, Knowledge, and Time: Alternating-time Temporal Epistemic Logic and its Applications*. *Studia Logica* 75, pp. 125–157, doi:10.1023/A:1026185103185.

[8] W. Jamroga & T. Ågotnes (2007): *Constructive knowledge: what agents can achieve under imperfect information*. *Journal of Applied Non-Classical Logics* 17(4), pp. 423–475, doi:10.3166/jancl.17.423-475.

[9] Wojciech Jamroga & Jürgen Dix (2006): *Model Checking Abilities under Incomplete Information Is Indeed Δ_2^P-complete*. In: *EUMAS'06*.

[10] Wojciech Jamroga & Wiebe van der Hoek (2004): *Agents that Know How to Play*. *Fundamenta Informaticae* Volume 63(2), pp. 185–219.

[11] Sascha Klüppelholz & Christel Baier (2008): *Alternating-Time Stream Logic for Multi-agent Systems*. In: *Coordination Models and Languages*, LNCS 5052, Springer, pp. 184–198, doi:10.1007/978-3-540-68265-3_12.

[12] A. Lomuscio, H. Qu & F. Raimondi (2009): *MCMAS: A Model Checker for the Verification of Multi-Agent Systems*. In: *Proceedings of CAV 2009, LNCS* 5643, Springer, pp. 682–688, doi:10.1007/978-3-642-02658-4_55.

[13] Alessio Lomuscio & Wojciech Penczek (2007): *Symbolic model checking for temporal-epistemic logics*. *SIGACT News* 38(3), pp. 77–99, doi:10.1145/1324215.1324231.

[14] Pierre-Yves Schobbens (2004): *Alternating-time logic with imperfect recall*. *Electronic Notes in Theoretical Computer Science* 85(2), pp. 82 – 93, doi:10.1016/S1571-0661(05)82604-0.

Reducing Validity in Epistemic ATL to Validity in Epistemic CTL

Dimitar P. Guelev
Institute of Mathematics and Informatics
Bulgarian Academy of Sciences
Sofia, Bulgaria
gelevdp@math.bas.bg

We propose a validity preserving translation from a subset of epistemic Alternating-time Temporal Logic (*ATL*) to epistemic Computation Tree Logic (*CTL*). The considered subset of epistemic *ATL* is known to have the finite model property and decidable model-checking. This entails the decidability of validity but the implied algorithm is unfeasible. Reducing the validity problem to that in a corresponding system of *CTL* makes the techniques for automated deduction for that logic available for the handling of the apparently more complex system of *ATL*.

Introduction

The strategic cooperation modalities of *alternating time temporal logic* (*ATL*, [AHK97, AHK02]) generalize the path quantifier $\forall$ of *computation tree logic* (*CTL*). Combinations of *ATL* with modal logics of knowledge [vdHW03, JvdH04] extend temporal logics of knowledge (cf. e.g [FHMV95]) in the way *ATL* extends *CTL*. Automated deduction for *CTL* and linear time epistemic temporal logics has been studied extensively [FDP01, BDF99, GS09a, GS09b]. There is much less work on the topic for *ATL*, and hardly any for its epistemic extensions. The decidability of validity in *ATL* with complete information was established in [GvD06] as a consequence of the *finite model property*, where the completeness of a Hilbert-style proof system was given too. Hilbert-style proof systems are known to be unsuitable for automating proof search. The situation was remedied by a tableau-based decision procedure developed in [GS09c]. Along with that, the same authors developed tableau systems for branching epistemic temporal logics in [GS09b]. Temporal resolution (cf. e.g. [FDP01]), which is well understood for linear time logics and their epistemic extensions, was considered for *ATL* in [Zha10], but only for the $\langle\langle . \rangle\rangle\circ$-subset, which is similar to *coalition logic* [Pau02] and enables only reasoning about a fixed number of steps. To our knowledge, no similar work has been done for systems epistemic *ATL*.

In this paper we continue the study [GDE11] of a system of *ATL* with the operator of distributed knowledge under the perfect recall assumption. In [GDE11] we established the finite model property for a subset, and a model-checking algorithm for the whole system. That algorithm assumed that coalition members can use the distributed knowledge of their coalitions to guide their actions. Dropping that assumption is known to render model-checking undecidable [DT11]. As expected, the validity-checking algorithm which these results imply is unfeasible.

In this paper we propose a validity preserving translation from another subset of that logic into epistemic *CTL*, with distributed knowledge and perfect recall again. As it becomes clear below, the need to consider a subset appears to be due to the lack of connectives in epistemic *CTL* to capture some interactions between knowledge and the progress of time. The translation makes no assumption on coordination within coalitions and there is no dependence on the availability of the past temporal modalities which

F. Mogavero, A. Murano, and M.Y. Vardi (Eds.):
1st Workshop on Strategic Reasoning 2013 (SR'13)
EPTCS 112, 2013, pp. 81–89, doi:10.4204/EPTCS.112.13

are featured in the axiomatization from [GDE11]. A semantic assumption that we keep is *finite branching*: only finitely many states should be reachable in one step from any state and models should have only finitely many initial states. Dropping that assumption would disable the fixpoint characterization of $(.\mathsf{U}.)$-objectives that we exploit, because of the requirement on strategies to be uniform. The translation enables the use of the known techniques for mechanized proof in the apparently simpler logic *CTL* and its epistemic extensions [BF99, GS09b]. Building on our previous work [GDE11], we work with the semantics of *ATL* on *interpreted systems* in their form adopted in [LR06].

1 Preliminaries

1.1 Propositional epistemic *ATL* with perfect recall (ATL^D_{iR})

The syntax of ATL^D_{iR} formulas can be given by the BNF

$$\varphi,\psi ::= \bot \mid p \mid (\varphi \Rightarrow \psi) \mid \mathsf{D}_\Gamma\varphi \mid \langle\langle\Gamma\rangle\rangle \circ \varphi \mid \langle\langle\Gamma\rangle\rangle(\varphi \mathsf{U} \psi) \mid [\![\Gamma]\!](\varphi \mathsf{U} \psi)$$

Here Γ ranges over finite sets of agents, and p ranges over propositional variables. In this paper we exclude the past temporal operators as their presence does not affect the working of our translation.

An *interpreted system* is defined with respect to some given finite set $\Sigma = \{1,\dots,N\}$ of *agents*, and a set of *propositional variables* (*atomic propositions*) *AP*. There is also an *environment* $e \notin \Sigma$; in the sequel we write Σ_e for $\Sigma \cup \{e\}$.

Definition 1 (interpreted systems) An *interpreted system* for Σ and *AP* is a tuple of the form

$$\langle\langle L_i : i \in \Sigma_e\rangle, I, \langle Act_i : i \in \Sigma_e\rangle, t, V\rangle \tag{1}$$

where:

L_i, $i \in \Sigma_e$, are nonempty sets of *local states*; L_Γ stands for $\prod_{i\in\Gamma} L_i$, $\Gamma \subseteq \Sigma_e$;

elements of L_{Σ_e} are called *global states*;

$I \subseteq L_{\Sigma_e}$ is a nonempty set of *initial global states*;

Act_i, $i \in \Sigma_e$, are nonempty sets of *actions*; Act_Γ stands for $\prod_{i\in\Gamma} Act_i$;

$t : L_{\Sigma_e} \times Act_{\Sigma_e} \to L_{\Sigma_e}$ is a *transition* function;

$V \subseteq L_{\Sigma_e} \times AP$ is a valuation of the atomic propositions.

For every $i \in \Sigma_e$ and $l', l'' \in L_{\Sigma_e}$ such that $l'_i = l''_i$ and $l'_e = l''_e$ the function t satisfies $(t(l',a))_i = (t(l'',a))_i$.

In the literature an interpreted system also includes a *protocol* to specify the actions which are permitted at every particular state. Protocols are not essential to our study here as the effect of a prohibited action can be set to that of some fixed permitted action (which is always supposed to exist) to produce an equivalent system in which all actions are always permitted. Our variant of interpreted systems is borrowed from [LR06] and has a technically convenient feature which is not present in other works [FHMV95, LQR]: every agent's next local state can be directlty affected by the local state of the environment through the transition function. Here follow the technical notions that are relevant to satisfaction of *ATL* formulas on interpreted systems.

Definition 2 (global runs and local runs) Given an $n \le \omega$, a *run of length n* is a sequence

$$r = l^0 a^0 l^1 0 a^1 \ldots \in L_{\Sigma_e}(Act_{\Sigma_e} L_{\Sigma_e})^n$$

such that $l^0 \in I$ and $l^{j+1} = t(l^j, a^j)$ for all $j < n$. A run is *infinite*, if $n = \omega$; otherwise it is *finite*. In either case we write $|r|$ for the *length n* of r. (Note that a run of length $n < \omega$ is indeed a sequence of $2n+1$ states and actions.)

Given r as above and $\Gamma \subseteq \Sigma$, we write r_Γ for the corresponding *local run*

$$l^0_\Gamma a^0_\Gamma \dots a^{n-1}_\Gamma l^n_\Gamma \in L_\Gamma (Act_\Gamma L_\Gamma)^n$$

of Γ in which $l^j_\Gamma = \langle l^j_i : i \in \Gamma \rangle$ and $a^j_\Gamma = \langle a^j_i : i \in \Gamma \rangle$.

We denote the set of all runs of some fixed length $n \leq \omega$, the set of all finite runs, and the set of all runs in IS by $R^n(IS)$, $R^{fin}(IS)$ and $R(IS)$, respectively.

Given $i, j < \omega$ and an r as above such that $i \leq j \leq |r|$, we write $r[i..j]$ for $l^i a^i \dots a^{j-1} l^j$.

Definition 3 (indiscernibility) Given $r', r'' \in R(IS)$ and $i \leq |r'|, |r''|$, we write $r' \sim_{\Gamma,i} r''$ if $r'[0..i]_\Gamma = r''[0..i]_\Gamma$. We write $r' \sim_\Gamma r''$ for the conjunction of $r' \sim_{\Gamma,|r'|} r''$ and $|r'| = |r''|$.

Sequences of the form $r_\emptyset$ consist of $\langle\rangle$s, and, consequently, $[r]_\emptyset$ is the class of all the runs of length $|r|$. Obviously $\sim_{\Gamma,n}$ and $\sim_\Gamma$ are equivalence relations on $R(IS)$.

Definition 4 We denote $\{r' \in R(IS) : r' \sim_\Gamma r\}$ by $[r]_\Gamma$.

Definition 5 (coalition strategies) A *strategy* for $\Gamma \subseteq \Sigma$ is a vector $s = \langle s_i : i \in \Gamma \rangle$ of functions s_i of type $\{r_i : r \in R^{fin}(IS)\} \to Act_i$. We write $S(\Gamma, IS)$ for the set of all the strategies for Γ in the considered interpreted system IS. Given $s \in S(\Gamma, IS)$ and $r \in R^{fin}(IS)$, we write $\mathrm{out}(r,s)$ for the set

$$\{r' = l^0 a^0 \dots a^{n-1} l^n \dots \in R^\omega(IS) : r'[0..|r|] = r, a^j_i = s_i(r[0..j]_\Gamma) \text{ for all } i \in \Gamma \text{ and } j \geq |r|\}.$$

of the *outcomes* of r when Γ sticks to s from step $|r|$ on. Given an $X \subseteq R^{fin}(IS)$, $\mathrm{out}(X,s)$ is $\bigcup\limits_{r \in X} \mathrm{out}(r,s)$.

Strategies, as defined above, are determined by the local views of the considered coalition members and are therefore *uniform*.

Definition 6 (modelling relation of ATL^D_{iR}) The relation $IS, r \models \varphi$ is defined for $r \in R^{fin}(IS)$ and formulas φ by the clauses:

$IS, r \not\models \bot$;		
$IS, l^0 a^0 \dots a^{n-1} l^n \models p$	iff	$V(l^n, p)$ for atomic propositions p;
$IS, r \models \varphi \Rightarrow \psi$	iff	either $IS, r \not\models \varphi$ or $IS, r \models \psi$;
$IS, r \models \mathsf{D}_\Gamma \varphi$	iff	$IS, r' \models \varphi$ for all $r' \in [r]_\Gamma$;
$IS, r \models \langle\langle\Gamma\rangle\rangle \circ \varphi$	iff	there exists an $s \in S(\Gamma, IS)$ such that $IS, r'[0..\|r\|+1] \models \varphi$ for all $r' \in \mathrm{out}([r]_\Gamma, s)$;
$IS, r \models \langle\langle\Gamma\rangle\rangle(\varphi \mathsf{U} \psi)$	iff	there exists an $s \in S(\Gamma, IS)$ s. t. for every $r' \in \mathrm{out}([r]_\Gamma, s)$ there exists a $k < \omega$ s. t. $IS, r'[0..\|r\|+i] \models \varphi$ for all $i < k$ and $IS, r'[0..\|r\|+k] \models \psi$;
$IS, r \models [\![\Gamma]\!](\varphi \mathsf{U} \psi)$	iff	for every $s \in S(\Gamma, IS)$ there exist an $r' \in \mathrm{out}([r]_\Gamma, s)$ and a $k < \omega$ s. t. $IS, r'[0..\|r\|+i] \models \varphi$ for all $i < k$ and $IS, r'[0..\|r\|+k] \models \psi$.

Validity of formulas in entire interpreted systems and on the class of all interpreted systems, that is, in the logic ATL^D_{iR}, is defined as satisfaction at all 0-length runs in the considered interpreted system, and at all the 0-length runs in all the systems in the considered class, respectively.

In this paper we assume that each coalition member uses only its own observation power in following a coalition strategy. Allowing coalition members to share their observations gives rise to a more general form of strategy, which are functions of type $\{r_\Gamma : r \in R^{fin}(IS)\} \to Act_\Gamma$, and which was assumed by the model-checkig algorithm proposed in [GDE11].

Abbreviations

$\top$, $\neg$, $\vee$, $\wedge$ and $\Leftrightarrow$ have their usual meanings. To keep the use of (and) down, we assume that unary connectives bind the strongest, the binary modalities $\langle\langle\Gamma\rangle\rangle(.\mathsf{U}.)$ and $[\![\Gamma]\!](.\mathsf{U}.)$, and the derived ones below, bind the weakest, and their parentheses are never omitted, and the binary boolean connectives come in the middle, in decreasing order of their binding power as follows: $\wedge$, $\vee$, $\Rightarrow$ and $\Leftrightarrow$. We enumerate coalitions without the { and }. E.g., the shortest way to write $\langle\langle\{1\}\rangle\rangle(((p \Rightarrow q) \wedge \mathsf{P}_{\{1\}} r)\mathsf{U}\mathsf{D}_{\{2,3\}}(r \vee q)))$ is $\langle\langle 1\rangle\rangle((p \Rightarrow q) \wedge \mathsf{P}_1 r \mathsf{U}\mathsf{D}_{2,3}(r \vee q))$. We write P for the dual of D:

$$\mathsf{P}_\Gamma\varphi \rightleftharpoons \neg\mathsf{D}_\Gamma\neg\varphi.$$

The rest of the combinations of the cooperation modality and future temporal connectives are defined by the clauses

$$\begin{array}{lll} \langle\langle\Gamma\rangle\rangle\Diamond\varphi \rightleftharpoons \langle\langle\Gamma\rangle\rangle(\top\mathsf{U}\varphi) & \langle\langle\Gamma\rangle\rangle\Box\varphi \rightleftharpoons \neg[\![\Gamma]\!]\Diamond\neg\varphi & \langle\langle\Gamma\rangle\rangle(\varphi\mathsf{W}\psi) \rightleftharpoons \neg[\![\Gamma]\!](\neg\psi\mathsf{U}\neg\psi\wedge\neg\varphi) \\ [\![\Gamma]\!]\Diamond\varphi \rightleftharpoons [\![\Gamma]\!](\top\mathsf{U}\varphi) & [\![\Gamma]\!]\Box\varphi \rightleftharpoons \neg\langle\langle\Gamma\rangle\rangle\Diamond\neg\varphi & [\![\Gamma]\!](\varphi\mathsf{W}\psi) \rightleftharpoons \neg\langle\langle\Gamma\rangle\rangle(\neg\psi\mathsf{U}\neg\psi\wedge\neg\varphi) \end{array}$$

1.2 ATL^D_{iR} with epistemic objectives only

In [GDE11] we axiomatized a subset of ATL^D_{iR} with past in which $\langle\langle.\rangle\rangle(.\mathsf{U}.)$ was allowed only in the derived construct $\langle\langle\Gamma\rangle\rangle\Diamond\mathsf{D}_\Gamma\varphi$, and $[\![.]\!](.\mathsf{U}.)$ was allowed only in the derived construct $\langle\langle\Gamma\rangle\rangle\Box\varphi$. Because of the validity of the equivalences

$$\langle\langle\Gamma\rangle\rangle\circ\varphi \Leftrightarrow \langle\langle\Gamma\rangle\rangle\circ\mathsf{D}_\Gamma\varphi \text{ and } \langle\langle\Gamma\rangle\rangle\Box\varphi \Leftrightarrow \langle\langle\Gamma\rangle\rangle\Box\mathsf{D}_\Gamma\varphi,$$

that entailed that all the objectives allowed in that subset were epistemic. We argued that, under some assumptions, any $\langle\langle.\rangle\rangle(.\mathsf{U}.)$ formula could be transformed into an equivalent one of the form $\langle\langle\Gamma\rangle\rangle\Diamond\mathsf{D}_\Gamma\varphi$ thus asserting the significance of the considered subset. Both the axiomatization and the reduction to epistemic goals relied on the presence of the past operators. In this paper we consider another subset of ATL^D_{iR}. Its formulas have the syntax

$$\varphi,\psi ::= \bot \mid p \mid (\varphi \Rightarrow \psi) \mid \mathsf{D}_\Gamma\varphi \mid \langle\langle\Gamma\rangle\rangle\circ\varphi \mid \langle\langle\Gamma\rangle\rangle(\mathsf{D}_\Gamma\varphi\mathsf{U}\mathsf{D}_\Gamma\psi) \tag{2}$$

Unlike the subset from [GDE11], here we allow formulas of the form $\langle\langle\Gamma\rangle\rangle(\mathsf{D}_\Gamma\varphi\mathsf{U}\mathsf{D}_\Gamma\psi)$. However, we exclude even the special case $\langle\langle\Gamma\rangle\rangle\Box\varphi$ of the use of $[\![\Gamma]\!](\mathsf{P}_\Gamma\varphi\mathsf{U}\mathsf{P}_\Gamma\psi)$. The reasons are discussed in the end of Section 2.

1.3 *CTL* with distributed knowledge

This is the target logic of our translation. Its formulas have the syntax

$$\varphi,\psi ::= \bot \mid p \mid (\varphi \Rightarrow \psi) \mid \mathsf{D}_\Gamma\varphi \mid \exists\circ\varphi \mid \exists(\varphi\mathsf{U}\psi) \mid \forall(\varphi\mathsf{U}\psi)$$

where Γ ranges over finite sets of agents as above. The clauses for the semantics of the connectives in common with ATL^D_{iR} are as in ATL^D_{iR}; the clauses about formulas built using $\exists$ and $\forall$ are as follows:

$IS,r \models \exists\circ\varphi$ iff there exists an $r' \in R^{|r|+1}(IS)$ such that $r = r'[0..|r|]$ and $IS,r' \models \varphi$;

$IS,r \models \exists(\varphi\mathsf{U}\psi)$ iff there exists an $r' \in R^\omega(IS)$ such that $r = r'[0..|r|]$ and a $k < \omega$ such that $IS,r'[0..|r|+i] \models \varphi$ for all $i < k$ and $IS,r'[0..|r|+k] \models \psi$;

$IS,r \models \forall(\varphi\mathsf{U}\psi)$ iff for every $r' \in R^\omega(IS)$ such that $r = r'[0..|r|]$ there exists a $k < \omega$ such that $IS,r'[0..|r|+i] \models \varphi$ for all $i < k$ and $IS,r'[0..|r|+k] \models \psi$.

