Master WIP

Kjetil Midtgarden Golid

April 6, 2016

Contents

Chapter 1

Introduction

1.1 Paradoxes

A theory in propositional logic is semantically *inconsistent* if it has no model, i.e., there exists no variable assignment making all formulae in the theory true. Consider the following example:

$$x \wedge \neg x \tag{1.1}$$

While a sentence like $\neg a \rightarrow b$ can be satisfied by, for instance, letting both a and b be true, no such assignment can be made for the statement above. The statement is therefore inconsistent.

A paradox is usually informally defined as something along the lines of *"a statement that can be neither true nor false"*. We can immediately note one thing from this intuitive definition: Since no paradoxes can be true, all paradoxes are, by definition, inconsistent. It is however not the case that all inconsistent theories are paradoxes. Just consider the inconsistent statement, $x \wedge \neg x$ again: this statement simply seems false, and not paradoxical.

A different view is that a paradox is a *dialetheia*, a sentence that is *both* true and false[1]. We will however not spend much time exploring these philosophical differences, as this is not a philosophical paper and it won't change much for our definitions.

The liar sentence is probably the most famous example of a paradox:

> "This sentence is false."

If the statement is true, then the statement is false, but if the statement is false, then the statement is true. It can thus neither be true nor false, since both lead to a contradiction.

Notice how the liar sentence is a statement about other statements (in this case itself). A collection of statements where some of them may refer to themselves or other statements, is called a *discourse* in [2], which we willl folow. In order to represent such discourses, we need a formal way of referencing other statements within statements. In propositional logic, this can be done simply by giving statements "names" in the form of adding fresh variables with equivalences to their correponding statements[1].

Consider the examples below; with normal propositional statements to the left, together with the corresponding *named* statements to the right.

$$a \qquad\qquad x_1 \leftrightarrow a \tag{1.2}$$

$$a \wedge \neg a \qquad\qquad x_2 \leftrightarrow a \wedge \neg a \tag{1.3}$$

$$a \vee \neg a \qquad\qquad x_3 \leftrightarrow a \vee \neg a \tag{1.4}$$

Labelling statements in this way obviously changes their truth value. Even though we have one consistent, one inconsistent and one tautological statement on the left, all the statements become consistent after they have been named. This is because we can find truth values for both x_1, x_2 and x_3 that match their corresponding statements, making each equivalence true. In other words, the truth value of a labelled statement does not refer to whether the unnamed statement is consistent, but whether or not a truth value can be found at all. Since we have defined a paradox to be a statement that is neither true nor false, we get that a statement is paradoxical if and only if labelling it makes it inconsistent.

Consider the liar sentence. We are able to label it and then use its label in order to reference itself. Doing this gives us the following result:

$$x \leftrightarrow \neg x \tag{1.5}$$

This statement is obviously inconsistent, making it a paradox by our definition.

The correspondence between paradoxical discourses and inconsistent sets of labelled statements motivates us to study labelled statements closer, trying to uncover patterns that prevent inconsistencies and thus paradoxes in the represented discourse.

1.2 Graph Normal Form

A propositional theory over a set of variables Σ is in *graph normal form (GNF)*[3] if all its formulae have the following form:

$$x \leftrightarrow \bigwedge_{y \in I_x} \neg y \tag{1.6}$$

[1] TODO: bad wording?

where $I_x \subseteq \Sigma$ and such that every variable occurs exactly once on the left of $\leftrightarrow$ across all the formulae in the theory.

There is a simple translation from a theory in conjunctive normal form to an equisatisfiable theory in graph normal form (shown in Chapter 4.1). Since conjunctive normal form is expressively complete, we get that any propositional theory, including our labelled statements, has an equisatisfiable GNF theory.

This means that any discourse can be represented with a GNF theory such that the discourse is paradoxical if and only if the GNF theory is inconsistent. This GNF representation of discourses is interesting to us because GNF theories have a tight correspondence to graphs. This correspondence lets us not only decide the satisfiability of a discourse theory by looking at certain features in the corresponding graph, but the graph also provides us with the actual models of the discourse, if they exist.

In order to express this logic/graph correspondence, we first need to establish some graph terminology.

1.3 Graphs, Kernels and Solutions

A directed graph (digraph) is a pair $\mathbf{G} = \langle G, N \rangle$ where G is a set of vertices while $N \subseteq G \times G$ is a binary relation representing the edges in $\mathbf{G}$. We use the notation $N(x)$ to denote the set of all vertices that are targeted by edges originating in x (*successors* of x). Similarly, $N^-(x)$ denotes the set of all vertices with edges targeting x (*predecessors* of x). We define these two functions formally as follows:

$$N(x) := \{y \mid (x, y) \in N\} \tag{1.7}$$

$$N^-(x) := \{y \mid (y, x) \in N\} \tag{1.8}$$

The number of successors a vertex has is often called the *out-degree* of that vertex.

