This is an author version of a paper published in: Matemática Contemporânea **46** (2019) pp 185–193
https://mc.sbm.org.br/wp-content/uploads/sites/15/2019/12/P19.pdf

Simulating digital circuits with clique graphs

C. Cedillo*† M.A. Pizaña†

Abstract

The *clique operator* transforms a graph G into its *clique graph* $K(G)$, which is the intersection graph of all the (maximal) cliques of G. Clearly, we can iterate the operator to obtain *iterated clique graphs*: $K^{n+1}(G) = K(K^n(G))$. The clique graph operator and the iterated clique graphs have been studied extensively.

The discrete dynamics generated by the clique operator when applied iteratively to graphs, has shown to be very rich and complex. This complexity has fascinated experts in the field of graph dynamics as well as it has defeated every endeavor to characterize the clique behavior of graphs. Indeed, it has been suspected since 2001 (by J. Meidanis [10]) that the clique operator is actually Turing-complete.

Motivated by this problem we have started simulating digital circuits using clique graphs, with the aim to explore the computational power of the clique operator. Here we report our advances.

1 Introduction

It is well known among Engineers that (many copies of) the gates in a *functionally complete set of logic gates* (for instance: {AND, NOT}, {OR, NOT} or simply {NAND} or {NOR}) is all you need to build an entire digital computer [11]. However, the statement depends on several implicit assumptions that may not be as straightforward as one might think. For instance, the construction of a digital computer also requires: channels, signal carriers, splitters and splices. Also, to construct flip-flops (an essential component of a digital computer, used to store bits of information) and other complex components, we need that the width of a signal pulse is at least long as the commutation time of gates plus the propagation time of the signals in the channels of the internal feedback loops of the components. Using graphs and the dynamics of the clique operator, we have been able to simulate most of these gadgets.

A simulation of *all* the required digital gadgets (not yet achieved), would imply that an entire digital computer could be simulated using the clique operator

*Partially supported by CONACYT, scholarship 397900.
†Partially supported by SEP-CONACYT, grant A1-S-45528.
2000 AMS Subject Classification: 05C76, 05C69 and 05C62.
Keywords: graph theory, graph dynamics, clique graphs, Turing-completeness, decidability.

dynamics, and hence that the clique operator would have at least the computational power of Linearly Bounded Automata (Turing Machines with a finite tape) for the class of finite graphs, and also that the clique operator would have the full computational power of Turing machines for the class of *quasi periodic graphs* (defined later).

Clique graphs and iterated clique graphs have been studied extensively [9,12, 14] and have found applications to Loop Quantum Gravity [13] and to the study of the Fixed-Point Property of partially ordered sets. An important notion in clique graph theory is that of dismantleability: we say that a graph G *is dismantleable to* H *in one step* (written $G \xrightarrow{\#} H$) if G contains an induced subgraph $H_0 \cong H$ such that every vertex in $G \setminus H_0$ is dominated by some vertex of H_0 (i.e. $\forall x \in G \setminus H_0\ \exists y \in H_0, N_G[x] \subseteq N_G[y]$). We shall use this theorem:

Theorem 1.1 *[4] If* $G \xrightarrow{\#} H$ *then* $K(G) \xrightarrow{\#} K(H)$.

2 First encoding

Our basic building block is shown in Fig. 1(a). It contains dominated vertices, which is undesirable for our purposes, since dominated vertices tend to disappear when we apply the clique operator; hence we add some extra vertices as in Fig. 1(b) whenever necessary to avoid that.

Using several basic building blocks, we can construct *channels* as in Fig. 1(c). Within channels, some local perturbations, which we call *photons*, move when we apply the clique operator: Indeed, the top graph in Fig. 1(d) is the original channel with such a local perturbation (a photon), and the two graphs below are obtained by applying K^2 and K^4 respectively. Here we use even powers of K because our basic constructions are (almost) always K^2-invariant but not K-invariant and hence it is easier to see what happens on even powers of K since then the only things that change are the local perturbations.

This kind of "traveling" behavior was first observed by Escalante in [3] and further studied in [6–8] among others. In the light of the theory already developed, it should be clear that the behavior under the clique operator of the example shown in Fig. 1(d) is exactly the one just described in the previous paragraph. Alternatively, a computer can be used to verify the claim (we use GAP+YAGS [1,5]).

