Modal Expressiveness of Graph Properties

Mario R. F. Benevides[*]and L. Menasché Schechter[†]

{mario,luis}@cos.ufrj.br

Abstract

Graphs are among the most frequently used structures in Computer Science. In this work, we analyze how we can express some important graph properties such as connectivity, acyclicity and the Eulerian and Hamiltonian properties in a modal logic. First, we show that these graph properties are not definable in a basic modal language. Second, we discuss an extension of the basic modal language with fix-point operators, the modal μ-calculus. Unfortunately, even with all its expressive power, the μ-calculus fails to express these properties. This happens because μ-calculus formulas are invariant under bisimulations. Third, we show that it is possible to express some of the above properties in a basic hybrid logic. Fourth, we propose an extension of CTL* with nominals, that we call hybrid-CTL*, and then show that it can express the Hamiltonian property in a better way than the basic hybrid logic. Finally, we introduce a promising way of expressing properties related to edges and use it to express the Eulerian property.

1 Introduction

Graphs are among the most frequently used structures in Computer Science [6]. In this discipline, usually 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 [3, 10], where the underlying model of computation is built on top of a graph. In addition to this central role, in distributed systems, graphs are also important as tools for the description of resource sharing problems, scheduling problems, deadlock issues, and so on. The case of distributed systems is also particularly appealing from the standpoint of the use of graphs as modeling tools 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.

In this work, we analyze how we can express these "global" graph properties in a modal logic. A similar attempt to do this was presented in [4]. That work also tries to use modal languages to express graph properties, but it approaches

[*]The author was partially supported by a grant from CNPq.

[†]The author was supported by a D.Sc. scholarship from CNPq.

this issue from a different point of view. The key differences between the two works will be discussed in the last section of the paper.

Trying to express graph properties using modal logic is an interesting idea for a number of reasons. First, modal logic achieves a good balance between what can be expressed in the language and how complex (computationally) it is to make inferences in it. It is a logic that certainly has more expressive power than propositional logic, but it is still decidable, unlike first-order logic. Second, modal logic formulas are evaluated in structures that are essentially graphs, which makes it a very natural choice for our work.

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 $1 < i, j < n$. 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 $G = (V, R)$ 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 graphs 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 to allow the presence of 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 express the Hamiltonian property in a better way than it was expressed in the basic hybrid logic. In section 6, we introduce a promising way of expressing properties related to edges and use it to express the Eulerian property. Finally, in section 7 we draw our concluding remarks.

2 Basic Graph Language

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

Definition 1. *The* basic graph language *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:*

$$A ::= p \mid \top \mid \neg A \mid A_1 \wedge A_2 \mid \Diamond A \mid \Diamond^+ A$$

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

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

Definition 2. *A* frame *for the basic graph language 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 language is essentially a graph. This confirms our statement in the first section that modal languages are a very natural choice for this work.

Definition 3. *A* model *for the basic graph language 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., $\mathbf{V} : \Phi \mapsto \mathcal{P}(V)$.*

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 A in a model $\mathcal{M}$ at a vertex v, notation $\mathcal{M}, v \Vdash A$, 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. *$\mathcal{M}, v \Vdash \neg A$ iff $\mathcal{M}, v \nVdash A$;*
4. *$\mathcal{M}, v \Vdash A \wedge B$ iff $\mathcal{M}, v \Vdash A$ and $\mathcal{M}, v \Vdash B$;*
5. *$\mathcal{M}, v \Vdash \Diamond A$ iff there is a $w \in V$ such that vRw and $\mathcal{M}, w \Vdash A$;*
6. *$\mathcal{M}, v \Vdash \Diamond^+ A$ iff there is a $w \in V$ such that vR^+w and $\mathcal{M}, w \Vdash A$.*

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

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

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

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

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.

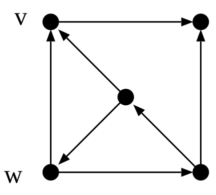

Figure 1: Model $\mathcal{M}$, where a formula A is satisfied at vertex v.