A *simple path* is a sequence of distinct vertices $x_1, x_2, \ldots, x_n$ such that for each consecutive pair x_i, x_{i+1} from the sequence, we have $(x_i, x_{i+1}) \in N$. We say that two paths are *disjoint* if they do not share any vertices (possibly with the exception of their initial vertices). We let the length of a path be defined by its number of edges.

The functions N and N^- can be extended pointwise to sets in the following way:

$$N(X) = \bigcup_{x \in X} N(x) \tag{1.9}$$

$$N^-(X) = \bigcup_{x \in X} N(x) \tag{1.10}$$

A kernel is a set of vertices $K \subseteq G$ such that:

$$G \setminus K = N^-(K) \tag{1.11}$$

The above equivalence can be split into two inclusions to be more easily understood:

$G \setminus K \subseteq N^-(K)$, saying that each vertex outside the kernel has an edge into the kernel (K is *absorbing*). A consequence of this is that a kernel has to be non-empty, unless the graph is empty.

$N^-(K) \subseteq G \setminus K$, saying that each edge targeting a vertex within the kernel has to come from outside, thus no two vertices in the kernel are connected by an edge (K is *independent*).

Kernels heve been studied over several decades, not only in graph theory, but also within the fields of game theory and economics. The concept was first defined and used by von Neumann and Morgenstern in [4]. In a graph representing some sort of a turn-based game, where vertices are states and edges are transitions between states, one can often work out winning strategies whenever one finds a kernel in the graph. Whenever one is outside of the kernel, one always has the possibility of moving inside the kernel (since the kernel is absorbing), while inside the kernel one *has* to move out of it (since the kernel is independent). If you are the player with the choice outside the kernel, you can control the game and choose to stabilize it by always moving into the kernel, forcing the opponent to move out again on the next turn.

Deciding the existence of kernels in finite graphs has been shown to be an NP-complete problem[5]. This should not be surprising, since we are in the middle of showing the equivalence between this problem and the problem of finding satisying models of PL theories (SAT), which we know is NP-complete [2].

We will get the correspondence between models of a discourse theory and kernels in a graph through an alternative, equivalent kernel definition called a *solution*.

Given a directed graph $\mathbf{G} = \langle G, N \rangle$, an assignment $\alpha \in 2^G$ is a function mapping every vertex in the graph to either 0 or 1. A *solution* is an assignment α such that for all $x \in G$:

$$\alpha(x) = 1 \iff \alpha(N(x)) = \{0\} \tag{1.12}$$

This means that for any node x, if x is assigned 1, then all its successors have to be assigned 0, and if x is assigned 0, then there has to exist a node assigned 1 among its successors. A consequence of this definition is that all sink nodes (nodes with no outgoing edges) in the graph have to be assigned 1, since it vacuously does not point to any node assigned 1. We use the notation $sol(\mathbf{G})$ to denote the set of all solutions of the graph $\mathbf{G}$.

We get the equivalence between kernels and solutions from the following two facts: Given a solution, the set of all vertices assigned 1 is a kernel. Given a kernel, the function assigning 1 to all vertices in the kernel and 0 to the rest, is a solution.

[2] We are concerned with SAT over infinitary formulae in this paper, not the finite version from computer science.

1.4 Discourse Theories and Digraphs

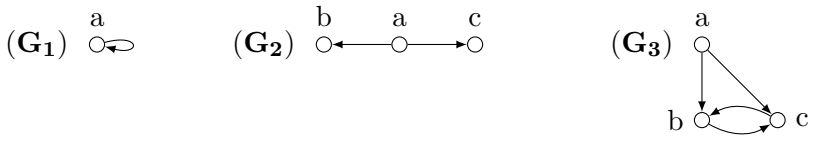
As mentioned earlier, there is a close connection between (1) models of a discourse, (2) kernels of a graph and (3) solutions of a graph. While we have the equivalence between (2) and (3), we will now look at two functions connecting (1) and (2). This correspondence was shown by Roy T. Cook in [6]. We get the following definitions from [3]

$\mathcal{T}$: translating a digraph $\mathbf{G}$ into a corresponding theory $\mathcal{T}(\mathbf{G})$ such that $sol(\mathbf{G}) = mod(\mathcal{T}(G))$.

$\mathcal{G}$: translating a theory T into a corresponding digraph $\mathcal{G}(T)$ such that $mod(T) = sol(\mathcal{G}(T))$.

Given any digraph $\mathbf{G}$ we get the theory $\mathcal{T}(\mathbf{G})$ by taking, for each $x \in G$, the formula $x \leftrightarrow \bigwedge_{y \in N(x)} \neg y$ where $\bigwedge \emptyset = 1$.

Example 1.



