Using Modal Logics to Express and Check Global Graph Properties

Mario R.F. Benevides L. Menasché Schechter

{mario,luis}@cos.ufrj.br

Abstract

Graphs are among the most frequently used structures in Computer Science. Some of the properties that must be checked in many applications are connectivity, acyclicity and the Eulerian and Hamiltonian properties. In this work, we analyze how we can express these four properties with modal logics. This involves two issues: whether each of the modal languages under consideration has enough expressive power to describe these properties and how complex (computationally) it is to use these logics to actually test whether a given graph has some desired property. First, we show that these properties are not definable in a basic modal logic or in any bisimulation-invariant extension of it, like the modal μ-calculus. We then show that it is possible to express some of the above properties in a basic hybrid logic. Unfortunately, the Hamiltonian and Eulerian properties still cannot be efficiently checked. In a second attempt, we propose an extension of CTL* with nominals and show that the Hamiltonian property can be more efficiently checked in this logic than in the previous one. In a third attempt, we extend the basic hybrid logic with the $\downarrow$ operator and show that we can check the Hamiltonian property with optimal (NP) complexity in this logic. Finally, we tackle the Eulerian property in two different ways. First, we develop a generic method to express edge-related properties in hybrid logics and use it to express the Eulerian property. Second, we express a necessary and sufficient condition for the Eulerian property to hold using a graded modal logic.

1 Introduction

Graphs are among the most frequently used structures in Computer Science [9]. Usually, in this discipline, many important concepts admit a graph representation, and sometimes a graph lies at the very kernel of the model of computation used. This happens, for instance, in the field of distributed systems [6, 20], where the underlying model of computation is built on top of a graph. In addition to this central role, graphs are also important in distributed systems as tools for the description of resource sharing problems, scheduling problems, deadlock issues, and so on. The case of distributed systems is particularly appealing because it illustrates well two different levels at which graph properties have to be described. One is the local level, encompassing properties that hold for vertices or constant-size vertex-neighborhoods. The other level is global and comprises properties that hold for the graph as a whole, as acyclicity and connectivity.

Graph theory provides a lot of tools to describe such problems and presents many efficient algorithmic methods to solve them. However, there is an important distinction between the two sides of this matter. In the "description" side, graphs provide a great level of generality, allowing for the description of very different problems in the same simple framework. But in the "solution" side, each graph problem has to be solved and each graph property has to be tested with a specific method that usually does not generalize to other different problems or properties.

A logical framework, on the other hand, may provide this level of generalization. In an intuitive and non-technical language, this can be stated as follows. Consider a logic $\mathbb{L}$ with its formulas and structures where these formulas are semantically evaluated. We need to be able to answer the following questions:

1. Can we encode a graph as a structure for $\mathbb{L}$?
2. Can we encode the graph properties that we want to verify as $\mathbb{L}$-formulas?
3. Does $\mathbb{L}$ has decidable inference methods to check whether a formula is satisfied (or valid) in a structure?

If the answers to all of these questions are positive, then we can use the inference methods of the logic to verify every graph property that we want, provided that we can express it as an $\mathbb{L}$-formula. Of course, there is still a fourth question that has to be answered:

4. Is the logical method as efficient (in terms of computational complexity) in testing a given property as the graph theoretical method?

In order to satisfy the first question, we choose to work with the family of modal logics. A very strong reason to choose modal logics for this task, instead of any other logic, is that modal logic formulas are evaluated in structures that are essentially graphs, which makes it a very natural choice for our work. As the first slogan in the preface of [8] states, "modal languages are simple yet expressive languages for talking about relational structures".

In this work, we analyze how we can express and efficiently check these global graph properties with modal logics. This involves two issues: the first is whether each of the modal languages that we consider has enough expressive power to describe the graph properties that we want; the second is how complex (computationally) it is to use these logics to actually test whether a given graph has some desired property.

A similar attempt to do this was presented in [7]. That work also tries to use modal logics to express graph properties, but it approaches this issue from a different point of view. The key differences between the two works will be discussed in the last section of this paper.

A *finite directed graph* (from now on called simply a *graph*) G is a pair (W, R), where W is a finite set of vertices and $R \subseteq W \times W$ is a set of ordered pairs of vertices (a binary relation on W), called edges. If $\langle v_i, v_j \rangle \in R$, we say that v_i is *adjacent to* v_j and v_j is *adjacent from* v_i. The *out-degree* of a vertex is the number of vertices adjacent from it and the *in-degree* the number of vertices adjacent to it. The set R of edges can also be written as a relation between two vertices v_i and v_j. We write $v_i R v_j$ to express the fact that v_i is adjacent to v_j.

A *path* in a graph G is a sequence of vertices $\langle v_1, v_2, \ldots, v_n\rangle$, $n \geq 2$, where $\langle v_i, v_{i+1}\rangle \in R$, for $0 < i < n$. We say that n is the length of the path. A *closed path* is a path such that $v_1 = v_n$. A *cycle* is a path where $v_1 = v_n$ and $v_i \neq v_j$, for $1 \leq i, j < n$, $i \neq j$. A graph G is said to be *acyclic* if there is no cycle in it, otherwise it is *cyclic*.

Every directed graph has an *underlying undirected graph*, in which we do not consider the particular orientation of the edges. This means that, if $G = (W, R)$ and $\langle v, w\rangle \in R$, then v is adjacent to w and w is adjacent to v in the underlying undirected graph of G.

The rest of this paper is organized as follows. In section 2, we present a simple modal logic suited for the description of graph properties. In section 3, we investigate the issue of whether some well-known graph properties are definable or not in the language presented in the previous section: connectivity, acyclicity and the Eulerian and Hamiltonian properties. In section 4, we extend the modal logic of the previous sections with nominals, obtaining a hybrid modal logic, and use it to express some of the above properties. In section 5, we introduce the branching-time temporal logic CTL* with nominals, which is a very expressive logic, and use it to check the Hamiltonian property in a better way than it was done in the basic hybrid logic. In section 6, we extend the basic hybrid logic of section 4 with the $\downarrow$ operator and show that we can check the Hamiltonian property with optimal (NP) complexity in this logic. In section 7, we tackle the Eulerian property in two different ways. First, we develop a generic method to express edge-related properties in hybrid logics and use it to express the Eulerian property. Second, we express a necessary and sufficient condition for the Eulerian property to hold using a graded modal logic. Finally, in section 8 we draw our concluding remarks.

2 Basic Graph Logic

In this section, we define a modal logic with two modal operators: $\Diamond$ and $\Diamond^+$. We call it *basic graph logic*.

Definition 1. *The language of the* basic graph logic *is a modal language consisting of a set Φ of countably many proposition symbols (the elements of Φ are denoted by $p_1, p_2, \ldots$), the boolean connectives $\neg$ and $\wedge$ and two modal operators: $\Diamond$ and $\Diamond^+$. The formulas are defined as follows:*

$$\varphi ::= p \mid \top \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \Diamond\varphi \mid \Diamond^+\varphi$$

We freely use the standard boolean abbreviations $\vee$, $\rightarrow$, $\leftrightarrow$ and $\bot$ and also the following abbreviations for the duals: $\Box\varphi := \neg\Diamond\neg\varphi$ and $\Box^+\varphi = \neg\Diamond^+\neg\varphi$. Also, in order to make the language more elegant, we introduce some abbreviations for the reflexive and transitive closures: $\Diamond^*\varphi = \varphi \vee \Diamond^+\varphi$ and $\Box^*\varphi = \neg\Diamond^*\neg\varphi$.

We now define the structures in which we evaluate formulas in modal logics: *frames* and *models*.

Definition 2. *A* frame *for the basic graph logic is a pair $\mathcal{F} = (W, R)$, where W is a non-empty set (finite or not) of vertices and R is a binary relation over W, i.e., $R \subseteq W \times W$.*

As we see, a frame for the basic graph logic is essentially a graph. This confirms our statement in the first section that modal logics are a very natural choice for this work.

Definition 3. *A* model *for the basic graph logic is a pair $\mathcal{M} = (\mathcal{F}, \mathbf{V})$, where $\mathcal{F}$ is a frame and $\mathbf{V}$ is a valuation function mapping proposition symbols into subsets of W, i.e., $\mathbf{V} : \Phi \mapsto \mathcal{P}(W)$.*

The semantical notion of satisfaction is defined as follows:

Definition 4. *Let $\mathcal{M} = (\mathcal{F}, \mathbf{V})$ be a model. The notion of* satisfaction *of a formula φ in a model $\mathcal{M}$ at a vertex v, notation $\mathcal{M}, v \Vdash \varphi$, can be inductively defined as follows:*

1. *$\mathcal{M}, v \Vdash p$ iff $v \in \mathbf{V}(p)$;*
2. *$\mathcal{M}, v \Vdash \top$ always;*
3. *$\mathcal{M}, v \Vdash \neg\varphi$ iff $\mathcal{M}, v \nVdash \varphi$;*
4. *$\mathcal{M}, v \Vdash \varphi_1 \wedge \varphi_2$ iff $\mathcal{M}, v \Vdash \varphi_1$ and $\mathcal{M}, v \Vdash \varphi_2$;*
5. *$\mathcal{M}, v \Vdash \Diamond\varphi$ iff there is a $w \in W$ such that vRw and $\mathcal{M}, w \Vdash \varphi$;*
6. *$\mathcal{M}, v \Vdash \Diamond^+\varphi$ iff there is a $w \in W$ such that vR^+w and $\mathcal{M}, w \Vdash \varphi$.*

Here, R^+ denotes the transitive closure of R.

A formula $\Diamond\varphi$ is satisfied at a vertex v if, for some vertex w, vRw and φ is satisfied at w. A formula $\Diamond^+\varphi$ is satisfied at a vertex v if there is a path $\langle v_1, \ldots, v_n \rangle$, $n \geq 2$, such that $v = v_1$, $w = v_n$, v_iRv_{i+1} for $1 \leq i < n$ and φ is satisfied at w.

Example 5. *Let $\mathcal{M}$ be the model shown (without its valuation) in figure 1. In order to illustrate the use of the logic, we can see that the following formulas are satisfied at vertex w in $\mathcal{M}$, supposing that φ is satisfied at vertex v in $\mathcal{M}$: $\mathcal{M}, w \Vdash \Diamond\varphi$, $\mathcal{M}, w \Vdash \Diamond\Diamond\Diamond\varphi$, $\mathcal{M}, w \Vdash \Diamond^+\varphi$ and $\mathcal{M}, w \Vdash \Diamond^+\Box\bot$.*

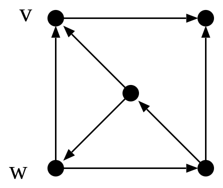

Figure 1: Model $\mathcal{M}$, where a formula φ is satisfied at vertex v.

If $\mathcal{M}, v \Vdash \varphi$ for every vertex v in a model $\mathcal{M}$, we say that φ is *globally satisfied* in $\mathcal{M}$, notation $\mathcal{M} \Vdash \varphi$. And if φ is globally satisfied in all models $\mathcal{M}$ of a frame $\mathcal{F}$, we say that φ is *valid* in $\mathcal{F}$, notation $\mathcal{F} \Vdash \varphi$.

In this work, we want to find a modal formula ϕ (for each property), such that a graph G has the desired property if and only if $\mathcal{F} \Vdash \phi$, where $\mathcal{F}$ is the frame that represents G.

For each modal logic that we consider for this task, there are two issues involved. The first one is whether the modal language has enough expressive power to describe the graph properties that we want. In case the answer is negative, we need to search for a language with greater expressive power. In case the answer is positive and we are able to find such a formula, then we need to calculate how complex (computationally) it is to use the inference mechanisms of the logic to actually test, using the formula that we found, whether a given graph has the desired property.