Note that the the occurrences of $\mathsf{D}_\emptyset$ is vital for the validity of the equivalences

$$\mathsf{P}_\emptyset\exists\circ\varphi \Leftrightarrow [\![\emptyset]\!]\circ\varphi,\ \ \mathsf{P}_\emptyset\exists(\varphi\mathsf{U}\psi) \Leftrightarrow [\![\emptyset]\!](\varphi\mathsf{U}\psi) \text{ and } \mathsf{D}_\emptyset\forall(\varphi\mathsf{U}\psi) \Leftrightarrow \langle\langle\emptyset\rangle\rangle(\varphi\mathsf{U}\psi).$$

in the combined language of ATL^D_{iR} and *CTL* because of the requirement on strategies to be uniform; e.g., $\langle\langle\emptyset\rangle\rangle(\varphi\mathsf{U}\psi)$ means that $(\varphi\mathsf{U}\psi)$ holds along all the extensions of all the runs *which are indiscernible from the reference run to the empty coalition.* Therefore here $\langle\langle\emptyset\rangle\rangle$ does not subsume $\forall$ in the straightforward way known about the case *ATL* of complete information.

The combination $\forall\circ$ and the combinations of $\exists$ and $\forall$ with the derived temporal connectives $(\mathsf{W}.)$, $\Diamond$ and $\Box$ are defined in the usual way.

2 A validity preserving translation into $CTL+\mathsf{D}$ with perfect recall

Our translation captures the subset of *ATL* which is given by the BNF

$$\varphi,\psi ::= \bot \mid p \mid (\varphi\Rightarrow\psi) \mid \ominus\varphi \mid (\varphi\mathsf{S}\psi) \mid \mathsf{D}_\Gamma\varphi \mid \langle\langle\Gamma\rangle\rangle\circ\varphi \mid \langle\langle\Gamma\rangle\rangle(\mathsf{D}_\Gamma\varphi\mathsf{U}\mathsf{D}_\Gamma\psi)$$

We explain how to eliminate occurrences of $\langle\langle.\rangle\rangle$ in formulas of the form $\langle\langle\Gamma\rangle\rangle(\mathsf{D}_\Gamma\varphi\mathsf{U}\mathsf{D}_\Gamma\psi)$ first. In the sequel we write $[\alpha/p]\beta$ for the substitution of the occurrences of atomic proposition p in β by α.

Proposition 7 *Assuming that p and q are fresh atomic propositions, the satisfiability of* $[\langle\langle\Gamma\rangle\rangle(\mathsf{D}_\Gamma\varphi\mathsf{U}\mathsf{D}_\Gamma\psi)/p]\chi$ *(at a* 0*-length run) is equivalent to the satisfiability of*

$$\begin{array}{lll}\chi & \wedge & \mathsf{D}_\emptyset\forall\Box(p\vee q\Rightarrow\mathsf{D}_\Gamma\psi\vee(\mathsf{D}_\Gamma\varphi\wedge\langle\langle\Gamma\rangle\rangle\circ q))\\ & \wedge & \mathsf{D}_\emptyset\forall\Box(p\Leftrightarrow\mathsf{D}_\Gamma\psi\vee(\mathsf{D}_\Gamma\varphi\wedge\langle\langle\Gamma\rangle\rangle\circ p))\\ & \wedge & \mathsf{D}_\emptyset\forall\Box(p\Rightarrow\mathsf{D}_\Gamma\psi\vee(\mathsf{D}_\Gamma\varphi\wedge\forall\circ\forall(q\Rightarrow\mathsf{D}_\Gamma\varphi\mathsf{U}q\Rightarrow\mathsf{D}_\Gamma\psi))).\end{array}\tag{3}$$

Next we explain how to eliminate occurrences of the "basic" *ATL* construct $\langle\langle\Gamma\rangle\rangle\circ\varphi$. Let *IS* stand for some arbitrary interpreted system (1) with finite branching, with $\Sigma=\{1,\ldots,N\}$ as its set of agents, *AP* as its vocabulary. We adapt the following simple observation, which works in case Act_i, $i\in\Sigma$ are fixed. Readers who are familiar with the original semantics of *ATL* on *alternating transition systems* (*ATS*) from [AHK97] will recognize the similarity of our technique with the transformation of *concurrent game structures* into equivalent *ATS* from [GJ04]. Assuming that Act_i, $i\in\Sigma_e$, are pairwise disjoint, and disjoint with *AP*, we consider the vocabulary $AP^{Act}=AP\cup\bigcup\limits_{i\in\Sigma_e}Act_i$.

Definition 8 Given *IS* and $*\notin\bigcup\limits_{i\in\Sigma_e}Act_i$, we define the interpreted system

$$IS^{Act}=\langle\langle L_i^{Act}:i\in\Sigma_e\rangle,I^{Act},\langle Act_i:i\in\Sigma_e\rangle,t^{Act},V^{Act}\rangle$$

by putting:

$$\begin{array}{lcl}L_i^{Act} & = & L_i\times(Act_i\cup\{*\}),\ i\in\Sigma_e;\\ I^{Act} & = & \{\langle\langle l_i,*\rangle:i\in\Sigma_e\rangle:l\in I\};\\ t^{Act}(\langle\langle l_i,a_i\rangle:i\in\Sigma_e\rangle,b) & = & \langle\langle(t(l,b))_i,b_i\rangle:i\in\Sigma_e\rangle;\\ V^{Act}(\langle\langle l_i,a_i\rangle:i\in\Sigma_e\rangle,p) & \leftrightarrow & V(\langle l_i,:i\in\Sigma_e\rangle,p)\text{ for }p\in AP;\\ V^{Act}(\langle\langle l_i,a_i\rangle:i\in\Sigma_e\rangle,b) & \leftrightarrow & b=a_j\text{ for }b\in Act_j,\ j\in\Sigma_e.\end{array}$$

In short, an IS^{Act} state is an *IS* state augmented with a record of the actions which lead to it, the dummy symbol $*$ being used in initial states. Let $R\subseteq L^{Act}_{\Sigma_e}\times L^{Act}_{\Sigma_e}$ and $R(\langle\langle l_i,a_i\rangle:i\in\Sigma_e\rangle,\langle\langle v_i,b_i\rangle:i\in\Sigma_e\rangle)$ iff $v=t^{Act}(l,b)$. Then $IS^{Act},r\models\exists\circ\varphi$ iff $IS^{Act},r\,a\,l'\models\varphi$ for some $l'\in R(l)$ and the only $a\in Act_{\Sigma_e}$ such that $r\,a\,l'\in R^{fin}(IS^{Act})$. The key observation in our approach is that

$$IS,r\models\langle\langle i_1,\ldots,i_k\rangle\rangle\circ\varphi\text{ iff }IS^{Act},r^{Act}\models\bigvee_{a_{i_1}\in Act_{i_1}}\ldots\bigvee_{a_{i_k}\in Act_{i_k}}\mathsf{D}_{\{i_1,\ldots,i_k\}}\forall\circ\left(\bigwedge_{j=1}^{k}a_{i_j}\Rightarrow\varphi\right)\tag{4}$$

For this observation to work without refering to the actions in the particular interpreted system, given an arbitrary *IS*, we enrich it with dedicated actions which are linked to the objectives occurring in the considered formula. We define the transition function on these actions so that if a particular $\circ\varphi$-objective can be achieved at finite run r at all, then it can be achieved by taking the corresponding dedicated actions at the last state of r. This can be achieved in forest-like systems where runs can be determined from their final states. Similarly, we introduce express actions for the environment that enable it to foil objectives at states at which they objectives cannot be achieved by the respective coalitions using any strategy based on the original actions. (Giving the environment such powers does not affect the satisfaction of formulas as it never participates in coalitions.) The sets Act_i, $i \in \Sigma_e$ of atomic propositions by which we model actions satisfy the formula

$$\mathsf{A}(Act_1,\ldots,Act_N,Act_e) \rightleftharpoons \bigwedge_{a_1\in Act_1} \ldots \bigwedge_{a_N\in Act_N} \bigwedge_{a_e\in Act_e} \exists\circ \bigwedge_{i\in\Sigma_e} a_i,$$

which states that any vector of actions from Act_{Σ_e} produces a transition. Consider an ATL^D_{iR} formula of the form below with no occurrences of $(.\mathsf{U}.)$-objectives:

$$\chi \wedge \mathsf{D}_\emptyset\forall\Box\mathsf{A}(Act_1,\ldots,Act_N,Act_e) \tag{5}$$

Here $Act_1,\ldots,Act_N,Act_e$ consist of the atomic propositions which have been introduced to eliminate $\langle\langle\Gamma\rangle\rangle\circ\varphi$-subformulas so far. For the original χ we assume $Act_i = \{\mathsf{nop}_i\}$, $i \in \Sigma_e$, where nop_i have no specified effect. We remove the occurrences of $\langle\langle\Gamma\rangle\rangle\circ\varphi$-subformulas in χ working bottom-up as follows.

Proposition 9 *Let* $\mathsf{a}_{\Gamma,i,\varphi}$, $i \in \Gamma\cup\{e\}$, *be fresh atomic propositions,* $Act'_i = Act_i \cup \{\mathsf{a}_{\Gamma,i,\varphi}\}$ *for* $i \in \Gamma\cup\{e\}$ *and* $Act'_i = Act_i$ *for* $i \in \Sigma\setminus\Gamma$. *Then the satisfiability of*

$$[\langle\langle\Gamma\rangle\rangle\circ\varphi/p]\chi \wedge \mathsf{D}_\emptyset\forall\Box\mathsf{A}(Act_1,\ldots,Act_N,Act_e) \tag{6}$$

entails the satisfiability of the formula

$$\begin{array}{l}\left[\mathsf{D}_\Gamma\forall\circ\left(\bigwedge_{i\in\Gamma}\mathsf{a}_{\Gamma,i,\varphi}\Rightarrow\varphi\right)/p\right]\chi\wedge\\ \mathsf{D}_\emptyset\forall\Box\left(\mathsf{D}_\Gamma\forall\circ\left(\bigwedge_{i\in\Gamma}\mathsf{a}_{\Gamma,i,\varphi}\Rightarrow\varphi\right)\vee\mathsf{P}_\Gamma\forall\circ(\mathsf{a}_{\Gamma,e,\varphi}\Rightarrow\neg\varphi)\right)\wedge\\ \mathsf{D}_\emptyset\forall\Box\mathsf{A}(Act'_1,\ldots,Act'_N,Act'_e).\end{array} \tag{7}$$

The above proposition shows how to eliminate one by one all the occurrences of the cooperation modalities in an any given ATL^D_{iR} formula χ with the cooperation modalities appearing only in subformulas of the form $\langle\langle\Gamma\rangle\rangle\circ\varphi$ and obtain a $CTL+\mathsf{D}$ formula χ' such that if χ is satisfiable, then so is χ'. Now consider a purely-$CTL+\mathsf{D}$ formula of the form (5). The satisfaction of (5) requires just a transition relation for the passage of time to define as it contains no $\langle\langle\Gamma\rangle\rangle$s and hence no reference to actions. That is, we assume a satisfying model of the form

$$IS^- = \langle\langle L_i : i\in\Sigma_e\rangle, I, -, V\rangle \tag{8}$$

where L_i, $i \in \Sigma_e$, I and V are as in interpreted systems, and $-$ is a serial binary relation on the set of the global states L_{Σ_e} that represents the passage of time. We define the remaining interpreted system components as follows. We choose the set of actions of each agent i, including the environment, to be the corresponding set of atomic propositions Act_i from (5). For any $a \in Act_{\Sigma_e}$ and any $l \in L_{\Sigma_e}$ we choose $t(l,a)$ to be an arbitrary member of $-(l)\cap\bigcap_{i\in\Sigma_e}\{l'\in L_{\Sigma_e} : V(l',a_i)\}$. The nonemptiness of the latter set is guaranteed by the validity of $\mathsf{A}(Act_1,\ldots,Act_N,Act_e)$ in IS^-, which states that every state has a successor

satisfying the conjunction $\bigwedge_{i\in\Sigma_e} a_i$ for any given vector of actions $a \in Act_{\Sigma_e}$. Let IS stand for the system obtained by this definition of Act_i, $i \in \Sigma_e$, and t. It remains to show that

$$IS, r \models \mathsf{D}_\Gamma \forall \circ \left(\bigwedge_{i\in\Gamma} \mathsf{a}_{\Gamma,i,\varphi} \Rightarrow \varphi \right) \tag{9}$$

is equivalent to $IS, r \models \langle\langle\Gamma\rangle\rangle \circ \varphi$ for any subformula $\langle\langle\Gamma\rangle\rangle \circ \varphi$ eliminated in the process of obtaining (5). For the forward direction, establishing that the actions $a_{\Gamma,i,\varphi}$, $i \in \Gamma$ provides Γ with a strategy to achieve φ in one step is easily done by a direct check. For the converse direction, if (9) is false, then the validity of the second conjunctive member of (7) entails that Γ cannot rule out the possibility that the environment can enforce $\neg\varphi$ in one step by choosing its corresponding action $\mathsf{a}_{\Gamma,e,\varphi}$.

Formulas of the form $[\![\Gamma]\!](\mathsf{P}_\Gamma\varphi\mathsf{U}\mathsf{P}_\Gamma\psi)$

We first note that no restriction on formulas of the respective more general form $[\![\Gamma]\!](\varphi\mathsf{U}\psi)$ is necessary in the case of complete information.

Proposition 10 (eliminating $[\![\Gamma]\!](\varphi\mathsf{U}\psi)$ in *ATL* with complete information) *Let p and q be some fresh atomic propositions. The satisfiability of*

$$[[\![\Gamma]\!](\varphi\mathsf{U}\psi)/p]\chi$$

in ATL with complete information is equivalent to the satisfiability of

$$\begin{array}{lll} \chi & \wedge & \forall\Box(p \vee q \Rightarrow \psi \vee (\varphi \wedge [\![\Gamma]\!] \circ q)) \\ & \wedge & \forall\Box(p \Leftrightarrow \psi \vee (\varphi \wedge [\![\Gamma]\!] \circ p)) \\ & \wedge & \forall\Box(p \Rightarrow \psi \vee (\varphi \wedge \forall \circ \forall(q \Rightarrow \varphi\mathsf{U}q \Rightarrow \psi))). \end{array} \tag{10}$$

In the incomplete information case our approach suggests replacing $[[\![\Gamma]\!](\mathsf{P}_\Gamma\varphi\mathsf{U}\mathsf{P}_\Gamma\psi)/p]\chi$ by

$$\begin{array}{lll} \chi & \wedge & \mathsf{D}_\emptyset\forall\Box(p \vee q \Rightarrow \mathsf{P}_\Gamma\psi \vee (\mathsf{P}_\Gamma\varphi \wedge [\![\Gamma]\!] \circ q)) \\ & \wedge & \mathsf{D}_\emptyset\forall\Box(p \Leftrightarrow \mathsf{P}_\Gamma\psi \vee (\mathsf{P}_\Gamma\varphi \wedge [\![\Gamma]\!] \circ p)) \\ & \wedge & \mathsf{D}_\emptyset\forall\Box(p \Rightarrow \mathsf{P}_\Gamma\psi \vee (\mathsf{P}_\Gamma\varphi \wedge \ldots)). \end{array}$$

where, in a forest-like system IS, q is supposed to mark states which are reached from runs r in which Γ cannot achieve $(\mathsf{P}_\Gamma\varphi\mathsf{U}\mathsf{P}_\Gamma\psi)$ when Γ's actions a are complemented on behalf of the non-members of Γ by some actions b_{a_1,r_1} that foil the objective, and $\ldots$ is supposed to express that any sequence of vectors of actions $a_1, a_2, \ldots \in Act_\Gamma$ when complemented by the corresponding b_{a_1,r_1}, $b_{a_2,r_2}, \ldots$ can generate a sequence $r_1, r_2, \ldots$ of finite runs, starting with the reference one, each of them being Γ-indiscernible from the extension of the previous one, by the outcome of the respective $a_k \cdot b_{a_k,r_k}$, such that there exists a $k < \omega$ with $IS, r_j \models q \wedge \mathsf{D}_\Gamma\varphi$, $j = 1, \ldots, k-1$, and $IS, r_k \models \neg q \vee \mathsf{D}_\Gamma\psi$. The fixpoint construct that would best serve expressing this condition can be written as $\mu X.\alpha \vee (\beta \wedge \mathsf{P}_\Gamma\forall \circ X)$ in the modal μ-calculus (cf. e.g. [BS06]). Finding a substitute for it in $CTL + \mathsf{D}$ appears problematic.

Concluding remarks

Our approach is inspired by temporal resolution [FDP01], which has been extended to epistemic *LTL* [DFW98] and to (non-epistemic) *CTL* and *CTL** [BF99, BDF99], the latter system being the closest to our target system $CTL + \mathsf{D}$. Following the example of these works, a resolution system for $CTL + \mathsf{D}$ could be proved complete by showing how to reproduce in it any proof in some complete, e.g., Hilbert

style proof system. A complete axiomatization for epistemic CTL^* with perfect recall can be found in [vdMK03], but the completeness was demonstrated with respect to the so-called *bundle* semantics, where a model may consist of some set of runs that need not be all the runs generated by a transition system. and the form of collective knowledge considered in [vdMK03] is *common knowledge*, whereas we have distributed knowledge. The setting for the complexity results from [HV86] is similar. The tableau-based decision procedure for epistemic *CTL* with both common and distributed knowledge from [GS09b] does not cover the case of perfect recall. To the best of our knowledge no decision procedure of feasible complexity such as the resolution- and tableau-based ones that are available for so many closely related systems from the above works has been developed yet for validity in $CTL + \mathsf{D}$ with perfect recall.

Acknowledgement

The research in this paper was partially supported through Bulgarian National Science Fund Grant DID02/32/2009.

References

[AHK97] Rajeev Alur, Tom Henzinger, and Orna Kupferman. Alternating-time Temporal Logic. In *Proceedings of FCS'97*, pages 100–109, 1997, doi:10.1007/3-540-49213-5_2.

[AHK02] Rajeev Alur, Tom Henzinger, and Orna Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):1–42, 2002, doi:10.1145/585265.585270.

[BDF99] Alexander Bolotov, Clare Dixon, and Michael Fisher. Clausal Resolution for CTL*. In *MFCS*, volume 1672 of *LNCS*, pages 137–148. Springer, 1999, doi:10.1007/3-540-48340-3_13.

[BF99] Alexander Bolotov and Michael Fisher. A clausal resolution method for CTL branching-time temporal logic. *J. Exp. Theor. Artif. Intell.*, 11(1):77–93, 1999, doi:10.1080/095281399146625.

[BS06] Julian Bradfield and Colin Stirling. Modal μ-Calculi. In *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*, pages 721–756. Elsevier, 2006.