The above graphs have the following theories:

$$\mathcal{T}(\mathbf{G_1}) = \{a \leftrightarrow \neg a\} \tag{1.13}$$
$$\mathcal{T}(\mathbf{G_2}) = \{a \leftrightarrow (\neg b \wedge \neg c), b, c\} \tag{1.14}$$
$$\mathcal{T}(\mathbf{G_3}) = \{a \leftrightarrow (\neg b \wedge \neg c), b \leftrightarrow \neg c, c \leftrightarrow \neg b\} \tag{1.15}$$

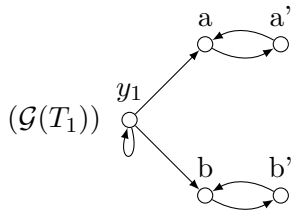
The fact that $sol(\mathbf{G}) = mod(\mathcal{T}(G))$ is shown in [3]. Allthough not proving it, we can observe that $\mathbf{G_1}$ has no solution, just like its corresponding theory $\mathcal{T}(\mathbf{G_1})$ has no models. $\mathbf{G_2}$ has one solution, where one assigns $a = 0, b = 1, c = 1$. This assignment also works as the only model for $\mathcal{T}(\mathbf{G_2})$. In $\mathbf{G_3}$, we get two solutions, both with a assigned 0, but with 0 and 1 distributed on b and c. These are also the only two models of $\mathcal{T}(\mathbf{G_3})$.

Conversely, given any discourse theory T (in fact, this will work given any PL theory, since we can translate CNF to GNF), we can derive the corresponding graph $\mathcal{G}(T)$ in the following way: All variables in the theory are vertices, and for each formula $x \leftrightarrow \bigwedge_{y \in I_x} y$ make a directed edge $\langle x, y \rangle$ for each $y \in I_x$.

Example 2.

$$(a \leftrightarrow \neg a'), (a' \leftrightarrow \neg a), (b \leftrightarrow \neg b'), (b' \leftrightarrow \neg b), (y_1 \leftrightarrow (\neg a \wedge \neg b \wedge \neg y_1)) \tag{1.16}$$

Using $\mathcal{G}$ on the above GNF theory gives us the following graph:



Again, will we not be proving the correspondence, but notice that T_1 has one model satisfying it, where $a = 0$. $\mathcal{G}(T_1)$ also has one solution, namely where $a = 0, a' = 1, y_1 = 0$. T_2 has three solutions, where either a, b or both are assigned 1. This reflects onto the graph since y_1 has to be assigned 0, thus forcing a or b to be assigned 1. The fact that $\mathcal{G}$ gives us the correspondence we are looking for is shown in [3].

With the problem of solutions in the graph being equivalent with SAT, we get our final equivalence between kernels in the graph and models of the theory. This equivalence connects logic with graph theory: A graph has a kernel if and only if its corresponding discourse is consistent (non-paradoxical). A theory is paradoxical if and only if its corresponding graph has no kernels. Because of this tight link, we will often refer to graphs without kernels as paradoxical graphs.

The applicability of kernels should by now be obvious. In the next section we will review some of the various findings within Kernel Theory, and especially the findings related to infinitary graphs.

1.5 Results in Kernel Theory

The end goal within Kernel Theory is ultimately to develop an easy way of answering the question "Does this digraph have a kernel?" no matter the graph, and no matter the answer. As of today, we are not quite there, but a lot of work has been put into trying to identify special circumstances under which one is guaranteed to have (or guaranteed to not have) a kernel in the given graph. One of the results is the theorem proven by Moses Richardson in 1953:

Theorem 3 ([7])**.** *If D is a finitary[3] digraph without odd cycles, then D has a kernel.*

Intuitively, one might be tempted to believe that *all* digraphs without odd cycles have kernels, but this is not the case. Until now, all our paradoxes have been statements that – directly or indirectly – have been referring back to themselves (giving cycles in the graph) and thus causing a logical conflict, and it is hard to imagine any other way to construct paradoxical statements. The following construction will however reveal our lack of imagination.

[3]In a finitary graph, every vertex has a finite number of out-neighbors; the graph has finite branching.

The *Yablo Graph*[8] is an example of an acyclic graph with no kernel. It is constructed with an infinite set of vertices $\{x_i \mid i \in \mathbb{N}\}$ and a set of edges N such that $\langle x_i, x_j \rangle \in N$ iff $i < j$.

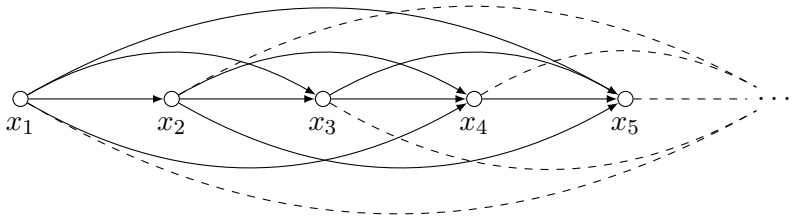


Figure 1.1: The Yablo Graph

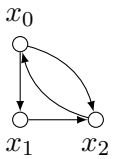
Since there exist no two numbers $x, y \in \mathbb{N}$ such that $x < y$ and $y < x$, we get that the Yablo graph indeed is acyclic[4]. Furthermore, since any natural number has infinitely many numbers strictly larger than it, we get that all the vertices are infinitely branching, making the Yablo graph infinitary.

The discourse represented by the Yablo graph would – informally – be the situation with an infinite number of statements, all saying "Every statement after this statement is false".

