Equivalence of Probabilistic μ-Calculus and p-Automata[*]

Claudia Cauli and Nir Piterman

University of Leicester, Department of Informatics, Leicester, UK
`cc488@le.ac.uk, nir.piterman@gmail.com`

Abstract

An important characteristic of Kozen's μ-calculus is its strong connection with parity alternating tree automata. Here, we show that the probabilistic μ-calculus, μ^p-calculus, and p-automata (parity alternating Markov chain automata) have an equally strong connection. Namely, for every μ^p-calculus formula we can construct a p-automaton that accepts exactly those Markov chains that satisfy the formula. For every p-automaton we can construct a μ^p-calculus formula satisfied in exactly those Markov chains that are accepted by the automaton. The translation in one direction relies on a normal form of the calculus and in the other direction on the usage of vectorial μ^p-calculus.

1998 ACM Subject Classification F.1.2. Models of Computation - Probabilistic Computation, F.4.1. Mathematical Logic, F.4.2. Formal Languages

Keywords and phrases μ-Calculus, Automata, Probabilistic Models

Digital Object Identifier 10.4230/OASIcs...

1 Introduction

The verification of probabilistic systems is an increasingly important area that has led to the development of new formalisms and tools for the evaluation of quantitative properties over stochastic models. These tools range from temporal logics and quantitative variants of Kozen's modal μ-calculus [16] to probabilistic automata and games [2, 14, 18, 19, 20].

This work focuses on two such formalisms, μ^p-calculus and p-automata. The μ^p-calculus has been introduced in [7] as a probabilistic extension of Kozen's modal μ-calculus. The so-called p-automata [9] are probabilistic alternating parity automata that read Markov chains as input. Acceptance of a Markov chain by a p-automaton is decided by an obligation game, that is, a turn-based stochastic parity game with obligations.

The key contribution given by this paper is the proof of the equivalence between μ^p-calculus and p-automata. We provide a framework to translate μ^p-formulas into p-automata and, using the vectorial syntax, define the inverse conversion from p-automata into μ^p-calculus. Thus, we show that the two formalisms have the same expressive power and that they enjoy a close relationship similar to those of Kozen's μ-calculus and alternating tree automata (see below).

Related Work. This study belongs to a general field of research that aims to define a connection between logics and automata theory. An interesting survey conducted by Kupferman et al. in [17] provides insights into this relationship by presenting translations

[*] A longer version of this paper appeared in the proceedings of CIAA 2017 [8]

from a number of temporal logics – linear-time, branching-time, μ-calculus, and its alternation-free fragment – into different classes of alternating tree automata.

Over the last three decades, several studies have focused on the definition of an automata-theoretic approach to Kozen's μ-calculus. In [11], Emerson and Jutla proposed a framework to convert μ-calculus formulas into alternating tree automata, then reduced to their non-deterministic counterpart. Their result complements previous studies by Niwiński [21] that defined the inverse translation from non-deterministic tree automata to μ-calculus, thus showing that Kozen's μ-calculus is equivalent to tree automata in expressive power. In [15], Janin and Walukiewicz introduced μ-automata, alternating automata with a parity acceptance condition that easily translate to and from μ-calculus formulas in disjunctive normal form. In a subsequent work, [22], Niwiński extends his previous result to a broader scope establishing the equivalence between alternating automata over arbitrary algebraic structures – thus including trees – and fixed point terms, a general fixpoint formalism that finds a natural interpretation as a system of equations. Wilke, in [24], addresses the interplay among μ-calculus, alternating tree automata, and games. In particular, he gives a translation from logic to automata and then defines the acceptance problem for automata by reduction to the winner problem in parity games. A comprehensive outline of the relationship among logics, automata, and parity games is given in [13]. Overviews of μ-calculus, including its mathematical foundation, axiomatic system, properties, guarded form, vectorial syntax, game semantics, and equivalence with automata, can be found in [1, 3, 4, 5].

Huth and Kwiatkowska suggested a quantitative μ-calculus to reason about Markov chains [14]. This calculus was extended in [10] by adding a bounded number of probabilistic quantifications and allowing to define PCTL*. The μ^p-calculus allows to nest probabilistic quantifications inside fixpoint operators and, thus, allows for unbounded probabilistic quantifications. It is a subset of the calculus defined in [18, 20]. The μ^p-calculus is expressive enough to include PCTL and PCTL*, the complexity of its decision procedures is reduced, and the algorithms involved wrap up standard algorithms for solution of (quantitative) two-player stochastic parity games in extra layer rather than bespoke algorithms.

