Probabilistic Alternating-Time Temporal Logic and Model Checking Algorithm *

Taolue Chen[1,2] Jian Lu[2]
[1] CWI, Department of Software Engineering,
P.O. Box 94079, 1090GB Amsterdam, The Netherlands
[2] State Key Laboratory of Novel Software Technology,
Nanjing University, Nanjing, Jiangsu, P.R.China, 210093

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 where 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*.*

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 [7] is a well-established technique for verifying whether a system (typically represented by some formal models, say automata) satisfies a given property. In system verification, especially in the area of control, it turns out to be essential to distinguish between *open* and *closed* systems [10]. In the literature, 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 Kripke 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 that it allows to express, for instance, 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

*This work is partially supported by Dutch Bsik project BRICKS, 973 program of China (2002CB312002), 863 program of China (2006AA01Z159), NSFC (60403014, 60603034), NSFJ (BK2006712).

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 [5] for a comprehensive exposition.

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*. Due to space restriction, many interesting results are omitted in this extended abstract and we refer the readers to the technical report [6] for complete proofs, further explanations of intuitions and explicit descriptions of algorithms. Moreover, [6] includes discussion of a even more expressive logic, called *probabilistic game logic* and expanded references on related works.

Related works. We only discuss the most relevant works on 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 [11], where model checking algorithm coping with DTMC is also provided. [3] 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, [4] first discusses the controller synthesis for PCTL, and [2] further shows that the problem is, generally undecidable.

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 $p \to q$ 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

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

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 $\Xi = \{\lambda \mid \pi(\lambda) \models \Psi\}$.

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:*

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

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.

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.

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

3 Model Checking PATL

In this section, we present our model-checking algorithms for PATL over pCGSs. The algorithms share the same basic structure of those for CTL (see [7] 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 $\mathcal{H} = \langle Q, \Pi, \pi, \Gamma_1, \Gamma_2, \gamma\rangle$ where for each $q \in Q$, $\Gamma_1(q) = \{(j_a)_{a\in A} \mid 1 \le j_a \le d_a(q)\}$, $\Gamma_2(q) = \{(j_a)_{a\in \bar{A}} \mid 1 \le j_a \le d_a(q)\}$ and for 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 $\bowtie \in \{<, \le, =, \ge, >\}$). Here, due to space restriction, we only present the simpler case, namely, the case that $\bowtie \in \{<, \le, \ge, >\}$ (we refer the readers to [6] for the remaining case). 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 4.

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 4 *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* $\bowtie \in \{\leq, <\}$. *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. In addition, due to the duality of $\leq, <$ and $\geq, >$, from now on we only consider the case for $\geq, >$ (i.e. case (i) in Lemma 4), and another case can be obtained by switching the roles of two players 1 and 2.

It remains 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 5 ([9]) *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* $\mathrm{NP} \cap \mathrm{coNP}$.

We note that obviously, here $\widehat{\psi_p}$ denotes an ω-regular property in $\Sigma_1^0 \cup \Pi_1^0$ 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 5, for each query, the complexity is in $\mathrm{NP} \cap \mathrm{coNP}$, it follows that the overall complexity of the this algorithm is in $\mathrm{P}^{\mathrm{NP} \cap \mathrm{coNP}}$.

3.1 Extensions

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

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

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 for the path formulae, it can be defined in an expected way. We also develop an algorithm, by appealing to *deterministic Muller automata*. Again, the readers are referred to [6].

References

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

[2] 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.

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

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

[5] 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.

[6] T. Chen and J. Lu. Probabilistic Alternating-Time Temporal Logic and Model Checking Algorithm. CWI Technical report, 2007.

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

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

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

[10] 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.

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