We will later show formally that the Yablo-graph is indeed without a kernel, but for now the following explanation should suffice.

Let us assume that the Yablo-graph *has* a kernel and that the vertex x_a is in it. Then all the vertices to the right of x_a are necessarily outside of the kernel, including x_{a+1}. But if x_{a+1} is outside of the kernel, it has to point to a vertex on the inside. This is now impossible, since the out-neighborhood of x_{a+1} is a subset of the out-neighborhood of x_a. Since x_a was chosen without any restrictions, no vertex can be inside the kernel, making it empty. Since no kernel can be empty, we have a contradiction, making the Yablo-graph without a kernel.

One thing should be mentioned at this point, the inverse of Richardson's statement is not valid; neither odd cycles nor infinitely branching vertices *entail* that their respective graphs are paradoxical. The following two graphs illustrate this point:



The above graph contains an odd cycle, but the singleton set $\{x_2\}$ is a kernel.

[4] TODO: This is a bit over-simplified

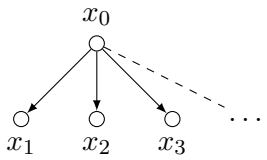


Figure 1.2

The above graph has an infinitely branching vertex x_0, but the infinite set $\{x_i \mid x > 0\}$ is a kernel.

It is shown in [6] that every digraph (with at least one edge) can be transformed into a infinitary dag such that the assignment α is a solution to the created dag if and only if it is a solution to the original digraph. This means that for any finitary graph that is paradoxical by the virtue of having an odd cycle, there is an infinitary, *acyclic* digraph that is also paradoxical. So if one is trying to find ways to identify paradoxical graphs, one does only need to look at dags.

This result will be of great importance to us, enabling us to narrow our search space when looking for paradoxes.

1.6 Recognizing dags without kernels

Knowing that any graph can be translated to an equisatisfiable dag, the challenge is now to find sufficient conditions for dags to have kernels, even weaker than the one proved by Richardson (the fact that any finitary dag has a kernel is a direct consequence of Richardson's Theorem).

Michał Walicki has proposed the following thesis:

> "If a dag has no kernel then it has a ray with infinitely many vertices dominating it."

Some terminology (given a graph $\mathbf{G} = \langle G, N \rangle$): A *ray* is an semi-infinite path, i.e. an infintie sequence $(x_1, x_2, \dots)$ of distinct vertices of G such that $(x_i, x_{i+1}) \in N$ for each i.

A vertex x_0 *dominates* a set of vertices $Y \subseteq G$ if there exists an infinite number of disjoint paths from x_0 to distinct vertices of Y.

The contrapositive of Walicki's thesis suggests a condition for a kernel. This condition is weaker than the one from Richardson's Theorem, since a dag having a ray with infinitely many vertices dominating it implies that the dag is infinitary.

1.7 Resolving GNF-theories

In this section, we present an inference system introduced by Walicki in [] which handles clausal theories induced from GNF-theories.

Recall that a thoeory written in GNF has formulae of the following form:

$$x \leftrightarrow \bigwedge_{i \in I_x} \neg y_i \tag{1.17}$$

Using simple operations only, one can manipulate these formulae into an equivalent set of clauses. We start by writing the above bi-implication as two implications:

$$x \rightarrow \bigwedge_{i \in I_x} \neg y_i \qquad \text{and} \qquad x \leftarrow \bigwedge_{i \in I_x} \neg y_i \tag{1.18}$$

The first implication can be rewritten in the following way:

$$x \rightarrow \bigwedge_{i \in I_x} \neg y_i \quad \Leftrightarrow \quad \neg x \vee \bigwedge_{i \in I_x} \neg y_i \quad \Leftrightarrow \quad \bigwedge_{i \in I_x} (\neg x \vee \neg y_i) \quad \Leftrightarrow \quad \bigwedge_{i \in I_x} \neg (x \wedge y_i) \tag{1.19}$$

The second implication can be rewritten in the following way:

$$x \leftarrow \bigwedge_{i \in I_x} \neg y_i \quad \Leftrightarrow \quad x \vee \neg \left(\bigwedge_{i \in I_x} \neg y_i \right) \quad \Leftrightarrow \quad x \vee \bigvee_{i \in I_x} y_i \tag{1.20}$$

By splitting the conjunction from the first implication up into individual clauses, we get the following two kinds of clauses for every variable x in the GNF theory:

$$\text{OR-clause:} \quad x \vee \bigvee_{i \in I_x} y_i \tag{1.21}$$

$$\text{NAND-clauses:} \quad \neg(x \wedge y_i), \text{ for every } i \in I_x \tag{1.22}$$

We will treat both the OR-clauses and the NAND-clauses as sets of atoms, denoting NAND-clauses $\neg(x \wedge y)$ as $\overline{xy}$ and OR-clauses $x \vee y_1 \vee y_2 \vee y_3$ as $xy_1y_2y_3$. This enables us to state things like $\overline{xy} \subset \overline{xyz}$. A theory will – as expected – be a set of OR- and NAND-clauses.