2 Background

2.1 Markov Chains

A Markov chain M over a set AP of atomic propositions is a probabilistic labelled transition system defined by the tuple $M = (S, s^{in}, L, P)$, where S is the set of locations; $s^{in} \in S$ is the initial location; L is a labelling function, overloaded to both denote $L : S \to 2^{AP}$ and $L : AP \to 2^{S}$; and P is the probability transition function $P : S \times S \to [0, 1]$. For every location s we define the set $succ(s)$ of its successors as the set of locations s' such that $P(s, s') > 0$. Clearly, the sum of the probabilities of moving from a location to all its successors must be equal to 1, that is $\sum_{s' \in succ(s)} P(s, s') = 1$, and every location has at least one successor.

2.2 Obligation Games

Obligation games [9] are two-player stochastic parity games with obligations that are played on a graph amongst a probabilistic system and two players, called Player 0 and Player 1.

▶ **Definition 1** (Obligation Game). An obligation game G is the tuple

$$G = \big((V, E), (V_0, V_1, V_p), \mathcal{K}, \langle \alpha, O \rangle\big),$$

where V is the set of configurations, partitioned in Player 0 (V_0), Player 1 (V_1), and probabilistic configurations (V_p); $E \subseteq V \times V$ is the set of edges; $\mathcal{K} : V_p \times V \to [0,1]$ is the probability function such that $(v, v') \notin E$ implies $\mathcal{K}(v, v') = 0$ and for every $v \in V_p$ we have $\sum_{v' \in V} \mathcal{K}(v, v') = 1$; and the pair $\langle \alpha, O \rangle$ defines the *goal*: $\alpha : V \to [0..k]$ is the parity condition, and $O : V \to (\{>, \geq\} \times [0,1]) \cup \{\perp\}$ marks the obligations, with the symbol $\perp$ denoting no obligation.

Obligations are statements applied to some configurations that impose constraints on the winning paths that depart from them. An obligation has the form $>x$ or $\geq x$, where $x \in [0,1]$, so as to indicate that the measure of the paths starting from that configuration must be greater than, or greater than or equal to, a given value x. Fixing a pair of strategies – σ for Player 0 and π for Player 1 – and a prefix of configurations $w \in V^+$, the game is reduced to only probabilistic vertices and, hence, can be seen as a Markov chain enriched with a *goal* $\langle \alpha, O \rangle$, that is, a winning condition and a set of obligations. We denote such structure as $G^{w(\sigma,\pi)}$. Value and winner of G are decided by analysing $G^{w(\sigma,\pi)}$ using the notion of *choice set*.

A choice set is the set of finite paths that extend the prefix w and end in a configuration with an obligation that can be met. It must be extended through infinite paths that either reach another obligation or never reach another obligation and are winning. The measure of the choice set is the measure of these infinite paths and determines the value of the game on every configuration for each player, denoted as $val_i(v)$ for $i \in \{0, 1\}$. We refer the reader to [9] for further details concerning the measure of choice sets and the value of obligation parity Markov chains. We write the value of game G on configuration v as $val_G(v)$ and we define it as the value for Player 0 on v.

Player 0 wins the game G from prefix w with a value of 1 if for every value $r < 1$ there exists a strategy σ such that for all Player 1's strategies π in the corresponding Markov chain $G^{w(\sigma,\pi)}$ it is possible to determine a choice set whose measure is at least r.

2.3 The μ^p-Calculus

The μ^p-calculus [7] is an extension of Kozen's μ-calculus [16] that allows one to specify properties that are bounded by a specific probability. This is done through the distinction between qualitative (Φ) and quantitative (Ψ) formulas that are evaluated to values in the sets $\{0, 1\}$ and $[0, 1]$, respectively. A μ^p-calculus sentence is qualitative and might contain one or more quantitative sub-formulas within a probabilistic quantification operator $[\cdot]_J$. The operator $[\cdot]_J$ checks whether the value of the enclosed formula satisfies the bound J and gets the value 1 or 0 accordingly. The syntax of the μ^p-calculus is given by the following BNF grammar.

$$\begin{aligned}
J &::= \{\geq, >\} \times [0,1] \\
\Phi &::= p \mid \neg p \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid [\Psi]_J \mid \mu X_i.\varphi \mid \nu X_i.\varphi \\
\Psi &::= \Phi \mid X_i \mid \psi_1 \wedge \psi_2 \mid \psi_1 \vee \psi_2 \mid \bigcirc \psi \mid \mu X_i.\psi \mid \nu X_i.\psi
\end{aligned}$$

Fixed point formulas are of the form $\sigma X_i.\varphi$, where $\sigma \in \{\mu, \nu\}$ and X_i is a variable in the set $\mathcal{V} = \{X_0, X_1, ...\}$. Variable X_i is bound by the fixed point operator to the formula φ, which we also denote by $\varphi(X_i)$ or φ_{X_i}.

