Probabilistic Alternating-Time Temporal Logic and Model Checking Algorithm

Taolue Chen *†
CWI, Department of Software Engineering,
P.O. Box 94079, 1090GB Amsterdam, The Netherlands

Abstract

Last decade witnesses an impressive development of embedded reactive systems, which motivates the research of open *systems, where multiple components interact with each other and their environment and these interactions decide the behavior of the system. A natural "common-denominator" model for open systems is the* concurrent game structure*, in which several players can concurrently decide on the behavior of the system.* Alternating-time temporal logic *(ATL), is a temporal logic geared towards the specification and verification of properties in open systems, which allows to reason on the existence of strategies for coalitions of players in order to enforce a given property.* Probabilistic systems*, i.e. system models in which transitions are equipped with random information, receive increasingly attention in recent years. In this paper, we propose to study the* open probabilistic system*. We extend the framework of ATL in the probabilistic setting, following the style of* probabilistic computation tree logic *(PCTL), and obtain two* probabilistic alternating-time temporal logics*, i.e. PATL and PATL*. They are interpreted over* probabilistic concurrent game structures*, which is a probabilistic extension of multi-player concurrent game structure. We develop model checking algorithms for both PATL and PATL*. A more expressive logic,* probabilistic game logic*, is also addressed.*

1 Introduction

Last decade witnesses an impressive development of embedded reactive systems, which calls for systematic design and verification methodologies that can cope with the complexity in the system design, analysis and implementation. Model checking [9] is a well-established technique for verifying whether a system (typically represented by some formal models, say automata) satisfies a given property. Usually, temporal logics have been used to specify these properties: *linear time* temporal logics (typically LTL, [26]) expresses properties on each single execution of the model, while *branching time* temporal logics (typically CTL, [8, 28]) handle the computation tree of the model. While the computer science community has concentrated on verification of systems, the control theory community has always had *synthesis* as their main objective, as opposed to analysis. This results in a new framework in the field of verification: control (and controller synthesis [29]), which was developed in the late 80's. Its goal is to build a controller that should prevent the (model of the) system from having unwanted behaviors.

In system verification, especially in the area of control, it turns out to be essential to distinguish between *open* and *closed* systems [14]. For a *closed* system, the behavior is completely determined by the state of the system while for an *open* system, the behavior is affected both by its internal state and by the ongoing interaction with its environment. Thus, while in a closed system all the nondeterministic choices are internal, and resolved by the system, in an open system there are also external nondeterministic choices, which are resolved by the environment [13]. In the literatures, while closed systems are naturally modelled as Kripke structures (or labelled transition systems), a natural "common-denominator" model for compositions of open systems is the *concurrent game structure* [1], in which several players concurrently decide on the behavior of the system. In particular, Kripke structures can be viewed as game structures with a single player which represents the player *system*. In this case, game structures degenerate to Kriple structures.

It is well recognized that the control problem is closely related to (multi-player) games: solving such a game amounts to computing a strategy (if it does exist) for a player so that she surely reaches a state where she is declared to be the winner. In this case, one often feels convenient to reason with strategies, for which a new flavor of temporal logics has been defined: *alternating-time* temporal logic [1]. This logic distinguishes itself in the sense that it allows to express, for instance, that

*The author is partially supported by Dutch Bsik project BRICKS (Basic Research in Informatics for Creating the Knowledge Society), 973 Program of China (2002CB312002), NSF of China (60233010, 60273034, 60403014), 863 Program of China (2005AA113160, 2004AA112090).

†Email address: chen@cwi.nl.

a coalition of players has a strategy in order to always reach a winning location, or to always avoid reaching a bad location. In [1], several alternating-time temporal logics are introduced to specify properties of game structures, including the CTL-like logic ATL, and the CTL*-like logic ATL*.

Most of the researchers agree that *uncertainty* is one of the major sources of system complexity. This motivates the investigation of *probabilistic systems*, where state transition encodes the probability of making a transition between states rather than just the existence of such transition. Recently, the specification and verification of the systems in this flavor have received a lot of attention and are subject of study of a rapidly growing research community, for which we invite the readers to [6] for a comprehensive exposition. In particular, from the modelling perspective, besides the basic stochastic models such as DTMC and CTMC, taking probabilities into account in addition to nondeterministic behavior also expands the possibilities of modelling certain aspects of the system under consideration; From the specification perspective, *probabilistic computation tree logic* (PCTL) was introduced by Hansson and Jonsson [20] and concentrates on probabilistic model checking w.r.t. probabilistic deterministic systems [20] or nondeterministic systems [4].

In this paper, we devote ourselves to coping with *open probabilistic system*. In a nutshell, we extend the framework of ATL to a probabilistic setting, following the style of PCTL, and obtain two *probabilistic alternating-time temporal logics*, i.e. PATL and PATL*. We propose the semantics over *probabilistic concurrent game structures*, which is a probabilistic extension of multi-player concurrent game structure [1]. Besides these contributions, the main focus of this paper is the model checking algorithms for both PATL and PATL*, where the basic idea is to reduce the model checking problem to the decidability of the theory of *real closed fields*. Under condition that *equality constraints* are not admitted (see below for explanation), we develop two model checking procedures for both PATL and PATL*. If we do not restrict the precision of the constraint, as for PATL, we still can handle it by some sophistic encodings while for PATL*, we conjecture it is undecidable. We also address a even more expressive logic, called *probabilistic game logic*, trading the lack of decidability of model checking. Some illuminating discussions are provided regarding these three logics. We assume familiarity of basic knowledge on temporal logic, including linear time and branching time temporal logic, ω-regular languages and their connections. For the former, the readers are referred to [9] whilst for the latter, one can consult to, say, [34]. We also assume that the readers have some basic notions on complexity theory and game theory, for which we refer the readers to [27] and [23] respectively.

This paper is set up as follows: In section 2, we first present the probabilistic concurrent game structure, then we introduce the probabilistic alternating-time temporal logic, including its syntax and semantics. In section 3, we present the model checking algorithms for PATL. In section 4, we propose PATL* and address its model checking algorithm, then we sketch a more expressive extension, i.e. the probabilistic game logic. We conclude our work in section 5 where some related works are also discussed in brief.

2 Probabilistic Alternating-time Temporal Logic

In this section, we shall introduce the *probabilistic alternating-time temporal logic* (PATL) and its semantics model, namely, *probabilistic concurrent game structure* (pCGS). Before starting our exposition, let us first fix some general notations. For a countable set X, a *probability distribution* on X is a function $\delta : X \mapsto [0, 1]$ such that $\sum_{x \in X} \delta(x) = 1$. We denote the set of probability distributions on X by $\mathcal{D}(X)$. For a probability distribution $\delta \in \mathcal{D}(X)$ we define $||\delta||$, the *support* of δ, as $||\delta|| = \{x \in X \mid \delta(x) > 0\}$. For all $k \in \mathbb{N}$, by $[k]$ we denote the set $\{1, \cdots, k\}$. For a finite set X, we denote the set of *finite words* over X by X^*, the set of *infinite words* (ω-words) over X by X^ω, and the set of nonempty finite words over X by X^+. We define X^∞ as $X^* \cup X^\omega$. For $\lambda \in X^\infty$ and $n \in \mathbb{N}$, we write $\lambda(n)$ for the n-th letter in λ, $\lambda[n]$ for the *suffix* starting at $\lambda(n)$, i.e. $\lambda[n] = \lambda(n), \lambda(n+1), \cdots$. In addition, we write $|\lambda|$ for the length of λ in case $\lambda \in X^*$ and $\lambda(\uparrow)$ for $\lambda(|\lambda| - 1)$, namely, the last element of λ.

2.1 Probabilistic Concurrent Game Structure

Definition 1 *[Probabilistic Concurrent Game Structure] A* probabilistic concurrent game structure *(pCGS) is a tuple $\mathcal{G} = \langle k, Q, \Pi, \pi, d, \delta \rangle$ with the following components:*

- *$k \in \mathbb{N}$ is the number of* players *(a.k.a.* agents*). We identify the players with the numbers in $[k]$;*
- *Q is a finite set of* states*;*
- *Π is a finite set of* atomic propositions *(a.k.a.* observations*);*
- *$\pi : Q \mapsto \wp(\Pi)$ is the* labelling function *(a.k.a.* observation mapping*). Namely, for each state $q \in Q$, π assigns to q a set $\pi(q) \subseteq \Pi$ of propositions that hold true at q.*
- *For each player $a \in [k]$ and each state $q \in Q$, a natural number $d_a(q) \geq 1$ of* moves *available at state to a. We identify the moves of player a at state q with the number $1, \cdots, d_a(q)$. For each state $q \in Q$, a* move vector *at q is a tuple $\langle j_1, \cdots, j_k \rangle$ such that $1 \leq j_a \leq d_a(q)$ for each*

player a. Given a state $q \in Q$, we write $D(q)$ for the set $\{1, \cdots, d_1(q)\} \times \cdots \times \{1, \cdots, d_k(q)\}$ of move vectors. The function D is called move function. *For each player $a \in [k]$, we write $C(a)$ for the set $\bigcup_{q \in Q}[d_a(q)]$ of all of the possible moves available for a. The function C is called* choice function.