[DFW98] Clare Dixon, Michael Fisher, and Michael Wooldridge. Resolution for Temporal Logics of Knowledge. *Journal of Logic and Computation*, 8(3):345–372, 1998, doi:10.1093/logcom/8.3.345.

[DT11] Catalin Dima and Ferucio Laurentiu Tiplea. Model-checking ATL under Imperfect Information and Perfect Recall Semantics is Undecidable. *CoRR*, abs/1102.4225, 2011.

[FDP01] Michael Fisher, Clare Dixon, and Martin Peim. Clausal Temporal Resolution. *ACM Trans. Comput. Log.*, 2(1):12–56, 2001, doi:10.1145/371282.371311.

[FHMV95] Ronald Fagin, Joseph Halpern, Yoram Moses, and Moshe Vardi. *Reasoning about Knowledge*. MIT Press, 1995.

[GDE11] Dimitar P. Guelev, Catalin Dima, and Constantin Enea. An Alternating-time Temporal Logic with Knowledge, Perfect Recall and Past: Axiomatisation and Model-checking. *Journal of Applied Non-Classical Logics*, 21(1):93–131, 2011, , doi:10.3166/jancl.21.93-131.

[GJ04] Valentin Goranko and Wojtek Jamroga. Comparing Semantics for Logics of Multi-agent Systems. *Synthese*, 139(2):241–280, 2004.

[GS09a] Valentin Goranko and Dmitry Shkatov. Tableau-based decision procedure for full coalitional multiagent temporal-epistemic logic of linear time. In *AAMAS (2)*, pages 969–976. IFAAMAS, 2009, doi:10.1145/1558109.1558147.

[GS09b] Valentin Goranko and Dmitry Shkatov. Tableau-based decision procedure for the full coalitional multiagent logic of branching time. In *MALLOW*, volume 494 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.

[GS09c] Valentin Goranko and Dmitry Shkatov. Tableau-based decision procedures for logics of strategic ability in multiagent systems. *ACM Trans. Comput. Log.*, 11(1), 2009, doi:10.1145/1614431.1614434.

[GvD06] Valentin Goranko and Govert van Drimmelen. Decidability and Complete Axiomatization of the Alternating-time Temporal Logic. *Theoretical Computer Science*, 353(1-3):93–117, 2006, doi:10.1016/j.tcs.2005.07.043.

[HV86] Joseph Y. Halpern and Moshe Y. Vardi. The complexity of reasoning about knowledge and time: Extended abstract. In Juris Hartmanis, editor, *STOC*, pages 304–315. ACM, 1986, doi:10.1145/12130.12161.

[JvdH04] Wojciech Jamroga and Wiebe van der Hoek. Agents That Know How to Play. *Fundamenta Informaticae*, 63(2-3):185–219, 2004.

[LQR] Alessio Lomuscio, Hongyang Qu, and Franco Raimondi. MCMAS: a Model Checker for Multi-Agents Systems. URL: `http://www-lai.doc.ic.ac.uk/mcmas/`. Accessed in January, 2010.

[LR06] Alessio Lomuscio and Franco Raimondi. Model checking knowledge, strategies, and games in multi-agent systems. In *Proceedings of AAMAS'06*, pages 161–168. ACM Press, 2006, doi:10.1145/1160633.1160660.

[Pau02] Marc Pauly. A Modal Logic for Coalitional Power in Games. *Journal of Logic and Computation*, 12(1):149–166, 2002, doi:10.1093/logcom/12.1.149.

[vdHW03] Wiebe van der Hoek and Michael Wooldridge. Cooperation, Knowledge and Time: Alternating-time Temporal Epistemic Logic and Its Applications. *Studia Logica*, 75:125–157, 2003, doi:10.1023/A:1026185103185.

[vdMK03] Ron van der Meyden and Ka-shu Wong. Complete Axiomatizations for Reasoning about Knowledge and Branching Time. *Studia Logica*, 75(1):93–123, 2003, doi:10.1023/A:1026181001368.

[Zha10] Lan Zhang. *Clausal reasoning for branching-time logics*. Ph.d. thesis, University of Liverpool, 2010. Accessed in December 2012 from `http://research-archive.liv.ac.uk/3373/4/ZhangLan_Dec2010_3373.pdf`.

Towards an Updatable Strategy Logic

Christophe Chareton Julien Brunel David Chemouil
Onera – The French Aerospace Lab
F-31055 Toulouse, France
firstname.lastname@onera.fr

This article is about temporal multi-agent logics. Several of these formalisms have been already presented (ATL-ATL*, ATL$_{sc}$, SL). They enable to express the capabilities of agents in a system to ensure the satisfaction of temporal properties. Particularly, SL and ATL$_{sc}$ enable several agents to interact in a context mixing the different strategies they play in a semantical game. We generalize this possibility by proposing a new formalism, Updating Strategy Logic (USL). In USL, an agent can also refine its own strategy. The gain in expressive power rises the notion of *sustainable capabilities* for agents.

USL is built from SL. It mainly brings to SL the two following modifications: semantically, the successor of a given state is not uniquely determined by the data of one choice from each agent. Syntactically, we introduce in the language an operator, called an *unbinder*, which explicitly deletes the binding of a strategy to an agent. We show that USL is strictly more expressive than SL.

1 Introduction

Multi-agent logics are receiving growing interest in contemporary research. Since the seminal work of Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman [2], one major and recent direction (ATL with Strategy Context [3, 6, 7], Strategy Logic (presented first in [5] and then extended in [8, 10]) aims at contextualizing the statements of capabilities of agents.

Basically, multi-agent logics enable assertions about the capability of agents to ensure temporal properties. Thus, ATL-ATL* [2] appears as a generalization of CTL-CTL*, in which the path quantifiers **E** and **A** are replaced by *strategy quantifiers*. Strategy quantifiers (the existential $\langle\langle A\rangle\rangle$ and the universal $[\![A]\!]$) have a (coalition of) agent(s) as parameter. $\langle\langle A\rangle\rangle\varphi$ means that agents in A can act so as to ensure the satisfaction of temporal formula φ. It is interpreted in *Concurrent Game Structures* (CGS), where agents can make choices influencing the execution in the system. Formula $\langle\langle A\rangle\rangle\varphi$ is true if agents in A have a strategy so that if playing it they force the execution to satisfy φ, whatever the other agents do.

A natural question is: how to interpret the imbrication of several quantifiers? Precisely, in the interpretation of such formula as

$$\psi_1 := \langle\langle a_1\rangle\rangle\Box(\varphi_1 \wedge \langle\langle a_2\rangle\rangle\Box\varphi_2)$$

(where $\Box\varphi$ is the temporal operator meaning φ is always true, and a_1 and a_2 are agents), is the evaluation of φ_2 made in a context that takes into account both the strategy quantified in $\langle\langle a_1\rangle\rangle$ and the strategy quantified in $\langle\langle a_2\rangle\rangle$?

In ATL-ATL*, only a_2 is bound: subformula $\langle\langle a_2\rangle\rangle\Box\varphi_2$ is true iff a_2 may ensure $\Box\varphi_2$, whatever the other agents do. Then $\langle\langle a_2\rangle\rangle$ stands for three successive operations: First, each agent is unbound from its current strategy, then an existential quantification is made for strategy σ. At last, a_2 is bound to strategy σ.

ATL$_{sc}$ [3, 6, 7], while keeping the ATL syntax, adapts the semantics in order to interpret formulas in a context which stores strategies introduced by earlier quantifiers.

F. Mogavero, A. Murano, and M.Y. Vardi (Eds.):
1st Workshop on Strategic Reasoning 2013 (SR'13)
EPTCS 112, 2013, pp. 91–98, doi:10.4204/EPTCS.112.14

Strategy Logic (SL [8, 10]) is another interesting proposition, which distinguishes between the quantifications over strategies and their bindings to agents. The operator $\langle\langle a \rangle\rangle$ is split into two different operators: a quantifier over strategies ($\langle\langle x \rangle\rangle$, where x is a strategy variable) and a binder ((a,x), where a is an agent) that stores into a context the information that a plays along the strategy instantiating variable x (let us write it σ_x in the remaining of this paper).The ATL formula ψ_1 syntactically matches the SL:

$$\psi_2 := \langle\langle x_1 \rangle\rangle (a_1, x_1) \Box (\varphi_1 \wedge \langle\langle x_2 \rangle\rangle (a_2, x_2) \Box \varphi_2)$$

In ψ_2, when evaluating $\Box\varphi_2$, a_1 remains bound to strategy σ_{x_1} except if a_1 and a_2 are the same agent. If they are the same, the binder (a_2, x_2) unbinds a from its current strategies before binding her to σ_{x_2}.

In this paper we present USL, a logic obtained from SL by making explicit the unbinding of strategies and allowing new bindings without previous unbinding. For that, we introduce an explicit unbinder $(a \not\rhd x)$ in the syntax (and the binder in USL is written $(a \rhd x)$) and we interpret USL in models where the choices of agents are represented by the set of potential successors they enable from the current state. When there is no occurrence of an unbinder, each agent remains bound to her current strategies. Then different strategies can combine together even for a single agent, provided that they are *coherent*, which means they define choices in non-empty intersection (the notion is formally defined in Sect. 2).

The main interest in such introduction is to distinguish between cases where an agent composes strategies together and situations where she revokes a current strategy for playing an other one. If a_1 and a_2 are the same agents, then ψ_2 is written in SL:

$$\psi_3 := \langle\langle x_1 \rangle\rangle (a, x_1) \Box (\varphi_1 \wedge \langle\langle x_2 \rangle\rangle (a, x_2) \Box \varphi_2),$$

which syntactically matches the USL:

$$\psi_4 := \langle\langle x_1 \rangle\rangle (a \rhd x_1) \Box (\varphi_1 \wedge \langle\langle x_2 \rangle\rangle (a \rhd x_2) \Box \varphi_2)$$

In ψ_3, subformula $\langle\langle x_2 \rangle\rangle (a, x_2) \Box \varphi_2$ states that a can adopt a new strategy that ensures $\Box\varphi_2$, no matter if it is coherent with the strategy σ_{x_1} previously adopted. In ψ_4, both strategies must combine coherently together. In natural language ψ_4 states that a can ensure φ_1 and leave open the possibility to ensure φ_2 in addition. The equivalent of ψ_3 in USL is actually not ψ_4 but

$$\psi_5 := \langle\langle x_1 \rangle\rangle (a \rhd x_1) \Box (\varphi_1 \wedge \langle\langle x_2 \rangle\rangle (a \not\rhd x_1)(a \rhd x_2) \Box \varphi_2)$$

There indeed, in subformula $(a \not\rhd x_1)(a \rhd x_2)\Box\varphi_2$, a is first unbound from σ_{x_1} and then bound to σ_{x_2}.

A consequence of considering these compositions of strategies is the expressiveness of *sustainable capabilities* of agents. Let us now consider the USL formula:

$$\psi_6 := \langle\langle x_1 \rangle\rangle (a \rhd x_1) \Box (\langle\langle x_2 \rangle\rangle (a \not\rhd x_1)(a \rhd x_2) \mathbf{X}\ p)$$

There the binder $(a \rhd x_2)$ is used with the unbinder $(a \not\rhd x_1)$, so that ψ_6 is equivalent to the SL:

$$\psi_7 := \langle\langle x_1 \rangle\rangle (a, x_1) \Box (\langle\langle x_2 \rangle\rangle (a, x_2) \mathbf{X}\ p)$$

It states that a can remain capable to perform the condition expressed by $\mathbf{X}\ p$ when she wants. But in case she actually performs it, the formula satisfaction does not require that she is still capable to perform it. The statement holds in state s_0 in structure $\mathscr{M}_1$ with single agent a. See Fig.1, where choices are defined by the set of transitions they enable. Since $\mathscr{M}_1$ interprets SL formulas with only agent a, the choices for a are deterministic: let s, s' be two states and c a choice, then the transition from s to s' is

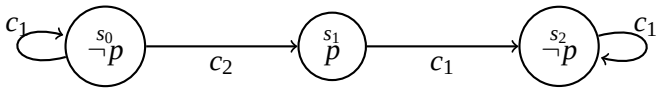


Figure 1: Structure $\mathscr{M}_1$

labelled with c iff $\{s'\}$ is a choice for a at s. Indeed, by always playing choice c_1, a remains in state s_0, where she can change her mind to ensure p. But if she chooses to reach p, she can do it only by moving to state s_1 and then to state s_2. Doing so, she loses her capability to ensure $\mathbf{X}\ p$ at any time. The only way for her to maintain her capability to reach p is to always avoid it, her capability is not sustainable.

A more game theoretical view is to consider strategies as commitments. In such view, by adopting a strategy, a adopts a behavior that holds in the following execution, as far as it is not explicitly deleted. Formula

$$\psi_8 := \langle\langle x_1\rangle\rangle(a \rhd x_1)\Box(\langle\langle x_2\rangle\rangle(a \rhd x_2)\mathbf{X}\ p)$$

is the counterpart of formula ψ_7 with such interpretation of composing strategies for a single agent. If a plays σ_{x_2}, it must be coherently with σ_{x_1}. Thus, ψ_8 is false in structure $\mathscr{M}_1$, since a cannot achieve p more than once.

Formula ψ_8 distinguishes between structures $\mathscr{M}_1$ and $\mathscr{M}_2$ from Fig.2 (Note that in this second structure the choices are not deterministic: from a given state a choice may be compatible with several potential successors). In $\mathscr{M}_2$, ψ_8 is true at s_0 since the strategy *always play* c_1 ensure the execution to remain in state s_0 or s_1 and is always coherent with strategy *play* c_2 *first and then always play* c_1, which ensures $\mathbf{X}\ p$ from states s_0 and s_1. What is at stake with it is the difference between *sustainable capabilities* and *one shot capabilities*. Formulas ψ_7 and ψ_8 both formalize the natural language proposition *a can always achieve p*. One shot capability (ψ_7) means she can achieve it once for all and choose when. Sustainable capability (ψ_8) means she can achieve it and choose when without affecting nor losing this capability for the future.

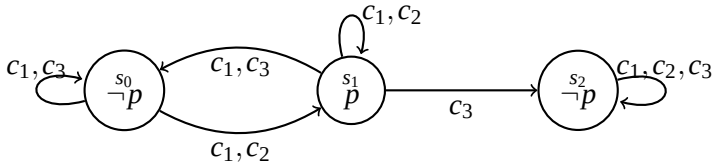


Figure 2: Structure $\mathscr{M}_2$

In Sect.3, we compare the expressive power of SL and USL by use of formula ψ_9, obtained from ψ_7 by adding to a the sustainable capability to ensure $\mathbf{X}\ \neg p$:

$$\psi_9 := \langle\langle x\rangle\rangle(a \rhd x)\Box(\langle\langle x_0\rangle\rangle(a \rhd x_0)\mathbf{X}\ p \wedge \langle\langle x_0\rangle\rangle(a \rhd x_0)\mathbf{X}\ \neg p)$$

ψ_9 states that a has sustainable capability to decide whether p or $\neg p$ holds at next state. We say that a has *sustainable* control on property p: she is sustainably capable to decide the truth value of p.

The main purposes of USL are to give a formalism for the composition of strategies and to unify it with the classical branching-time mechanisms of strategy revocation. So, both treatments can be combined in a single formalism. In the remaining of this paper we define USL syntax and semantics, and we introduce the comparison of its expressive power with that of SL.

2 Syntax and semantics

In this section we present the syntax and semantics of USL, together with the related definitions they require. The USL formulas distinguish between *path* and *state* formulas.

Definition 1. *Let Ag be a set of agents, At a set of propositions and X a set of variables, USL (Ag, At, X) is given by the following grammar:*

- *State formulas:* $\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle x \rangle\rangle \varphi \mid (A \rhd x)\psi \mid (A \not\rhd x)\psi$
- *Path formulas:* $\psi ::= \varphi \mid \neg\psi \mid \psi \wedge \psi \mid \psi \mathbf{U} \psi \mid \mathbf{X}\, \psi$

where $p \in At, A \subseteq Ag, x \in X$.

These formulas hold a notion of *free* variable that is similar to that in [8, 10]: an atom has an empty set of free variables, a binder adds a free variable to the set of free variables of its direct subformula and a quantifier deletes it. Upon formulas on this grammar, those that can be evaluated with no context are the *sentences*. They are formulas with empty set of free variables, which means each of their bound variables is previously quantified. We now come to the definitions for USL semantics.

Definition 2. *A Non-deterministic Alternating Transition System (NATS) is a tuple* $\mathscr{M} = \langle Ag, M, At, v, Ch \rangle$ *where:*

- *M is a set of states, called the domain of the NATS, At is the set of atomic propositions and v is a valuation function, from M to $\mathscr{P}(At)$.*
- *Ch: $Ag \times M \to \mathscr{P}(\mathscr{P}(M))$ is a choice function mapping a pair $(agent, state)$ to a non-empty family of choices of possible next states. It is such that for every state $s \in M$ and for every agents a_1 and a_2 in Ag, for every $c_1 \in Ch(a_1, s)$ and $c_2 \in Ch(a_2, s), c_1 \cap c_2 \neq \emptyset$.*

We call a finite sequence of states in M a *track* τ. The last element of a track τ is denoted by $last(\tau)$. The set of tracks that are possible in $\mathscr{M}$ is denoted by $track_{\mathscr{M}}$: $\tau = s_0 s_1 \ldots s_k \in track_{\mathscr{M}}$ iff for every $i < k$, for every $a \in Ag$, there is $c_a \in \mathscr{P}(M)$ s.t. $c_a \in Ch(a, s_i)$ and $s_{i+1} \in c_a$. Similarly, an infinite sequence of states such that all its prefixes are in $track_{\mathscr{M}}$ is called a *path* (in $\mathscr{M}$).

Definition 3 (Strategies and coherence)**.** *A* strategy *is a function σ from $Ag \times track_{\mathscr{M}}$ to $\mathscr{P}(M)$ such that for all $(a, \tau) \in Ag \times track_{\mathscr{M}}, \sigma(a, \tau) \in Ch(a, last(\tau))$. By extension, we write $\sigma(A, \tau)$ for $\bigcap_{a \in A} \sigma(a, \tau)$ for every $A \subseteq Ag$. Two strategies σ_1 and σ_2 are* coherent *iff for all (a, τ) in $Ag \times track_{\mathscr{M}}, \sigma_1(a, \tau) \cap \sigma_2(a, \tau) \neq \emptyset$. In this case, we also say that $\sigma_1(a, \tau)$ and $\sigma_2(a, \tau)$ are* coherent choices.

A *commitment* κ is a finite sequence upon $(\mathscr{P}(Ag) \times X)$, representing the active bindings. An *assignment* α is a partial function from X to $Strat$. A *context* χ is a pair of an assignment and a commitment. Note that an agent can appear several times in a commitment. Furthermore commitments store the order in which pairs (A, x) are introduced. Therefore our notion of contexts differs from the notion of *assignments* that is used in SL [8, 10].

A context defines a function from $track_{\mathscr{M}}$ to $\mathscr{P}(M)$. We use the same notation for the context itself and its induced function. Let $\kappa_\emptyset$ be the empty sequence upon $(\mathscr{P}(Ag) \times X)$, then:

- $(\alpha, \kappa_\emptyset)(\tau) = M$
- $(\alpha, (A, x))(\tau) =$
  - $\bigcap_{a \in A} \alpha(x)(a, \tau)$ if $A \neq \emptyset$
  - else M
- $(\alpha, \kappa \cdot (A, x))(\tau) =$

- $(\alpha,\kappa)(\tau) \cap (\alpha,(A,x))(\tau)$ if this intersection is not empty.
- otherwise (which means the context induces contradictory choices), $(\alpha,\kappa)(\tau)$.

