Modal Expressiveness of Graph Properties

LSFA 2007

Mario Benevides

and

Luis Menasché Schechter

`{mario,luis}@cos.ufrj.br`

COPPE - UFRJ

Introduction

Goal: In this work, we analyze how we can express global *graph properties* (connectivity, acyclicity and the Hamiltonian and Eulerian properties) in a *modal logic*.

Why graphs? Graphs are among the most frequently used structures in Computer Science. 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.

Introduction (cont.)

Example: In the field of distributed systems, 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.

Introduction (cont.)

Why modal logic? Trying to express graph properties using modal logic is an interesting idea for a number of reasons.

1. 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. In fact, modal logic is a very well-behaved fragment of first-order logic.

Introduction (cont.)

2. Modal logic formulas are evaluated in structures that are essentially graphs, which makes it a very natural choice for our work.

Basic Graph Language

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 $\vee$, $\rightarrow$ and $\leftrightarrow$ and also the following abbreviations for the duals: $\Box A := \neg \Diamond \neg A$ and $\Box^+ A = \neg \Diamond^+ \neg A$.

Frames and Models

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 that modal languages are a very natural choice for this work.

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., $\mathbf{V} : \Phi \mapsto Pow(V)$.

Modal Definability

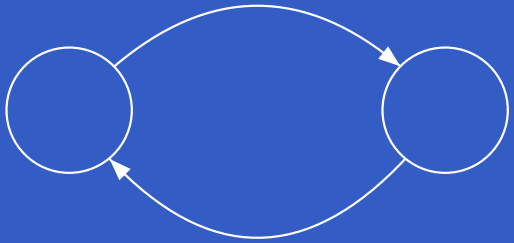
Theorem: Frame validities in the basic graph language are preserved under disjoint unions and bounded morphic images.
Corollary: Connectivity (both weak and strong), acyclicity and the Hamiltonian and Eulerian properties cannot be expressed in the basic graph language.

Connectivity

The disjoint union of connected graphs is not a connected graph. Since connectivity is not preserved under disjoint unions, it is not definable in the basic graph language.

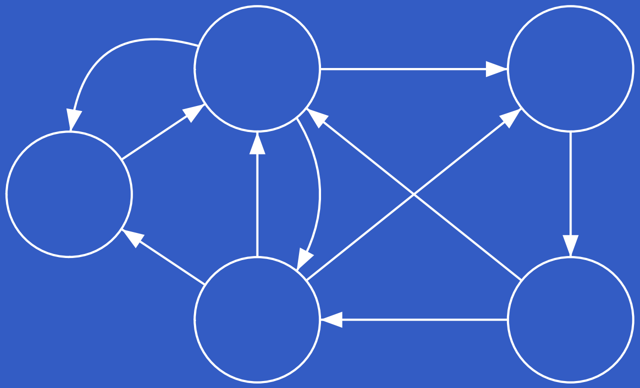
Acyclicity

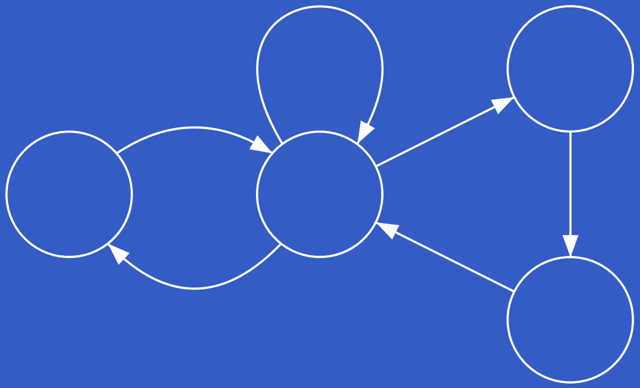
There is a bounded morphism between the (infinite) frame isomorphic to $\mathbb{N}$ and the graph below. The first is acyclic, while the second is not. Since acyclicity is not preserved under bounded morphic images, it is not definable in the basic graph language.