- *For each state $p \in Q$ and each move vector $\langle j_1, \cdots j_k \rangle \in D(p)$, a* probabilistic transition function *δ, which gives the (conditional) probability $\delta(q \mid p, \langle j_1, \cdots, j_k \rangle)$ of a transition from state p to state q for all $q \in Q$ if each player $a \in [k]$ chooses move j_a. Note that we also write this probability as $\delta(p, \langle j_1, \cdots, j_k \rangle)(q)$, to emphasize that $\delta(p, \langle j_1, \cdots, j_k \rangle)$ is a probability distribution on Q.*

The number of states of the structure $\mathcal{G}$ is $n = |Q|$. The number of transitions of $\mathcal{G}$ is $m = \sum_{q \in Q} d_1(q) \times \cdots \times d_k(q) \times n$, namely, $m = |D| \cdot n$. Note that unlike in Kripke structure, the number of transitions is not bounded by n^2. For a fixed alphabet Π of propositions, the *size* of $\mathcal{G}$ is $\mathcal{O}(m)$.

For two states p and q, we say that q is a *successor* of p if there is a move vector $\langle j_1, \cdots j_k \rangle \in D(p)$ such that $q \in ||\delta(p, \langle j_1, \cdots j_k \rangle)||$. We write $p \rightarrow q$ if q is a successor of p.

The semantics of pCGS, intuitively, is as follows: At each state $q \in Q$, each player $a \in [k]$ chooses a move $1 \leq j_a \leq d_a(q)$ *simultaneously* and *independently*. The game then proceeds to the successor state q' with probability $\delta(q' \mid q, \langle j_1, \cdots, j_k \rangle)$, for every state $q' \in Q$. A *path* of $\mathcal{G}$ from a state q_0 is an infinite sequence $\lambda = q_0, q_1, \cdots$ of states such that for all positions $i \geq 0$, the state q_{i+1} is a successor of state q_i. We denote by Ω the set of all paths. Moreover, we associate the set

$$\Omega_q = \{\lambda = q_0, q_1, \cdots \mid q_0 = q \text{ and } q_i \rightarrow q_{i+1} \text{ for any } i \in \mathbb{N}\}$$

the paths starting at state q. Given a path $\lambda = q_0, q_1, \cdots$, we associate $\pi(\lambda)$ with λ as $\pi(q_0), \pi(q_1), \cdots$. By this way, we can use an LTL (linear time temporal logic) formula Ψ over Π to denote a set of paths $\Xi \subseteq \Omega$ by defining $\Xi = \{\lambda \mid \pi(\lambda) \models \Psi\}$.

Some natural model can be recovered from the general definition of pCGS. In particular, we distinguish the following special classes.

- A Markov Decision Process (MDP, [24]) is a probabilistic concurrent game structure with $k = 1$, namely, only one player.
- A concurrent game structure $\mathcal{G}$ is *deterministic*, if for all $q \in Q$ and all move vector $\langle j_1, \cdots j_k \rangle \in D(q)$, there is a $p \in Q$ such that $\delta(p \mid q, \langle j_1, \cdots, j_k \rangle) = 1$. In this case, a pCGS is simply the "concurrent game structure" in [1].

The interesting point with the framework of ATL, compared with standard temporal logics, is that it allows quantifications on *strategies* of (coalitions of) players. A *coalition* A is a subset of the set of players, namely, $A \subseteq [k]$. We write $\bar{A}$ for $[k] \setminus A$. In the sequel we introduce the notions of *strategy* and *outcome*, in order to define the semantics of PATL.

Definition 2 *Assume a probabilistic concurrent game structure $\mathcal{G} = \langle k, Q, \Pi, \pi, d, \delta \rangle$.*

- *A* strategy *for player $a \in [k]$ is a mapping $\sigma_a : Q^+ \mapsto \mathcal{D}(C(a))$ that associates with every nonempty finite sequence $\lambda \in Q^+$ of states, representing the past history of the game, a* probability distribution *$\sigma_a(\lambda)$ used to select the next move. Thus, the choice of the next move can be* history-dependent *and* randomized. *The strategy σ_a can prescribe only moves that are available to player a, i.e. for all sequences $\lambda \in Q^+$ and $q \in Q$, it is required that $||\sigma_a(\lambda(\uparrow))|| \subseteq [d_a(\lambda(\uparrow))]$. We denote by Υ_a the set of all strategies for player $a \in [k]$.*
- *Let $A \subseteq [k]$ be a coalition. A* move *for A from a state q is a family $(j_a)_{a \in A}$: one move for each player in A. We write $\mathsf{Mv}(q, A)$ to represent the set of all possible moves for A from q. Moreover, a* coalition strategy *σ_A for A is a family $(\sigma_a)_{a \in A}$, where $\sigma_a \in \Upsilon_a$. Given a move $c \in \mathsf{Mv}(q, A)$ and $\bar{c} \in \mathsf{Mv}(q, \bar{A})$, by abuse of notations, we write $\delta(q, c \cdot \bar{c})$ for the probabilistic transition function corresponding to these choices. Note that here $c \cdot \bar{c}$ is the* move vector *defined in Definition 1, namely, $c \cdot \bar{c} \in D(q)$.*
- *Let $A \subseteq [k]$ be a coalition and σ_A and $\sigma_{\bar{A}}$ be the coalition strategies for A and $\bar{A}$ respectively. Once the starting state q and the coalition strategies σ_A and $\sigma_{\bar{A}}$ for the two coalitions have been chosen, the game is reduced to an ordinary* stochastic process *(typically, a discrete time Markov chain). Hence, the probabilities of* events *are uniquely defined in a standard way, where an event $\mathcal{E} \subseteq \Omega$ is measurable set of paths. For an event $\mathcal{E} \subseteq \Omega$, we denote the probability that a path belongs to $\mathcal{E}$ when the game starts form q and the players follow the strategies σ_A and $\sigma_{\bar{A}}$ by $\mathbb{P}_q^{\sigma_A, \sigma_{\bar{A}}}(\mathcal{E})$. Similarly, for a measurable function f that associates a number in $\mathbb{R} \cup \{\infty\}$ with each path, we denote by $E_q^{\sigma_A, \sigma_{\bar{A}}}\{f\}$ the expected value of f when the game starts from q and the strategies σ_A and $\sigma_{\bar{A}}$ are followed. As usual, we denote by Θ_i the* random variable *over sample space Ω representing the i-th state of a path; formally, Θ_i is a variable that assumes value q_i on the path $q_0, q_1, \cdots$.*

  Given a coalition strategy $\sigma_A = (\sigma_a)_{a \in A}$, we define the set of possible outcomes *of σ_A from a state $q \in Q$ to be the set $\mathsf{Outcomes}(q, \sigma_A)$ of* probabilistic measure *that the players in A enforce when they follow the strategy σ_A, namely, for each $a \in A$, player*

a follows strategy σ_a. *We use* $\mathbb{O}_q^{\sigma_A}$ *to range over* $\mathsf{Outcomes}(q, \sigma_A)$.

2.2 Probabilistic Alternating-time Temporal Logic

The temporal logic PATL is defined w.r.t. a finite set Π of *atomic propositions* and a finite set $[k]$ of *players*.

Definition 3 *The syntax of* PATL *is defined by the following grammar:*

$$\begin{array}{rcl} \phi_s, \psi_s & ::= & \top \mid P \mid \neg\phi_s \mid \phi_s \wedge \psi_s \mid \langle\langle A\rangle\rangle[\psi_p]_{\bowtie r} \\ \psi_p & ::= & \mathcal{X}\phi_s \mid \phi_s \mathcal{U} \psi_s \mid \phi_s \mathcal{R} \psi_s \end{array}$$

where $P \in \Pi$, $A \subseteq [k]$, $\bowtie \in \{<, \leq, =, \geq, >\}$ *and* $r \in [0, 1]$.

In addition we also can define some standard abbreviations. For instance, $\bot$ *for* $\neg\top$, $\langle\langle A\rangle\rangle[\mathcal{F}\psi_s]_{\bowtie r}$ *for* $\langle\langle A\rangle\rangle[\top \mathcal{U} \psi_s]_{\bowtie r}$, $\langle\langle A\rangle\rangle[\mathcal{G}\psi_s]_{\bowtie r}$ *for* $\langle\langle A\rangle\rangle[\bot \mathcal{R} \psi_s]_{\bowtie r}$, *etc.*

The logic PATL is similar to the probabilistic branching-time temporal logic PCTL, only that $[\cdot]_{\bowtie r}$ quantifier is parameterized by sets of players. $\mathcal{X}$, $\mathcal{U}$, $\mathcal{R}$ are standard *temporal operators*. As usual, the formulae with a subscript s are *state* formulae while the ones with subscript p are *path* formulae. We will stick to these conventions throughout the rest of the paper.

Note that contrary to usual definitions of ATL, we include the "release" modality $\mathcal{R}$. The reason lies in that in [18], it is proved that modality $\langle\langle A\rangle\rangle\mathcal{R}$ can *not* be expressed using only $\langle\langle A\rangle\rangle\mathcal{U}$ and $\langle\langle A\rangle\rangle\mathcal{G}$ in ATL [1]. It is not difficult to see that their arguments hold true in the probabilistic setting. Moreover, due to the same reason, we include $=$ in the constraint $[\cdot]_{\bowtie r}$, in contrast to the case in PCTL.

