An Unified Characterization of Chordal, Interval, Indifference, and Other Classes of Graphs

João Meidanis
DCC - IMECC - UNICAMP
meidanis@dcc.unicamp.br

Relatório Técnico DCC–27/93

Setembro de 1993

An Unified Characterization of Chordal, Interval, Indifference, and Other Classes of Graphs

João Meidanis
DCC - IMECC - UNICAMP
meidanis@dcc.unicamp.br

Abstract

Motivated by very similar characterizations given in the literature for chordal, interval, and indifference graphs we introduce a way of defining classes of graphs in terms of boolean functions of two variables. Each such function originates a class of graphs. Denoting the variables by **1** and **2**, the functions $\mathbf{1} \Rightarrow \mathbf{2}$, **1**, and $\mathbf{1} \wedge \mathbf{2}$ originate the classes of chordal, interval, and indifference graphs, respectively. In this paper we study the classes corresponding to all other boolean functions, as well as general properties of these classes. A satisfactory identification of the class corresponding to function $\mathbf{1} \oplus \mathbf{2}$ (exclusive or) is still open.

1 Introduction

Chordal graphs, interval graphs, and indifference graphs have numerous applications and have been extensively studied [Gol80]. Our goal in this paper is to point out that among the several known characterizations for these classes three of them (one for each class) differ only slightly. The difference lies in a certain formula that includes a boolean function of two variables. We then study the classes obtained by using any of the sixteen possible boolean functions of two variables in this general framework. Our results are presented in form of a chart showing also the containment relations among classes.

Such an unified characterization is useful in several ways. It helps us understand the nature of these classes and the properties they have in common. Also, many of the classes admit polynomial-time algorithms for membership and for various combinatorial problems that are NP-hard in general, such as coloring, maximum clique, Hamiltonian tour, etc. The unified approach may be the basis for designing efficient algorithms for these problems that work uniformly for many classes.

The rest of the paper is organized as follows. The next section contains some basic definitions needed throughout. In Section 3 we present the characterizations that motivated this work. Section 4 contains the definitions of the classes studied, a correspondence between classes and boolean functions, and a chart showing containment relations. General properties of the classes also appear here. Our final comments, open problems, and plans for future work are given in Section 5.

2 Basic definitions

A *graph* is a pair (V, E) where V is a finite set and E is a subset of $V \times V$ with the following properties. We will use the notation uv to represent the pair (u, v).

- for all $u \in V$, $uu \notin E$.
- if $uv \in E$ then $vu \in E$.

The elements of V are called *nodes* and the elements of E are called *edges* of the graph.

If $uv \in E$, nodes u and v are *adjacent* to each other. The *degree* of a node v, denoted by $\deg(v)$, is the number of nodes in V adjacent to it. We also say that edge uv is *incident* to nodes u and v.

In a graph $G = (V, E)$, given a subset H of V, the *subgraph induced* by H is the graph $G' = (H, E')$ where $E' = \{(x, y) \in E \mid x \in H \text{ and } y \in H\}$.

The *neighborhood* of a node v is the set $\mathrm{N}(v) = \{v\} \cup \{x \in V \mid xv \in E\}$.

A *clique* is a subset of V whose elements are pairwise adjacent. Singleton sets are cliques.

A node is *simplicial* if its neighborhood is a clique.

A *path* is a sequence $(v_0, v_1, \ldots, v_k)$ of nodes such that $v_{i-1}v_i \in E$ for $i = 1, 2, \ldots, k$. A path is called *simple* if all nodes in it are distinct. If $(v_0, v_1, \ldots, v_k)$ is a simple path and every node not in it is adjacent to at least one v_i, this is a *dominating* path.

A *cycle* is a simple path $(v_0, v_1, \ldots, v_k)$ such that $k \geq 2$ and $v_k v_0 \in E$.

A *chord* in a path or cycle is an edge incident to two nonconsecutive nodes.

In a graph $G = (V, E)$, a subset $S \subseteq V$ is *connected* when there is a path containing all nodes in S. A *connected component* of a graph is a maximal (with respect to set inclusion) connected subset. When V itself is connected we say that G is a connected graph.

A *linear order* on a finite set S is a binary relation "$<$" satisfying the following properties.

- for all $a \in S$, $a \not< a$.
- if $a < b$ and $b < c$ then $a < c$.
- for all $a, b \in S$ with $a \neq b$, either $a < b$ or $b < a$ holds, but not both.

3 Motivation

Interval graphs are graphs whose nodes can be put into one-to-one correspondence with closed intervals in the real line in such a way that adjacent nodes correspond to intervals that intersect. If all intervals can be chosen of the same size, we have an *indifference graph*[1]. Interval graphs are special cases of *chordal graphs*, which are those where every cycle with four or more nodes has at least one chord. (A chord is an

[1] The denominations *unit interval graph*, *proper interval graph*, and *time graph* are also widely used for these graphs.

edge linking two nonconsecutive nodes in the cycle.) Chordal graphs are also called *triangulated graphs* in the literature.