Semantics. The semantics of a μ^p-calculus formula φ is given with respect to a Markov chain M and an interpretation $\rho : \mathcal{V} \to (S \to [0,1])$. That is, ρ associates a function from locations to values in the domain $[0,1]$ with each variable X_i appearing in φ. Therefore, the semantics is a mapping of type $S \to [0,1]$ denoted by $[\![\varphi]\!]^{\rho}_{M}$ and defined as follows:

$$\begin{array}{rclrcl}
[\![p]\!]^{\rho}_{M} & = & \chi_{L(p)} & [\![\neg p]\!]^{\rho}_{M} & = & 1-\chi_{L(p)} \\
[\![\varphi_1 \wedge \varphi_2]\!]^{\rho}_{M} & = & \min\{[\![\varphi_1]\!]^{\rho}_{M}, [\![\varphi_2]\!]^{\rho}_{M}\} & [\![\varphi_1 \vee \varphi_2]\!]^{\rho}_{M} & = & \max\{[\![\varphi_1]\!]^{\rho}_{M}, [\![\varphi_2]\!]^{\rho}_{M}\} \\
[\![X]\!]^{\rho}_{M} & = & \rho(X) & & & \\
 & & & [\![[\varphi]_J]\!]^{\rho}_{M} & = & \begin{cases} 1 & \text{If } [\![\varphi]\!]^{\rho}_{M}(s)J \\ 0 & \text{Otherwise} \end{cases} \\
[\![\bigcirc\varphi]\!]^{\rho}_{M} & = & \lambda s. \sum_{s'} P(s,s')[\![\varphi]\!]^{\rho}_{M}(s') & & & \\
[\![\mu X.\varphi]\!]^{\rho}_{M} & = & \mathrm{lfp}(\lambda f. [\![\varphi]\!]^{\rho[f/X]}_{M}) & [\![\nu X.\varphi]\!]^{\rho}_{M} & = & \mathrm{gfp}(\lambda f. [\![\varphi]\!]^{\rho[f/X]}_{M})
\end{array}$$

Where $\chi_{L(p)}$ is the function that associates to a location s the value 0 if $s \notin L(p)$, or the value 1 if $s \in L(p)$. The only elements of the calculus that are evaluated exclusively to values in the set $\{0,1\}$ are p, $\neg p$, and $[\cdot]_J$. All the other operators get real values in $[0,1]$, thus, can specify both quantitative and qualitative properties depending on their nested sub-formulas.

The alternation depth of a formula φ, denoted by $ad(\varphi)$, is the maximum number of fixed point operators that occur nested and alternated [12, 7]. In addition to the alternation depth, with every μ^p-calculus sub-formula ψ of φ is associated a colour $c(\psi)$. If ψ is a greatest fixed point then $c(\psi) = 2\big(ad(\varphi)-ad(\psi)\big)$; if ψ is a least fixed point then $c(\psi) = 2\big(ad(\varphi)-ad(\psi)\big)+1$; and in every other case $c(\psi) = 2ad(\varphi) - 1$.

Game Semantics [7]. Given a μ^p-calculus formula φ and a Markov chain M, we construct an obligation game $G_{M,\varphi}$ and we refer to such game as *semantics game*. The value on configuration (s,φ) equals the semantics of the μ^p-formula φ over the location s of the Markov chain. We say that M *satisfies* φ, denoted $M \models \varphi$ iff $[\![\varphi]\!]^{\rho}_{M}(s^{in}) = 1$, which in terms of game semantics corresponds to $M \models \varphi$ iff $val_{G_{M,\varphi}}(s^{in},\varphi) = 1$.

2.4 p-Automata

A p-automaton A is an alternating parity automaton that reads Markov chains as input [9]. From a state q, the p-automaton reads a location s of a Markov chain M and, according to the labelling $L(s)$, performs a transition. Since Markov chains have probabilities and the paths starting from a location s are characterised by a measure, the p-automaton A might need to mark the states by a bound J. The bound J is an element of the set $(\{\geq, >\} \times [0,1])$ and, analogously to the obligations over configurations of games, imposes a constraint over the measure of the accepted paths starting in s.