Semantics. PATL formulae are interpreted over states of pCGS which has the same propositions and players. The labelling of the states of Q with propositions is used to evaluate the atomic formulae of PATL. The logical connectives $\neg$ and $\wedge$ have the standard interpretations. The most interesting case is the state-formula $\langle\langle A\rangle\rangle[\psi_p]_{\bowtie r}$. Intuitively, it holds in a state q of $\mathcal{G}$ iff there exists a *coalition strategy* for coalition A in order to enforce the probability of paths which satisfy the subformula ψ_p along all the outcomes to meat the constraint specified by $[\cdot]_{\bowtie r}$. To put it concretely, we can consider a game between a protagonist and an antagonist. The former represents coalition A and accordingly, the latter represents coalition $\bar{A}$ and they both follow their own coalition strategies. The protagonist wins the game if in the resulting stochastic process, the probability measure of pathes which satisfy the subformula ψ_p, read as a linear temporal formula whose outermost operator is $\mathcal{X}, \mathcal{U}$ or $\mathcal{R}$, fulfills the constraint $[\cdot]_{\bowtie r}$; otherwise, the antagonist wins. The PATL formula $[\psi_p]_{\bowtie r}$ is satisfied at the state q iff the protagonist has a winning strategy in this game.

We are now in a position to define the semantics formally. We write $\mathcal{G}, q \models \varphi$ to indicate that the state q satisfies the formula φ in the structure $\mathcal{G}$. When $\mathcal{G}$ is clear from the context, we omit it and write $q \models \varphi$. The *satisfaction relation* $\models$ is defined for all states q and paths λ of $\mathcal{G}$, inductively as follows.

$$\begin{array}{lcl} q \models true & & \\ q \models P & \Leftrightarrow & P \in \pi(q) \\ q \models \neg\phi_s & \Leftrightarrow & q \not\models \phi_s \\ q \models \phi_s \wedge \psi_s & \Leftrightarrow & q \models \phi_s \text{ and } q \models \psi_s \\ q \models \langle\langle A\rangle\rangle[\psi_p]_{\bowtie r} & \Leftrightarrow & \text{there is a coalition strategy } \sigma_A \text{ such that for any } \mathbb{O}_q^{\sigma_A} \in \mathsf{Outcomes}(q, \sigma_A), \mathbb{O}_q^{\sigma_A}(\{\lambda \in \Omega_q \mid \lambda \models \psi_p\}) \bowtie r \\ \lambda \models \mathcal{X}\phi_s & \Leftrightarrow & \lambda(1) \models \phi_s \\ \lambda \models \phi_s \mathcal{U} \psi_s & \Leftrightarrow & \exists i \in \mathbb{N}.\ \lambda(i) \models \psi_s \text{ and for any } 0 \leq j < i,\ \lambda(j) \models \phi_s \\ \lambda \models \phi_s \mathcal{R} \psi_s & \Leftrightarrow & \lambda \not\models \neg\phi_s \mathcal{U} \neg\psi_s \end{array}$$

3 Model Checking PATL

In this section, we aim at presenting algorithms for model checking PATL. As we have seen in the previous section, the semantics of PATL has a close relationship with two-player concurrent game, which is in turn the case for our model checking algorithm. To this end, first we introduce some notions and properties regarding the two-player concurrent game. Most of materials for this are taken from [11].

3.1 Two-player concurrent game

A two-player concurrent game structure is a special case of probabilistic concurrent game structure when $k = 2$ (recall that in this case, the two players are named after 1 and 2). In game theory, the notion of *winning condition* plays an important role. Here, let us consider the cast that it is expressed by LTL formulae. Namely, given an LTL winning condition Ψ, by abuse of notation we denote equally by Ψ the set of paths (ω-regular set) $\lambda \in \Omega$ that satisfy Ψ. It is well known that this set is *measurable* for any choice of strategies for the two players. Hence, the probability that a path satisfies Ψ starting from state $q \in Q$ under strategies σ_1, σ_2 of the two players can be denoted by $\mathbb{P}_q^{\sigma_1,\sigma_2}(\Psi)$. Given a state $q \in Q$ and a winning condition Ψ, one is often interested in finding the maximal probability with which player $a \in \{1, 2\}$ can ensure that Ψ holds from q. This probability is called the *value of the game* Ψ at q for player $a \in \{1, 2\}$. This value for player 1 is given by the function $\langle\langle 1\rangle\rangle\Psi : Q \mapsto [0, 1]$, defined as

$$\langle\langle 1\rangle\rangle\Psi(q) = \sup_{\sigma_1 \in \Upsilon_1} \inf_{\sigma_2 \in \Upsilon_2} \mathbb{P}_q^{\sigma_1,\sigma_2}(\Psi)$$

and the value for player 2 is given by the function $\langle\langle 2\rangle\rangle\Psi$, defined symmetrically.

It is also well known that concurrent games enjoy a *quantitative* version of determinacy [21], which implies that for all LTL conditions Ψ and all $q \in Q$, the following equation holds:

$$\langle\langle 1\rangle\rangle\Psi(q) + \langle\langle 2\rangle\rangle\neg\Psi(q) = 1$$

A strategy σ_1 is *optimal* if for all $q \in Q$,

$$\inf_{\sigma_2 \in \Upsilon_2} \mathbb{P}_q^{\sigma_1,\sigma_2}(\Psi) = \langle\langle 1\rangle\rangle\Psi(q)$$

A strategy σ_1 is ϵ-*optimal* if for all $q \in Q$,

$$\inf_{\sigma_2 \in \Upsilon_2} \mathbb{P}_q^{\sigma_1,\sigma_2}(\Psi) \geq \langle\langle 1\rangle\rangle\Psi(q) - \epsilon$$

The same notions for player 2 can be defined symmetrically. We note that the quantitative determinacy of concurrent games is equivalent to the existence of ϵ-optimal strategies for both players for all $\epsilon > 0$ at all states $q \in Q$.

In order to compute the value of the game, the following notion, namely, *predecessor operator* is essential. Let $\mathcal{F}$ be the space of all functions $Q \mapsto [0,1]$ that map states into the interval $[0,1]$. Given two functions $f, g \in \mathcal{F}$, we write $f > g$ (resp. $f \geq g$) if $f(q) > g(q)$ (resp. $f(q) \geq g(q)$) at all $q \in Q$, and we define $f \wedge g$ and $f \vee g$ by

$$\begin{array}{rcl}(f \wedge g)(q) & = & \min\{f(q), g(q)\} \\ (f \vee g)(q) & = & \max\{f(q), g(q)\}\end{array}$$

for all $q \in Q$. We denote by $\mathbf{0}$ and $\mathbf{1}$ the constant functions that map all states into 0 and 1. For all $f \in \mathcal{F}$, we denote by $\mathbf{1} - f$ the function defined by $(\mathbf{1} - f)(q) = 1 - f(q)$ for all $q \in Q$. Given a subset $S \subseteq Q$ of states, we denote by $[S]$ the *indication function* of S, defined by $[S](q) = 1$ if $q \in S$ and $[S](q) = 0$ o.w.

The *quantitative predecessor operators* Ppre_1, $\mathsf{Ppre}_2 : \mathcal{F} \mapsto \mathcal{F}$ are defined for every $f \in \mathcal{F}$ by

$$\mathsf{Ppre}_1(f) = \sup_{\sigma_1 \in \Upsilon_1} \inf_{\sigma_2 \in \Upsilon_2} E_s^{\sigma_1,\sigma_2}\{f(\Theta_1)\}$$

and symmetrically for Ppre_2. Recall that Θ_1 is a random variable that assumes value q_1 on the path $q_0, q_1, \cdots$. Intuitively, the value $\mathsf{Ppre}_i(f)$ is the maximum expectation for the next value of f that player $a \in \{1,2\}$ can achieve. Given $f \in \mathcal{F}$ and $a \in \{1,2\}$, the function $\mathsf{Ppre}_a(f)$ can be computed by solving the following *matrix game* at each state $q \in Q$:

$$\mathsf{Ppre}_1(f) = val_1[\sum_{p \in Q} f(p) \cdot \delta(p \mid q, j_1, j_2)]_{1 \leq j_1 \leq d_1(q), 1 \leq j_2 \leq d_2(q)}$$

where $val_1(A)$ denotes the value obtained by player 1 in the *matrix game* A. The existence of solutions to the above matrix games, and the existence of *optimal randomized strategies* for players 1 and 2, are guaranteed by the celebrated minmax theorem [22]. The matrix games may be solved using traditional linear programming algorithms.

Quantitative μ-calculus over game structure In order to write the solution of games w.r.t. ω-regular winning conditions, we introduce the *quantitative game μ-calculus*. The formulae of the *quantitative game μ-calculus* are generated by the grammar

$$\phi ::= [S] \mid x \mid \phi \vee \phi \mid \phi \wedge \phi \mid \mathsf{Ppre}_1(\phi) \mid \mathsf{Ppre}_2(\phi) \mid \mu x.\phi \mid \nu x.\phi$$

for atomic propositions $[S]$ where $S \subseteq Q$ and variables x from some fixed set X.

The atomic propositions of quantitative μ-calculus formulae correspond to subsets of states of the probabilistic concurrent game structure. As usual, a formula ϕ is *closed* if every variable x in ϕ occurs in the scope of a fixpoint quantifier μx or νx.

