Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 247 (2009) 123–138
www.elsevier.com/locate/entcs

A Logical Approach to Hamiltonian Graphs

L. Menasché Schechter[1,2]

Systems and Computer Engineering Program
Federal University of Rio de Janeiro
Brazil

Abstract

Graphs are among the most frequently used structures in computer science. A lot of problems can be modelled using a graph and can then be solved by checking whether the graph satisfies some property. In this work, we are interested in how to use logical frameworks as a generic tool to express and efficiently check graph properties. In order to reason about this, we choose to analyze the Hamiltonian property and choose the family of modal logics as our framework. Our analysis has to deal with two central issues: whether each of the modal languages under consideration has enough expressive power to describe this property and how complex (computationally) it is to use these logics to actually test whether a given graph has this property. First, we show that this property is 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 it in a basic hybrid logic. Unfortunately, the Hamiltonian property still cannot be efficiently checked in this logic. In a second attempt, we extend this basic hybrid logic with the $\downarrow$ operator and show that we can check the Hamiltonian property with optimal (NP-Complete) complexity in this logic.

Keywords: Hamiltonian Graphs, Modal Logics, Model-Checking, Frame-Checking, Computational Complexity

1 Introduction

Graphs are among the most frequently used structures in computer science [8]. In this discipline, many important problems admit a graph representation and the solution of the original problem is often reduced to checking whether the graph satisfies some property. As an example, in the field of distributed systems, graphs are used to describe and to solve problems such as resource sharing problems, scheduling problems and deadlock issues, among others [4,14].

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

[1] The author was supported by a D.Sc. scholarship from CNPq.
[2] Email: luis@cos.ufrj.br

doi:10.1016/j.entcs.2009.07.052

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:

(i) Can we encode a graph as a structure for $\mathbb{L}$?

(ii) Can we encode the graph properties that we want to verify as $\mathbb{L}$-formulas?

(iii) 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:

(iv) Is the logical method as efficient 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 [7]. 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 [7] states, "modal languages are simple yet expressive languages for talking about relational structures".

Many important graph properties are what we can call *global graph properties*, which means that they are properties that hold for the graph as a whole and that depend on the structure of the whole graph. In order to reason about the issues stated above, we choose one of these properties, the Hamiltonian property, to guide our exposition throughout this work.

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

This work can be considered as a follow-up to [6]. In that work, the goal was also to find formulas that could describe global graph properties, but the practical issue of how computationally complex it would be to use those formulas to check whether a graph satisfies the correspondent property was not addressed.

A *finite directed graph* (from now on called simply a *graph*) G is a pair (V, R), where V is a finite set of vertices and $R \subseteq V \times V$ is a set of ordered pairs of vertices (a binary relation on V), 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 $\langle v_1, v_2, \ldots, v_n \rangle$, where $\langle v_i, v_{i+1} \rangle \in R$, for $0 < i < n$. 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*.

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 the Hamiltonian property is definable or not in the language presented in the previous section. In Section 4, we extend the modal logic of the previous sections with nominals, obtaining a hybrid modal logic, and use it to express the Hamiltonian property. In Section 5, 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-Complete) complexity in this logic, thus answering positively the fourth question above. Finally, in Section 6, we draw our concluding remarks.

2 Basic Graph Logic

In this section, we define a modal logic [3] with two modal operators: $\Diamond$ and $\Diamond^+$. We call it *basic graph logic*.

Definition 2.1 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 2.2 A *frame* for the basic graph logic is a pair $\mathcal{F} = (V, R)$, where V is a set (finite or not) of vertices and R is a binary relation over V, i.e., $R \subseteq V \times V$.

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 2.3 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 V, i.e., $\mathbf{V} : \Phi \mapsto \mathcal{P}(V)$.

[3] For a broad reference on modal logics, [7] can be consulted.

The semantical notion of satisfaction is defined as follows:

Definition 2.4 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:

(i) $\mathcal{M}, v \Vdash p$ iff $v \in \mathbf{V}(p)$;

(ii) $\mathcal{M}, v \Vdash \top$ always;

(iii) $\mathcal{M}, v \Vdash \neg\varphi$ iff $\mathcal{M}, v \nVdash \varphi$;

(iv) $\mathcal{M}, v \Vdash \varphi_1 \wedge \varphi_2$ iff $\mathcal{M}, v \Vdash \varphi_1$ and $\mathcal{M}, v \Vdash \varphi_2$;

(v) $\mathcal{M}, v \Vdash \Diamond\varphi$ iff there is a $w \in V$ such that vRw and $\mathcal{M}, w \Vdash \varphi$;

(vi) $\mathcal{M}, v \Vdash \Diamond^+\varphi$ iff there is a $w \in V$ such that vR^+w and $\mathcal{M}, w \Vdash \varphi$.

