Probabilistic Regular Graphs

Nathalie Bertrand, Christophe Morvan

Probabilistic regular graphs

Nathalie Bertrand
INRIA Rennes Bretagne Atlantique
nathalie.bertrand@inria.fr

Christophe Morvan
Université Paris-Est
INRIA Rennes Bretagne Atlantique
christophe.morvan@univ-paris-est.fr

Abstract. Deterministic graph grammars generate regular graphs, that form a structural extension of configuration graphs of pushdown systems. In this paper, we study a probabilistic extension of regular graphs obtained by labelling the terminal arcs of the graph grammars by probabilities. Stochastic properties of these graphs are expressed using PCTL, a probabilistic extension of computation tree logic. We present here an algorithm to perform approximate verification of PCTL formulae. Moreover, we prove that the exact model-checking problem for PCTL on probabilistic regular graphs is undecidable, unless restricting to qualitative properties. Our results generalise those of [8], on probabilistic pushdown automata, using similar methods combined with graph grammars techniques.

1 Introduction

Formal methods have proven their importance in the validation of hardware and software systems. In order to represent real systems more accurately, several aspects need to be reflected in the model. Recursion and random events are examples of such extra features and lead to complex models that incorporate two sources of complexity: probabilities and infinite state space. For each of these features independently, verification techniques have been established.

Infinite state systems, on the one hand, cover a large range of expressive power. Among them pushdown systems offer a simple infinite framework by extending finite state systems with a stack. Despite the fact that their configurations graph is infinite, pushdown systems enjoy several interesting properties. In particular, the reachability problem is decidable, and the reachability set is effectively regular [3]. Moreover, monadic second order logic (MSO) [11] is decidable over the graph of configurations for pushdown automata. Alternatively, the configurations graphs of pushdown automata can be generated by deterministic graph grammars, introduced by Courcelle [7]. Deterministic graph grammars generate *regular graphs* which also have decidable MSO [7], and which characterise the same structures as pushdown systems [6] when restricting to finite degree. We advocate that these grammars offer a simple presentation and emphasize the structural properties of graphs. Indeed, contrary to pushdown automata, graph grammars are more robust to transformations. Precisely, many transformations of pushdown automata affect the configurations graph, and thus its stucture-based properties. On the contrary, graph grammars allow for transformations in the representations which preserve the structure. Indeed, most graph grammar transformations presented in [5] preserve, up to isomorphism, the generated graph. Using such representations thus seems promising in order to express structural properties of systems.

Probabilistic systems, on the other hand, also raised intensive research concerning verification, starting with model-checking algorithms for Markov chains, and Markov decision processes for various logics. In the last decade, models combining probabilities and infinite-state spaces have been investigated. Examples of such models are probabilitic pushdown systems and probabilistic lossy channel systems. These systems are finitely described and generate infinite Markov chains on which one can express probabilistic properties, for example using the probabilistic extension of CTL, PCTL [9]. This logic allows

Submitted to:
INFINITY 10

to express, *e.g.*, the probability of satisfying a given CTL path formula. More generally, PCTL can be seen as a variant of CTL where the usual forall quantifier is replaced with a probabilistic comparison to a threshold: the whole state formula is satisfied if the probability of the set of executions satisfying the CTL path formula meets the constraint expressed by the threshold. A restricted fragment of this logic, called *qualitative* PCTL is obtained when allowing values 0 and 1 only for the thresholds. In constrast, the general case (where threshold values are arbitrary) is referred to as *quantitative* PCTL. The model-checking problem for probabilistic logics over infinite Markov chains generated by probabilistic lossy channel systems or probabilistic pushdown automata is a natural and deeply investigated issue. Concerning probabilistic pushdown automata, a series of papers established fundamental model checking results [2, 8, 10, 1], some of the most significant ones being the decidability of the model checking of qualitative PCTL formulae, and the undecidability of the quantitative version.

In this paper, we consider a probabilistic extension of regular graphs. To this aim, we define probabilistic graph grammars as graph grammars where terminal arcs are labelled with probabilities. Probabilistic graph grammars hence generate infinite-state Markov chains, and form a natural generalisation of probabilistic pushdown automata. For these models, we extend the results of [8] concerning the model-checking of PCTL. Precisely, for probabilistic graph grammars we prove the decidability of the qualitative PCTL model-checking ; we detail how to approximate the probability of path formula ; and we prove the undecidability of the exact quantitative PCTL model-checking.

2 Regular graphs and probabilistic regular graphs

2.1 Hypergraphs and graphs

Let F be a ranked alphabet, and $\rho : F \to \mathbb{N}$ its ranking function that assigns to each element of F its *arity*. We denote by F_n the set of symbols of arity n. Given V an arbitrary set of vertices, a *hypergraph* G is a subset of $\cup_{n\geq 1} F_n V^n$. The vertex set of G, denoted V_G, is defined as the set $V_G = \{v \in V \mid FV^*vV^* \cap G \neq \emptyset\}$. In our setting, this set is always countable. An element of F_nV^n is an *hyperarc* of arity n, denoted by $f\ v_1\ v_2\ \cdots\ v_n$.

Graphs form a restricted class of hypergraphs where hyperarcs have arity at most 2. Precisely, a *graph* G over V is a subset of $F_2VV \cup F_1V$. For $a \in F_2$, and $s,t \in V$, $ast \in G$ is an *arc* of G with source s, target t and label a. For $a \in F_1$ and $s \in V$, if as is an element of G, a is referred to as the *colour* of vertex s (observe that a vertex may have several colours). $Dom(G)$, $Im(G)$ and V_G denote respectively the set of sources, targets and vertices of G. The *in-degree* (*resp. out-degree*) of a vertex v is the number of arc having source (*resp.* target) v; its *degree* is the sum of the in and out-degrees. The transition relation underlying G is composed of transitions $s \xrightarrow{a}_G t$ for $ast \in G$. A *path* in G is a finite sequence of transitions $v_1 \xrightarrow{a_1} v_2 \cdots \xrightarrow{a_{n-1}} v_n$, also noted $v_1 \overset{a_1\cdots a_n}{\Longrightarrow}_G v_n$.

A *graph morphism* from G to G', is a mapping $g : V_G \to V_{G'}$ such that for all $u,v \in V_G$, $u \xrightarrow{a}_G v$ implies $g(u) \xrightarrow{a}_{G'} g(v)$. Such a morphism is an *isomorphism* if g is a bijection, and its inverse is also a morphism.

2.2 Graph grammars

Graph grammars are a convenient tool to represent graph transformations. Starting from a hyperarc, the axiom, and using rewriting rules, these grammars generate families of infinite graphs that enjoy interesting properties (for example the decidability of MSO theory, or the fact that they generate context-free languages). Graphs generated by graph grammars form a slight extension of the graphs of configurations for pushdown automata, namely such a graph may have vertices of infinite degree (still there are only finitely many distinct degrees). A motivation for generating these graphs using graph grammars rather than pushdown automata is to emphasize the structural properties of the obtained graphs, since they are defined up to isomorphism. In particular, stochastic properties of Markov chains (like probability of a path or a set of paths) are invariant under graph isomorphism, this justifies the use of structural characterizations such as graph grammars.

Definition 2.1. A *hypergraph grammar* (HR-grammar for short), is a tuple $\mathscr{G} = (N,T,R,Z)$, where:

- N and T are two ranked alphabets of *non-terminal* and *terminal* symbols, respectively;
- $Z \in N$ is a 0-arity non-terminal, the *axiom*;
- R is a set of *rewriting rules* assigning to each non-terminal $A \in N$ a pair (H_A, ι_A) where H_A is a finite hypergraph, and $\iota_A : \{1,\cdots,\rho(A)\} \hookrightarrow V_{H_A}$ is an injective mapping associating to each position in an hyperedge labelled A a vertex in H_A.