Let $\mathcal{E} : X \to \mathcal{F}$ be a *variable valuation* that associates a function $\mathcal{E}(x) \in \mathcal{F}$ with each variable $x \in X$. We write $\mathcal{E}[x \mapsto f]$ for the valuation that agrees with $\mathcal{E}$ on all variables, except that $x \in X$ is mapped to $f \in \mathcal{F}$. Given a valuation $\mathcal{E}$, every formula ϕ of quantitative game μ-calculus defines a function $[\![\phi]\!]_{\mathcal{E}} \in \mathcal{F}$:

$$\begin{array}{lcl}[\![f]\!]_{\mathcal{E}} & = & f \\ [\![x]\!]_{\mathcal{E}} & = & \mathcal{E}(x) \\ [\![\mathsf{Ppre}_1(\phi)]\!]_{\mathcal{E}} & = & \mathsf{Ppre}_1([\![\phi]\!]_{\mathcal{E}}) \\ [\![\mathsf{Ppre}_2(\phi)]\!]_{\mathcal{E}} & = & \mathsf{Ppre}_2([\![\phi]\!]_{\mathcal{E}}) \\ [\![\phi_1 \{\begin{smallmatrix}\wedge\\ \vee\end{smallmatrix}\} \phi_2]\!]_{\mathcal{E}} & = & [\![\phi_1]\!] \{\begin{smallmatrix}\wedge\\ \vee\end{smallmatrix}\} [\![\phi_2]\!] \\ [\![\{\begin{smallmatrix}\mu\\ \nu\end{smallmatrix}\} x.\phi]\!]_{\mathcal{E}} & = & \{\begin{smallmatrix}\sup\\ \inf\end{smallmatrix}\} \{f \mid f = [\![\phi]\!]_{\mathcal{E}[x \mapsto f]}\}\end{array}$$

The quantitative game μ-calculus defined above suffices for writing the solution formulae of games with ω-regular winning conditions, which are, in our case, expressed by LTL formulae. To put it more concretely, given a concurrent game with LTL winning condition Ψ, one can solve it by providing a quantitative game μ-calculus formula ϕ such that $\langle\langle 1\rangle\rangle\Psi = [\![\phi]\!]$. We stress that this fact essentially boils down to an algorithm, which is based on the observation that the value of a quantitative μ-calculus formula can be expressed as an elementary formula in the theory of *real closed fields*, and uses a decision procedure for the theory of reals with addition and multiplication [33]. In order to understand the algorithm in the sequel, here we include a simple introduction on this aspect.

An ordered field H is real-closed if no proper algebraic extension of H is ordered. We denote by $\mathbf{R}$ the real-closed field $(\mathbb{R}, +, \cdot, 0, 1, \leq)$ of the reals with addition and multiplication. An *atomic formula* a is an expression of the form $p > 0$ or $p = 0$ where p is a (possibly) multi-variate polynomial with integer coefficients. An *elementary formula* is constructed from atomic formulae by the grammar

$$\phi ::= a \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \exists x.\phi \mid \forall x.\phi$$

where a is an atomic formula. The semantics of elementary formula are given in a standard way. It is well-known that the theory of real-closed fields in the language of ordered fields is decidable [33].

A classical observation is that the minmax value can be written as an elementary formula in the theory of ordered fields. Namely, let $A = (a_{ij})$ be a matrix with entries in the ordered field H. Then the statement $y = val_1(A)$ can be written as an elementary formula over H. Following from this, we have:

Lemma 4 ([11]) *Let $\mathcal{G}$ be a two-player game and q a state in $\mathcal{G}$. Let Ψ be a Rabin chain condition and let $\langle\langle 1\rangle\rangle\Psi = [\![\phi]\!]$. Then statement $\vec{y} = [\![\phi]\!]$ can be written as an elementary formula in the theory of real closed fields.*

We note that in the lemma above, the vector $\vec{y}$ is indexed by the states of $\mathcal{G}$. Thus for each state $q \in Q$, we denote the q-indexed coordinate of $\vec{y}$ as $(\vec{y})_q$.

3.2 Algorithm

In this section, we present our model-checking algorithms for PATL over pCGSs. The algorithms share the same basic structure of those proposed in [19] for CTL (see [9] for a leisure demonstration). In a nutshell, given a *state* formula ϕ_s, the algorithm recursively evaluates the truth-values of the state subformulae ψ_s of ϕ_s at all states, starting from the propositional formulae of ϕ_s and following the recursive definitions of each modality. The whole process will be gathered up in a global labelling algorithm. Indeed, since PATL differs from CTL only in the presence of the $\langle\langle A\rangle\rangle [\psi_p]_{\bowtie r}$, we can exploit the same techniques proposed for CTL to deal with the operators $\wedge, \vee$, etc. In the algorithms below , we only need to examine the case corresponding to $\langle\langle A\rangle\rangle [\psi_p]_{\bowtie r}$.

As expected, the problem of model checking a *multi-player* concurrent game structure w.r.t. $\langle\langle A\rangle\rangle [\psi_p]_{\bowtie r}$ boils down to the problem of solving a *two-player* concurrent game. To see this, it suffices to note that any *coalition strategy* can be decomposed into single strategy of each player in this coalition. Assume a probabilistic concurrent game structure $\mathcal{G} = \langle k, Q, \Pi, \pi, d, \delta\rangle$. The first step is to define a two-player concurrent game $\mathcal{H}$ which is played by A and $\bar{A}$ where $A \subseteq [k]$ and $\bar{A} = [k] \setminus A$, as follows. For simplicity, we denote the player A by 1 and accordingly, 2 for $\bar{A}$.

$$\mathcal{H} = \langle Q, \Pi, \pi, \Gamma_1, \Gamma_2, \gamma\rangle$$

where

- For each $q \in Q$, $\Gamma_1(q) = \{(j_a)_{a\in A} \mid 1 \le j_a \le d_a(q)\}$;
- For each $q \in Q$, $\Gamma_2(q) = \{(j_a)_{a\in \bar{A}} \mid 1 \le j_a \le d_a(q)\}$; and
- For each $q \in Q$, any $c_1 \in \Gamma_1(q)$ and $c_2 \in \Gamma_2(q)$, $\gamma(p \mid q, c_1, c_2) = \delta(q, c_1 \cdot c_2)(p)$ for any $p \in Q$.

As stated before, the rest of this subsection will be devoted to demonstrating how to model check $\langle\langle A\rangle\rangle[\psi_p]_{\bowtie r}$. We distinguish two cases, depending on $\bowtie$ (recall that $\bowtie \in \{<, \le, =, \ge, >\}$), which are presented in the following two subsections respectively.

3.2.1 Case $\bowtie \in \{<, \le, \ge, >\}$

It turns out that in this case, we can exploit the results of two-player game for our model checking algorithm in a somehow direct way. However, due to the intricacy of the concurrent game structure, we have to cope with the optimal strategies and ϵ-optimal strategies more carefully. The following facts are well-known: concurrent games with reachability winning condition have *memoryless ϵ-optimal* strategies. However, there are deterministic concurrent games without *optimal* strategies. This means in such a game, player 1 can obtain the value *arbitrarily close* to the *optimal value*, but can *not* achieve the optimal one, which leads to the following Lemma 5.

Given a *path* formula ψ_p, we assume that we have already computed (recursively) the satisfaction sets of all maximal *state* PATL-subformulae $\gamma_1, \cdots, \gamma_n$ of ψ_p, so we can view them as atomic propositions. In addition we have labelled the states of $\mathcal{G}$ appropriately with new atomic propositions $r_1, \cdots, r_n$ (to save notations, we still denote it by $\mathcal{G}$). Let

$$\widehat{\psi_p} = \psi_p\{\gamma_1 \leftarrow r_1, \cdots, \gamma_n \leftarrow r_n\}$$

be the formula we obtain from ψ_p by replacing each occurrence of γ_i by r_i for $1 \le i \le n$.

Lemma 5 *Given probabilistic concurrent game structure $\mathcal{G} = \langle k, Q, \Pi, \pi, d, \delta\rangle$ and PATL formula $\phi_s = \langle\langle A\rangle\rangle[\psi_p]_{\bowtie r}$, where $A \subseteq [k]$. The following properties hold:*

(i) Assume $\bowtie \in \{\ge, >\}$. Then $\mathcal{G} \models \phi_s$ if and only if $\langle\langle 1\rangle\rangle\widehat{\psi_p} > r$.

(ii) Assume $\bowtie \in \{\le, <\}$. Then $\mathcal{G} \models \phi_s$ if and only if $\langle\langle 2\rangle\rangle\widehat{\neg\psi_p} < r$.

Here, we note that $\widehat{\neg\psi_p}$ can be rewritten as a path formula, by pushing the negation inwards jumping $\mathcal{X}$ and interchanging $\mathcal{U}$ and $\mathcal{R}$. The intuition underlying this lemma lies in that for model checking, we only need to consider *strict inequality*, since generally, we can only achieve ϵ-optimal strategies. The correctness of this lemma follows from the semantics of PATL, the definitions of the optimal values and ϵ-optimal values and the determinacy theorem mentioned before. We omit the details because of the space restriction. In addition, due to the duality of $\le, <$ and $\ge, >$, from now on we only consider the case for $\ge, >$ (i.e. case (i) in Lemma 5), and another case can be obtained by switching the roles of two players 1 and 2.

In the sequel, we demonstrate how to deal with three temporal modalities, namely $\mathcal{X}$, $\mathcal{U}$ and $\mathcal{R}$, separately.