In 1976, Rose, Tarjan, and Lueker [RTL76] gave the following characterization of chordal graphs.

Characterization 1 *A graph is chordal if and only if there is a linear order of its nodes such that for every triple u, v, w with $u < v < w$ we have*

$$(uw \in E \text{ and } vw \in E) \Rightarrow uv \in E.$$

In other words, there is a linear ordering such that each node is simplicial in the graph induced by itself and all its predecessors.

For interval graphs, a similar characterization, given below, appears in the literature. It can be found, for instance, in work by Olariu [Ola91], Ramalingam and Pandu Rangan [RPR88] and is implicitly present is a seminal paper by Benzer [Ben59], which constitutes one of the first applications of interval graphs in genetics.

Characterization 2 *A graph is interval if and only if there is a linear order of its nodes such that for every triple u, v, w with $u < v < w$ we have*

$$uw \in E \Rightarrow uv \in E.$$

Finally, for indifference graphs we have the following characterization, proved by Maehara [Mae80], among others.

Characterization 3 *A graph is indifference if and only if there is a linear order of its nodes such that for every triple u, v, w with $u < v < w$ we have*

$$uw \in E \Rightarrow (uv \in E \text{ and } vw \in E).$$

The similarity among these characterizations is striking. Denoting by **1**, **2**, and **3** the predicates $uv \in E$, $vw \in E$, and $uw \in E$, respectively (see Figure 1), we see that the only difference in the three characterizations is the final statement, according to the following list.

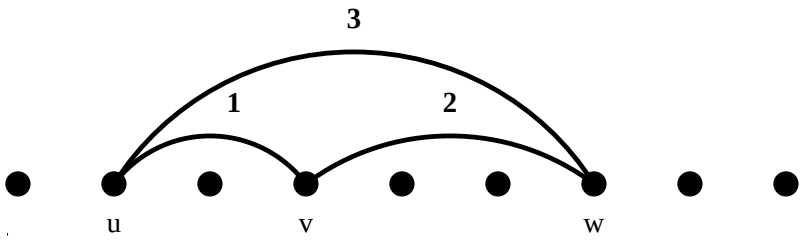

Figure 1: Common framework for the unified characterization.

chordal:	(**3** and **2**)	$\Rightarrow$	**1**
interval:	**3**	$\Rightarrow$	**1**
indifference:	**3**	$\Rightarrow$	(**1** and **2**)

If we now replace the statement (**3** and **2**) $\Rightarrow$ **1** by its logical equivalent **3** $\Rightarrow$ (**2** $\Rightarrow$ **1**), we reduce all three statements to the general form

$$\mathbf{3} \Rightarrow f(\mathbf{1}, \mathbf{2}),$$

where f is a boolean function of two variables. It is natural to ask which classes of graphs are defined by other choices of the function f, and this will be the object of the next section.

4 Boolean functions and classes of graphs

Given a boolean function f of two variables, we denote by $\mathcal{C}(f)$ the class of graphs defined as follows.

Definition 1 *A graph G belongs to class $\mathcal{C}(f)$ if and only if there is a linear order of its nodes such that for every triple u, v, w with $u < v < w$ we have*

$$\mathbf{3} \Rightarrow f(\mathbf{1}, \mathbf{2}),$$

where **1**, **2**, *and* **3** *denote the existence of edges uv, vw, and uw, respectively.*

$\mathbf{1}$	$\mathbf{2}$	$\mathbf{F}$	$\mathbf{1} \wedge \mathbf{2}$	$\mathbf{1} \wedge \overline{\mathbf{2}}$	$\mathbf{1}$	$\mathbf{2} \wedge \overline{\mathbf{1}}$	$\mathbf{2}$	$\mathbf{1} \oplus \mathbf{2}$	$\mathbf{1} \vee \mathbf{2}$
F	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**
F	**T**	**F**	**F**	**F**	**F**	**T**	**T**	**T**	**T**
T	**F**	**F**	**F**	**T**	**T**	**F**	**F**	**T**	**T**
T	**T**	**F**	**T**	**F**	**T**	**F**	**T**	**F**	**T**

$\mathbf{1}$	$\mathbf{2}$	$\overline{\mathbf{1}} \wedge \overline{\mathbf{2}}$	$\mathbf{1} \equiv \mathbf{2}$	$\overline{\mathbf{2}}$	$\mathbf{2} \Rightarrow \mathbf{1}$	$\overline{\mathbf{1}}$	$\mathbf{1} \Rightarrow \mathbf{2}$	$\overline{\mathbf{1}} \vee \overline{\mathbf{2}}$	$\mathbf{T}$
F	**F**	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**
F	**T**	**F**	**F**	**F**	**F**	**T**	**T**	**T**	**T**
T	**F**	**F**	**F**	**T**	**T**	**F**	**F**	**T**	**T**
T	**T**	**F**	**T**	**F**	**T**	**F**	**T**	**F**	**T**

Table 1: Truth-tables of all possible two-variable boolean functions. The variables are indicated by **1** and **2**. Logical symbols are defined in the text.

