UNIVERSITY OF BIRMINGHAM
University of Birmingham Research at Birmingham

Beyond formulas-as-cographs

Calk, Cameron; Das, Anupam; Waring, Tim

Citation for published version (Harvard):
Calk, C, Das, A & Waring, T 2020 'Beyond formulas-as-cographs: an extension of Boolean logic to arbitrary graphs'.

Link to publication on Research at Birmingham portal

General rights
Unless a licence is specified above, all rights (including copyright and moral rights) in this document are retained by the authors and/or the copyright holders. The express permission of the copyright holder must be obtained for any use of this material other than for purposes permitted by law.

•Users may freely distribute the URL that is used to identify this publication.
•Users may download and/or print one copy of the publication from the University of Birmingham research portal for the purpose of private study or non-commercial research.
•User may use extracts from the document in line with the concept of 'fair dealing' under the Copyright, Designs and Patents Act 1988 (?)
•Users may not further distribute the material nor use it for the purposes of commercial gain.

Where a licence is displayed above, please note the terms and conditions of the licence govern your use of this document.

When citing, please reference the published version.

Take down policy
While the University of Birmingham exercises care and attention in making items available there are rare occasions when an item has been uploaded in error or has been deemed to be commercially or otherwise sensitive.

If you believe that this is the case for this document, please contact UBIRA@lists.bham.ac.uk providing details and we will remove access to the work immediately and investigate.

Download date: 20. Apr. 2024

arXiv:2004.12941v1 [cs.LO] 27 Apr 2020

BEYOND FORMULAS-AS-COGRAPHS: AN EXTENSION OF BOOLEAN LOGIC TO ARBITRARY GRAPHS

CAMERON CALK[1], ANUPAM DAS[2], AND TIM WARING[3]

Abstract. We propose a graph-based extension of Boolean logic called Boolean Graph Logic (BGL). Construing formula trees as the cotrees of cographs, we may state semantic notions such as evaluation and entailment in purely graph-theoretic terms, whence we recover the definition of BGL. Naturally, it is conservative over usual Boolean logic.

Our contributions are the following:

(1) We give a natural semantics of BGL based on Boolean relations, i.e. it is a multivalued semantics, and show the adequacy of this semantics for the corresponding notions of entailment.
(2) We show that the complexity of evaluation is **NP**-complete for arbitrary graphs (as opposed to **ALOGTIME**-complete for formulas), while entailment is Π_2^p-complete (as opposed to **coNP**-complete for formulas).
(3) We give a 'recursive' algorithm for evaluation by induction the modular decomposition of graphs. (Though this is not polynomial-time, cf. point (2) above).
(4) We characterise evaluation in a game-theoretic setting, in terms of both static and sequential strategies, extending the classical notion of positional game forms beyond cographs.
(5) We give an axiomatisation of BGL, inspired by deep-inference proof theory, and show soundness and completeness for the corresponding notions of entailment.

One particular feature of the graph-theoretic setting is that it escapes certain no-go theorems such as a recent result of Das and Strassburger [DS15, DS16], that there is no linear axiomatisation of the linear fragment of Boolean logic (equivalently the multiplicative fragment of Japaridze's Computability Logic or Blass Game Semantics for Multiplicative Linear Logic).

1. Introduction

Boolean logic, a.k.a classical propostional logic, lies at the heart of multiple areas, including algebra, proof theory and computational complexity. Axiomatisations and proof systems for this logic generally manipulate Boolean formulas, built from $\neg, \vee, \wedge$ or some other adequate basis of connectives. These include the classical Hilbert-Frege style axiomatic systems and Gentzen style sequent and natural deduction systems, as well as more recent methodologies such as Belnap's display logic, cf. [Jr.82], and Guglielmi's deep inference, cf. [Gug15]. The latter, along with Girard's linear logic [Gir87], has also been responsible for a more graph-based viewpoint on proof theory, for instance via proof nets [Gir87], expansion proofs [Mil84], atomic flows [GG08] and combinatorial proofs [Hug06]. Nonetheless such systems

[1]Laboratoire d'Informatique de l'École Polytechnique, Palaiseau, 91120, France.
[2]University of Birmingham, Birmingham, B15 2TT, UK.
[3]University of Copenhagen, Copenhagen, 2100, Denmark.
Date: April 28, 2020.

still fundamentally deal with the same underlying structure, formulas, with various annotations/decorations.

One viewpoint of formulas is to construe formula trees as the cotrees of cographs, leading to the notion of 'co-occurrence graph' (in Boolean function theory, e.g. [CH11b]) or 'relation web' (in structural proof theory, e.g. [Gug07]). This viewpoint has been exploited several times in order to prove fundamental properties of Boolean logic. A notable result is seminal theorem of Gurvich:

Theorem ([Gur77])**.** *A Boolean function f is computed by a read-once formula if and only if every minterm and maxterm of f has singleton intersection.*

More recently, one the current authors, Das, proved with Strassburger that the linear fragment of Boolean logic admits no polynomial-time axiomatisation (unless **coNP** = **NP**), crucially exploiting the graph-theoretic viewpoint [DS15, DS16]. This result is important since it immediately implies the non-axiomatisability of multiplicative fragments of Japaridze's Computability logic, cf. [Jap05], and Blass' game semantics for linear logic [Bla92], resolving a problem that was open since the '90s (see [Jap17]).

One feature of the graph-theoretic setting is that semantic notions such as evaluation and entailment may be framed in purely graph-theoretic terms, e.g. as exploited in [Gur77, DS16]. In particular, these characterisations are meaningful for all graphs, not just the cographs that correspond to formulas. In this work we further develop this idea into a bona fide logical system, which we call 'Boolean Graph Logic' (BGL). We establish various fundamental properties of BGL, from the viewpoints of complexity theory, game theory and proof theory, using graph theoretic tools throughout. BGL is conservative over usual Boolean logic (as expected), and semantically it corresponds to a natural extension to Boolean relations rather than Boolean functions. One particular feature of BGL is that entailment is much more fine-grained, admitting interpolations that are impossible in usual Boolean logic. An example of this is given in Section 3.3, where a minimal 10 variable linear inference from [Das13] is decomposable in BGL.

At least one motivation of this work is the issue of (linearly) axiomatising the linear fragment of Boolean logic. The impossibility result of [DS15, DS16] crucially exploited the fact that the graphs corresponding to formulas are P_4-free (i.e. cographs). In particular, their critical Lemma 5.8 does not scale to arbitrary graphs. It is natural to ask whether this impossibility result can be extended to BGl or whether BGL might admit a linear axiomatisation. One way of looking at this motivation is with the following question: can we establish a Boolean proof theory without any structural behaviour, such as duplication and erasure? A positive answer would be complementary to the established orthodoxy in structural proof theory.

Our contributions are the following:

(1) We give a natural semantics of BGL based on Boolean relations, i.e. it is a multivalued semantics, and show the adequacy of this semantics for the corresponding notions of entailment.
(2) We show that the complexity of evaluation is **NP**-complete for arbitrary graphs (as opposed to **ALOGTIME**-complete for formulas), while entailment is Π_2^p-complete (as opposed to **coNP**-complete for formulas).

(3) We give a 'recursive' algorithm for evaluation by induction the modular decomposition of graphs. (Though this is not polynomial-time, cf. point (2) above).
(4) We characterise evaluation in a game-theoretic setting, in terms of both static and sequential strategies, extending the classical notion of positional game forms beyond cographs.
(5) We give an axiomatisation of BGL, inspired by deep-inference proof theory, and show soundness and completeness for the corresponding notions of entailment.

1.1. **History and related work.** The ideas behind the Boolean Graph Logic project were already mentioned in the aforementioned paper [DS16], where Section 9 proposed the basic notions of entailment as the possible basis of a proof theory on arbitrary graphs. The research behind BGL properly began in 2016 when the author Calk conducted his Bachelor's thesis project under the supervision of the author Das [Cal16]. This comprised mainly of the results from Sections 3 and 5. The results of Section 4 were established by Das and have been circulated in private correspondence [Das19]. The author Waring conducted his Master's thesis under the supervision of Das in 2019 [War19], comprising of the results of Sections 6 and parts of 3 and 7, as well as supplementary material found in the appendix.

In parallel other groups of authors have become interested in the prospect of logic and proof theory based on arbitrary graphs. In particular, Acclavio, Horne and Strassburger have investigated a structural proof theory of arbitrary graphs, from a linear logic perspective [AHS20]. They obtain a cut-elimination result for a conservative extension of multiplicative linear logic, via the 'splitting' method of deep inference proof theory. Their syntax of graphs is identical to ours, only interpreting conjunction and disjunction by their multiplicative variants. However their logic and BGL seem to be incomparable, at the level of validity, distinguishing these two graph settings from their restriction to formulas. It would be interesting to establish a graph level semantics of their logic to more fully understand the differences between these two approaches; relevant work in this direction includes recent work of Seiller and Nguyen, who developed a model of multiplicative linear logic with a form of nondeterminism via 'interaction graphs' [NS18].

Beyond structural proof theory, there are several systems that operate with objects other than formulas. These include circuit-based systems (e.g. [Jap05] and [Jeř04]), algebraic systems (e.g. [BIK$^+$94] and [CCT87]) and systems operating with forms of decision diagrams (e.g. [AKV04] and [BDK20]). Of these, algebraic systems are particularly interesting (and thematically relevant to this work) since they extend the usual Boolean semantics conservatively to an arithmetic setting.

1.2. **Perspectives on Boolean Graph Logic.** Given that this article collects various research carried out by the three authors, it is natural that different readers will take interest in different parts. We give the mutual dependencies between sections in Figure 1, and we give some possible readings of this article below:

- Complexity theoretic viewpoint: Sections 2, 3 and 4.
- Game theoretic viewpoint: Sections 2, 3, 5 and 6.
- Proof theoretic viewpoint: Sections 2, 3, 5 and 7.

It is our intention that this article be an introduction to BGL and serve as a reference for later research.

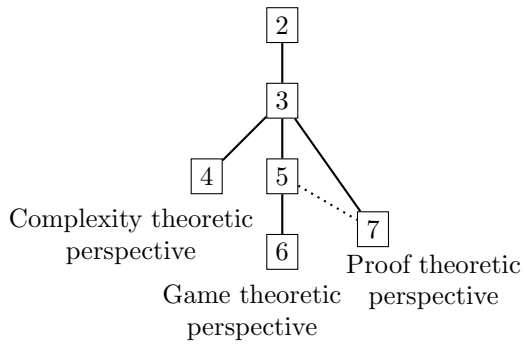


FIGURE 1. Dependencies of sections in this paper, and the viewpoints of Boolean Graph Logic given by different readings of the paper. For Section 7, familiarity with the content of Section 5 is helpful but not strictly necessary.

2. Preliminaries

Throughout this work let us fix a set $\mathcal{V}$ of *variables* that will be used in various contexts, e.g. nodes of a graph, Boolean variables etc. This overloading is intentional as we will often identify these objects later on. We write $\binom{X}{2}$ for the set of unordered pairs of a set X.

2.1. **Graphs.** In this work we deal with undirected finite graphs that are simple and loopless. We specify a graph G by a pair (V, E) such that V is a finite set of *nodes* or *vertices*, typically a subset of $\mathcal{V}$, and $E \subseteq \binom{V}{2}$ the set of *edges*. The *dual* or *complement* $\overline{G}$ of a graph $G = (V, E)$ is $\left(V, \binom{V}{2} \setminus E\right)$. The *size* of a graph G, written $|G|$, is just its number of vertices, i.e. $|V|$.

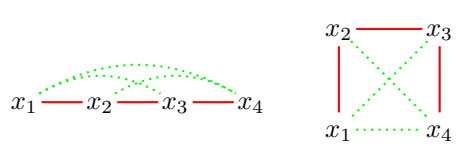
We will use standard graph-theoretic terminology, e.g. neighbours, adjacency, cliques, stable sets, connected etc. Sometimes we will write $V(G)$ (or $E(G)$) to denote the set of vertices (respectively edges) of a graph G when it is not completely specified.

For convenience, we will often (but not always) draw graphs with red solid lines to indicate edges and green dotted lines to indicate non-edges, to allow for incomplete information to be presented. I.e. we may say "x —— y *in* G" to denote that $\{x, y\} \in E(G)$ and "x ····· y *in* G" to denote that $\{x, y\} \notin E(G)$.

Example 1 (Paths, cycles and complete graphs)**.** *Let* $V_n = \{x_1, \ldots, x_n\}$*. We will use the following graphs throughout this work.*

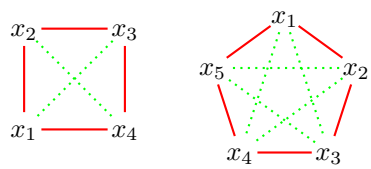
- $P_n := (V_n, \{\{x_1, x_2\}, \ldots, \{x_{n-1}, x_n\}\})$ *is the* n-path*. For instance, here are some (equivalent) representations of the* 4*-path,* P_4*:*

(1)

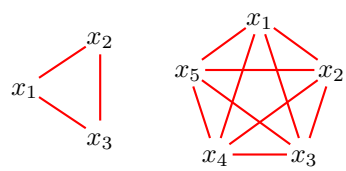


- $C_n := P_n \cup \{x_n, x_1\}$ *is the* n-cycle. *For instance, here are the* 4*-cycle* C_4 *and* 5*-cycle* C_5*, respectively:*

(2)



- $K_n := \left(V_n, \binom{V_n}{2}\right)$ *is the* n-complete-graph. *For instance here are the graphs* K_3 *and* K_5*, respectively:*



Definition 2 (Subgraphs)**.** *A graph* G *is a* subgraph *of a graph* H*, written* $G \leq H$*, if* $V(G) \subseteq V(H)$ *and* $E(G) \subseteq E(H)$*. For* $X \subseteq V(G)$ *we say that* $G \restriction X := \left(X, \binom{X}{2} \cap E(G)\right)$ *is the subgraph of* G induced *by* X*. We will sometimes simply write* X *instead of* $G \restriction X$*, as an abuse of notation, when the ambient graph* G *is clear from context.*

Definition 3 (Homomorphisms and freeness)**.** *For graphs* G *and* H*, we say that* $\varphi : V(G) \to V(H)$ *is a* homomorphism *if* $\{x, y\} \in E(G)$ *implies that* $\{\varphi(x), \varphi(y)\} \in E(H)$*, or* $\varphi(x) = \varphi(y)$.[1] *If* φ *is a bijection then we say it is an* isomorphism.

We say that G *is* H-free *if no induced subgraph of* G *is isomorphic to* H*.*

Remark 4 (Variants of homomorphisms)**.** *It would have been quite natural for us to consider homomorphisms that further satisfy the following dual property: if* $\{\varphi(x), \varphi(y)\} \notin V(H)$ *then* $\{x, y\} \notin V(G)$*, i.e.* φ reflects *green edges. This is pertinent because of the notion of disjunctive entailment* $\underset{\vee}{\Rightarrow}$ *we introduce in the next section, but we do not develop this notion of homomorphism formally.*

We could rather consider homomorphisms that preserve more structure, for instance 'preserving green edges' too. Such a property is satisfied by the 'quotient homomorphisms' we discuss in Section 5. More detailed investigations studying the links between logical phenomena and classes of homomorphisms have been carried out in [RS19].

We will often speak about graphs up to isomorphism, when there is no ambiguity.

Example 5 (Cographs)**.** *A* cograph *is a graph for which every induced subgraph is either disconnected or its complement is disconnected.*

For instance, the P_4 *(cf.* (1)*) is not a cograph, but the* C_4 *is (cf.* (2)*). Notice that the* C_5 *is not a cograph since it has a* P_4 *as an induced subgraph. This means that the* C_5 *is not* P_4*-free.*

We have the following well-known characterisation of cographs, immediately yielding an efficient algorithm for cograph-recognition.

[1] This caveat is because we deal with loopless graphs. However note that this convention does not affect the notions of clique, stable set and connectedness that we rely on throughout this work.

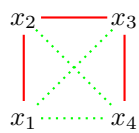
Fact 6. *G is a cograph if and only if it is P_4-free.*

Finally, we introduce some topological aspects of graphs that we use throughout.

Definition 7 (Cliques, stable sets and connectedness)**.** *For a graph G, a subset of its nodes $X \subseteq V(G)$ is a* clique *if $G{\restriction}X$ is isomorphic to some K_n. If $G{\restriction}X$ is isomorphic to some $\overline{K}_n$ then we say that X is a* stable set*. We say that a clique (stable set) is* maximal *in G if it cannot be extended to a larger clique (respectively stable set). We write $MC(G)$ ($MS(G)$) for the set of maximal cliques (respectively stable sets) of G.*

A graph G is connected *if, for every $x, y \in V(G)$, there is an induced subgraph isomorphic to some P_n containing x and y. Otherwise G is* disconnected*. G is* co-connected *if, for every $x, y \in G$, there is an induced subgraph isomorphic to some $\overline{P}_n$ containing x and y. Otherwise G is* co-disconnected*.*

Note, in particular, that a graph may be both connected and coconnected, in particular the P_4:



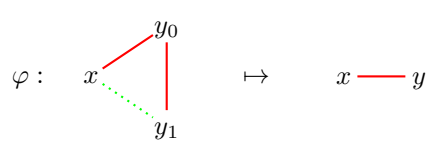
Notice that co-connectedness can be seen visually as 'green' connectedness, whereas connectedness is 'red' connectedness.

The following result is immediate from the definitions of homomorphisms and cliques:

Observation 8 (Homomorphisms preserve cliques)**.** *Let $\varphi : V(G) \to V(H)$ be a homomorphism from a graph G to a graph H. Then, if S is a clique in G then $\varphi(S)$ is a clique in H.*

Following on from Remark 4, for homomorphisms that reflect green edges we have a dual result to that above: if T is stable in H then $\varphi^{-1}(T)$ is stable in G. On the other hand if green edges are preserved then so are stable sets.