The issue of expressive power with respect to the language of the basic graph logic will be addressed in the next section. The issue of the complexity for testing the graph properties involves four basic decision problems.

Definition 6. *The* satisfiability problem *consists of, given a formula ϕ, determining whether there is a model $\mathcal{M}$ and a vertex v in $\mathcal{M}$ such that $\mathcal{M}, v \Vdash \phi$.*

Definition 7. *The* validity problem *consists of, given a formula ϕ, determining whether $\mathcal{F} \Vdash \phi$, for all frames $\mathcal{F}$.*

The satisfiability problem and the validity problem are duals to each other.

Definition 8. *The* model-checking problem *consists of, given a formula ϕ and a finite model $\mathcal{M} = (W, R, \mathbf{V})$, determining the set $S_{\mathcal{M}}(\phi) = \{v \in W : \mathcal{M}, v \Vdash \phi\}$.*

Definition 9. *The* frame-checking problem *consists of, given a formula ϕ and a finite frame $\mathcal{F}$, determining whether $\mathcal{F} \Vdash \phi$.*

Definition 10. *We define the length of a formula φ, denoted by $|\varphi|$, inductively in the following way: $|p| = |\top| = 1$, $|\neg\phi| = |\Diamond\phi| = |\Diamond^+\phi| = 1 + |\phi|$ and $|\phi_1 \wedge \phi_2| = 1 + |\phi_1| + |\phi_2|$. In the following logics, analogous rules apply to the new operators.*

Definition 11. *Let $\mathcal{M} = (W, R, \mathbf{V})$ be a model. Let $|W|$ be the number of vertices in W and $|R|$ the number of pairs in R. We define the size of the model (or the frame, or the graph) as $|W| + |R|$.*

Theorem 12 ([8])**.** *The satisfiability problem and the validity problem for the basic graph logic are EXPTIME-Complete in the length of the formula.*

Theorem 13. *The model-checking problem for the basic graph logic is PTIME (linear) in the product of the size of the model and the length of the formula.*

Proof. The basic graph logic is a fragment of CTL, so this result follows from the complexity of the model-checking problem for CTL, presented in [11]. □

We can provide a simple upper bound for the complexity of the frame-checking problem based on the complexity of the correspondent model-checking problem. We have that $\mathcal{F} \Vdash \phi$ if and only if $S_{\mathcal{M}}(\neg\phi) = \emptyset$ for every model $\mathcal{M}$ of $\mathcal{F}$. So, let FC be the complexity of the frame-checking problem and MC be the complexity of the model-checking problem. Then,

$$FC = O(2^{|p| \times n} \times MC),$$

where $|p|$ is the number of distinct proposition symbols that occur in the given formula ϕ and n is the number of vertices in $\mathcal{F}$. We need to apply the model-checking algorithm to every model $\mathcal{M}$ of the given frame $\mathcal{F}$. Every proposition symbol p that appears in ϕ may receive 2^n possible valuations $\mathbf{V}(p)$.

Theorem 14. *The frame-checking problem for the basic graph logic is PTIME (linear) in the length of the formula and EXPTIME in the size of the frame and in the number of distinct proposition symbols that occur in the formula.*

Proof. This result follows directly from the discussion above. □

It should be noticed that this calculation of the complexity of the frame-checking problem is just a general upper-bound and it can possibly be reduced in some concrete situations.

3 Basic Graph Logic Definability

In this section, we investigate whether some well-known global graph properties are definable or not in the language of the basic graph logic. These properties are: connectivity, acyclicity and the Hamiltonian and Eulerian properties.

The limits to the expressive power of basic modal languages are fairly well known. There are a series of standard results that state that frames that are "similar" in a number of ways must agree on the validity of formulas. We can then use these results to prove that a certain property *cannot* be expressed by any formula in the basic graph logic. To do this, we take two frames that are "similar" and show that in one the desired property holds, while in the other it does not. We present two of these "similarity" results (more details about them and other related results may be found in [8]), and then we prove some theorems for global graph properties using them.

Definition 15. *Let $\mathcal{F} = (W, R)$ and $\mathcal{F}' = (W', R')$ be two frames. A function $f : W \to W'$ is a* bounded morphism *from $\mathcal{F}$ to $\mathcal{F}'$ if it satisfies the following conditions:*

1. *f is a homomorphism with respect to R (if wRv, then $f(w)R'f(v)$);*
2. *if $f(w)R'v'$, then there is a v such that wRv and $f(v) = v'$.*

If there is a surjective bounded morphism from $\mathcal{F}$ to $\mathcal{F}'$, then we say that $\mathcal{F}'$ is a *bounded morphic image* of $\mathcal{F}$ and use the notation $\mathcal{F} \Rightarrow \mathcal{F}'$.

Definition 16. *Let $\mathcal{M} = (\mathcal{F}, \mathbf{V})$ and $\mathcal{M}' = (\mathcal{F}', \mathbf{V}')$ be two models. A function $f : W \to W'$ is a* bounded morphism *from $\mathcal{M}$ to $\mathcal{M}'$ if it satisfies the following conditions:*

1. *f is a bounded morphism from $\mathcal{F}$ to $\mathcal{F}'$;*
2. *w and $f(w)$ satisfy the same proposition symbols.*

If there is a surjective bounded morphism from $\mathcal{M}$ to $\mathcal{M}'$, then we say that $\mathcal{M}'$ is a *bounded morphic image* of $\mathcal{M}$ and use the notation $\mathcal{M} \Rightarrow \mathcal{M}'$.

Other important definitions concern collections of disjoint frames and collections of disjoint models. We say that two frames $\mathcal{F}_1 = (W_1, R_1)$ and $\mathcal{F}_2 = (W_2, R_2)$ are *disjoint frames* if and only if $W_1 \cap W_2 = \emptyset$ and we say that two models $\mathcal{M}_1 = (\mathcal{F}_1, \mathbf{V}_1)$ and $\mathcal{M}_2 = (\mathcal{F}_2, \mathbf{V}_2)$ are *disjoint models* if and only if $\mathcal{F}_1$ and $\mathcal{F}_2$ are disjoint frames.

Definition 17. *Let $\mathcal{F}_i = (W_i, R_i)$ be a collection (finite or not) of disjoint frames. Their* disjoint union *is the frame $\biguplus \mathcal{F}_i = (W, R)$, where $W = \bigcup_i W_i$ and $R = \bigcup_i R_i$.*

Definition 18. *Let $\mathcal{M}_i = (\mathcal{F}_i, \mathbf{V}_i)$ be a collection (finite or not) of disjoint models. Their* disjoint union *is the model $\biguplus \mathcal{M}_i = (\mathcal{F}, \mathbf{V})$, where $\mathcal{F}$ is the disjoint union of the frames $\mathcal{F}_i$ and, for each proposition symbol p, $\mathbf{V}(p) = \bigcup_i \mathbf{V}_i(p)$.*

Below are two basic theorems about the definability of properties that are going to be used throughout the next subsections. Their proofs for a basic modal language that contains only $\Diamond$ can be found in [8]. It is not difficult to extend that proof to the language of the basic graph logic, which contains both $\Diamond$ and $\Diamond^+$.

Theorem 19. *Let $\mathcal{M} = (W, R, \mathbf{V})$ and $\mathcal{M}' = (W', R', \mathbf{V}')$ be two models such that $\mathcal{M} \Rrightarrow \mathcal{M}'$. Then, $\mathcal{M}, w \Vdash \phi$ if and only if $\mathcal{M}', f(w) \Vdash \phi$.*

Corollary 20. *Let $\mathcal{F} = (W, R)$ and $\mathcal{F}' = (W', R')$ be two frames such that $\mathcal{F} \Rrightarrow \mathcal{F}'$. If $\mathcal{F} \Vdash \phi$, then $\mathcal{F}' \Vdash \phi$.*

Theorem 21. *Let $\mathcal{M}_i = (W_i, R_i, \mathbf{V}_i)$ be a collection (finite or not) of disjoint models and $\biguplus \mathcal{M}_i = (W, R, \mathbf{V})$ their disjoint union. Then, $\mathcal{M}_i, w \Vdash \phi$ if and only if $\biguplus \mathcal{M}_i, w \Vdash \phi$.*

Corollary 22. *Let $\mathcal{F}_i = (W_i, R_i)$ be a collection (finite or not) of disjoint frames and $\biguplus \mathcal{F}_i = (W, R)$ their disjoint union. If $\mathcal{F}_i \Vdash \phi$ for every i, then $\biguplus \mathcal{F}_i \Vdash \phi$.*

Theorem 23. *The class of finite frames (which is equivalent to our definition of a graph) is not definable in the basic graph logic.*

Proof. The disjoint union of an infinite collection of finite disjoint frames is not finite. By corollary 22, since this property is not preserved under taking disjoint unions, it is not definable in the basic graph logic. ☐

3.1 Connectivity

Definition 24. *We can define two levels of connectivity for a graph. Firstly, a graph G is said to be* (weakly) connected *if and only if, for any two vertices v and w in G, there is a path from v to w in the underlying undirected graph of G. Secondly, a graph G is said to be* strongly connected *if and only if, for any two vertices v and w in G, there is a path from v to w in G itself.*

Theorem 25. *Weak and strong connectivity are not definable in the basic graph logic.*

Proof. The disjoint union of connected graphs is not a connected graph. By corollary 22, since connectivity is not preserved under taking disjoint unions, it is not definable in the basic graph logic. ☐

3.2 Acyclicity

Definition 26. *A graph G is said to be acyclic if and only if there is no path in G that is a cycle, as defined in the first section.*

Theorem 27. *Acyclicity is not definable in the basic graph logic.*

Proof. We can take a frame $\mathcal{F} = (W, R)$ where $W = \mathbb{N}$ and $R = \{\langle i, i+1\rangle, i \in \mathbb{N}\}$ and a frame $\mathcal{F}' = (W', R')$ where $W' = \{O, E\}$ and $R' = \{\langle O, E\rangle, \langle E, O\rangle\}$. If we define f as $f(i) = E$ if i is even and $f(i) = O$ otherwise, we have that f is a surjective bounded morphism between $\mathcal{F}$ and $\mathcal{F}'$. But $\mathcal{F}$ is acyclic while $\mathcal{F}'$ is not. Hence, by corollary 20, since acyclicity is not preserved under bounded morphic images, it is not definable in the basic graph logic. $\square$

At first, it may seem that the above theorem is too generic for our needs, since we only need to be able to define acyclicity on finite frames and we use an infinite frame as part of our proof. However, as theorem 23 shows, we cannot write a formula that describes "acyclicity on finite frames" or any other property "on finite frames".

3.3 Hamiltonian Graphs

Definition 28. *A connected graph G is said to be Hamiltonian if and only if there is a cycle in G that goes through every vertex of it.*

Theorem 29. *The class of Hamiltonian graphs is not definable in the basic graph logic.*

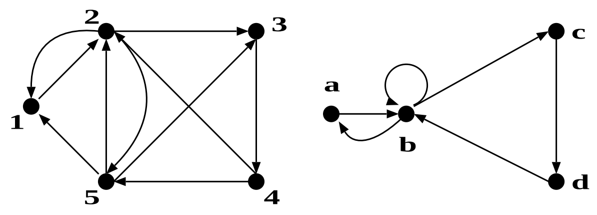

Figure 2: Graph 1,2,3,4,5 is Hamiltonian and graph a,b,c,d is not.