If we interpret the initial GNF-theory as a graph $\mathbf{G} = \langle G, N \rangle$, for every vertex $x \in G$, there will be one OR-clause $\{x\} \cup N(x)$ and for every edge $\langle x, y \rangle \in N$ there will be a NAND-clause $\overline{xy}$. The graphs from Example 1 will have the following clausal theories:

$$\mathcal{T}(\mathbf{G_1}) = \{a, \overline{a}\} \tag{1.23}$$

$$\mathcal{T}(\mathbf{G_2}) = \{abc, b, c, \overline{ab}, \overline{ac}\} \tag{1.24}$$

$$\mathcal{T}(\mathbf{G_3}) = \{abc, bc, \overline{ab}, \overline{bc}\} \tag{1.25}$$

Further notation: $A \subseteq G$ denotes an OR-clause while $\overline{A} \subseteq G$ denotes a NAND-clause. Given a graph $\mathbf{G} = \langle G, N \rangle$, we denote the set of all NAND-clauses induced from the graph as NAND and all induced OR-clauses as OR. The combined set $\Gamma = NAND + OR$ will be our initial clauses in the inference system.

1.7.1 The inference system

We consider the following inference system, but we will focus mainly on proofs using the Axioms together with the (Rneg) rule.

$$\text{(Ax)} \quad \Gamma \vdash C, \quad \text{for } C \in \Gamma \tag{1.26}$$

$$\text{(Rneg)} \quad \frac{\{\Gamma \vdash \overline{a_i A_i} \mid i \in I\} \quad \Gamma \vdash \{a_i \mid i \in I\}}{\Gamma \vdash \overline{\bigcup_{i \in I} A_i}} \tag{1.27}$$

$$\text{(Rpos)} \quad \frac{\Gamma \vdash A \quad \{\Gamma \vdash B_i K_i \mid i \in I\} \quad \{\Gamma \vdash \overline{a_i k} \mid i \in I, k \in K_i\}}{\Gamma \vdash (A \setminus \{a_i \mid i \in I\}) \cup \bigcup_{i \in I} B_i} \tag{1.28}$$

(Rneg) is creating NAND-clauses from NAND-clauses using OR as a side-condition. (Rpos) is creating OR-clauses from OR-clauses using NAND as a side-condition. In (Rneg), $\overline{a_i A_i}$ denotes the NAND $\overline{\{a_i\} \cup A_i}$ with a potentially empty A_i.

The premise of the (Rneg) rule is a set of I NAND-clauses together with one OR-clause with I elements such that each atom a_i in the OR-clause is contained within a NAND-clause, and such that each NAND-clause contains an atom from the OR-clause. The correspondence between the NAND-clauses and the elements of the OR-clause should in other words be bijective. The conclusion is the union of all the NAND-clauses without their corresponding atom from the OR-clause.

Here are some examples of incorrect applications of the (Rneg)-rules, followed by some correct applications:

$$\text{(1)} \ \frac{\overline{ax} \quad \overline{by} \quad \overline{cz}}{\overline{xyz}} abx \qquad \text{(3)} \ \frac{\overline{ax} \quad \overline{by}}{\overline{xy}} abx \qquad \text{(2)} \ \frac{\overline{ax} \quad \overline{by} \quad \overline{bz}}{\overline{xyz}} abx \tag{1.29}$$

(1) is incorrect because the NAND $\overline{cz}$ contains no atoms from the OR abx. (2) is incorrect because the number of NAND-clauses does not match the length of the OR-clause. (3) is incorrect because there exist no bijective correspondence of the type described above.

$$\text{(4)} \ \frac{\overline{ax} \quad \overline{by} \quad \overline{cz}}{\overline{xyz}} abc \qquad \text{(5)} \ \frac{\overline{ax} \quad \overline{b}}{\overline{x}} ab \qquad \text{(6)} \ \frac{\overline{ax} \quad \overline{by} \quad \overline{xyz}}{\overline{xyz}} abx \tag{1.30}$$

The three above applications are all correct, since all the atoms in each OR-clause get matched to exactly one NAND-clause in such a way that no NAND-clause stays unmatched.

We set no restrictions on the number and cardinality of our clauses, meaning that there might be an infinite number of clauses, and both the OR-clausess and the NAND-clauses might be either finite or infinite in size. Note that an infinite graph gives infinitely many NAND-clauses, while an infinitary graph also gives us infinitely long OR-clauses.

We study the refutation system that arises from the Axioms and the (Rneg)-rule, calling it Neg. It is shown in the submitted paper [**michal-completeness**] that Neg is sound

and refutationally complete for theories with only a countable number of OR-clauses. Soundness gives us that proving $\overline{C}$ for any $C \subseteq G$ implies that the vertices in C cannot all be assigned 1 in the graph model. Refutational completeness gives us that whenever a graph/theory is inconsistent, we are able to prove $\varnothing$ in Neg.