Notice, however, that maximality of a clique is not preserved by a homomorphism. A very simple example is given by the map below:



by the homomorphism $\{x \mapsto x, y_0 \mapsto y, y_1 \mapsto y\}$. Here the maximal clique $\{y_0, y_1\}$ on the LHS is mapped to the clique y on the RHS, but this is clearly not maximal. This example also shows that, a priori, stable sets are not reflected by homomorphisms: $\{y\}$ is stable on the RHS but its preimage $\{y_0, y_1\}$ is clearly not stable on the LHS.

2.2. **Boolean logic.** We will consider *positive Boolean formulas* (henceforth simply *formulas*), built as follows:

- Any variable $x \in \mathcal{V}$ is a formula;
- If A and B are formulas then so is $(A \vee B)$;
- If A and B are formulas then so is $(A \wedge B)$.

We call a formula *linear* or *read-once* if each variable occurs at most once in it.[2] Formulas compute Boolean functions in the usual way. We identify Boolean assignments $\mathcal{V} \to \{0,1\}$, with subsets of $\mathcal{V}$ as expected: $\alpha : \mathcal{V} \to \{0,1\}$ is identified with the subset $\{x \in \mathcal{V} : \alpha(x) = 1\} \in \mathcal{P}(\mathcal{V})$. In this way, a Boolean function is a map $\mathcal{P}(\mathcal{V}) \to \{0,1\}$. Note that, throughout this work, we assume that the support of assignments is finite, so that only finite subsets of variables need be considered.

Formally, for a formula A, we write $A(X)$ for its Boolean output on input X, defined recursively as follows:

$$\begin{aligned} x(X) &:= \begin{cases} 1 & x \in X \\ 0 & \text{otherwise} \end{cases} \\ (A \vee B)(X) &:= \max(A(X), B(X)) \\ (A \wedge B)(X) &:= \min(A(X), B(X)) \end{aligned}$$

In our setting, where we do not admit negation, the functions computed by formulas are *monotone*, i.e. if $X \subseteq Y$ then $A(X) \leq A(Y)$. For us this is sufficiently general since any Boolean tautology $A(x_1, \ldots, x_n, \overline{x}_1, \ldots \overline{x}_n)$, with all literals displayed, is equivalent to a monotone implication $\bigwedge_{i=1}^{n} (x_i \vee x'_i) \Rightarrow A(\vec{x}, \vec{x}')$. Note that if A is read-once then the resulting monotone implication is a linear inference, i.e. both the LHS and RHS are read-once on the same variables.

More generally, we may speak of arbitrary Boolean functions $f : \mathcal{P}(\mathcal{V}) \to \{0,1\}$ being monotone if $X \subseteq Y \implies f(X) \leq f(Y)$. Again, since we are only considering finite Boolean functions, such functions f are determined by their actions on some finite set of variables $V_n = \{x_1, \ldots, x_n\}$ that they 'depend on'.

We introduce now a standard semantic abstraction of monotone Boolean functions, which we will later link to graph-theoretic notions.

Definition 9 (Minterms and maxterms)**.** *Let $f : \mathcal{P}(\mathcal{V}) \to \{0,1\}$ be a monotone Boolean function.*

- *A* minterm *of f is a minimal set $X \subseteq \mathcal{V}$ such that $f(X) = 1$.*
- *Dually, a* maxterm *of f is a minimal set X such that $f(\overline{X}) = 0$.*

Since we identify formulas with the monotone Boolean functions they compute, we may also speak of the minterms and maxterms of a formula.

Notice that minterms correspond to the terms of the irredundant disjunctive normal form of a function, while maxterms correspond to the clauses of the irredundant conjunctive normal form of a function (see, e.g., [CH11a]). Consequently we have the following (well-known) characterisations of evaluation and entailment in terms of their minterms and maxterms:

Proposition 10 (Characterisations of evaluation)**.** *For a monotone Boolean function $f : \mathcal{P}(\mathcal{V}) \to \{0,1\}$, we have the following:*

(1) $f(X) = 1$ if and only if there is a minterm S of f such that $S \subseteq X$.
(2) $f(X) = 0$ if and only if there is a maxterm T of f such that $T \cap X = \varnothing$.

Proposition 11 (Characterisations of entailment)**.** *For a monotone Boolean function $f : \mathcal{P}(\mathcal{V}) \to \{0,1\}$, the following are equivalent*

(1) $f \leq g$, i.e. if $f(X) = 1$ then $g(X) = 1$.

[2]The nomenclature 'linear' comes from term rewriting, e.g. [Ter03], where as 'read-once' comes from complexity theory and Boolean function theory, e.g. [CH11a].

(2) For each minterm S of f there is a minterm S' of g such that $S' \subseteq S$.
(3) For each maxterm T of g there is a maxterm T' of f such that $T' \subseteq T$.

Both results are folklore, but proofs may be found in, e.g., [DS16], along with several examples. Naturally, the book [CH11a] also constitutes good reference material.

2.3. **Relation webs.** Now we introduce a useful abstraction of Boolean formulas, that relates them (and their semantics) to the graph theoretic notions we introduced earlier.

Definition 12 (Least common connectives and relation webs)**.** *For a linear formula A containing (distinct) variables x and y, the* least common connective *(lcc) of x and y in A is the main connective, $\vee$ or $\wedge$, of the smallest subformula of A containing both x and y.*

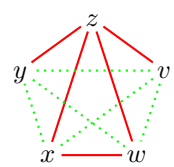
The relation web*, or simply* web*, of A, written $\mathcal{W}(A)$, is the graph defined as follows:*

- *The nodes of $\mathcal{W}(A)$ are the variables of A.*
- *The edges of $\mathcal{W}(A)$ are those $\{x, y\}$ where the lcc of x and y is $\wedge$ in A.*

Relation webs are also known as 'co-occurrence' graphs, though we follow the nomenclature from structural proof theory here.

Example 13. *The formula $((v \vee (w \wedge x)) \vee y) \wedge z$ has the following relation web:*

(3)



Notice that we can also see the web as drawing a 'red edge' between variables whose lcc is $\wedge$ and a 'green edge' between variables whose lcc is $\vee$.

The graph above is also the relation web of, e.g., $z \wedge (((x \wedge w) \vee y) \vee v)$.

In fact, webs represent precisely the quotients of linear formulas modulo associativity and commutativity of $\vee$ and $\wedge$:

Fact 14. *$\mathcal{W}(A) = \mathcal{W}(B)$ if and only if A and B are equivalent modulo the following equational theory:*[3]

$$\begin{array}{lll} A \vee B & =_{AC} & B \vee A \\ A \wedge B & =_{AC} & B \wedge A \end{array} \qquad \begin{array}{lll} A \vee (B \vee C) & =_{AC} & (A \vee B) \vee C \\ A \wedge (B \wedge C) & =_{AC} & (A \wedge B) \wedge C \end{array}$$

This fact is well-known and may be proved by induction on the length of a $=_{AC}$-derivation. The graph classification of relation webs is well-known:

Fact 15. *Any relation web is a cograph. Conversely, any cograph (with nodes in $\mathcal{V}$) is the web of some linear formula.*

This is readily proved by induction on the size of graphs, using either the underlying formula structure of a web or the fact that cograph-ness is closed under taking induced subgraphs.

[3] As usual, equational theories operate under 'deep inference', i.e. if $A =_{AC} A'$ then $B[A] =_{AC} B[A']$ for any formula context $B[\cdot]$.

Notice that the above fact also gives us another characterisation of relation webs in terms of forbidden induced subgraphs:

Corollary 16 (of Proposition 6)**.** *Any relation web is a P_4-free. Conversely, any P_4-free graph (with nodes in $\mathcal{V}$) is the web of some linear formula.*

This alternative characterisation makes it abundantly clear that the relation webs really only span a small subset of all possible graphs: almost all graphs contain P_4s, in a standard sense, by a simple counting argument:

Remark 17 (Counting cographs)**.** *Appealing to the probabilistic method, note that the uniform distribution on graphs (of fixed size) is induced by independently assigning edges to pairs of variables with probability $\frac{1}{2}$. In this case, for any fixed four nodes, the chance that they do not form a P_4 is some fixed $\varepsilon < 1$. Thus for a graph with, say, $4n$ nodes, the chance there is no P_4 is bounded above by ε^n by partitioning the nodes into sets of 4 (since P_4-ness of disjoint sets are independent events). Therefore the class of cographs, as we increase the number of nodes, is sparse in the class of all graphs.*

As we have mentioned the point of this work is to study an extension of the notion of web to arbitrary graphs. In order to do so we will need characterisations of logical concepts in terms of graphs. Namely, we have the following results from [DS15, DS16]:

Proposition 18 (Characterisation of maximal cliques and stables sets of webs)**.** *We have the following, for any linear formula A:*

(1) S is a minterm of A if and only if it is a maximal clique of $\mathcal{W}(A)$.
(2) T is a maxterm of A if and only if it is a maximal stable set of $\mathcal{W}(A)$.

This result is proved by a routine induction on the structure of the formula A. This gives us purely graph theoretic characterisations of evaluation and entailment, thanks to Propositions 10 and 11, which will induce the graph logic we define in the next section. Notice also that this means that distinct webs correspond to distinct Boolean functions, since two webs are the same just if they have the same maximal cliques and stable sets, just if they have the same minterms and maxterms, just if they compute the same Boolean function. Since we also have that webs quotient formulas exactly by associativity and commutativity of $\vee$ and $\wedge$, cf. Fact 14, we also have the following well-known result:

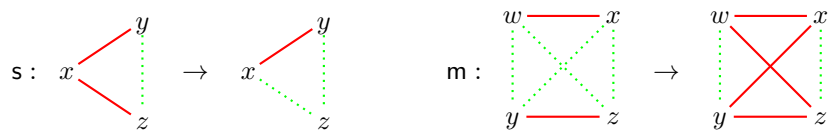
Corollary 19 (E.g., [DS16])**.** *Read-once/linear formulas compute the same Boolean function if and only if they are equivalent modulo $=_{AC}$.*

Example 20. *Revisiting Example 13, notice that the two formulas whose web is* (3) *are equivalent modulo $=_{AC}$, so let us write f for the Boolean function they compute. Notice that $X = \{w, y, z\}$ contains the maximal clique $\{y, z\}$, and so $f(X) = 1$. Dually, $Y = \{v, w, y\}$ is disjoint from the stable set $\{z\}$, and so $f(Y) = 0$. Both of these facts are readily verifiable by computing the output of f using the formulas from Example 13.*

Example 21 (Switch and medial)**.** *Here are two common rules from deep inference proof theory* [BT01]*:*

$$\begin{array}{rrcl} \mathsf{s}: & x \wedge (y \vee z) & \to & (x \wedge y) \vee z \\ \mathsf{m}: & (w \wedge x) \vee (y \wedge z) & \to & (w \vee y) \wedge (x \vee z) \end{array}$$

In terms of relation webs, the two rules above induce the following action on graphs:



We may verify that these rules are indeed sound by checking that every maximal clique on the LHS has a subset that is a maximal clique on the RHS, under Propositions 18 and 11. For s *the two maximal cliques* $\{x, y\}$ *and* $\{x, z\}$ *are sent to* $\{x, y\}$ *and* $\{z\}$ *respectively. For* m *the two maximal cliques* $\{w, x\}$ *and* $\{y, z\}$ *are sent to themselves. Notice that* s *removes edges and reduces the size of cliques, whereas* m *adds edges but does not increase the size of any maximal cliques, it only adds new ones, here* $\{w, z\}$ *and* $\{x, y\}$*.*

We may also verify soundness using the dual condition, that every maximal stable set of the RHS has a subset maximally stable in the LHS. The argument is similar and we leave the details to the reader.

Several other examples on viewing evaluation and entailment in the setting of relation webs can be found in [DS16].

3. Boolean graph logic

We will now consider arbitrary graphs, not just the cographs, and study the notions of evaluation and entailment induced on them by the characterisations of the previous section. We call the resulting framework *Boolean Graph Logic* (BGL).

We present the basic logic in the next subsection, give some properties of evaluation and entailment in BGL in Subsection 3.2. We present a case study of these concepts in action in Subsection 3.3, and we conclude in Subsection 3.4 by characterising the graphs which compute deterministic and total Boolean relations, i.e. the Boolean functions.

We will only present the case of 'linear' graphs here, where each variable is associated to at most one node. This is sufficiently general for all of our theoretical development, but we nonetheless give an extension to the nonlinear case in Section 7.

3.1. Evaluation and entailment.

While formulas compute Boolean functions, our extension to graphs has a natural semantics based on Boolean *relations*.

Definition 22 (Evaluation)**.** *We construe graphs (with vertices in* $\mathcal{V}$*) as binary relations* $\mathcal{P}(\mathcal{V}) \times \{0, 1\}$*, defined as follows:*

- $G(X, 0)$ *if* $\exists T \in MS(G).X \cap T = \varnothing$*.*
- $G(X, 1)$ *if* $\exists S \in MC(G).X \supseteq S$*.*

Notice that, under the results of the previous section, we have that, for any linear formula A and $X \subseteq \mathcal{V}$:

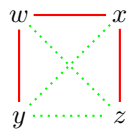
- $\mathcal{W}(A)(X, 0)$ iff $A(X) = 0$; and,
- $\mathcal{W}(A)(X, 1)$ iff $A(X) = 1$.

Thus the notion of evaluation above is indeed conservative over usual evaluation on formulas.

One immediate observation is that while evaluation is total and deterministic for cographs, since they are the webs of linear formulas which compute Boolean

functions, this is no longer necessarily the case for arbitrary graphs. In general they compute Boolean relations which might be nondeterministic, partial, or both. Let us consider an example of evaluation being nonfunctional, known already from [DS16].

Example 23 (Evaluating P_4)**.** *Let us again consider the following graph G that is isomorphic to the P_4:*



We have that $G(\{w,y\},1)$, since $\{w,y\} \in MC(G)$, and $G(\{x,y\},0)$, since $\{x,y\}$ is disjoint from $\{w,z\} \in MS(G)$. It is not difficult to see that G is indeed functional on these assignments, i.e. it is not the case that $G(\{w,y\},0)$ or $G(\{x,y\},1)$.

On the other hand, we have that both $G(\{w,x\},0)$ and $G(\{w,x\},1)$, since $\{w,x\} \in MC(G)$ and $\{w,x\}$ is disjoint from $\{y,z\} \in MS(G)$. So G is not deterministic on this assignment. Furthermore, we have that neither $G(\{y,z\},0)$ nor $G(\{y,z\},1)$, by inspection of the maximal cliques and maximal stable sets of G. So G is not total on this assignment.

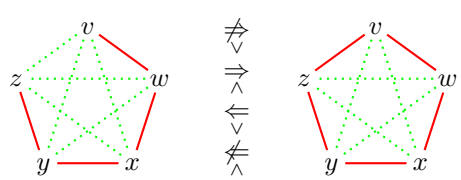
Entailment in BGL is similarly induced by our previous characterisations:

Definition 24 (Entailment)**.** *We define binary relations $\underset{\wedge}{\Rightarrow}$ and $\underset{\vee}{\Rightarrow}$ on graphs (with vertices in $\mathcal{V}$) as follows:*

- $G \underset{\wedge}{\Rightarrow} H$ *if* $\forall S \in MC(G).\exists S' \in MC(H).S' \subseteq S$.
- $G \underset{\vee}{\Rightarrow} H$ *if* $\forall T \in MS(H).\exists T' \in MS(G).T' \subseteq T$.

Again, the richer setting of arbitrary graphs means that previously equivalent notions of entailment no longer coincide, which is why we distinguish $\underset{\vee}{\Rightarrow}$ and $\underset{\wedge}{\Rightarrow}$. Let us consider the following example, which partly appeared in [DS16], to highlight the difference between the two forms of entailment.

Example 25 (5-cycle and 5-path)**.** *We have the following relationships between P_5 and C_5:*



The arguments are as follows:

- $\underset{\vee}{\not\Rightarrow}$ *since $\{v,x\}$ and $\{z,x\}$ are maximally stable on the RHS, but on the LHS only the maximal stable set $\{v,x,z\}$ concerns these three nodes.*
- $\underset{\wedge}{\Rightarrow}$ *since every maximal clique of the LHS is just an edge which is preserved in the RHS.*
- $\underset{\vee}{\Leftarrow}$ *by sending $\{v,x,z\}$ to $\{v,x\}$ or $\{z,x\}$, and every other maximal stable set of the LHS to itself in the RHS.*
- $\underset{\wedge}{\not\Leftarrow}$ *since the maximal clique $\{v,z\}$ on the RHS has no subset that is a maximal clique on the LHS.*

Despite the fact that we now have two notions of evaluation and two notions of entailment, they are still 'compatible' in a natural sense. Before proving this, we make the following remark to simplify proofs throughout this work.

Remark 26 (Duality)**.** *Many of the arguments in this work follow by 'duality'. By this we mean not only that an argument is similar to a previous one, but further that there is a formal reduction. Such reduction is exhibited by the following facts:*

- $G(X,1)$ *if and only if* $\overline{G}(\overline{X},0)$.
- $G \underset{\wedge}{\Rightarrow} H$ *if and only if* $\overline{H} \underset{\vee}{\Rightarrow} \overline{G}$.

The proofs are routine and left to the reader.

Theorem 27 (Adequacy)**.** *For graphs G and H, we have:*

$$G \underset{\wedge}{\Rightarrow} H \quad \textit{iff} \quad \forall X.(\ \textit{if}\ G(X,1)\ \textit{then}\ H(X,1)) \tag{4}$$

$$G \underset{\vee}{\Rightarrow} H \quad \textit{iff} \quad \forall X.(\ \textit{if}\ H(X,0)\ \textit{then}\ G(X,0)) \tag{5}$$

Proof. We prove only (4), the argument for (5) being dual.