Proof. From figure 2, let $f = \{(1, a), (2, b), (3, c), (4, d), (5, b)\}$. It is straightforward to prove that f is a surjective bounded morphism. By corollary 20, since the Hamiltonian property is not preserved under bounded morphic images, it is not definable in the basic graph logic. $\square$

3.4 Eulerian Graphs

Definition 30. *A connected graph G is said to be Eulerian if and only if there is a closed path in G in which every edge of it appears exactly once.*

Theorem 31 ([9]). *A connected graph G is Eulerian if and only if the out-degree of every vertex of G is equal to its in-degree.*

Theorem 32. *The class of Eulerian graphs is not definable in the basic graph logic.*

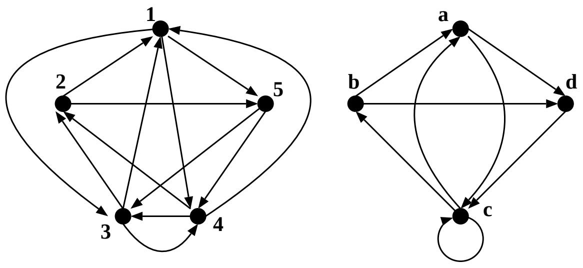

Figure 3: Graph 1,2,3,4,5 is Eulerian and graph a,b,c,d is not.

Proof. From figure 3, let $f = \{(1, a), (2, b), (3, c), (4, c), (5, d)\}$. It is straightforward to prove that f is a surjective bounded morphism. By corollary 20, since the Eulerian property is not preserved under bounded morphic images, it is not definable in the basic graph logic. $\square$

3.5 The Modal μ-Calculus

Looking at the results of the previous subsections, we see that, unfortunately, the language of the *basic graph logic* does not have enough expressive power to define the properties that we want. We need a stronger language. One idea could be to use the modal μ-calculus [10, 26]. Its language incorporates fixpoint operators and is very expressive. In fact, not only the *basic graph logic* can be embedded into the μ-calculus, but so can be the temporal logics LTL, CTL and CTL* [12].

Unfortunately, even with all this expressive power, the language of the μ-calculus fails to express these properties because of the same reasons exposed in the previous subsections. This happens because μ-calculus formulas, as the basic graph formulas, are invariant under bisimulations (disjoint unions and bounded morphisms are special cases of bisimulation). In fact, the μ-calculus is the bisimulation-invariant fragment of Monadic Second-Order Logic (MSOL) [10].

To bypass this problem, we introduce a different kind of language in the next section. This language has a mechanism to name vertices of the model and allows us to express the graph properties that we want.

4 Hybrid Graph Logic

As was shown in the previous section, the language of the basic graph logic does not have enough expressive power to describe the properties that we want. In order to achieve our goal, we need a logic that has a language with more expressive power but, if possible, is still decidable with respect to the problems stated in the definitions 6 until 9.

One interesting class of logics to take into consideration is the class of *hybrid logics* [3, 8]. In these logics, there is a new kind of atomic symbol: *nominals*. Nominals behave similarly to proposition symbols. The key difference between them is related to their valuation in a model. While the set $\mathbf{V}(p)$ for a proposition symbol p can be any element of $\mathcal{P}(W)$, the set $\mathbf{V}(i)$ for a nominal i has to be a singleton set. This way, each nominal is satisfied at exactly one vertex, and thus, can be used to reference a unique vertex of the model.

A hybrid extension of our previous logic is an interesting choice because of a combination of factors. Its language has an improved expressive power, since hybrid formulas are no longer invariant under neither disjoint unions nor bounded morphic images [3], but it is still a decidable logic, as discussed in the following subsection.

In this section, we define an extension of the basic graph logic that includes nominals. We call it *hybrid graph logic*. After that, we try to express, in this new logic, the graph properties that we are discussing.

4.1 Language

Definition 33. *The language of the* hybrid graph logic *is a hybrid language consisting of a set Φ of countably many proposition symbols (the elements of Φ are denoted by $p_1, p_2, \ldots$), a set $\mathcal{L}$ of countably many nominals (the elements of $\mathcal{L}$ are denoted by $i_1, i_2, \ldots$) such that $\Phi \cap \mathcal{L} = \emptyset$ (the elements of $\Phi \cup \mathcal{L}$ are called* atoms*), the boolean connectives $\neg$ and $\wedge$ and the modal operators $@_i$ (called* satisfaction operators*), for each nominal i, $\Diamond$ and $\Diamond^+$. The formulas are defined as follows:*

$$\varphi ::= p \mid i \mid \top \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \Diamond\varphi \mid \Diamond^+\varphi \mid @_i\varphi$$

Again, we freely use the standard abbreviations $\vee$, $\rightarrow$, $\leftrightarrow$, $\bot$, $\Box\varphi$, $\Box^+\varphi$, $\Diamond^*\varphi$ and $\Box^*\varphi$.

The definition of a *frame* is the same as the one from section 2. The definition of a *model* is slightly different.

Definition 34. *A* model *for the hybrid graph logic is a pair $\mathcal{M} = (\mathcal{F}, \mathbf{V})$, where $\mathcal{F}$ is a frame and $\mathbf{V}$ is a valuation function mapping proposition symbols into subsets of W, i.e., $\mathbf{V} : \Phi \mapsto \mathcal{P}(W)$, and mapping nominals into singleton subsets of W, i.e, if i is a nominal then $\mathbf{V}(i) = \{v\}$ for some $v \in W$. We call this unique vertex that belongs to $\mathbf{V}(i)$ the* denotation *of i under $\mathbf{V}$. We can also say that i* denotes *or* names *the single vertex belonging to $\mathbf{V}(i)$.*

Definition 35. *The notion of satisfaction is defined adding two extra clauses to definition 4:*

1. *$\mathcal{M}, v \Vdash i$ iff $v \in \mathbf{V}(i)$;*
2. *$\mathcal{M}, v \Vdash @_i\varphi$ iff $\mathcal{M}, d_i \Vdash \varphi$, where d_i is the denotation of i under $\mathbf{V}$.*

For each nominal i, the formula $@_i\varphi$ means that if $\mathbf{V}(i) = \{v\}$ then φ is satisfied at v. As in section 2, if $\mathcal{M}, v \Vdash \varphi$ for every vertex v, we say that φ is *globally satisfied* in the model $\mathcal{M}$ ($\mathcal{M} \Vdash \varphi$) and if φ is globally satisfied in all models $\mathcal{M}$ of a frame $\mathcal{F}$, we say that φ is *valid* in $\mathcal{F}$ ($\mathcal{F} \Vdash \varphi$).

Theorem 36.
1. *$\mathcal{M}, v \Vdash \neg @_i\phi$ iff $\mathcal{M}, v \Vdash @_i\neg\phi$;*
2. *$\mathcal{M}, v \Vdash @_i(\phi_1 \wedge \phi_2)$ iff $\mathcal{M}, v \Vdash @_i\phi_1 \wedge @_i\phi_2$;*
3. *$\mathcal{M}, v \Vdash @_i(\phi_1 \vee \phi_2)$ iff $\mathcal{M}, v \Vdash @_i\phi_1 \vee @_i\phi_2$;*
4. *$\mathcal{M}, v \Vdash @_i(\phi_1 \rightarrow \phi_2)$ iff $\mathcal{M}, v \Vdash @_i\phi_1 \rightarrow @_i\phi_2$.*

Proof. 1. $\mathcal{M}, v \Vdash \neg @_i \phi$ iff $\mathcal{M}, v \nVdash @_i \phi$ iff $\mathcal{M}, d_i \nVdash \phi$, where d_i is the denotation of i under the valuation of $\mathcal{M}$, iff $\mathcal{M}, d_i \Vdash \neg\phi$ iff $\mathcal{M}, v \Vdash @_i \neg\phi$;

2. $\mathcal{M}, v \Vdash @_i(\phi_1 \wedge \phi_2)$ iff $\mathcal{M}, d_i \Vdash \phi_1 \wedge \phi_2$, where d_i is the denotation of i under the valuation of $\mathcal{M}$, iff $\mathcal{M}, d_i \Vdash \phi_1$ and $\mathcal{M}, d_i \Vdash \phi_2$ iff $\mathcal{M}, v \Vdash @_i\phi_1$ and $\mathcal{M}, v \Vdash @_i\phi_2$ iff $\mathcal{M}, v \Vdash @_i\phi_1 \wedge @_i\phi_2$;

3. It follows directly from the previous items;

4. It follows directly from the previous items.

□

Theorem 37 ([2]). *The satisfiability problem and the validity problem for the hybrid graph logic are EXPTIME-Complete in the length of the formula.*

Theorem 38 ([16]). *The model-checking problem for the hybrid graph logic is PTIME (linear) in the product of the size of the model and the length of the formula.*

The upper bound for the complexity of the frame-checking problem is a little different in the case of a hybrid logic, because of the special restriction on the valuation of nominals. For hybrid logics, the upper bound has the form

$$FC = O(2^{|p| \times n} \times n^{|i|} \times MC), \tag{1}$$

where $|p|$ is the number of distinct proposition symbols that occur in the given formula ϕ, $|i|$ is the number of distinct nominals that occur in ϕ and n is the number of vertices in $\mathcal{F}$. We need to apply the model-checking algorithm to every model $\mathcal{M}$ of the given frame $\mathcal{F}$. Every proposition symbol p that appears in ϕ may receive 2^n possible valuations $\mathbf{V}(p)$, while every nominal i may only receive n possible valuations $\mathbf{V}(i)$.

Theorem 39. *The frame-checking problem for the hybrid graph logic is PTIME (linear) in the length of the formula and EXPTIME in the size of the frame, in the number of distinct proposition symbols that occur in the formula and in the number of distinct nominals that occur in the formula.*

Proof. This result follows directly from the discussion above. □

We can see then that the hybrid graph logic is indeed a very interesting choice, since we get a greater expressive power without any relevant increase in computational complexity.

4.2 Hybrid Graph Logic Definability

In the *hybrid graph logic* we can now express at least two of the properties that we want.

Theorem 40. *Let G be a graph, G' be its underlying undirected graph, $\mathcal{F}$ be the frame that represents G and $\mathcal{F}'$ be the frame that represents G'. G is* strongly connected *if and only if $\mathcal{F} \Vdash \phi$ and* (weakly) connected *if and only if $\mathcal{F}' \Vdash \phi$, where ϕ is the formula*

$$\phi = @_i(\neg j \to \Diamond^+ j).$$

Proof. We prove the theorem only for strong connectivity. The other case is completely analogous.

($\Leftarrow$) Suppose that $\mathcal{F} \Vdash \phi$ but G is not strongly connected. Then, there are at least two distinct vertices v, w in G such that w is not reachable from v. We will evaluate ϕ in a model with a valuation $\mathbf{V}$ such that $\mathbf{V}(i) = \{v\}$ and $\mathbf{V}(j) = \{w\}$. Then, for any vertex u in G, $(\mathcal{F}, \mathbf{V}), u \nVdash \phi$, contradicting the fact that ϕ is valid in $\mathcal{F}$.