1.7.2 Inconsistency of the Yablo-graph

The inconsistency of the Yablo-graph is easily proven using Neg only. Since every vertex x_i (using the notation from Figure 1.1) has an edge to each vertex x_j where $j > i$, we get that every pair of distinct vertices is connected by an edge. This means that our set of axioms from the Yablo-graph looks like this:

$$\text{NAND} = \{\overline{x_i x_j} \mid i < j\} \qquad \text{OR} = \{x_i x_{i+1} x_{i+2} \dots \mid i \in \mathbb{N}\} \tag{1.31}$$

For any vertex x_i from the Yablo-graph, we are now able to prove $\overline{x_i}$ in the following way:

$$\dfrac{\overline{x_i x_{i+1}} \quad \overline{x_i x_{i+2}} \quad \overline{x_i x_{i+3}} \quad \dots \qquad x_i x_{i+1} x_{i+2} \dots}{\overline{x_i}}$$

Proving $\varnothing$ is now simple:

$$\dfrac{\dfrac{\dots}{\overline{x_1}} \quad \dfrac{\dots}{\overline{x_2}} \quad \dfrac{\dots}{\overline{x_3}} \quad \dots \qquad x_1 x_2 x_3 \dots}{\varnothing}$$

A less trivial inconsistency proof is the one on the *Stretched Yablo-graph*. This proof can be found in the appendix together with the definition of Stretched Yablo.

It is worth mentioning that even though our focus has been – and will be – on theories originating from graphs, the results on soundness and completeness holds for any theory consisting of a set of NANDs and a set of ORs.

An example of this is the pigeonhole problem which easily can be represented as a set of NANDs and ORs, but does not directly correspond to a graph (it can of course be translated to a graph theory, like any other propositional theory). Proofs in Neg of pigeonhole problems of different size are to be found in the appendix[5].

1.8 This thesis

1.8.1 Motivation

Having that (Neg) is both sound and refutationally complete, together with results linking inconcistency proofs in (Neg) to structures in graphs is of great interest to us,

[5] TODO: some reference system needed

since it can potentially further weaken the current conditions we have for kernels in graphs. This thesis will thus be concerned mainly with two kinds of questions;

1. Are there any characterizing qualities of the proof system (Neg), potentially under certain restrictions?
2. If the axioms come from a graph, are there any provable results in (Neg) that correspond to certain structures in the graph?

The first question covers qualities like proof length (complexity), size and number of clauses used throughout a proof, the existence of certain clauses in certain proofs, or maybe even the existence of an entire proof normal form. For instance, an answer to the following question could be of great interest to us: "Are there certain kinds of clauses that needs to be present in a proof in order to prove $\varnothing$?".

The second question covers questions like "What does it mean for the graph that its axioms can prove $\varnothing$?". Of course, we know the answer to this question, it means that the graph has no kernel (by soundness of (Neg)). But what about the provability of $\overline{x}$ for a node x in the graph? What does that mean for the graph and the node x?

Finding a type of clause neccesarily present in an inconsistency proof, together with results on the graph-structural counterpart of that type of clause, gives us in combination a graph-structure neccesarily present in a kernel-free graph (by soundness of (Neg)).

1.8.2 Thesis Overview

THIS SECTION IS NOT UP TO DATE WITH THE FOLLOWING CHAPTERS Section 2 will take a look at the potential to control the size of NAND-clauses in an inconsistency proof without any restrictions on the axioms. Section 3 will study the same potential as above, but with graph-based axioms. We are in both cases above particularly concerned with whether we are able to restrict our proofs to only use binary NAND-clauses. Section 4 will explore the graph structure that corresponds to unary and binary NAND-clauses, and explain their potential importance.

Chapter 2

NAND-clauses in graphs

In this chapter we will motivate the search for graph structural equivalents of unary and binary NAND-clauses proven in Neg. An actual graph structure corresponding to the NAND-clauses will not be presented, but some necessary conditions will be shown for both directions. In other words, we will search for graph structures implying certain provable NAND-clauses, as well as what certain provable NAND-clauses imply about the corresponding graph.

2.1 Motivation

Since our proof system, Neg, has only one rule, the last step of any inconsistency proof will always look the same:

$$\frac{\overline{x_1} \quad \overline{x_2} \quad \overline{x_3} \quad \dots}{\varnothing}\ x_1x_2x_3\dots$$

The premise will always consist of a collection of NAND-clauses, each of length 1, together with an OR-clause equal to the union of all the NAND-clauses. It is easy to see that none of the NAND-clauses can be larger than unary, since that would result in a non-empty NAND-clause in the conclusion. The OR-clause has to equal the union of the NAND-clauses simply by definition of the (RNeg)-rule.