Example 2.2. Figure 2.1 presents an example of a HR-grammar. Formally, it is defined by $\mathscr{G} = (\{Z\}_0 \cup \{A\}_2, \{V_1,V_2\}_1 \cup \{a,d\}_2, \{(H_Z,\iota_Z),(H_A,\iota_A)\}, Z)$. Non-terminal Z (*resp.* A) is the only arity 0 (*resp.* 2) non-terminal symbol; $\{V_1,V_2\}$ (*resp.* $\{a,d\}$) are the two colours (*resp.* arc-labels); hypergraphs H_Z, H_A and injection ι_A are represented in the first part of the figure. For simplicity, $\overline{V_1}$ denotes the absence of colour V_1. The injection ι_A is used to identify vertices of H_A with vertices of an arc labelled A in the rewriting process defined later on.

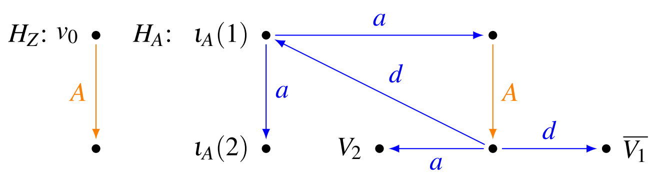


Figure 2.1: An example of a graph grammar.

Remark 2.3. Note that Definition 2.1 corresponds to the classical definition of deterministic hypergraph grammars [7, 5], since there is exactly one rewriting rule for each non-terminal symbol. Moreover, we implicitely assume that terminal symbols have arity one or two (Markov chains are transitions systems, thus arities greater than 2 do not make sense in this context). This way, the generated graphs are coloured graphs (or transition systems where transitions and states are labelled).

Let $\mathscr{G} = (N,T,R,Z)$ be a hypergraph grammar. Given $A \in N$ a non-terminal, we denote by $\underset{R,A}{\longrightarrow}$ the rewriting relation between hypergraphs with respect to the rule $(H_A, \iota_A) \in R$. Formally, a hypergraph M rewrites into M', written $M \underset{R,A}{\longrightarrow} M'$, if there exists a hyperarc $X = Av_1v_2\ldots v_p$ in M such that $M' = (M - X) \cup h(H_A)$ where h is an injective morphism that maps $\iota(i)$ to v_i and other vertices of H_A to vertices outside M. Intuitively, M' is obtained from M by replacing X (of non-terminal label A) with H_A. The rewriting relation extends to the *complete parallel rewriting* relation: the rewriting of each non-terminal simultenaously. We write $M \underset{R}{\Rightarrow} M'$ for the complete parallel rewriting of M into M'. In other words, all non-terminal hyperedges of M have been replaced in M' using their respective rewriting rules in R. The set of all images of a graph M by $\underset{R}{\Rightarrow}$ is denoted by $R[M]$. This set contains all isomorphic graphs obtained by applying the rules of R to M. For $n > 1$, this notation is extended inductively into $R^n[M] = \bigcup_{M' \in R^{n-1}[M]} R[M']$, it is the set of all isomorphic graphs obtained after n applications of the complete parallel rewriting.

Let N and T be sets of non-terminals, respectively terminals. Given h a hypergraph labelled by $N \cup T$, we denote by $[H]$ the set of terminal arcs and colours in H: $[H] = H \cap (T_2\ V_H\ V_H \cup T_1\ V_H)$. For $\mathscr{G} = (N,T,R,Z)$ a HR-grammar, the set of graphs generated by $\mathscr{G}$ is defined as follows:

$$\mathscr{G}^{\omega} = \left\{ \cup_{n \geq 0}[H_n] \mid H_0 = Z \wedge \forall n \geq 0, H_n \underset{R}{\Rightarrow} H_{n+1} \right\}$$

Note that if $H_n \underset{R}{\Rightarrow} H_{n+1}$, then $[H_n] \subseteq [H_{n+1}]$. Thus the set $\mathscr{G}^{\omega}$ contains graphs which are all isomorphic. A graph H is *generated* by $\mathscr{G}$ if it belongs to $\mathscr{G}^{\omega}$. Let $H \in \mathscr{G}^{\omega}$, for each vertex $v \in V_H$, we let $\mathsf{Lev}(v)$ be the *level* at which v is generated. Formally, $\mathsf{Lev}(v) = \min\{k \mid v \in [H_k]\}$. Furthermore, notation $\mathsf{Can}(v)$ stands for the *canonical* image of v in the finite set of vertices $\bigcup_{A \in N} V_{H_A}$. Assuming $H_{k-1} \xrightarrow{R,A} H'_{k-1}$ for some $A \in N$ and $v \in H'_{k-1}$, $\mathsf{Can}(v)$ is the unique vertex in H_A whose image by h is v. When vertex v is generated in H_A at the i-th position of an arc labelled by $B \in N$, we write $\mathsf{Can}(v) = (B,i)_A$. Observe that, since $v \notin H_{k-1}$, for each j, v is distinct from $\iota_A(j)$.

Example 2.4. Figure 2.1 presents an example of a HR-grammar, Figure 2.2 illustrates, starting from the axiom Z, two successive applications of the complete parallel rewriting (which coincides here with the rewriting of a single non-terminal) and the iteration of this process. In this example, each application of the rewriting rules adds new vertices as well as new arcs to the graph. Observe that the names of the vertices (except for v_0 that is distinguished) are not depicted, since they are not relevant to our purpose. Up to renaming of the vertices, there is a unique generated infinite graph.

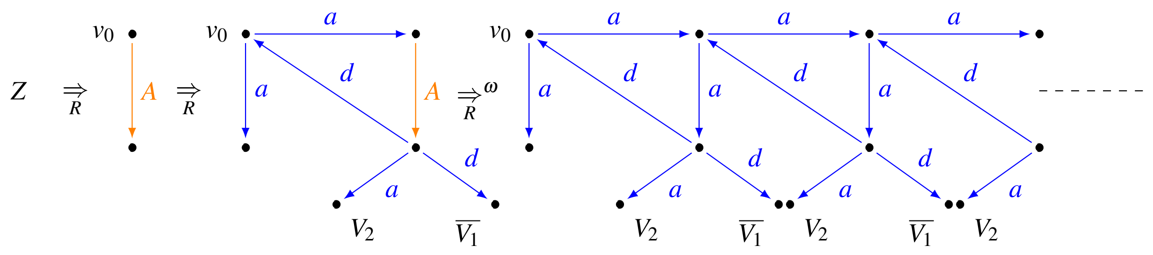

v_0
Z
$\underset{R}{\Rightarrow}$
A
a
d
$\underset{R}{\Rightarrow}^{\omega}$
V_2
$\overline{V_1}$

Figure 2.2: Application of successive complete parallel rewritings and the generated graph.

2.3 Basic Properties and Normal Forms for Regular Graphs

For any rule (H_A, ι_A), we say that the vertices $\iota_A(\{1,\cdots,\rho(A)\})$ are the *inputs* of H_A, and $\bigcup_{Y \in H_A \wedge Y(1) \in N_R} V_Y$ are the *outputs* of H_A. In particular, output vertices belong to non-terminal hyperedges.

Given a non-terminal $A \in N$, we denote by $\mathsf{Succ}(A)$ the set of non-terminals appearing in H_A.