($\Rightarrow$) Suppose that G is strongly connected but $\mathcal{F} \nVdash \phi$. Then, there is a valuation $\mathbf{V}$ and a vertex u such that $(\mathcal{F}, \mathbf{V}), u \nVdash \phi$, which is equivalent, by theorem 36, to $(\mathcal{F}, \mathbf{V}), u \Vdash \theta$, where $\theta = @_i \neg j \wedge @_i \neg \Diamond^+ j$. Let $\mathbf{V}(i) = \{v\}$ and $\mathbf{V}(j) = \{w\}$. If $v = w$, then θ is not satisfied, so we may assume that $v \neq w$. Then, for θ to be satisfied, we need $(\mathcal{F}, \mathbf{V}), u \Vdash @_i \neg \Diamond^+ j$. This, on the other hand, is equivalent to vR^+w being false in G, which means that w is not reachable from v. This contradicts the fact that G is strongly connected. □

We now want to determine how complex it is to test whether a graph is connected using the above formula ϕ. This consist of, given a frame $\mathcal{F}$ that represents the graph, frame-check whether $\mathcal{F} \Vdash \phi$. We should notice first that there are no proposition symbols in ϕ and only two distinct nominals. Finally, the length of ϕ is constant and does not depend on the size of the graph. Let CON be the complexity of testing whether a graph is connected through a frame-checking of ϕ. Then, taking into account these observations and the formula in equation (1), we have that

$$CON = O(n^2 \times MC),$$

where, for the formula ϕ, MC is PTIME (in fact linear) in the size of the graph.

Theorem 41. *The complexity to check whether a graph is connected using the above formula ϕ is PTIME (cubic) in the size of the graph.*

Proof. This result follows directly from the discussion above. □

Theorem 42. *A graph G contains a closed path if and only if it contains a cycle.*

Proof. The right to left direction is immediate, as cycles are a particular case of closed paths. For the left to right direction, we proceed by induction on the length $n \geq 2$ of the closed path. If G contains a closed path of length $n = 2$, then this closed path is also a cycle. Now suppose that, for any $n < k$, if G contains a closed path of length n, then it contains a cycle. If G contains a closed path $\langle v_1, \ldots, v_k \rangle$ of length k that is not a cycle, then there are i and j such that $1 \leq i < j < k$ and $v_i = v_j$. But then, $\langle v_i, \ldots, v_j \rangle$ is a closed path of length smaller than k. Then, by the induction hypothesis, G contains a cycle. □

Theorem 43. *A graph G with frame $\mathcal{F}$ is* acyclic *if and only if $\mathcal{F} \Vdash \phi$, where ϕ is the formula*

$$\phi = @_i \neg \Diamond^+ i.$$

Proof. ($\Leftarrow$) Suppose that $\mathcal{F} \Vdash \phi$ but G is not acyclic. Then, there is at least one vertex v in G such that there is a path in G from v to itself. We will evaluate ϕ in a model with a valuation $\mathbf{V}$ such that $\mathbf{V}(i) = \{v\}$. Then, for any vertex u in G, $(\mathcal{F}, \mathbf{V}), u \nVdash \phi$, contradicting the fact that ϕ is valid in $\mathcal{F}$.

($\Rightarrow$) Suppose that G is acyclic but $\mathcal{F} \not\Vdash \phi$. Then, there is a valuation $\mathbf{V}$ and a vertex u such that $(\mathcal{F}, \mathbf{V}), u \not\Vdash \phi$, which is equivalent, by theorem 36, to $(\mathcal{F}, \mathbf{V}), u \Vdash \theta$, where $\theta = @_i \Diamond^+ i$. Let $\mathbf{V}(i) = \{v\}$. Then this, on the other hand, is equivalent to vR^+v being true in G, which means that v is reachable from itself. This implies that G contains a closed path, which, by theorem 42, contradicts the fact that G is acyclic. $\square$

Let us now determine how complex it is to test whether a graph is acyclic using the above formula ϕ. Again, there are no proposition symbols in ϕ and, this time, ϕ has only one nominal. Also, the length of ϕ is, again, constant and does not depend on the size of the graph. Let ACY be the complexity of testing whether a graph is acyclic through a frame-checking of ϕ. Then, taking into account these observations and the formula in equation (1), we have that

$$ACY = O(n \times MC),$$

where, for the formula ϕ, MC is PTIME (in fact linear) in the size of the graph.

Theorem 44. *The complexity to check whether a graph is acyclic using the above formula ϕ is PTIME (quadratic) in the size of the graph.*

Proof. This result follows directly from the discussion above. $\square$

We consider the complexities in theorems 41 and 44 to be satisfactory. It is not necessary to search for alternative forms to express these properties in this logic or in another logic.

Before trying to find a formula to describe the Hamiltonian graphs, we need to consider some graph-theoretical issues. In graph theory [9], there is no known result that states a necessary and sufficient condition for a graph to be Hamiltonian. If we could find a formula that describes the Hamiltonian graphs without having to describe the Hamiltonian cycle itself, we would be finding such necessary and sufficient condition. Thus, what our formula does is to inspect all of the paths in the graph, searching for a Hamiltonian cycle. Not surprisingly then, the only formula we could find in this simple language to describe the Hamiltonian property has length proportional to $n!$, where n is the number of vertices in the graph.

Let $\mathcal{L}_n = \{i_1, \ldots, i_n\}$ be a set containing n nominals. Before defining a formula for the Hamiltonian property, we will define a formula that is globally satisfied in a model under a valuation $\mathbf{V}$ if and only if $\mathbf{V}(i_k) \neq \mathbf{V}(i_l)$, for all $i_k, i_l \in \mathcal{L}_n$ such that $k \neq l$.

Lemma 45. *A valuation satisfies $\mathbf{V}(i_k) \neq \mathbf{V}(i_l)$, for all $i_k, i_l \in \mathcal{L}_n$ such that $k \neq l$, if and only if $(\mathcal{F}, \mathbf{V}) \Vdash \psi_n$, where ψ_n is the formula*

$$\psi_n = \bigwedge_{1 \leq k < n} \left(@_{i_k} \bigwedge_{k < l \leq n} \neg i_l \right).$$

Proof. It follows directly from the definitions of a valuation for a nominal and of satisfaction for a nominal and for a formula $@_i \varphi$. $\square$

Definition 46. *Let $\mathcal{L}_n = \{i_1, \ldots, i_n\}$ be a set of n nominals. Then, we define the following formulas with these nominals:*

$$\Delta_n = i_n \wedge \Diamond i_1;$$

$$\Delta_k = i_k \wedge \Diamond \Delta_{k+1}, \text{ for } 1 \leq k \leq n-1.$$

We now define a set F of permutations of the nominals in $\mathcal{L}_n$. This set has $n!$ elements. We represent a permutation as a bijective function $\sigma : \{1, \ldots, n\} \mapsto \mathcal{L}_n$.

Theorem 47. *A connected graph G (with n vertices) with frame $\mathcal{F}$ is* Hamiltonian *if and only if $\mathcal{F} \Vdash \phi$, where ϕ is the formula*

$$\phi = \psi_n \to \delta_n,$$

where ψ_n is the formula from lemma 45 and

$$\delta_n = \bigvee_{\sigma \in F} (\sigma(1) \wedge \Diamond(\sigma(2) \wedge \Diamond(\sigma(3) \ldots (\sigma(n-1) \wedge \Diamond(\sigma(n) \wedge \Diamond\sigma(1)) \ldots).$$

Proof. ($\Leftarrow$) Suppose that the formula ϕ is valid in $\mathcal{F}$. We will evaluate ϕ in an arbitrary vertex v of a model with a valuation $\mathbf{V}$ such that $\mathbf{V}$ satisfies ψ_n and $\mathbf{V}(i_1) = \{v\}$. First, this means that each nominal is denoting a different vertex. Second, $\mathbf{V}$ must also satisfy δ_n. If δ_n is satisfied, at least one of the members in its disjunction is satisfied. Let σ' be the permutation correspondent to this member. To simplify the notation and without loss of generality, we consider that $\sigma'(k) = i_k$. Thus, $(\mathcal{F}, \mathbf{V}), v \Vdash \Delta_1$. From this and from the construction rule of the formulas Δ_k, we have that there are vertices w_k in G such that $(\mathcal{F}, \mathbf{V}), w_k \Vdash \Delta_k$, $w_k R w_{k+1}$, for $2 \leq k \leq n-1$, vRw_2 and $w_n R v$. We then have that $\langle v, w_2, \ldots, w_n, v\rangle$ is a Hamiltonian cycle in G.

($\Rightarrow$) Suppose that there is a Hamiltonian cycle $\langle v_1, \ldots v_n, v_1\rangle$ in G. We denote the vertices with nominals in such a way that $\mathcal{L}_n = \{i_1, \ldots, i_n\}$ and i_k denotes v_k. This valuation satisfies ψ_n. We have that $v_n R v_1$, so Δ_n is satisfied at v_n. Similarly, Δ_k is satisfied at v_k. Since Δ_1 is a member of the disjunction in δ_n, δ_n is satisfied at v_1. Repeating the previous line of thought, but starting the cycle at v_2, v_3 and so on, we can see that δ_n is also satisfied at all the vertices in the cycle. Since the cycle is Hamiltonian, this means that δ_n is satisfied in all the vertices of G. Since ϕ is trivially satisfied in all the valuations that do not satisfy ψ_n, we only need to think about the ones that do. If we change the valuation of the nominals in $\mathcal{L}_n$ to another one that satisfies ψ_n, this is equivalent to applying a permutation to the nominals. As δ_n contains a member in its disjunction for each permutation, we conclude that in fact ϕ is valid in $\mathcal{F}$. □

In this case, the number of distinct nominals in the formula and the length of the formula are both linked to the size of the graph. Even though the frame-checking complexity is PTIME in the length of the formula, since the formula has a factorial length in the size of the graph, it is infeasible to frame-check this formula. In the case of the Hamiltonian property, we must search for alternative forms to express it using other logics. Two attempts to do this are presented in the next two sections.

The difficulty in finding a formula to describe Eulerian graphs is of a completely different nature. Here, the limitation is on the language, not on the theoretical definition of the property. There is a known result that states a necessary and sufficient condition for a graph to be Eulerian (theorem 31), so the argument used above for Hamiltonian graphs is not valid here. However, the hybrid graph language does not have the expressive power, at least not without performing a brute-force search through every possible path in the graph, in a similar way as the formula in theorem 47 does, to state cardinality conditions on edges incident from and to a vertex, as is needed in theorem 31.

The other way to describe the Eulerian property would be to find a formula that explicitly describes an Eulerian path in the graph. However, it is very hard to find such a formula, since the hybrid graph logic and many other modal logics are not good to talk about edges. One of the reasons for that is the fact that the modal operator $\Diamond$ does not differentiate between edges incident from a vertex. We now, using nominals, have names for vertices, but we still cannot keep track of which edges we are using when we walk in a graph. This suggests that a possible solution would be to find a way to name the edges in some similar way to the use of nominals to name vertices. We do this in section 7, where we describe a method to name edges within the framework of a hybrid logic and use it to find a formula for the Eulerian property.

5 The Temporal Logic Hybrid CTL*

The fact that the formula previously introduced to describe Hamiltonian graphs has factorial size makes its verification infeasible. Obviously, we can never expect to verify the Hamiltonian property in polynomial time, since determining whether a graph is Hamiltonian is an NP-Complete problem [18], but we certainly wish to verify it with a lower complexity than factorial time. This is our goal in this section and in the next one.

In our first attempt to write a short (with polynomial length) formula to describe the class of Hamiltonian graphs, we use the temporal branching-time logic CTL* [21] with nominals (*hybrid CTL**). We could use the full hybrid μ-calculus presented in [22], since it contains the hybrid CTL* [12, 22]. But we will not do this, since the hybrid CTL* is strong enough for what we want to do and its formulas are easier to read and to understand than hybrid μ-calculus formulas.