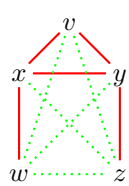
For the left-right implication, suppose $G \underset{\wedge}{\Rightarrow} H$ and $G(X,1)$. Then there is some $S \in MC(G)$ that is contained in X. Since $G \underset{\wedge}{\Rightarrow} H$, there is some $S' \in MC(H)$ s.t. $S' \subseteq S$ and so also $S' \subseteq X$, so we indeed have that $H(X,1)$.

For the right-left implication, suppose the RHS and let $S \in MC(G)$. Clearly we have that $G(S,1)$, since $S \subseteq S$, and so by assumption we have that $H(S,1)$. So there is some $S' \in MC(H)$ with $S' \subseteq S$. Since the choice of S was arbitrary, we may indeed conclude that $G \underset{\wedge}{\Rightarrow} H$, as required. □

The above result shows that evaluation essentially yields a Tarskian-style semantics for entailment by reduction to classical material implication.

3.2. **Some properties of evaluation and entailment.** As we saw in the previous subsection, there are graphs that compute nonfunctional relations, for instance the P_4. However we also have examples of graphs that are deterministic but not total (i.e. partial Boolean functions), and total but not deterministic (i.e. Boolean multifunctions), as we see in the following two examples.

Example 28 (Determinism of the Bull)**.** *While P_4 is not deterministic, it has an extension that is deterministic by adding a 'settling' node:*

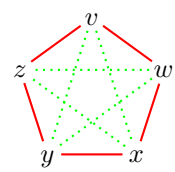


The new node v is called the 'nose of the bull', which is particularly important in so-called 'prime graphs', that we discuss in Section 5.

The Bull computes a deterministic relation, in the sense that it never evaluates to both 0 *and* 1*. We leave it to the reader to verify the cases, but this fact is also an immediate consequence of Proposition 32 later in this section.*

The Bull does not compute a total relation since, by taking the assignment $\{v,x,y\}$.

Example 29 (Totality of the 5-cycle)**.** *Let us consider the* 5*-cycle:*



Notice that C_5 has the special property that it is isomorphic to its dual, i.e. $C_5 \cong \overline{C}_5$. Its maximal cliques and maximal stable sets are just pairs.

In fact, the C_5 computes a total Boolean relation. To see this, let us consider the possible assignments X: if $|X| \leq 2$ then there is always a maximal stable set disjoint from X, whereas if $|X| \geq 3$ then there is always a maximal clique contained in X.

However, the C_5 is not deterministic: for instance, writing $X = \{v, x, z\}$, we have that $C_5(X, 1)$ since X contains the maximal clique $\{v, z\}$, but also $C_5(X, 0)$ since X is disjoint from the maximal stable set $\{w, y\}$.

We point out that the relational properties of totality and determinism behave well under duality:

Observation 30. *G is deterministic (or total) if and only if $\overline{G}$ is deterministic (respectively total).*

This follows immediately from Remark 26 by dualising assignments.

One interesting observation is that the class of deterministic graphs coincides with a well-studied class in graph theory, namely the *CIS* graphs, where every maximal clique and maximal stable set intersect [ABG06b].

Definition 31 (CIS graphs)**.** *A graph G is* CIS *if $\forall S \in MC(G).\forall S \in MS(G).S \cap T \neq \varnothing$*

Notice that cliques and stable sets may intersect at most once, so the above definition is equivalent to requiring that every maximal clique and maximal stable set have singleton intersection.

It is not hard to see the following:

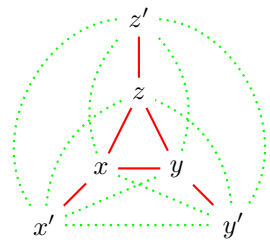
Proposition 32. *A graph G is deterministic if and only if it is CIS.*

Proof. For the left-right implication we prove the contrapositive. Suppose $S \in MC(G)$ and $T \in MS(G)$ such that $S \cap T = \varnothing$. Then $G(S, 0)$ since $S \subseteq S$ and also $G(S, 1)$ since $S \cap T = \varnothing$.

For the right-left implication we also prove the contrapositive. Suppose $G(X, 0)$ and $G(X, 1)$ for some assignment X. Thus there is some $S \in MC(G)$ and $T \in MS(G)$ with $S \subseteq X$ and $T \cap X = \varnothing$, and so $S \cap T = \varnothing$. □

As we saw in Example 28, adding a 'settling node' was a way to force the CIS property for the P_4. However, every P_4 configuration in a graph being 'settled' does not suffice to conclude that the graph is CIS. Indeed, consider the following

example from [ABG06a]:



This graph contains three P_4 configurations, all of which are 'settled', but $\{x, y, z\}$ is a maximal clique which is disjoint from the maximal stable set $\{x', y', z'\}$. Indeed, characterising CIS graphs is not easy; it is an open problem whether they can be recognised in polynomial time [ABG06a] pp. 2 (though obviously CIS is in **coNP**). Therefore, deciding whether a graph computes a deterministic relation is a priori computationally difficult.

It seems more difficult to establish a characterisation of the total graphs by structural graph theoretic properties. This is because the definition of totality a priori is more logically complex that that of determinism, a priori a Π_2^p-property. We do not know of any better upper bound for recognising totality.

Determinism and totality of a graph also has an effect on when the two notions of entailment, $\underset{\wedge}{\Rightarrow}$ and $\underset{\vee}{\Rightarrow}$ coincide:

Proposition 33. *If G is deterministic and H is total, then:*

$$\textit{if} \quad G \underset{\vee}{\Rightarrow} H \quad \textit{then} \quad G \underset{\wedge}{\Rightarrow} H \tag{6}$$

$$\textit{if} \quad H \underset{\wedge}{\Rightarrow} G \quad \textit{then} \quad H \underset{\vee}{\Rightarrow} G \tag{7}$$

Proof. We rely on the adequacy of our relational semantics, Theorem 27. For (6) we proceed as follows:

$G \underset{\vee}{\Rightarrow} H$	$\therefore$	if $H(X,0)$ then $G(X,0)$	by adequacy
	$\therefore$	if $\neg G(X,0)$ then $\neg H(X,0)$	by contraposition
	$\therefore$	if $G(X,1)$ then $\neg H(X,0)$	by determinism of G
	$\therefore$	if $G(X,1)$ then $H(X,1)$	by totality of H
	$\therefore$	$G \underset{\wedge}{\Rightarrow} H$	by adequacy

The proof of (7) is similar. □

Of course, an immediate consequence of the above result is that, on the class of deterministic and total graphs, $\underset{\wedge}{\Rightarrow} = \underset{\vee}{\Rightarrow}$. This is subsumed by our later characterisation of the graphs computing Boolean functions as just the P_4-free ones, i.e. the relation webs.

It would be interesting to establish some sort of converses to the above result, determining whether a graph is total or deterministic based on how it interacts with respect to the two notions of entailment. We leave this for future work.

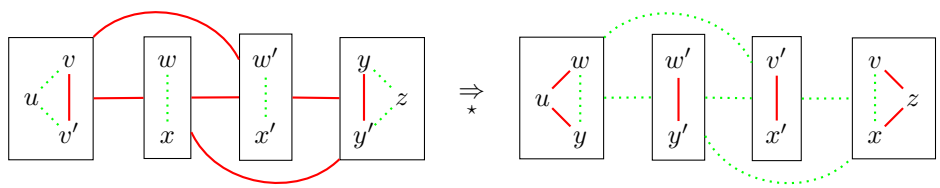
3.3. **Case study: finer interpolation of linear inferences.** At least one advantage of Boolean Graph Logic is that we have a finer notion of entailment that in turn admits a richer notion of 'proof'. We discuss a particular example in this subsection.

From [Das13] we have that the following is a valid implication:

$$\begin{array}{ll} & (u \vee (v \wedge v')) \wedge (w \vee x) \wedge (w' \vee x') \wedge ((y \wedge y') \vee z) \\ \Rightarrow & (u \wedge (w \vee y)) \vee (w' \wedge y') \vee (v' \wedge x') \vee ((v \vee x) \wedge z) \end{array} \tag{8}$$

In terms of relation webs, the implication above corresponds to the following entailments on graphs,

(9)



for $\star \in \{\vee, \wedge\}$.

This is a minimal linear inference in the sense that no linear formula (over the same variables) simultaneously implies the RHS and is implied by the LHS. However, in our graph-theoretic setting, we may indeed 'interpolate' this inference thanks to the following intermediate graph:

$$\begin{array}{ccccccccccc} & & & & w' & — & v' & — & v & — & z & — & x \\ & & & & / & \backslash & / & \backslash & / & & & & \\ w & — & u & — & y & — & y' & — & x' & & & & \end{array} \tag{10}$$

Note that we have indicated only edges, not non-edges, with black lines, for clarity.

(10) contains several P_4s, e.g. $\{w, u, y, y'\}$ and $\{y, y', v', v\}$. It does not compute a deterministic relation since $\{w, y, v', x\} \in MS(10)$ is disjoint from $\{v, z\} \in MC(10)$. On the other hand:

Claim 34. (10) *computes a total relation.*

Proof. Let X be an assignment such that $\neg(10)(X, 1)$, and let us try to construct a maximal stable set T disjoint from it. Note that X cannot contain all of $S = \{y, w', y'\} \in MC(10)$ and all of $S' = \{v', x, v\} \in MC(10)$, so let $b \in S$ and $b' \in S'$ be distinct nodes with $b, b' \notin X$.

Similarly, we must have that X does not contain one of w or u, say a, and that X does not contain one of z or x, say c.

In fact, all possible values of a, b, b', c yield a maximal stable set, which by construction is disjoint from X. Note in particular that, if $a = u$ then $b \neq y$, otherwise we would have $(10)(X, 1)$, and if $c = z$ then $b' \neq v$, for the same reason. Thus $(10)(X, 0)$ as required. □

As we mentioned, (10) indeed interpolates the inference from (8), in fact for both versions of entailment, as we will now show. The entailments $LHS(9) \underset{\wedge}{\Rightarrow} (10)$ and $(10) \underset{\vee}{\Rightarrow} RHS(9)$ require quite a large case analysis, due to the high number of maximal cliques in $LHS(9)$ and the high number of maximal stable sets in $RHS(9)$. However the other two entailments are much easier to prove directly:

Proposition 35. *We have that* $LHS(9) \underset{\vee}{\Rightarrow} (10)$ *and* $(10) \underset{\wedge}{\Rightarrow} RHS(9)$.

Proof. For $LHS(9) \underset{\vee}{\Rightarrow} (10)$, we describe how to map each maximal stable set T of (10) to a subset that is maximally stable in $LHS(9)$. We have the following cases:

- If $T \ni u$ then we map it to $\{u\}$.
- if $T \ni z$ then we map it to $\{z\}$.
- Otherwise T must contain w and x, so we may map it to $\{w, x\}$.

For $(10) \underset{\wedge}{\Rightarrow} RHS(9)$, we describe how to map each maximal clique of (10) to a subset that is a maximal clique of $RHS(9)$:

- $\{w, u\}$, $\{u, y\}$, $\{v, z\}$ and $\{z, x\}$ are all mapped to themselves.
- $\{y, w', y\}$ and $\{w', y', v'\}$ are mapped to $\{w', y'\}$.
- $\{y', v', x'\}$ and $\{v', x', v\}$ are mapped to $\{v', x'\}$. □

Now, since (10) computes a total relation, by Claim 34 above, we may conveniently appeal to Proposition 33 to immediately recover the other two entailments.

Corollary 36. *We have both of the following:*

- $LHS(9) \underset{\wedge}{\Rightarrow} (10) \underset{\wedge}{\Rightarrow} RHS(9)$
- $LHS(9) \underset{\vee}{\Rightarrow} (10) \underset{\vee}{\Rightarrow} RHS(9)$

We point out that we only used the fact that $LHS(8)$ and $RHS(8)$ are deterministic to obtain the other two entailments, not that they are total. It would be interesting to develop this case study further, establishing a maximal sequence of graphs interpolating (9). This is also related to the question of finding the 'minimal linear inference', cf. [Sip12]. Such development, however, is beyond the scope of this work.

3.4. **Deterministic and total graphs are P_4-free.** We finish this section with a somewhat surprising result: the only graphs that are both deterministic and total (i.e. Boolean functions) are already P_4-free, i.e. they are the relation webs of formulas.

Theorem 37. *A graph is deterministic and total if and only if it is P_4-free.*

One proof of this result is by a reduction to a classical result of Gurvich:

Theorem 38 ([Gur77])**.** *Suppose $f : \mathcal{P}(\mathcal{V}) \to \{0, 1\}$ is monotone and depends on variables $V_n = x_1, \ldots, x_n$. f is computed by a read-once formula (over V_n) if and only if, for every minterm S of f and every maxterm T of f, $|S \cap T| = 1$.*

We give the reduction of Theorem 37 to Theorem 38 here, but a self-contained proof can be found in the appendix, Section A.

Proof of Theorem 37. Let G be a deterministic and total graph, and define the following two Boolean functions:

$$T_G(X) := \begin{cases} 1 & G(X, 1) \\ 0 & \text{otherwise} \end{cases} \qquad F_G(X) := \begin{cases} 0 & G(X, 0) \\ 1 & \text{otherwise} \end{cases}$$

Since G is deterministic and total, we have that T_G and F_G are actually the same Boolean function, say g. What is more, g may be written simultaneously as the

following (irredundant) DNF and CNF:

$$\bigvee_{S \in MC(G)} \bigwedge S \quad = \quad T_G \quad = \quad g \quad = \quad F_G = \bigwedge_{T \in MS(G)} \bigvee T$$

Thus the minterms of g are just the maximal cliques of G and the maxterms of g are the maximal stable sets of G. Since cliques and stable sets may only intersect at most once (by simplicity of the graph), and minterms and maxterms must intersect at least once (by determinism of functions), we have from Gurvich's theorem above that g is computed by some read-once formula, say A. But now we must have that, indeed $\mathcal{W}(A) = G$, since otherwise they would have different maximal cliques and stable sets, and so compute different relations. Thus G is the web of some formula and, indeed, P_4-free. □

4. Computational Complexity of Boolean Graph Logic

In this section we study the computational complexity of evaluation and entailment in BGL. In particular we show that evaluation (to either 0 or to 1) is **NP**-complete and entailment (either disjunctive or conjunctive) is Π_2^p-complete. In contrast, for Boolean formulas evluation is **ALOGTIME**-complete and entailment is **coNP**-complete, suggesting that BGL is much more computationally rich than Boolean logic.

4.1. **Preliminaries on computational complexity.** We will assume prior knowledge of deterministic and (co-)nondeterministic Turing and oracle machines. For a language L we write $\mathbf{NP}(L)$ for the class of languages accepted by a nondeterministic Turing machine in polynomial time with access to an oracle for L. For a class of languages $\mathcal{C}$ we write $\mathbf{NP}(\mathcal{C})$ for $\bigcup_{L \in \mathcal{C}} \mathbf{NP}(L)$. From here recall that the levels of the polynomial hierarchy are defined as follows:

- $\Sigma_0^p = \Pi_0^p = \mathbf{P}$.
- $\Sigma_{i+1}^p = \mathbf{NP}(\Sigma_i^p)$.
- $\Pi_{i+1}^p = \mathbf{co}\Sigma_{i+1}^p$.

Of course, Σ_1^p is just **NP** and Π_1^p is just **coNP**.

Let us write $\vec{\exists}\mathsf{CNF}$ for the class of true (closed) quantified Boolean formulas (QBFs) of the form $\exists\vec{x}.\varphi$, where φ is a CNF. Similarly we write $\vec{\forall}\vec{\exists}\mathsf{CNF}$ for the class of true (closed) quantified Boolean formulas (QBFs) of the form $\forall\vec{x}.\exists\vec{y}.\varphi$, where φ is a CNF. It is well-known that $\vec{\exists}\mathsf{CNF}$ is **NP**-complete (being the same as SAT), and that $\vec{\forall}\vec{\exists}\mathsf{CNF}$ is Π_2^p-complete (e.g. see [SU08]).

4.2. **Complexity of entailment.** We show that the relations $\underset{\wedge}{\Rightarrow}$ and $\underset{\vee}{\Rightarrow}$ are complete for Π_2^p, i.e. $\mathbf{coNP}(\mathbf{NP})$.

Theorem 39. *$\underset{\wedge}{\Rightarrow}$ and $\underset{\vee}{\Rightarrow}$ are Π_2^p-complete.*

Proof. We reduce $\vec{\forall}\vec{\exists}\mathsf{CNF}$ to $\underset{\vee}{\Rightarrow}$, whence the case for $\underset{\wedge}{\Rightarrow}$ follows by duality. Fix an instance φ,

$$\forall\vec{x}.\exists\vec{y}. \bigwedge_{n=1}^{N} \bigvee C_n$$

where each C_n is a set of literals over the variables $\vec{x}, \vec{y}$. Write φ_0 for the matrix of φ, i.e. $\bigwedge_{n=1}^{N} \bigvee C_n$.