In a general level, the problem of determining whether there is a model $\mathcal{M}$ and a vertex v in $\mathcal{M}$ such that, for a given formula ϕ, $\mathcal{M}, v \Vdash \phi$ is called the *satisfiability problem* for ϕ and the problem of determining whether, for a given formula ϕ, $\mathcal{F} \Vdash \phi$, for all frames $\mathcal{F}$, is called the *validity problem* for ϕ. These two problems are duals to each other. In the basic graph logic, they are decidable, having EXPTIME complexity [5]. As we mentioned in the first section of the paper, this decidability is one of the reasons why it is an interesting idea to use modal logics in this work.

3 Modal Definability

In this section, we investigate whether some well-known graph properties are modally definable or not in the basic graph language. 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 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 two of these "similarity" results (more details about them and other related results may be found in [5]), and then we prove some theorems for graph properties using them.

Definition 5. *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:*

1. *w and $f(w)$ satisfy the same proposition symbols;*
2. *f is a homomorphism with respect to R (if wRv, then $f(w)R'f(v)$);*
3. *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 $\mathcal{M} \Rightarrow \mathcal{M}'$.*

Another important definition concerns collections of disjoint models. We say that two models $\mathcal{M}_1 = (W_1, R_1, \mathbf{V_1})$ and $\mathcal{M}_2 = (W_2, R_2, \mathbf{V_2})$ are *disjoint models* if and only if $W_1 \cap W_2 = \emptyset$.

Definition 6. *Let $\mathcal{M}_i = (W_i, R_i, \mathbf{V_i})$ be a collection of disjoint models. The* disjoint union $\uplus\mathcal{M}_i = (W, R, \mathbf{V})$ *is defined as*

- *W - the union of W_i;*
- *R - the union of R_i;*
- *$\mathbf{V}$ - for each proposition symbol p, $\mathbf{V}(p) = \bigcup \mathbf{V_i}(p)$.*

Similar definitions can be given for a bounded morphism and a disjoint union of frames, just removing the parts of the above definitions that deal with valuations.

Below are two basic theorems about modal definability that are going to be used throughout the next subsections. Their proofs for a language that contains only $\Diamond$ can be found at [5]. It is not difficult to extend that proof to a language that contains both $\Diamond$ and $\Diamond^+$.

Theorem 7. *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 8. *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 $\mathcal{F}' \Vdash \phi$.*

Theorem 9. *Let $\mathcal{M}_i = (W_i, R_i, \mathbf{V_i})$ be a collection of disjoint models and $\uplus\mathcal{M}_i = (W, R, \mathbf{V})$ their disjoint union. Then, $\mathcal{M}_i, w \Vdash \phi$ if and only if $\uplus\mathcal{M}_i, w \Vdash \phi$.*

Corollary 10. *Let $\mathcal{F}_i = (W_i, R_i)$ be a collection of disjoint frames and $\uplus\mathcal{F}_i = (W, R)$ their disjoint union. If $\mathcal{F}_i \Vdash \phi$ for every i, then $\uplus\mathcal{F}_i \Vdash \phi$.*

3.1 Connectivity

We can define two levels of connectivity for a graph. On a first level, 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. On a second level, 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 11. *Weak and strong connectivity are not modally definable.*

Proof. The disjoint union of connected graphs is not a connected graph. By corollary 10, since connectivity is not preserved under taking disjoint unions, it is not modally definable. □

3.2 Acyclicity

A graph G is said to be acyclic if and only if there is no path in G from any vertex v to itself.

Theorem 12. *Acyclicity is not modally definable.*

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 8, since acyclicity is not preserved under bounded morphic images, it is not modally definable. □

3.3 Hamiltonian Graphs

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

Theorem 13. *The class of Hamiltonian graphs is not modally definable.*

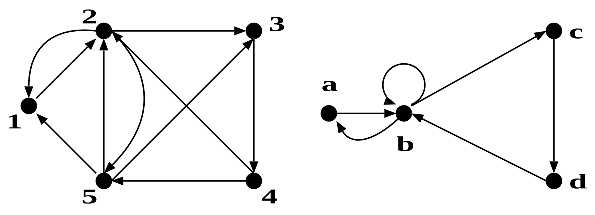

Figure 2: Graph 1,2,3,4,5 is Hamiltonian and graph a,b,c,d is not.