Modality $\mathcal{X}$ First we consider the case of formula $\langle\langle A\rangle\rangle[\mathcal{X}\phi_s]_{\bowtie r}$. Let $T \subseteq Q$ be defined as $T = \{q \in Q \mid$

$q \models \phi_s\}$. Clearly,

$$\langle\langle A\rangle\rangle[\mathcal{X}\phi_s]_{\bowtie r} = [\![\mathsf{Ppre}_A([T])]\!] \quad (1)$$

Note that since $\mathsf{Ppre}_A([T])$ is a quantitative μ-calculus formula, by Lemma 4 the statement $\vec{y} = [\![\mathsf{Ppre}_A([T])]\!]$ can be written as an elementary formula in the theory of real closed fields. It follows from Lemma 5 that $q \models \langle\langle A\rangle\rangle[\mathcal{X}\phi_s]_{\bowtie r}$ (where $\bowtie \in \{\geq, >\}$) can be encoded as the formula in $(\mathbb{R}, +, \cdot, \leq)$, as follows

$$\exists \vec{y}.(\vec{y} = [\![\mathsf{Ppre}_A([T])]\!] \wedge (\vec{y})_q \bowtie r)$$

Modality $\mathcal{U}$ In this case we consider the case of formula $\langle\langle A\rangle\rangle[\phi_s \mathcal{U} \psi_s]_{\bowtie r}$. As in the previous case, all the same let $S, T \subseteq Q$ be defined as $S = \{q \in Q \mid q \models \phi_s\}$ and $T = \{q \in Q \mid q \models \psi_s\}$. Then it is not difficult to show that

$$\langle\langle A\rangle\rangle[\phi_s \mathcal{U} \psi_s]_{\bowtie r} = [\![\mu x.[T] \vee ([S] \wedge \mathsf{Ppre}_A(x))]\!]$$

Note that since $\mu x.[T] \vee ([S] \wedge \mathsf{Ppre}_A(x))$ is a quantitative μ-calculus formula, by Lemma 4 the statement $\vec{y} = [\![\mu x.[T] \vee ([S] \wedge \mathsf{Ppre}_A(x))]\!]$ can be written as an elementary formula in the theory of real closed fields. It follows from Lemma 5 that $q \models \langle\langle A\rangle\rangle[\phi_s \mathcal{U} \psi_s]_{\bowtie r}$ (where $\bowtie \{\geq, >\}$) can be encoded as the formula in $(\mathbb{R}, +, \cdot, \leq)$, as follows

$$\exists \vec{y}(\vec{y} = [\![\mu x.[T] \vee ([S] \wedge \mathsf{Ppre}_A(x))]\!] \wedge (\vec{y})_q \bowtie r) \quad (2)$$

Modality $\mathcal{R}$ In this case we consider the case of formula $\langle\langle A\rangle\rangle[\phi_s \mathcal{R} \psi_s]_{\bowtie r}$. As in the previous case, all the same let $S, T \subseteq Q$ be defined as $S = \{q \in Q \mid q \models \phi_s\}$ and $T = \{q \in Q \mid q \models \psi_s\}$. Then it is not difficult to show that

$$\begin{aligned}&\langle\langle A\rangle\rangle[\phi_s \mathcal{R} \psi_s]_{\bowtie r}\\ &\quad = [\![(\nu x.[T] \wedge \mathsf{Ppre}_A(x)) \vee (\mu x.[S] \vee ([T] \wedge \mathsf{Ppre}_A(x)))]\!]\end{aligned}$$

For brevity, let $\chi = (\nu x.[T] \wedge \mathsf{Ppre}_A(x)) \vee (\mu x.[S] \vee ([T] \wedge \mathsf{Ppre}_A(x)))$. Note that since χ is a quantitative μ-calculus formula, by Lemma 4 the statement $\vec{y} = [\![\chi]\!]$ can be written as an elementary formula in the theory of real closed fields. It follows from Lemma 5 that $q \models \langle\langle A\rangle\rangle[\phi_s \mathcal{U} \psi_s]_{\bowtie r}$ can be encoded as the formula in $(\mathbb{R}, +, \cdot, \leq)$ as

$$\exists \vec{y}(\vec{y} = [\![\chi]\!] \wedge (\vec{y})_q \bowtie r) \quad (3)$$

To conclude, we note (1)(2) and (3) are all formulae in the theory of real closed fields and thus they are *decidable*, which constitute the model checking algorithm.

3.2.2 Case $\bowtie \in \{=\}$

The general idea of the algorithm for this case is the same as in the previous subsection, namely, to reduce the model checking problem to the decision problem of theory of *real closed fields*. However, here we follow a rather different reduction, since we have to cope with equality rather than inequality and thus only finding the value of two-player concurrent game is not sufficient for our purpose.

Assume we are checking formula $\langle\langle A\rangle\rangle[\psi_p]_{=r}$ w.r.t. a state $q_0 \in Q$. What we intent to encode is, intuitively,

$$\exists \sigma_A . \forall \sigma_{\bar{A}} . \mathbb{P}_q^{\sigma_A, \sigma_{\bar{A}}}(\{\lambda \in \Omega_q \mid \lambda \models \psi_p\}) = r \quad (4)$$

Generally, it is not possible to do this. However, thanks to the simplicity of the *path* formula ψ_p in our setting, which only involves the modalities $\mathcal{X}, \mathcal{U}$ and $\mathcal{R}$ and thus denotes the ω-regular set in $\Sigma_1^0 \cup \Pi_1^0$ (safety and reachability, which lie in the lowest level of Borel hierarchy), it is not difficult to see that *memoryless* strategies suffice for coalition A to win the game. This is stated in the following lemma, for which the proof is omitted due to space restriction.

Lemma 6 *Memoryless (randomized) strategies are sufficient to win the concurrent game with winning conditions expressed by ω-regular set which is in $\Sigma_1^0 \cup \Pi_1^0$ with exact probability.*

In order to instantiate (4), for each state $q \in Q$ and action $c \in \Gamma_1(q)$, we introduce a first-order variable $X_{q,c}$. Intuitively, $X_{q,c}$ denotes the *probability* of choosing the action c in the state $q \in Q$, thus it represents the strategy of player 1 (i.e., A). Correspondingly, for each state $q \in Q$ and action $d \in \Gamma_2(q)$, we also introduce a first-order variable $Y_{q,d}$ for player 2. Then (4) turns out to be

$$\begin{aligned}&\exists\{X_{q,c} \mid q \in Q, c \in \Gamma_1(q)\}\\ &\quad \textstyle\bigwedge_{X_{q,c}}(0 \leq X_{q,c} \leq 1) \wedge \bigwedge_{q \in Q}(\sum_{c \in \Gamma_1(q)} X_{q,c} = 1)\\ &\forall\{Y_{q,d} \mid q \in Q, d \in \Gamma_2(q)\}\\ &\quad \textstyle\bigwedge_{Y_{q,d}}(0 \leq Y_{q,d} \leq 1) \wedge \bigwedge_{q \in Q}(\sum_{d \in \Gamma_2(q)} Y_{q,d} = 1)\\ &\qquad \wedge\ \mathbb{P}_q^{\sigma_A, \sigma_{\bar{A}}}(\{\lambda \in \Omega_{q_0} \mid \lambda \models \psi_p\}) = r\end{aligned}$$

It remains to encode $\mathbb{P}_q^{\sigma_1, \sigma_2}(\{\lambda \in \Omega_{q_0} \mid \lambda \models \psi_p\}) = r$. To this end, we proceed by a case analysis on the form of ψ_p.

1. $\psi_p = \bigcirc \phi_s$. First note that here ϕ_s is a state formula. It is easy to see that the probability equals

$$\sum_{c \in \Gamma_1(q)} \sum_{d \in \Gamma_2(q)} \sum_{q' \in Q, q' \models \psi_s} X_{q,c} \cdot Y_{q,d} \cdot \gamma(q, c, d)(q')$$

It follows that (4) can be instantiated further as

$$\begin{aligned}&\exists\{X_{q,c} \mid q \in Q, c \in \Gamma_1(q)\}\\ &\quad \textstyle\bigwedge_{X_{q,c}}(0 \leq X_{q,c} \leq 1) \wedge \bigwedge_{q \in Q}(\sum_{c \in \Gamma_1(q)} X_{q,c} = 1)\\ &\forall\{Y_{q,d} \mid q \in Q, d \in \Gamma_2(q)\}\\ &\quad \textstyle\bigwedge_{Y_{q,d}}(0 \leq Y_{q,d} \leq 1) \wedge \bigwedge_{q \in Q}(\sum_{d \in \Gamma_2(q)} Y_{q,d} = 1)\\ &\qquad \wedge\ \sum_{c \in \Gamma_1(q_0)} \sum_{d \in \Gamma_2(q_0)}\\ &\qquad\qquad \textstyle\sum_{q' \in Q, q' \models \psi_s} X_{q,c} \cdot Y_{q,d} \cdot \gamma(q, c, d)(q') = r\end{aligned} \quad (5)$$

2. $\psi_p = \psi_1 \mathcal{U} \psi_2$. As in the previous case, here both ψ_1 and ψ_2 are *state* formulae. This case is much more involved than the previous one. However, the basic idea remains the same. Recall that the main task is to compute the probability $\mathbb{P}^{\sigma_1,\sigma_2}_{q_0}(\{\lambda \in \Omega_{q_0} \mid \lambda \models \psi_1 \mathcal{U} \psi_2\})$ for which we have to consult to the well known Chapman-Kolmogorov equation, as follows.

$$\mathbb{P}_s(\psi_p) = \begin{cases} 1 & \text{if } s \models \psi_2 \\ 0 & \text{if } s \not\models \psi_1 \\ \sum_{c\in\Gamma_1(q)} \sum_{d\in\Gamma_2(q)} \sum_{q'\in Q} & \\ \quad X_{q,c} \cdot Y_{q,d} \cdot \gamma(q,c,d)(q') \cdot \mathbb{P}_{q'}(\psi_p) & \text{o.w.} \end{cases}$$