Remark 40. *Without loss of generality, we assume $\vec{x}$ and $\vec{y}$ are disjoint and that each x_i and y_i occurs both positively and negatively in φ (otherwise replace it by 0 or 1 appropriately). Furthermore, we suppose that that each C_n contains some x_i and some y_j (i.e. a universally bound variable and an existentially bound variable), either positively or negatively. This is because if this were not the case for some clause C, then it could equivalently be replaced by clauses $C \cup \{z\}$ and $C \cup \{\neg z\}$, for any variable z. Such a procedure at most multiplies the size of φ_0 by 2.*

Now we define the graphs $G = (V, E_G)$ and $H = (V, E_H)$ as follows:

- The set V of vertices of both G and H is the set of literal occurrences in φ_0. Formally, we write $V = \{x_i^j, \overline{x}_i^k, y_i^j, \overline{y}_i^j\}_{i,j}$, where j identifies the specific occurrence of each literal and i, j range appropriately.
- The set E_G of edges of G consists of:
  - an edge between any two nodes in the same clause (so that each C_n is a clique); and,
  - an edge between any two nodes of the form y_i^j and $\overline{y}_i^k$ (i.e. any dual literal occurrences that are existentially bound).
- E_H consists only of edges between nodes of the form x_i^j and $\overline{x}_i^k$ (i.e. any dual literal occurrences that are universally bound).

We claim that $G \underset{\vee}{\Rightarrow} H$ if and only if φ is true, as required. First suppose that $G \underset{\vee}{\Rightarrow} H$ and let $X \subseteq \vec{x}$ be an assignment to the x_i's. Let $X_1 \subseteq V$ identify the true literal occurrences under X, i.e.:

$$X_1 \quad := \quad \{x_i^j : x_i \in X\} \cup \{\overline{x}_i^k : x_i \notin X\} \tag{11}$$

Now define T to be the set of all $\vec{y}$-literal occurrences and the true $\vec{x}$-literal occurrences under X, i.e.:

$$T = \{y_i^j\}_{i,j} \cup \{y_i^k\}_{i,k} \cup X_1$$

Notice that T is a stable set in H by definition of E_H and the fact that X_1 does not contain dual literals. Furthermore it is maximally stable since the only remaining nodes are $\vec{x}$-literals that are false under X, and so have an edge to some true literal occurrence in X_1 (cf. Remark 40).

Thus, since $G \underset{\vee}{\Rightarrow} H$, there is a set $T' \subseteq T$ that is maximally stable in G. Now, notice that,

(1) By maximality, T' must intersect every C_n, since there is some x_i^j or $\overline{x}_i^k$ in each C_n, by Remark 40 and by the definition of E_G; and,
(2) T' cannot contain any dual pair y_j^l and $\overline{y}_j^m$, by the definition of E_G.

Thus T' induces a consistent assignment $Y \subseteq \vec{y}$ to the y_is (just take the set of $\vec{y}$-literals in T'), by 2, that further ensures that there is a true literal in each C_n under X, Y, by 1, as required.

Conversely, suppose that φ is true and let $T \in MS(H)$. By the definition of E_H, T induces a consistent assignment $X \subseteq \vec{x}$ to the x_is in the natural way, so let $Y \subseteq \vec{y}$ be an assignment to the y_is obtained by the truth of φ, i.e. such that $\varphi_0(X, Y)$ is true. Let X_1 be defined as above in (11) and Y_1 be defined similarly, i.e. $X_1 \subseteq V$ and $Y_1 \subseteq V$ consist of the true $\vec{x}$-literal occurrences and $\vec{y}$-liter occurrences,

respectively, under X and Y, respectively. Note that $X_1 \cup Y_1$ must intersect each C_n, since $\varphi_0(X, Y)$ is true.

Now we are almost able to define an appropriate subset of T that is maximally stable in G, but for one technicality: $X_1 \cup Y_1$ could intersect some clause C_n *twice*, i.e. there could be two true literals in C_n.[4] For this reason, we rather set T' to be an arbitrary subset of $X_1 \cup Y_1$ that intersects each C_n *exactly* once:[5]

$$T' \quad := \quad \{\text{least literal in } (X_1 \cup Y_1) \cap C_n \ : 0 < n \leq N\}$$

Now we have the following:

- T' is stable in G, since it only contains one node in each C_n and is consistent with the truth assignment Y to the y_is, cf. the definition of E_G; and,
- Furthermore T' is maximally stable in G, since it already intersects each C_n.

We have $T' \subseteq T$ (since T contains, in particular, all $\vec{y}$-literal occurrences), so we indeed have that $G \underset{\vee}{\Rightarrow} H$, as required. □

Remark 41. *In* [Str12] *Straßburger showed that the linear fragment of Boolean logic is* **coNP***-complete, by rewriting every Boolean tautology $\forall\vec{x}.\varphi$ as a linear inference $\forall\vec{x}.(\varphi \supset \psi)$, where φ and ψ are monotone and linear. It would be natural to try use this approach here to show Π_2^p-completeness of entailment, but we point out that such an approach could not work, unless polynomial hierarchy collapses to* **coNP***. This is because the set of true $\vec{\forall}\vec{\exists}$-formulas whose matrices are even monotone implications, let alone linear, is already in* **coNP***. To see this, suppose otherwise, and consider a closed formula:*

$$\forall\vec{x}.\exists\vec{y}.(\varphi(\vec{x},\vec{y}) \supset \psi(\vec{x},\vec{y})) \tag{12}$$

where φ and ψ are monotone. We have the following:

$$\begin{array}{lcll} \exists\vec{y}.(\varphi(\vec{x},\vec{y}) \supset \psi(\vec{x},\vec{y})) & \iff & \forall\vec{y}\varphi(\vec{x},\vec{y}) \supset \exists\vec{y}\varphi(\vec{x},\vec{y}) & \textit{by De Morgan equivalences} \\ & \iff & \varphi(\vec{x},\vec{0}) \supset \psi(\vec{x},\vec{1}) & \textit{by monotonicity} \end{array}$$

Thus we have reduced the truth of (12) *to the truth of $\forall\vec{x}.(\varphi(\vec{x},\vec{0}) \supset \psi(\vec{x},\vec{1}))$, which is, of course, in* **coNP***.*

4.3. **Complexity of evaluation.** By adequacy, Theorem 94, and unwinding the definition of evaluation, we already have as a corollary of Theorem 39 that there cannot be a polynomial-time algorithm for evaluation, unless the polynomial-hierarchy collapses to **coNP**. In fact we even have that there cannot be such an algorithm in $\mathbf{coNP} \cap \mathbf{NP}$, for the same reason.

In this section we go further and show that evaluation is in fact **NP**-complete. This suggests that there is no 'local' vertex-contraction style procedure for evaluation, unless $\mathbf{P} = \mathbf{NP}$.

Theorem 42. *The graph evaluation problems, i.e. $G(X, 0)$ and $G(X, 1)$, are* **NP***-complete.*

[4]We could have avoided this issue by working with generalisations of exactly-one-in-three-SAT, but this is not standard and is beyond the scope of this work.

[5]We assume there is some fixed established ordering of the literals.

Proof. We reduce $\vec{\exists}\mathsf{CNF}$ (i.e. SAT) to graph evaluation, namely the problem $G(X,0)$. (Again, the case of $G(X,1)$ is obtained by duality).

Let φ be a formula $\exists\vec{x}.\bigwedge_{n=1}^{N}\bigvee C_n$ where each C_n is a set of literals over the variables $\vec{x}$. Write φ_0 for the matrix of φ, i.e. $\bigwedge_{n=1}^{N}\bigvee C_n$.

Remark 43. *We assume without loss of generality that each C_n contains at most one of x and $\overline{x}$, for any variable x (otherwise just delete the clause).*

Let L denote the set of literal occurrences in φ_0 and define a set of fresh nodes $C = \{c_1, \ldots, c_N\}$. We define the graph $G = (V, E)$ as follows:

- The set V of nodes of G is $L \cup C$.
- The set E of edges of G consists of:
  - an edge between any two literal occurrences in the same clause; and,
  - an edge between any occurrence of x_i and any occurrence of $\overline{x}_i$; and,
  - an edge between any literal occurrence in a clause C_n and c_n.

We claim that $G(C,0)$ if and only if φ is true, as required. First, suppose that $G(C,0)$ and let $T \in MS(G)$ be disjoint from C, by definition of $G(C,0)$. We have the following:

- T intersects each C_n. Otherwise, it would contain c_n and thus intersect C.
- T is *consistent*, i.e. if it contains an occurrence of x it does not contain an occurrence of $\bar{x}$, by definition of E.

Thus T induces an assignment $X \subseteq \vec{x}$ in the natural way such that $\varphi_0(X)$ is true.

Conversely, suppose that φ is true and let $X \subseteq \vec{x}$ be a satisfying assignment for φ_0. The set of true literal occurrences in G under X (defined just like X_1 was defined in (11)) almost serves as an appropriate maximal stable set but for the technicality, as before, that it may intersect some clause more than once. Again, we avoid this issue by making arbitrary choices. Define:

$$T \quad := \quad \{\text{least true literal under } X \text{ in } C_n : 0 < n \leq N\}$$

We have the following:

- T is stable, by Remark 43 and consistency of X;
- T intersects each C_n, since X was a satisfying assignment for φ_0, and so T is furthermore maximally stable by definition of E; and,
- T does not intersect C, by construction.

Thus $T \in MS(G)$ such that $T \cap C = \varnothing$, as required. □

5. Modular decomposition and an algorithm for evaluation

Despite evaluation being **NP**-complete, we can still define a "recursive" algorithm for it based on known graph decompositions. While this algorithm does not operate in polynomial time, it does allow us to reduce evaluation, as well as determining whether a graph is deterministic or total, to the so-called "prime" graphs.

5.1. Modules: 'zooming' out of graphs.

The notion of a module generalises the notion of a formula context to arbitrary graphs.

Definition 44 (Modules)**.** *For a graph $G = (V, E)$ a* module *is a set $M \subseteq V$ such that every element of M has the same neighbourhood outside M in G. I.e. $\forall x, y \in M. \forall z \in V \setminus M. (\{x, z\} \in E$ iff $\{y, z\} \in E)$.*

The sets $V, \varnothing$ and $\{x\}$, for $x \in V$, are always modules and are known as the trivial modules*. Any module $M \subsetneq V$ is a* proper *module.*

Rephrasing the above definition visually, M being a module means that, for any $z \notin M$, either:

- $\forall m \in M.\, m$ —— y ; or
- $\forall m \in M.\, m$ ····· y .

As a notational convention, given a graph G, for two sets of nodes X, Y we write X —— Y (or X ····· Y) to express that for all $x \in X$ and all $y \in Y$, x —— y (respectively, x ····· y) in G.

Observation 45. *For disjoint modules M, N, we have either M —— N or M ····· N*

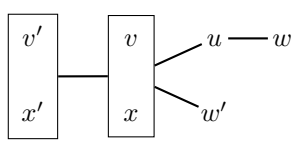
In this way, modules allow us to 'zoom out' and see graphs as compositions of smaller graphs. This is similar to how we displayed the graphs in the case study of Section 3.3. Let us elaborate on this idea more formally.

Definition 46 (Quotients)**.** *Fix a graph $G = (V, E)$ and let $P \subseteq \mathcal{P}(V)$ be a partition of V into (nonempty) modules (called a* modular partition*). The* quotient graph *G/P is defined as follows:*

- *The set of vertices of G/P is just P.*
- *$\{M, N\}$ is an edge of G/P just if M —— N in G.*

Notice that we deliberately use modules themselves as nodes in a quotient graph. This allows us to freely switch between consideration of the entire graph and just its quotient when seeing graphs visually. For the same reason, we will often only speak about quotients up to isomorphism.

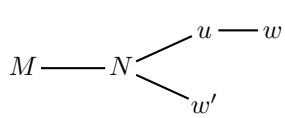
Example 47. *Consider the following graph G, written with only edges indicated:*



Here we have identified two nontrivial modules, $M = \{v', x'\}$ and $N = \{v, x\}$ whose elements have the same adjacencies. From this presentation, we may isolate a particular partition of the vertices into modules:

$$P = \{M, N, \{u\}, \{w'\}, \{w\}\}$$

The graph G/P is thus the following:



Notice that, as an overloading of notation, we may use the same diagram above as a representation of G itself, identifying M and N with $G{\restriction}M$ and $G{\restriction}N$ respectively.

Notice that the module M in the previous example is not maximally proper, as it may be extended by w'. The point of modular decompositions we later define is to take quotients as finely as possible, recursively expressing it as a 'graph of graphs'. However, some graphs cannot be simplified in this way, and these form the critical points of modular decomposition.

Definition 48 (Prime graphs)**.** *Let G be a graph of size at least 3. If every module in G is trivial, we say that G is a* prime *graph.*

Prime graphs have been studied extensively, see for example [CI98] [IV14] [HP10].

5.2. **Modular decomposition of a graph.** We have the following natural algebraic properties of modules:

Proposition 49 (Algebra of modules)**.** *Let G be a graph with modules M and N.*

(1) $M \cap N$ is a module.
(2) If $M \cap N \neq \varnothing$ then $M \cup N$ is a module.

As a result of this algebraic structure, we have the following well-known decomposition result for graphs:

Proposition 50 (Modular decomposition of a graph, [Gal67])**.** *For every nonempty graph G, we have exactly one of the following:*

(1) G is a singleton graph $\{x\}$.
(2) G is disconnected.
(3) G is co-disconnected (i.e. $\overline{G}$ is disconnected).
(4) $V(G)$ is partitioned by its maximal proper modules.

This motivates the following definition:

Definition 51 (Prime quotient)**.** *Given a biconnected graph $G = (V, E)$, its* prime quotient*, written P_G, is the set of its maximal proper modules.*

Under Proposition 50, we have that G/P_G (often simply written P_G, as abuse) is a graph with edges $\{M, N\}$ just if M —— N in G. As the name suggests, the prime quotient is indeed a prime graph:

Fact 52. *The prime quotient of a bi-connected graph is a prime graph.*

Example 53. *Revisiting Example 47, notice that the module M is not maximally proper, since it may be extended by w'. Let us call the resulting module M'. The prime quotient of G is $P_G = \{M', N, \{u\}, \{w\}\}$, so that G/P_G is actually a P_4:*

$$M' \text{——} N \text{——} u \text{——} w$$

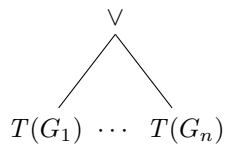
The P_4 is the smallest prime graph.

We are ready to define the modular decomposition tree of a graph.

Definition 54 (Decomposition tree)**.** *We define the* decomposition tree *of a graph G, written $T(G)$ by induction on its size, under the classification of Proposition 50:*

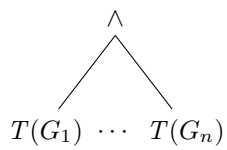
(1) If G has just one node, i.e. is $(\{x\}, \varnothing)$, then $T(G) := G$.

(2) If G is disconnected with connected components $G_1, \ldots, G_n$, then $T(G)$ is:



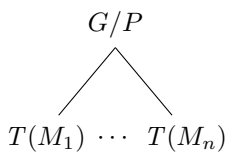
We write $T(G) = \vee(T(G_1), \ldots, T(G_n))$ as a more compact notation.

(3) If $\overline{G}$ is disconnected with connected components $\overline{G}_1, \ldots \overline{G}_n$, then $T(G)$ is:



We write $T(G) = \wedge(T(G_1), \ldots, T(G_n))$ as a more compact notation.

(4) Otherwise let $P_G = \{M_1, \ldots, M_n\}$ and define $T(G)$ as,

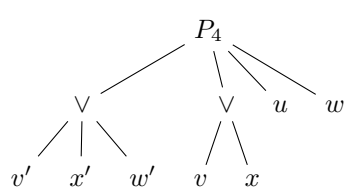


where we identify each M_i with the corresponding induced subgraph $G{\restriction}M_i$. We write $T(G) = (G/P)(T(M_1), \ldots, T(M_n))$ as more compact notation.

Remark 55. *Notice that, identifying formulas with their formula trees, the mapping $T(\cdot)$ on cographs is precisely the inverse of the mapping $\mathcal{W}(\cdot)$, mapping a formula to its relation web, up to associativity and commutativity of $\vee$ and $\wedge$.*

Let us see some examples of decomposition trees in action.

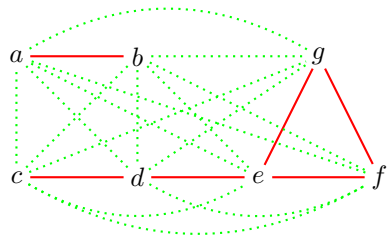
Example 56. *Revisiting Examples 47 and 53, we have that $T(G)$ is as follows:*



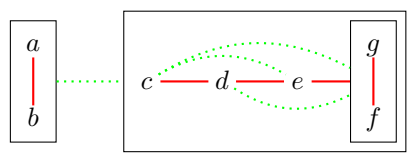
Notice that we have simply written P_4 as the root of $T(G)$ rather than the proper isomorphic quotient graph, but this causes no ambiguity here.

The compact notation for $T(G)$ is $P_4(\vee(v', x', w'), \vee(v, x), u, w)$.

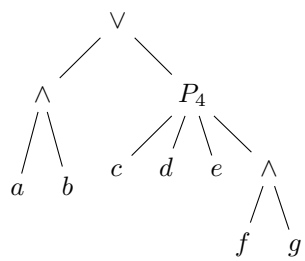
Example 57. *Let G be the following graph:*



Notice that we may equivalently write G in the following way,



and thus $T(G)$ has the following form:



5.3. **Maximal cliques and stable sets via modular decomposition.** In this subsection we outline an algorithm for evaluation that operates recursively on the modular decomposition tree. This essentially reduces the problem of evaluation to the prime graphs.

Observation 58. *Let $G = (V, E)$ be a graph and P a modular partition. The map $\varphi : V \to V/P$ by $\varphi(x) = M$ unique such that $M \ni x$ induces a homomorphism from G to G/P. Therefore the images of cliques under φ are again cliques.*

In fact the particular homomorphisms induced by quotients preserve a lot more structure. For instance stable sets are also preserved, as well as maximality, as we will now show. The point of this is that, in order to evaluate graphs recursively on their modular decomposition, we need to first classify their maximal cliques and stable sets in this way.