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 bounded morphism. By corollary 8, since the Hamiltonian property is not preserved under bounded morphic images, it is not modally definable. □

3.4 Eulerian Graphs

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 14 ([6]). *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 15. *The class of Eulerian graphs is not modally definable.*

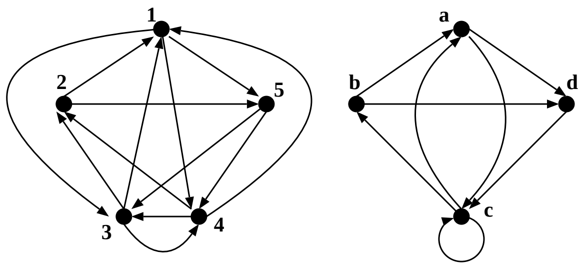

Figure 3: Graph 1,2,3,4,5 is Eulerian and graph a,b,c,d is not.

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 bounded morphism. By corollary 8, since the Eulerian property is not preserved under bounded morphic images, it is not modally definable. □

3.5 The Modal μ-Calculus

Looking at the results of the previous subsections, we see that, unfortunately, the *basic graph language* 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 [7, 15]. This language incorporates fix-point operators and is very expressive. In fact, not only the *basic graph language* can be embedded into the μ-calculus, but so can be the temporal languages LTL, CTL and CTL* [8].

Unfortunately, even with all this expressive power, 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) [7].

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 Language

As was shown in the previous section, the basic graph language does not have enough expressive power to describe the properties that we want. In order to achieve our goal, we need a language that is more expressive but, if possible, still decidable.

One interesting class of languages to take into consideration is the class of *hybrid languages* [2, 5]. In these languages, 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 language is an interesting choice because of a combination of factors. It improves the expressive power of that language, since hybrid formulas are no longer invariant under neither disjoint unions nor bounded morphic images [2], but it is still a decidable language (in fact, with complexity no worse than the one from the previous language) [1].

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

4.1 Language

Definition 16. *The* hybrid graph language *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:

$$A ::= p \mid i \mid \top \mid \neg A \mid A_1 \wedge A_2 \mid \Diamond A \mid \Diamond^+ A \mid @_i A$$

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

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

Definition 17. *A* model *for a hybrid graph language 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 the single vertex belonging to $\mathbf{V}(i)$.*

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

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

It is important to see that the operators $@_i$ are normal modal operators. For every nominal i, $@_i(A \rightarrow B) \rightarrow (@_i A \rightarrow @_i B)$ is a formula valid in all frames. Also, it is interesting to see that the operators $@_i$ are duals to themselves.

4.2 Hybrid Definability

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

Theorem 18. *A graph G, where G' is its underlying undirected graph, is* strongly connected *if and only if $\mathcal{F} \Vdash \phi$ and* (weakly) connected *if and only if $\mathcal{F}' \Vdash \phi$, where $\mathcal{F}$ is the frame that represents G, $\mathcal{F}'$ is the frame that represents G' and ϕ is the formula*

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

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 $(\mathcal{F}, \mathbf{V}), u \nVdash \phi$. Let $\mathbf{V}(i) = \{v\}$ and $\mathbf{V}(j) = \{w\}$. If $v = w$, then ϕ is satisfied, so we may assume that $v \neq w$. Then for ϕ to be falsified, 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. $\square$

Theorem 19. *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, $(\mathcal{F}, \mathbf{V}), u \not\Vdash \phi$, 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 $(\mathcal{F}, \mathbf{V}), u \not\Vdash \phi$. Let $\mathbf{V}(i) = \{v\}$. Then for ϕ to be falsified, we need $(\mathcal{F}, \mathbf{V}), u \Vdash @_i \Diamond^+ i$. This, on the other hand, is equivalent to vR^+v being true in G, which means that v is reachable from itself. This contradicts the fact that G is acyclic. $\square$