5.1 Language

Definition 48. *The hybrid CTL* language is a temporal language consisting of a set Φ of countably many proposition symbols (the elements of Φ are denoted by $p_1, p_2, \ldots$), a set $\mathcal{L}$ of countably many nominals (the elements of $\mathcal{L}$ are denoted by $i_1, i_2, \ldots$) such that $\Phi \cap \mathcal{L} = \emptyset$, the boolean connectives $\neg$ and $\wedge$ and the operators $@_i$, for each nominal i, and* **A**, **E**, **X**, **F**, **G** *and* **U**. *Formulas are divided into* vertex formulas **S** *and* path formulas **P** *defined by the following mutual induction:*

$$\mathbf{S} ::= p \mid i \mid \top \mid \neg\mathbf{S} \mid \mathbf{S_1} \wedge \mathbf{S_2} \mid \mathbf{AP} \mid \mathbf{EP} \mid @_i\mathbf{S}$$

$$\mathbf{P} ::= \mathbf{S} \mid \neg\mathbf{P} \mid \mathbf{P_1} \wedge \mathbf{P_2} \mid \mathbf{XP} \mid \mathbf{FP} \mid \mathbf{GP} \mid \mathbf{P_1 U P_2}$$

The language of hybrid CTL* is then the set of all vertex formulas *generated by the above rules.*

The definition of a *frame* and of a *model* are the same as the ones from the previous section. The notion of satisfaction in hybrid CTL* is defined as follows:

Definition 49. *Let $\mathcal{M} = (\mathcal{F}, \mathbf{V})$ be a model. We also need the auxiliary notation that if $\pi = \langle s_0, s_1, \ldots \rangle$ is a path, we denote by π^i the suffix of π starting at s_i. The notion of* satisfaction *of a vertex formula S in a model $\mathcal{M}$ at a vertex v or of a path formula P in a model $\mathcal{M}$ at a path π, notation $\mathcal{M}, v \Vdash S$ and $\mathcal{M}, \pi \Vdash P$, can be inductively defined as follows:*

1. *$\mathcal{M}, v \Vdash p$ iff $v \in \mathbf{V}(p)$;*
2. *$\mathcal{M}, v \Vdash i$ iff $v \in \mathbf{V}(i)$;*
3. *$\mathcal{M}, v \Vdash \top$ always;*
4. *$\mathcal{M}, v \Vdash \neg S$ iff $\mathcal{M}, v \nVdash S$;*
5. *$\mathcal{M}, v \Vdash S_1 \wedge S_2$ iff $\mathcal{M}, v \Vdash S_1$ and $\mathcal{M}, v \Vdash S_2$;*
6. *$\mathcal{M}, v \Vdash \mathbf{A}P$ iff for every path π starting in v, $\mathcal{M}, \pi \Vdash P$;*
7. *$\mathcal{M}, v \Vdash \mathbf{E}P$ iff there is a path π starting in v such that $\mathcal{M}, \pi \Vdash P$;*
8. *$\mathcal{M}, v \Vdash @_i S$ iff $\mathcal{M}, d_i \Vdash S$, where d_i is the denotation of i under $\mathbf{V}$;*
9. *$\mathcal{M}, \pi \Vdash S$ iff v is the first vertex of π and $\mathcal{M}, v \Vdash S$;*
10. *$\mathcal{M}, \pi \Vdash \neg P$ iff $\mathcal{M}, \pi \nVdash P$;*
11. *$\mathcal{M}, \pi \Vdash P_1 \wedge P_2$ iff $\mathcal{M}, \pi \Vdash P_1$ and $\mathcal{M}, \pi \Vdash P_2$;*
12. *$\mathcal{M}, \pi \Vdash \mathbf{X}P$ iff $\mathcal{M}, \pi^1 \Vdash P$;*
13. *$\mathcal{M}, \pi \Vdash \mathbf{F}P$ iff there is a $k \geq 0$ such that $\mathcal{M}, \pi^k \Vdash P$;*
14. *$\mathcal{M}, \pi \Vdash \mathbf{G}P$ iff for all $k \geq 0$, $\mathcal{M}, \pi^k \Vdash P$;*
15. *$\mathcal{M}, \pi \Vdash P_1 \mathbf{U} P_2$ iff there is a $k \geq 0$ such that $\mathcal{M}, \pi^k \Vdash P_2$ and for all $0 \leq j < k$, $\mathcal{M}, \pi^j \Vdash P_1$.*

We should think of **A** as "for all paths starting in the current vertex", **E** as "there is a path starting in the current vertex such that...", **X** as "in the next vertex of the current path", **F** as "in the current vertex or at some future vertex in the current path", **G** as "in the current vertex and for all future vertices in the current path" and **U** as "the first formula is satisfied in a path until the second formula is satisfied in this path, and the second formula will be satisfied eventually".

Lemma 50 ([13])**.** *The satisfiability problem and the validity problem for CTL* are 2-EXPTIME-Complete in the length of the formula.*

Lemma 51 ([12])**.** *Every CTL* formula can be translated into a μ-calculus formula. The complexity of this translation is 2-EXPTIME in the length of the original formula. The translated formula can be written with exponential length with respect to the length of the original formula.*

Theorem 52. *Every hybrid CTL* formula can be translated into a hybrid μ-calculus formula.*

Proof. The only formulas that are in hybrid CTL*, but are not in CTL* are the formulas that contain nominals or satisfaction operators. We can easily extend the translation of lemma 51 to also cover these cases, since nominals and satisfaction operators are present in the hybrid μ-calculus [22]. First, nominals are translated in the same way as proposition symbols: $tr(i) = i$. Second, we translate formulas of the form $@_i\psi$ in the obvious way: $tr(@_i\psi) = @_i tr(\psi)$. □

Theorem 53 ([22])**.** *The satisfiability problem and the validity problem for the hybrid μ-calculus are EXPTIME-Complete in the length of the formula.*

Theorem 54. *The satisfiability problem and the validity problem for hybrid CTL* are 2-EXPTIME-Hard in the length of the formula and are decidable.*

Proof. The 2-EXPTIME-hardness follows from lemma 50 and the decidability follows from theorems 52 and 53. □

Theorem 55. *The model-checking problem for the hybrid CTL* is PTIME (linear) in the size of the model and EXPTIME in the length of the formula.*

Proof. This follows from an elementary combination of the CTL* model-checking algorithm presented in [11], which is PTIME in the size of the model and EXPTIME in the length of the formula, with the basic hybrid logic model-checking algorithm presented in [16], which is PTIME both in the size of the model and in the length of the formula. □

The upper bound for the complexity of the frame-checking problem is the same as in the previous section:

$$FC = O(2^{|p| \times n} \times n^{|i|} \times MC), \tag{2}$$

where $|p|$ is the number of distinct proposition symbols that occur in the given formula ϕ, $|i|$ is the number of distinct nominals that occur in ϕ and n is the number of vertices in $\mathcal{F}$.

Theorem 56. *The frame-checking problem for the hybrid CTL* is EXPTIME in the length of the formula, in the size of the frame, in the number of distinct proposition symbols that occur in the formula and in the number of distinct nominals that occur in the formula.*

Proof. This result follows directly from the discussion above. □

5.2 The Hamiltonian Property

Let G be a graph with n vertices and let $\mathcal{L}_n = \{i_1, \ldots, i_n\}$. Let us add a loop to all the vertices in G. We can then define the formula that is valid if and only if G is Hamiltonian.

Theorem 57. *A connected graph G (with n vertices) with frame $\mathcal{F}$ is* Hamiltonian *if and only if $\mathcal{F} \Vdash \phi$, where ϕ is the formula*

$$\phi = \psi_n \to \varepsilon_n,$$

where ψ_n is the formula from lemma 45 and

$$\varepsilon_n = @_{i_1}\mathbf{E}[\mathbf{XF}i_1 \wedge \mathbf{F}i_2 \wedge \ldots \wedge \mathbf{F}i_n \wedge$$
$$\wedge \mathbf{XG}(i_1 \to \mathbf{G}i_1) \wedge \mathbf{G}(i_2 \to \mathbf{XG}\neg i_2) \wedge \ldots \wedge \mathbf{G}(i_n \to \mathbf{XG}\neg i_n)].$$

Proof. ($\Leftarrow$) Suppose that the formula ϕ is valid in $\mathcal{F}$. We will evaluate ϕ in an arbitrary vertex v of a model with a valuation $\mathbf{V}$ such that $\mathbf{V}$ satisfies ψ_n. First, this means that each nominal is denoting a different vertex. Second, $\mathbf{V}$ must also satisfy ε_n. If ε_n is satisfied, then there is a path π in G starting at the vertex denoted by i_1 such that the formula inside the brackets in ε_n is satisfied in this path for the valuation $\mathbf{V}$. This means that $\mathbf{F}i_k$ is satisfied for $2 \leq k \leq n$ and $\mathbf{XF}i_1$ is satisfied. Thus, every vertex in G appears at least once in π. Also, the formulas $\mathbf{G}(i_k \to \mathbf{XG}\neg i_k)$ are satisfied for $2 \leq k \leq n$. This means that the vertices denoted by i_k, for $2 \leq k \leq n$, appear exactly once in the path. Finally, $\mathbf{XG}(i_1 \to \mathbf{G}i_1)$ is satisfied, which means that after the second visit to the vertex denoted by i_1, no other vertex in G is visited anymore in the path. So, if we disregard the final looping in the vertex denoted by i_1, we have a path that starts and ends in this vertex and visit every other vertex of G exactly once. This is exactly a Hamiltonian cycle.

($\Rightarrow$) Suppose that there is a Hamiltonian cycle $\langle v_1, \ldots v_n, v_1\rangle$ in G. We denote the vertices with nominals in such a way that $\mathcal{L}_n = \{i_1, \ldots, i_n\}$ and i_k denotes v_k. This valuation satisfies ψ_n. Consider the extended path $\langle v_1, \ldots v_n, v_1, v_1, \ldots\rangle$. Clearly, the components of the conjunction inside brackets in ε_n are satisfied in this path. Then, since this path starts at v_1, $\mathbf{E}[\ldots]$ is satisfied in v_1 and ε_n is satisfied at all vertices, because v_1 is denoted by i_1. Since ϕ is trivially satisfied in all the valuations that do not satisfy ψ_n, we only need to think about the ones that do. Changing the valuation of the nominals in $\mathcal{L}_n$ to another one that satisfies ψ_n is harmless, since the @ operator in the beginning of the formula ε_n is marking a starting point in the cycle, which contains all the vertices. This means that no matter where $\mathbf{V}(i_1)$ send us, it will be a point inside the cycle. Thus, ϕ is valid in $\mathcal{F}$. □

Let us now determine how complex it is to test whether a graph is Hamiltonian using the above formula ϕ. First of all, we now have a formula with polynomial (quadratic) length in the size of the graph. Also, there are no proposition symbols. From the structure of the formula, it is not difficult to see that it would be a waste of time to check valuations that do not assign distinct vertices to each nominal, since ψ_n is a precondition of an implication. Besides that, the formula ε_n is completely symmetric with respect to the nominals $i_2, \ldots, i_n$. Thus, for each possible valuation of i_1, we only need to check one arbitrary valuation for $i_2, \ldots, i_n$ such that each of these nominals names a distinct vertex. Let HAM be the complexity of testing whether a graph is Hamiltonian through a frame-checking of ϕ. Then, taking into account the above observations and the formula in equation (2), we have that

$$HAM = O(n \times MC),$$

where, for the formula ϕ, MC is EXPTIME in the size of the graph.

Theorem 58. *The complexity to check whether a graph is Hamiltonian using the above formula ϕ is EXPTIME in the size of the graph.*

Proof. This result follows directly from the discussion above. □

6 Hybrid Graph Logic with the $\downarrow$ binder

In the previous section, we were able to reduce the upper-bound of the complexity of testing whether a graph is Hamiltonian using a frame-checking method from a factorial upper-bound ($O(k!)$, where k is the size of the frame) to an exponential upper-bound ($O(2^k)$). The key point involved in this reduction was the fact that in the second logic, we were able to describe the Hamiltonian property with a formula that has polynomial length in the size of the graph, instead of factorial length as in the first logic.