Hamiltonian Property

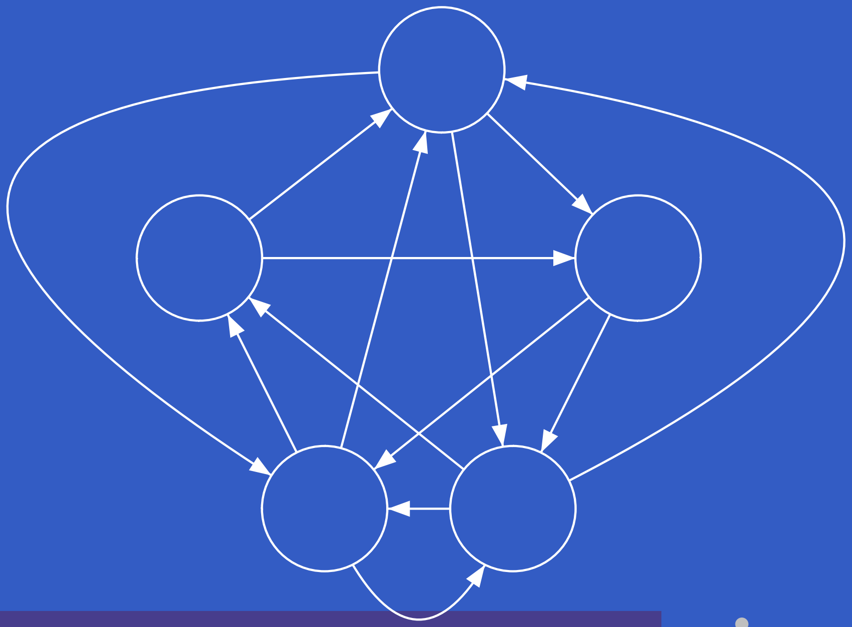
There is a bounded morphism between the two graphs below. The first is Hamiltonian, while the second is not. Since the Hamiltonian property is not preserved under bounded morphic images, it is not definable in the basic graph language.

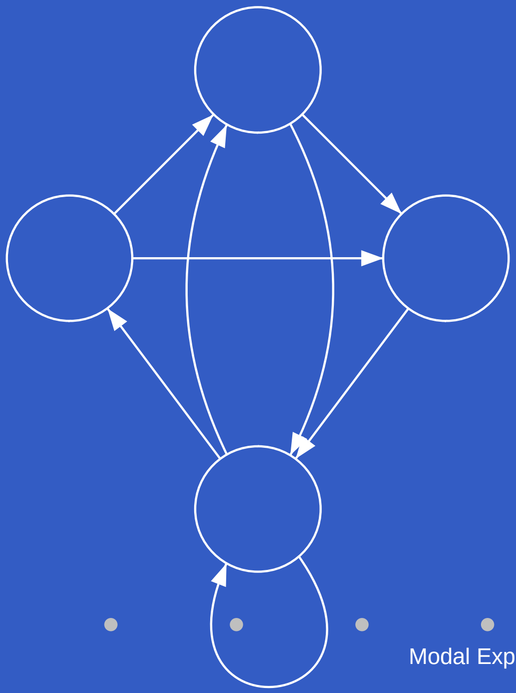




Eulerian Property

There is a bounded morphism between the two graphs below. The first is Eulerian, while the second is not. Since the Eulerian property is not preserved under bounded morphic images, it is not definable in the basic graph language.





Hybrid Language

In order to achieve our goal, we need a language that is more expressive but, if possible, still decidable. The hybrid language is an interesting choice because of a combination of factors. It improves the expressive power of the language presented previously, since hybrid formulas are no longer invariant under neither disjoint unions nor bounded morphic images, while the language is still decidable.

Hybrid Graph Language

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 $\Diamond^+$.

Hybrid Graph Language (cont.)

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 $\vee$, $\rightarrow$, $\leftrightarrow$, $\Box A$ and $\Box^+ A$.