Now we can define the outcomes of a context χ, *out* (χ): let $\pi = \pi_0, \pi_1, \ldots$ be an infinite sequence over M, then $\pi \in out(s,\chi)$ iff π is a path in $\mathscr{M}$, $s = \pi_0$ and for every $n \in \mathbb{N}$, $\pi_{n+1} \in \chi(\pi_0 \ldots \pi_n)$.

Definition 4 (Strategy and assignment translation)**.** *Let σ be a strategy and τ be a track. Then σ^τ is the strategy s.t. for every $\tau' \in track_{\mathscr{M}}$, $\sigma^\tau(\tau') = \sigma(\tau\tau')$. The notion is extended to an assignment: for every α, α^τ is the assignment with domain equal to that of α and s.t. for every $x \in dom(\alpha), \alpha^\tau(x) = (\alpha(x))^\tau$*

We also define the following transformations of commitments and assignments. Given a commitment κ, coalitions A and B, a strategy variable x, an assignment α and a strategy σ:

- $\kappa[A \to x] = \kappa \cdot (A \rhd x)$
- $((B,x)\cdot\kappa)[A \nrightarrow x] = (B \backslash A, x) \cdot (\kappa[A \nrightarrow x])$ and $\kappa_\emptyset[A \nrightarrow x] = \kappa_\emptyset$
- $\alpha[x \to \sigma]$ is the assignment with domain $dom(\alpha) \cup \{x\}$ s.t. $\forall y \in dom(\alpha)\backslash\{x\}, \alpha[x \to \sigma](y) = \alpha(y)$ and $\alpha[x \to \sigma](x) = \sigma$

Definition 5 (Satisfaction relation)**.** *Let $\mathscr{M}$ be a NATS, then for every assignment α, commitment κ, state s and path π:*

- *State formulas:*
  - $\mathscr{M}, \alpha, \kappa, s \models p$ *iff* $p \in v(s)$*, with* $p \in At$
  - $\mathscr{M}, \alpha, \kappa, s \models \neg\varphi$ *iff it is not true that* $\mathscr{M}, \alpha, \kappa, s \models \varphi$
  - $\mathscr{M}, \alpha, \kappa, s \models \varphi_1 \wedge \varphi_2$ *iff* $\mathscr{M}, \alpha, \kappa, s \models \varphi_1$ *and* $\mathscr{M}, \alpha, \kappa, s \models \varphi_2$
  - $\mathscr{M}, \alpha, \kappa, s \models \langle\langle x \rangle\rangle \varphi$ *iff there is a strategy* $\sigma \in Strat$ *s.t.* $\mathscr{M}, \alpha[x \to \sigma], \kappa, s \models \varphi$
  - $\mathscr{M}, \alpha, \kappa, s \models (A \rhd x)\varphi$ *iff for every* π *in* $out(\alpha, \kappa[A \to x]), \mathscr{M}, \alpha, \kappa[A \to x], \pi \models \varphi$
  - $\mathscr{M}, \alpha, \kappa, s \models (A \not\rhd x)\varphi$ *iff for all* π *in* $out(\alpha, \kappa[A \nrightarrow x]), \mathscr{M}, \alpha, \kappa[A \nrightarrow x], \pi \models \varphi$
- *Path formulas :*
  - $\mathscr{M}, \alpha, \kappa, \pi \models \varphi$ *iff* $\mathscr{M}, \alpha, \kappa, \pi_0 \models \varphi$*, for every state formula* φ
  - $\mathscr{M}, \alpha, \kappa, \pi \models \neg\psi$ *iff it is not true that* $\mathscr{M}, \alpha, \kappa, \pi \models \psi$
  - $\mathscr{M}, \alpha, \kappa, \pi \models \psi_1 \wedge \psi_2$ *iff* $\mathscr{M}, \alpha, \kappa, \pi \models \psi_1$ *and* $\mathscr{M}, \alpha, \kappa, \pi \models \psi_2$
  - $\mathscr{M}, \alpha, \kappa, \pi \models \mathbf{X}\ \psi$ *iff* $\mathscr{M}, \alpha^{\pi_0}, \kappa, \pi^1 \models \psi$.
  - $\mathscr{M}, \alpha, \kappa, \pi \models \psi_1\ \mathbf{U}\ \psi_2$ *iff there is* $i \in \mathbb{N}$ *s.t.* $\mathscr{M}, \alpha^{\pi_0 \ldots \pi_{i-1}}, \kappa, \pi^i \models \psi_2$ *and for every* $0 \leq j < i, \mathscr{M}, \alpha^{\pi_0 \ldots \pi_{j-1}}, \kappa, \pi^j \models \psi_1$

Let $\alpha_\emptyset$ be the unique assignment with empty domain. Let φ be a sentence in USL (Ag, At, X). Then $\mathscr{M}, s \models \varphi$ iff $\mathscr{M}, \alpha_\emptyset, \kappa_\emptyset \models \varphi$.

Let us give the following comment over these definitions: for every context $\chi = (\alpha, \kappa)$, the definition of *out* (χ) ensures that the different binders encoded in χ compose their choices together, *as far as possible*. In case two contradictory choices from an agent are encoded in the context, the priority is given to the first binding that was introduced in this context (the left most binding in the formula). This guarantees that a formula requiring the composition of two contradictory strategies is false. For example, suppose that $\langle\langle x_1 \rangle\rangle (a \rhd x_1)\varphi_1$ and $\langle\langle x_2 \rangle\rangle (a \rhd x_2)\varphi_2$ are both true in a state of a model, and suppose that strategies σ_{x_1} and σ_{x_2} necessarily rely on contradictory choices of a (this means that a cannot play in a way that ensures both φ_1 and φ_2). Then, $\langle\langle x_1 \rangle\rangle (a \rhd x_1)(\varphi_1 \wedge \langle\langle x_2 \rangle\rangle (a \rhd x_2)\varphi_2)$ is false in the same state of the same model. If the priority was given to the most recent binding (right most binding in the formula), the strategy σ_{x_1} would be revoked and the formula would be satisfied.

3 Comparison with SL [8, 10]

SL syntax can be basically described from SL by deleting the use of the unbinder. Furthermore, the binders are limited to sole agents and are written (a,x) instead of $(a \triangleright x)$. USL appears to be more expressive than SL [8, 10]. More precisely, SL can be embedded in USL, while ψ_9 is not expressible in SL, even by extending its semantics to non-deterministic models. Here we give the three related propositions. By lack of space, the proofs are only sketched in this article. Detailed proofs of these propositions can be found in [4]. Note that, since SL is strictly more expressive than ATL_{sc} [6], the following results also hold for comparing USL with ATL_{sc}.

Proposition 1. *There is an embedding of SL into USL.*

Proof (Sketch). The embedding consists in a parallel transformation from SL models and formulas to that of USL. The transformation preserves the satisfaction relation. The differences between SL and USL lie both in the definition of strategies in SL semantics and the difference of interpretation for the binding operator. The first is treated by defining an internal transformation for SL. By this transformation, the constraints of agents playing the same choices, issued from SL actions framework, are expressed in the syntax. Then we define a new operator in USL that is equivalent to SL binding, and show the equivalence: the operator $[a \triangleright x]$ is an abbreviation for a binder $(a \triangleright x)$ preceded by the set of unbinders $(a \not\triangleright x_i)$, one for every variable x_i in the language. □

Proposition 2. *A model is said* deterministic *if the successor of a state is uniquely determined by one choice for every agent. Then, sustainable control is not expressible over deterministic models, neither in SL nor in USL.*

Proof (Sketch). One checks that for every deterministic NATS $\mathscr{M}$, for any state s of $\mathscr{M}$, $\mathscr{M}, s \not\models \psi_9$. Proposition 1 then straightly brings proposition 2 □

Proposition 3. *Sustainable control is not expressible in SL interpreted over NATSs.*

Proof (Sketch). The proof uses a generalization of SL semantics over *NATSs*. Its definition is in [4] and holds, for example, the following cases:

- $\mathscr{M}, \alpha, \kappa, s \models_{\text{NATS}} \mathbf{X}\, \varphi$ iff for every $\pi \in out(s, (\alpha, \kappa)), \mathscr{M}, \alpha^{\pi_0}, \kappa, \pi_1 \models_{\text{NATS}} \varphi$
- $\mathscr{M}, \alpha, \kappa, \pi \models_{\text{NATS}} \varphi_1\ \mathbf{U}\ \varphi_2$ iff for every $\pi \in out(s, (\alpha, \kappa))$, there is $i \in \mathbb{N}$ s.t. $\mathscr{M}, \alpha^{\pi_0 \cdots \pi_{i-1}}, \kappa, \pi^i \models_{\text{NATS}} \varphi_2$ and for all $0 \leq j \leq i, \mathscr{M}, \alpha^{\pi_0 \cdots \pi_{i-1}}, \kappa, \pi^j \models_{\text{NATS}} \varphi_1$.
- $\mathscr{M}, \alpha, \kappa, s \models_{\text{NATS}} \langle\langle x \rangle\rangle \varphi$ iff there is a strategy $\sigma \in \mathit{Strat}$ s.t. $\mathscr{M}, \alpha[x \to \sigma], \kappa, s \models_{\text{NATS}} \varphi$.
- $\mathscr{M}, \alpha, \kappa, s \models_{\text{NATS}} (a,x)\varphi$ iff $\mathscr{M}, \alpha, \kappa[x \backslash \kappa(a)], s \models_{\text{NATS}} \varphi$.

where $\kappa[x \backslash \kappa(a)]$ designates the context obtained from κ by replacing every (a,y) in it by (a,x).

Formula ψ_9 states that a can always control whether p or not. Suppose there is a formula φ in SL equivalent to ψ_9 and let us call *existential* a formula in SL in which every occurrence of $\langle\langle x \rangle\rangle$ is under an even number of quantifiers. If φ is existential then under binary trees it is equivalent to a formula in Σ^1_1 (the fragment of second order logic with only existential set quantifiers).

We now consider a set of formulas $\{\Gamma_i\}_{i \in \mathbb{N}}$, each one stating that a can choose i times between p and $\neg p$. The set $\{\Gamma_i\}_{i \in \mathbb{N}}$ is defined by induction over i:

- $\Gamma_0 := \langle\langle x \rangle\rangle (a,x) \Box (\langle\langle x_0 \rangle\rangle (a,x_0) \mathbf{X}\, p \wedge \langle\langle x_0 \rangle\rangle (a,x_0) \mathbf{X}\, \neg p)$

- for all $i \in \mathbb{N}, \Gamma i+1 = \Gamma_i[p \wedge \Box(\langle\langle x_{i+1}\rangle\rangle(a,x_{i+1})\mathbf{X}\ p \wedge \langle\langle x_{i+1}\rangle\rangle(a,x_{i+1})\mathbf{X}\ \neg p \backslash p]$
  $[\neg p \wedge \Box(\langle\langle x_{i+1}\rangle\rangle(a,x_{i+1})\mathbf{X}\ p \wedge \langle\langle x_{i+1}\rangle\rangle(a,x_{i+1})\mathbf{X}\ \neg p) \backslash \neg p]$.

where the notation $\theta_1[\theta_2 \backslash \theta_3]$ designates the formula obtained from θ_1 by replacing any occurrence of subformula θ_3 in it by θ_2. $\{\Gamma_i\}_{i\in\mathbb{N}}$ is equivalent to φ. A compactness argument shows that it is not equivalent to a formula in Σ_1^1 under binary trees, hence φ is not an existential formula. Then, we notice that φ is true in structures where, from any state, a can ensure any labelling of sequences over p. So, if φ has a subformula $(a,x)\psi$ where x is universally quantified, ψ must be equivalent to $\Box(p \vee \neg p)$. Then, by iteration, φ is equivalent to an existential formula in SL. Hence a contradiction. □

4 Conclusion

In this article we defined a strategy logic with updatable strategies. By updating a strategy, agents remain playing along it but add further precision to their choices. This mechanism enables to express such properties as sustainable capability and sustainable control. To the best of our knowledge, this is the first proposition for expressing such properties. Especially, the comparison introduced with SL in this article could be adapted to ATL with Strategy Context [3].

The revocation of strategies is also questioned in [1]. The authors propose a formalism with definitive strategies, that completely determine the behaviour of agents. They also underline the difference between these strategies and revocable strategies in the classical sense. We believe that updatable strategies offer a synthesis between both views: updatable strategies can be modified without being revoked.

Strategies in USL can also be explicitly revoked. This idea is already present in [3] with the operator $\cdot\rangle A\langle\cdot$. But the operator $\langle\cdot A\cdot\rangle$ also implicitly unbinds current strategy for agents in A before binding them a new strategy. Thus it prevents agents from updating their strategy or composing several strategies.

Further study perspectives about USL mainly concern the model checking. Further work will provide it with a proof of non elementary decidability, adapted from the proof in [10]. We are also working on a semantics for USL under memory-less strategies and **PSPACE** algorithm for its model-checking. Satisfiability problem should also be addressed. Since SL SAT problem is not decidable, similar result is expectable for USL. Nevertheless, decidable fragments of USL may be studied in the future, in particular by following the directions given in [9].

References

[1] Thomas Ågotnes, Valentin Goranko & Wojciech Jamroga (2007): *Alternating-time temporal logics with irrevocable strategies*. In: *Theoretical aspects of rationality and knowledge*, pp. 15–24, doi:10.1145/1324249.1324256.

[2] Rajeev Alur, Thomas A. Henzinger & Orna Kupferman (2002): *Alternating-time temporal logic*. *J. ACM* 49(5), pp. 672–713, doi:10.1145/585265.585270.

[3] T. Brihaye, A. Da Costa, F. Laroussinie & N. Markey (2009): *ATL with strategy contexts and bounded memory*. *Logical Foundations of Computer Science*, pp. 92–106, doi:10.1007/978-3-540-92687-0_7.

[4] Christophe Chareton, Julien Brunel & David Chemouil (2013): *Updatable Strategy Logic*. hal-00785659. Available at http://hal.archives-ouvertes.fr/hal-00785659. Submitted.

[5] Krishnendu Chatterjee, Thomas A. Henzinger & Nir Piterman (2010): *Strategy logic*. *Inf. & Comp.* 208(6), pp. 677–693, doi:10.1016/j.ic.2009.07.004.

[6] Arnaud Da Costa Lopes (2011): *Propriétés de jeux multi-agents*. Phd thesis, École normale supérieure de Cachan.

[7] Arnaud Da Costa Lopes, François Laroussinie & Nicolas Markey (2010): *ATL with Strategy Contexts: Expressiveness and Model Checking*. In: *FSTTCS*, pp. 120–132, doi:10.4230/LIPIcs.FSTTCS.2010.120.

[8] Fabio Mogavero, Aniello Murano, Giuseppe Perelli & Moshe Y. Vardi (2011): *Reasoning About Strategies: On the Model-Checking Problem*. *CoRR* abs/1112.6275. Available at http://arxiv.org/abs/1112.6275.

[9] Fabio Mogavero, Aniello Murano, Giuseppe Perelli & Moshe Y. Vardi (2012): *What Makes Atl* Decidable? A Decidable Fragment of Strategy Logic*. In: *CONCUR*, pp. 193–208, doi:10.1007/978-3-642-32940-1_15.

[10] Fabio Mogavero, Aniello Murano & Moshe Y. Vardi (2010): *Reasoning about strategies*. In: *FSTTCS*, 8, pp. 133–144, doi:10.4230/LIPIcs.FSTTCS.2010.133.

A rewriting point of view on strategies

Hélène Kirchner
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex France
e-mail:Helene.Kirchner@inria.fr

This paper is an expository contribution reporting on published work. It focuses on an approach followed in the rewriting community to formalize the concept of strategy. Based on rewriting concepts, several definitions of strategy are reviewed and connected: in order to catch the higher-order nature of strategies, a strategy is defined as a proof term expressed in the rewriting logic or in the rewriting calculus; to address in a coherent way deduction and computation, a strategy is seen as a subset of derivations; and to recover the definition of strategy in sequential path-building games or in functional programs, a strategy is considered as a partial function that associates to a reduction-in-progress, the possible next steps in the reduction sequence.

1 Introduction

Strategies frequently occur in automated deduction and reasoning systems and more generally are used to express complex designs for control in modeling, proof search, program transformation, SAT solving or security policies. In these domains, deterministic rule-based computations or deductions are often not sufficient to capture complex computations or proof developments. A formal mechanism is needed, for instance, to sequentialize the search for different solutions, to check context conditions, to request user input to instantiate variables, to process subgoals in a particular order, etc. This is the place where the notion of strategy comes in.

This paper deliberately focuses on an approach followed in the rewriting community to formalize a notion of strategy relying on rewriting logic [17] and rewriting calculus [7] that are powerful formalisms to express and study uniformly computations and deductions in automated deduction and reasoning systems. Briefly speaking, rules describe local transformations and strategies describe the control of rule application. Most often, it is useful to distinguish between rules for computations, where a unique normal form is required and where the strategy is fixed, and rules for deductions, in which case no confluence nor termination is required but an application strategy is necessary. Regarding rewriting as a relation and considering abstract rewrite systems leads to consider derivation tree exploration: derivations are computations and strategies describe selected computations.

Based on rewriting concepts, that are briefly recalled in Section 2, several definitions of strategy are reviewed and connected. In order to catch the higher-order nature of strategies, a strategy is first defined as a proof term expressed in rewriting logic in Section 3 then in rewriting calculus in Section 4. In Section 5, a strategy is seen as a set of paths in a derivation tree; then to recover the definition of strategy in sequential path-building games or in functional programs, a strategy is considered as a partial function that associates to a reduction-in-progress, the possible next steps in the reduction sequence. In this paper, the goal is to show the progression of ideas and definitions of the concept, as well as their correlations.

F. Mogavero, A. Murano, and M.Y. Vardi (Eds.):
1st Workshop on Strategic Reasoning 2013 (SR'13)
EPTCS 112, 2013, pp. 99–105, doi:10.4204/EPTCS.112.15

2 Rewriting

Since the 80s, many aspects of rewriting have been studied in automated deduction, programming languages, equational theory decidability, program or proof transformation, but also in various domains such as chemical or biological computing, plant growth modeling, etc. In all these applications, rewriting definitions have the same basic ingredients. Rewriting transforms syntactic structures that may be words, terms, propositions, dags, graphs, geometric objects like segments, and in general any kind of structured objects. Transformations are expressed with patterns or rules. Rules are built on the same syntax but with an additional set of variables, say $\mathscr{X}$, and with a binder $\Rightarrow$, relating the left-hand side and the right-hand side of the rule, and optionally with a condition or constraint that restricts the set of values allowed for the variables. Performing the transformation of a syntactic structure t is applying the rule labeled ℓ on t, which is basically done in three steps: (1) match to select a redex of t at position p denoted $t_{|p}$ (possibly modulo some axioms, constraints,...); (2) instantiate the rule variables by the result(s) of the matching substitution σ; (3) replace the redex by the instantiated right-hand side. Formally: t rewrites to t' using the rule $\ell : l \Rightarrow r$ if $t_{|p} = \sigma(l)$ and $t' = t[\sigma(r)]_p$. This is denoted $t \longrightarrow_{p,\ell,\sigma} t'$.