This fact was also observed and formalized by Walicki[**michal-completeness**]:

$$\Gamma \vdash_{Neg} \{\} \Leftrightarrow \exists K \in \mathrm{OR} : (\forall k \in K : \Gamma \vdash_{Neg} \overline{k}) \tag{2.1}$$

We know from the definition that any OR-clause used in the proof system corresponds to a single vertex with its successors in the graph. We do however not know what the NAND-clauses of length 1 might correspond to. Knowing this would, by soundness and

completeness of Neg, give us the graph structure needed for a kernel not to exist. [1]

The only thing we *do* know about unary NAND-clauses is that they correspond to vertices that are assigned 0 in all models of the graph. We get this from soundness of Neg. This is however not the graph structural property we are ultimately looking for, but we can at least say that we have reduced the question "What does an inconsistent graph look like?" to the question "What does a provably false vertex look like?".

So what does a unary NAND-clause proven in Neg correspond to in the graph? Similarly to a proof of $\varnothing$, there is really just one way to prove a unary NAND-clause:

$$\frac{\overline{xy_1} \quad \overline{xy_2} \quad \dots \quad \overline{y_n} \quad \overline{y_{n+1}} \quad \dots}{\overline{x}} \; y_1 y_2 \dots y_n y_{n+1} \dots$$

Any derivation of a unary NAND-clause $\overline{x}$ must end with a rule application using $K \in \mathrm{OR}$ where for each $k \in K$, there is a NAND-clause in the premise that is either unary, $\overline{k}$ or binary, $\overline{xk}$. In addition, we require the premise to contain at least one binary NAND-clause, in order to actually be able to conclude with $\overline{x}$ and not $\varnothing$.

We formalize this observation in the following way:

$$\Gamma \vdash_{Neg} \overline{x} \Leftrightarrow \exists K \in \mathrm{OR} : (\forall k \in K : \Gamma \vdash_{Neg} \overline{kx} \vee \Gamma \vdash_{Neg} \overline{k}) \wedge (\exists k \in K : \Gamma \vdash_{Neg} \overline{kx}) \quad (2.2)$$

Just as we reduced the problem of inconsistency to the problem of unary NAND-clauses, we are able to reduce the problem further to binary NAND-clauses. We could even continue the reduction further to ternary clauses, quaternary clauses and so on, but without a change of strategy at some point, this seems pointless.

Our current reduction lets us ask how two vertices x, y are *connected* in the graph when their binary NAND $\overline{xy}$ is proven in Neg. Observe that we have parts of this correspondence down already, with our axioms being all binary. Vertices directly connected by an edge must therefore be a part of the corresponding graph structure.

Finding some graph strucural relation that corresponds to binary NAND-clauses might also give us some knowledge about a potential standard form of binary-NAND-proofs.[2]

The next sections will present various graph structures, each more general than the previous ones, that imply the provability of certain binary NAND-clauses in Neg. Ultimately, we are aiming to find a structure such that the opposite implication holds, i.e. such that the absense of the structure in the graph implies the inability to prove a certain binary NAND-clause.

For each structure presented, the first implication will be shown, followed by an example disproving the second, opposite implication.

[1] TODO: Currently brushing over the restrictions set to the proofs of completeness and soundness. Will fix this later.

[2] TODO: does this make sense?

These counterexamples will be graphs with the presented structure absent, but with certain NAND-clauses still provable.

2.2 Odd paths

Given two vertices x_1 and x_2 in a graph G, if we in Neg are able to prove $\overline{x_1 x_2}$ from the axioms we get from G, we say that x_1 and x_2 are *vel-connected*. This section will try to answer the question "what kind of graph structures imply that certain vertices are vel-connected?".

Because of soundness of Neg, if two vertices are vel-connected in a graph, for any kernel K in that graph, at least one of the two vertices are outside K. The counterpositive of this observation being that for a graph G, if there exists a kernel $K \subseteq G$ such that two vertices, x_1 and x_2 are both in K, then x_1 and x_2 cannot be vel-connected in G. This fact will be used when arguing why some graph structures are *not* giving us vel-connected vertices.

We already know that whenever two vertices are connected by an edge, their binary NAND is trivially provable in Neg, since it is a part of the axioms.

We illustrate this case with the figure below, where dashed lines represent possible out-edges to irrelevant parts of the graph. If a vertex has no dashed edges, it means that we disallow any additional edges out from this vertex.

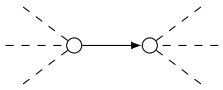


Figure 2.1

The basic example above tells us that two vertices are vel-connected when one is the successor of the other. It is however easy to find a case examplifying how the implication does not hold in the opposite direction.

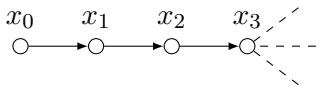


Figure 2.2

The above graph provides us the following axioms: $NAND = \{\overline{x_0 x_1}, \overline{x_1 x_2}, \overline{x_2 x_3}\}$, $OR = \{x_0 x_1, x_1 x_2, x_2 x_3\}$. From these axioms, we can now, despite the fact that the vertices x_0 and x_3 are not connected by an edge, easily prove the NAND-clause $\overline{x_0 x_3}$:

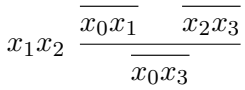


Figure 2.3

Intuitively, one can imagine that the proof above is connecting two NAND-clauses using an OR-clause, resulting in a new binary NAND-clause containing vertices that are weaklier connected than the ones we started with. This can be done repeatedly, resulting in the ability to prove binary NAND-clauses from vertices that are connected by arbitrarily long paths of the kind above.