We say that such a linear order *satisfies* $\mathbf{3} \Rightarrow f$ or that it is *f-admissible* for G (or just *admissible* if f is clear from the context).

There are a total of sixteen distinct boolean functions of two variables. Table 1 shows their truth-tables, and the symbolic names we will use for them throughout the paper. We use $\overline{a}$ to indicate the logical negation of a, $\wedge$ for logical AND, $\vee$ for logical OR, $\Rightarrow$ for implication ($a \Rightarrow b$ is the same as $\overline{a} \vee b$), $\equiv$ for logical equivalence, and $\oplus$ for exclusive-OR. In addition, **T** and **F** stand for the logical values *true* and *false*, respectively.

The corresponding classes of graphs are shown in Fig. 2. For each class, its associated boolean function appears above the name of the class. Class names and their definitions are listed in the sequel. The classes Indifference and Paths occur twice in the hierarchy.

Define the *level* of a boolean function as the number of **T**'s in its truth-table. Functions of the same level are located in the same horizontal line in the diagram.

To better understand this diagram, two observations are in order. The first one is that $\mathcal{C}(f(\mathbf{1},\mathbf{2}))$ coincides with $\mathcal{C}(f(\mathbf{2},\mathbf{1}))$. This is a con-

sequence of the fact that if a linear order satisfies $\mathbf{3} \Rightarrow f(\mathbf{1},\mathbf{2})$, the reverse order satisfies $\mathbf{3} \Rightarrow f(\mathbf{2},\mathbf{1})$. Thus, we put $f(\mathbf{1},\mathbf{2})$ and $f(\mathbf{2},\mathbf{1})$ in the same place in the diagram, separated by a comma. Symmetric functions appear by themselves.

The other observation is that if two functions f and g are such that $f \Rightarrow g$, then $\mathcal{C}(f) \subseteq \mathcal{C}(g)$. This follows from the classical properties of logical implication: if a linear order satisfies $\mathbf{3} \Rightarrow f$ then it will surely satisfy $\mathbf{3} \Rightarrow g$ as well. To indicate these inclusions, we drew an arrow in Figure 2 from class $\mathcal{C}(f)$ to class $\mathcal{C}(g)$ whenever $f \Rightarrow g$ and g is exactly one level above f. Other inclusions follow by transitivity. Notice that if $f(\mathbf{1},\mathbf{2}) \Rightarrow g(\mathbf{1},\mathbf{2})$ then $f(\mathbf{2},\mathbf{1}) \Rightarrow g(\mathbf{2},\mathbf{1})$, so this *boolean containment* is well defined for classes.

Proofs of the correspondences between functions and classes in this diagram will be given in Section 4.3. We were not able to find a name for class $\mathcal{C}(\mathbf{1} \oplus \mathbf{2})$ in the literature.

4.1 Definition of the classes

Although most of the classes are well known, we formally define them here for completeness. Indifference, interval, and chordal graphs were already defined in Section 3.

For the others, a few more basic graph concepts are needed. The *complement* of a graph $G = (V, E)$ is the graph $(V, (V \times V) \setminus E)$.

We say that a graph can be *transitively oriented* when we can orient the edges, i. e., select a node $s(e)$ incident to e for each edge e ($s(e)$ is the *head* of the edge), in a transitive way, i. e., whenever we have edges uv and vw with $s(uv) = v$ and $s(vw) = w$, uw is an edge in the graph and $s(uw) = w$.

A list of our class definitions follows.

Forest: A graph is a forest when it contains no cycles.

Paths: A graph belongs to this class when each of its connected components is a simple path.