For the set of states Q, we denote by $[\![Q]\!]_{>}$ the set of states $q \in Q$ that are marked by a bound J defined as $\{[\![q]\!]_J \mid q \in Q,\ J \in (\{\geq,>\} \times [0,1])\}$. Moreover, we denote by $B^+(X)$ the set of *positive boolean formulas* over elements x in the set X, given by the following grammar:

$$\theta ::= x \mid true \mid false \mid \theta_1 \wedge \theta_2 \mid \theta_1 \vee \theta_2$$

Given a formula $\theta \in B^+(X)$ its *closure* $\mathsf{cl}(\theta)$ is the set of all sub-formulas of θ defined according to the grammar above. For a set Θ of formulas, the closure is computed as $\mathsf{cl}(\Theta) = \bigcup_{\theta \in \Theta} \mathsf{cl}(\theta)$.

▶ **Definition 2** (p-Automata). A p-automaton A over the set AP of atomic propositions is defined by the tuple:

$$A = (\Sigma, Q, \varphi^{in}, \delta, \Omega)$$

where $\Sigma = 2^{AP}$ is a finite input alphabet, Q is a possibly infinite set of states, $\varphi^{in} \in B^+([\![Q]\!]_{>})$ is the initial condition, $\delta : Q \times \Sigma \to B^+(Q \cup [\![Q]\!]_{>})$ is the transition function, and $\Omega : Q \to [0 \dots k]$ is the parity acceptance condition.

Acceptance Game [9]. The set of Markov chains accepted by a p-automaton A is the language of A, denoted by $\mathcal{L}(A)$. Acceptance of a Markov chain M by A is decided through the obligation game $G_{M,A}$. The Markov chain M is accepted if the configuration (s^{in}, φ^{in}) has value 1 in $G_{M,A}$. That is, $M \in \mathcal{L}(A)$ iff $val_{G_{M,A}}(s^{in}, \varphi^{in}) = 1$.

3 Vectorial μ^p-Calculus

We introduce the vectorial form as an alternative syntax for formulas in μ^p-calculus. This form exposes the distinction between the fixpoint operators, which appear as a prefix of the formula, and the modal formulas that they bind, allowing one to focus on the modal properties rather than on an intricate nesting of fixed-point terms. Through this syntax, the alternation depth of a sentence is easier to identify, as the number of pairwise distinct fixpoint operators within the prefix of the formula, and the most complex properties can be expressed in a succinct way.

Let F_i be the set of functions $(S \to [0,1])_i$ from locations to values in the unit interval, and φ_i be a modal μ^p-formula over the product lattice $(F_1 \times \ldots \times F_n)^m$ with range F_i, i.e. φ_i takes m vectors of n variables $\langle X_1^1, \ldots, X_n^1, \ldots, X_1^m, \ldots, X_n^m\rangle$ and evaluates to a single function in F_i. If we consider the vector $\vec{\varphi}$ of all modal terms $\langle \varphi_1, \ldots, \varphi_n\rangle$, each of which has range F_i, then, $\vec{\varphi}$ can be seen as a mapping of type $\vec{\varphi} : (F_1 \times \ldots \times F_n)^m \to F_1 \times \ldots \times F_n$, whose monotonicity derives from the monotonicity of each single component and for which, by the Knaster-Tarski theorem, least and greatest fixpoints are always defined. For $m = 1$, we denote as $\mu\vec{X}.\vec{\varphi}$, resp. $\nu\vec{X}.\vec{\varphi}$, the least, resp. greatest, fixpoint of the mapping $\vec{\varphi}$, as a compact notation for:

$$\sigma \begin{pmatrix} X_1 \\ \vdots \\ X_n \end{pmatrix} . \begin{pmatrix} \varphi_1(X_1, \ldots, X_n) \\ \vdots \\ \varphi_n(X_1, \ldots, X_n) \end{pmatrix} = \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix} .$$

Vectorial μ^p-calculus has the same expressive power as scalar μ^p-calculus. By the application of the Bekič principle [1], whose effect is to push the fixpoint operators inwards, every vectorial formula $\sigma\vec{X}.\vec{\varphi}$ can be reduced to a vector $\vec{f}$ of scalar formulas $\langle f_1, \ldots, f_n\rangle$.