Given a HR-grammar $\mathscr{G} = (N,T,R,Z)$ and a non-terminal hyperarc $X = Av_1v_2\ldots v_p$, we introduce notations R^{ω} (resp. $R^{\omega}[X]$) to denote a particular graph in $\mathscr{G}^{\omega}$ (resp. in $(\mathscr{G}[X])^{\omega}$ with $\mathscr{G}[X] := (N,T,R,X)$).

Let $\mathscr{G} = (N,T,R,Z)$ and $\mathscr{G}' = (N',T',R',Z')$ be two HR-grammars we say that $\mathscr{G}'$ is a *colouring* of $\mathscr{G}$ if, for any graphs $H \in \mathscr{G}^{\omega}$ and $H' \in \mathscr{G}'^{\omega}$, there is a graph isomorphism between H and H' which also preserves colours of H, and there is a *colour* in T'_1 which does not belong to T_1.

We conclude these preliminaries by giving a normal form for HR-grammars.

Theorem 2.5. *[5] Any regular hypergraph can be generated in an effective way by a complete outside grammar.*

The *complete outside* property ensures that the only input vertices that are also outputs are vertices of infinite degree. It also implies that each output vertex belongs to a single non-terminal hyperarc. This property enables one to identify efficiently grammars having vertices of infinite degree, and it also ensures that whenever there is no such vertex, inputs and outputs are distinct. In the sequel we assume that all HR-grammars we consider are complete outside.

2.4 Probabilistic Regular Graphs

In order to obtain a probabilistic graph from one generated by a HR-grammar, we define, for each HR-grammar $\mathcal{G}$, and each graph H in $\mathcal{G}^\omega$, the counting function $\# : V_H \times T_2 \to \mathbb{N}$, with $\#(v,a) = |\{v' \mid v \xrightarrow{a} v'\}|$, that associates with each pair (v,a) the number of a-labelled arcs originating from v. Observe that two distinct vertices v and v' in H have identical valuations for # as soon as $\mathsf{Can}(v) = \mathsf{Can}(v')$.

Definition 2.6 (Probabilistic graph grammar)**.** A *probabilistic hypergraph grammar* (PHR-grammar for short) $\mathcal{P}$, is a pair $(\mathcal{G},\mu)$ where $\mathcal{G} = (N,T,R,Z)$ is a HR-grammar, $\mu : T_2 \to [0,1]$ is a mapping, and for each vertex $v \in R^\omega$ the sum of the μ-values of all arcs from v is 1: $\sum_{a\in T_2} \mu(a)\#(v,a) = 1$.

Remark 2.7. This definition obviously precludes vertices with infinite out-degree. In fact, it is not straightforward to introduce a meaningful definition enabling vertices having infinite out-degree. On the contrary, vertices with infinite in-degree are acceptable with this definition.

Proposition 2.8. *Given a* HR*-grammar $\mathcal{G}$ and a mapping $\mu : T_2 \to [0,1]$, one can decide whether $(\mathcal{G},\mu)$ is a* PHR*-grammar.*

Proof. From Theorem 2.5 we may assume that $\mathcal{G}$ is complete outside. It enables to identify vertices of infinite out-degree. Let v be such a vertex, and a a label such that $\#(v,a) = +\infty$, it forbids $(\mathcal{G},\mu)$ to be a PHR-grammar for any value of $\mu(a)$. If there is no such vertex, from Proposition 3.13 (b) of [5], there exists an effective colouring of $\mathcal{G} = (N,T,R,Z)$ with colours representing the degree of each vertex (relative to each label). We produce a colouring representing the exact *out*-degree relative to each element of T_2. There are only finitely many such degrees (from the same proposition, (a)). Now from these colours we are able to compute # at each vertex v in the grammar and therefore we may check that $\sum_{a\in T_2} \mu(a)\#(v,a) = 1$. □

Example 2.9. We consider the graph from Example 2.2. The probabilistic mapping μ, defined by $\mu(a) = \frac{1}{2}$ and $\mu(d) = \frac{1}{4}$, yields a probabilistic regular graph. Clearly the sum of out-going edges is 1 for each vertex of the graph.

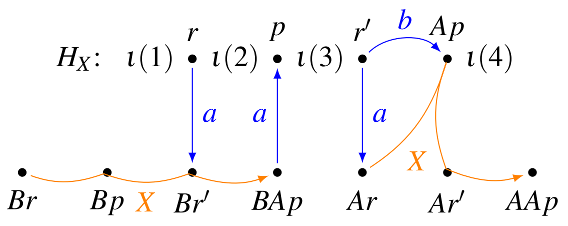
2.5 Connection between regular graphs and pushdown automata

There is a strong connection between regular graphs and configuration graphs of pushdown automata. Indeed restricted to finite in- and outdegrees, these graphs coincide: see, *e.g.*, [5, Theorem 5.11]. In particular, given a pushdown automaton, the transformation into a graph grammar which generates a infinite regular graph isomorphic to the configuration graph of the pushdown system is straightforward and may be adapted from the proof of Proposition 5.4 in [5]. This proposition states that the suffix graph of any rewriting system may be generated by a one rule grammar from the non-terminal. We illustrate this construction on the following example.

Example 2.10. Let us consider the following pushdown system

$$r \xrightarrow{a} Br' \quad r' \xrightarrow{a} Ar \quad r' \xrightarrow{b} Ap \quad BAp \xrightarrow{a} p.$$

To match more closely [5, Proposition 5.4] it is presented as a suffix rewriting system: the state of the pushdown automaton is on the top of the stack, and rules are applied to suffixes of the stack. For example, when in state r, and whatever the contents of the stack, while reading an a, stack-symbol B is pushed and the new state is r'. The transformation of this pushdown automaton into a graph grammar goes as follows. There is a unique non-terminal X (which, hence, serves as axiom). The vertices of H_X are words: each strict suffix (distinct from the empty suffix) of the words appearing in the rewriting rules (in the left- and right-hand sides) belongs to the image of ι_X. Here r, p, r' and Ap are the non-empty strict suffixes and they are represented on the top line of the graph H_X. For every stack symbol (here A and B), and every non-empty strict suffix, a vertex is formed by the concatenation of the stack symbol and the suffix. This yields new vertices, such as Br and all the ones on the bottom line of H_X, but some vertices might already be present, as Ap in this example. For each stack symbol, a non-terminal arc, labelled by X connects these vertices: Ar, Ap, Ar', AAp and Br, Bp, Br', BAp, respectively. This construction ensures that each left- and right-hand side of the rewriting rules is one vertex. It now suffices to add terminal arcs between the vertices according to the rules. For example the a-edge from r to Br' encodes the first rewriting rule.



Notice that this construction produces several connected components. Yet, given an initial configuration only the connected component (co-)reachable from this configuration will be relevant.

A similar transformation can be applied to any pushdown automaton in order to obtain a graph grammar which generates the configuration graph of the pushdown system. This underlines the generality of the model of graph grammars. Moreover, we argue the framework of graph grammars is more convenient than the pushdown automata view. Indeed, transformations presented in Subsection 2.3 on graph grammars do not affect the graph they generate, contrary to most transformations on pushdown automata that affect the structure of the configuration graph.

Esparza *et al.* propose in [8] a model of probabilistic pushdown automata, derived from pushdown automata by assigning weights to rules. The configuration graphs of such systems are infinite state Markov chains. Probabilistic pushdown automata and PHR-grammar relate in the same way than pushdown automata and graph grammars do: the Markov chains defined by both models are the same. Moreover, any probabilistic pushdown automaton can be turned into a PHR-grammar which generated exactly the same infinite state Markov chain. In this sense our model does not generalize the previous model. On the other hand, [8] makes several syntactical assumptions on pushdown automata which do not restrict the class of Markov chains, but make it more difficult to manipulate. Transformations of probabilistic pushdown automata in order to fit these assumptions may alter the properties of the Markov chain. On the contrary, transformations of PHR-grammars do not affect the Markov chain generated.