Now we can propose the *first encoding*: The presence of a photon in a channel encodes a 'one' and the absence of a photon encodes a 'zero'. With this first encoding, Fig. 1(g) shows a simulation of an `OR` gate, where each successive graph is obtained from the previous one by applying the operator K^2. Many of the required gadgets can also be simulated:

Theorem 2.1 *Using clique graphs and the first encoding, we can simulate channels, signal carriers (photons),* `OR`*-gates, splitters and splices.*

The claim in Theorem 2.1 is a computational fact and we think that it is best verified by computer; to that end, we have prepared supplementary media

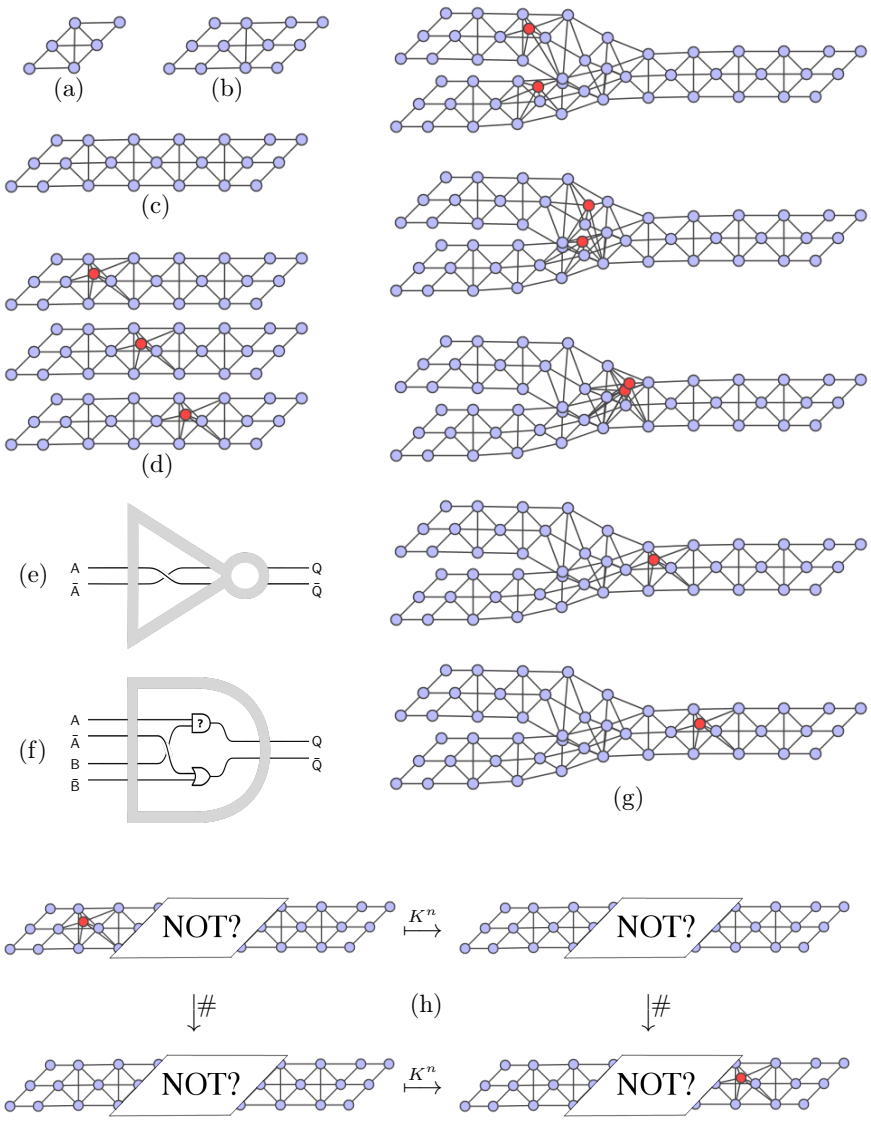

FIGURE 1. (a) and (b) Basic building blocks. (c) Channel. (d) Traveling photon. (e) Second-encoding `NOT` gate. (f) Hypothetical second-encoding `AND` gate. (g) First-encoding `OR` gate. (h) Inexistence of the first-encoding `NOT` gate.

for this paper that can be found in a web page here: [2]. The supplementary media contains code and data and an interactive web page to let the reader explore the gadgets mentioned in this paper. A theoretical explanation is also possible: The main consideration is that, save for the photons (and the dirty-

AND gate mentioned below), our gadgets are all clique-Helly graphs without dominated vertices, and Escalante proved that such graphs are all K^2-invariant [3]. Then, the problem is reduced to analyzing what happen near the introduced local perturbations.