Lemma 59. *Let $G = (V, E)$ be a graph and $P \subseteq \mathcal{P}(V)$ be a modular partition. For $X \subseteq V$ we have:*

- *$X \in MC(G)$ iff there exists $S \in MC(G/P)$ and some $S_M \in MC(M)$, for $M \in S$, s.t. $X = \bigcup_{M \in S} S_M$.*
- *$X \in MS(G)$ iff there exists $T \in MS(G/P)$ and some $T_M \in MC(M)$, for $M \in S$, s.t. $X = \bigcup_{M \in T} T_M$.*

Proof sketch. We prove only the statements regarding maximal cliques, the ones for maximal stable sets following by duality.

For the left-right implication, let $X \in MC(G)$ and set $S = \{M \in P : X \cap M \neq \varnothing\}$.

- S is a clique of G/P: for any distinct $M, M' \in S$ there are some $x \in M$ and $y \in M'$ by nonemptiness, and we have x —— y in G since $x, y \in X \in MC(G)$. Thus M —— M' in G/P by modularity.
- S is maximal: suppose there is some $M' \notin S$ such that M' —— S in G/P. Any $x \in M'$ is not in X, by definition of S, and we have x —— M in G for all $M \in S$. Therefore x —— X in G (since P partitions V) contradicting maximality of X.

Now we may define $S_M = X \cap M$ for each $M \in S$.

- Each S_M is a clique of M, since $S_M \subseteq X \in MC(G)$.
- Each S_M is maximal: suppose there is some $x \in M \setminus S_M$ with x —— S_M in M. By modularity we must have that x —— S in G and hence already $x \in S$ by maximality of S.

Since P is a partition of V, we also have that $\{S_M : M \in S\}$ partitions X, so $X = \bigcup_{M \in S} S_M$, as required.

For the right-left implication, suppose $S \in MC(G/P)$ and $S_M \in MC(M)$, for $M \in S$, s.t. $X = \bigcup_{M \in S} S_M$. To show that X is a clique, let $x, y \in X$ be distinct. We have two cases:

- there is M s.t. $x, y \in M$, in which case $x, y \in S_M$ so x —— y in G; or,
- there are disjoint M, M' with $x \in M$ and $y \in M'$, in which case M —— M' in S and so also x —— y in G.

For maximality, suppose $x \notin X$ s.t. x —— X in G. Again we have two cases:

- if there is $M \in S$ with $x \in M$, then S_M can be extended by x;
- if $x \in M \notin S$, then we must have M —— S by modularity, and so S can be extended by M.

In either case we have a contradiction, concluding the proof. □

Since the disjunctive and conjunctive nodes in the decomposition tree are special cases of a modular partition, the following characterisation is now immediate from the preceding lemma:

Proposition 60 (Maximal cliques and stable sets via modular decomposition)**.** *Let G be a graph and $X \subseteq V(G)$. We have:*

(1) If G is the singleton graph $\{x\}$ then $MC(G) = MS(G) = \{\{x\}\}$.
(2) If $T(G) = \vee(T(G_1), \ldots, T(G_n))$ then:
  - *$X \in MC(G)$ if and only if, for some i, $X \in MC(G_i)$.*
  - *$X \in MS(G)$ if and only if, for every i, $X \cap V(G_i) \in MS(G_i)$.*

(3) If $T(G) = \wedge(T(G_1), \ldots, T(G_n))$ then:
  - *$X \in MC(G)$ if and only if, for every i, $X \cap V(G_i) \in MC(G_i)$.*
  - *$X \in MS(G)$ if and only if, for some i, $X \in MS(G_i)$.*

(4) If $T(G) = (G/P)(T(M_1), \ldots, T(M_n))$ then,

- *$X \in MC(G)$ iff there exists $S \in MC(G/P)$ and some $S_i \in MC(M_i)$, for $M_i \in S$, s.t. $X = \bigcup_{M_i \in S} S_i$.*
- *$X \in MS(G)$ iff there exists $T \in MS(G/P)$ and some $T_i \in MS(M_i)$, for $M_i \in T$, s.t. $X = \bigcup_{M_i \in T} T_i$ for some $T_i \in MC(M_i)$.*

5.4. **Evaluation via modular decomposition.** We give a characterisation of evaluation by recursion on the decomposition tree of graphs. In effect, this yields an algorithm for evaluation by reduction to prime graphs.

Since our semantics is multivalued, we have to be a little careful with how we construct assignments during recursion on the decomposition tree.

Definition 61 (Positive and negative quotient assignments)**.** *Let $G = (V, E)$ be a graph and $P \subseteq \mathcal{P}(V)$ be a modular partition. For an assignment $X \subseteq V$, we define the following subsets of P, relative to G:*

$$\begin{array}{rcl} \mathsf{Pos}_P(X) & := & \{M \in P : M(X, 1)\} \\ \mathsf{Neg}_P(X) & := & \{M \in P : \neg M(X, 0)\} \end{array}$$

Lemma 62. *Let $G = (V, E)$ be a graph and $P \subseteq \mathcal{P}(V)$ be a modular partition. For an assignment $X \subseteq V$ we have:*

(1) $G(X, 1)$ if and only if $(G/P)(\mathsf{Pos}_P(X), 1)$
(2) $G(X, 0)$ if and only if $(G/P)(\mathsf{Neg}_P(X), 0)$.

Proof. We prove only (1), the case of (2) following by duality.

For the left-right implication, let $S \in MC(G)$ with $S \subseteq X$. By Lemma 59, there is $S_P \in MC(G/P)$ s.t. $S = \bigcup_{M \in S_P} S_M$ for some $S_M \in MC(M)$, for $M \in S_P$. Now, for each $M \in S_P$, we have that $S_M \subseteq S \subseteq X$, so $M(X, 1)$ and $S_M \in \mathsf{Pos}_P(X)$. Thus we have that $(G/P)(\mathsf{Pos}_P(X), 1)$, as required.

For the right-left implication, let $S \in MC(G/P)$ with $S \subseteq \mathsf{Pos}_P(X)$. By definition of $\mathsf{Pos}_P(X)$ we have $\forall M \in S.M(X, 1)$. so for each $M \in S$ fix $S_M \in MC(M)$ s.t. $S_M \subseteq X$. Now, by Lemma 59 we have $S' \in MC(G)$ with $S' = \bigcup_{M \in S} S_M$. Since $S_M \subseteq X$ for each $M \in S$, we also have $S' \subseteq X$, so $G(X, 1)$ as required. □

The following result, computing evaluation by recursion on a decomposition tree, is now immediate from the Lemma above and Proposition 60:

Theorem 63 (Evaluation by recursion on decomposition trees)**.** *Let G be a graph and $X \subseteq V(G)$. We have:*

(1) If G is a singleton graph with $V(G) = \{x\}$ then:
- *$G(X, 1)$ if and only if $x \in X$.*
- *$G(X, 0)$ if and only if $x \notin X$.*

(2) If $T(G) = \vee(T(G_1), \ldots, T(G_n))$ then:
- *$G(X, 1)$ if and only if, for some i, $G_i(X, 1)$.*
- *$G(X, 0)$ if and only if, for every i, $G_i(X, 0)$.*

(3) If $T(G) = \wedge(T(G_1), \ldots, T(G_n))$ then:
- *$G(X, 1)$ if and only if, for every i, $G_i(X, 1)$.*
- *$G(X, 0)$ if and only if, for some i, $G_i(X, 0)$.*

(4) If $T(G) = (G/P)(T(G_1), \ldots, T(G_n))$ then:
- *$G(X, 1)$ if and only if $(G/P)(\mathsf{Pos}_P(X), 1)$*
- *$G(X, 0)$ if and only if $(G/P)(\mathsf{Neg}_P(X), 0)$.*

6. Positional games beyond cographs

A refined view of evaluation can be given in a game theoretic setting; in particular, *extensive game forms* forms have been studied in connection with read-once formulas, leading to the notion of *positional games*, e.g. [Gur82, GGCH11]. In this section, we will see how to extend the 'evaluation game' on formulas to arbitrary graphs. Along the way, we recover a distinction between the static and sequential versions of this game, a distinction which does not exist at the level of formulas.

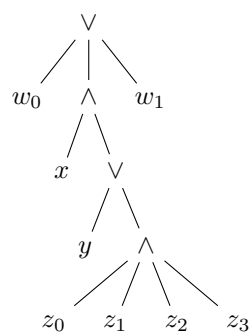
The evaluation game of a formula is played as follows. Two players construct a path through the formula tree, with one player ('Eloise') choosing directions at disjunction nodes and the other ('Abelard') choosing directions at conjunction nodes. The possible 'outcomes' of the play are then the leaves of the tree, i.e. the variables of the formula, and a notion of winning can be imposed via some Boolean payoff set (i.e. an assignment). In this way, the strategies of the Eloise and Abelard are determined precisely by the minterms and maxterms, respectively, of the formula they are playing on. One interesting pursuit is to establish which game forms correspond to evaluation games, a question resolved for formulas in [Gur82], though extending this result to arbitrary graphs is beyond the scope of this work.

Let us consider an example of the evaluation game on formulas.

Example 64 (Evaluation games)**.** *Let A be the following formula:*

$$w_0 \vee (x \wedge (y \vee (z_0 \wedge z_1 \wedge z_2 \wedge z_3))) \vee w_1$$

The formula tree $T(A)$ is as follows:[6]



A possible play of the corresponding game would be as follows:

- *The root of the tree is labelled with a $\vee$, so Eloise plays one of its children. If she chooses w_0 or w_1 then the play ends with outcome w_0 or w_1, respectively. Suppose she picks the $\wedge$ child.*
- *It is now Abelard to choose, and he has two choices: either x, again terminating the play with outcome x, or $\vee$. Suppose he chooses the latter.*
- *At this point let us suppose that Eloise ends the play by choosing y.*

The strategy employed by Eloise in this play corresponds to the minterm $\{x, y\}$, in the sense that she always makes a choice that admits the possibility of an outcome in that set. Abelard's strategy corresponds to several maxterms, namely $\{w_0, w_1, y, z_i\}$

[6]Notice that there is some mismatch here in the sense that formulas have binary connectives but our decomposition trees allow arbitrary fan-in. We gloss over this mismatch in this example, and wherever it is not ambiguous.

for any $i < 4$, for the same reason. The play terminated before Abelard had to choose a z_i, which is why it is consistent with multiple strategies.

Equipped with an assignment $X \subseteq \{w_0, w_1, x, y, z_0, z_1, z_2, z_3\}$, we may say that the play above is winning *for Eloise if the outcome y is in X, and otherwise winning for Abelard.*

6.1. **Games on arbitrary graphs.** In order to extend the evaluation game to arbitrary graphs, it thus suffices to establish rules of play at prime nodes in the decomposition tree. As hinted at in the previous example, one way to see the play is that Eloise is playing according to some maximal clique whereas Abelard plays according some maximal stable set. We directly import this idea into the following definition of games on arbitrary graphs, though we will see that the games induced become more sensitive to the format of gameplay than before.

Definition 65 (Games on graphs)**.** *We define a two-player game, with players Eloise and Abelard by recursion on the decomposition tree of a graph.* Plays *of the game have* outcome *that is either a variable $x \in \mathcal{V}$ or $\varnothing$.*

Let G be a graph, and let us define a play of G and the outcome of a play as follows:

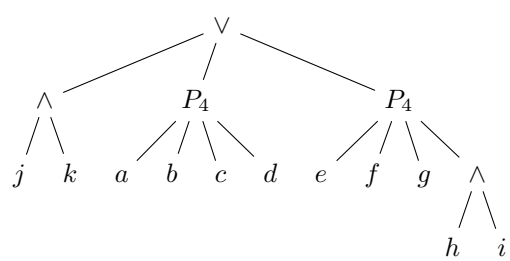
(1) If $V(G)$ is the singleton $\{x\}$ then the play is empty and the outcome is x.
(2) if $T(G)$ has the form $\vee(T(G_1), \ldots, T(G_n))$, then Eloise chooses one of the G_is and the play continues on G_i.
(3) If $T(G)$ has the form $\wedge(G_1, \ldots, G_n)$, then Abelard chooses one of the G_is and the play continues on G_i.
(4) If $T(G)$ has the form $(G/P)(M_1, \ldots, M_n)$, then:
   - *Eloise chooses some $S \in MC(G/P)$; and,*
   - *Abelard chooses some $T \in MS(G/P)$,*

   and the play continues on the unique module M_i in the intersection $S \cap T$, if it exists. If there is a deadlock, i.e. $S \cap T = \varnothing$, then the play ends with outcome $\varnothing$.

We construe plays as the sequence of graphs induced by the choices of Eloise and Abelard following the rules above. This sequence always determines a path through the decomposition tree of G from the root to either a leaf, if the outcome is the corresponding variable, or an internal prime node, if the play hits a deadlock.

Remark 66 (Deadlocks and determinism)**.** *Note that on deterministic graphs maximal cliques and stable sets always intersect, since they are CIS, and therefore a play always has a non-empty outcome.*

Example 67 (A play of a graph)**.** *Let G be the graph in Figure 6.1, given in both the usual red-green presentation, left, and its 'modular' presentation, right. The decomposition tree $T(G)$ is:*



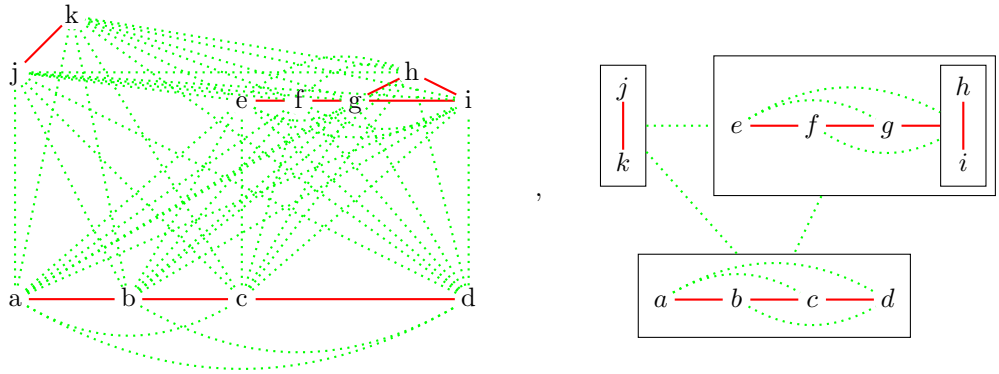
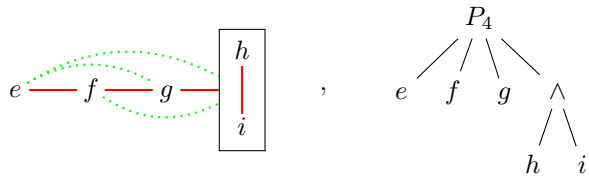


FIGURE 2. An example of a graph and its modular presentation, obtained by recursively partitioning the graph into maximal proper modules.

We will now run through a possible play on G:

(1) The root of $T(G)$ is $\vee$, so Eloise plays first and chooses the rightmost child, P_4, corresponding to the following maximal proper module, left, and subtree, right:



(2) The root is now P_4, which is prime.
- *Eloise chooses the maximal clique $\{g, \{h, i\}\}$, consisting of the third and fourth child.*
- *Abelard chooses the maximal stable set $\{e, \{h, i\}\}$, consisting of the first and fourth child.*

(3) The intersection is nonempty, corresponding to the following module, left, and subtree, right:

Since the root is $\wedge$, it is Abelard to play, and he chooses h.

(4) h is a leaf of the decomposition tree, so the play ends with outcome h.

If at step (2) Eloise had chosen $\{f, g\}$ and Abelard had chosen $\{e, \{h, i\}\}$ then the play would end with outcome $\varnothing$, since the intersection of these sets is empty.

6.2. **Boolean payoffs and strategies.** In order to talk about the 'winner' of a game, we introduce standard Boolean payoff sets, playing the role of assignments earlier.

Definition 68 (Winning under an assignment)**.** *Let $G = (V, E)$ be a graph and $X \subseteq V$ be an assignment. Given a play of G, we say that:*

- *Eloise wins if the outcome of the play is some $x \in X$.*
- *Abelard wins if the outcome of the play is some $x \notin X$.*
- *The play is a draw if the outcome of the play is $\varnothing$.*

As mentioned earlier, evaluation games with Boolean payoff gave us a game-theoretic characterisation of evaluation in the case of formulas, equivalently cographs. There the situation was simple: evaluating to 1 corresponds to a winning strategy for Eloise, and evaluating to 0 corresponds to a winning strategy for Abelard. Here it is not so simple, not only because our semantics is relational but also since determinacy is sensitive to the 'mode' of play.

Recalling Definition 65, notice that we did not specify at a prime node whether Eloise and Abelard make their choices independently of each other, or whether one player is able to make their choice once the other has already declared theirs. In the formula setting this makes no difference: the player with a winning strategy has a uniform strategy no matter the moves of the opponent, - just play according to some minterm or maxterm. For arbitrary graphs it turns out that there is an advantage for the second player, since they may react to the choices of the first.