3 Verification of probabilistic regular graphs

3.1 Markov chains and PCTL

A (discrete-time) *Markov chain* is a tuple $\mathcal{M} = (S, s_0, p)$ consisting of a (possibly infinite) set S of states, an initial state s_0, and a probabilistic transition function $p : S \times S \to [0,1]$ such that for every state s, $\sum_{s' \in S} p(s,s') = 1$. For simplicity, we assume the transition system is finitely branching, *i.e.*, in any state s there are only finitely many states s' with $p(s,s') > 0$; the condition $\sum_{s' \in S} p(s,s') = 1$ is thus well-defined. Given a set of atomic propositions AP, a *labelled Markov chain* $\mathcal{M} = (S, s_0, p, \ell)$ is a Markov chain (S, s_0, p) equipped with a labelling function $\ell : S \to \mathsf{AP}$.

Introduced in [9], PCTL is an extention of CTL with probabilities. It can express quantitative properties about executions in Markov chains, *e.g.*, with probability 0.9 any sent message will be acknowledged in the future. The syntax of PCTL is the following:

$$\varphi ::= \mathtt{tt} \mid a \mid \neg\varphi \mid \varphi \wedge \psi \mid \mathbf{X}^{\sim\rho}\varphi \mid \varphi\,\mathbf{U}^{\sim\rho}\,\psi$$

where $a \in \mathsf{AP}$ is an atomic proposition, $\rho \in [0,1]$ and $\sim \in \{\leq, <, >, \geq\}$. Operators $\mathbf{X}^{\sim\rho}$ and $\mathbf{U}^{\sim\rho}$ are respectively the probabilistic next-state and until operators and generalise their nonprobabilistic counterparts. Recall the shortcuts in CTL for eventually ($\mathbf{F}$) and globally ($\mathbf{G}$): $\mathbf{F}\,\varphi \equiv \mathtt{tt}\,\mathbf{U}\,\varphi$ and $\mathbf{G}\,\varphi \equiv \neg\mathbf{F}\,\neg\varphi$. Their probabilistic extensions $\mathbf{F}^{\sim\rho}$ and $\mathbf{G}^{\sim\rho}$ will also be convenient in the sequel.

Let $\mathcal{M} = (S, s_0, p, \ell)$ be a labelled Markov chain, and $s \in S$. For a (non-probabilistic) formula ϕ of CTL, we write $\mathbb{P}(s \models \phi)$ for the measure of the set of paths in $\mathcal{M}$ issued from s and which satisfy ϕ. Note that for V_1 and V_2 sets of states, the set of paths from s satisfying $\mathbf{X}\,V_1$ or $V_1\,\mathbf{U}\,V_2$ is clearly measurable. The semantics of a PCTL formula φ over $\mathcal{M}$ is defined inductively:

$$\begin{array}{ll}
[\![\mathtt{tt}]\!] = S & [\![\varphi \wedge \psi]\!] = [\![\varphi]\!] \cap [\![\psi]\!] \\
[\![a]\!] = \{s \in S \mid a \in \ell(s)\} & [\![\mathbf{X}^{\sim\rho}\varphi]\!] = \{s \in S \mid \mathbb{P}(s \models \mathbf{X}\,[\![\varphi]\!]) \sim \rho\} \\
[\![\neg\varphi]\!] = S \setminus [\![\varphi]\!] & [\![\varphi\,\mathbf{U}^{\sim\rho}\,\psi]\!] = \{s \in S \mid \mathbb{P}(s \models [\![\varphi]\!]\,\mathbf{U}\,[\![\psi]\!]) \sim \rho\} \\
[\![\mathbf{F}^{\sim\rho}\varphi]\!] = \{s \in S \mid \mathbb{P}(s \models \mathbf{F}\,[\![\varphi]\!]) \sim \rho\} & [\![\mathbf{G}^{\sim\rho}\varphi]\!] = \{s \in S \mid \mathbb{P}(s \models \mathbf{G}\,[\![\varphi]\!]) \sim \rho\}
\end{array}$$

and we write $s \models \varphi$ for $s \in [\![\varphi]\!]$.

In the following, we will interpret PCTL formulae over labelled Markov chains induced by PHR-grammar. Atoms in these formulae will be sets of vertices and will form the set of atomic propositions AP.

Example 3.1. Considering the graph presented in Example 2.2, the probabilistic mapping given in Example 2.9, and predicates V_1 and V_2 satisfied by vertices labelled by these respective colours, the following formulae are of interest:

- $\varphi_1 = V_1 \wedge \mathbf{X}^{\geq\frac{1}{2}}V_2$: Vertices that satisfy φ_1 belong to V_1 and with probability greater than $\frac{1}{2}$, their successors in one step are in V_2. In particular, vertices at a fork on the lower line of Figure 2.2 satisfy φ_1.
- $\varphi_2 = v_0 \wedge V_1\,\mathbf{U}^{>\frac{2}{3}}V_2$: Vertex v_0 satisfies φ_2 if the probability of all paths issued from v_0 that eventually reach V_2 passing through vertices of V_1 only is greater than $\frac{2}{3}$.

3.2 Qualitative model checking for probabilistic regular graphs

The qualitative fragment of PCTL only involves the probability thresholds 0 and 1. Let $\mathscr{P} = (N, T, R, Z, \mu)$ be a PHR-grammar. Up to isomorphism $\mathscr{P}$ generates a unique infinite state Markov chain $\mathcal{M}_{\mathscr{P}}$ (or $\mathcal{M}$

when there is no ambiguity on $\mathscr{P}$). The qualitative model checking problem for probabilistic regular graphs is, given a PHR-grammar $\mathscr{P}$ with initial vertex v_0 and a qualitative PCTL formula φ, to answer whether in $\mathscr{M}_{\mathscr{P}}, v_0 \models \varphi$. Mimicking the finite Markov chain approach, the set of vertices satisfying a qualitative formula can be effectively computed.

Theorem 3.2. *Let φ be a qualitative PCTL formula, and $\mathscr{P}$ a* PHR*-grammar. There is an effective colouring $\mathscr{P}'$ in which the set $\{v \in V_G \mid v \models \varphi\}$ is identified by a new colour.*

Proof. The proof is by induction on the structure of φ, using the fact that the following sets of vertices can be effectively coloured in the graph grammar: $\{v \in V_G \mid \mathbb{P}(v, \mathbf{X}\, V) = 1\}$, $\{v \in V_G \mid \mathbb{P}(v, \mathbf{X}\, V) = 0\}$, $\{v \in V_G \mid \mathbb{P}(v, V_1 \,\mathbf{U}\, V_2) = 1\}$ and $\{v \in V_G \mid \mathbb{P}(v, V_1 \,\mathbf{U}\, V_2) = 0\}$.

Let us start with the two first cases: $\{v \in V_G \mid \mathbb{P}(v, \mathbf{X}\, V) = 1\}$ and $\{v \in V_G \mid \mathbb{P}(v, \mathbf{X}\, V) = 0\}$. The function Can induces a finite partition on vertices of the infinite Markov chain generated by $\mathscr{P}$. Two vertices with same image by Can have equivalent successors. By hypothesis on the grammar, for every vertex generated at level n, all successor vertices are generated between levels $n-1$ and $n+1$. Hence, if v is generated in H_A, it is sufficient to identify in $R^2[A]$ whether all successors of v belong to V or $\overline{V}$. One can thus, in the hypergraphs H_A (for each $A \in N$), annotate by colours the vertices which have all their successors in V, as well as those which have no successors in V. These colours precisely correpond to the sets $\{v \in V_G \mid \mathbb{P}(v, \mathbf{X}\, V) = 1\}$ and $\{v \in V_G \mid \mathbb{P}(v, \mathbf{X}\, V) = 0\}$.