Semantics. The semantics of a vectorial formula $\sigma\vec{X}.\vec{\varphi}$ is the vector of semantics: $[\![\sigma\vec{X}.\vec{\varphi}]\!]_M^\rho = [\![\sigma X_1.\vec{\varphi}]\!]_M^\rho \times \ldots \times [\![\sigma X_n.\vec{\varphi}]\!]_M^\rho$. We use the projection operator on vectors $\downarrow_i$ to select the i-th component: $[\![\sigma\vec{X}.\vec{\varphi}\downarrow_i]\!]_M^\rho = [\![\sigma X_i.\vec{\varphi}]\!]_M^\rho = f_i$. The meaning of choosing a component is to define an *entry point* to the computation performed by the vectorial formula [6].

Game Semantics. As for standard μ^p-calculus, the semantics of a vectorial sentence can be defined by an obligation game $G_{M,\vec{\varphi}}$ that results from the composition of sub-games for each of the n-th components of the vectorial formula. The value of such game is a vector of values, each corresponding to $val_{G_{M,\vec{\varphi}}}(s, \varphi_i)$.

4 From μ^p-Calculus to p-Automata

We show that every qualitative μ^p-calculus formula can be translated into an equivalent p-automaton. The translation relies on the formulas satisfying some syntactic requirements.

Well-Formedness The set of well-formed μ^p-calculus formulas is semantically equivalent to the standard form of the calculus; however, it poses some constraints on the syntax allowing

$$
\begin{aligned}
\delta(p,a) &= \begin{cases} \top & \text{if } p \in a \\ \bot & \text{if } p \notin a \end{cases} &\qquad \delta_\epsilon(p,c) &= p \\
& & \delta_\epsilon(\neg p,c) &= \neg p \\
\delta(\neg p,a) &= \begin{cases} \top & \text{if } p \notin a \\ \bot & \text{if } p \in a \end{cases} & \delta_\epsilon(\psi_1 \vee \psi_2,c) &= \delta_\epsilon(\psi_1,c) \vee \delta_\epsilon(\psi_2,c) \\
& & \delta_\epsilon(\psi_1 \wedge \psi_2,c) &= \delta_\epsilon(\psi_1,c) \wedge \delta_\epsilon(\psi_2,c) \\
\delta(\top,a) &= \top & \delta_\epsilon\big(\sigma X.\varphi(X),c\big) &= \delta_\epsilon\big(\varphi(X),c\big) \\
\delta(\bot,a) &= \bot & \delta_\epsilon\big(X,c\big) &= \delta_\epsilon\big(\varphi(X),c(X)\big) \\
\delta\big((\bigcirc\psi,c),a\big) &= \delta_\epsilon(\psi,c) & \delta_\epsilon(\bigcirc\psi,c) &= (\bigcirc\psi,c) \\
& & \delta_\epsilon([\bigcirc\psi]_J,c) &= [\![(\bigcirc\psi,c)]\!]_J
\end{aligned}
$$

Figure 1 Transition of A_φ.

for the conversion into p-automata. We require that the variables be bound exactly once and that well-formed formulas be *guarded*; that is, all the occurrences of a variable must be in the scope of a next modality, which is itself in the scope of a fixpoint operator. To this end, formulas can be re-written in guarded form, as explained in [17, 5], by the iterated replacement of every open occurrence of a variable X by *false* in least fixed point formulas and by *true* in greatest fixed point formulas. Also, we consider the probabilistic quantification operator over a bound J that is restricted to the set $(\{\geq,>\} \times [0,1]) \setminus \{\geq 0, > 1\}$; this restriction does not affect the expressive power of the language since properties of the form $[\cdot]_{\geq 0}$ and $[\cdot]_{>1}$ correspond to *true* and *false* statements. Moreover, we are interested in formulas where all the instances of the probabilistic quantification operator $[\cdot]_J$ are directly applied to a next $\bigcirc$. This requirement is necessary because the statements enclosed in a probabilistic operator will translate into states of the corresponding automaton that performs a transition moving to read the next locations of the model. One can achieve this form by transforming the formulas according to the equivalences stated in the lemma below.