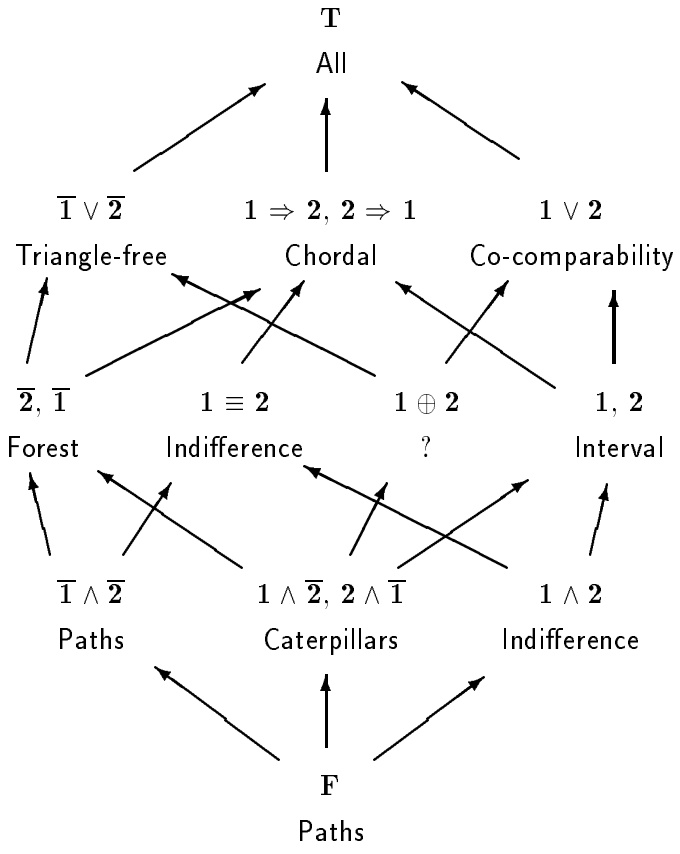

Figure 2: Classes of graphs corresponding to boolean functions. Arrows indicate containment relations.

Caterpillars: A graph is in this class when it is a forest whose internal nodes induce a graph in Paths.

Triangle-free: A graph is triangle-free when it has no clique with three or more nodes.

Co-comparability: A graph is in this class when its complement can be transitively oriented.

All: This class includes all graphs.

Graphs in Pathscan be characterized as forests in which every node has degree at most two.

Caterpillars can be also characterized as the graphs obtainable from a graph $G \in$ Paths by adding neighbors of degree one to the nodes of G. In a connected caterpillar such a graph will always be a dominating path. Given a dominating path, its nodes are the *body segments* and the remaining nodes are the *feet* of the caterpillar.

4.2 General properties

The classes $\mathcal{C}(f)$ have a number of properties that do not depend on the particular function f used to construct them. We now give some of these common properties.

Theorem 1 *For any f, if $G \in \mathcal{C}(f)$ and H is an induced subgraph of G, then $H \in \mathcal{C}(f)$.*

Proof: The removal of all nodes not in H and their incident edges from an admissible order for G gives an admissible order for H. □

Theorem 2 *For any f, a graph G belongs to $\mathcal{C}(f)$ if and only if all its connected components belong to $\mathcal{C}(f)$.*

Proof: If all connected components of a graph admit linear orderings in which $\mathbf{3} \Rightarrow f$, the concatenation of all these orderings produces an admissible order for G. On the other hand, each connected component is an induced subgraph, so the result follows from Theorem 1. □

4.3 Proofs

Many of the associations presented in Figure 2 are proved in the literature. We already mentioned references for chordal, interval, and indifference graphs. The fact that Co-comparability equals $\mathcal{C}(\mathbf{1} \vee \mathbf{2})$ is well known [Dam92].

The class $\mathcal{C}(T)$ is trivially All, because no restrictions are placed on the order. For the class Triangle-free the result is also easy. Any order will do for graphs in the class, and no order will work for graphs not in the class.

We illustrate the remaining results presented in Figure 2 with the following examples.

Theorem 3 $\mathcal{C}(\mathbf{F}) =$ Paths.

Proof: A graph $G \in \mathcal{C}(\mathbf{F})$ admits a linear ordering of its nodes satisfying $\mathbf{3} \Rightarrow \mathbf{F}$, which is logically equivalent to $\overline{\mathbf{3}}$. Therefore, all edges connect consecutive nodes in this order. No cycles are possible, hence G is a forest. Each node is connected to at most two nodes, namely, its predecessor and its successor (if they exist), so it has degree at most two.

Conversely, we can prove that Paths $\subseteq \mathcal{C}(\mathbf{F})$ by induction on n, the number of nodes in the graph. We claim that any graph in Paths admits a linear ordering without **3** edges. Furthermore, we claim that this ordering can start at any node of degree at most one. We prove both claims simultaneously by induction.

A graph with one node obviously satisfies the claims. Given a graph G with $n > 1$ nodes, select arbitrarily a node $x \in G$ with degree at most one. Such a node necessarily exists since G is a forest. Removal of x and all its incident edges leaves a graph G' that also belongs to Paths. By

induction hypothesis, there is a linear ordering for G' without **3** edges. Add x in front of this ordering. If $\deg(x) = 0$ we are done. Otherwise, start the order in G' with the only neighbor y of x in G, which is possible since y has degree at most one in G'. The only new edge introduced by x connects it to its successor y and is not a **3** edge. □

Theorem 4 $\mathcal{C}(\mathbf{2} \wedge \overline{\mathbf{1}}) = \mathsf{Caterpillars}$.