Frames and Models

The definition of a *frame* is the same as the one for the basic graph language.
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., $\mathbf{V} : \Phi \mapsto Pow(V)$ 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 $\mathbf{V}(i)$.

Hybrid Definability - Connectivity

Theorem: 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

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

Acyclicity

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

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

The Hamiltonian Property

Let $\mathcal{L}_n = \{i_1, \ldots, i_n\}$ be a set containing n nominals. Before defining a formula for the Hamiltonian property, we define a formula that is true 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: 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

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

The Hamiltonian Property (cont.)

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: 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

$$\phi = \psi_n \longrightarrow \delta_n,$$

The Hamiltonian Property (cont.)

with

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

$$\wedge \Diamond(\sigma(n) \wedge \Diamond \sigma(1)) \ldots).$$

The Eulerian Property

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 to state the Euler condition.
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

The Eulerian Property (cont.)

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 will do this, describing a method to name edges within the framework of a hybrid language and using it to find a formula for the Eulerian property.

The Temporal Logic Hybrid-CTL*

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* with nominals (hybrid-CTL*).
The hybrid-CTL* language is a temporal language consisting of a set Φ of countably many proposition symbols (the elements of Φ are denoted by $p_1, p_2, \ldots$), a set $\mathcal{L}$ of countably many nominals (the elements of $\mathcal{L}$ are denoted by $i_1, i_2, \ldots$) such that $\Phi \cap \mathcal{L} = \emptyset$, the boolean connectives $\neg$ and $\wedge$ and the operators $@_i$, for each nominal i, and **A**, **E**, **X**, **F**, **G** and **U**.

Hybrid-CTL* (cont.)

Formulas are divided into *vertex formulas* **S** and *path formulas* **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 hybrid graph language.

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: 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

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

The Hamiltonian Property (cont.)

with

$$\delta_n = @_{i_1}\mathbf{E}[\mathbf{XF}i_1 \wedge \mathbf{F}i_2 \wedge \ldots \wedge \mathbf{F}i_n \wedge$$

$$\wedge \mathbf{XG}(i_1 \longrightarrow \mathbf{G}i_1) \wedge \mathbf{G}(i_2 \longrightarrow \mathbf{XG}\neg i_2) \wedge \ldots \wedge$$

$$\wedge \mathbf{G}(i_n \longrightarrow \mathbf{XG}\neg i_n)].$$

Edge-Related Properties

Definition: 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: 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.

Edge-Related Properties (cont.)

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.

Edge-Related Properties (cont.)

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.

Edge-Related Properties (cont.)

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 $\mathbf{V}(p) \in Pow(V \cup V^*)$, if p is a proposition symbol,

Edge-Related Properties (cont.)

$\mathbf{V}(i) = \{v\}$, 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 $j_1, j_2, \ldots$.

The Eulerian Property

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: 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

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

The Eulerian Property (cont.)

with

$$\delta_m = @_{i_1}\mathbf{E}[\mathbf{F}j_1 \wedge \mathbf{F}j_2 \wedge \ldots \wedge \mathbf{F}j_m \wedge$$
$$\wedge \mathbf{G}(j_1 \to \mathbf{XG}\neg j_1) \wedge \mathbf{G}(j_2 \to \mathbf{XG}\neg j_2) \wedge \ldots \wedge$$
$$\wedge \mathbf{G}(j_m \to \mathbf{XG}\neg j_m) \wedge \mathbf{XG}(i_1 \to \mathbf{G}i_1)].$$

Concluding Remarks

- 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.

Concluding Remarks (cont.)

- This work is an interesting way of exposing an important issue. Sometimes, standard modal languages, even the ones that are incredibly expressive, are not capable of expressing some important properties. This happens because of some strong invariance conditions 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, which increase the number of definable properties.

Concluding Remarks (cont.)

- We described a way to name edges in the hybrid language using graph subdivisions, which does not require any major change in the language. It would 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.