The two other cases $\{v \in V_G \mid \mathbb{P}(v, V_1 \,\mathbf{U}\, V_2) = 1\}$ and $\{v \in V_G \mid \mathbb{P}(v, V_1 \,\mathbf{U}\, V_2) = 0\}$ are treated similarly. We detail here the colouring of $\{v \in V_G \mid \mathbb{P}(v, V_1 \,\mathbf{U}\, V_2) = 1\}$. For $B \in \mathsf{Succ}(A)$ and $i \leq \rho(B)$ we let $R((B,i)_A) = \{A_j | \mathbb{D}(B_i, A_j) > 0\}$. We then define inductively the sets:

- $W_0 = (H_Z \cap V_2) \cup \{v \,|\, \mathsf{Can}(v) = (B,i)_A$ and $\mathbb{W}(B_i)_A = 1\}$, and
- $W_{n+1} = W_n \cup \{v \,|\, \mathsf{Can}(v) = (B,i)_A$ and $\mathbb{W}(B_i)_A + \sum_{A_j \in R((B,i)_A)} \mathbb{D}(B_i, A_j) = 1$ and $R((B,i)_A) \subseteq W_n\}$.

Vertices in W_0 are directly winning, either because they already belong to V_2 or because from B_i in context A, the probability to win without decreasing level is 1. Vertices in W_{n+1} are also almost surely winning (*i.e.* satisfy $V_1 \,\mathbf{U}\, V_2$ with probability 1) because they are winning without decreasing level (factor $\mathbb{W}(B_i)_A$) or firstly decreasing level and then win from A_j with probability 1 (since $A_j \in W_n$).

Clearly, $\bigcup_{n=0}^{\infty} W_n = \{v \in V_G \mid \mathbb{P}(v, V_1 \,\mathbf{U}\, V_2) = 1\}$ and the W_n's can be iteratively computed and annotated in the grammar by colours. $\square$

3.3 Probability computation for probabilistic regular graphs

We now face the problem of computing, given v_0 an initial vertex in H_Z and ϕ a CTL formula, the probability in $\mathscr{M}_{\mathscr{P}}$ of the set of paths starting in v_0 and satisfying ϕ: $\mathbb{P}_{\mathscr{M}_{\mathscr{P}}}(v_0 \models \phi)$. This can be done inductively on the structure of ϕ, and the difficult part amounts to computing, given V_1 and V_2 colours, the probability starting in v_0 to satisfy $V_1 \,\mathbf{U}\, V_2$, written $\mathbb{P}(v_0 \models V_1 \,\mathbf{U}\, V_2)$. This subsection focuses on solving this problem.

3.3.1 Preliminaries and notations

Without loss of generality we assume that vertices of V_1 and V_2 are annotated in the grammar by colours (terminals of arity 1) and that v_0 appears in H_Z the hypergraph of the rewriting rule associated to the axiom Z of $\mathscr{P}$. Using the levelwise decomposition of the Markov chain $\mathscr{M}_{\mathscr{P}}$, we show how to express $\mathbb{P}(v_0 \models V_1 \,\mathbf{U}\, V_2)$ as a solution of a system of polynomial equations derived from the axiom and the rules.

The hypotheses we demand on PHR-grammars ensure that the first step of any path issued from a vertex of level n either remains at level n or reaches one of the neighbour levels, $n-1$ and $n+1$ (from

Theorem 2.5, it corresponds to restricting to finite degree). This fact will enable levelwise decomposition of paths in the Markov chain.

To compute probabilities in Markov chains generated by PHR-grammars we exploit the regularities of the underlying graphs. For v a vertex of $\mathscr{M}_{\mathscr{P}}$ with $\mathsf{Can}(v) \in H_A$, we write $\mathscr{M}[v]$ for the part of $\mathscr{M}_{\mathscr{P}}$ with underlying graph $R^{\omega}[A]$ which contains v and no vertices of level $\mathsf{Lev}(v)-1$. Intuitively, if v has been generated by a non-terminal A, we consider the infinite (sub-)Markov chain generated from this non-terminal. For two vertices v and v' of $\mathscr{M}_{\mathscr{P}}$ with $\mathsf{Can}(v) = \mathsf{Can}(v') \in H_A$, the isomorphism of $\mathscr{M}[v]$ and $\mathscr{M}[v']$ ensures that for any CTL formula ϕ, $\mathbb{P}_{\mathscr{M}[v]}(v \models \phi) = \mathbb{P}_{\mathscr{M}[v']}(v' \models \phi)$. In particular, if ϕ is the formula $V_1 \,\mathbf{U}\, V_2$, we obtain that: the probability to succeed satisfying $V_1 \,\mathbf{U}\, V_2$ without decreasing level is the same from v and from v'. The probability to satisfy $(V_1 \setminus V_2)$ while decreasing level of 1 is also independent of the level, provided the initial state corresponds to a fixed canonical representant $(B,i)_A$. This motivates the introduction of notations for such probabilities, that are determined by the context and are independent of the level.

Let $A,B \in N$ be non-terminals such that $B \in \mathsf{Succ}(A)$. Starting in state v, with $\mathsf{Can}(v) = (B,i)_A$, each successor state belongs to $R^2[A]$, the sub-graph obtained from non-terminal A by two successive complete parallel rewritings. Given $i \leq \rho(B)$ and $j \leq \rho(A)$ we introduce:

- $\mathbb{D}(B_i,A_j)$ as the probability from v', with $\mathsf{Can}(v') = (B,i)_A$, to reach v such that $\mathsf{Lev}(v) = \mathsf{Lev}(v') - 1$ and $v = \iota_A(j)$ satisfying along the path: $(V_1 \setminus V_2) \cap (\mathsf{Lev} \geq \mathsf{Lev}(v'))$;
- $\mathbb{W}(B_i)_A$ as the probability from v', with $\mathsf{Can}(v') = (B,i)_A$, to fulfill $(V_1 \cap \mathsf{Lev} \geq \mathsf{Lev}(v'))\,\mathbf{U}\, V_2$.

(Here $\mathsf{Lev} \geq k$ denotes that the current level is greater than a given natural k.)

As explained before, $\mathbb{D}(B_i,A_j)$ and $\mathbb{W}(B_i)_A$ do not depend on v' and v but only on their images by Can. Moreover, $\mathbb{D}(B_i,A_j)$ expresses the probability to decrease level by one while satisfying a given property and $\mathbb{W}(B_i)_A$ is the probability to win, *i.e.*, to fulfill $V_1 \,\mathbf{U}\, V_2$ without decreasing level. This justifies the chosen notations.

The levelwise decomposition of paths is given by vertices belonging (when generated) to non-terminal. Thus, given $A,B,C,D \in N$ such that $B,D \in \mathsf{Succ}(A)$ and $C \in \mathsf{Succ}(B)$, we introduce notations for some probabilities that can be computed directly in any portion $R^2[A]$ of the Markov chain.