In this process, there are many possible choices: the rule itself, the position(s) in the structure, the matching substitution(s). For instance, one may choose to apply a rule concurrently at all disjoint positions where it matches, or using matching modulo an equational theory like associativity-commutativity, or also according to some probability.

3 Rewriting logic

The Rewriting Logic is due to J. Meseguer and N. Martí-Oliet [17].
As claimed on `http://wrla2012.lcc.uma.es/`:

Rewriting logic (RL) is a natural model of computation and an expressive semantic framework for concurrency, parallelism, communication, and interaction. It can be used for specifying a wide range of systems and languages in various application fields. It also has good properties as a metalogical framework for representing logics. In recent years, several languages based on RL (ASF+SDF, CafeOBJ, ELAN, Maude) have been designed and implemented.

In Rewriting Logic, the syntax is based on a set of terms $\mathscr{T}(\mathscr{F})$ built with an alphabet $\mathscr{F}$ of function symbols with arities, a theory is given by a set $\mathscr{R}$ of labeled rewrite rules denoted $\ell(x_1,\ldots,x_n) : l \Rightarrow r$, where labels $\ell(x_1,\ldots,x_n)$ record the set of variables occurring in the rewrite rule. Formulas are sequents of the form $\pi : t \to t'$, where π is a *proof term* recording the proof of the sequent: $\mathscr{R} \vdash \pi : t \to t'$ if $\pi : t \to t'$ can be obtained by finite application of equational deduction rules given below. In this context, a proof term π encodes a sequence of rewriting steps called a derivation.

Reflexivity For any $t \in \mathscr{T}(\mathscr{F})$:

$$\mathbf{t} : t \to t$$

Congruence For any $f \in \mathscr{F}$ with $arity(f) = n$:

$$\frac{\pi_\mathbf{1} : t_1 \to t'_1 \quad \ldots \quad \pi_\mathbf{n} : t_n \to t'_n}{\mathbf{f}(\pi_\mathbf{1},\ldots,\pi_\mathbf{n}) : f(t_1,\ldots,t_n) \to f(t'_1,\ldots,t'_n)}$$

Transitivity

$$\frac{\pi_\mathbf{1} : t_1 \to t_2 \qquad \pi_\mathbf{2} : t_2 \to t_3}{\pi_\mathbf{1};\pi_\mathbf{2} \ : \ t_1 \to t_3}$$

Replacement For any $\ell(x_1,\ldots,x_n) : l \Rightarrow r \in \mathscr{R}$,

$$\frac{\pi_1 : t_1 \to t'_1 \quad \ldots \quad \pi_n : t_n \to t'_n}{\ell(\pi_1,\ldots,\pi_n) : l(t_1,\ldots,t_n) \to r(t'_1,\ldots,t'_n)}$$

The ELAN language, designed in 1997, introduced the concept of strategy by giving explicit constructs for expressing control on the rule application [5]. Beyond labeled rules and concatenation denoted ";", other constructs for deterministic or non-deterministic choice, failure, iteration, were also defined in ELAN. A strategy is there defined as a set of proof terms in rewriting logic and can be seen as a higher-order function : if the strategy ζ is a set of proof terms π, applying ζ to the term t means finding all terms t' such that $\pi : t \to t'$ with $\pi \in \zeta$. Since rewriting logic is reflective, strategy semantics can be defined inside the rewriting logic by rewrite rules at the meta-level. This is the approach followed by Maude in [8, 18].

4 Rewriting Calculus

The rewriting calculus, also called ρ-calculus, has been introduced in 1998 by Horatiu Cirstea and Claude Kirchner [7]. As claimed on `http://rho.loria.fr/index.html`:

The rho-calculus has been introduced as a general means to uniformly integrate rewriting and λ-calculus. This calculus makes explicit and first-class all of its components: matching (possibly modulo given theories), abstraction, application and substitutions.

The rho-calculus is designed and used for logical and semantical purposes. It could be used with powerful type systems and for expressing the semantics of rule based as well as object oriented paradigms. It allows one to naturally express exceptions and imperative features as well as expressing elaborated rewriting strategies.

Some features of the rewriting calculus are worth emphasizing here: first-order terms and λ-terms are ρ-terms ($\lambda x.t$ is $(x \Rightarrow t)$); a rule is a ρ-term as well as a strategy, so rules and strategies are abstractions of the same nature and "first-class concepts"; application generalizes $\beta-$reduction; composition of strategies is like function composition; recursion is expressed as in λ calculus with a recursion operator μ.

In order to illustrate the use of ρ-calculus, let us consider the Abstract Biochemical Calculus (or ρ_{Bio}-calculus) [2]. This rewriting calculus models autonomous systems as *biochemical programs* which consist of the following components: collections of molecules (objects and rewrite rules), higher-order rewrite rules over molecules (that may introduce new rewrite rules in the behaviour of the system) and strategies for modeling the system's evolution. A visual representation via *port graphs* and an implementation are provided by the PORGY environment described in [1]. In this calculus, strategies are abstract molecules, expressed with an arrow constructor ($\Rightarrow$ for rule abstraction), an application operator $\bullet$ and a constant operator stk for explicit failure.

Below are examples of useful strategies in ρ_{Bio}-calculus:

$$\begin{array}{rcl}
\mathsf{id} & \triangleq & X \Rightarrow X \\
\mathtt{fail} & \triangleq & X \Rightarrow \mathsf{stk} \\
\mathtt{seq}(S_1,S_2) & \triangleq & X \Rightarrow S_2 \bullet (S_1 \bullet X) \\
\mathtt{first}(S_1,S_2) & \triangleq & X \Rightarrow (S_1 \bullet X) \;\; (\mathsf{stk} \Rightarrow (S_2 \bullet X)) \bullet (S_1 \bullet X) \\
\mathtt{try}(S) & \triangleq & \mathtt{first}(S, \mathsf{id}) \\
\mathtt{not}(S) & \triangleq & X \Rightarrow \mathtt{first}(\mathsf{stk} \Rightarrow X, X' \Rightarrow \mathsf{stk}) \bullet (S \bullet X) \\
\mathtt{ifTE}(S_1,S_2,S_3) & \triangleq & X \Rightarrow \mathtt{first}(\mathsf{stk} \Rightarrow S_3 \bullet X, X' \Rightarrow S_2 \bullet X) \bullet (S_1 \bullet X) \\
\mathtt{repeat}(S) & \triangleq & \mu X.\mathtt{try}(\mathtt{seq}(S,X))
\end{array}$$

Based on such constructions, the ρ_{Bio}-calculus allows failure handling, repair instructions, persistent application of rules or strategies, and more generally strategies for autonomic computing, as described in [3]. In [2], it is shown how to do invariant verification in biochemical programs. Thanks to ρ_{Bio}-calculus, an invariant property can in many cases, be encoded as a special rule in the biochemical program modeling the system and this rule is dynamically checked at each execution step. For instance, an invariant of the system is encoded by a rule $G \Rightarrow G$ and the strategy verifying such an invariant is encoded with a persistent strategy $\mathtt{first}(G \Rightarrow G, X \Rightarrow \mathsf{stk})$. In a similar way, an unwanted occurrence of a concrete molecule G in the system can be modeled with the rule $(G \Rightarrow \mathsf{stk})$. And instead of yielding failure stk, the problem can be "repaired" by associating to each property the necessary rules or strategies to be inserted in the system in case of failure.

5 Abstract Reduction Systems

Another view of rewriting is to consider it as an abstract relation on structural objects. An *Abstract Reduction System (ARS)* [19, 15, 6] is a labeled oriented graph $(\mathscr{O},\mathscr{S})$ with a set of labels $\mathscr{L}$. The nodes in $\mathscr{O}$ are called *objects*. The oriented labeled edges in $\mathscr{S}$ are called *steps*: $a \xrightarrow{\phi} b$ or (a,ϕ,b), with *source* a, *target* b and *label* ϕ. Derivations are composition of steps.

For a given ARS $\mathscr{A}$, an *$\mathscr{A}$-derivation* is denoted $\pi : a_0 \xrightarrow{\phi_0} a_1 \xrightarrow{\phi_1} a_2 \ldots \xrightarrow{\phi_{n-1}} a_n$ or $a_0 \xrightarrow{\pi} a_n$, where $n \in \mathbb{N}$. The *source* of π is a_0 and its domain $Dom(\pi) = \{a_0\}$. The *target* of π is a_n and applying π to a_0 gives the singleton set $\{a_n\}$, which is denoted $\pi \bullet a_0 = \{a_n\}$.

Abstract strategies are defined in [15] and in [6] as follows: for a given ARS $\mathscr{A}$, an *abstract strategy* ζ is a subset of the set of all derivations (finite or not) of $\mathscr{A}$. The notions of domain and application are generalized as follows: $Dom(\zeta) = \bigcup_{\pi \in \zeta} Dom(\pi)$ and $\zeta \bullet a = \{b \mid \exists \pi \in \zeta$ such that $a \xrightarrow{\pi} b\} = \{\pi \bullet a \mid \pi \in \zeta\}$. Playing with these definitions, [6] explored adequate definitions of termination, normal form and confluence under strategy.

Since abstract reduction systems may involve infinite sets of objects, of reduction steps and of derivations, we can schematize them with constraints at different levels: (i) to describe the objects occurring in a derivation (ii) to describe, via the labels, requirements on the steps of reductions (iii) to describe the structure of the derivation itself (iv) to express requirements on the histories. The framework developed in [16] defines a strategy ζ as all instances $\sigma(D)$ of a derivation schema D such that σ is solution of a constraint C involving derivation variables, object variables and label variables. As a simple example, the infinite set of derivations of length one that transform a into $f(a^n)$ for all $n \in \mathbb{N}$, where $a^n = a * \ldots * a$ (n times), is simply described by: $(a \rightarrow f(X) \mid X * a =_A a * X)$, where $=_A$ indicates that the constraint is solved modulo associativity of the operator $*$. This very general definition of abstract strategies is called

extensional in [6] in the sense that a strategy is defined explicitly as a set of derivations of an abstract reduction system. The concept is useful to understand and unify reduction systems and deduction systems as explored in [15].

But abstract strategies do not capture another point of view, also frequently adopted in rewriting: a strategy is a partial function that associates to a reduction-in-progress, the possible next steps in the reduction sequence. Here, the strategy as a function depends only on the object and the derivation so far. This notion of strategy coincides with the definition of strategy in sequential path-building games, with applications to planning, verification and synthesis of concurrent systems [9]. This remark leads to the following *intensional* definition given in [6]. The essence of the idea is that strategies are considered as a way of constraining and guiding the steps of a reduction. So at any step in a derivation, it should be possible to say whether a contemplated next step obeys the strategy ζ. In order to take into account the past derivation steps to decide the next possible ones, the history of a derivation has to be memorized and available at each step. Through the notion of traced-object $[\alpha]\,a = [(a_0,\phi_0),\ldots,(a_n,\phi_n)]\,a$ in $\mathscr{O}^{[\mathscr{A}]}$, each object a memorizes how it has been reached with the trace α.

An *intensional strategy* for $\mathscr{A} = (\mathscr{O},\mathscr{S})$ is a partial function λ from $\mathscr{O}^{[\mathscr{A}]}$ to $2^{\mathscr{S}}$ such that for every traced object $[\alpha]\,a$, $\lambda([\alpha]\,a) \subseteq \{\pi \in \mathscr{S} \mid Dom(\pi) = a\}$. If $\lambda([\alpha]\,a)$ is a singleton, then the reduction step under λ is deterministic.

As described in [6], an intensional strategy λ naturally generates an abstract strategy, called its *extension*: this is the abstract strategy ζ_λ consisting of the following set of derivations:

$\forall n \in \mathbb{N}, \pi : a_0 \xrightarrow{\phi_0} a_1 \xrightarrow{\phi_1} a_2 \ldots \xrightarrow{\phi_{n-1}} a_n \in \zeta_\lambda \quad$ iff $\quad \forall j \in [0,n], \quad (a_j \xrightarrow{\phi_j} a_{j+1}) \in \lambda([\alpha]\,a_j)$.

This extension may obviously contain infinite derivations; in such a case it also contains all the finite derivations that are prefixes of the infinite ones, and so is closed under taking prefixes.

A special case are memoryless strategies, where the function λ does not depend on the history of the objects. This is the case of many strategies used in rewriting systems, as shown in the next example. Let us consider an abstract reduction system $\mathscr{A}$ where objects are terms, reduction is term rewriting with a rewrite rule in the rewrite system, and labels are positions where the rewrite rules are applied. Let us consider an order $<$ on the labels which is the prefix order on positions. Then the intensional strategy that corresponds to innermost rewriting is $\lambda_{inn}(t) = \{\pi : t \xrightarrow{p} t' \mid p = max(\{p' \mid t \xrightarrow{p'} t' \in \mathscr{S}\})\}$. When a lexicographic order is used, the classical *rightmost-innermost* strategy is obtained.

Another example, to illustrate the interest of traced objects, is the intensional strategy that restricts the derivations to be of bounded length k. Its definition makes use of the size of the trace α, denoted $|\alpha|$: $\lambda_{ltk}([\alpha]\,a) = \{\pi \mid \pi \in \mathscr{S},\, Dom(\pi) = a,\, |\alpha| < k-1\}$. However, as noticed in [6], the fact that intensional strategies generate only prefix closed abstract strategies prevents us from computing abstract strategies that look straightforward: there is no intensional strategy that can generate a set of derivations of length exactly k. Other solutions are provided in [6].

6 Conclusion

A lot of interesting questions about strategies are yet open, going from the definition of this concept and the interesting properties we may expect to prove, up to the definition of domain specific strategy languages. As further research topics, two directions seem really interesting to explore:

- The connection with Game theory strategies. In the fields of system design and verification, *games* have emerged as a key tool. Such games have been studied since the first half of 20th century in descriptive set theory [14], and they have been adapted and generalized for applications in formal verification; intro-

ductions can be found in [13, 20]. It is worth wondering whether the coincidence of the term "strategy" in the domains of rewriting and games is more than a pun. It should be fruitful to explore the connection and to be guided in the study of the foundations of strategies by some of the insights in the literature of games.
- Proving properties of strategies and strategic reductions. A lot of work has already begun in the rewriting community and have been presented in journals, workshops or conferences of this domain. For instance, properties of confluence, termination, or completeness for rewriting under strategies have been addressed, either based on schematization of derivation trees, as in [12], or by tuning proof methods to handle specific strategies (innermost, outermost, lazy strategies) as in [10, 11]. Other approaches as [4] use strategies transformation to equivalent rewrite systems to be able to reuse well-known methods. Finally, properties of strategies such as fairness or loop-freeness could be worthfully explored by making connections between different communities (functional programming, proof theory, verification, game theory,...).

Acknowledgements The results presented here are based on pioneer work in the ELAN language designed in the Protheo team from 1997 to 2002. They rely on joint work with many people, in particular Marian Vittek and Peter Borovanský, Claude Kirchner and Florent Kirchner, Dan Dougherty, Horatiu Cirstea and Tony Bourdier, Oana Andrei, Maribel Fernandez and Olivier Namet. I am grateful to José Meseguer and to the members of the PROTHEO and the PORGY teams, for many inspiring discussions on the topics of this talk.

References

[1] Oana Andrei, Maribel Fernández, Hélène Kirchner, Guy Melançon, Olivier Namet & Bruno Pinaud (2011): *PORGY: Strategy-Driven Interactive Transformation of Graphs*. In Rachid Echahed, editor: *TERMGRAPH*, *EPTCS* 48, pp. 54–68. Available at `http://dx.doi.org/10.4204/EPTCS.48.7`.

[2] Oana Andrei & Hélène Kirchner (2009): *A Port Graph Calculus for Autonomic Computing and Invariant Verification*. *Electronic Notes In Theoretical Computer Science* 253(4), pp. 17–38, doi:`10.1016/j.entcs.2009.10.015`.

[3] Oana Andrei & Hélène Kirchner (2009): *A Higher-Order Graph Calculus for Autonomic Computing*. In Marina Lipshteyn, Vadim E. Levit & Ross M. McConnell, editors: *Graph Theory, Computational Intelligence and Thought, Lecture Notes in Computer Science* 5420, Springer, pp. 15–26. Available at `http://dx.doi.org/10.1007/978-3-642-02029-2_2`.

[4] Emilie Balland, Pierre-Etienne Moreau & Antoine Reilles (2012): *Effective strategic programming for Java developers*. *Software: Practice and Experience*, doi:`10.1002/spe.2159`.

[5] Peter Borovanský, Claude Kirchner, Hélène Kirchner & Pierre-Etienne Moreau (2002): *ELAN from a rewriting logic point of view*. *Theoretical Computer Science* 2(285), pp. 155–185, doi:`10.1016/S0304-3975(01)00358-9`.

[6] Tony Bourdier, Horatiu Cirstea, Daniel J. Dougherty & Hélène Kirchner (2009): *Extensional and Intensional Strategies*. In Maribel Fernández, editor: *WRS*, *EPTCS* 15, pp. 1–19. Available at `http://dx.doi.org/10.4204/EPTCS.15.1`.

[7] Horatiu Cirstea & Claude Kirchner (2001): *The rewriting calculus — Part I* and *II*. *Logic Journal of the Interest Group in Pure and Applied Logics* 9(3), pp. 339–410, doi:`10.1093/jigpal/9.3.339`.

[8] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer & Carolyn L. Talcott, editors (2007): *All About Maude - A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic*. *Lecture Notes in Computer Science* 4350, Springer.

[9] Daniel J. Dougherty (2008): *Rewriting strategies and game strategies*. Internal report.

[10] J. Giesl & A Middeldorp (2003): *Innermost Termination of Context-Sensitive Rewriting*. In: *Proceedings of the 6th International Conference on Developments in Language Theory (DLT 2002), LNCS* 2450, Springer, Kyoto, Japan, pp. 231–244, doi:10.1007/3-540-45005-X_20.

[11] Jürgen Giesl, Matthias Raffelsieper, Peter Schneider-Kamp, Stephan Swiderski & René Thiemann (2011): *Automated termination proofs for Haskell by term rewriting*. *ACM Trans. Program. Lang. Syst.* 33(2), pp. 7:1–7:39, doi:10.1145/1890028.1890030.

[12] Isabelle Gnaedig & Hélène Kirchner (2009): *Termination of rewriting under strategies*. *ACM Trans. Comput. Logic* 10(2), pp. 1–52, doi:10.1145/1462179.1462182.

[13] Erich Grädel, Wolfgang Thomas & Thomas Wilke, editors (2002): *Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*. *Lecture Notes in Computer Science* 2500, Springer.

[14] Alexander S. Kechris (1995): *Classical Descriptive Set Theory*. *Graduate Texts in Mathematics* 156, Springer, doi:10.1007/978-1-4612-4190-4.

[15] Claude Kirchner, Florent Kirchner & Hélène Kirchner (2008): *Strategic Computations and Deductions*. In Christoph Benzmüller, Chad E. Brown, Jörg Siekmann & Richard Statman, editors: *Reasoning in Simple Type Theory. Festchrift in Honour of Peter B. Andrews on His 70th Birthday*, *Studies in Logic and the Foundations of Mathematics* 17, College Publications, pp. 339–364.

[16] Claude Kirchner, Florent Kirchner & Hélène Kirchner (2010): *Constraint Based Strategies*. In: *Proceedings 18th International Workshop on Functional and (Constraint) Logic Programming (WFLP 2009), Brasilia, LNCS* 5979, pp. 13–26, doi:10.1007/978-3-642-11999-6_2.