▶ **Lemma 3.** *The following μ^p-calculus formulas are semantically equivalent.*

$$
\begin{aligned}
[p]_J &\equiv p \\
[\neg p]_J &\equiv \neg p \\
[\varphi_1 \wedge \varphi_2]_J &\equiv [\varphi_1]_J \wedge [\varphi_2]_J \\
[\varphi_1 \vee \varphi_2]_J &\equiv [\varphi_1]_J \vee [\varphi_2]_J \\
[\sigma X.\varphi(X)]_J &\equiv \big[\varphi\big(\sigma X.\varphi(X)\big)\big]_J
\end{aligned}
$$

Proof. The proof arises from the semantics of the μ^p-calculus and the fixed point axioms. ◀

Translation Let φ be a qualitative well-formed μ^p-calculus formula over the set AP of atomic propositions. The p-automaton A_φ is the tuple $(2^{AP}, Q, \delta, \varphi^{in}, \Omega)$, where 2^{AP} is the alphabet, Q is the set of states $\{\bot,\top\} \cup \{p, \neg p, (\bigcirc\psi, c) \mid \text{ for all } p, \neg p, \bigcirc\psi \in sub(\varphi) \text{ and } c \in [0 \ldots 2ad(\varphi)-1]\}$, the transition function δ (and the auxiliary function δ_ϵ) is defined by the rules in Figure 1.

The initial condition φ^{in} is the expression $\delta_\epsilon\big(\varphi, c(\varphi)\big)$; the priority Ω is $\Omega(\bot) = 1$, $\Omega(\top) = 0$, $\Omega(\bigcirc\psi, c) = c$, and maximum colour otherwise.

Transitions of A_φ always consume the input label a of a location in a Markov chain and move forward to its successors. The computation starts from the states within the initial condition φ^{in}. From a state p or $\neg p$, the p-automaton reads the current label a and moves to one of the special states $\top$ or $\bot$ defining an infinite computation that is accepting or rejecting, respectively. When in a state $(\bigcirc\psi, c)$ reading a label a, the p-automaton moves to a new set of states determined by unfolding the formula ψ through the epsilon transition function δ_ϵ. The outcome of δ_ϵ, as well as of δ, is a positive boolean formula over states q and bounded states $[\![q]\!]_J$ that represents the requirement from the system. States within such formula are evaluated over the successor locations in M. Acceptance of a Markov chain M by the p-automaton A_φ is decided by the *acceptance game* G_{M,A_φ}: if the value in such game of the initial configuration is 1 M is accepted, otherwise, it is rejected.

The following theorem states the correctness of the translation from μ^p-calculus into p-automata.

▶ **Theorem 4.** *Let φ be a well-formed μ^p-calculus formula and A_φ the automaton resulting from its translation. Then, φ and A_φ are equivalent: the set of Markov chains that satisfy the formula φ corresponds to the language $\mathcal{L}(A_\varphi)$ recognised by the p-automaton A_φ. That is, $M \models \varphi$ iff $M \in \mathcal{L}(A_\varphi)$.*

The proof is conducted by showing that for all Markov chains M the *acceptance game* G_{M,A_φ} simulates the *semantics game* $G_{M,\varphi}$. In particular, there is a mapping between prefixes of paths in the two games, within which probabilities, obligations, and infinite winning sets are preserved. Therefore, the acceptance game has the same value as that of the semantics game, leading the p-automaton A_φ to accept all the Markov chains that satisfy the formula φ.

5 From p-Automata to μ^p-Calculus

We show that every p-automaton can be translated into an equivalent μ^p-calculus formula. Transitions of p-automata define an infinite computation tree whose nodes are states marked by priorities. The sequence of such priorities within the paths of the tree determines whether the computation is accepted or not: infinitely many visits to a minimal even priority mean acceptance, whereas passing infinitely often through a minimal odd priority causes rejection. All these elements have their analogue in μ^p-calculus: transitions and modal formulas, applying a transition from a state and passing through variables, odd/even priorities and least/greatest fixpoints, and levels of priorities and nesting of fixpoint formulas. We exploit this analogy in the conversion from p-automata to μ^p-calculus, using the syntax that most emphasises the role of each component, the vectorial form.

Translation Let A be a p-automaton over the set AP of atomic propositions defined by the tuple $(2^{AP}, Q, \delta, \varphi^{in}, \Omega)$, with n the number of states of the automaton. Let $i_1, \ldots, i_m$ be the ordered chain of increasing priorities in the set $\bigcup_{q \in Q} \Omega(q)$. For each index $j \leq m$ we introduce a vector $\vec{X}_{i_j}$ of $n+1$ fresh variables. The first variable of each j-th vector is a dummy variable that refers to the initial condition of the automaton, and we indicate it as $X^{in}_{i_j}$. The remaining n variables of each j-th vector bind the n formulas corresponding to the transitions that the p-automaton A performs from each of its n states; we write such variables as $X^1_{i_j}, \ldots, X^n_{i_j}$. Accounting for the initial condition of the automaton as the first component of these vectors allows one to retrieve the semantics of the resulting formula as the semantics of its first element. As a consequence, the ordering of the other n components is not relevant. In order to turn states into variables we use the following function t:

$$\begin{aligned} t(\theta_1 \vee \theta_2) &= t(\theta_1) \vee t(\theta_2) \\ t(\theta_1 \wedge \theta_2) &= t(\theta_1) \wedge t(\theta_2) \\ t([\![q]\!]_J) &= [X^q_{\Omega(q)}]_J \\ t(q) &= X^q_{\Omega(q)} \end{aligned}$$