Proof: We will prove the inclusion $\mathcal{C}(\mathbf{2} \wedge \overline{\mathbf{1}}) \subseteq \mathsf{Caterpillars}$ by induction on n, the number of nodes of G. In light of Theorem 2, we may restrict ourselves to connected graphs. Let $G \in \mathcal{C}(\mathbf{2} \wedge \overline{\mathbf{1}})$ be a connected graph with an admissible order. We claim that G is a caterpillar and the first node in the order belongs to a dominating path. For one-node graphs the claim is obvious. For $n > 1$, consider a linear ordering of the nodes of G satisfying $\mathbf{3} \Rightarrow (\mathbf{2} \wedge \overline{\mathbf{1}})$. Take the first node x in the order (see Figure 3). Since $n > 1$ and G is connected, there is at least one edge incident to x. Observe that this must be the only edge incident to x, otherwise $\mathbf{3} \Rightarrow \mathbf{1}$ would hold for at least one triple. Let y be x's only neighbor. The condition $\mathbf{3} \Rightarrow (\mathbf{2} \wedge \overline{\mathbf{1}})$ now implies that all nodes strictly between x and y must be adjacent to y and that they all have degree one. Removing x and all these nodes we obtain a smaller $G' \in \mathcal{C}(\mathbf{2} \wedge \overline{\mathbf{1}})$. Notice that we did not disconnect the graphs because we removed nodes of degree one only. By induction hypothesis, this is a caterpillar and y is part of a dominating path in G'. It follows that G is also a caterpillar with x preceding y in the dominating path and the nodes between x and y being extra feet adjacent to the body segment y.

Conversely, a caterpillar can be easily given an admissible order by selecting a dominating path, which will give the skeleton of the linear order, and inserting each remaining node immediately before its neighbor in the path. □

Theorem 5 $\mathcal{C}(\overline{\mathbf{2}}) = \mathsf{Forest}$.

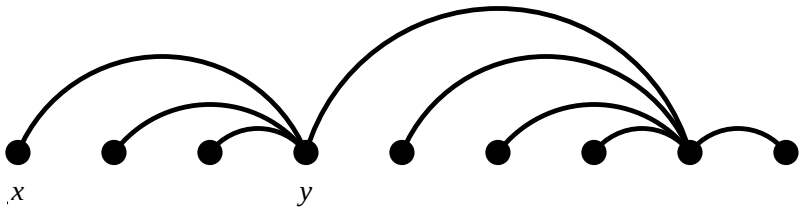

Figure 3: A caterpillar.

Proof: For $\mathcal{C}(\overline{\mathbf{2}}) \subseteq \mathsf{Forest}$ let G be a graph in $\mathcal{C}(\overline{\mathbf{2}})$ with an $(\overline{\mathbf{2}})$-admissible order. From each node x there is at most one backward edge (that is, an edge going to a node $y < x$). Hence, no cycles may exist. The rightmost node in a cycle would have two backward edges.

Conversely, let G be a graph in Forest. For each connected component, choose an arbitrary node and perform a breadth-first search (BFS) starting at x [Tar83]. Since there are no cycles, all nodes except the first will be adjacent to exactly one previous node. Hence, the BFS order is $(\overline{\mathbf{2}})$-admissible. □

Theorem 6 $\mathcal{C}(\overline{\mathbf{1}} \wedge \overline{\mathbf{2}}) = \mathsf{Paths}$.

Proof: By boolean containment and Theorem 3, it suffices to show that $\mathcal{C}(\overline{\mathbf{1}} \wedge \overline{\mathbf{2}}) \subseteq \mathsf{Paths}$. Let G be a graph with an $(\overline{\mathbf{1}} \wedge \overline{\mathbf{2}})$-admissible order. In view of Theorem 2, we may assume that G is connected.

If u, v, w are such that $u < v < w$ and $uw \in E$, we know that $uv \notin E$, and $vw \notin E$. Since G is connected, there must be a path $(v_0, \ldots, v_k)$ with $v_0 = u$ and $v_k = v$. Let x be the leftmost node in this path. Clearly $x \neq v_k$. If $x \neq v_0$, x has two neighbors in the path, both to its right, contradicting the $(\overline{\mathbf{1}} \wedge \overline{\mathbf{2}})$-admissibility of the order. Therefore $x = v_0 = u$. An analogous argument shows that v is the rightmost node in the path. But then $v_1 < v < w$ and u is adjacent to two nodes to its right, namely, v_1 and w, a contradiction. Thus, no triple $u < v < w$ with $uw \in E$ may exist. Hence $G \in \mathcal{C}(\mathbf{F}) = \mathsf{Paths}$. □

Theorem 7 $\mathcal{C}(\mathbf{1} \equiv \mathbf{2}) = \mathsf{Indifference}$.

Proof: It suffices to show that $\mathcal{C}(\mathbf{1} \equiv \mathbf{2}) \subseteq \mathcal{C}(\mathbf{1} \wedge \mathbf{2})$. Let G be a graph with an $(\mathbf{1} \equiv \mathbf{2})$-admissible order. Without loss of generality assume that G is connected.