In this section, we describe a third logic, which is a slight extension of the hybrid graph logic. We then use it to build a formula that expresses the Hamiltonian property. With this formula, we are able to reduce even further the complexity of testing whether a graph is Hamiltonian using a frame-checking method. This happens because we are able, for this formula, to reduce the frame-checking problem to a particular case of the model-checking problem with lower complexity than the general case.

6.1 Language

Definition 59. *The language of the* hybrid graph logic with the $\downarrow$ binder *is a hybrid language consisting of a set Φ of countably many proposition symbols (the elements of Φ are denoted by $p_1, p_2, \ldots$), a set $\mathcal{L}$ of countably many nominals (the elements of $\mathcal{L}$ are denoted by $i_1, i_2, \ldots$), a set $\mathcal{S}$ of countably many state-variables (the elements of $\mathcal{S}$ are denoted by $x_1, x_2, \ldots$), such that Φ, $\mathcal{L}$ and $\mathcal{S}$ are pairwise disjoint (the elements of $\Phi \cup \mathcal{L} \cup \mathcal{S}$ are called* atoms*), the boolean connectives $\neg$ and $\wedge$ and the modal operators $@_i$, for each nominal i, $@_x$, for each state-variable x, $\Diamond$, $\Diamond^+$ and $\downarrow$. The formulas are defined as follows:*

$$\varphi ::= p \mid i \mid x \mid \top \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \Diamond\varphi \mid \Diamond^+\varphi \mid @_i\varphi \mid @_x\varphi \mid \downarrow x.\varphi$$

Again, we freely use the standard abbreviations $\vee$, $\rightarrow$, $\leftrightarrow$, $\bot$, $\Box\varphi$, $\Box^+\varphi$, $\Diamond^*\varphi$ and $\Box^*\varphi$.

The definition of a *frame* and of a *model* are the same as the ones from section 4

In order to deal with the state-variables, we need to introduce the notion of *assignments*.

Definition 60. *An* assignment *is a function g that maps state-variables to vertices of the model, i.e., $g : \mathcal{S} \mapsto W$. We use the notation $g' = g[v_1/x_1, \ldots, v_n/x_n]$ to denote an assignment such that $g'(x) = g(x)$ if $x \notin \{x_1, \ldots, x_n\}$ and $g'(x_i) = v_i$, otherwise.*

The semantical notion of satisfaction is defined as follows:

Definition 61. *The notion of satisfaction of a formula ϕ in a model $\mathcal{M}$ at a vertex v with an assignment g, notation $\mathcal{M}, g, v \Vdash \phi$, is inductively defined adding the assignment g to all the clauses in definitions 4 and 35 and the following three extra clauses:*

1. *$\mathcal{M}, g, v \Vdash x$ iff $g(x) = v$;*
2. *$\mathcal{M}, g, v \Vdash @_x\phi$ iff $\mathcal{M}, g, d \Vdash \phi$, where $d = g(x)$;*

3. $\mathcal{M}, g, v \Vdash \downarrow x.\phi$ *iff* $\mathcal{M}, g[v/x], v \Vdash \phi$.

The formula $\downarrow x.\phi$ means that, using x as a name for the present vertex (state-variables can be thought as "on-the-fly nominals"), ϕ is satisfied. The $\downarrow$ operator is the only operator that binds a variable. Free and bound variables are defined in the usual way. The only case worth mentioning is that in the formula $@_x\psi$, x is free. A *sentence* is a formula with no free variables. We only consider formulas that are sentences, because we do not want to include the assignments in the model-checking and frame-checking problems.

The $\downarrow$ binder, just as the satisfaction operators, is dual to itself: $\mathcal{M}, g, v \Vdash \neg \downarrow x.\phi$ iff $\mathcal{M}, g, v \nVdash \downarrow x.\phi$ iff $\mathcal{M}, g[v/x], v \nVdash \phi$ iff $\mathcal{M}, g[v/x], v \Vdash \neg\phi$ iff $\mathcal{M}, g, v \Vdash \downarrow x.\neg\phi$.

Theorem 62 ([1])**.** *The satisfiability problem and the validity problem for the hybrid graph logic with the $\downarrow$ binder are* undecidable.

This result shows that the inclusion of state-variables and the $\downarrow$ operator turns a logic that had the same computational complexity of the basic graph logic into an undecidable logic.

Theorem 63 ([16])**.** *The model-checking problem for the hybrid graph logic with the $\downarrow$ binder is PSPACE-Complete both in the size of the model and in the length of the formula.*

In [24], it is shown that, for a family of hybrid logics with the $\downarrow$ binder, there are fragments of these logics in which the complexities of the satisfiability problem and the model-checking problem are lower than in the full logics. One of these fragments, which is defined using the notion of formulas in negation normal form, turns out to be also a fragment of the hybrid graph logic with the $\downarrow$ binder. According to [24], the complexity of the model-checking problem in this fragment is lower than the one stated above for the full hybrid graph logic with the $\downarrow$ binder. This result will be central to our discussion in this section.

Definition 64. *A formula of the hybrid graph logic with the $\downarrow$ binder is in* negation normal form *(NNF) if the negation symbol ($\neg$) appears only in front of proposition symbols, nominals and state-variables.*

Lemma 65. *If we consider the dual operators of $\top, \wedge, \Diamond$ and $\Diamond^+$, i.e., $\bot, \vee, \Box$ and $\Box^+$ as primitive operators of the language, then each formula of the hybrid graph logic with the $\downarrow$ binder is semantically equivalent to a formula in NNF.*

Proof. Let $(\triangle, \bigtriangledown)$ be one of the following pairs of dual operators: $(\wedge, \vee), (\Diamond, \Box), (\Diamond^+, \Box^+), (\downarrow, \downarrow)$ and $(@_z, @_z)$, where z is either a nominal or a state-variable. Using the semantic equivalences $\neg\top \equiv \bot$, $\neg\bot \equiv \top$, $\neg\neg\phi \equiv \phi$ and $\neg\triangle\phi \equiv \bigtriangledown\neg\phi$, we can push the negation symbols inward until they appear only in front of proposition symbols, nominals and state-variables. $\square$

Theorem 66 ([24])**.** *The model-checking problem for a formula in the hybrid graph logic with the $\downarrow$ binder that, when put in NNF, does not have any occurrence of $\Box$, $\Diamond^+$ or $\Box^+$ is NP-Complete both in the size of the model and in the length of the formula.*

The upper bound for the complexity of the frame-checking problem is again

$$FC = O(2^{|p|\times n} \times n^{|i|} \times MC), \tag{3}$$

where $|p|$ is the number of distinct proposition symbols that occur in the given formula ϕ, $|i|$ is the number of distinct nominals that occur in ϕ and n is the number of vertices in $\mathcal{F}$. It should be noticed that, if the model-checking problem can be solved in polynomial space, then the frame-checking problem can also be solved in polynomial space, since the frame-checking of a formula in a frame $\mathcal{F}$ is done through a finite series of completely independent model-checkings of that formula, one for each model $\mathcal{M}$ of $\mathcal{F}$.

Theorem 67. *The frame-checking problem for the hybrid graph logic with the $\downarrow$ binder is PSPACE in the length of the formula and in the size of the frame.*

Proof. This result follows directly from theorem 63 and the discussion above. □

6.2 The Hamiltonian Property

Let G be a graph with n vertices. We now use the language introduced in the previous subsection to define a formula that is valid if and only if G is Hamiltonian.

Theorem 68. *A connected graph G (with n vertices) with frame $\mathcal{F}$ is* Hamiltonian *if and only if $\mathcal{F} \Vdash \phi$, where ϕ is the formula*

$$\phi = \downarrow x_1.\Diamond \downarrow x_2.(\neg x_1 \wedge \Diamond \downarrow x_3.(\bigwedge_{1\le k<3} \neg x_k \wedge \Diamond \downarrow x_4.(\ldots \wedge \downarrow x_{n-1}.(\bigwedge_{1\le k<n-1} \neg x_k \wedge \Diamond \downarrow x_n.$$

$$(\bigwedge_{1\le k<n} \neg x_k \wedge \Diamond x_1)\ldots).$$

Proof. ($\Leftarrow$) Suppose that the formula ϕ is valid in $\mathcal{F}$. We will evaluate ϕ in an arbitrary vertex v_1 of a model with an arbitrary valuation $\mathbf{V}$ and an arbitrary assignment g. If $\mathcal{M}, g, v_1 \Vdash \phi$, then $\mathcal{M}, g[v_1/x_1], v_1 \Vdash \Diamond \downarrow x_2.\neg x_1 \ldots$. This means that there is a vertex v_2 such that $v_1 R v_2$ and $\mathcal{M}, g[v_1/x_1], v_2 \Vdash \downarrow x_2.\neg x_1 \wedge \Diamond \downarrow x_3 \ldots$, which means that $\mathcal{M}, g[v_1/x_1, v_2/x_2], v_2 \Vdash \neg x_1 \wedge \Diamond \downarrow x_3 \ldots$. This implies that $v_2 \neq v_1$ and $\mathcal{M}, g[v_1/x_1, v_2/x_2], v_2 \Vdash \Diamond \downarrow x_3 \ldots$. If we keep repeating this, we conclude that there are n distinct vertices $v_1, \ldots, v_n$ such that $v_i R v_{i+1}$, $1 \le i < n$ and $v_n R v_1$. We have a path that starts and ends in the vertex v_1 and visit every other vertex of G exactly once. This is exactly a Hamiltonian cycle.

($\Rightarrow$) Suppose that there is a Hamiltonian cycle $\langle v_1, \ldots v_n, v_1\rangle$ in G. We have that $\mathcal{M}, g[v_1/x_1, \ldots, v_n/x_n], v_n \Vdash \bigwedge_{1\le k<n} \neg x_k \wedge \Diamond x_1$. This means that $\mathcal{M}, g[v_1/x_1, \ldots, v_{n-1}/x_{n-1}], v_n \Vdash \downarrow x_n. \bigwedge_{1\le k<n} \neg x_k \wedge \Diamond x_1$, which implies that $\mathcal{M}, g[v_1/x_1, \ldots, v_{n-1}/ x_{n-1}], v_{n-1} \Vdash \bigwedge_{1\le k<n-1} \neg x_k \wedge \Diamond \downarrow x_n.(\bigwedge_{1\le k<n} \neg x_k \wedge \Diamond x_1)$. If we keep repeating this, we conclude that $\mathcal{M}, g, v_1 \Vdash \phi$, for an arbitrary assignment g. If we start the Hamiltonian cycle in another vertex, the same argument easily applies. Thus, ϕ is globally satisfied in $\mathcal{M}$. As the valuation in $\mathcal{M}$ is completely irrelevant, ϕ is valid in $\mathcal{F}$. □

Let us now determine how complex it is to test whether a graph is Hamiltonian using the above formula ϕ. First of all, we now have a formula that is a sentence with polynomial (quadratic) length in the size of the graph. Also, there are no proposition symbols and no nominals. This means that the valuation is completely irrelevant to the satisfaction of this sentence. Thus, the frame-checking problem is reduced to the model-checking problem for an arbitrary model of the frame. Let HAM be the complexity of testing whether a graph is Hamiltonian through a frame-checking of ϕ. Then, taking into account the above observations and the formula in equation (3), we have that

$$HAM = MC.$$

Now, we should notice that ϕ is already in NNF and it does not have any occurrence of $\Box$, $\Diamond^+$ or $\Box^+$. So, the reduction in the model-checking complexity stated in theorem 66 applies, and the model-checking complexity for this formula is in NP.