Before trying to find a formula to describe the Hamiltonian graphs, we need to consider some graph-theoretical issues. In graph theory [6], 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 20. *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 $@_i \varphi$. $\square$

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 21. *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 $\mathcal{F}$. 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 $\sigma'(k) = i_k$. 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

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

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 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 $\mathcal{F}$. $\square$

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 14), 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 falling again in a factorial-length formula, to state cardinality conditions on edges incident from and to a vertex, as is needed in theorem 14.

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 languages 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 6, where we describe a method to name edges within the framework of a hybrid language and use it to find a formula for the Eulerian property.

5 The Temporal Logic Hybrid-CTL*

The fact that the formula describing Hamiltonian graphs has factorial size makes its verification impossible. Of course, we can never expect to verify the Hamiltonian property in polynomial time, since determining whether a graph is Hamiltonian is an NP-Complete problem [9], but we may try to verify it a little faster than in factorial time. That is our goal in this section.

In order to write a short (with polynomial size) formula to describe the class of Hamiltonian graphs, we use the temporal branching-time logic CTL* [11] with nominals (*hybrid-CTL**). We could use the full hybrid μ-calculus presented in [13], since it contains the hybrid-CTL* [8]. But we will not do this, since the hybrid-CTL* is strong enough for what we want to do and its formulas are incredibly easier to read and to understand than hybrid μ-calculus formulas (which is another reason for the introduction of the hybrid-CTL*).

The hybrid-CTL* is also decidable. We can establish an exponential upper bound to the complexity of the validity problem in this hybrid temporal language, following the validity problem's complexity in the hybrid μ-calculus.

5.1 Language

Definition 22. *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 $\mathbf{A}$, $\mathbf{E}$, $\mathbf{X}$, $\mathbf{F}$, $\mathbf{G}$ and $\mathbf{U}$. Formulas are divided into* vertex formulas $\mathbf{S}$ *and* path formulas $\mathbf{P}$ *co-inductively defined as follows:*

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

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

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 23. *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 $\mathcal{M}, d \Vdash S$, where d 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".

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 24. *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*

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

with

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

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 δ_n. If δ_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 δ_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 δ_n are satisfied in this path. Then, since this path starts at v_1, $\mathbf{E}[\ldots]$ is satisfied in v_1 and δ_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 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 $\mathcal{F}$. $\square$

6 Edge-Related Properties

As was mentioned in the end of section 4, one way to describe the Eulerian property would be to use theorem 14. However, this would be very hard to do using our standard $\Diamond$ operators, or even the temporal operators defined in the previous section. 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 14. Although they exist in the literature ([12], for example), we don't want to introduce a whole new formalism. We want to express the Eulerian property in a hybrid language.

So, if we are not going to use theorem 14, 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 languages only have names for vertices. So, we first develop a method to name edges *within* the formalism of a hybrid language and later use it to define the Eulerian property.

6.1 Graph Subdivisions

Definition 25. *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 26. *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 V the set of vertices of G and V' the set of vertices of G', we have that $V' = V \cup V^*$ ($V \cap V^* = \emptyset$), where V^* is the set of new vertices added during the subdivision. We also have that every edge of G' has an extremity in V and the other in V^* and that there is a bijective map between elements of V^* and edges of G.

This bijective map between the set V^* 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 V^*. 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 V or in V^*. Thus, instead of working with one set of nominals $\mathcal{L}$, we will be working with two such sets, $\mathcal{L}_1$ and $\mathcal{L}_2$ ($\mathcal{L}_1 \cap \mathcal{L}_2 = \emptyset$). Instead of writing $G' = (V', R')$, we write $G' = (V, V^*, R')$, to make clear the difference between the two sets of vertices, and define valuations $\mathbf{V}$ as $\mathbf{V}(p) \in \mathcal{P}(V \cup V^*)$, if p is a proposition symbol, $\mathbf{V}(i) = \{v\}$, such that $v \in V$, if $i \in \mathcal{L}_1$ and $\mathbf{V}(j) = \{w\}$, such that $w \in V^*$, if $j \in \mathcal{L}_2$. We will denote the nominals in $\mathcal{L}_1$ by $i_1, i_2, \ldots$ and the nominals in $\mathcal{L}_2$ by $j_1, j_2, \ldots$.