If there is a triple (u, v, w) with $u < v < w$, $uw \in E$, $uv \notin E$, and $vw \notin E$, choose one such triple with minimum number of nodes between u and w. Consider a path $(v_0, \ldots, v_k)$ with $v_0 = u$ and $v_k = v$. Take this path as short as possible, that is, with minimum k. Shortest paths between two nodes are necessarily chordless.

Let x be the leftmost node in this path. Clearly $x \neq v_k$. If $x \neq v_0$, x has two neighbors in the path, both to its right, say x_1 and x_2. By the $(\mathbf{1} \equiv \mathbf{2})$-admissibility of the order, x_1 and x_2 are adjacent, which is impossible since the path is chordless. Therefore $x = v_0 = u$. An analogous argument shows that v is the rightmost node in the path. But then $u < v_1 < v < w$ and by admissibility we have $v_1 w \in E$ and $v_1 v \notin E$. This contradicts our choice of (u, v, w) and we conclude that no such triple may exist. Hence $G \in \mathcal{C}(\mathbf{1} \wedge \mathbf{2})$. □

4.4 Intersection of classes

In this section we present an interesting property of the correspondence between classes and boolean functions. We first need a canonical way of selecting one function from each pair of the form $f(\mathbf{1}, \mathbf{2}), f(\mathbf{2}, \mathbf{1})$ that define the same class. Following a suggestion by Jorge Stolfi, we consider only functions f satisfying $f(\mathbf{F}, \mathbf{T}) \Rightarrow f(\mathbf{T}, \mathbf{F})$.

Definition 2 *A function f is* leftist *when $f(\mathbf{F}, \mathbf{T}) \Rightarrow f(\mathbf{T}, \mathbf{F})$.*

Theorem 8 *If f and g are leftist boolean functions of two variables, then*

$$\mathcal{C}(f \wedge g) = \mathcal{C}(f) \cap \mathcal{C}(g).$$

Proof: (sketch.) Unfortunately, we don't have an elegant proof of this nice result. All we can offer is a tedious analysis of all cases.

First of all, observe that the result is obviously true if $f \Rightarrow g$ or $g \Rightarrow f$. Second, notice that $\mathcal{C}(f \wedge g) \subseteq \mathcal{C}(f) \cap \mathcal{C}(g)$ is also true in all cases by boolean containment. A final observation is that it suffices to prove the claim for functions of the same level.

In the end, we need to prove directly only the five cases below.

Triangle-free $\cap$ Chordal $\subseteq$ Forest: If a chordal graph has a cycle, it must have a triangle. Hence, chordal, triangle-free graphs cannot have cycles.

Triangle-free $\cap$ Co-comparability $\subseteq \mathcal{C}(\mathbf{1} \oplus \mathbf{2})$: This follows from the fact that any $(\mathbf{1} \vee \mathbf{2})$-admissible order for a triangle-free graph is also $(\mathbf{1} \oplus \mathbf{2})$-admissible.

Chordal $\cap$ Co-comparability = Interval: This is a well-known characterization of interval graphs [Sim91, GH64].

Forest $\cap$ Co-comparability $\subseteq$ Caterpillars: Let $G = (V, E)$ be a connected forest (i.e., a *tree*) that is in Co-comparability. Consider an $(\mathbf{1} \vee \mathbf{2})$-admissible order for G. Take a simple path $p = (v_0, \ldots, v_k)$ from the first to the last node in this order. We will show that p is a dominating path. If x is any node in G not in the path, x must be an internal node in the order and there must be an index i such that $0 \leq i < k$ and $v_i < x < v_{i+1}$. Since $v_i v_{i+1} \in E$, by $(\mathbf{1} \vee \mathbf{2})$-admissibility x is adjacent to either v_i or v_{i+1}. This shows that p is a dominating path and therefore G is a caterpillar.

Forest $\cap$ Indifference $\subseteq$ Paths: It is well-known that indifference graphs do not have $K_{1,3}$ as an induced subgraph [Jac92]. ($K_{1,3}$ is the graph with four nodes, with one of them adjacent to the other three and no other edges.) It follows that forests that are indifference graphs cannot have nodes of degree three or higher.