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

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 $\mathcal{M}, w \Vdash \Diamond^+\Box\bot$.

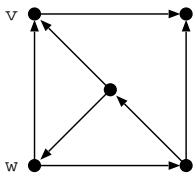

Fig. 1. Model $\mathcal{M}$, where a formula φ is satisfied at vertex v.

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, for each logic that we consider, we want to find a modal formula ϕ, such that a graph G has the Hamiltonian 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 property 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 estimate 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 Hamiltonian 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 graph properties involves four basic decision problems.

Definition 2.5 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 2.6 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, as a formula ϕ is valid if and only if the formula $\neg\phi$ is not satisfiable.

Definition 2.7 The *model-checking problem* consists of, given a formula ϕ, a model $\mathcal{M}$ and a vertex v in $\mathcal{M}$, determining whether $\mathcal{M}, v \Vdash \phi$.

Definition 2.8 The *frame-checking problem* consists of, given a formula ϕ and a frame $\mathcal{F}$, determining whether $\mathcal{F} \Vdash \phi$.

Theorem 2.9 ([7]) *The satisfiability problem and the validity problem for the basic graph logic are EXPTIME-Complete in the length of the formula.*

Definition 2.10 Let $\mathcal{M} = (V, R, \mathbf{V})$ be a model. Let $|V|$ be the number of vertices in V and $|R|$ the number of pairs in R. We define the size of the model (or the frame, or the graph) as $|V| + |R|$.

Theorem 2.11 ([10]) *The model-checking problem for the basic graph logic is PTIME both in the size of the model and in the length of the formula.*

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. Let FC be the complexity of the frame-checking problem and MC be the complexity of the model-checking problem. Then,

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

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

Theorem 2.12 *The frame-checking problem for the basic graph logic is PTIME 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.*

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 Modal Definability

In this section, we investigate whether the Hamiltonian property is definable or not in the language of the basic graph logic.

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 modal formula. 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 one of these "similarity" results (more details about it and other related results may be found in [7]), and then we prove a theorem for the Hamiltonian property using it.

Definition 3.1 Let $\mathcal{M} = (W, R, \mathbf{V})$ and $\mathcal{M}' = (W', R', \mathbf{V}')$ be two models. A function $f : W \to W'$ is a *bounded morphism* if it satisfies the following conditions:

(i) w and $f(w)$ satisfy the same proposition symbols;

(ii) f is a homomorphism with respect to R (if wRv, then $f(w)R'f(v)$);

(iii) 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 W to W', then we say that $\mathcal{M}'$ is a *bounded morphic image* of $\mathcal{M}$ and use the notation $\mathcal{M} \Rightarrow \mathcal{M}'$.

A similar definition can be given for a bounded morphism of frames, just removing the part of the above definition that deals with valuations.

Below is a basic theorem about modal definability that is going to be used in the next subsection. Its proof for a language that contains only $\Diamond$ can be found in [7]. It is not difficult to extend that proof to a language that contains both $\Diamond$ and $\Diamond^+$.

Theorem 3.2 *Let $\mathcal{M} = (W, R, \mathbf{V})$ and $\mathcal{M}' = (W', R', \mathbf{V}')$ be two models such that $\mathcal{M} \Rightarrow \mathcal{M}'$. Then, $\mathcal{M}, w \Vdash \phi$ if and only if $\mathcal{M}', f(w) \Vdash \phi$.*

Corollary 3.3 *Let $\mathcal{F} = (W, R)$ and $\mathcal{F}' = (W', R')$ be two frames such that $\mathcal{F} \Rightarrow \mathcal{F}'$. If $\mathcal{F} \Vdash \phi$, then $\mathcal{F}' \Vdash \phi$.*

3.1 Hamiltonian Graphs

Definition 3.4 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 3.5 *The class of Hamiltonian graphs is not modally definable.*

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 bounded morphism. By Corollary 3.3, since the Hamiltonian property is not preserved under bounded morphic images, it is not modally definable. □

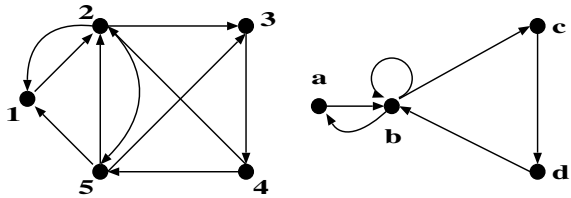

Fig. 2. Graph 1,2,3,4,5 is Hamiltonian and graph a,b,c,d is not.