We employ this function in the definition of the vector $\vec{\varphi}$ of modal μ^p-formulas. The first component of $\vec{\varphi}$ is $t(\varphi^{in})$, the other n components are denoted by φ^k for $k \leq n$ and are specified by the following modal formula

$$\varphi^k \; = \; \bigvee_{a \in 2^{AP}} \left(\bigcirc\, t\left(\delta\left(q^k, a\right)\right) \wedge \bigwedge_{p \in a} p \; \wedge \bigwedge_{p \notin a} \neg p \right).$$

Finally, the vectorial μ^p-calculus formula $\vec{\varphi}_A$ is defined as the prefix chain of ordered fixpoints and vectors enclosing $\vec{\varphi}$, where $\sigma_{i_j} = \mu$ if i_j is odd or $\sigma_{i_j} = \nu$ if i_j is even:

$$\vec{\varphi}_A \; = \; \sigma_{i_1} \begin{pmatrix} X^{in}_{i_1} \\ X^1_{i_1} \\ \vdots \\ X^n_{i_1} \end{pmatrix} \ldots \sigma_{i_m} \begin{pmatrix} X^{in}_{i_m} \\ X^1_{i_m} \\ \vdots \\ X^n_{i_m} \end{pmatrix} . \begin{pmatrix} t\left(\varphi^{in}\right) \\ \varphi^1 \\ \vdots \\ \varphi^n \end{pmatrix}.$$

The semantics of the vectorial formula $\vec{\varphi}_A$ for a Markov chain M and a valuation ρ is the semantics of its first component over the initial location s^{in} of M and it is equivalent to the value of the configuration $\left(s^{in}, t\left(\varphi^{in}\right)\right)$ in the semantics game $G_{M,\vec{\varphi}_A}$. That is, $[\![\vec{\varphi}_A \downarrow_1]\!]^{\rho}_M(s^{in}) \; = \; val_{G_{M,\vec{\varphi}_A}}\left(s^{in}, t\left(\varphi^{in}\right)\right)$.

It is worth noticing that only a maximum of n out of $m \times (n+1)$ variables are bound within the formula $\vec{\varphi}_A$. Therefore, $\vec{\varphi}_A$ can be seen as a system of $n+1$ equations in n variables that can be reduced by substitution and Gauss elimination techniques to a single scalar μ^p-calculus sentence (see [23]). In particular, it is sufficient to derive a solution, or expression, for each of the n variables and by syntactical substitution embed such expressions in $t(\varphi^{in})$. However, we are interested in giving a characterization in terms of semantics game $G_{M,\vec{\varphi}_A}$, in which the effect of the syntactical substitution of variables is handled by the edges connecting configurations (s, X^k) to (s, φ^k).

▶ **Theorem 5.** *Let A be a p-automaton over the set AP of atomic propositions and $\vec{\varphi}_A$ the vectorial μ^p-calculus formula resulting from its conversion. Then, A and $\vec{\varphi}_A$ are equivalent: the set of Markov chains that constitute the language $\mathcal{L}(A)$ recognised by the p-automaton A coincides with the set of Markov chains that satisfy the vectorial formula $\vec{\varphi}_A$. That is, $M \in \mathcal{L}(A)$ iff $M \models \vec{\varphi}_A$.*

Similarly to the case of the inverse translation, the proof shows that the semantics game $G_{M,\vec{\varphi}_A}$ for the vectorial formula $\vec{\varphi}_A$ simulates the acceptance game $G_{M,A}$ for the original p-automaton A. As a result, the two games have the same value and, therefore, the Markov chains that satisfy the formula $\vec{\varphi}_A$ are exactly those that are accepted by A.