[17] Narciso Martí-Oliet & José Meseguer (1996): *Rewriting logic as a logical and semantic framework*. *Electr. Notes Theor. Comput. Sci.* 4, pp. 190–225. Available at http://dx.doi.org/10.1016/S1571-0661(04)00040-4.

[18] Narciso Martí-Oliet, José Meseguer & Alberto Verdejo (2008): *A rewriting semantics for Maude strategies*. *Electronic Notes in Theoretical Computer Science* 238(3), pp. 227–247, doi:10.1016/j.entcs.2009.05.022.

[19] Vincent van Oostrom & Roel de Vrijer (2003): *Term Rewriting Systems*, chapter 9: Strategies. *Cambridge Tracts in Theoretical Computer Science* 2, Cambridge University Press.

[20] Igor Walukiewicz (2004): *A Landscape with Games in the Background*. In: *19th IEEE Symposium on Logic in Computer Science (LICS 2004)*, pp. 356–366, doi:10.1109/LICS.2004.1319630.

Synthesizing Structured Reactive Programs via Deterministic Tree Automata

Benedikt Brütsch
RWTH Aachen University, Lehrstuhl für Informatik 7, Germany
bruetsch@automata.rwth-aachen.de

Existing approaches to the synthesis of reactive systems typically involve the construction of transition systems such as Mealy automata. However, in order to obtain a succinct representation of the desired system, structured programs can be a more suitable model. In 2011, Madhusudan proposed an algorithm to construct a structured reactive program for a given ω-regular specification without synthesizing a transition system first. His procedure is based on two-way alternating ω-automata on finite trees that recognize the set of "correct" programs.

We present a more elementary and direct approach using only deterministic bottom-up tree automata that compute so-called *signatures* for a given program. In doing so, we extend Madhusudan's results to the wider class of programs with bounded delay, which may read several input symbols before producing an output symbol (or vice versa). As a formal foundation, we inductively define a semantics for such programs.

1 Introduction

Algorithmic synthesis is a rapidly developing field with many application areas such as reactive sytems, planning and economics. Most approaches to the synthesis of reactive systems, for instance [2, 12, 11, 8], revolve around synthesizing transition systems such as Mealy or Moore automata. Unfortunately, the resulting transition systems can be very large. This has motivated the development of techniques for the reduction of their state space (for example, [6]). Furthermore, the method of bounded synthesis [14, 4] can be used to synthesize minimal transition systems by iteratively increasing the bound on the size of the resulting system until a solution is found. However, it is not always possible to obtain small transition systems. For example, for certain specifications in linear temporal logic (LTL), the size of the smallest transition systems satisfying these specifications is doubly exponential in the length of the formula [13].

Aminof, Mogavero and Murano [1] provide a round-based algorithm to synthesize hierarchical transition systems, which can be exponentially more succinct than corresponding "flat" transition systems. The desired system is constructed in a bottom-up manner: In each round, a specification is provided and the algorithm constructs a corresponding hierarchical transition system from a given library of available components and the hierarchical transition systems created in previous rounds. Thus, in order to obtain a small system in the last round, the specifications in the previous rounds have to be chosen in an appropriate way.

Current techniques for the synthesis of (potentially) succinct implementations in the form of circuits or programs typically proceed in an indirect way, by converting a transition system into such an implementation. For example, Bloem et al. [3] first construct a symbolic representation (a binary decision diagram) of an appropriate transition system and then extract a corresponding circuit. However, this indirect approach does not necessarily yield a succinct result.

F. Mogavero, A. Murano, and M.Y. Vardi (Eds.):
1st Workshop on Strategic Reasoning 2013 (SR'13)
EPTCS 112, 2013, pp. 107–113, doi:10.4204/EPTCS.112.16

Madhusudan addresses this issue in [10], where he proposes a procedure to synthesize programs without computing a transition system first. He considers *structured reactive programs* over a given set of Boolean variables, which can be significantly smaller (regarding the length of the program code) than equivalent transition systems. To some degree, these programs separate control flow from memory. Such a separation can also be found in a related approach that has recently been introduced by Gelderie [5], where strategies for infinite games are represented by *strategy machines*, which are equipped with control states and a memory tape.

Given a finite set of Boolean variables and a nondeterministic Büchi automaton recognizing the complement of the specification, Madhusudan constructs a two-way alternating ω-automaton on finite trees that recognizes the set of *all* programs over these variables that satisfy the specification. This automaton can be transformed into a nondeterministic tree automaton (NTA) to check for emptiness and extract a minimal program (regarding the height of the corresponding tree) from that set. In contrast to the transition systems constructed by classical synthesis algorithms, the synthesized program does not depend on the specific syntactic formulation of the specification, but only on its meaning.

In this paper, we present a direct construction of a deterministic bottom-up tree automaton (DTA) recognizing the set of correct programs, without a detour via more intricate types of automata. The DTA inductively computes a representation of the behavior of a given program in the form of so-called *signatures*. A similar representation is used by Lustig and Vardi in their work on the synthesis of reactive systems from component libraries [9] to characterize the behavior of the components.

Our approach is not limited to programs that read input and write output in strict alternation, but extends Madhusudan's results to the more general class of programs with *bounded delay*: In general, a program may read multiple input symbols before writing the next output symbol, or vice versa, causing a delay between the input sequence and the output sequence. In a game-theoretic context, such a program corresponds to a strategy for a controller in a game against the environment where in each move the controller is allowed to either choose at least one output symbol or skip and wait for the next input (see [7]). We consider programs that never cause a delay greater than a given bound $k \in \mathbb{N}$.

For a fixed k, the complexity of our construction matches that of Madhusudan's algorithm. In particular, the size of the resulting DTA is exponential in the size of the given nondeterministic Büchi automaton recognizing the complement of the specification, and doubly exponential in the number of program variables. In fact, we establish a lower bound, showing that the set of all programs over n Boolean variables that satisfy a given specification cannot even be recognized by an NTA with less than $2^{2^{n-1}}$ states, if any such programs exist. However, note that a DTA (or NTA) accepting precisely these programs enables us to extract a minimal program for the given specification and the given set of program variables. Hence, the synthesized program itself might be rather small.

To lay a foundation for our study of the synthesis of structured reactive programs, we define a formal semantics for such programs, which is only informally indicated by Madhusudan. To that end, we introduce the concept of *Input/Output/Internal machines (IOI machines)*, which are composable in the same way as structured programs. This allows for an inductive definition of the semantics.

2 Syntax and Semantics of Structured Programs

We consider a slight modification of the structured programming language defined in [10], using only single Boolean values as input and output symbols to simplify notation. *Expressions* and *programs* over a finite set B of Boolean variables are defined by the following grammar, where $b \in B$:

$$\langle expr \rangle \ ::= \ \texttt{true} \mid \texttt{false} \mid b \mid \langle expr \rangle \wedge \langle expr \rangle \mid \langle expr \rangle \vee \langle expr \rangle \mid \neg \langle expr \rangle$$

$$\langle prog \rangle \ ::= \ b \coloneqq \langle expr \rangle \mid \texttt{input } b \mid \texttt{output } b \mid \langle prog \rangle ; \langle prog \rangle$$
$$\texttt{if } \langle expr \rangle \texttt{ then } \langle prog \rangle \texttt{ else } \langle prog \rangle \mid \texttt{while } \langle expr \rangle \texttt{ do } \langle prog \rangle$$

Intuitively, "`input` b" reads a Boolean value and stores it in the variable b. Conversely, "`output` b" writes the current value of b. To define a formal semantics we associate with each program a so-called *IOI machine*. An IOI machine is a transition system with designated entry and exit states. It can have input, output and internal transitions, with labels of the form $(a_{\text{in}}, \varepsilon)$, $(\varepsilon, a_{\text{out}})$ or $(\varepsilon, \varepsilon)$, respectively, where $a_{\text{in}}, a_{\text{out}} \in \mathbb{B} = \{0,1\}$. An IOI machine is equipped with a finite set B of Boolean variables, whose valuation is uniquely determined at each state. A *valuation* is a function $\sigma \colon B \to \mathbb{B}$ that assigns a Boolean value to each variable.

The associated IOI machine of an atomic program (i.e., an input statement, output statement or assignment) has one entry state and exit state for each possible variable valuation, and its transitions lead from entry states to exit states. For example, at each entry state of the associated IOI machine of an atomic program of the form "`input` b", there are two outgoing input transitions – one for each possible input symbol. The target of such an input transition is the exit state whose variable valuation is obtained by replacing the value of b with the respective input symbol. The IOI machine of a composite program can be constructed inductively from the IOI machines of its subprograms, leveraging their entry and exit states and the variable valuations of these states.

A *computation* ϱ of a program is a finite or infinite sequence of subsequent transitions of the corresponding IOI machine:

$$\varrho = q_1 \xrightarrow{(a_1,b_1)} q_2 \xrightarrow{(a_2,b_2)} q_3 \xrightarrow{(a_3,b_3)} \cdots$$

The *label* of ϱ is the pair of finite or infinite words $(a_1a_2a_3\ldots,\ b_1b_2b_3\ldots) \in (\mathbb{B}^* \cup \mathbb{B}^\omega) \times (\mathbb{B}^* \cup \mathbb{B}^\omega)$. An *initial computation* starts at the unique entry state where all variables have the value 0. The *infinite behavior* $\langle\langle p \rangle\rangle$ of a program p is the set of infinite input/output sequences $(\alpha, \beta) \in \mathbb{B}^\omega \times \mathbb{B}^\omega$ that can be produced by an initial computation of p. Furthermore, we call a program *reactive* if all its initial computations can be extended to infinite computations that yield an infinite input and output sequence.

At any given time during a computation ϱ as above, the length of the input sequence $a_1a_2\ldots a_i$ and the output sequence $b_1b_2\ldots b_i$ might differ. The supremum of these length differences along a computation is called the *delay* of the computation. If the delay of a computation does not exceed a given bound $k \in \mathbb{N}$ then we call this computation *k-bounded*. A program is said to be k-bounded if all its computations are k-bounded. By restricting the infinite behavior of a program p to labels of k-bounded initial computations, we obtain the *k-bounded infinite behavior* $\langle\langle p \rangle\rangle_k$ of p.

3 Solving the Synthesis Problem Using Deterministic Tree Automata

The synthesis problem for structured reactive programs with bounded delay can be formulated as follows: Given an ω-regular specification $R \subseteq (\mathbb{B} \times \mathbb{B})^\omega$ representing the permissible input/output sequences, a finite set of Boolean variables B and a delay bound $k \in \mathbb{N}$, the task is to construct a structured reactive program p over B with k-bounded delay such that $\langle\langle p \rangle\rangle \subseteq R$ – or detect that no such program exists. (However, our results can easily be generalized to finite input and output alphabets other than $\mathbb{B}$ by allowing input and output statements that process multiple Boolean values as in [10].) In the following we assume that the specification R is provided in the form of a *nondeterministic Büchi automaton (NBA)* $\mathscr{A}_{\overline{R}}$ over the alphabet $\mathbb{B} \times \mathbb{B}$ that recognizes the complement of the specification, i.e., $\mathscr{L}(\mathscr{A}_{\overline{R}}) = (\mathbb{B} \times \mathbb{B})^\omega \setminus R$, which is always possible for ω-regular specifications.

Our synthesis procedure is based on the fact that programs can be viewed as trees. Figure 1 shows an example for a tree representation of a program. We use *deterministic bottom-up tree automata* (*DTAs*, see, for example, [15]) to recognize sets of programs. More specifically, we show the following theorem:

Theorem 1. *Let B be a finite set of Boolean variables, let $k \in \mathbb{N}$ and let $\mathcal{A}_{\bar{R}}$ be a nondeterministic Büchi automaton recognizing the complement of a specification $R \subseteq (\mathbb{B} \times \mathbb{B})^\omega$. We can construct a DTA that accepts a tree p iff p is a reactive program over B with k-bounded delay and $\langle\langle p \rangle\rangle \subseteq R$, such that the size of this DTA is doubly exponential in $|B|$ and k and exponential in the size of $\mathcal{A}_{\bar{R}}$.*

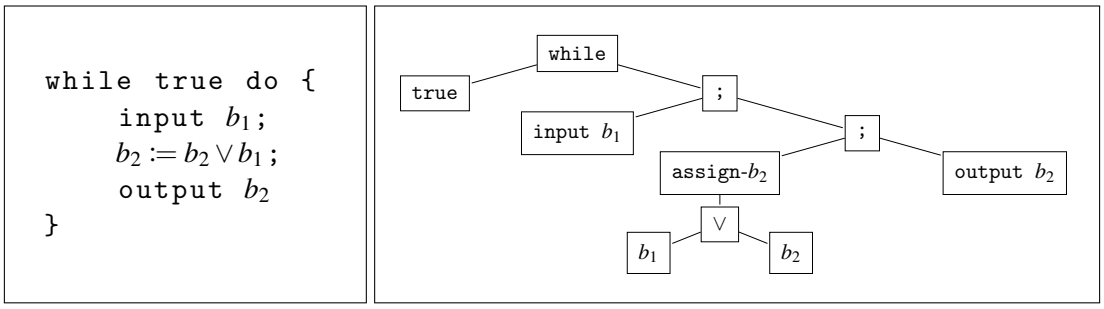


Figure 1: Example: A program and its tree representation.

An emptiness test on this DTA yields a solution to the synthesis problem. We obtain the desired tree automaton by intersecting three DTAs: The first DTA $\mathcal{B}_{\text{sat}}(B,k,\mathcal{A}_{\bar{R}})$ recognizes the set of programs over B whose k-bounded computations satisfy the specification R. That means, a program p is accepted iff $\langle\langle p \rangle\rangle_k \subseteq R$. The second DTA $\mathcal{B}_{\text{reactive}}(B)$ recognizes the reactive programs over B. Finally, we use a third DTA $\mathcal{B}_{\text{delay}}(B,k)$ to recognize the programs over B with k-bounded delay. We only consider the construction of $\mathcal{B}_{\text{sat}}(B,k,\mathcal{A}_{\bar{R}})$ here, as the other two DTAs can be constructed in a very similar way.

The DTA $\mathcal{B}_{\text{sat}}(B,k,\mathcal{A}_{\bar{R}})$ evaluates a given program p in a bottom-up manner, thereby assigning one of its states to each node of the program tree. The state reached at the root node must provide enough information to decide whether $\langle\langle p \rangle\rangle_k \subseteq R$, or equivalently, whether $\langle\langle p \rangle\rangle_k \cap \mathcal{L}(\mathcal{A}_{\bar{R}}) = \emptyset$. To that end, we are interested in the possible runs of $\mathcal{A}_{\bar{R}}$ on the input/output sequences generated by the program. Thus, we consider pairs of program computations and corresponding runs of $\mathcal{A}_{\bar{R}}$, which we call *co-executions*. Intuitively, $\mathcal{B}_{\text{sat}}(B,k,\mathcal{A}_{\bar{R}})$ inductively computes a representation of the possible co-executions of a given program and $\mathcal{A}_{\bar{R}}$. We define these representations, called *co-execution signatures*, in the following.

The beginning and end of a co-execution can be indicated by a valuation of the program variables and a state of $\mathcal{A}_{\bar{R}}$. However, we have to consider the following: The input sequence of a computation might be longer or shorter than its output sequence, but a run of $\mathcal{A}_{\bar{R}}$ only consumes input and output sequences of the same length. The suffix of the input/output sequence after the end of the shorter sequence, called the *overhanging suffix*, is hence still waiting to be consumed by $\mathcal{A}_{\bar{R}}$. Thus, we indicate the start and end of a co-execution by tuples of the form $\gamma = (\sigma, s, u, v)$, called *co-configurations*, where σ is a variable valuation, s is a state of $\mathcal{A}_{\bar{R}}$ and $(u,v) \in (\mathbb{B}^* \times \{\varepsilon\}) \cup (\{\varepsilon\} \times \mathbb{B}^*)$ is an overhanging suffix. Since we are only interested in k-bounded computations, we only consider co-configurations with $|u| \leq k$ and $|v| \leq k$. The set of these co-configurations for a given set of variables B and a given NBA $\mathcal{A}_{\bar{R}}$ is denoted by $CoCfg_k(B,\mathcal{A}_{\bar{R}})$.

A finite co-execution is called *complete* if the program terminates at the end of the computation. The *finite co-execution signature* $cosig^{\text{fin}}(p,\mathcal{A}_{\bar{R}},k)$ of a program p (with respect to $\mathcal{A}_{\bar{R}}$) is a relation consisting of tuples of the form (γ, f, γ') with $f \in \mathbb{B}$, which indicate that there exists a complete k-bounded co-execution that starts with the co-configuration γ and ends with γ' such that the corresponding run of

$\mathscr{A}_{\bar{R}}$ visits a final state iff $f = 1$. The *infinite co-execution signature* $cosig^{\infty}(p, \mathscr{A}_{\bar{R}}, k)$ of p is a set of co-configurations with $\gamma \in cosig^{\infty}(p, \mathscr{A}_{\bar{R}}, k)$ iff there exists an infinite k-bounded co-execution starting with γ such that the run of $\mathscr{A}_{\bar{R}}$ visits a final state infinitely often. We use pairs consisting of a finite and infinite co-execution signature as states of the DTA $\mathscr{B}_{\text{sat}}(B, k, \mathscr{A}_{\bar{R}})$. The size of the DTA is hence determined by the number of possible co-execution signatures, which is doubly exponential in the number of variables and k and exponential in the size of $\mathscr{A}_{\bar{R}}$. For a fixed k, this matches the complexity of Madhusudan's construction [10].

If σ_0 is the initial variable valuation (where all variables have the value 0) and s_0 is the initial state of $\mathscr{A}_{\bar{R}}$, then $(\sigma_0, s_0, \varepsilon, \varepsilon) \in cosig^{\infty}(p, \mathscr{A}_{\bar{R}}, k)$ iff there is an initial k-bounded computation of p such that some corresponding run of $\mathscr{A}_{\bar{R}}$ visits a final state infinitely often, so $cosig^{\infty}(p, \mathscr{A}_{\bar{R}}, k)$ is indeed sufficient to decide whether $\langle\langle p \rangle\rangle_k \subseteq R$. It remains to be shown that the co-execution signatures can be computed inductively. Exemplarily, we consider the case of programs of the form $p =$ "`while` e `do` p_1". First, we construct a representation $cosig_e^*(p_1, \mathscr{A}_{\bar{R}}, k)$ of all finite sequences of consecutive co-executions of p_1 that are compatible with the loop condition e. To that end, we consider only those tuples (γ, f, γ') in $cosig^{\text{fin}}(p_1, \mathscr{A}_{\bar{R}}, k)$ where the variable valuation in γ satisfies the loop condition e, and compute the reflexive transitive closure of the resulting relation. Formally, we have $cosig_e^*(p_1, \mathscr{A}_{\bar{R}}, k) = closure(C)$ with $C = \left\{ ((\sigma, s, u, v), f, \gamma') \in cosig^{\text{fin}}(p_1, \mathscr{A}_{\bar{R}}, k) \mid \sigma \in [\![e]\!] \right\}$. Here, $[\![e]\!]$ denotes the set of variable valuations that satisfy e, and $closure(C)$ is the smallest relation $D \subseteq CoCfg_k(B, \mathscr{A}_{\bar{R}}) \times \mathbb{B} \times CoCfg_k(B, \mathscr{A}_{\bar{R}})$ such that

- $(\gamma, 0, \gamma) \in D$ for all $\gamma \in CoCfg_k(B, \mathscr{A}_{\bar{R}})$, and
- $(\gamma, f_1, \gamma') \in D$, $(\gamma', f_2, \gamma'') \in C$ implies $(\gamma, \max\{f_1, f_2\}, \gamma'') \in D$.