We also have a gadget that we call a dirty-AND gate (see [2]). It does perform the AND function but also generates a lot of undesirable side effects (hence the name). It cannot be used in simulations of digital circuits as it is, but it gives a hint on what to study next to obtain a clean AND gate: Namely, further study is necessary to understand the exact phenomenon by which the dirty-AND gate is producing the photon required by the AND function, and then to try to isolate this behavior to produce the photon without the side effects.

However, the much needed NOT gate is not achievable as a *modular* gate. By *modular* we mean a gate that is a finite graph, which is not clique-divergent (i.e. that $|K^n(G)|$ stays bounded by a constant for all n), and whose internal structure does not depend on the surrounding circuit. The modularity condition is needed for Theorem 2.2:

Theorem 2.2 *A modular* NOT *gate cannot be simulated with clique graphs using the first encoding.*

Proof: Assume we have a modular NOT gate, we shall treat it as a black box, but it surely must have an input channel and an output channel as in Fig. 1(h).

Now, let H_1, H_2, H_3 and H_4 the graphs depicted in Fig. 1(h) (from top-to-bottom and left-to-right). Then H_1 and H_3 represent the NOT gate receiving a 1 and a 0 (respectively), which after a finite number of iterations of the clique operator, must then be transformed into graphs H_2 and H_4 (i.e. $K^n(H_1) = H_2$ and $K^n(H_3) = H_4$). But the photon in H_1 is a dominated vertex of H_1 and hence (since the gate is modular) H_1 is dismantleable to H_3 in one step, in symbols $H_1 \xrightarrow{\#} H_3$.

By Theorem 1.1, we have that $H_2 = K^n(H_1) \xrightarrow{\#} K^n(H_3) = H_4$ which, by definition, implies that H_2 contains a subgraph H_4' isomorphic to H_4.

Suppose first that the isomorphism from H_4' to H_4 sends the output channel of H_4' onto the output channel of H_4. Then we get a contradiction since no vertex of H_4' could possibly be mapped onto the outgoing photon in H_4.

Now, suppose the contrary: that the isomorphism between H_4' and H_4 does not preserve the output channel. This could happen, in principle, if some photon within the internal structure of the NOT gate in H_4' is the one who is mapped onto the outgoing photon in H_4. However, our NOT gate is modular by hypothesis, and hence its internal structure cannot depend on the surrounding circuit. Since we can attach any desired circuit to the output channel of the NOT gate (for instance, an output channel of any desired length) and the NOT gate must deliver the isomorphism between H_4' and H_4 in each and every case, it follows that the NOT gate must either (1) *contain* a preexisting internal copy of every possible circuit attached to the output channel (which contradicts the finiteness assumption of modularity) or (2) *construct* an internal copy of every possible circuit attached to

the output channel (which contradicts the non clique-divergence assumption of modularity). It follows that the isomorphism between H_4' and H_4 must preserve the output channel as in the previous paragraph and hence that the modular NOT gate does not exist. □

3 Second encoding

In view of Theorem 2.2 a second encoding is necessary, but we shall reuse the results of the previous encoding. Now we shall use dual channels (each having a top channel and a bottom channel) to transport information and we shall encode a 'one' with a dual channel having a photon in the top channel but not in the bottom channel and a 'zero' with a dual channel having a photon in the bottom channel but not in the top channel (the other two combinations are considered invalid). This way, a NOT gate can be simulated by a simple exchange in its channels (see Fig. 1(e)). Many of the previous gadgets can be readily reimplemented with this second encoding by simply making copies (the top copy and the bottom copy) of the gadgets already found for the first

Theorem 3.1 *Using clique graphs and the second encoding, we can simulate channels, signal carriers, splitters, splices and* NOT *gates.*

From now on, we really need a first-encoding AND gate (but we only have a dirty-AND gate), hence our next results depend on the existence of that gate. For Theorem 3.2, Fig. 1(f) fully explains how to construct a second-encoding AND gate using first-encoding AND and OR gates.