For each $s \in Q$, we introduce a variable Z_s. And for simplicity, we define $\Im(Z_s, Z_t)$ as

$$\begin{array}{l} \Im(Z_s, Z_t) \stackrel{\text{def}}{=} \\ \quad (s \models \psi_2 \Rightarrow Z_s = 1) \wedge (s \not\models \psi_1 \Rightarrow Z_s = 0) \wedge \\ \quad (s \not\models \psi_2 \wedge s \models \psi_1 \\ \qquad \Rightarrow Z_s = \sum_{c\in\Gamma_1(q)} \sum_{d\in\Gamma_2(q)} \sum_{t\in Q} \\ \qquad\qquad X_{q,c} \cdot Y_{q,d} \cdot \gamma(s,c,d)(t) \cdot Z_t) \end{array}$$

It follows that that (4) can be instantiated as

$$\exists \{Z_q \mid q \in Q\}.(0 \le Z_q \le 1) \wedge \bigwedge_{p,q\in Q} \Im(Z_p, Z_q) \wedge (Z_{q_0} = r) \tag{6}$$

3. $\psi_p = \psi_1 \mathcal{R} \psi_2$. This case is almost the same as the previous one and is left to the readers to work out the details.

To conclude, since (5) and (6) are formulae in the theory of real closed fields and thus they are *decidable*, we are done.

Complexity It is easy to see that for the algorithm of each modality, the resulting formula in order field language is *polynomial* in the size of a given pCGS and PATL formula (we take the standard measure on the size of the formula). In addition, the number of quantifier alternations is *fixed*. Thus we obtain the EXPTIME upper complexity bound for each oracle query following [33][2]. So the overall complexity of our algorithm is in $\mathrm{P}^{\mathrm{EXPTIME}}$, which is in turn in EXPTIME.

3.3 Alternative Decision Procedure

In the last subsection, we consider both the cases that $\bowtie \in \{<, \le, =, \ge, >\}$. In this subsection, we present yet another algorithm, inspired by a recent result on solving two-player concurrent game. However, this only applies to the case that strict equality $=$ is *not* allowed in the constraint.

Recall that following the constructions in Section 3.2.1, we can reduce the problem of model checking PATL formula $\phi_s = \langle\langle A \rangle\rangle [\psi_p]_{\bowtie r}$ to the problem of deciding $\langle\langle 1 \rangle\rangle \widehat{\psi_p} > r$ in a two-player game, in the case of $\bowtie \in \{\ge, >\}$ (for $\bowtie \in \{<, \le\}$, this can be done in a symmetric way). Thus, to seek an efficient way to solve two-player game is essential. Fortunately, the following result actually provides an *oracle* for solving two-player concurrent game, which is more efficient, compared with the oracle in the previous subsections.

Theorem 7 ([12]) *For all concurrent game structure $\mathcal{G}$, for all parity objective Ω_e and Ω_o, and for all rationals $\varepsilon > 0$, for all rationals r, whether $\langle\langle 1 \rangle\rangle(\Omega_e)(s) \in [r - \varepsilon, r + \varepsilon]$ can be decided in* NP $\cap$ coNP.

We note that obviously, here $\widehat{\psi_p}$ denotes an ω-regular property in $\Sigma^0_1 \cup \Pi^0_1$ and thus is a parity objective. Let Ω_o be the ω-regular set corresponding to ψ_p. By this theorem, our algorithm is sketched as follows:

(i) Query the oracle whether $\langle\langle 1 \rangle\rangle(\Omega_e)(s) \in [r - 2\varepsilon, r]$? If the answer is "yes", then the algorithm returns "no";

(ii) Otherwise, query the oracle whether $\langle\langle 1 \rangle\rangle(\Omega_e)(s) \in [r, r + 2\varepsilon]$? If the answer is "yes", then algorithm returns "yes", otherwise, it returns "no".

Complexity By Theorem , for each query, the complexity is in NP $\cap$ coNP, it follows that the overall complexity of the this algorithm is in $\mathrm{P}^{\mathrm{NP}\cap\mathrm{coNP}}$, which is a lower complexity bound than EXPTIME [1].

4 Beyond PATL

In this section, we suggest two more formalisms for the specification of open probabilistic systems. One is PATL*, which is very natural in the sense that it is akin to PCTL*, and thus the extension is in analogy with CTL* w.r.t. CTL. The other is called *probabilistic game logic* (PGL), which introduces the *strategy* quantifier. Both of these two extensions are more expressive than PATL by definition. However, as we shall demonstrate, the model checking problem for PATL* (excluding $=$ in the constraint) is still decidable, although in a higher complexity while the PGL is even more intractable in the sense that the model checking becomes undecidable. Consequently, from the perspective of formal verification, PGL is not a good candidate.

4.1 PATL* and Model Checking Algorithm

The logic PATL is a fragment of a more expressive logic called PATL*, where the *state* formulae are kept unchanged while the *path* formulae turn to

$$\phi_p, \psi_p \quad ::= \quad \phi_s \mid \neg\phi_p \mid \phi_p \wedge \psi_p \mid \mathcal{X}\phi_p \mid \phi_p \mathcal{U} \psi_p \mid \phi_p \mathcal{R} \psi_p$$

[1] Of course, we adopt the widely-accepted hypothesis that P$\neq$ NP and the polynomial hierarchy does not collapse.

where $P \in \Pi$, $A \subseteq [k]$, $\bowtie \in \{<, \leq, =, \geq, >\}$ and $r \in [0,1]$.

As for the semantics, in the case of state formulae, the definition is unchanged w.r.t. PATL, while the path formulae are interpreted as follows:

$$\begin{array}{lcl}
\lambda \models \phi_s & \Leftrightarrow & \lambda(0) \models \phi_s \\
\lambda \models \neg\phi_p & \Leftrightarrow & \lambda \not\models \phi_p \\
\lambda \models \phi_p \wedge \psi_p & \Leftrightarrow & \lambda \models \phi_p \wedge \lambda \models \psi_p \\
\lambda \models \mathcal{X}\phi_p & \Leftrightarrow & \lambda[1] \models \phi_p \\
\lambda \models \phi_p \mathcal{U} \psi_p & \Leftrightarrow & \exists i \in \mathbb{N}.\ \lambda[i] \models \psi_p \text{ and for} \\
 & & \text{any } 0 \leq j < i,\ \lambda[j] \models \phi_p \\
\lambda \models \phi_p \mathcal{R} \psi_p & \Leftrightarrow & \lambda \models \neg(\neg\phi_p \mathcal{U} \neg\psi_p)
\end{array}$$

Model Checking Algorithm In this section, we explain how to model check a pCGS w.r.t. a PATL* formula. We emphasize that here, we exclude the strict equality $=$ in the PATL* formulae, namely, the results here can be read as the counterparts of the ones in Section 3.2.1. We leave the more general case as an open problem.

As before, let ϕ_s be a PATL* state formula. We only need to consider ϕ_s to be of the form $\phi_s = \langle\langle A\rangle\rangle[\psi_p]_{\bowtie r}$ since the other cases are either trivial or can be reduced to this one. Again, we assume that we have already computed (recursively) the satisfaction sets of all maximal state PATL*-subformulae $\gamma_1, \cdots, \gamma_n$ of ψ_p, so we can view them as atomic propositions. And we have labelled the states of $\mathcal{G}$ appropriately with new atomic propositions $r_1, \cdots, r_n$. Define $\widehat{\psi_p}$ as before. However, note that in this case $\widehat{\psi_p}$ is a general LTL formula over the atomic propositions $r_1, \cdots, r_n$ and for all infinite paths λ of given system, it holds that $\lambda \models \psi_p$ iff $\lambda \models \widehat{\psi_p}$.

Following the construction of Section 3.2.1, we shall compute $\langle\langle 1\rangle\rangle\widehat{\psi_p}$. This can be done by the standard ω-automaton approach: It suffices to encode the LTL as a Rabin-chain automaton, coping with then the game structure consisting of the synchronous product of the original game structure with the Rabin-chain automaton. An ω-automaton accepts infinite words over a give alphabet. In the probabilistic setting, a kind of *determinism* is necessary.

Definition 8 *[Deterministic Rabin chain automata] A deterministic Rabin chain automaton is a tuple*

$$\mathcal{A} = (\Sigma, S, s_0, \rho, \alpha)$$

where

- Σ *is a finite alphabet;*
- S *is a finite set of states;*
- s_0 *is an initial state;*
- $\rho : S \times \Sigma \to S$ *is a transition function; and*
- $\alpha = \bigvee_{i=0}^{l-1}(\Box\Diamond U_{2i} \wedge \neg\Box\Diamond U_{2i+1})$

where $l > 0$ *and* $\emptyset = U_{2l} \subseteq U_{2l-1} \subseteq U_{2l-2} \subseteq \cdots \subseteq U_0 = S$.