We leave to the reader the verification that all other cases can be proved by repeated application of the above results. $\square$

The functions must be leftist otherwise the theorem does not necessarily hold as the following example shows. We know that $\mathcal{C}(\overline{\mathbf{1}}) = \mathsf{Forest} = \mathcal{C}(\overline{\mathbf{2}})$ and $\mathcal{C}(\mathbf{1}) = \mathsf{Interval} = \mathcal{C}(\mathbf{2})$. If we use the leftist function $f = \mathbf{1}$ with the nonleftist $g = \overline{\mathbf{1}}$ we get

$$\mathcal{C}(f \wedge g) = \mathcal{C}(\mathbf{1} \wedge \overline{\mathbf{1}}) = \mathcal{C}(\mathbf{F}) = \mathsf{Paths},$$

while

$$\mathcal{C}(f) \cap \mathcal{C}(g) = \mathsf{Forest} \cap \mathsf{Interval} = \mathsf{Caterpillars}.$$

4.5 The class $\mathcal{C}(\mathbf{1} \oplus \mathbf{2})$

We have not been able to identify $\mathcal{C}(\mathbf{1} \oplus \mathbf{2})$ satisfactorily. We know it is contained in Co-comparability, Triangle-free, and contains Caterpillars. These containments are proper, as witnessed by C_3, C_5, and C_4, respectively. (C_i is the cycle with i nodes.)

Theorem 8 implies that $\mathcal{C}(\mathbf{1} \oplus \mathbf{2})$ can be characterized as the intersection between Co-comparability and Triangle-free, but it would be interesting to identify this class as some already studied class.

5 Conclusions

We have characterized several important classes of graphs in an uniform way. Every boolean function of two variables originates a class. Such uniform characterization may be useful for the design of algorithms for membership and combinatorial problems that work for all the classes.

The main open problem we leave is to determine whether the class $\mathcal{C}(\mathbf{1} \oplus \mathbf{2})$ coincides with some previously studied class in the literature.

In the future we plan to follow a suggestion of Cláudio Lucchesi and extend the analysis done here to functions of the form $f(\mathbf{1}, \mathbf{2}, \mathbf{3})$, which includes $\mathbf{3} \Rightarrow f(\mathbf{1}, \mathbf{2})$ as special cases.

Acknowledgements

We thank our colleagues at the Dept. of Computer Science for their encouragement and their interest in the ideas expressed here. Jorge

Stolfi, Cláudio Lucchesi, Célia P. de Mello, and Marcus Vinicius P. de Aragão contributed many suggestions and ideas used here. Celina M. Herrera de Figueiredo and Cândido Xavier de Mendonça read an early version and provided valuable comments.

References

[Ben59] S. Benzer. On the topology of the genetic fine structure. *Proceedings of the National Academy of Sciences of the U. S. A.*, 45:1607–1620, 1959.

[Dam92] P. Damaschke. Distances in cocomparability graphs and their powers. *Discrete Math.*, 35:67–72, 1992.

[GH64] P. C. Gilmore and A. J. Hoffman. A characterization of comparability graphs and interval graphs. *Canad. J. Math.*, 16:539–548, 1964.

[Gol80] Martin Charles Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, 1980.

[Jac92] Z. Jackowski. A new characterization of proper interval graphs. *Discrete Math.*, 105:103–109, 1992.

[Mae80] H. Maehara. On time graphs. *Discrete Math.*, 32:281–289, 1980.

[Ola91] Stephen Olariu. An optimal greedy heuristic to color interval graphs. *Information Processing Letters*, 37:21–25, 1991.

[RPR88] G. Ramalingam and C. Pandu Rangan. A unified approach to domination problems on interval graphs. *Information Processing Letters*, 27:271–274, April 1988.

[RTL76] D. J. Rose, R. E. Tarjan, and G. S. Leuker. Algorithmic aspects of vertex elimination on graphs. *SIAM J. Comput.*, 5:266–283, 1976.

[Sim91] K. Simon. A new simple linear algorithm to recognize interval graphs. In H. Bieri and H. Noltemeier, editors, *Computational Geometry – Methods, Algorithms and Applications*, volume 553 of *Lecture Notes in Computer Science*, pages 289–308. Springer-Verlag, 1991.

[Tar83] Robert Endre Tarjan. *Data Structures and Network Algorithms*. Society for Industrial and Applied Mathematics (SIAM), 1983.

Relatórios Técnicos – 1992

01/92 **Applications of Finite Automata Representing Large Vocabularies,** *C. L. Lucchesi, T. Kowaltowski*

02/92 **Point Set Pattern Matching in d-Dimensions,** *P. J. de Rezende, D. T. Lee*

03/92 **On the Irrelevance of Edge Orientations on the Acyclic Directed Two Disjoint Paths Problem,** *C. L. Lucchesi, M. C. M. T. Giglio*

04/92 **A Note on Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams,** *W. Jacometti*

05/92 **An (l, u)-Transversal Theorem for Bipartite Graphs,** *C. L. Lucchesi, D. H. Younger*

06/92 **Implementing Integrity Control in Active Databases,** *C. B. Medeiros, M. J. Andrade*

07/92 **New Experimental Results For Bipartite Matching,** *J. C. Setubal*

08/92 **Maintaining Integrity Constraints across Versions in a Database,** *C. B. Medeiros, G. Jomier, W. Cellary*