Definition 69 (Sequential and static strategies)**.** *Let $G = (V, E)$ be a graph. A* strategy *on G is a specification of choices for a player at each relevant node of $T(G)$.*[7] *Formally, a strategy for Eloise is a map that associates:*

- *to each $\vee$ node of $T(G)$ a child of that node; and,*
- *to each prime node G/P of $T(G)$ some $S \in MC(G/P)$.*

Dually, a strategy for Abelard is a map that associates:

- *to each $\wedge$ node of $T(G)$ a child of that node; and,*
- *to each prime node G/P of $T(G)$ some $T \in MS(G/P)$.*

Furthermore, we may distinguish different modes *of strategy:*

- *If a strategy σ for a player p at prime nodes may depend on the other player's choice, then we call σ a* reactionary *strategy for p.*
- *Otherwise, if a strategy σ for a player p does not depend on the choice of the other player at prime nodes, then we call σ a* static *strategy for p.*

We say that a p-strategy σ is winning *with respect to an assignment X if every play according to σ results in a win for p. We say that σ is* drawing *with respect to X if every play according to σ results in a win or draw for p.*

6.3. **Characterisation of evaluation via static strategies.** Let us first consider the somewhat simpler case of static games, since the results and arguments are similar to the formula setting, only accounting for nonfunctionality and the possibility of draws. Table 6.3 summarises the circumstances in which each of the players have a winning or drawing strategy, given an assignment X. This of course depends on what G may evaluate to on X, but we also distinguish whether G is deterministic or not, since in the former case draws are impossible, by the CIS property. The entries of the table are justified by the following result:

[7]Note that we only consider 'positional' strategies here, where the choice of a player does not depend on the history of the play. It is not hard to see that this is sufficiently general for determinacy.

Static games:

	$G(X) = \{1\}$	$G(X) = \{0\}$	$G(x) = \varnothing$	$G(X) = \{0, 1\}$
Det.	Eloise has a winning strategy	Abelard has a winning strategy	-	N/A
Non-det.	Eloise has a drawing strategy	Abelard has a drawing strategy	-	Eloise and Abelard have a drawing strategy

TABLE 1. Winning/drawing strategies for static games. We write $G(X) = B$ if $G(X, b) \iff b \in B$. - denotes the situation where neither player has a winning or drawing strategy, and N/A denotes a situation that cannot occur.

Theorem 70. *Let G be a graph and $X \subseteq V(G)$ an assignment. We have the following:*

(1) $G(X, 1)$ if and only if Eloise has a static drawing strategy.
(2) $G(X, 0)$ if and only if Abelard has a static drawing strategy.

If G is deterministic then the above can be strengthened to static winning strategies.

Proof. Notice that the final clause on deterministic graphs follows immediately from (1) and (2) by the CIS property, Proposition 32.

For the left-right implication of 1, the idea is that Eloise plays to remain in a maximal clique inside the given assignment. Formally, given a graph G, an assignment X, and some maximal clique $S \subseteq X$, we describe what it means for Eloise to play *according to* S, appealing to the classification of maximal cliques in Propostion 60:

- If $T(G)$ is a singleton then there is nothing to play and, since S is nonempty, Eloise wins.
- If $T(G) = \vee(T(G_1), \ldots, T(G_n)$, then we must have $S \in MC(G_i)$, and so Eloise chooses this G_i and continues playing according to S.
- If $T(G) = \wedge(T(G_1), \ldots, T(G_n))$, then we must have that $S_i = S \cap V(G_i) \in MC(G_i)$, so if Abelard chooses G_i Eloise continues playing according to S_i.
- If $T(G) = (G/P)(T(M_1), \ldots, T(M_n))$, then $S = \bigcup_{i \in I} S_i$, for some $\{M_i\}_{i \in I} \in MC(G/P)$ and $S_i \in MC(M_i)$. Eloise plays the maximal clique $\{M_i\}_{i \in I}$ and, if the play continues on some module M_i, with $i \in I$, then she continues playing according to S_i.

There are two options for a play of this strategy: either a variable is reached (the first case) and Eloise wins, or at a prime node (fourth case) the intersection of Eloise's and Abelard's sets is empty and the game is drawn.

Conversely, suppose σ is a static drawing strategy for Eloise and write $T(\sigma)$ for the subtree of $T(G)$ induced by it. Formally, $T(\sigma)$ is the smallest subtree satisfying:

- The root of $T(G)$ belongs to $T(\sigma)$;
- For any $\vee$ node in $T(\sigma)$ the child chosen by σ belongs to $T(\sigma)$;
- For any $\wedge$ node in $T(\sigma)$, all its children also belong to $T(\sigma)$;
- For a prime node G/P in $T(\sigma)$, all children in the maximal clique chosen by σ belong to $T(\sigma)$.

Let us write $L(\sigma)$ for the set of leaves of $T(\sigma)$. We have the following:

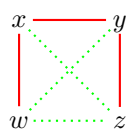
- $L(\sigma)$ is a clique of G: let $x, y \in L(\sigma)$ be distinct and let us inspect the root of the smallest subtree of $T(\sigma)$ (equivalently $T(G)$) containing both x and y. It cannot be a $\vee$ node by construction of $T(\sigma)$, and if it is a $\wedge$ or G/P node, then x —— y in G by construction of $T(\sigma)$.
- $L(\sigma)$ is maximal: suppose not and let $z \notin L(\sigma)$ such that z —— $L(\sigma)$ in G. Let us take the smallest subtree of $T(\sigma)$ whose root ν induces a subtree of $T(G)$ containing z. ν cannot be a $\vee$ node, since that contradicts z —— $L(\sigma)$; ν cannot be a $\wedge$ node since that contradicts leastness of ν; finally, if ν is a prime node G/P, then by modularity we have that the clique chosen by σ can be extended to the module containing z, contradicting maximality.

2 follows by duality. □

If X evaluates to neither 0 nor 1, it follows that neither player necessarily has a drawing/winning strategy in a static setting. Note that this does not contradict finite determinacy, which a priori holds only for sequential games.

Example 71 (Non-determinacy)**.** *Let us revisit the P_4:*

(13)



Recall that under the assignment $X = \{w, z\}$, we have neither $G(X, 0)$ nor $G(X, 1)$. It turns out that neither player has a static drawing strategy in this circumstance, as we now argue. Since the P_4 is prime, the corresponding game is one-shot, so it suffices to show that neither player has a drawing move.

- *If Eloise plays $\{x, y\}$ then she will lose if Abelard plays $\{x, z\}$ or $\{w, z\}$;*
- *If Eloise plays $\{x, w\}$ then she will lose if Abelard plays $\{x, z\}$;*
- *If Eloise plays $\{y, z\}$ then she will lose if Abelard plays $\{w, y\}$.*

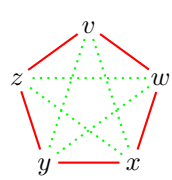
Since the P_4 is isomrophic to its dual, it is immediate that Abelard too has no drawing strategy.

As we will see in the next subsection, finite derminacy *does* apply to sequential games, and in situations like the P_4 above, it is the second player who actually has a winning strategy.

It is natural to wonder whether any of the results of Table 6.3 can be strengthened from drawing to winning. This is not the case, as we can see from the following example.

Example 72. *Let us recall the 5-cycle:*

(14)



Recall that this graph is not deterministic (though it is total) since the assignment, say, $\{v, w\}$ evaluates to both 0 (being disjoint from the maximal stable set $\{x, z\}$)

Sequential games:

	$G(X) = \{1\}$	$G(X) = \{0\}$	$G(X) = \varnothing$	$G(X) = \{0, 1\}$
Det.	As the second player, Eloise has a winning strategy	As the second player, Abelard has an winning strategy	As the second player, Eloise and Abelard have a winning strategy	N/A
Non-det.	As the second player, Eloise has a winning strategy	As the second player, Abelard has a winning strategy	As the second player, Eloise and Abelard have a winning strategy	As the second player, Eloise and Abelard have a drawing strategy

TABLE 2. Winning/drawing strategies for sequential games. N/A denotes a situation that cannot occur. We write $G(X) = B$ if $G(X, b) \iff b \in B$.

and 1 *(being a maximal clique itself). It is also a prime graph and so the corresponding game is one-shot.*

Now, if we take the slightly larger assignment $\{v, z, w\}$*, the graph still evaluates to* 1 *but no longer evaluates to* 0*. Nonetheless, Eloise cannot force a win: whichever maximal clique she chooses, Abelard may choose a disjoint maximal stable set, as reasoned above, by the symmetry of the graph.*

6.4. **Characterisation of evaluation via sequential strategies.** For sequential games, the second player has an advantage since they have strictly more information to exploit at each prime node, namely the opponent's move at that node.

Example 73 (Winning sequentially)**.** *Revisiting Example 72, where the drawing player could not force a win, let us consider what happens in a sequential setting. Taking the same graph* (14) *and the same assignment* $\{v, z, w\}$*, it is clear that Eloise can force a win since every maximal stable set intersects the assignment, and so she can react to Abelard's move with the appropriate maximal clique.*

What is more, finite determinacy now applies and so the game is completely determined.

Example 74 (Determinacy and non-totality)**.** *Revisiting Example 71, where neither player could force even a draw, let us consider what happens in a sequential setting. Taking the same graph* (13) *and assignment* $X = \{w, z\}$*, it turns out that the second play can actually force a win. In fact, the case analysis of Example 71 immediately shows that Abelard wins when playing second.*

Table 6.4 summarises the circumstances in which a player has a winning or drawing reactionary strategy. The results for static strategies naturally still hold for the first player. Again, we distinguish deterministic graphs from general graphs.

The entries of the table are justified by the following result, along with Theorem 70:

Theorem 75. *Let* G *be a graph and* $X \subseteq V(G)$ *an assignment. We have the following:*

(1) $\neg G(X, 0)$ *if and only if Eloise has a reactionary winning strategy.*
(2) $\neg G(X, 1)$ *if and only if Abelard has a reactionary winning strategy.*

Proof. For the left-right implication of 1, the idea is that Eloise aims to maintain the property that all maximal stable sets consistent with the play thus far intersect the given assignment. Formally, given an assignment X intersecting every stable set of a graph G, we define what it means to play *according to* X, appealing to the classification of maximal stable sets in Propostion 60:

- If $T(G)$ is a singleton there is nothing to play and, since X is nonempty, Eloise wins.
- If $T(G) = \vee(T(G_1), \ldots, T(G_n)$ then X must intersect some G_i, and so Eloise chooses this G_i and continues to play according to $X \cap V(G_i)$.
- If $T(G) = \wedge(T(G_1), \ldots, T(G_n))$ then X must intersect every G_i, so if Abelard chooses some G_i then Eloise continues to play according to $X \cap V(G_i)$.
- If $T(G) = (G/P)(T(G_1), \ldots, T(G_n))$ and Abelard plays $T \in MS(G/P)$, then we must have that X intersects some $M \in T$, and so Eloise chooses any maximal clique of G/P extending M.

Notice that any play of this strategy cannot reach a deadlock, by construction, and so it must terminate in the first case, at a variable, and Eloise wins.

Conversely, let σ be a reactionary winning strategy for Eloise and, for any maximal stable set T, take the play induced by σ when Abelard plays according to T (cf. the proof of Theorem 70). By induction on the length of the play, it is not difficult to see that the play always intersects T, and hence $\neg G(X, 0)$, as required.

2 follows by duality. □

Notice that the bottom-right entry of the table, when G evaluates to both 0 and 1, is inherited directly from the static case, Theorem 70, and cannot be strengthened for obvious reasons.

7. A proof system for entailment via non-linear graphs

In this section we give a complete inference system for Boolean Graph Logic in the style of 'deep inference': inference rules may rewrite induced subgraphs of a graph under certain situations. In order to admit a complete system, we first (conservatively) extend BGL to account for non-linearity.

7.1. Nonlinear graphs.

We now consider graphs where the same variable may occur many times as a node. To avoid ambiguity, the graphs considered until now will now be referred to as 'linear' graphs. We now reserve the set $\mathcal{V}$ for Boolean variables, equipping graphs with an explicit labelling function assigning a variable to each node:

Definition 76 ((Non-linear) graphs)**.** *A (labelled) graph is a tuple* $G = (V, E, L)$ *where* V *is an arbitrary finite set and, as expected,* $E \subseteq \binom{V}{2}$*. Furthermore* L *is a function* $V \to \mathcal{V}$*. For a set* $U \subseteq V$*, we write* $\lfloor U \rfloor := \{L(v) : v \in U\}$*.*

We may extend the notions of evaluation and entailment to non-linear graphs in a natural way.

Definition 77. *Let* G *and* H *be (non-linear) graphs and* $X \subseteq \mathcal{V}$*. We define the following notions of evaluation,*

- $G(X, 1)$ *if* $\exists S \in MC(G).\lfloor S \rfloor \subseteq X$.
- $G(X, 0)$ *if* $\exists T \in MS(G).\lfloor T \rfloor \cap X = \varnothing$.

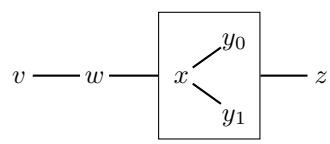
and the following notions of entailment:

- $G \underset{\wedge}{\Rightarrow} H$ *if* $\forall S \in MC(G).\exists S' \in MC(H).\lfloor S' \rfloor \subseteq \lfloor S \rfloor$.
- $G \underset{\vee}{\Rightarrow} H$ *if* $\forall T \in MS(H).\exists T' \in MS(H).\lfloor T' \rfloor \subseteq \lfloor T \rfloor$.

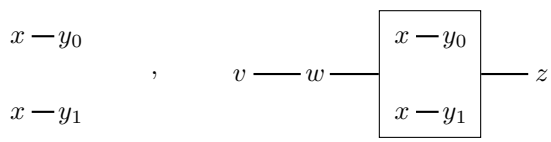
Clearly these notions admit the anlogous ones for linear graphs as special cases. What is more, they are conservative over the analogous notions for non-read-once formulas, when restricted to non-linear cographs. We do not go into detail on this point but leave the verification of this fact as an exercise to the reader.

7.2. **Deep inference and rules on modules.** Let us write $G[M]$ to distinguish a module M in a graph G. We may then write $G[M']$ for the graph where M is replaced by M' in G, retaining all the edges connected to the module.

Example 78. *Let $G[M]$ be the following graph, with only edges indicated, where M is the module $\{x, y\}$:*



If we set M' to be the graph below, left, then $G[M']$ is the graph below, right:



Formally speaking, the two occurrences of x above are different nodes with the same label. When displaying graphs visually, we do not make this distinction explicitly, but formally if we indicate multiple occurrences of the same graph G, it means that they are all label-preserving isomorphic to G.

Notice that M and M' are relation webs, and that, say, $M \underset{\wedge}{\Rightarrow} M'$. We also have that $G[M] \underset{\wedge}{\Rightarrow} G[M']$. This is no coincidence, as we will see in the next result.

Lemma 79 (Deep inference on graphs)**.** *Suppose M is a module of G and $M \underset{\star}{\Rightarrow} M'$, for $\star \in \{\vee, \wedge\}$. Then $G[M] \underset{\star}{\Rightarrow} G[M']$.*

Proof. We consider only the case when $\star = \wedge$, the case of $\star = \vee$ being dual.

Let $S \in MC(G[M])$. We have the following cases:

- If $S \cap M = \varnothing$ then also $S \in MC(G[M'])$.
- Otherwise, we may write $S = S_M \sqcup S'$ for some $S_M \in MC(M)$, by Lemma 59. Now, since $M \underset{\wedge}{\Rightarrow} M'$, we have some $S'_M \subseteq S_M$ with $S'_M \in MC(M')$. Therefore, again by Lemma 59, we have that $S'_M \sqcup S' \in MC(G[M'])$.

Thus indeed $G[M] \underset{\wedge}{\Rightarrow} G[M']$. □

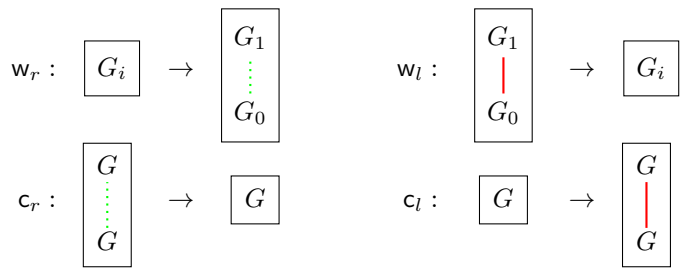
The proposition above is a generalisation of 'deep inference' reasoning on formulas, where we may operate under arbitrary alternations of $\vee$ and $\wedge$.

Definition 80 (Inference rules, systems and derivations)**.** *An* inference rule *on graphs is simply a binary relation $\to$ on graphs. A* proof system *is a set of inference rules,*[8] *and a* derivation *in a proof system is just a sequence of graphs $(G_1, \ldots, G_n)$ where each (G_i, G_{i+1}) is an instance of a rule in the system.*

Inference rules may be specified in many different ways, and we will introduce some bespoke notation in what follows in order to compactly write inference rules.

Given the proposition above, we may safely import the standard structural rules from deep inference proof theory, restricted to modules:

Definition 81 (Structural rules)**.** *The system* str *consists of the following rules:*[9]



for $i \in \{0, 1\}$. The fact that the LHS and RHS of these rules are boxed indicates that the corresponding induced subgraphs must be modules in the LHS and RHS, respectively, of an instance of the rule. There may be an ambient surrounding graph that is not indicated, but no nodes outside the module are affected by the rule. There are no further restrictions on the graphs indicated or the surrounding graph outside the module.

For comparison, we give also the formula theoretic versions of some of the rules above:

$$\begin{array}{rrcl} \mathsf{w}_r : & A_i & \to & A_0 \vee A_1 \\ \mathsf{c}_r : & A \vee A & \to & A \end{array} \qquad \begin{array}{rrcl} \mathsf{w}_l : & A_0 \wedge A_1 & \to & A_i \\ \mathsf{c}_l : & A & \to & A \wedge A \end{array} \tag{15}$$

Usually in deep inference proof theory we must operate under some equational theory, here associativity and commutativity of $\vee$ and $\wedge$. However this is implicit in the graph theoretic setting.