Given a deterministic Rabin chain automaton $\mathcal{A}$ and an infinite word $w \in \Sigma^\omega$ over Σ, we call the unique sequence $r = q_0, q_1, \cdots$ with $q_{i+1} = \rho(q_i, w_{i+1})$ for $i \geq 0$ the run of $\mathcal{A}$ over w. For a run $r = s_0 s_1 \cdots$, let

$$\lim(r) = \{s \in S \mid s = s_i \text{ for infinitely many } i\text{'s}\}$$

A run r is *accepting* according to the Rabin chain condition if there exists some $0 \leq i \leq l-1$ such that $\lim(r) \cap U_{2i} = \emptyset$ and $\lim(r) \cap U_{2i+1} \neq \emptyset$. We say $\mathcal{A}$ accepts w if and only if r is accepting.

The language of a deterministic Rabin chain automaton $\mathcal{A}$ is the set

$$\mathcal{L}_\omega(\mathcal{A}) = \{w \in \Sigma^\omega \mid \mathcal{A} \text{ accepts } w\}$$

The following result is standard. We note that [25] provides the most efficient construction, to the best of our knowledge.

Lemma 9 *Given an LTL formula φ over Π, a deterministic Rabin chain automaton $\mathcal{A}$ with the alphabet $\Sigma = \wp(\Pi)$ can be constructed such that*

$$\mathcal{L}_\omega(\varphi) = \mathcal{L}_\omega(\mathcal{A})$$

Now we are in a position to present the model checking algorithm. As the first step, we construct a deterministic Rabin chain automata $\mathcal{A} = (\wp(\Pi), S, s_0, \rho, \alpha)$ with $\mathcal{L}_\omega(\widehat{\psi_p}) = \mathcal{L}_\omega(\mathcal{A})$. Then given $\mathcal{G} = (k, Q \times S, \Pi, \pi, d, \delta)$, we build a product pCGS $\mathcal{G}' = \mathcal{G} \times \mathcal{A}$ for $\mathcal{G}$ and $\mathcal{A}$ as follows:

Definition 10 $\mathcal{G}' = (k, Q \times S, \Pi, \pi', d', \delta')$, *where*

- $\pi'((q,s)) = \pi(q)$;
- *for any* $a \in [k]$, $d'_a((q,s)) = d_a(q)$;
- *let* $\vec{c} = (c_a)_{a\in[k]}$ *such that* $1 \leq c_a \leq d'_a((q,s))$ *for each* a,

$$\delta'((q,s),\vec{c})((q',t)) = \begin{cases} \delta(q,\vec{c})(q') & \text{if } t = \rho(s,\pi(q)) \\ 0 & \text{o.w.;} \end{cases}$$

Remark 1 *Here, the condition of "deterministic" is essential to ensure the soundness of the construction.*

It is easy to see that there is a one-to-one correspondence between the pathes of $\mathcal{G}$ and those of $\mathcal{G} \times \mathcal{A}$ by path-lifting. In addition, taking a path λ of $\mathcal{G}$, lifting it to a path λ' of $\mathcal{G} \times \mathcal{A}$ and then reducing it to its second component, gives then run of $\mathcal{A}$ over $\pi(\lambda)$. Therefore $\mathcal{G} \times \mathcal{A}$ also keeps track whether $\pi(\lambda)$ will be accepted by $\mathcal{A}$ or not, namely, whether λ satisfies ψ'_p or not.

Define

$$\alpha' = \bigvee_{i=0}^{l-1}(\Box\Diamond U'_{2i} \wedge \neg\Box\Diamond U'_{2i+1})$$

where $U_i' = U_i \times S$.

Taking this into account it is easy to show that

$$\sup_{\sigma_A \in \Upsilon_A} \inf_{\sigma_{\bar{A}} \in \Upsilon_{\bar{A}}} \mathbb{P}_q^{\sigma_A, \sigma_{\bar{A}}}(\{\lambda \in \Omega_q \mid \lambda \models \psi_p'\}) =$$

$$\sup_{\sigma_A' \in \Upsilon_A'} \inf_{\sigma_{\bar{A}}' \in \Upsilon_{\bar{A}}'} \mathbb{P}_q^{\sigma_A', \sigma_{\bar{A}}'}(\{\lambda' \in \Omega_{q \times s_0} \mid \lambda \models \alpha'\})$$

By this construction, we can turn the *external* winning condition which is specified by an LTL formula $(\widehat{\psi_p})$ to *internal* winning condition on the concurrent game structure. Then we can apply the same techniques exploited in previous section for PATL. The main difference lies in that here we have to deal with more involved winning condition. The following theorem is taken from [11]

$$\langle\langle 1 \rangle\rangle \alpha' = [\![\eta_{2l-1} x_{2l-1} \cdots \mu x_1 \nu x_0 (\bigvee_{i=0}^{2l-1} ([C_i] \wedge \mathsf{Ppre}_1(x_i)))]\!]$$

where $\eta_n = \nu$ if n is even and $\eta_n = \mu$ if n is odd; and $C_i = U_i \setminus U_{i+1}$ for $0 \leq i \leq 2l-1$.

By Lemma 4, clearly, this can be encoded in the theory of real closed fields. We have done.

Remark 2 *We discuss the difficulty of adopting $=$ in the specification briefly. In the case of PATL, the formula $\widehat{\psi_p}$ is only a strict subset of LTL, for which the memoryless strategies suffice to win the game, as we state in Lemma 6; However, in the case of PATL*, $\widehat{\psi_p}$ is considerably more involved, and thus we have to use Rabin chain (parity) condition to encode it. The consequence is that generally winning strategies need infinite memory, namely, Lemma 6 fails, which, in turn invalidates the approach in Section 3.2.2. We conjecture that in this case, the model checking is undecidable. For more discussions we refer the readers to Remark 3.*

Complexity of the algorithm As the case in PATL, we first analyze the complexity of algorithm of each modality. Since from LTL to Rabin chain automata, there is a 2EXPTIME blow up. It follows that the resulting formula is 2EXPTIME in the size of a given probabilistic concurrent game structure and PATL* formula with fixed number of quantifier alternations. Thus we obtain the 3EXPTIME upper complexity bound following [33][2]. So the overall complexity of our algorithm is in $\mathrm{P}^{3\mathrm{EXPTIME}}$, which is in turn in 3EXPTIME.

4.2 Probabilistic Game Logic

In this section, we propose yet another extension, namely *probabilistic game logic* (PGL), of which even PATL* is a fragment. To understand this, let us note that in PATL, the parameterized quantifier $\langle\langle A \rangle\rangle$ first stipulates the *existence* of strategies for the players in A, and then *universally* quantifies over the outcomes of the stipulated strategies. One may generalize it by separating the two concerns into *strategy* quantifiers and *path* quantifiers, which leads to the following definition:

Definition 11 *The syntax of* PGL *is defined by the following grammar:*

$$\begin{array}{rcl} \phi_s & ::= & \top \mid P \mid \neg\phi_s \mid \phi_s \wedge \phi_s \mid \boxplus A.\theta_t \\ \theta_t & ::= & \phi_s \mid \neg\theta_t \mid \theta_t \wedge \theta_t \mid [\psi_p]_{\bowtie r} \\ \psi_p & ::= & \theta_t \mid \neg\psi_p \mid \psi_p \wedge \psi_p \mid \mathcal{X}\phi_p \mid \phi_p \mathcal{U} \psi_p \mid \phi_p \mathcal{R} \psi_p \end{array}$$

where $P \in \Pi$, $A \subseteq [k]$, $\bowtie \in \{<, \leq, \geq, >\}$ and $r \in [0,1]$.

We note that in PGL, the constraint $=$ is redundant, since $\boxplus A.[\psi_p]_{=r}$ can be written as $\boxplus A.([\psi_p]_{\geq r} \wedge [\psi_p]_{\leq r})$, which is allowed in PGL, but not in PATL or PATL*.

We now define the semantics of PGL w.r.t. pCGS $\mathcal{G}$. Recall that $A \subseteq [k]$ is a coalition and σ_A and $\sigma_{\bar{A}}$ be the coalition strategies for A and $\bar{A}$ respectively. Once the starting state q and the coalition strategies σ_A and $\sigma_{\bar{A}}$ for the two coalitions have been chosen, we can obtain a *probabilistic execution tree*, which is, an unwinding of a (possibly infinite state) *Discrete Time Markov Chain* (DTMC). We interpret the *tree formulae* on DTMC (or equivalently, on probabilistic execution tree), which follows similar way as in PCTL. Formally, given a coalition strategy $\sigma_A = (\sigma_a)_{a \in A}$, we define the set of possible *probabilistic execution trees* of σ_A from a state $q \in Q$ to be the set $\mathsf{PET}(q, \sigma_A)$ of unwinding DTMCs that the players in A enforce when they follow the strategy σ_A. We use $\mathbb{T}_q^{\sigma_A}$ to range over $\mathsf{PET}(q, \sigma_A)$. Except for $\boxplus A.\theta_t$, the interpretation of *state* formulae, *tree* formulae and *path* formulae is defined in the same way as in PATL and PATL*. The only interesting case is

$$\begin{array}{ll} q \models \boxplus A.\theta_t & \Leftrightarrow \text{there is a coalition strategy } \sigma_A \text{ such} \\ & \quad \text{that for any } \mathbb{T}_q^{\sigma_A} \in \mathsf{PET}(q, \sigma_A), \mathbb{T}_q^{\sigma_A} \models \theta_t \end{array}$$

We note in the definition above, $\mathbb{T}_q^{\sigma_A}$ is essentially a DTMC.