Theorem 69. *The complexity to check whether a graph is Hamiltonian using the above formula ϕ is NP in the size of the graph.*

Proof. This result follows directly from the discussion above. $\square$

The above formula ϕ is an "optimal" formula to describe the Hamiltonian property, in the following sense: since the problem of testing whether a graph is Hamiltonian is NP-Complete [18] and the test that we developed using this formula is in NP, it is impossible to find any other formula, in this logic or in any other logic, that describes the Hamiltonian property and can be frame-checked with a lower complexity than ϕ (assuming that NP $\neq$ P).

7 Edge-Related Properties

As was mentioned in the end of section 4, the best way to describe the Eulerian property would be to use theorem 31. However, as was also mentioned in the end of that section, our standard $\Diamond$ operators do not have the expressive power, at least not without performing a brute-force search through every possible path in the graph, in a similar way as the formula in theorem 47 does, to state cardinality conditions on edges incident from and to a vertex, as is needed in theorem 31. This remains true for the temporal operators defined in section 5. This happens because all of these operators are "existential" operators. All they can do is to differentiate between things like "there is some edge" and "there is no edge", or "there is some path" and "there is no path". We would need "counting" operators to be able to efficiently express theorem 31. Modal logics with this sort of operator exist in the literature and are called *graded modal logics*, being first introduced in [15].

We present a graded modal logic in the end of this section and use it to express the Eulerian property with the help of theorem 31. But first, we develop a method to express the Eulerian property in the hybrid logics already presented in the previous sections. This method could be useful to express other edge-related properties for which there is no theorem that states a necessary and sufficient condition for the property to hold, as does theorem 31.

7.1 Graph Subdivisions

If we do not use theorem 31, we need to define the Eulerian property with a formula that explicitly describes an Eulerian path in the graph. This is also a difficult task, because of the reasons exposed in the end of section 4. We need a way to identify particular edges, but hybrid logics only have names for vertices. So, we first develop a method to name edges *within* the formalism of a hybrid logic and later use it to define the Eulerian property.

Definition 70. *Let $\langle v, w\rangle$ be an edge in a graph G. An* edge subdivision *consists of adding a new vertex u to G, deleting the edge $\langle v, w\rangle$ and adding the edges $\langle v, u\rangle$ and $\langle u, w\rangle$ to G. A* graph subdivision *of a graph G is a graph G' obtained from G by a finite number of edge subdivisions.*

Definition 71. *Let G be a graph. We define $G' = \mathcal{E}(G)$ to be the graph obtained from G by subdividing every edge of G exactly once. We call G' an $\mathcal{E}$-graph.*

Thus, if G has n vertices and m edges, G' will have $m + n$ vertices. In fact, if we call W the set of vertices of G and W' the set of vertices of G', we have that $W' = W \cup W^*$ ($W \cap W^* = \emptyset$), where W^* is the set of new vertices added during the subdivision. We also have that every edge of G' has an extremity in W and the other in W^* and that there is a bijective map between elements of W^* and edges of G.

This bijective map between the set W^* and the edges of G is the key point in this construction. In the original graph G, we cannot identify particular edges using just an hybrid language. So, if we want to define a property in G, described using its edges, we build $G' = \mathcal{E}(G)$ and describe it in G', using the elements in W^*. These elements can be identified by standard nominals. This is what we do to express the Eulerian property.

For this method to work, we just have to pay attention to an important detail. In $\mathcal{E}$-graphs, it is fundamental to be able to distinguish whether a given vertex is in W or in W^*. Thus, instead of working with one set of nominals $\mathcal{L}$, we work with two such sets, $\mathcal{L}$ and $\mathcal{L}^*$ ($\mathcal{L} \cap \mathcal{L}^* = \emptyset$). Instead of writing $G' = (W', R')$, we write $G' = (W, W^*, R')$, to make clear the difference between the two sets of vertices, and define valuations $\mathbf{V}$ as $\mathbf{V}(p) \in \mathcal{P}(W \cup W^*)$, if p is a proposition symbol, $\mathbf{V}(i) = \{v\}$, such that $v \in W$, if $i \in \mathcal{L}$ and $\mathbf{V}(j) = \{w\}$, such that $w \in W^*$, if $j \in \mathcal{L}^*$. We will denote the nominals in $\mathcal{L}$ by $i_1, i_2, \ldots$ and the nominals in $\mathcal{L}^*$ by $j_1, j_2, \ldots$.

7.2 The Eulerian Property in the Hybrid Logics

Since we are going to define the Eulerian property with a formula that explicitly describes an Eulerian path in the graph, we can borrow ideas from three previously presented formulas: the formulas in theorems 47, 57 and 68. But the formula in the first theorem has factorial length, so we will only adapt the formulas in the second and third theorems to the Eulerian case.

This is not a difficult task. The formulas in theorem 57 and 68 state that, for a graph $G = (W, R)$, there is a cycle that visits every vertex in W exactly once. To define the Eulerian property, we need formulas that check that, for a graph G, there is a closed path such that every edge in G appears exactly once in it. This is equivalent to check, in $G' = \mathcal{E}(G)$, whether there is a closed path that visits every vertex in W^* exactly once.

First, we adapt the formula in theorem 57 to express the Eulerian property. Let $G' = \mathcal{E}(G)$ be a $\mathcal{E}$-graph with n vertices in W and m vertices in W^* and let $\mathcal{L}_m = \{j_1, \ldots, j_m\} \subset \mathcal{L}^*$. Let us add a loop to all its vertices in W^*. We can then define the formula that is valid if and only if G is Eulerian.

Theorem 72. *A connected graph G (with m edges) is* Eulerian *if and only if $\mathcal{F} \Vdash \phi$, where $\mathcal{F}$ is the frame that represents $G' = \mathcal{E}(G)$ and ϕ is the formula*

$$\phi = \psi_m \rightarrow \varepsilon_m,$$

where ψ_m is the formula from lemma 45 and

$$\varepsilon_m = @_{j_1}\mathbf{E}[\mathbf{XF}j_1 \wedge \mathbf{F}j_2 \wedge \ldots \wedge \mathbf{F}j_m \wedge$$

$$\wedge \mathbf{XG}(j_1 \rightarrow \mathbf{G}j_1) \wedge \mathbf{G}(j_2 \rightarrow \mathbf{XG}\neg j_2) \wedge \ldots \wedge \mathbf{G}(j_m \rightarrow \mathbf{XG}\neg j_m)].$$

Proof. The proof of the above theorem uses the same ideas that are presented in the proof of theorem 57. □

Now, we adapt the formula in theorem 68.

Theorem 73. *A connected graph G (with m edges) is* Eulerian *if and only if $\mathcal{F} \Vdash \phi$, where $\mathcal{F}$ is the frame that represents $G' = \mathcal{E}(G)$ and ϕ is the formula*

$$\phi = @_{j_1} \downarrow x_1.\Diamond\Diamond \downarrow x_2.(\neg x_1 \wedge \Diamond\Diamond \downarrow x_3.(\bigwedge_{1 \leq k < 3} \neg x_k \wedge \Diamond\Diamond \downarrow x_4.(\ldots \wedge \downarrow x_{m-1}.$$

$$(\bigwedge_{1 \leq k < m-1} \neg x_k \wedge \Diamond\Diamond \downarrow x_m.(\bigwedge_{1 \leq k < m} \neg x_k \wedge \Diamond\Diamond x_1)\ldots).$$

Proof. Here, we use a satisfaction operator at the beginning of the formula because we are now dealing with two sets of vertices: W and W^*. We need the satisfaction operator to guarantee that we are starting the path in a vertex belonging to W^*. The double occurrences of the $\Diamond$ operator ($\Diamond\Diamond$) are designed to discard the vertices belonging to W and analyze only the behaviour of the vertices belonging to W^*. The rest of the proof of the above theorem uses the same ideas that are presented in the proof of theorem 68. □

As the formulas in theorems 72 and 73 were adapted from the ones in theorems 57 and 68, the complexity results presented in theorems 58 and 69, respectively, also apply to them.

7.3 The Eulerian Property in a Graded Modal Logic

In this subsection, we define a graded modal logic that is appropriate for our needs and use it to express the Eulerian property. We call it *graded graph logic.*

Definition 74. *The language of the* graded graph logic *is a modal language consisting of a set Φ of countably many proposition symbols (the elements of Φ are denoted by $p_1, p_2, \ldots$), the boolean connectives $\neg$ and $\wedge$ and the modal operators $\Diamond_i$ and $\Diamond_i^{-1}$, where $i \in \mathbb{N}$. The formulas are defined as follows:*

$$\varphi ::= p \mid \top \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \Diamond_i\varphi \mid \Diamond_i^{-1}\varphi$$

Again, we freely use the standard boolean abbreviations $\vee$, $\rightarrow$, $\leftrightarrow$ and $\perp$ and also the following abbreviations for the duals: $\Box_i\varphi := \neg\Diamond_i\neg\varphi$ and $\Box_i^{-1}\varphi = \neg\Diamond_i^{-1}\neg\varphi$. Also, in order to make the language more elegant, we introduce the following abbreviations: $\blacklozenge_i\varphi = \Diamond_i\varphi \wedge \neg\Diamond_{i+1}\varphi$ and $\blacklozenge_i^{-1}\varphi = \Diamond_i^{-1}\varphi \wedge \neg\Diamond_{i+1}^{-1}\varphi$.

The definition of a *frame* and of a *model* are the same as the ones from section 2. The semantical notion of satisfaction is defined as follows:

Definition 75. *The notion of satisfaction of a formula ϕ in a model $\mathcal{M}$ at a vertex v, notation $\mathcal{M}, v \Vdash \phi$, is inductively defined as in definition 4, with the following modifications:*

1. *$\mathcal{M}, v \Vdash \Diamond_i\phi$ iff $\#\mathcal{R}(v,\phi) \geq i$;*
2. *$\mathcal{M}, v \Vdash \Diamond_i^{-1}\phi$ iff $\#\mathcal{R}^{-1}(v,\phi) \geq i$,*

where $\mathcal{R}(v,\phi) = \{w : vRw \text{ and } \mathcal{M}, w \Vdash \phi\}$, $\mathcal{R}^{-1}(v,\phi) = \{w : wRv \text{ and } \mathcal{M}, w \Vdash \phi\}$ and $\#S$ denotes the cardinality of the set S.

A formula $\Diamond_i\varphi$ is satisfied at a vertex v if there are *at least* i distinct vertices v_k, $1 \leq k \leq i$, such that vRv_k and φ is satisfied at v_k. A formula $\Diamond_i^{-1}\varphi$ is satisfied at a vertex v if there are *at least* i distinct vertices v_k, $1 \leq k \leq i$, such that v_kRv and φ is satisfied at v_k. A formula $\blacklozenge_i\varphi$ is satisfied at a vertex v if there are *exactly* i distinct vertices v_k, $1 \leq k \leq i$, such that vRv_k and φ is satisfied at v_k. A formula $\blacklozenge_i^{-1}\varphi$ is satisfied at a vertex v if there are *exactly* i distinct vertices v_k, $1 \leq k \leq i$, such that v_kRv and φ is satisfied at v_k.