3.2 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 property that we want. We need a stronger language. One idea could be to use the modal μ-calculus [9,16]. 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* [11].

Unfortunately, even with all this expressive power, the language of the μ-calculus fails to express this property because of the same reason exposed in the previous subsection. This happens because μ-calculus formulas, as the basic graph formulas, are invariant under bisimulations (bounded morphisms are a special case of bisimulation). In fact, the μ-calculus is the bisimulation-invariant fragment of Monadic Second-Order Logic (MSOL) [9].

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 Hamiltonian property.

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 property 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 2.5 until 2.8.

One interesting class of logics to take into consideration is the class of *hybrid logics* [3,7]. 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 $\mathcal{P}(V)$, 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 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 Hamiltonian property.

4.1 Language

Definition 4.1 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 $@_i$, 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 4.2 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 V, i.e., $\mathbf{V} : \Phi \mapsto \mathcal{P}(V)$, and mapping nominals into singleton subsets of V, i.e, if i is a nominal then $\mathbf{V}(i) = \{v\}$ for some $v \in V$. 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 4.3 The notion of satisfaction is defined adding two extra clauses to Definition 2.4:

(i) $\mathcal{M}, v \Vdash i$ iff $v \in \mathbf{V}(i)$;

(ii) $\mathcal{M}, v \Vdash @_i\varphi$ iff $\mathcal{M}, d \Vdash \varphi$, where d is the denotation of i under $\mathbf{V}$.

It is interesting to see that the operators $@_i$ are duals to themselves: $\mathcal{M}, v \Vdash \neg@_i\phi$ iff $\mathcal{M}, v \nVdash @_i\phi$ iff $\mathcal{M}, d \nVdash \phi$, where d is the denotation of i under the valuation of $\mathcal{M}$, iff $\mathcal{M}, d \Vdash \neg\phi$ iff $\mathcal{M}, v \Vdash @_i\neg\phi$.

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 4.4 ([2]) *The satisfiability problem and the validity problem for the hybrid graph logic are EXPTIME-Complete in the length of the formula.*

Theorem 4.5 ([12]) *The model-checking problem for the hybrid graph logic 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 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

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

where $|p|$ is the number of distinct proposition symbols that occur on 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 pair $(\mathcal{M}, v)$ based on the given frame $\mathcal{F}$. Every proposition symbol that appears in ϕ may receive 2^n possible valuations, while every nominal may only receive n possible valuations.

Theorem 4.6 *The frame-checking problem for the hybrid graph logic is PTIME 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.*

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

4.2 Hybrid Definability

Before trying to find a formula to describe the Hamiltonian graphs, we need to consider some graph-theoretical issues. In graph theory [8], 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 formulas do is to inspect all of the paths in the graph, searching for a Hamiltonian cycle. Not surprisingly then, a formula for a graph with n vertices uses n nominals.

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 4.7 *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*

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

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

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 4.8 *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*

$$\phi = \psi_n \rightarrow \delta_n,$$

with

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

Proof. ($\Leftarrow$) Suppose that the formula ϕ is valid in $\mathcal{F}$. This means that, for any arbitrary vertex v and any arbitrary valuation $\mathbf{V}$, $(\mathcal{F}, \mathbf{V}), v \Vdash \phi$. In particular,

$(\mathcal{F}, \mathbf{V}^*), v^* \Vdash \phi$, where $\mathbf{V}^*$ satisfies ψ_n and $\mathbf{V}^*(i_1) = \{v^*\}$. First, this means that, under this valuation, each nominal is denoting a different vertex. Second, $\mathbf{V}^*$ must also satisfy δ_n at v^*. 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 $\sigma'(k) = i_k$. Let then $\Delta_n = i_n \wedge \Diamond i_1$. We also define the formulas Δ_k, for $1 \leq k \leq n-1$, as

$$\Delta_k = i_k \wedge \Diamond \Delta_{k+1}.$$

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, $2 \leq k \leq n$, in G such that $(\mathcal{F}, \mathbf{V}^*), w_k \Vdash \Delta_k$, with $w_k R w_{k+1}$, for $2 \leq k \leq n-1$, $v^* R w_2$ 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 $\mathcal{F}$. □

This is the simplest and most direct attempt to describe the Hamiltonian property: pure brute force. The formula has factorial length on the size of the graph, which makes it impossible to be frame-checked. However, the length of the formula is not the only problem. In order to see that, we present an alternative formula that has polynomial length on the size of the graph.