6 Conclusion

The aim of this paper was to investigate the connection between μ^p-calculus and p-automata and to assess their equivalence in expressive power. We presented the notion of well-formed formulas as a necessary preliminary step for their translation into p-automata. We showed that for every well-formed μ^p-calculus sentence there exists an equivalent p-automaton that recognises exactly all the Markov chains that model the formula. Conversely, we proved that for every p-automaton there is an equivalent μ^p-formula, in vectorial syntax, that is satisfied by the same Markov chains that form the language of the p-automaton.

Throughout this work, obligation games have played a key linking role in defining the semantics of the structures resulting from the conversions and, therefore, proving the correctness of our claims.

References

1 Andreé Arnold and Damian Niwinski. Rudiments of μ-calculus, volume 146 of. *Studies in Logic and the foundations of mathematics*, 2001.

2 Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

3 Patrick Blackburn, Johan van Benthem, and Frank Wolter. *Handbook of Modal Logic*, volume 3. Elsevier, 2007.

4 Julian Bradfield and Colin Stirling. Modal μ-calculi. *Studies in logic and practical reasoning*, 3:721–756, 2007.

5 Julian Bradfield and Igor Walukiewicz. The μ-calculus and model-checking. *Handbook of Model Checking. Springer-Verlag*, pages 35–45, 2015.

6 Florian Bruse, Oliver Friedmann, and Martin Lange. On guarded transformation in the modal μ-calculus. *Logic Journal of the IGPL*, 23(2):194–216, 2014.

7 Pablo Castro, Cecilia Kilmurray, and Nir Piterman. Tractable probabilistic μ-calculus that expresses probabilistic temporal logics. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 30. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

8 Claudia Cauli and Nir Piterman. Equivalence of probabilistic μ-calculus and p-automata. In A. Carayol and C. Nicaud, editors, *Implementation and Application of Automata: 22nd International Conference, CIAA 2017, Marne-la-Vallée, France, June 27-30, 2017*, pages 64–75, Cham, 2017. Springer.

9 Krishnendu Chatterjee and Nir Piterman. Obligation blackwell games and p-automata. *Journal of Symbolic Logic*, 82(2):420–452, 2017.

10 Rance Cleaveland, S Purushothaman Iyer, and Murali Narasimha. Probabilistic temporal logics via the modal μ-calculus. *Theoretical Computer Science*, 342(2-3):316–350, 2005.

11 E Allen Emerson and Charanjit S Jutla. Tree automata, μ-calculus and determinacy. In *Foundations of Computer Science, 1991. Proceedings., 32nd Annual Symposium on*, pages 368–377. IEEE, 1991.

12 E Allen Emerson and C-L Lei. Efficient model checking in fragments of the propositional μ-calculus. In *Proceedings of the First Annual IEEE Symposium on Logic in Computer Science, LICS*, pages 267–278, 1986.

13 E Grädel, W Thomas, and T Wilke. Automata, logics, and infinite games: A guide to current research, volume 2500 of lncs. *Springer*, 2002.

14 Michael Huth and Marta Kwiatkowska. Quantitative analysis and model checking. In *Logic in Computer Science, 1997. LICS'97. Proceedings., 12th Annual IEEE Symposium on*, pages 111–122. IEEE, 1997.

15 David Janin and Igor Walukiewicz. Automata for the modal μ-calculus and related results. *Mathematical Foundations of Computer Science 1995*, pages 552–562, 1995.

16 Dexter Kozen. Results on the propositional μ-calculus. *Theoretical computer science*, 27(3):333–354, 1983.

17 Orna Kupferman, Moshe Y Vardi, and Pierre Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM (JACM)*, 47(2):312–360, 2000.

18 Matteo Mio. Game semantics for probabilistic modal μ-calculi, 2012.

19 Matteo Mio. Probabilistic modal μ-calculus with independent product. *Logical Methods in Computer Science*, 2012.

20 Matteo Mio and Alex Simpson. Lukasiewicz μ-calculus. *arXiv:1510.00797*, 2015.

21 Damian Niwinski. Fixed points vs. infinite generation. In *Logic in Computer Science, 1988. LICS'88., Proceedings of the Third Annual Symposium on*, pages 402–409. IEEE, 1988.

22 Damian Niwiński. Fixed point characterization of infinite behavior of finite-state systems. *Theoretical Computer Science*, 189(1-2):1–69, 1997.

23 Klaus Schneider. *Verification of reactive systems: formal methods and algorithms*. Springer, 2004.

24 Thomas Wilke. Alternating tree automata, parity games, and modal μ-calculus. *Bulletin of the Belgian Mathematical Society*, 8(2):359, 2001.