Using $cosig_e^*(p_1, \mathscr{A}_{\bar{R}}, k)$, the co-execution signatures for p can be computed by the following reasoning: A finite co-execution of $p =$ "`while` e `do` p_1" (and $\mathscr{A}_{\bar{R}}$) can be decomposed into a finite sequence of co-executions of p_1. An infinite co-execution of p can either eventually stay inside a loop iteration forever or traverse infinitely many iterations. It can therefore be decomposed either into a finite sequence of co-executions of p_1 followed by an infinite co-execution of p_1, or into a finite sequence of co-executions of p_1 followed by a cycle of co-executions of p_1, leading back to a previous co-configuration. Thus, we obtain the following formal representation of the co-execution signatures for p:

- $(\gamma, f, (\sigma', s', u', v')) \in cosig^{\text{fin}}(p, \mathscr{A}_{\bar{R}}, k)$ iff $(\gamma, f, (\sigma', s', u', v')) \in cosig_e^*(p_1, \mathscr{A}_{\bar{R}}, k)$ and $\sigma' \notin [\![e]\!]$.
- $\gamma \in cosig^{\infty}(p, \mathscr{A}_{\bar{R}}, k)$ iff at least one of the following holds:
  - There exist $\gamma' = (\sigma', s', u', v') \in CoCfg_k(B, \mathscr{A}_{\bar{R}})$ and $f \in \mathbb{B}$ such that $(\gamma, f, \gamma') \in cosig_e^*(p_1, \mathscr{A}_{\bar{R}}, k)$, $\sigma' \in [\![e]\!]$ and $\gamma' \in cosig^{\infty}(p_1, \mathscr{A}_{\bar{R}}, k)$.
  - There exist $\gamma' = (\sigma', s', u', v') \in CoCfg_k(B, \mathscr{A}_{\bar{R}})$ and $f \in \mathbb{B}$ such that $(\gamma, f, \gamma') \in cosig_e^*(p_1, \mathscr{A}_{\bar{R}}, k)$, $\sigma' \in [\![e]\!]$ and $(\gamma', 1, \gamma') \in cosig_e^*(p_1, \mathscr{A}_{\bar{R}}, k)$.

4 Lower Bound for the Size of the Tree Automata

We show the following lower bound for the size of any nondeterministic tree automaton (NTA) recognizing the desired set of programs:

Theorem 2. *Let B be a set of n Boolean variables, let $k \in \mathbb{N}$ and let $R \subseteq (\mathbb{B} \times \mathbb{B})^{\omega}$ be a specification that is realizable by some program over B with k-bounded delay. Let $\mathscr{C}$ be an NTA that accepts a tree p iff p is a reactive program over B with k-bounded delay and $\langle\langle p \rangle\rangle \subseteq R$. Then $\mathscr{C}$ has at least $2^{2^{n-1}}$ states.*

For a sketch of the proof, consider a set of Boolean variables $B = \{b_1, \ldots, b_n\}$. There are $2^{2^{n-1}}$ functions of the type $\mathbb{B}^{n-1} \to \mathbb{B}$. Each of these functions can be implemented by a program that checks the values of $b_1, \ldots, b_{n-1}$ and sets b_n to the corresponding function value. An NTA as in Theorem 2 must be able to distinguish all of these programs. Otherwise, let p_i and p_j be two such programs that cannot be distinguished by the NTA. We could then construct a program that satisfies the specification and contains p_i as a subprogram, but runs into a non-reactive infinite loop if this subprogram is replaced by p_j. The NTA would accept both variants, including the non-reactive program, which contradicts the premise.

5 Conclusion

The contributions of this paper are threefold, advancing the study of structured reactive programs: We introduced a formal semantics for structured reactive programs in the sense of [10]. Furthermore, we presented a new synthesis algorithm for structured reactive programs with bounded delay, using the elementary concept of deterministic bottom-up tree automata. Finally, we showed a lower bound for the size of any nondeterministic tree automaton that recognizes the set of specification-compliant programs, emphasizing the importance of choosing a small yet still sufficient set of program variables. Estimating the number of Boolean variables that are needed to realize a given specification is a major open problem. While [13] implies an exponential upper bound for the required number of variables in the case of LTL specifications, a corresponding lower bound is still to be determined.

Acknowledgments. The author would like to thank Wolfgang Thomas for his helpful advice and Marcus Gelderie for fruitful discussions.

References

[1] Benjamin Aminof, Fabio Mogavero & Aniello Murano (2012): *Synthesis of Hierarchical Systems*. In Farhad Arbab & Peter Csaba Ölveczky, editors: *Formal Aspects of Component Software*, *Lecture Notes in Computer Science* 7253, Springer Berlin Heidelberg, pp. 42–60, doi:10.1007/978-3-642-35743-5_4.

[2] J. Richard Büchi & Lawrence H. Landweber (1969): *Solving Sequential Conditions by Finite-State Strategies*. *Transactions of the American Mathematical Society* 138, pp. 295–311, doi:10.2307/1994916.

[3] Roderick Bloem, Stefan Galler, Barbara Jobstmann, Nir Piterman, Amir Pnueli & Martin Weiglhofer (2007): *Specify, Compile, Run: Hardware from PSL*. *Electronic Notes in Theoretical Computer Science* 190(4), pp. 3 – 16, doi:10.1016/j.entcs.2007.09.004.

[4] Rüdiger Ehlers (2010): *Symbolic Bounded Synthesis*. In Tayssir Touili, Byron Cook & Paul Jackson, editors: *Computer Aided Verification*, *Lecture Notes in Computer Science* 6174, Springer Berlin Heidelberg, pp. 365–379, doi:10.1007/978-3-642-14295-6_33.

[5] Marcus Gelderie (2012): *Strategy Machines and Their Complexity*. In Branislav Rovan, Vladimiro Sassone & Peter Widmayer, editors: *Mathematical Foundations of Computer Science 2012*, *Lecture Notes in Computer Science* 7464, Springer Berlin Heidelberg, pp. 431–442, doi:10.1007/978-3-642-32589-2_39.

[6] Marcus Gelderie & Michael Holtmann (2011): *Memory Reduction via Delayed Simulation*. In Johannes Reich & Bernd Finkbeiner, editors: *iWIGP*, *EPTCS* 50, pp. 46–60, doi:10.4204/EPTCS.50.4.

[7] Michael Holtmann, Lukasz Kaiser & Wolfgang Thomas (2010): *Degrees of Lookahead in Regular Infinite Games*. In Luke Ong, editor: *Foundations of Software Science and Computational Structures*, *Lecture Notes in Computer Science* 6014, Springer Berlin Heidelberg, pp. 252–266, doi:10.1007/978-3-642-12032-9_18.

[8] Orna Kupferman & Moshe Y. Vardi (1999): *Church's Problem Revisited*. *The Bulletin of Symbolic Logic* 5(2), pp. 245–263, doi:10.2307/421091.

[9] Yoad Lustig & Moshe Y. Vardi (2009): *Synthesis from Component Libraries*. In Luca Alfaro, editor: *Foundations of Software Science and Computational Structures*, *Lecture Notes in Computer Science* 5504, Springer Berlin Heidelberg, pp. 395–409, doi:10.1007/978-3-642-00596-1_28.

[10] Parthasarathy Madhusudan (2011): *Synthesizing Reactive Programs*. In Marc Bezem, editor: *Computer Science Logic (CSL'11) - 25th International Workshop/20th Annual Conference of the EACSL*, *Leibniz International Proceedings in Informatics (LIPIcs)* 12, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp. 428–442, doi:10.4230/LIPIcs.CSL.2011.428.

[11] Amir Pnueli & Roni Rosner (1989): *On the Synthesis of a Reactive Module*. In: *POPL*, pp. 179–190. Available at `http://doi.acm.org/10.1145/75277.75293`.

[12] Michael Oser Rabin (1972): *Automata on Infinite Objects and Church's Problem*. American Mathematical Society, Boston, MA, USA.

[13] Roni Rosner (1992): *Modular Synthesis of Reactive Systems*. Ph.D. thesis, Weizmann Institute of Science.

[14] Sven Schewe & Bernd Finkbeiner (2007): *Bounded Synthesis*. In Kedar S. Namjoshi, Tomohiro Yoneda, Teruo Higashino & Yoshio Okamura, editors: *Automated Technology for Verification and Analysis*, *Lecture Notes in Computer Science* 4762, Springer Berlin Heidelberg, pp. 474–488, doi:10.1007/978-3-540-75596-8_33.

[15] Wolfgang Thomas (1997): *Languages, Automata, and Logic*. In Grzegorz Rozenberg & Arto Salomaa, editors: *Handbook of Formal Languages*, Springer Berlin Heidelberg, pp. 389–455, doi:10.1007/978-3-642-59126-6_7.

The Complexity of Synthesizing Uniform Strategies

Bastien Maubert
IRISA
Université de Rennes 1
Rennes, France
bastien.maubert@irisa.fr

Sophie Pinchinat
IRISA
Université de Rennes 1
Rennes, France
sophie.pinchinat@irisa.fr

Laura Bozzelli
Facultad de Informática
UPM
Madrid, Spain
laura.bozzelli@fi.upm.es

We investigate *uniformity properties* of strategies. These properties involve sets of plays in order to express useful constraints on strategies that are not μ-calculus definable. Typically, we can state that a strategy is observation-based. We propose a formal language to specify uniformity properties, interpreted over two-player turn-based arenas equipped with a binary relation between plays. This way, we capture *e.g.* games with winning conditions expressible in epistemic temporal logic, whose underlying equivalence relation between plays reflects the observational capabilities of agents (for example, synchronous perfect recall). Our framework naturally generalizes many other situations from the literature. We establish that the problem of synthesizing strategies under uniformity constraints based on regular binary relations between plays is non-elementary complete.

1 Introduction

In extensive infinite duration games, the arena is represented as a graph whose vertices denote positions of players and whose paths denote plays. In this context, a strategy of a player is a mapping prescribing to this player which next position to select provided she has to make a choice at this current point of the play. As mathematical objects, strategies can be seen as infinite trees obtained by pruning the infinite unfolding of the arena according to the selection prescribed by this strategy; outcomes of a strategy are therefore the branches of the trees.

Strategies of players are not arbitrary in general, since players aim at achieving some objectives. Infinite-duration game models have been intensively studied for their applications in computer science [3] and logic [13]. First, infinite-duration games provide a natural abstraction of computing systems' non-terminating interaction [2] (think of a communication protocol between a printer and its users, or control systems). Second, infinite-duration games naturally occur as a tool to handle logical systems for the specification of non-terminating behaviors, such as for the propositional μ-calculus [10], leading to a powerful theory of automata, logics and infinite games [13] and to the development of algorithms for the automatic verification ("model-checking") and synthesis of hardware and software systems. In both cases, outcomes of strategies are submitted to ω-regular conditions representing some desirable property of a system.

Additionally, the cross fertilization of multi-agent systems and distributed systems theories has led to equip logical systems with additional modalities, such as epistemic ones, to capture uncertainty [27, 21, 11, 24, 20, 15], and more recently, these logical systems have been adapted to game models in order to reason about knowledge, time and strategies [17, 19, 9]. The whole picture then becomes intricate, mainly because time and knowledge are essentially orthogonal, yielding a complex theoretical universe to reason about. In order to understand to which extent knowledge and time are orthogonal, the angle of view where strategies are infinite trees is helpful: Time is about the *vertical* dimension of the trees as it relates to the ordering of encountered positions along plays (branches) and to the branching in the tree.

F. Mogavero, A. Murano, and M.Y. Vardi (Eds.):
1st Workshop on Strategic Reasoning 2013 (SR'13)
EPTCS 112, 2013, pp. 115–122, doi:10.4204/EPTCS.112.17

On the contrary, Knowledge is about the *horizontal* dimension, as it relates plays carrying, e.g., the same information.

As far as we know, this horizontal dimension, although extensively studied when interpreted as knowledge or observation [4, 17, 19, 8, 1, 9], has not been addressed in its generality. In this paper, we aim at providing a unified setting to handle it. We introduce the generic notion of *uniformity properties* and associated so-called *uniform strategies* (those satisfying uniformity properties). Some notions of "uniform" strategies have already been used, e.g., in the setting of strategic logics [29, 5, 19] and in the evaluation game of Dependence Logic [28], which both fall into the general framework we present here.

We use a simple framework with two-player turn-based arenas and where information lies in positions, but the approach can be extended to other settings. Additionally, although uniformity properties can be described in a set-theoretic framework, we propose the logical formalism $\mathbb{R}$LTL which can be exploited to address fundamental automated techniques such as the verification of uniformity properties and the synthesis of uniform strategies – arbitrary uniformity properties are in general hopeless for automation. The formalism we use combines the Linear-time Temporal Logic LTL [12] and a new modality $\mathbb{R}$ (for "for all related plays"), the semantics of which is given by a binary relation between plays. Modality $\mathbb{R}$ generalizes the knowledge operator "K" of [15] for the epistemic relations of agents in Interpreted Systems. The semantic binary relations between plays are very little constrained: they are not necessarily equivalences, to capture, *e.g.* plausibility (pre)orders one finds in doxastic logic [16], neither are they knowledge-based, to capture particular strategies in games where epistemic aspects are irrelevant. Formulas of the logic are interpreted over outcomes of a strategy. The $\mathbb{R}$ modality allows to universally quantify over all plays that are in relation with the current play. Distinguishing between the universal quantification over all plays in the game and the universal quantification over all the outcomes in the strategy tree yields two kinds of uniform strategies: the *fully-uniform strategies* and the *strictly-uniform strategies*.

As extensively demonstrated in [22], uniform properties turn out to be many in the literature: they occur in games with imperfect information, in games with opacity conditions and more generally with epistemic conditions, as non-interference properties of computing systems, as diagnosability of discrete-event systems, in the game semantics of Dependence Logic.

We investigate the automated synthesis of fully-uniform strategies, for the case of finite arenas and binary relations between plays that are rational in the sense of [6]. Incidentally, all binary relations that are involved in the relevant literature seem to follow this restriction. In this context, two problems can be addressed: the *fully-uniform strategy problem* and the *strictly-uniform strategy problem*, which essentially can be formulated as "given a finite arena, a finite state transducer describing a binary relation between plays, and a formula expressing a uniformity property, does there exist a fully-uniform (resp. strictly-uniform) strategy for Player 1?". From [22], the fully-uniform strategy problem is decidable but non-elementary – since then we have established that it is non-elementary hard. The algorithm involves an iterated non-trivial powerset construction from the arena and the finite state transducer which enables to eliminate innermost $\mathbb{R}$ modalities. Hence, the required number of iterations matches the maximum number of nested $\mathbb{R}$ modalities of the formula expressing the uniformity property. As expected, each powerset construction is computed in exponential time. This procedure amounts to solving an ultimate LTL game, for which a strategy can be synthesized [25] and traced back as a solution in the original problem. The decidability of the strictly-uniform strategy problem is an open question.

The rest of the paper is organized in five sections. In Section 2, we present the standard material two-player turn-based arenas. We set up the framework and define uniform strategies in Section 3, and we illustrate the notion with two examples in Section 4. Finally in Section 5, we give tight complexity bounds for the fully-uniform strategy problem, and we discuss future work in Section 6.

2 Preliminaries

We consider two-player turn-based games that are played on graphs with vertices labelled with propositions. These propositions represent the relevant information for the uniformity properties one wants to state. From now on and for the rest of the paper, we let AP be an infinite set of *atomic propositions*.

An *arena* is a structure $\mathcal{G} = (V, E, v_0, \ell)$ where $V = V_1 \uplus V_2$ is the set of *positions*, partitioned between positions of Player 1 (V_1) and those of Player 2 (V_2), $E \subseteq (V_1 \times V_2) \cup (V_2 \times V_1)$ is the set of *edges*, $v_0 \in V$ is the *initial position* and $\ell : V \to \mathscr{P}(AP)$ is a *valuation function*, mapping each position to the finite set of atomic propositions that hold in this position. $Plays_*$ and $Plays_\omega$ are, respectively, the set of finite and infinite *plays*. For an infinite play $\pi = v_0 v_1 \dots$ and $i \in \mathbb{N}$, $\pi[i] := v_i$ and $\pi[0,i] := v_0 \dots v_i$. For a finite play $\rho = v_0 v_1 \dots v_n$, $\text{last}(\rho) = v_n$.

A *strategy* for Player 1 is a partial function $\sigma : Plays_* \to V$ that maps a finite play ending in V_1 to the next position to play. Let σ be a strategy for Player 1. We say that a play $\pi \in Plays_\omega$ is *induced by* σ if for all $i \geq 0$ such that $\pi[i] \in V_1$, $\pi[i+1] = \sigma(\pi[0,i])$, and the *outcome of* σ, noted $\text{Out}(\sigma) \subseteq Plays_\omega$, is the set of all infinite plays that are induced by σ. Definitions are similar for Player 2's strategies.

3 Uniform strategies

We define the formal language $\mathbb{R}\mathsf{LTL}$ to specify uniformity properties. This language enables to express properties of the dynamics of plays, and resembles the Linear Temporal Logic (LTL) [12]. However, while LTL formulas are evaluated on individual plays (paths), we want here to express properties on "bundles" of plays. To this aim, we equip arenas with a binary relation between finite plays, and we enrich the logic with a modality $\mathbb{R}$ that quantifies over related plays, the intended meaning of "$\mathbb{R}\varphi$ holds in ρ" being "φ holds in every play related to ρ".

The syntax of $\mathbb{R}\mathsf{LTL}$ is similar to that of linear temporal logic with knowledge [15]. However, we use $\mathbb{R}$ instead of the usual knowledge operator K to emphasize that it need not be interpreted in terms of knowledge in general, but merely as a way to state properties of bundles of plays. The syntax is:

$$\varphi, \psi ::= p \mid \neg\varphi \mid \varphi \wedge \psi \mid \bigcirc\varphi \mid \varphi \mathbf{U} \psi \mid \mathbb{R}\varphi \qquad p \in AP$$

Consider an arena $\mathcal{G} = (V, E, v_0, \ell)$ and a rational relation $\rightsquigarrow \subseteq Plays_* \times Plays_*$. A formula φ of $\mathbb{R}\mathsf{LTL}$ is evaluated at some point $i \in \mathbb{N}$ of an infinite play $\pi \in Plays_\omega$, within a *universe* $\Pi \subseteq Plays_\omega$. The semantics is given by induction over formulas.

$\Pi, \pi, i \models p$ if $p \in \ell(\pi[i])$
$\Pi, \pi, i \models \neg\varphi$ if $\Pi, \pi, i \not\models \varphi$
$\Pi, \pi, i \models \varphi \wedge \psi$ if $\Pi, \pi, i \models \varphi$ and $\Pi, \pi, i \models \psi$
$\Pi, \pi, i \models \bigcirc\varphi$ if $\Pi, \pi, i+1 \models \varphi$
$\Pi, \pi, i \models \varphi \mathbf{U} \psi$ if there is $j \geq i$ such that $\Pi, \pi, j \models \psi$ and for all $i \leq k < j$, $\Pi, \pi, k \models \varphi$
$\Pi, \pi, i \models \mathbb{R}\varphi$ if for all $\pi' \in \Pi$, $j \in \mathbb{N}$ such that $\pi[0,i] \rightsquigarrow \pi'[0,j]$, $\Pi, \pi', j \models \varphi$

From this semantics, we derive two notions of uniform strategies, which differ only in the universe the $\mathbb{R}$ modality quantifies over: $\text{Out}(\sigma)$ or $Plays_\omega$ (with the latter, related plays not induced by the strategy also count). The motivation for these two definitions is clear from [22] where many examples from the literature are given.