Theorem 4.9 *A connected graph G (with n vertices) with frame $\mathcal{F}$ is* not Hamiltonian *if and only if $\mathcal{F} \Vdash \neg\phi$, where ϕ is the formula*

$$\phi = \bigwedge_{1 \leq k < n} @_{i_k} \Diamond i_{k+1} \wedge @_{i_n} \Diamond i_1 \wedge \psi_n.$$

Proof. It is not difficult to show, using a proof similar to the one in the previous theorem, that G is Hamiltonian if and only if there is a valuation $\mathbf{V}^*$ such that $(\mathcal{F}, \mathbf{V}^*) \Vdash \phi$. □

Now, the formula has polynomial length on the size of the graph and the frame-checking complexity is PTIME in the length of the formula (Theorem 4.6), but we still cannot perform an efficient frame-check. The reason for this is that not only the length of the formula is linked to the size of the graph, but so is the number of distinct nominals in the formula. If we look at the complexity of the frame-checking

problem in the hybrid graph logic (Theorem 4.6 and the formula above it), this means that $FC = O(n^n * MC)$. Even if we consider only the valuations that satisfy ψ_n, i.e., valuations that assign n distinct vertices to the n distinct nominals in the formula, we still get $FC = O(n! * MC)$. Even though the length of the formula is now polynomial, we still have a factorial time complexity to check the Hamiltonian property in the hybrid graph logic. Hence, we must search for an alternative form to express this property using another logic. This is what we do in the next section.

5 Hybrid Graph Logic with the ↓ Binder

In the previous section, we were only able to test whether a graph is Hamiltonian using a frame-checking method in factorial time. In this section, we describe a third logic, which is an extension of the hybrid graph logic with state-variables and the ↓ binder [4]. We then use it to build a formula that expresses the Hamiltonian property. With this formula, we are able to significantly reduce the complexity of testing whether a graph is Hamiltonian using a frame-checking method. This happens because we are able, for this particular formula, to reduce the frame-checking problem to a model-checking problem.

5.1 Language

Definition 5.1 The language of the *hybrid graph logic with the ↓ 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 5.2 An *assignment* is a function g that maps state-variables to vertices of the model, i.e., $g : \mathcal{S} \mapsto V$. 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:

[4] For more information on hybrid logics with the ↓ binder, [3] and [15] can be consulted.

Definition 5.3 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 2.4 and 4.3 and the following three extra clauses:

(i) $\mathcal{M}, g, v \Vdash x$ iff $g(x) = v$;

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

(iii) $\mathcal{M}, g, v \Vdash \downarrow x.\phi$ 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 of as "on-the-fly nominals"), ϕ is satisfied. The $\downarrow$ operator is the only operator that binds a variable. Free and bounded variables are defined in the usual way. The only case worth mentioning is that in the formula $@_x\psi$, the variable x occurring in the satisfaction operator 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 5.4 ([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 complexity as the basic graph logic into an undecidable logic.

Theorem 5.5 ([12]) *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 [15], 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 [15], 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 5.6 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 5.7 *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 $(\triangle, \triangledown)$ 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 $\neg \triangle \phi \equiv \triangledown\neg\phi$, we can push the negation symbols inside the formulas until they appear only in front of proposition symbols, nominals and state-variables. □

Theorem 5.8 ([15]) *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 $\Box$, $\Diamond^+$ and $\Box^+$ 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

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

where $|p|$ is the number of distinct proposition symbols that occur on 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. This happens because the frame-checking is done through a series of completely independent model-checkings, which means that the same amount of memory space used to perform a single model-checking can be reused multiple times to perform a frame-checking.