6.2 The Eulerian Property

Since, as stated in the beginning of the section, 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 two previously presented formulas: the formulas in theorems 21 and 24. But the formula in the first theorem has factorial length, so we will only adapt the formula in the second theorem to the Eulerian case.

This is not a difficult task. The formula in theorem 24 states that, for a graph $G = (V, R)$, there is a cycle that visits every vertex in V exactly once (with the exception of the first vertex of the cycle). To define the Eulerian property, we need a formula that checks 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 V^* exactly once.

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

Theorem 27. *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*

$$\phi = \psi_m \to \delta_m,$$

with

$$\delta_m = @_{i_1}\mathbf{E}[\mathbf{F}j_1 \wedge \mathbf{F}j_2 \wedge \ldots \wedge \mathbf{F}j_m \wedge$$

$$\wedge \mathbf{G}(j_1 \to \mathbf{XG}\neg j_1) \wedge \mathbf{G}(j_2 \to \mathbf{XG}\neg j_2) \wedge \ldots \wedge \mathbf{G}(j_m \to \mathbf{XG}\neg j_m) \wedge \mathbf{XG}(i_1 \to \mathbf{G}i_1)].$$

Proof. As a consequence of G being connected, if a closed path goes through every edge in G, it also goes through every vertex in G and, as a consequence, through every vertex in G'. From this observation, the proof of the above theorem follows using the same ideas that are present in the proof of theorem 24. □

7 Conclusions

Our goal in this paper is to try to express, using modal logics, some "global" graph properties that are central to many computer science applications. [4] is a work closely related to this one. In that work, the interest was also in how to use modal logics to express "global" graph properties. One of the differences is that, in [4], only the basic modal logic, increased with the transitive modalities $\Diamond^+$ and $\Box^+$, was used. Also, that work had the goal of providing axiomatizations to classes of graphs with these "global" properties, while our approach in this work is to test whether a single formula could express each of these properties. Finally, [4] did not explore the Hamiltonian and the Eulerian properties. The work in [14] also has some similarity with ours. In that work, a fragment of first-order logic is used to describe graph properties in the particular context of graph transformations and graph rewriting. It would be interesting to analyze the interconnection between our work and [14].

In the present work, we presented various formalisms, from a very basic modal logic to a very powerful temporal logic and used them to define 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 languages, 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 languages satisfy. In these cases, the use of a hybrid language is a very simple way to bypass this problem. Hybrid languages have much weaker invariance conditions [2], which increases the number of definable properties.

Also, we showed a practical application of a hybrid temporal language. We used a hybridized version of CTL* to describe the class of Hamiltonian graphs. The introduction of hybrid-CTL* is also important on its own, since it is expressive enough for many applications, which makes the use of the full hybrid μ-calculus of [13], a language with very poor readability (as most fixpoint languages), unnecessary in such cases.

Finally, we describe in section 6 a way to name edges in the hybrid language using graph subdivisions, which does not require any major change in the language. 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 6, 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 in the hybrid language using this construction that allows us to name not only vertices but also edges.

References

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

[2] 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.

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

[4] 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.

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

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

[7] 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.

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

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

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

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

[12] H. J. Ohlbach, R. A. Schmidt, and U. Hustadt. Translating graded modalities into predicate logic. Technical Report MPI-I-95-2-008, Max-Planck-Institut für Informatik, Saarbrücken, 1995.

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

[14] M. Strecker. Modeling and verifying graph transformations in proof assistants. In *4th International Workshop on Computing with Terms and Graphs*, pages 112–124, 2007.

[15] Y. Venema. Lectures on the modal mu-calculus. 2007.