09/92 **On Clique-Complete Graphs,** *C. L. Lucchesi, C. P. Mello, J. L. Szwarcfiter*

10/92 **Examples of Informal but Rigorous Correctness Proofs for Tree Traversing Algorithms,** *T. Kowaltowski*

11/92 **Debugging Aids for Statechart-Based Systems,** *V. G. S. Elias, H. Liesenberg*

12/92 **Browsing and Querying in Object-Oriented Databases,** *J. L. de Oliveira, R. de O. Anido*

Relatórios Técnicos – 1993

01/93 **Transforming Statecharts into Reactive Systems,** *Antonio G. Figueiredo Filho, Hans K. E. Liesenberg*

02/93 **The Hierarchical Ring Protocol: An Efficient Scheme for Reading Replicated Data,** *Nabor das C. Mendonça, Ricardo de O. Anido*

03/93 **Matching Algorithms for Bipartite Graphs,** *Herbert A. Baier Saip, Cláudio L. Lucchesi*

04/93 **A lexBFS Algorithm for Proper Interval Graph Recognition,** *Celina M. H. de Figueiredo, João Meidanis, Célia P. de Mello*

05/93 **Sistema Gerenciador de Processamento Cooperativo,** *Ivonne. M. Carrazana, Nelson. C. Machado, Célio. C. Guimarães*

06/93 **Implementação de um Banco de Dados Relacional Dotado de uma Interface Cooperativa,** *Nascif A. Abousalh Neto, Ariadne M. B. R. Carvalho*

07/93 **Estadogramas no Desenvolvimento de Interfaces,** *Fábio N. de Lucena, Hans K. E. Liesenberg*

08/93 **Introspection and Projection in Reasoning about Other Agents,** *Jacques Wainer*

09/93 **Codificação de Seqüências de Imagens com Quantização Vetorial,** *Carlos Antonio Reinaldo Costa, Paulo Lício de Geus*

10/93 **Minimização do Consumo de Energia em um Sistema para Aquisição de Dados Controlado por Microcomputador,** *Paulo Cesar Centoducatte, Nelson Castro Machado*

11/93 **An Implementation Structure for RM-OSI/ISO Transaction Processing Application Contexts,** *Flávio Morais de Assis Silva, Edmundo Roberto Mauro Madeira*

12/93 **Boole's conditions of possible experience and reasoning under uncertainty,** *Pierre Hansen, Brigitte Jaumard, Marcus Poggi de Aragão*

13/93 **Modelling Geographic Information Systems using an Object Oriented Framework,** *Fatima Pires, Claudia Bauzer Medeiros, Ardemiris Barros Silva*

14/93 **Managing Time in Object-Oriented Databases,** *Lincoln M. Oliveira, Claudia Bauzer Medeiros*

15/93 **Using Extended Hierarchical Quorum Consensus to Control Replicated Data: from Traditional Voting to Logical Structures,** *Nabor das Chagas Mendonça, Ricardo de Oliveira Anido*

16/93 **$\mathcal{LL}$ – An Object Oriented Library Language Reference Manual,** *Tomasz Kowaltowski, Evandro Bacarin*

17/93 **Metodologias para Conversão de Esquemas em Sistemas de Bancos de Dados Heterogêneos,** *Ronaldo Lopes de Oliveira, Geovane Cayres Magalhães*

18/93 **Rule Application in GIS – a Case Study,** *Claudia Bauzer Medeiros, Geovane Cayres Magalhães*

19/93 **Modelamento, Simulação e Síntese com VHDL,** *Carlos Geraldo Krüger e Mário Lúcio Côrtes*

20/93 **Reflections on Using Statecharts to Capture Human-Computer Interface Behaviour,** *Fábio Nogueira de Lucena e Hans Liesenberg*

21/93 **Applications of Finite Automata in Debugging Natural Language Vocabularies,** *Tomasz Kowaltowski, Cláudio Leonardo Lucchesi e Jorge Stolfi*

22/93 **Minimization of Binary Automata,** *Tomasz Kowaltowski, Cláudio Leonardo Lucchesi e Jorge Stolfi*

23/93 **Rethinking the DNA Fragment Assembly Problem,** *João Meidanis*

24/93 **EGOLib — Uma Biblioteca Orientada a Objetos Gráficos,** *Eduardo Aguiar Patrocínio, Pedro Jussieu de Rezende*

25/93 **Compreensão de Algoritmos através de Ambientes Dedicados a Animação,** *Rackel Valadares Amorim, Pedro Jussieu de Rezende*

26/93 **GeoLab: An Environment for Development of Algorithms in Computational Geometry,** *Pedro Jussieu de Rezende, Welson R. Jacometti*

Departamento de Ciência da Computação — IMECC
Caixa Postal 6065
Universidade Estadual de Campinas
13081-970 – Campinas – SP
BRASIL

reltec@dcc.unicamp.br