Theorem 5.9 *The frame-checking problem for the hybrid graph logic is PSPACE in the length of the formula and in the size of the frame.*

5.2 The Hamiltonian Property

Theorem 5.10 *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*

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

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

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 $v_i R v_{i+1}$, $1 \leq i < n$ 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 $\mathcal{M}, g[v_1/x_1, \ldots, v_n/x_n], v_n \Vdash \bigwedge_{1\leq k<n} \neg x_k \wedge \Diamond x_1$. This means that $\mathcal{M}, g[v_1/x_1, \ldots, v_{n-1}/x_{n-1}], v_n \Vdash\downarrow x_n.\bigwedge_{1\leq k<n} \neg x_k \wedge \Diamond x_1$, which implies that $\mathcal{M}, g[v_1/x_1, \ldots, v_{n-1}/x_{n-1}], v_{n-1} \Vdash \bigwedge_{1\leq k<n-1} \neg x_k \wedge \Diamond \downarrow x_n.(\bigwedge_{1\leq k<n} \neg x_k \wedge \Diamond x_1)$. 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 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. From the fact that an Hamiltonian cycle goes through every vertex of the graph, it is not difficult to see that the formula is satisfied in one vertex of the model if and only if it is satisfied in all vertices of the model. Thus, the frame-checking problem is reduced to the model-checking problem for an arbitrary vertex of 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, 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 5.8 applies, and the model-checking complexity for this formula is NP-Complete.

Theorem 5.11 *The complexity to check whether a graph is Hamiltonian using the above formula ϕ is NP-Complete in the size of the graph.*

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 [13] and the test that we developed using this formula is also NP-Complete, it is impossible to find any other formula, in this logic or in any other logic, that describes the Hamiltonian property and can be tested faster than ϕ (assuming that NP $\neq$ P).

6 Conclusions

Our goal in this paper is to try to express and efficiently check, using modal logics, a graph property that is central in computer science: the Hamiltonian property. The works presented in [5] and [6] are closely related to this one. In [5], the interest was also in how to use modal logics to express global graph properties. However, in that work, only the basic graph logic was considered and the properties analyzed were connectivity and acyclicity. 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 finding formulas expressing a global graph property that can be efficiently used to test whether a graph satisfies it. In [6], the goal was also to find formulas that could describe global graph properties, but the practical issue of how computationally complex it would be to use those formulas to check whether a graph satisfies the correspondent property was not addressed.

In this work, we present three increasingly expressive formalisms, from a very basic modal logic to a hybrid logic with variables and use them to express and test the Hamiltonian property. It would also be interesting to continue this line of work and try to express some other global 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 that these logics satisfy (for example, the modal logic defined in Section 2 can only express properties that are invariant under bounded-morphisms, as shown in Section 3). 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 5, we are able to find a formula in a hybrid logic with the $\downarrow$ operator that expresses the Hamiltonian property and can be checked in NP-Complete time. This is an optimal result, since the problem of deciding whether a graph is Hamiltonian is NP-Complete [13]. Also, if we think about the results presented in Section 5 in the reverse order, the fact that we can express the Hamiltonian property with a formula that contain no $\Box$, $\Diamond^+$ or $\Box^+$ provides an alternative proof for the NP-hardness in Theorem 5.8, different from the one presented in [15].

Besides that, the formula that expresses the Hamiltonian property and the formula presented in [15] to express the propositional satisfiability problem (another NP-Complete problem) are remarkably similar, consisting of an alternating sequence of $\downarrow$'s and $\Diamond$'s. This suggests that the fragment presented in [15] 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.

References

[1] Areces, C., P. Blackburn and M. Marx, *A road-map on complexity for hybrid logics*, in: *Proceedings of the 8th Annual Conference of the EACSL*, Lecture Notes in Computer Science **1683** (1999), pp. 307–321.

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

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

[4] Barbosa, V. C., "An Introduction to Distributed Algorithms," MIT Press, 1996.

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

[6] Benevides, M. R. F. and L. M. Schechter, *Modal expressiveness of graph properties*, in: *Proceedings of the 2nd Workshop on Logical and Semantic Frameworks, with Applications*, Electronic Notes in Theoretical Computer Science **205** (2008), pp. 31–47.

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

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

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

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

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

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

[13] Karp, R., *Reducibility among combinatorial problems*, in: *Complexity of Computer Computations*, Plenum, New York, 1972 pp. 85–103.

[14] Lynch, N., "Distributed Algorithms," Morgan Kaufmann Publishers, San Mateo, 1996.

[15] ten Cate, B. and M. Franceschet, *On the complexity of hybrid logics with binders*, in: *Proceedings of the 19th International Workshop of Computer Science Logic*, Lecture Notes in Computer Science **3634** (2005), pp. 339–354.

[16] Venema, Y., *Lectures on the modal mu-calculus* (2008). URL http://staff.science.uva.nl/~yde/teaching/ml/mu/mu.pdf