Definition 76. *We may define the length of a formula ϕ in the graded graph logic in two different ways. We call them the* standard length*, denoted as $|\phi|$, and the* ungraded length*, denoted as $||\phi||$. We define both of these lengths in the same way as the length in definition 10, except for formulas of the form $\phi = \Diamond_i\psi$ or $\phi = \Diamond_i^{-1}\psi$, where the standard length is defined as $|\phi| = i + |\psi|$ and the ungraded length is defined as $||\phi|| = 1 + ||\psi||$.*

Theorem 77 ([25])**.** *The satisfiability problem and the validity problem for the graded graph logic are PSPACE-Complete in the standard length of the formula.*

Theorem 78. *The model-checking problem for the graded graph logic is PTIME (linear) in the product of the size of the model and the ungraded length of the formula.*

Proof. The graded graph logic without the converse modalities ($\Diamond_k^{-1}$) is a fragment of Graded-CTL, a logic presented in [14]. Thus, the result follows from the complexity of the model-checking problem for Graded-CTL, which is also presented in [14], and the fact that the presence of converse modalities does not increase the model-checking complexity, as shown in [16]. $\Box$

The upper bound for the complexity of the frame-checking problem is the same as in section 2:

$$FC = O(2^{|p| \times n} \times MC), \tag{4}$$

where $|p|$ is the number of distinct proposition symbols that occur in the given formula ϕ and n is the number of vertices in $\mathcal{F}$.

Theorem 79. *The frame-checking problem for the graded graph logic is PTIME (linear) in the ungraded length of the formula and EXPTIME in the size of the frame and in the number of distinct proposition symbols that occur in the formula.*

Proof. This result follows directly from the discussion above. □

Let G be a graph with n vertices. We now use the language introduced above to define a formula that is valid if and only if G is Eulerian.

Theorem 80. *A connected graph G (with n vertices) with frame $\mathcal{F}$ is* Eulerian *if and only if $\mathcal{F} \Vdash \phi$, where ϕ is the formula*

$$\phi = \bigvee_{0 \leq k \leq n} (\blacklozenge_k \top \wedge \blacklozenge_k^{-1} \top)$$

Proof. First, it is easy to see that $\blacklozenge_{i_v} \top$ is satisfied at a vertex v if and only if the out-degree of v is i_v and $\blacklozenge_{j_v}^{-1} \top$ is satisfied at a vertex v if and only if the in-degree of v is j_v. Then, by theorem 31, G is Eulerian if and only if the out-degree of every vertex v in G is equal to its in-degree, which is true if and only if there is k_v for each vertex v, $0 \leq k_v \leq n$ such that both the out-degree and the in-degree of v are equal to k_v. This is true if and only if $\blacklozenge_{k_v} \top \wedge \blacklozenge_{k_v}^{-1} \top$ is satisfied in v, which is true if and only if ϕ is satisfied in every vertex v. □

Let us now determine how complex it is to test whether a graph is Eulerian using the above formula ϕ. First of all, we have a formula with linear ungraded length in the size of the graph. Also, there are no proposition symbols, which means that the valuation is completely irrelevant to the satisfaction of this formula. Let EUL be the complexity of testing whether a graph is Eulerian through a frame-checking of ϕ. Then, taking into account the above observation and the formula in equation (4), we have that

$$EUL = MC,$$

where, for the formula ϕ, MC is PTIME (in fact quadratic) in the size of the graph.

Theorem 81. *The complexity to check whether a graph is Eulerian using the above formula ϕ is PTIME (quadratic) in the size of the graph.*

Proof. This result follows directly from the discussion above. □

8 Conclusions

Our goal in this paper is to express, using modal logics, some global graph properties that are central to many computer science applications. The work presented in [7] is closely related to this one. In that work, the interest was also in how to use modal logics to express global graph properties. However, in [7], only the basic graph logic was considered and only connectivity and acyclicity were analyzed. Moreover, the focus of that work was on how to build axiomatizations for classes of graphs with these global properties, while our focus is on how to find formulas expressing each of these properties that can be efficiently used to test whether a graph satisfies them.

The use of model-checking algorithms to verify properties encoded as logical formulas is each day getting more attention and more applications. This happens because many data structures in Computer Science can be encoded as graphs. Our use of model-checking to verify global graph properties is just one more application in a wide and diverse field from which we mention a few other interesting and recent examples. In [23], a fragment of first-order logic is used to express preconditions for graph transformations in a system of graph rewriting. In order to verify if a given graph satisfies the precondition, the system uses a model-checking algorithm. In [16], model-checking algorithms for hybrid logics are used to perform queries in XML files. In [19], the verification of business workflows is conducted using a model-checking algorithm for temporal logics. In [4], an automated model-checking technique is used in the verification of a security protocol. In [5] and [17], an extension of the temporal logic CTL, together with an appropriate model-checking algorithm, are used in the automated analysis of quantum information protocols.

In this work, we present various formalisms, from a very basic modal logic to a very powerful temporal logic and use them to express and test four graph properties: connectivity, acyclicity and the Hamiltonian and Eulerian properties. It would also be interesting to continue this line of work and try to express some other graph properties such as planarity and k-colorability of vertices and edges.

This work is an interesting way of exposing an important issue. Sometimes, standard modal logics, even the ones that are incredibly expressive, such as the μ-calculus, are not capable of expressing some important properties. This happens because of some strong invariance conditions (such as the ones defined in section 3) that these logics satisfy. In these cases, the use of a hybrid logic is a very simple way to bypass this problem. Hybrid logics have much weaker invariance conditions [3], which increases the number of definable properties.

In section 6, we are able to find a formula (theorem 68) in a hybrid logic with the $\downarrow$ operator that expresses the Hamiltonian property and can be checked in NP time. This is an optimal result, since the problem of deciding whether a graph is Hamiltonian is NP-Complete [18].

Theorem 68 also implies that we can polynomially reduce the problem of deciding whether a graph is Hamiltonian, which is NP-Complete, to the problem of model-checking a formula of the hybrid graph logic with the $\downarrow$ operator that contain no $\Box$, $\Diamond^+$ or $\Box^+$. This is an alternative proof of the NP-hardness of this model-checking problem stated in theorem 66. The original proof for the NP-hardness in theorem 66, presented in [24], uses a different polynomial reduction, starting not from the problem of deciding whether a graph is Hamiltonian, but from the problem of deciding whether a propositional logic formula is satisfiable, which is also a NP-Complete problem [18].

In fact, the formula that expresses the Hamiltonian property and the formula presented in [24] to express the propositional satisfiability problem are remarkably similar, consisting of an alternating sequence of $\downarrow$'s and $\Diamond$'s. This suggests that the fragment presented in [24] could be a simple framework for the description of NP-Complete problems. Expressing these problems in a common language could highlight the underlying similarities between them. As graph theory has a rich collection of NP-Complete problems, it would also be interesting to analyze how we can express them in this fragment.

In section 7, we use a graded modal logic to express the Eulerian property.

The resulting formula can be checked with very good efficiency. In fact, graded modal logics seem to be a very good choice for the efficient verification of graph properties that involve numerical conditions.

Finally, also in section 7, we describe a way to name edges in a hybrid logic using graph subdivisions. One open issue with this method is that some formulas satisfied at a vertex v in G are no longer satisfied in the same vertex in $\mathcal{E}(G)$. For instance, a formula $\Diamond\varphi$ may be satisfied at v in G but not in G', because of the new vertices that are added between the old ones. If we just want to evaluate formulas in G' and forget about G, as we did in section 7, then this is not a problem. But if we want to work with both graphs at the same time, then it would be very interesting to define a translation $\mathcal{T}$ between formulas, such that if φ is satisfied at v in G, then $\mathcal{T}(\varphi)$ is satisfied at v in G'. It would also be interesting to study what other properties could be expressed with hybrid logics using this construction that allows us to name not only vertices but also edges.

Acknowledgements

The first author was partially supported by a grant from CNPq. The second author was supported by a D.Sc. scholarship from CNPq.

References

[1] C. Areces, P. Blackburn, and M. Marx. A road-map on complexity for hybrid logics. In *8th Annual Conference of the EACSL*, volume 1683 of *LNCS*, pages 307–321. Springer, 1999.

[2] C. Areces, P. Blackburn, and M. Marx. The computational complexity of hybrid temporal logics. *Logic Journal of the IGPL*, 8:653–679, 2000.

[3] C. Areces and B. ten Cate. Hybrid logics. In P. Blackburn, J. van Benthem, and F. Wolter, editors, *Handbook of Modal Logic*, pages 821–868. Elsevier, 2006.

[4] A. Armando, R. Carbone, and L. Compagna. LTL model checking for security protocols. In *20th IEEE Computer Security Foundations Symposium*, pages 385–396. IEEE, 2007.

[5] P. Baltazar, R. Chadha, and P. Mateus. Quantum computation tree logic – model checking and complete calculus. *International Journal of Quantum Information*, 6(2):219–236, 2008.

[6] V. C. Barbosa. *An Introduction to Distributed Algorithms.* MIT Press, 1996.

[7] M. R. F. Benevides. Modal logics for finite graphs. In R. Queiroz, editor, *Logic for Synchronization and Concurrency*, Trends in Logic, pages 239–267. Kluwer Academic Publisher, 2003.

[8] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic.* Theoretical Tracts in Computer Science. Cambridge University Press, 2001.

[9] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications.* Elsevier, New York, 1979.

[10] J. Bradfield and C. Stirling. Modal mu calculi. In P. Blackburn, J. van Benthem, and F. Wolter, editors, *Handbook of Modal Logic*, pages 721–756. Elsevier, 2006.

[11] E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking.* MIT Press, 2000.

[12] M. Dam. CTL* and ECTL* as fragments of the modal mu-calculus. *Theoretical Computer Science*, 126:77–96, 1994.

[13] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pages 995–1072. North-Holland Pub. Co./MIT Press, 1990.

[14] A. Ferrante, M. Napoli, and M. Parente. CTL model-checking with graded quantifiers. In *6th International Symposium on Automated Technology for Verification and Analysis*, volume 5311 of *LNCS*, pages 18–32. Springer, 2008.

[15] K. Fine. In so many possible worlds. *Notre Dame Journal of Formal Logic*, 13(4):516–520, 1972.

[16] M. Franceschet and M. de Rijke. Model checking hybrid logics (with an application to semistructured data). *Journal of Applied Logic*, 4(3):279–304, 2006.

[17] S. J. Gay, R. Nagarajan, and N. Papanikolaou. Model-checking quantum protocols. http://www.
dcs.warwick.ac.uk/∼nikos/downloads/gnp.pdf.

[18] R. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Plenum, New York, 1972.

[19] M. Kovács, L. Gönczy, and D. Varró. Formal analysis of BPEL workflows with compensation by model checking. *International Journal of Computer Systems and Engineering*, to appear.

[20] N. Lynch. *Distributed Algorithms.* Morgan Kaufmann Publishers, San Mateo, 1996.

[21] F. Moller and A. Rabinovich. On the expressive power of CTL*. In *XIV IEEE Symposium on Logic in Computer Science*, pages 360–369, 1999.

[22] U. Sattler and M. Y. Vardi. The hybrid μ-calculus. In *First International Joint Conference in Automated Reasoning*, volume 2083 of *LNAI*, pages 76–91. Springer, 2001.

[23] M. Strecker. Modeling and verifying graph transformations in proof assistants. In *4th International Workshop on Computing with Terms and Graphs*, volume 203 of *Electronic Notes in Theoretical Computer Science*, pages 135–148. Elsevier, 2008.

[24] B. ten Cate and M. Franceschet. On the complexity of hybrid logics with binders. In *19th International Workshop of Computer Science Logic*, volume 3634 of *LNCS*, pages 339–354. Springer, 2005.

[25] W. van der Hoek and M. de Rijke. Counting objects. *Journal of Logic and Computation*, 5(3):325–345, 1995.

[26] Y. Venema. Lectures on the modal mu-calculus. http://staff.science.uva.nl/∼yde/teaching/ml/mu/mu.pdf.