- $p(B_i)_A$ is the probability in $R^2[A]$ from v with $\mathsf{Can}(v) = (B,i)_A$ to fulfill $V_1 \,\mathbf{U}\, V_2$ without visiting any $v' = \iota_A(j)$ nor v' with $\mathsf{Can}(v') \in \{(C,k)_B, (B,h)_A\}$.
- $p(B_i,D_h)_A$ is the probability in $R^2[A]$ from v with $\mathsf{Can}(v) = (B,i)_A$ to fulfill $\mathbf{G}\,(V_1 \setminus V_2)$ and reach v' with $\mathsf{Can}(v') = (D,j)_A$ before any v'' such that $\mathsf{Can}(v'') \in \{(B,l)_A, (D,h')_A\}$ and $\mathsf{Lev}(v'') = \mathsf{Lev}(v)$.
- $\overleftarrow{p}(B_i,A_j)$ is the probability in $R^2[A]$ from vertex v with $\mathsf{Can}(v) = (B,i)_A$ to reach v' with $v' = \iota_A(j)$ and $\mathsf{Lev}(v') = \mathsf{Lev}(v) - 1$ and satisfy $\mathbf{G}\,(V_1 \setminus V_2)$ without seeing any $v'' \in \{(C,k)_B, (B,l)_A, (D,h)_A\}$.
- $\overrightarrow{p}(B_i,C_k)_A$ is the probability in $R^2[A]$ from v with $\mathsf{Can}(v) = (B,i)_A$ to reach v' with $\mathsf{Can}(v') = (C,k)_B$ satisfying $\mathbf{G}\,(V_1 \setminus V_2)$ without visiting any $v'' = \iota_A(j)$ nor $v'' \in \{(C,k')_B, (B,h)_A\}$.

Intuitively, there are several alternatives for paths starting in v (with $\mathsf{Can}(v) = (B,i)_A$) and for which $V_1 \,\mathbf{U}\, V_2$ is not falsified: either they satisfy $V_1 \,\mathbf{U}\, V_2$ without visiting any vertex at some position on a non-terminal hyperarc, or they satisfy $\mathbf{G}\, V_1 \setminus V_2$ and reach some vertex v' at a given position on a non-terminal hyperarc. The above probabilities split these cases according the first v' encountered: v' can be at the level of v (at the h-th position in hyperarc D), or at levels $n-1$ (thus of the form $\iota_A(j)$) or $n+1$ (at the k-th position in hyperarc C). As argued before, $p(B_i)_A$, $p(B_i,D_j)_A$, $\overleftarrow{p}(B_i,A_j)$, and $\overrightarrow{p}(B_i,C_k)_A$ can be computed directly in $R^2[A]$, obtained from H_A, H_B, and H_E for all $E \in \mathit{Succ}(A) \cup \mathit{Succ}(B)$.

Example 3.3. We compute these probabilities on Example 2.2: $p(A_2)_A = a$, $p(A_1,A_2)_A = a$, $\overleftarrow{p}(A_2,A_1) = d$ and $\overrightarrow{p}(A_1,A_2)_A = 0$.

3.3.2 Computation of $\mathbb{P}(v_0 \models V_1 \,\mathbf{U}\, V_2)$

Theorem 3.4. *The $\mathbb{D}(B_i,A_j)$'s and $\mathbb{W}(B_i)_A$'s satisfy the following equations:*

$$\mathbb{D}(B_i,A_j) = \overleftarrow{p}(B_i,A_j) + \sum_{D_h} p(B_i,D_h)_A \cdot \mathbb{D}(D_h,A_j) + \sum_{C_k} \overrightarrow{p}(B_i,C_k)_A \cdot \sum_{B_\ell} \mathbb{D}(C_k,B_\ell) \cdot \mathbb{D}(B_\ell,A_j) \quad (1)$$

$$\mathbb{W}(B_i)_A = p(B_i)_A + \sum_{D_h} p(B_i,D_h)_A \cdot \mathbb{W}(D_h)_A + \sum_{C_k} \overrightarrow{p}(B_i,C_k)_A \Big(\mathbb{W}(C_k)_B + \sum_{B_j} \mathbb{D}(C_k,B_j) \cdot \mathbb{W}(B_j)_A \Big). \quad (2)$$

Moreover, if we add the following constraints:

- *if $B_i \notin (V_1 \setminus V_2)$ then $\mathbb{D}(B_i,A_j) = 0$ for every A_j, and*
- *if $B_i \in V_2$ then $\mathbb{W}(B_i)_A = 1$, and if $B_i \notin (V_1 \cup V_2)$ then $\mathbb{W}(B_i)_A = 0$;*

the $\mathbb{D}(B_i,A_j)$'s and $\mathbb{W}(B_i)_A$'s form the least solution of this system of polynomial equations.

Proof. The correctness of Equations 1 and 2 is proved by partitioning the set of paths issued from vertex v with $\mathsf{Can}(v) = (B,i)_A$.

Precisely, concerning Equation 1, any path from v with $\mathsf{Can}(v) = (B,i)_A$ to $v' = \iota_A(j)$ (and $\mathsf{Lev}(v') = \mathsf{Lev}(v) - 1$) satisfying $\mathbf{G}\, V_1 \setminus V_2$ falls in exactly one of the following cases:

- either it goes directly from v to v' without leaving v's level;
- or it reaches vertex v'' with $\mathsf{Can}(v'') = (D,h)_A$ and $\mathsf{Lev}(v'') = \mathsf{Lev}(v)$, and then goes from v'' to v';
- or it reaches some vertex v'' with $\mathsf{Can}(v'') = (C,k)_B$ and $\mathsf{Lev}(v'') = \mathsf{Lev}(v)+1$, and then returns to v's level at vertex $v^{(3)}$ with $\mathsf{Can}(v^{(3)}) = (B,\ell)_A$ and from there finally reaches v'.

This case distinction is illustrated on Figure 3.1 where plain arrows represent paths in $R^2[A]$ (as presented earlier) and dotted arrows represent recursive probabilities to decrease level.

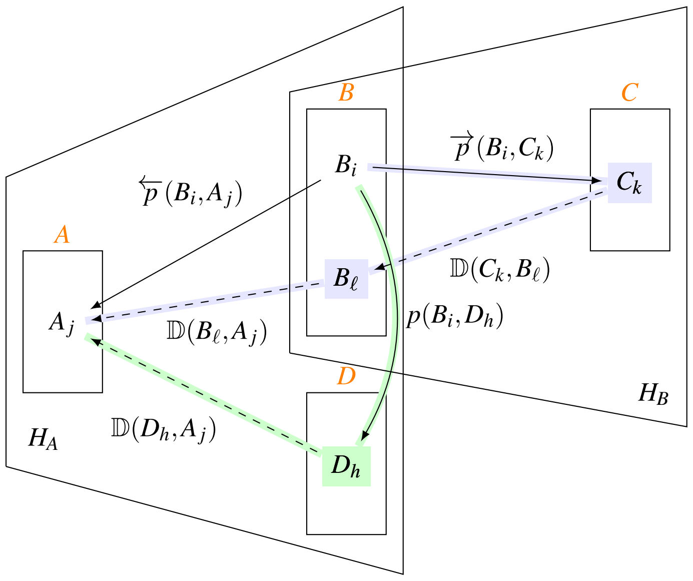


Figure 3.1: Illustration of Equation (1) for $\mathbb{D}(B_i,A_j)$.

For Equation 2, the reasoning is similar. Any path issued from v satisfying $V_1 \,\mathbf{U}\, V_2$ without visiting vertices of level smaller than $\mathsf{Lev}(v)$:

- either satisfies $V_1 \,\mathbf{U}\, V_2$ without visiting any other non-terminals (and hence at v's level)
- or reaches a vertex v' with $\mathsf{Can}(v') = (D,h)_A$ and $\mathsf{Lev}(v') = \mathsf{Lev}(v)$ and from then on satisfies $V_1 \,\mathbf{U}\, V_2$ without decreasing level
- or goes to vertex v' with $\mathsf{Can}(v') = (C,k)_B$ and $\mathsf{Lev}(v') = \mathsf{Lev}(v)+1$, and from there either satisfies $V_1 \,\mathbf{U}\, V_2$ without going back to verticesat v's level, or reaches some v'' with $\mathsf{Can}(v'') = (B,\ell)_A$ and $\mathsf{Lev}(v'') = \mathsf{Lev}(v)$ and from v'' satisfy $V_1 \,\mathbf{U}\, V_2$ without decreasing level.