Definition 82 (Soundness and completeness)**.** *We say that an inference rule $\to$ is* sound *for $\underset{\star}{\Rightarrow}$, for $\star \in \{\vee, \wedge\}$, if whenever $G \to H$ we have $G \underset{\star}{\Rightarrow} H$. A system is sound whenever all its inference rules are. We say that a system is* complete *for $\underset{\star}{\Rightarrow}$ if, whenever $G \underset{\star}{\Rightarrow} H$, there is a derivation from G to H.*

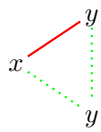
Since str was induced by a sound system on formulas, we immediately have the following from Lemma 79:

Proposition 83. str *is sound for both $\underset{\wedge}{\Rightarrow}$ and $\underset{\vee}{\Rightarrow}$.*

[8]Note that, in this presentation there is not much difference between a rule and a system, but we maintain the distinction as it is natural from the proof theory and rewriting theory points of view.

[9]The subscripts l and r are usually written as annotations $\uparrow$ and $\downarrow$ respectively, but we chose a different notation to reduce the number of arrows in use. The l and r subscripts is a reference to sides of the sequent calculus.

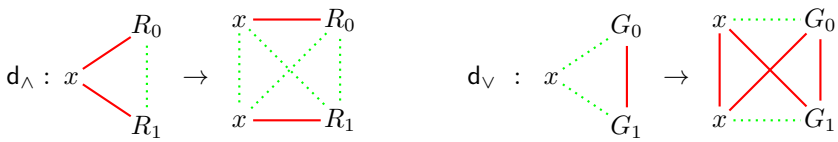
Remark 84 (Structural rules beyond modules)**.** *Despite the fact that we have restricted our deep inference rules to modules, it is not hard to see that there are variations of the rules* w_r *and* c_l *that operate on arbitrary (induced) subgraphs yet remain sound. We do not give details here, being beyond the scope of this work, but we do point out a fundamental problem with extending the other structural rules to non-modular subgraphs, in particular the* c_r *rule. Consider, for example, the following situation:*



Attempting to apply c_r *to the two* y *nodes would leave us with a choice of how to resolve the clash between the upper* x —— y *and the lower* x ····· y *.*

7.3. **Entailment-specific rules.** Notice that all the rules thus far introduced are sound for both $\underset{\vee}{\Rightarrow}$ and $\underset{\wedge}{\Rightarrow}$. Since we know that these two entailments are distinct, any complete system for either entailment must have additional rules. We introduce these in the following definition:

Definition 85. *We define the following rules,*[10]



where:

- *For* $\mathsf{d}_\wedge$*,* $\{R_0, R_1\}$ *partitions the set* $\{y : \{x, y\} \in E(LHS)\}$*, i.e.* $R_0 \sqcup R_1$ *is the set of 'red' edges of the LHS including* x*.*
- *For* $\mathsf{d}_\vee$*,* $\{G_0, G_1\}$ *partitions the set* $\{y : \{x, y\} \notin E(LHS)\}$*, i.e.* $G_0 \sqcup G_1$ *is the set of 'green' edges of the LHS including* x*.*

Neither the LHS nor the RHS of each rule need form modules, but the indicated edge-relationships must hold. There are no restrictions on the unindicated edges further to the conditions on R_0, R_1, G_0, G_1 *above. On the RHS of both rules there are two occurrences of* x*; formally, these are two different nodes with the same label* x*.*

We will soon use these rules to achieve normal forms of graphs that drive our ultimate completeness proof. The point of these rules is not only that they are sound for $\underset{\wedge}{\Rightarrow}$ and $\underset{\vee}{\Rightarrow}$, respectively, but moreover that they do not change the Boolean relation computed, as shown in the following result.

Lemma 86. *We have the following:*

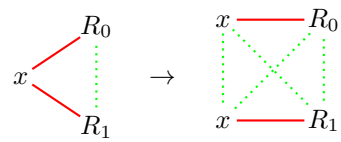
(1) If $G \underset{\mathsf{d}_\wedge}{\rightarrow} H$ *then* $\lfloor MC(G) \rfloor = \lfloor MC(H) \rfloor$*, i.e.* G *and* H *have the same maximal cliques, up to the variables occurring in them.*

(2) If $G \underset{\mathsf{d}_\vee}{\rightarrow} H$ *then* $\lfloor MS(G) \rfloor = \lfloor MS(H) \rfloor$*, i.e.* G *and* H *have the same maximal stable sets, up to the variables occurring in them.*

[10]Notice that we could have allowed x to be an arbitrary module M instead, but the following exposition is slightly simpler by restricting to this atomic version.

Proof. We prove only 1, the case of 2 being dual.

Consider the corresponding instance of $\mathsf{d}_\wedge$,



and let us call the node labelled x on the LHS v, the upper node labelled x on the RHS v_0, and the lower node labelled x on the RHS v_1. Assume all other nodes of the RHS have the same name as their corresponding nodes on the LHS.

Suppose $S \in MC(LHS)$, and notice that S must be disjoint from either R_0 or R_1, since $R_0 \cdots\cdots R_1$. We define $S' \in MC(RHS)$ with $\lfloor S' \rfloor = \lfloor S \rfloor$ as follows:

- If $v \notin S$ then we define $S' = S$;
- If $v \in S$ and $S \cap V(R_0) = \varnothing$, then we define $S' = (S \setminus \{v\}) \cup \{v_1\}$;
- If $v \in S$ and $S \cap V(R_1) = \varnothing$, then we define $S' = (S \setminus \{v\}) \cup \{v_0\}$.

By construction we have that $S \cong S'$, so S' is a clique, and maximality is immediate.

For the converse direction, notice that any maximal clique of the RHS also cannot intersect both R_0 and R_1, and furthermore cannot contain both v_0 and v_1, since $v_0 \cdots\cdots v_1$. Thus the mapping from S to S' above is a bijection $MC(LHS) \to MC(RHS)$, finishing the proof. □

The following is an immediate consequence of the preceding lemma:

Proposition 87. *We have the following:*

(1) Both $\mathsf{d}_\wedge$ and $\mathsf{d}_\wedge^{-1}$ are sound for $\underset{\wedge}{\Rightarrow}$.

(2) Both $\mathsf{d}_\vee$ and $\mathsf{d}_\vee^{-1}$ are sound for $\underset{\vee}{\Rightarrow}$.

7.4. **Reductions to DNF and CNF.** Our completeness strategy is motivated by the disjunctive and conjunctive normal forms of Boolean functions. In fact, Boolean relations may naturally be associated with a DNF, determining evaluation to 1, and a CNF, determining evaluation to 0. Let us frame these normal forms in a graph-theoretic context.

Definition 88 (DNFs and CNFs)**.** *A DNF is a graph where all maximal cliques are disjoint. A CNF is a graph where all maximal stable sets are disjoint.*

Of course, DNFs are just (non-linear) relation webs of formulas of the form $\bigvee_i \bigwedge_j x_{ij}$ and CNFs are just (non-linear) relation webs of formulas of the form $\bigwedge_i \bigvee_j x_{ij}$. We will thus identify DNFs and CNFs with their formula representations when it is convenient.

The main result of this subsection is that we may generate DNFs and CNFs of arbitrary graphs using our entailment-specific rules:

Lemma 89. *For any graph G we have the following:*

(1) There is a DNF A with $G \underset{\mathsf{d}_\wedge}{\to}^* A$

(2) There is a CNF B with $G \underset{\mathsf{d}_\vee}{\to}^* B$.

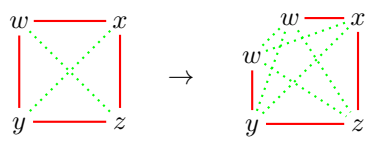
In what follows, we will only concern ourselves with DNFs and completeness for $\underset{\wedge}{\Rightarrow}$, with the case of CNFs and $\underset{\vee}{\Rightarrow}$ following by duality.

Notice that the rule $\mathsf{d}_\wedge$ bears semblance to the distibutivity rule on formulas:

$$\mathsf{d}'_\wedge : \quad A \wedge (B \vee C) \quad \Rightarrow \quad (A \wedge B) \vee (A \wedge C)$$

Indeed, the rule above is a special case of $\mathsf{d}_\wedge$ when all nodes are indicated: if $\mathsf{d}'_\wedge : A \to B$ then $\mathsf{d}_\wedge : \mathcal{W}(A) \to \mathcal{W}(B)$.[11] On the other hand, $\mathsf{d}_\wedge$ is strictly more general than $\mathsf{d}'_\wedge$, as shown in the following example.

Example 90. *Consider the following instance of* $\mathsf{d}_\wedge$*:*



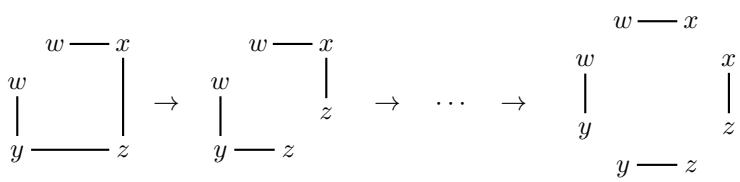
Notice that the LHS is the relation web of the formula $(w \vee z) \wedge (x \vee y)$*, while the RHS is isomorphic to the* P_5*, and so corresponds to no formula.*

However, it is not hard to see that DNFs remain the only normal forms of $\mathsf{d}_\wedge$:

Observation 91. *The only normal forms of* $\mathsf{d}_\wedge$ *are DNFs, and the only normal forms of* $\mathsf{d}_\vee$ *are CNFs.*

Proof sketch. Any graph that is not a DNF has two intersecting maximal cliques, which would form a redex for $\mathsf{d}_\wedge$. The argument for $\mathsf{d}_\vee$ is dual. □

Example 92. *Revisiting the above Example 90, we may continue applying* $\mathsf{d}_\wedge$ *as follows (now with only edges indicated):*



The first step applies $\mathsf{d}_\wedge$ *on* z*, and then there are two steps applying* $\mathsf{d}_\wedge$ *to* x *and* y *in any order. The resulting graph is a DNF. There is also a derivation from the original formula* $(w \vee z) \wedge (x \vee y)$ *to DNF by continuously applying* $\mathsf{d}'_\wedge$ *(using deep inference), but by definition that derivation only includes cographs, unlike the one above.*

We are now ready to prove the main result of this subsection.

Proof of Lemma 89. We prove only (1), since (2) follows by duality. By Observation 91 above, it suffices to show that the rule $\mathsf{d}_\wedge$ is *terminating*, i.e. that any $\mathsf{d}_\wedge$-derivation has finite length.

Recall from the proof of Lemma 86 that the number of maximal cliques remains constant for any instance of $\mathsf{d}_\wedge$. However, we have strictly reduced the number of intersections, since any maximal cliques intersecting at the duplicated variable become disjoint on the RHS. Thus any $\mathsf{d}_\wedge$ derivation must terminate, as required. □

[11] This holds also for deep versions of $\mathsf{d}'_\wedge$ if we allow $\mathsf{d}_\wedge$ to operate within modules too.

7.5. **A completeness result.** We are now ready to give our systems for both entailment relations.

Definition 93 (Systems for entailment)**.** *We define the following systems of inference rules:*

- $\underset{\wedge}{\rightarrow} := \mathsf{str} \cup \{\mathsf{d}_\wedge, \mathsf{d}_\wedge^{-1}\}$.
- $\underset{\vee}{\rightarrow} := \mathsf{str} \cup \{\mathsf{d}_\vee, \mathsf{d}_\vee^{-1}\}$.

Notice that it is immediate from Propositions 83 and 87 that $\underset{\wedge}{\rightarrow}$ and $\underset{\vee}{\rightarrow}$ are sound for $\underset{\wedge}{\Rightarrow}$ and $\underset{\vee}{\Rightarrow}$ respectively. The main goal of this section is to establish the converse result:

Theorem 94 (Completeness)**.** *We have:*

(1) If $G \underset{\wedge}{\Rightarrow} H$ then $G \underset{\wedge}{\rightarrow} H$.
(2) if $G \underset{\vee}{\Rightarrow} H$ then $G \underset{\vee}{\rightarrow} H$.

We are almost ready to prove this, but we need the following intermediate result, which is well-known in deep inference proof theory:

Lemma 95 (Completness for DNFs and CNFs)**.** *We have the following:*

(1) If a DNF A logically implies a DNF B then there is a str *derivation $A \rightarrow^* B$.*
(2) If a CNF A logically implies a CNF B then there is a str *derivation $A \rightarrow^* B$.*

Proof. We only prove (1), the case of (2) being dual. Since the statement only concerns formulas and any rule instance of str preserves cograph-ness, we will present the argument in terms of formulas, cf. (15).

Suppose we have a valid implication,

$$\bigvee_{i \in I} \bigwedge S_i \Rightarrow \bigvee_{j \in J} \bigwedge S'_j \tag{16}$$

for some sets of variables S_i and S'_j. Recall that we each $\bigwedge S_i$ and $\bigwedge S'_j$ are called *terms*. Appealing to the definition of $\underset{\wedge}{\Rightarrow}$, there must be some function $f : I \rightarrow J$ such that $\lfloor S'_{f(i)} \rfloor \subseteq \lfloor S_i \rfloor$. We thus have derivations:

$$\{\mathsf{w}_l, \mathsf{c}_l\} \ : \ \bigwedge S_i \ \rightarrow^* \ \bigwedge S'_{f(i)} \tag{17}$$

(c_l is required in case there are multiple occurrences of variables in the RHS). Applying these to each term of LHS(16) yields a DNF $\bigvee_{i \in I} \bigwedge S'_{f(i)}$. To arrive at RHS(16), we need to use the dual rules:

- If f is not surjective, e.g. $\forall i \in I. f(i) \neq j$, then we may apply w_r to recover the missing terms, e.g.:
$$\mathsf{w}_r \ : \ \bigvee_{i \in I} \bigwedge S'_{f(i)} \ \rightarrow \ \bigvee_{i \in I} \bigwedge S'_{f(i)} \ \vee \ \bigwedge S'_j$$
- If f is not injective, e.g. $f(i) = f(i') = j$ for some distinct i, i', then we may apply c_r to remove duplicate terms, e.g.:
$$\mathsf{c}_r \ : \ \bigwedge S_j \vee \bigwedge S'_j \ \rightarrow \ \bigwedge S_j$$

We have thus derived (16) in $\{\mathsf{w}_l, \mathsf{c}_l, \mathsf{w}_r, \mathsf{c}_r\}$, as required. □

We are now ready to prove the main completeness result:

Proof of Theorem 94. We prove only (1), the case of (2) following by duality. Suppose $G \underset{\wedge}{\Rightarrow} H$, and by Lemma 89 let A and B be DNFs such that:

$$G \underset{\mathsf{d}_\wedge}{\rightarrow}^* A \tag{18}$$

$$H \underset{\mathsf{d}_\wedge}{\rightarrow}^* B \tag{19}$$

Notice that, since G and A have the same maximal cliques up to variables occurring, by Lemma 86, we have that G and A are equivalent. Similarly for H and B so, since $G \underset{\wedge}{\Rightarrow} H$, we also have that A logically implies B. We may thus build the following $\underset{\wedge}{\rightarrow}$-derivation:

$$\begin{array}{lll} G & \underset{\mathsf{d}_\wedge}{\rightarrow}^* & A \quad \text{by (18)} \\ & \underset{\mathsf{str}}{\rightarrow}^* & B \quad \text{by Lemma 95} \\ & \underset{\mathsf{d}_\wedge^{-1}}{\rightarrow}^* & H \quad \text{by (19)} \end{array}$$

This concludes the proof. □

7.6. **On atomicity and linearity.** Though the structural proof theory of Boolean Graph Logic is beyond the scope of this paper, we make some observations here that suggest that BGL should enjoy decomposition theorems similar to those of deep inference proof theory [BT01]. We do not give a formal development, saving that for future work.

A key feature of deep inference is that the contraction rules, c_l and c_r may be reduced to atomic form. This is thanks to the *medial* rule:

$$\mathsf{m} \ : \ (A \wedge B) \vee (C \wedge D) \ \rightarrow \ (A \vee C) \wedge (B \vee D)$$

For instance, we may transform a c_r inference $(A \wedge B) \vee (A \wedge B) \rightarrow (A \wedge B)$ into one with smaller contraction redexes as follows:

$$\begin{array}{lll} \underline{(A \wedge B) \vee (A \wedge B)} & \underset{\mathsf{m}}{\rightarrow} & \underline{(A \vee A)} \wedge (B \vee B) \\ & \underset{\mathsf{c}_r}{\rightarrow} & A \wedge \underline{(B \vee B)} \\ & \underset{\mathsf{c}_r}{\rightarrow} & A \wedge B \end{array}$$

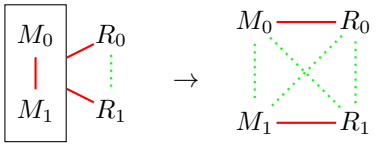
The reduction for c_l is dual. A similar reduction can be carried out in the graph theoretic setting by introducing the following medial rule for prime graphs $P[v_1, \ldots, v_n]$ (all nodes indicated):

$$P[G_1, \ldots, G_n] \vee P[G'_1, \ldots, G'_n] \quad \rightarrow \quad P[G_1 \cdots\cdots G'_1, \ldots, G_n \cdots\cdots G'_n]$$

It is not hard to verify that this rule is sound for both $\underset{\wedge}{\Rightarrow}$ and $\underset{\vee}{\Rightarrow}$, and also allows us to reduce contraction inferences to atomic form, though we omit the details here.

The medial rule above is an example of a *linear* rule: it does not duplicate or erase any nodes in a graph. At the level of formulas such rules are important, since decomposition theorems of deep inference are typically agnostic about the choice of linear rules, once all the structural rules have been made atomic, cf. [GG08]. One issue for our system $\underset{\wedge}{\rightarrow}$ (and $\underset{\vee}{\rightarrow}$) is that the rule $\mathsf{d}_\wedge$ (dually $\mathsf{d}_\vee$) is not linear.