In [3], the authors show, among others, that the existence of winning strategies in MDPs (a.k.a $1\frac{1}{2}$ games) with *PCTL objectives* is highly undecidable (Σ_1^1 hard). Since as we state before, MDP is only a special case of pCGS, by a straightforward reduction, we show this negative result implies that:

Theorem 12 *Model checking PGL is undecidable.*

Remark 3 *We note that for PATL,* memoryless *(but randomized) strategies suffice for coalition of players to achieve the desired value of the game, thus force the game to favor them; While for PATL*, both the* memory *and the* randomness *are required, although in a* finite *fashion. The difference lies in the* path *formulae admitted in the two logics. From the ω-regular property perspective, in PATL, the ω-regular sets specified by path formulae are in $\Sigma_1^0 \cup \Pi_1^0$ while in PATL*, they need to*

be specified by parity condition, and thus beyond this low Borel hierarchy (although still in $\Sigma_3^0 \cap \Pi_3^0$). Furthermore, in probabilistic game logic, the winning conditions are interpreted on trees *rather than* paths*, which lifts the* linear *time notion to* branching *time one. We emphasize that this is precisely the source of the undecidability, as Theorem 12 indicates. In this sense, the logic PATL identifies a class of properties for open probabilistic systems for which it suffices to solve iterated finite game in a rather efficient way, which might be a strong support for it to be a good candidate of specification.*

5 Conclusion and Related Works

In our points of view, there are (at least) two main trends in the research of system verification: (1) from closed systems to open systems and (2) from the qualitative analysis to quantitative analysis, which are motivated by the state-of-art of current embedded reactive systems. To meet these challenges, in this paper, we aim at investigating *open probabilistic system*: By analyzing the *open* system, one can cope with not only traditional verification but also control and synthesis challenges; By introducing probabilistic extension, one can deal with systems that are subject to various phenomena of stochastic nature, which is, in our point of view, more akin to the reality and covers one of the most significant aspects of quantitative analysis. To instantiate our goal, we extend the framework of alternating-time temporal logic to a probabilistic setting, following the style of classical probabilistic computation tree logic, and thus obtain two *probabilistic alternating-time temporal logics*, i.e. PATL and PATL*. They can be used to specify the desire properties of open probabilistic systems. We also propose probabilistic concurrent game structures as the general model for such systems, which, are also natural semantics models for the proposed logics. Based on them, we develop model checking algorithms for PATL and PATL*. Moreover, some more expressive logic, which, unfortunately lacks the decidability of model checking, is also discussed. We believe the works presented here establish the first (albeit preliminary) step of a general formal framework to model, specify, analyze, design and even program open probabilistic systems, in particular, embedded reactive systems.

Related Works There have been extensive works on ATL and its extensions. In [1], ATL has been proposed and defined over CGS. In [16, 18], expressiveness issues are considered for ATL and ATL*. Complexity results about model checking (for ATL, ATL$^+$, ATL*) can be found in, among others, [1, 30, 18]. In [31], the decidability of satisfiability for alternating-time μ-calculus is shown. In [32], the authors present a sound and complete proof system for proving ATL* properties over infinite-state systems. The abstraction-refinement technique is also considered in, among others, [7]. There are also some extensions of the framework of ATL to quantitative models, in particular, concerning on real-time aspect, we are aware of two recent works: [15] and [17], where the former address TATL, which adds freeze quantifiers to ATL while the latter considers the durational concurrent game structure and extends ATL following the TCTL style. Model checking algorithms are given in their own framework respectively. However, for the probabilistic extension of ATL, to the best of our knowledge, we are not aware of any other works.

On the other hand, there is a sea of works addressing the formal verification and control of probabilistic systems. Here, we only mention the tip of the iceberg and we restrict ourselves to the discrete time setting. The probabilistic computation tree logic is introduced in [20], where model checking algorithm coping with DTMC is also provided. [4] extends this work to MDP, where the algorithms for MDP w.r.t. PCTL and PCTL* are proposed. These algorithm can be seen as the special cases of our algorithms, sice both the MDP and PCTL are (very) special cases of pCGS and PATL respectively. As for the control problem, as far as we know, [5] first discusses the controller synthesis for PCTL, and [3] further shows that the problem is, generally undecidable. Our works on probabilistic model checking might include [10]. Research on (stochastic) game is another line of related works, for which, we only refer to [35] for a survey and to its numerous references there.

Acknowledgement We are grateful to Wan Fokkink and Tingting Han for their careful reading of a draft of this paper and valuable comments. We also thank Orna Kupferman for stimulating discussions at CONCUR'06.

References

[1] R. Alur, T. A. Henzinger and O. Kupferman. Alternating-time temporal logic. *Journal of ACM* 49(5): 672-713, 2002.

[2] S. Basu. New results on quantifier elimination over real closed fields and applications to constraint databases. *Journal of ACM* 46(4): 537-555, 1999.

[3] T. Brazdil, V. Brozek, V. Forejt, and A. Kucera. Stochastic games with branching-time winning objectives. In Proc. LICS 2006, pp. 349-358, IEEE Computer Society, 2006.

[4] A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In Proc. FSTTCS 1995. LNCS 1026, pp. 499-513, Springer, 1995.

[5] C. Baier, M. Größer, M. Leucker, B. Bollig, and F. Ciesinski. Controller synthesis for probabilistic systems. In Proc. IFIP TCS 2004. Kluwer, 2004.

[6] C. Baier, B. R. Haverkort, H. Hermanns, J. -P. Katoen and M. Siegle. Validation of Stochastic Systems - A Guide to Current Research. LNCS 2925, Springer, 2004.

[7] T. Ball, and O. Kupferman. An abstraction-refinement framework for multi-agent systems. In Proc. LICS 2006, pp. 379-388, IEEE Computer Society, 2006.

[8] E. Clarke, and E. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Proceeding Logic of Programs 1981*, LNCS 131, pp. 52-71, Springer, 1981.

[9] E. Clarke, O. grumberg, and D. Peled. Model Checking. MIT Press, 1999.

[10] C. Courcoubetis and M. Yannakakis. The compleixity of probabilistic verification. *Journal of ACM*, 42(4):857-907, 1995.

[11] L. de Alfaro, and R. Majumdar. Quantitative solution of omega-regular games. *Journal of Computer and System Science*. 68(2): 374-397, (2004).

[12] K. Chatterjee, L. de Alfaro, and T. Henzinger. The complexity of quantitative concurrent parity games. In Proc. SODA 2006, pp. 678-687, 2006.

[13] C. Hoare. *Communicating Sequential Processes*. Prentics-Hall, 1985.

[14] D. Harel and A. Pnueli. On the development of reactive systems. In *Logics and Models of Concurrent Systems*, volume F-13 of *NATO Advanced Summer Institutes*, 477-498. Springer, 1985.

[15] T. A. Henzinger and V. S. Prabhu. Timed Alternating-Time Temporal Logic. In Proc. FORMATS 2006, LNCS 4202, pp. 1-17, 2006

[16] A. Harding, M. Ryan, and P. -Y. Schobbens. Approximating ATL* in ATL. In Proc. VMCAI 2002, LNCS 2294, pp. 289-301, 2002.

[17] F. Laroussinie, N. Markey, and G. Oreiby. Model checking timed ATL for durational concurrent game strucutures. In Proc. FORMATS 2006, LNCS 4202, pp. 245-259, 2006.

[18] F. Laroussinie, N. Markey, and G. Oreiby. On the Expressiveness and Complexity of ATL. In Proc. FoSSaCS 2007, to appear.

[19] E. M. Clarke, E. A. Emerson and A. P. Sistla. Automatic verification of finite-State concurrent systems using temporal logic specifications. *ACM Transaction of Program Language and System*, 8(2): 244-263, 1986.

[20] H. Hansson and B. Jonsson. A Logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5): 512-535, 1994.

[21] D. Martin. Then determinacy of blackwell games. *Jouranl of Symbolic Logic*, 63(4):1565-1581, 1998.

[22] J. van Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, 1947.

[23] G. Owen. *Game Theory*. Academic Press, 1995.

[24] M. Puterman. *Markov Decision Processes*. Wiley, 1994.

[25] N. Piterman. From nondeterministic buchi and streett automata to deterministic parity automata. In Proc. LICS 2006, pp. 255-264, IEEE Computer Society, 2006.

[26] A. Pnueli. The temporal logic of pragrams. In Proc. FOCS 1977, pp. 46-57. IEEE Computer Society, 1977.

[27] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[28] J. Queille and J. Sifakis. Specification and verification of concurrent systems in CESAR. In Proc. 5th *International Symposium on Programming*, LNCS 137, pp. 337-351, Springer, 1982.

[29] P. Ramadge and W. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1): 81-89, 1989.

[30] P. -Y. Schobbens. Alternating-time logic with imperfect recall. In Proc. LCMAS 2003, ENTCS 85, Elsevier, 2004.

[31] S. Schewe and B. Finkbeiner. Satisfiability and Finite Model Property for the Alternating-Time mu-Calculus. In Proc. CSL 2006, LNCS 4207, pp. 591-605, Springer, 2006.

[32] M. Slanina, H. Sipma, and Z. Manna. Proving ATL* Properties of Infinite-State Systems. In Proc. ICTAC 2006, LNCS 4281, pp. 242-256, 2006.

[33] A. Traski. *A Decision Method for Elementary Algebra and Geometry*. Univ. of California Press, Berkeley, 1951.

[34] W. Thomas. Languages, automata, and logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume III, pp. 389-455. Springer, 1997.

[35] I. Walukiewics. A landscape with games in the background. In Proc. LICS 2004, pp. 356-366. IEEE Computer Society, 2004.