These partitions of the set of paths issued from vertex v with $\mathsf{Can}(v) = (B,i)_A$ justify Equations 1 and 2.

The system of equations defines an operator $\mathscr{F} : [0,1]^n \to [0,1]^n$ where n is the number of variables appearing in the system. The valuation $\mathscr{F}(\nu)$ of the variables is obtained by evaluating each equation the right-hand side where each variable is substituted with its value in ν. This operator is monotonic and continuous, and hence admits a unique least fixed-point, which is eventually reached by iterating $\mathscr{F}$ on the null-valuation which assigns 0 to all variables. Note that the convergence towards the least fixed-point might require infinitely many iterations.

To prove that the $\mathbb{D}(B_i,A_j)$'s and $\mathbb{W}(B_i)$'s form the *least* solution of the system, we consider the probabilities approximated by truncating the paths at length k. Precisely, let $\mathbb{D}(B_i,A_j)^k$ be the probability-mass of $\mathbb{D}(B_i,A_j)$ restricted to paths of length at most k, ; similarly let $\mathbb{W}(B_i)^k_A$ be the probability-mass of paths of length at most k in $\mathbb{W}(B_i)_A$. As k tends to infinity, those probabilities tend to $\mathbb{D}(B_i,A_j)$ and $\mathbb{W}(B_i)_A$, respectively. It is thus sufficient to prove that, for any $k \in \mathbb{N}$, $\mathbb{D}(B_i,A_j)^k$ and $\mathbb{W}(B_j)_A$ are no greater than the least solution of the system. This is easily done by induction on k. □

Recall that our goal is to compute $\mathbb{P}(v_0 \models V_1 \,\mathbf{U}\, V_2)$. This probability can be expressed using the $\mathbb{D}(B_i,A_j)$'s and $\mathbb{W}(B_i)_A$'s:

$$\mathbb{P}(v_0 \models V_1 \,\mathbf{U}\, V_2) = p(v_0)_Z + \sum_{A_i \in \mathsf{Succ}(Z)} \overrightarrow{p}(v_0,A_i)_Z \,\mathbb{W}(A_i)_Z, \tag{3}$$

where

- $p(v_0)_Z$ is the probability in H_Z from v_0 to fulfill $V_1 \,\mathbf{U}\, V_2$ without visiting any vertex v' with $\mathsf{Can}(v') = (A,i)_Z$ for some $A \in Succ(Z)$;
- $\overrightarrow{p}(v_0,A_i)_Z$ is the probability in H_Z from v_0 to v' with $\mathsf{Can}(v') = (A,i)_Z$ while satisfying $\mathbf{G}\,(V_1 \setminus V_2)$ and without visiting any vertex v'' such that $\mathsf{Can}(v'') = (B,j)_Z$ (for some $B \in \mathsf{Succ}(Z)$) in between.

Example 3.5. We illustrate the computation of $\mathbb{P}(v_0 \models V_1 \,\mathbf{U}\, V_2)$ on our running example. Since $\mathsf{Can}(v_0) = (A,1)_Z$ and $v_0 \notin V_2$, $p(v_0)_Z = 0$ and $\overrightarrow{p}(v_0,A_1) = 1$. From Equation 3 we deduce $\mathbb{P}(v_0 \models V_1 \,\mathbf{U}\, V_2) = \mathbb{W}(A_1)_Z$. Let us detail some steps of the computation.

$$\begin{aligned}\mathbb{W}(A_1)_Z &= a\mathbb{W}(A_2)_Z + a\big(\mathbb{W}(A_1)_A + \mathbb{D}(A_1,A_1)\mathbb{W}(A_1)_Z + \mathbb{D}(A_1,A_2)\mathbb{W}(A_2)_Z\big)\\ &= a\mathbb{W}(A_1)_A + a\mathbb{D}(A_1,A_1)\mathbb{W}(A_1)_Z,\end{aligned}$$

since $\mathbb{W}(A_2)_Z = 0$. The probability $\mathbb{W}(A_2)_A$ is easily computed: $\mathbb{W}(A_2)_A = a$. Then $\mathbb{D}(A_1,A_1)$ is the least solution of a quadratic equation:

$$a\mathbb{D}(A_1,A_1)^2 - \mathbb{D}(A_1,A_1) + ad = 0.$$

Letting that $a = \frac{1}{2}$ and $d = \frac{1}{4}$, we get $\mathbb{D}(A_1,A_1) = 1 - \frac{\sqrt{3}}{2}$. Finally

$$\mathbb{W}(A_1)_Z = \frac{a\mathbb{W}(A_1)_A}{1 - a\mathbb{D}(A_1,A_1)} = \frac{a^3}{(1 - a - a\mathbb{D}(A_1,A_1))(1 - a\mathbb{D}(A_1,A_1))}\mathbb{W}(A_1)_Z = \frac{2}{3}(2\sqrt{3} - 3) \approx 0.31.$$

Note that the exact computation of the solutions of the system may not always be performed. Indeed, in general, the equations are polynomials (of arbitrary degree) in the variables. However, similarly as in [8], approximate values for the solutions can be computed.

Theorem 3.6. *Let $\mathscr{P} = (N,T,R,Z,\mu)$ be a* PHR*-grammar, and v_0 a vertex in H_Z. For $\rho \in \mathbb{Q} \cap [0,1]$ and $\sim \in \{\leq,<,\geq,>\}$, it is decidable whether $\mathbb{P}(v_0 \models V_1 \mathbf{U} V_2) \sim \rho$. Moreover, given $0 < \lambda < 1$, one can compute $\rho_1, \rho_2 \in \mathbb{Q}$ such that $\rho_1 \leq \mathbb{P}(v_0 \models V_1 \mathbf{U} V_2) \leq \rho_2$, and $\rho_2 - \rho_1 \leq \lambda$.*

Proof. Deciding $\mathbb{P}(v_0 \models V_1 \mathbf{U} V_2) \sim \rho$ is equivalent to deciding $p(v_0)_Z + \sum_{A_i \in \mathsf{Succ}(Z)} \overrightarrow{p}(v_0, A_i)_Z \mathbb{W}(A_i)_Z \sim \rho$. Using Equations 1 and 2, the decidability of the first order arithmetics of reals [12] yields the decidability of our problem. An iterative application of the decision algorithm allows to compute in a dichotomic way the desired approximations ρ_1 and ρ_2. □

3.4 Undecidability of quantitative model checking

In this subsection, we give a proof of the undecidability of the exact quantitative PCTL model-checking problem for PHR-grammars. Since PHR-grammars generalise probabilistic pushdown automata, this result is a consequence of the undecidability of quantitative PCTL model-checking for probabilistic pushdown automata [2]. We however adapt the proof presented in [2] to graph grammars, which, in our opinion, enable a simpler exposition.

The undecidability is proved by a reduction of Post Correspondance Problem (PCP). Recall that an instance of the PCP is a sequence of pairs of words $((u_i, v_i))_{i \leq n}$ over a fixed alphabet Σ, and the problem is to determine whether there is an integer k, and a sequence $(i_\ell)_{\ell \leq k}$ such that $u_{i_1} u_{i_2} \ldots u_{i_k} = v_{i_1} v_{i_2} \ldots v_{i_k}$.