However, it turns out that we may decompose it into atomic contraction inferences and the following linear rule:[12]



where $\{R_0, R_1\}$ partition the set of edges to the indicated module on the LHS.

It would be interesting to establish whether the incorporation of these linear rules (and perhaps others) and atomisation of the structural rules could lead to a well-behaved proof theory on arbitrary graphs, similar to deep inference.

8. Conclusions

In this work we presented a graph theoretic extension of Boolean logic that we called Boolean Graph Logic (BGL). BGL extended the semantics of Boolean logic from Boolean functions to more general Boolean relations, and we recovered a decomposition of entailment into two dual notions. The purpose of this article was to establish the fundamental theory behind BGL, for which we gave perspectives via complexity (Section 4), games (Section 6) and proofs (Section 7).

In future work we are most interested in developing the structural proof theory of BGL, building on the discussion of Section 7.6 to establish *decomposition theorems*, à la deep inference. Such an investigation would also help compare BGL with the approach of [AHS20], complementary to a semantic investigation. Conversely, it would be interesting to see how the logic of [AHS20] may be extended by structural rules, which can be problematic for the 'splitting' technique there used.

It would also be interesting to examine how to incorporate forms of negation and implication natively into BGL. For this it would be natural to consider the behaviour of analogous connectives from Computability Logic, cf. [Jap17].

References

[ABG06a] D Andrade, Endre Boros, and Vladimir Gurvich. On graphs whose maximal cliques and stable sets intersect. Technical report, 2006.

[ABG06b] Diogo V Andrade, Endre Boros, and Vladimir Gurvich. On graphs whose maximal cliques and stable sets intersect. 2006.

[AHS20] Matteo Acclavio, Ross Horne, and Lutz Strassburger. Logic beyond formulas: A graphical linear proof system. Accepted to LICS 2020, 2020.

[AKV04] Albert Atserias, Phokion G. Kolaitis, and Moshe Y. Vardi. Constraint propagation as a proof system. In Mark Wallace, editor, *Principles and Practice of Constraint Programming - CP 2004, 10th International Conference, CP 2004, Toronto, Canada, September 27 - October 1, 2004, Proceedings*, volume 3258 of *Lecture Notes in Computer Science*, pages 77–91. Springer, 2004.

[BDK20] Sam Buss, Anupam Das, and Alexander Knop. Proof complexity of systems of (non-deterministic) decision trees and branching programs. In Maribel Fernández and Anca Muscholl, editors, *28th EACSL Annual Conference on Computer Science Logic, CSL 2020, January 13-16, 2020, Barcelona, Spain*, volume 152 of *LIPIcs*, pages 12:1–12:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[BIK+94] Paul Beame, Russell Impagliazzo, Jan Krajícek, Toniann Pitassi, and Pavel Pudlák. Lower bound on hilbert's nullstellensatz and propositional proofs. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 794–806. IEEE Computer Society, 1994.

[12]This observation was made by Lutz Strassburger.

[Bla92] Andreas Blass. A game semantics for linear logic. *Ann. Pure Appl. Log.*, 56(1-3):183–220, 1992.

[BT01] Kai Brünnler and Alwen Fernanto Tiu. A local system for classical logic. In R. Nieuwenhuis and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, volume 2250 of *Lecture Notes in Computer Science*, pages 347–361. Springer-Verlag, 2001.

[Cal16] Cameron Calk. A graph-theoretic extension of boolean logic. 2016. `http://www.anupamdas.com/graph-bool.pdf`.

[CCT87] William J. Cook, Collette R. Coullard, and György Turán. On the complexity of cutting-plane proofs. *Discret. Appl. Math.*, 18(1):25–38, 1987.

[CH11a] Y. Crama and P.L. Hammer. *Boolean Functions: Theory, Algorithms, and Applications*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2011.

[CH11b] Yves Crama and Peter L Hammer. *Boolean functions: Theory, algorithms, and applications*. Cambridge University Press, 2011.

[CI98] Alain Cournier and Pierre Ille. Minimal indecomposable graphs. *Discrete Mathematics*, 183(1):61 – 80, 1998.

[Das13] Anupam Das. Rewriting with linear inferences in propositional logic. In Femke van Raamsdonk, editor, *24th International Conference on Rewriting Techniques and Applications (RTA)*, volume 21 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 158–173. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2013.

[Das19] Anupam Das. Complexity of evaluation and entailment in Boolean Graph Logic. 2019. `www.anupamdas.com/complexity-graph-bool-note.pdf`.

[DS15] Anupam Das and Lutz Straßburger. No complete linear term rewriting system for propositional logic. In Maribel Fernández, editor, *26th International Conference on Rewriting Techniques and Applications (RTA 2015)*, volume 36 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 127–142, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[DS16] Anupam Das and Lutz Straßburger. On linear rewriting systems for boolean logic and some applications to proof theory. *Logical Methods in Computer Science*, 12(4):9:1–27, 2016.

[ER90] Andrzej Ehrenfeucht and Grzegorz Rozenberg. Primitivity is hereditary for 2-structures. *Theoretical Computer Science*, 70(3):343–358, 1990.

[Gal67] Tibor Gallai. Transitiv orientierbare graphen. *Acta Mathematica Hungarica*, 18(1-2):25–66, 1967.

[GG08] Alessio Guglielmi and Tom Gundersen. Normalisation control in deep inference via atomic flows. *Logical Methods in Computer Science*, 4(1):9:1–36, 2008.

[GGCH11] Martin C. Golumbic, Vladimir Gurvich, Yves Crama, and Peter L. Hammer. *Read-once functions*, page 448486. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2011.

[Gir87] Jean-Yves Girard. Linear logic. *Theor. Comput. Sci.*, 50:1–102, 1987.

[Gug07] Alessio Guglielmi. A system of interaction and structure. *ACM Transactions on Computational Logic*, 8(1):1:1–64, 2007.

[Gug15] Alessio Guglielmi. Deep inference. In David Delahaye and Bruno Woltzenlogel Paleo, editors, *All About Proofs, Proofs for All*, volume 55 of *Mathematical Logic and Foundations*, pages 164–172. College Publications, 2015.

[Gur77] V. A. Gurvich. Repetition-free boolean functions. *Uspekhi Matematicheskikh Nauk*, 32(1):183–184, 1977.

[Gur82] Vladimir Alexander Gurvich. On the normal form of positional games. In *Doklady Akademii Nauk*, volume 264, pages 30–33. Russian Academy of Sciences, 1982.

[HP10] Michel Habib and Christophe Paul. A survey of the algorithmic aspects of modular decomposition. *Computer Science Review*, 4(1):41–59, 2010.

[Hug06] Dominic J.D. Hughes. Proofs Without Syntax. *Annals of Mathematics*, 143(3):1065–1076, November 2006.

[IV14] P. Ille and R. Villemaire. Recognition of prime graphs from a prime subgraph. *Discrete Mathematics*, 327:76 – 90, 2014.

[Jap05] Giorgi Japaridze. Introduction to cirquent calculus and abstract resource semantics. *CoRR*, abs/math/0506553, 2005.

[Jap17] Giorgi Japaridze. Elementary-base cirquent calculus I: parallel and choice connectives. *CoRR*, abs/1707.04823, 2017.

[Jeř04] Emil Jeřábek. Dual weak pigeonhole principle, Boolean complexity, and derandomization. *Annals of Pure and Applied Logic*, 129:1–37, 2004.

[Jr.82] Nuel D. Belnap Jr. Display logic. *J. Philosophical Logic*, 11(4):375–417, 1982.

[Mil84] Dale Miller. Expansion tree proofs and their conversion to natural deduction proofs. In Robert E. Shostak, editor, *7th International Conference on Automated Deduction, Napa, California, USA, May 14-16, 1984, Proceedings*, volume 170 of *Lecture Notes in Computer Science*, pages 375–393. Springer, 1984.

[NS18] Lê Thành Dung Nguyên and Thomas Seiller. Coherent interaction graphs. In Thomas Ehrhard, Maribel Fernández, Valeria de Paiva, and Lorenzo Tortora de Falco, editors, *Proceedings Joint International Workshop on Linearity & Trends in Linear Logic and Applications, Linearity-TLLA@FLoC 2018, Oxford, UK, 7-8 July 2018*, volume 292 of *EPTCS*, pages 104–117, 2018.

[RS19] Benjamin Ralph and Lutz Straßburger. Towards a combinatorial proof theory. In Serenella Cerrito and Andrei Popescu, editors, *Automated Reasoning with Analytic Tableaux and Related Methods - 28th International Conference, TABLEAUX 2019, London, UK, September 3-5, 2019, Proceedings*, volume 11714 of *Lecture Notes in Computer Science*, pages 259–276. Springer, 2019.

[Sip12] Alvin Sipraga. An automated search of linear inference rules. 2012. `http://arcturus.su/mimir/autolininf.pdf`.

[Str12] Lutz Straßburger. Extension without cut. *Annals of Pure and Applied Logic*, 163(12):1995–2007, 2012.

[SU08] Marcus Schaefer and Christopher Umans. Completeness in the polynomial-time hierarchy: A compendium. 2008. `http://ovid.cs.depaul.edu/documents/phcom.pdf`.

[Ter03] Terese. *Term rewriting systems*, volume 55 of *Cambridge tracts in theoretical computer science*. Cambridge University Press, 2003.

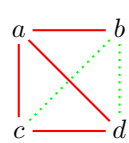
[War19] Tim Waring. A graph-theoretic extension of boolean logic. 2019. `http://anupamdas.com/thesis_tim-waring.pdf`.

Appendix A. Deterministic and total graphs are P_4-free

In this section we give a self-contained proof of Theorem 37, showing that a graph that is deterministic and total is P_4-free. We need to introduce an additional concept first.

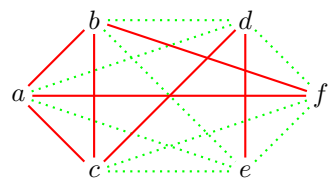
Definition 96. *Let G be a graph and $Y \subseteq V(G)$. A* selection *with respect to Y is a set $Sel = \{T_x \mid T_x \in MS(G)$ and $T_x \cap Y = \{x\}\}$.*
We call a selection w.r.t Y a covering *if there is a $D \in MS(G)$ with $D \subseteq \cup_{x \in Y} T_x$ and $D \cap Y = \varnothing$.*
We call a selection w.r.t Y a non-covering *if it is not a covering.*

Example 97. *If we choose Y so that Y is a clique, then a covering always exists. On the other hand, if we choose $Y = V(G)$, then a selection w.r.t Y does not exist. Take the following simple example: Let G be the following graph:*



Then G is deterministic, and thus CIS. Let $Y = \{a, c, d\}$. Then Y is a maximal clique, and a selection exists with $Sel = \{\{a\}, \{b, c\}, \{b, d\}\}$.

For a less trivial example, take the following graph:



Let $Y = \{c, e\}$. Then the only stable set intersecting c is $\{c, e, f\}$, but that also intersects e, so there is no selection. Now let $Y = \{b, c, d\}$. Then we have the selection $Sel = \{\{b, d\}, \{c, e, f\}, \{d, a\}\}$. Not only is it a selection, it is also covering, because we have $\{a, e\} \subseteq V(G) = \{b, d\} \cup \{c, e, f\} \cup \{d, a\}$, and $\{a, e\} \in MS(G)$.

The following Lemma sheds some more light on coverings, and is the key piece to proving the theorem:

Lemma 98. *Let G be a total graph, and $Y \subseteq V(G)$ such that Y is not a clique. Then every selection w.r.t Y is covering.*

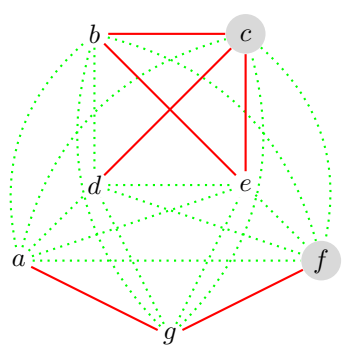
Proof. We prove the contrapositive. Assume there is a non-covering selection $Sel = \{T_x \mid T_x \in MS(G) \text{ and } T_x \cap Y = \{x\}\}$ w.r.t. Y. Define the set

$$B := (V(G) \setminus \bigcup_{x \in Y} T_x) \cup Y$$

Notice that B intersects every $T \in MS(G)$: If T is a maximal stable set that intersects Y, then, as $Y \subseteq B$, T also intersects B. If T is a maximal stable set that doesn't intersect Y, then, because Sel is not a covering, we have $T \nsubseteq \cup_{x \in Y} T_x$, so T intersects B, by the definition of B.
There is no maximal stable set disjoint from B, so G doesn't evaluate B to 0. G is total, so $e_G(B, 1)$, i.e. there exists a $S \in MC(G)$ with $S \subseteq B$.

By the definition of the T_x's, we get $T_x \cap Y = \{x\}$, and by the definition of B therefore also $T_x \cap B = \{x\}$ for every $x \in Y$. We have $T_x \in MS(G)$ for every $x \in Y$, and $S \in MC(G)$, so, because G is deterministic, by the CIS property we get $|S \cap T_x| = 1$ for all $x \in Y$. Because S is contained in B, we get $S \cap T_x = \{x\}$ for all $x \in Y$. So $Y \subseteq S$. □

Example 99. *Take the following graph, and let $Y = \{c, f\}$:*

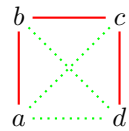


Then Y is not a clique. We have $MS(G) = \{\{g, c\}, \{gde\}, \{gdb\}, \{acf\}, \{abdf\}, \{aedf\}\}$. Then with $T_c = \{g, c\}$, $T_f = \{a, b, d, f\}$, the set $Sel = \{T_c, T_f\}$ is a selection.

We have $\cup_{x\in Y} T_x = T_c \cup T_f = \{a, b, c, d, f\}$. *Also,* $\{g, d, b\}$ *is a maximal stable set with* $\subseteq \{a, b, c, d, f\}$ *and* $\{g, d, b\} \cap Y = \varnothing$. *Therefore, Sel is not only a selection, but also a covering.*

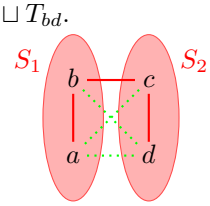
We now have all the results we need to prove the characterisation of Boolean functions.

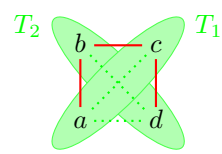
of Theorem 37. We prove the left-right implication. Assume G has a P_4 generated by the nodes $\{a, b, c, d\}$ like seen below.



We can extend the edges $\{a, b\}, \{c, d\}$ to maximal cliques $S_1, S_2 \in MC(G)$.

We write $S_1 = \{a, b\} \sqcup S_{ab}$, $S_2 = \{c, d\} \sqcup S_{cd}$, where $S_{ab}, S_{c,d}$ are (possibly empty) sets of nodes, and $\sqcup$ denotes the disjoint union. Likewise, we extend the edges $\{a, c\}, \{b, d\}$ to maximal stable sets T_1, T_2, and write $T_1 = \{a, c\} \sqcup T_{ac}$, $T_2 = \{b, c\} \sqcup T_{bd}$.





Notice that by the CIS property, we get $|S_1 \cap T_1| = |S_1 \cap T_2| = |S_2 \cap T_1| = |S_2 \cap T_2| = 1$, so we have $S_1 \cap T_1 = \{a\}$, $S_1 \cap T_2 = \{b\}$, $S_2 \cap T_1 = \{c\}$, $S_2 \cap T_2 = \{d\}$. Therefore, we can easily check that the three sets $i)$ $\{a, b, c, d\}$, $ii)$ $S_{ab} \cup S_{cd}$, $iii)$ $T_{ac} \cup T_{bd}$ are all pairwise disjoint:

We show that $i), ii)$ are disjoint:
By definition, $a, b \notin S_{ab}$. Without loss of generality, assume $a \in S_{cd}$. Then $|S_2 \cap T_1| = 2$, which is a contradiction. So $a, b \notin S_{cd}$. A completely symmetric argument shows that $c, d \notin S_{ab}$, and therefore the two sets are disjoint.
The sets $i), iii)$ are disjoint by the exact same argument.

To show that $ii), iii)$ are disjoint, notice that

$$(S_{ab} \cup S_{cd}) \cap (T_{ac} \cup T_{bd}) = (S_{ab} \cap T_{ac}) \cup (S_{ab} \cap T_{bd}) \cup (S_{cd} \cap T_{ac}) \cup (S_{cd} \cap T_{bd})$$

Due to CIS, the only candidates for these intersections would be a, b, c, d, but by the previous observations, these cannot be contained in the intersection. Thus, each of the four intersections must be empty, and therefore $ii), iii)$ must be disjoint.

The set $\{a, d\}$ is not a clique, so by the previous lemma, every selection with respect to it is covering. Notice that the set $Sel = \{T_1, T_2\}$ is a selection w.r.t to $\{a, d\}$ and is therefore covering. So there is a $D \in MS(G)$ with $D \cap \{a, d\} = \varnothing$ and $D \subseteq T_1 \cup T_2$. By the previous observation, we have $(T_1 \cup T_2) \cap S_1 = \{a, c\}$, and $(T_1 \cup T_2) \cap S_2 = \{b, d\}$. D is a maximal stable set, so by CIS, $|D \cap S_1| = |D \cap S_2| = 1$, so, because $a, d \notin D$, we get $D \cap S_1 = \{c\}$, and $D \cap S_2 = \{b\}$.

So we get $b, c \in D \in MS(G)$, which is a contradiction, because there is a red edge between b and c. $\square$