Definition 1 *Let $\mathcal{G}$ be an arena, $\rightsquigarrow$ be a rational relation and φ be an $\mathbb{R}$LTL formula. A strategy σ is:*
$(\rightsquigarrow, \varphi)$-strictly-uniform *if for all $\pi \in Out(\sigma)$, $Out(\sigma), \pi, 0 \models \varphi$,*
$(\rightsquigarrow, \varphi)$-fully-uniform *if for all $\pi \in Out(\sigma)$, $Plays_\omega, \pi, 0 \models \varphi$.*

4 Concrete examples

In this section we illustrate our notions of strictly and fully uniform strategies defined in the previous section with the examples of observation-based strategies in games with imperfect information, and games with opacity condition.

4.1 Observation-based strategies

Games with imperfect information, in general, are games in which some of the players do not know exactly what is the current position of the game. Poker is an example of imperfect-information game: one does not know which cards her opponents have in hands. One important aspect of imperfect-information games is that not every strategy is "playable". Indeed, a player cannot plan to play differently in situations that she is unable to distinguish. This is why players are required to use strategies that select moves uniformly over observationally equivalent situations. This kind of strategies is sometimes called *uniform strategies* in the community of strategic logics ([29, 5, 19]), or *observation-based strategies* in the community of computer-science oriented game theory ([8]). In fact, all the additional complexity of solving imperfect-information games, compared to perfect-information ones, lies in this constraint put on strategies.

We show that the notion of observation-based strategy, and hence the essence of games with imperfect information, can be easily embedded in our notion of uniform strategy. In two-player imperfect-information games as studied for example in [26, 8, 7], Player 1 only partially observes the positions of the game, such that some positions are indistinguishable to her, while Player 2 has perfect information (the asymmetry is due to the focus being on the existence of strategies for Player 1). Arenas are labelled directed graphs together with a finite set of *actions Act*, and in each round, if the position is a node v, Player 1 chooses an available action a, and Player 2 chooses a next position v' reachable from v through an a-labelled edge.

We equivalently define this framework in a manner that fits our setting by putting Player 1's actions inside the positions. We have two kinds of positions, of the form v and of the form (v,a). In a position v, when she chooses an action a, Player 1 actually moves to position (v,a), then Player 2 moves from (v,a) to some v'. So an imperfect-information game arena is a structure $\mathscr{G}_{imp} = (\mathscr{G}, \sim)$ where $\mathscr{G} = (V, E, v_0, \ell)$ is a two-player game arena with positions in V_1 of the form v and positions in V_2 of the form (v,a). We require that $vE(v',a)$ implies $v = v'$, and $v_0 \in V_1$. For a position $(v,a) \in V_2$, we note $(v,a).act := a$. We assume that $p_1 \in AP$, and for every action a in Act, $p_a \in AP$. p_1 holds in positions belonging to Player 1, and p_a holds in positions of Player 2 where the last action chosen by Player 1 is a: $\ell(v) = \{p_1\}$ for $v \in V_1$, $\ell(v,a) = \{p_a\}$ for $(v,a) \in V_2$. Finally, $\sim \subseteq V_1^2$ is an observational equivalence relation on positions, that relates positions indistinguishable for Player 1. We define its extension $\simeq$ to finite plays: $v_0(v_0,a_1)v_1 \dots (v_{n-1},a_n)v_n \simeq v_0(v_0,a'_1)v'_1 \dots (v'_{n-1},a'_n)v'_n$ if for all $i > 0$, $v_i \sim v'_i$ and $a_i = a'_i$.

We add the classic requirement that the same actions must be available in indistinguishable positions: for all $v, v' \in V_1$, if $v \sim v'$ then $vE(v,a)$ if, and only if, $v'E(v',a)$. In other words, if Player 1 has different options, she can distinguish the positions.

Definition 2 *A strategy σ for Player* 1 *is* observation-based *if for all $\rho, \rho' \in v_0(V_2V_1)^*$, $\rho \approx \rho'$ implies $\sigma(\rho).act = \sigma(\rho').act$.*

We define the formula

$$\texttt{SameAct} := \mathbf{G}(p_1 \rightarrow \bigvee_{a \in Act} \mathbb{R}\bigcirc p_a)$$

which, informally, expresses that whenever it is Player 1's turn to play, there is an action a that is played in every equivalent finite play.

Proposition 1 *A strategy σ for Player 1 is observation-based iff it is $(\approx, \texttt{SameAct})$-strictly-uniform.*

Here we have to make use of the notion of strict uniformity, and not the full uniformity. Indeed, after a finite play $\pi[0,i]$ ending in V_1, we want to enforce that in all equivalent prefixes of infinite plays *that conform to the strategy considered*, Player 1 plays the same action. It would obviously make no sense to enforce the same on equivalent prefixes of every possible play in the game, which encompass all possible behaviours of Player 1.

Notice that in order to embed the case of players with different memory abilities, *e.g.* imperfect-recall, one would just have to replace $\approx$ with the appropriate relation.

For the moment we have not mentioned any winning condition. For a strategy, being $(\approx, \texttt{SameAct})$-strictly-uniform only characterizes that it is "playable" for a player with imperfect information, but it does not characterize the outcome of this strategy. However, if one considers a game with imperfect information in which the winning condition for Player 1 is an LTL formula φ, then the set of $(\approx, \texttt{SameAct} \wedge \varphi)$-strictly-uniform strategy is exactly the set of winning observation-based strategy.

When talking about knowledge and strategic abilities, the question of *objective* vs *subjective* ability should be raised (see [18]). The difference is basically whether a strategy is defined only on "concrete" plays, starting from the initial position, or if it has to be defined on all "plays" starting from any position the player confuses with the initial one. In the setting presented here, the initial position is part of the description of the arena, hence players are assumed to know it and all plays considered start from this position. But in order to model in this setting the case of Player 1 not knowing the initial position, one could add a fresh artificial initial position v_0', from which no matter the action Player 1 chooses, Player 2 can move to any position that Player 1 confuses with v_0. Then, for a winning condition $\varphi \in \mathsf{LTL}$, the existence of an observation-based winning strategy for Player 1 from v_0 (resp. v_0') would denote objective (resp. subjective) ability to enforce φ.

4.2 Games with opacity condition

Games with opacity condition, studied in [23], are based on two-player imperfect-information arenas, with Alice having perfect information as opposed to Bob who partially observes positions. In such games, some positions are "secret" as they reveal a critical information that Bob aims at discovering. We are interested in Alice's ability to prevent Bob from "knowing" the secret, in the epistemic sense.

More formally, assume that a proposition $p_S \in AP$ represents the secret. Let $\mathscr{G}_{inf} = (\mathscr{G}, \sim)$ be an imperfect-information arena as described in Section 4.1, with a distinguished set of positions $S \subseteq V_1$ that denotes the secret. Bob is Player 1 as he has imperfect information, and Alice is Player 2. Letting $\mathscr{G} = (V, E, v_I, \ell)$, we require that $\ell^{-1}(\{p_S\}) = S$ (positions labeled by p_S are exactly positions $v \in S$). For a finite play ρ with $last(\rho) \in V_1$, Bob's *information set* or *knowledge* after ρ is $I(\rho) := \{last(\rho') \mid \rho' \in Plays_*, \rho \approx \rho'\}$. It is the set of all the positions he considers possible after observing ρ. An infinite play is winning for Bob if there exists a finite prefix ρ of this play whose information set is contained in S, *i.e.* $I(\rho) \subseteq S$, otherwise Alice wins. It can easily be shown that:

Proposition 2 *A strategy σ for Alice is winning if, and only if, σ is $(\approx, \mathbf{G}\neg K p_S)$-fully-uniform.*

Here we are interested in Alice's strategies and Bob's knowledge. Since Bob only partially observes what Alice is playing, some plays that are not brought about by Alice's strategy are considered possible by Bob. Full uniformity is therefore the right notion to capture correctly Bob's knowledge.

Here again, to model different memory and observational abilities of Bob, one can use the appropriate binary relation, provided it is rational. Also, notice that though we chose to illustrate our framework with opacity aspects, any winning condition that is expressible by a formula of the epistemic linear temporal logic with one knowledge operator would fit in our setting.

5 Synthesizing fully-uniform strategies

In this section, we investigate the complexity of synthesizing a fully-uniform strategy. We first consider the associated decision problem, called here the *fully-uniform strategy problem*: given a uniform property $\varphi \in \mathbb{R}\mathsf{LTL}$, a finite arena $\mathscr{G} = (V, E, v_0, \ell)$, and a finite state transducer T over alphabet V representing a rational binary relation between plays (see [6]), does there exist a $([T], \varphi)$-fully-uniform strategy in $\mathscr{G}$, where $[T]$ is the binary relation denoted by T.

Definition 3 *For a formula $\varphi \in \mathbb{R}LTL$, the $\mathbb{R}$-*depth *of φ, written $d_{\mathbb{R}}(\varphi)$, is the maximum number of nested $\mathbb{R}$ modalities in φ. For each $k \in \mathbb{N}$, we let $\mathbb{R}LTL_k := \{\varphi \in \mathbb{R}LTL \mid d_{\mathbb{R}}(\varphi) = k\}$.*

Theorem 3 *The fully-uniform strategy problem for formulas ranging over $\bigcup_{k \leq n} \mathbb{R}LTL_k$ is n-*EXPTIME*-complete for $n > 2$, and* 2EXPTIME*-complete for $n \leq 2$.*

The proof for the upper bounds in Theorem 3 can be found in [22], in which we devise a decision procedure based on a powerset construction which simulates the execution of the transducer along plays in the arena, enabling the computation of information sets. Dealing with information sets enables us to perform $\mathbb{R}$-modalities elimination, yielding a reduction of the initial problem to solving some LTL game. The procedure is however non-elementary since it requires one powerset construction per nesting of $\mathbb{R}$-modalities. The proof for the matching lower bounds is a direct reduction from the word problem for $\exp[n]$-space bounded alternating Turing Machines, which is $(n+1)$-EXPTIME complete. Due to lack of space, it is omitted here.

Corollary 4 *The fully-uniform strategy problem is non-elementary complete.*

Regarding the synthesis problem, the procedure of [25] for solving the terminal LTL game in the decision procedure of Theorem 3 is an effective construction of a winning strategy when it exists. This strategy provides a fully-uniform strategy of the initial game, by means of a transducer mapping plays of the initial game to plays in the terminal game. This transducer itself is straightforwardly built from the arena of the last game itself.

6 Discussion

We are currently working on sufficient conditions on the binary relation between plays to render the fully-uniform strategy synthesis problem elementary. It appears that being an equivalence relation is not enough, but if moreover the relation verifies a weak form of *no learning* property (see [14]), the problem seems to be elementary. Concerning the strictly-uniform strategy problem, we conjecture undecidability in general, but we are investigating interesting subclasses of rational relations that make the problem decidable.

It would then be interesting to extend the language to the case of n modalities $\mathbb{R}_i$ with n relations $\rightsquigarrow_i$. Also, the difference between the fully-uniform semantics and the strictly-uniform one could be at the level of modalities instead of the decision problems level. In Section 4.1 we have seen that uniformity

properties can represent *uniformity constraints* on the set of elegible strategies, and in Section 4.2 we have seen how they can represent *epistemic winning conditions*. However, while some properties require strict uniformity, others require full uniformity. Allowing to use both kinds of modalities in a formula would enable, for example, to express that a strategy must both be winning for some condition on the opponent's knowledge (with a fully-uniform modality, see Section 4.2), and to be observation based for the player considered (with a strictly-uniform modality). A formula of the following kind could be used for a variant of games with opacity condition where Alice would also have imperfect information (note that the arenas should be modified, and we assume that p_2 would mark positions where Alice has to choose an action):

$$\varphi := \underbrace{\mathbf{G}(p_2 \rightarrow \bigvee_{a \in Act} I\!R_{Alice}^{strictly} \bigcirc p_a)}_{\text{Observation-based constraint}} \quad \wedge \quad \underbrace{\mathbf{G} \neg I\!R_{Bob}^{fully} p_S}_{\text{Winning condition}}$$

In a next step, we would like to consider how our framework adapts if we take as base language the one of Alternating-time Temporal Logic [2] instead of LTL, so as to obtain an Alternating-time Temporal Epistemic Logic-like language. It would enable us to express the existence of uniform strategies directly in the logic, and not only at the level of decision problems as it is the case for now. This step will require to pass from the two-player turn-based arenas considered so far to multiplayer concurrent game structures, that are ATL models, but the definitions should adapt without difficulties. However we should be cautious in generalizing these notions as undecidability will easily be attained.

References

[1] R. Alur, P. Černỳ & S. Chaudhuri (2007): *Model checking on trees with path equivalences*. *Tools and Algorithms for the Construction and Analysis of Systems*, pp. 664–678, doi:10.1007/978-3-540-71209-1_51.

[2] R. Alur, T.A. Henzinger & O. Kupferman (2002): *Alternating-time temporal logic*. *Journal of the ACM (JACM)* 49(5), pp. 672–713, doi:10.1145/585265.585270.

[3] K.R. Apt & E. Grädel (2011): *Lectures in Game Theory for Computer Scientists*. Cambridge University Press, doi:10.1017/CBO9780511973468.

[4] A. Arnold, A. Vincent & I. Walukiewicz (2003): *Games for synthesis of controllers with partial observation*. *Theoretical Computer Science* 1(303), pp. 7–34, doi:10.1016/S0304-3975(02)00442-5. Available at http://www.labri.fr/Perso/~igw/Papers/igw-synthesis.ps.

[5] J. Benthem (2005): *The Epistemic Logic of IF Games*. *The Philosophy of Jaakko Hintikka* 30.

[6] J. Berstel (1979): *Transductions and context-free languages*. 4, Teubner Stuttgart.

[7] Dietmar Berwanger & Laurent Doyen (2008): *On the Power of Imperfect Information*. In Ramesh Hariharan, Madhavan Mukund & V. Vinay, editors: *FSTTCS*, *LIPIcs* 2, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, pp. 73–82. Available at http://dx.doi.org/10.4230/LIPIcs.FSTTCS.2008.1742.

[8] K. Chatterjee, L. Doyen, T. Henzinger & J.F. Raskin (2006): *Algorithms for omega-regular games with imperfect information*. In: *Computer Science Logic*, Springer, pp. 287–302, doi:10.1007/11874683_19.

[9] C. Dima, C. Enea & D. Guelev (2010): *Model-Checking an Alternating-time Temporal Logic with Knowledge, Imperfect Information, Perfect Recall and Communicating Coalitions*. *Electronic Proceedings in Theoretical Computer Science* 25, doi:10.4204.

[10] E. Allen Emerson & Charanjit S. Jutla (1991): *Tree Automata, Mu-Calculus and Determinacy (Extended Abstract)*. In: *FOCS*, IEEE Computer Society, pp. 368–377. Available at http://doi.ieeecomputersociety.org/10.1109/SFCS.1991.185392.

[11] R. Fagin, J.Y. Halpern & M.Y. Vardi (1991): *A model-theoretic analysis of knowledge*. *Journal of the ACM (JACM)* 38(2), pp. 382–428, doi:10.1145/103516.128680.

[12] Dov M. Gabbay, Amir Pnueli, Saharon Shelah & Jonathan Stavi (1980): *On the Temporal Basis of Fairness*. In Paul W. Abrahams, Richard J. Lipton & Stephen R. Bourne, editors: *POPL*, ACM Press, pp. 163–173. Available at http://doi.acm.org/10.1145/567446.567462.

[13] E. Grädel, W. Thomas & T. Wilke, editors (2002): *Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*. *Lecture Notes in Computer Science* 2500, Springer.

[14] Joseph Y. Halpern, Ron van der Meyden & Moshe Y. Vardi (2004): *Complete Axiomatizations for Reasoning about Knowledge and Time*. *SIAM J. Comput.* 33(3), pp. 674–703. Available at http://dx.doi.org/10.1137/S0097539797320906.

[15] J.Y. Halpern & M.Y. Vardi (1989): *The complexity of reasoning about knowledge and time. 1. Lower bounds*. *Journal of Computer and System Sciences* 38(1), pp. 195–237, doi:10.1145/12130.12161.

[16] J. Hintikka (1962): *Knowledge and belief*. 13, Cornell University Press Ithaca.

[17] W. van der Hoek & M. Wooldridge (2003): *Cooperation, knowledge, and time: Alternating-time temporal epistemic logic and its applications*. *Studia Logica* 75(1), pp. 125–157, doi:10.1023/A:1026185103185.

[18] W. Jamroga & N. Bulling (2011): *Comparing variants of strategic ability*. In: *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume One*, AAAI Press, pp. 252–257, doi:10.1023/A:1026171312755.

[19] Wojciech Jamroga & Wiebe van der Hoek (2004): *Agents that Know How to Play*. *Fundamenta Informaticae* 63(2-3), pp. 185–219. Available at http://iospress.metapress.com/content/xh738axb47d8rchf/.

[20] Richard E. Ladner & John H. Reif (1986): *The Logic of Distributed Protocols*. In Joseph Y. Halpern, editor: *TARK*, Morgan Kaufmann, pp. 207–222.

[21] D. Lehmann (1984): *Knowledge, common knowledge and related puzzles (Extended Summary)*. In: *Proceedings of the third annual ACM symposium on Principles of distributed computing*, ACM, pp. 62–67, doi:10.1145/800222.806736.

[22] Bastien Maubert & Sophie Pinchinat (2012): *Uniform Strategies*. Rapport de recherche RR-8144, INRIA. Available at http://hal.inria.fr/hal-00760370.

[23] Bastien Maubert, Sophie Pinchinat & Laura Bozzelli (2011): *Opacity Issues in Games with Imperfect Information*. In Giovanna D'Agostino & Salvatore La Torre, editors: *GandALF*, *EPTCS* 54, pp. 87–101, doi:10.4204/EPTCS.54.7.

[24] R. Parikh & R. Ramanujam (1985): *Distributed processes and the logic of knowledge*. *Logics of Programs*, pp. 256–268, doi:10.1007/3-540-15648-8_21.

[25] A. Pnueli & R. Rosner (1989): *On the Synthesis of an Asynchronous Reactive Module*. In: *Proc. 16th Int. Coll. on Automata, Languages and Programming, ICALP'89, Stresa, Italy, LNCS 372*, Springer-Verlag, pp. 652–671, doi:10.1007/BFb0035790.

[26] J.H. Reif (1984): *The complexity of two-player games of incomplete information*. *Journal of computer and system sciences* 29(2), pp. 274–301, doi:10.1016/0022-0000(84)90034-5.

[27] JM Sato (1977): *A study of Kripke style methods for some modal logic by Gentzen's sequential method*. Technical Report, Technical report, Publication Research Institute for Mathematical Science.

[28] J. Väänänen (2007): *Dependence Logic*.

[29] J. Van Benthem (2001): *Games in Dynamic-Epistemic Logic*. *Bulletin of Economic Research* 53(4), pp. 219–248, doi:10.1111/1467-8586.00133.