The quantitative model-checking problem of PCTL for PHR-grammars is the following:

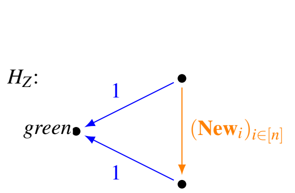
Instance: A PHR-grammar $\mathscr{P}$, and a PCTL formula φ.
Question: Is φ valid on $\mathscr{M}_{\mathscr{P}}$?

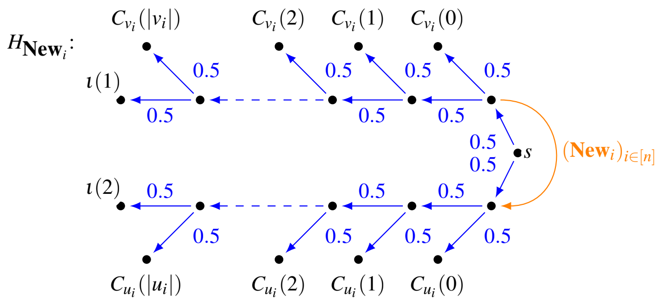
Theorem 3.7 ([2])**.** *The quantitative model-checking problem of PCTL for* PHR*-grammars is undecidable.*

Proof. This result is a consequence of [2] but we give here a direct proof. Let $((u_i, v_i))_{i \leq n}$ be a sequence of pairs of words on $\Sigma = \{0,1\}$. From this instance of PCP, we define the following PHR-grammar: $\mathscr{P} = (N,T,R,Z,\mu)$, where:

- $N = \{Z\}_0 \cup \{\mathbf{New}_i \mid i \leq n\}_2$;
- $T = \{s, green, red\}_1 \cup \{a,b\}_2$;
- $\mu(a) = 0.5, \mu(b) = 1$;

and the set $R = (H_B, \iota_B)_{B \in N}$ of rewriting rules is depicted below:





Colours *green*, and *red* label vertices as follows. For each $i \leq n$, $k \leq |u_i|$, and $k' \leq |v_i|$,

$$C_{u_i}(k) = \begin{cases} green & \text{if } u_i(k) = 1 \\ red & \text{if } u_i(k) = 0 \end{cases}, \quad C_{v_i}(k') = \begin{cases} green & \text{if } v_i(k') = 0 \\ red & \text{if } v_i(k') = 1 \end{cases}$$

Consider the following PCTL formula:

$$\varphi_0 = S \wedge (\mathtt{tt}\,\mathbf{U}^{=\frac{1}{2}} Green)$$

where *Green* and S are atomic propositions corresponding to vertices labelled respectively by *green* and *s*, terminals of arity 1. We claim that φ_0 is valid on $\mathscr{M}_{\mathscr{P}}$ if and only if there is a solution to the Post instance $((u_i, v_i))_{i \leq n}$.

In the infinite graph generated by $\mathscr{P}$, each vertex labelled s is connected to the origin (labelled *green* in H_Z) via a sequence of u_i's on the lower branch, and of v_i's on the upper branch (with the same indices). Let $I = (I_0, I_1, \cdots, I_m)$ be a sequence of indices in $\{1, \cdots, n\}$, and consider v_I the s-vertex corresponding to this sequence. The probability to reach *red* from v_I is the following: $\mathbb{P}(v_I \models \mathtt{tt}\,\mathbf{U}\,Red) = \frac{1}{2}(u_{\mathsf{path}} + v_{\mathsf{path}})$ with

$$u_{\mathsf{path}} = \sum_{j \leq m} \sum_{k \leq |u_j|} \frac{1}{2^{(\Sigma_{\ell<j}|u_{I_\ell}|)+k}} (u_j(k) = 0) \quad \text{and} \quad v_{\mathsf{path}} = \sum_{j \leq m} \sum_{k' \leq |v_j|} \frac{1}{2^{(\Sigma_{\ell<j}|v_{I_\ell}|)+k}} (v_j(k') = 1).$$

The only situation where $\mathbb{P}(v_I \models \mathtt{tt}\,\mathbf{U}\,Red) = \frac{1}{2}$ (and hence $\mathbb{P}(v_I \models \mathtt{tt}\,\mathbf{U}\,Green) = \frac{1}{2}$) occurs when the same sequence of letters appear in u_{path} and v_{path} (from the unicity of the binary expansion). $\square$

4 Conclusion

In this paper we introduced probabilistic regular graphs, as graphs generated by graph grammars where terminal arcs are labelled with probabilities. Results concerning the model-checking of probabilistic pushdown automata extend to this context. Precisely, both the approximate PCTL and qualitative PCTL model checking problems are decidable, whereas the exact quantitative model-checking problem is undecidable.

We believe that our model of PHR-grammars offers a major benefit compared to pushdown systems: it focuses on structural aspects whereas configurations graphs of pushdown automata emphasise combinatorial aspects. Furthermore in order to identify classes of infinite state systems with a decidable quantitative PCTL model checking we believe that *structural* restrictions on the grammar might prove worth studying. A natural extension of our work is to extend the positive results to graphs where infinite in-degree in allowed. Another research direction is to try to climb up the Caucal hierarchy, like [4], and pursue our work on higher-order pushdown systems.

References

[1] T. Brázdil, V. Brozek, J. Holecek & A. Kučera (2008): *Discounted Properties of Probabilistic Pushdown Automata.* In: *Proceedings of the 15th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'08)*, *Lecture Notes in Computer Science* 5330, Springer, pp. 230–242.

[2] T. Brázdil, A. Kučera & O. Strazovský (2005): *On the Decidability of Temporal Properties of Probabilistic Pushdown Automata.* In: *Proceedings of the 22nd Annual Symposium on Theoretical Aspects of Computer Science (STACS'05)*, *Lecture Notes in Computer Science* 3404, Springer, pp. 145–157.

[3] J. R. Büchi (1964): *Regular Canonical Systems*. *Archiv für Mathematische Logik und Grundlagenforshung* 6, pp. 91–111.

[4] A. Carayol & S. Woerhle (2003): *The Caucal Hierarchy of Infinite Graphs in Terms of Logic and Higher-Order Pushdown Automata*. In: *Proceedings of the 23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'03)*, *Lecture Notes in Computer Science* 2914, Springer, pp. 112–123.

[5] D. Caucal (2007): *Deterministic graph grammars*, *Texts in logics and games* 2, pp. 169–250. Amsterdam University Press.

[6] D. Caucal & T. Knapik (2001): *An internal presentation of regular graphs by prefix-recognizable ones*. *Theory of Computing Systems* 34(4).

[7] B. Courcelle (1990): *Graph rewriting: an algebraic and logic approach*, *Handbook of Theoretical Computer Science* B: Formal Models and Semantics, pp. 193–242. Elsevier.

[8] J. Esparza, A. Kučera & R. Mayr (2006): *Model Checking Probabilistic Pushdown Automata*. *Logical Methods in Computer Science* 2(1).

[9] H. Hansson & B. Jonsson (1994): *A logic for reasoning about time and reliability*. *Formal Aspects of Computing* 6(5), pp. 512–535.

[10] A. Kučera (2006): *Methods for Quantitative Analysis of Probabilistic Pushdown Automata*. *Electronic Notes in Theoretical Computer Science* 149(1), pp. 3–15.

[11] D. Muller & P. Schupp (1985): *The theory of ends, pushdown automata, and second-order logic*. *Theoretical Computer Science* 37, pp. 51–75.

[12] A. Tarski (1951): *A Decision Method for Elementary Algebra and Geometry*. University of California Press, Berkeley.