Theorem 3.2 *If an* AND *gate can be simulated with clique graphs in the first encoding then an* AND *gate can also be simulated in the second encoding.*

With a functionally complete set of gates (in this case $\{\texttt{AND}, \texttt{NOT}\}$), if we can use infinite graphs, any Turing machine can be simulated. We do not need a huge class of infinite graphs for the simulation (the class of numerable graphs has the cardinality of the continuum), it is sufficient to use *quasi periodic graphs* i.e. a finite graph that is repeated a countable number of times where some vertices in the i-th copy may be identified with some vertices in the $(i+1)$-th copy in a periodic manner, with just a finite number of additional vertices and edges (quasi periodic graphs can be finitely represented and hence the class of quasi-periodic graphs is numerable).

Theorem 3.3 *If an* AND *gate can be simulated in the first encoding, then the clique graph operator is Turing-complete for (infinite but) quasi periodic graphs.*

If we restrict ourselves to *finite* graphs and assume that the first-encoding AND gate exists, it should be clear that the same techniques can be used to simulate a finite memory computer (e.g. a *real* computer) using only finite graphs. In particular, Linearly Bounded Automata (LBA) could be simulated using finite clique graphs if the first-encoding AND gate exists.

Acknowledgment We thank the anonymous referees who helped us improve the presentation of this paper and in particular, the one who asked whether our techniques could be used to prove LBA-completeness in the case of finite graphs, thus motivating the inclusion of that discussion here.

References

[1] C. Cedillo, R. MacKinney-Romero, M.A. Pizaña, I.A. Robles and R. Villarroel-Flores. *YAGS - Yet Another Graph System, Version 0.0.3*, 2016. `http://xamanek.izt.uam.mx/yags/`.

[2] C. Cedillo and M.A. Pizaña. *Simulating digital circuits with clique graphs (supplementary media).* Available (2019) at : `http://xamanek.izt.uam.mx/map/gadgets/`.

[3] F. Escalante. *Über iterierte Clique-Graphen.* Abh. Math. Sem. Univ. Hamburg **39** (1973) 59–68.

[4] M.E. Frías-Armenta, V. Neumann-Lara and M.A. Pizaña. *Dismantlings and iterated clique graphs.* Discrete Math. **282** (2004) 263–265.

[5] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.3*, 2002. `(http://www.gap-system.org)`.

[6] F. Larrión and V. Neumann-Lara. *A family of clique divergent graphs with linear growth.* Graphs Combin. **13** (1997) 263–266.

[7] F. Larrión and V. Neumann-Lara. *On clique-divergent graphs with linear growth.* Discrete Math. **245** (2002) 139–153.

[8] F. Larrión, V. Neumann-Lara and M.A. Pizaña. *Clique divergent clockwork graphs and partial orders.* Discrete Appl. Math. **141** (2004) 195–207.

[9] F. Larrión, M.A. Pizaña and R. Villarroel-Flores. *On self-clique shoal graphs.* Discrete Applied Mathematics **205** (2016) 86 – 100.

[10] J. Meidanis. *The clique operator.* Available (2001) at : `http://www.ic.unicamp.br/~meidanis/research/clique/`.

[11] M. Morris Mano. *Digital design.* Prentice-Hall, 1984.

[12] E. Prisner. *Graph dynamics.* Longman, Harlow, 1995.

[13] M. Requardt. *(Quantum) spacetime as a statistical geometry of lumps in random networks.* Classical Quantum Gravity **17** (2000) 2029–2057.

[14] J.L. Szwarcfiter. A survey on clique graphs. In B.A. Reed and C. Linhares-Sales, editors, *Recent advances in algorithms and combinatorics*, volume 11 of *CMS Books Math./Ouvrages Math. SMC*, pages 109–136. Springer, New York, 2003.

C. Cedillo
Departamento de Ingeniería Eléctrica
Universidad Autónoma Metropolitana
Mexico City, Mexico.

Centro Universitario UAEM Nezahualcóyotl,
Nezahualcoyotl City, Mexico.
`mc.cedilloc@gmail.com`
`mdcedilloc@uaemex.mx`

M. Pizaña
Departamento de Ingeniería Eléctrica
Universidad Autónoma Metropolitana
Mexico City, Mexico.
`map@xanum.uam.mx`