Consider the following graph:

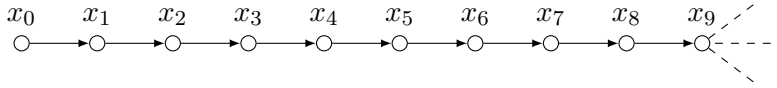


Figure 2.4

With axioms from the above graph, $\overline{x_0x_9}$ can be proven in Neg in the following way:

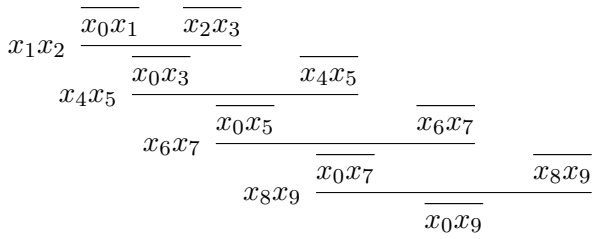


Figure 2.5

Observe that the above proof also proves $\overline{x_0x_3}$, $\overline{x_0x_5}$ and $\overline{x_0x_7}$ along the way, all of which contain vertices connected by paths of *odd* length. This is an important point. NAND-clauses containing vertices connected only by paths of *even* length cannot be proven in the same manner as above.

In many cases, such NAND-clauses cannot be proven at all. This is examplified by the below graph, with a kernel containing the vertices a and b, showing that the NAND $\overline{ab}$ is unprovable in Neg. The kernel in the graph below is represented by the black vertices.

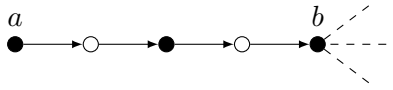


Figure 2.6

Restricting our paths to be of odd length lets us avoid cases like the one above, but there are still some cases left to avoid, in order for our implication to hold.

2.3 Trimming

Given a path and two consecutive vertices x and y from that path, we will say that x is *trimmed*, with respect to that path, if $N(x) = \{y\}$. In order words, given any path, if one of its vertices is pointing to its successor in the path only, that vertex is trimmed in that path. If all the vertices of a path, except the terminal vertex, is trimmed, we will call the path a *fully trimmed path*. All the paths so far in this chapter have been fully trimmed.

If two vertices a and b are connected by an odd path that is *not* fully trimmed, $\overline{ab}$ is not neseccarily provable in Neg. The below graph examplifies this with a kernel containing both a and b.

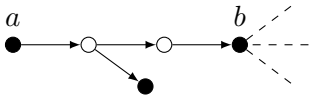


Figure 2.7

Generalizing this, we get that whenever two vertices, x_0 and x_k, are connected by a fully trimmed path of odd length, they are vel-connected. Their corresponding NAND-clause $\overline{x_0 x_k}$ can be proven in Neg in the following way:

$$
\dfrac{
x_{k-2}x_{k-1}\quad \dfrac{
\dfrac{
x_3x_4\quad \dfrac{
x_1x_2\quad \overline{x_0x_1}\quad \overline{x_2x_3}
}{\overline{x_0x_3}}
\quad \overline{x_4x_5}
}{\overline{x_0x_5}}
\quad \vdots \quad \overline{x_0x_{k-2}}
}{\overline{x_0x_k}}
\quad \overline{x_{k-1}x_k}
}{}
$$

Figure 2.8

All the OR-clauses used are indeed binary, letting us prove our NAND-clause without introducing any other clauses than the ones we get from the path itself. However, notice that only half of the OR-clauses is actually in use in such a proof. Thus, we need only to restrict half of the vertices in the path to not branch.

With the above proof in mind, consider the following graph:

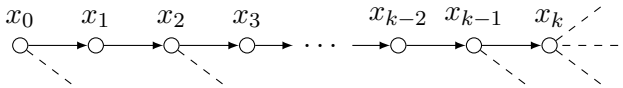


Figure 2.9: An oddly trimmed path of odd length

In the above path between x_1 and x_k, only every other vertex is trimmed. We will call this path variant an *oddly trimmed path*, and define it formally as follows: A path $\langle x_0, x_k \rangle$[3] is oddly trimmed if for each odd $i < k$, x_i is trimmed with respect to that path. One can immediately note that any fully trimmed path is also oddly trimmed.

The axioms we get from the oddly trimmed path above does not differ significantly from the fully trimmed variant. Since the vertices $x_0, x_2, x_4, \ldots, x_{k_1}$ no longer have single successors, their corresponding OR-clauses will no longer be binary. However, since none of these OR-clauses are used in the above proof, the proof will remain valid also for the oddly trimmed path.

We can thus generalize further and say that any two vertices connected by an oddly trimmed path of odd length is vel-connected.

2.4 Odd vels

We observed that the addition of successors to some vertices in a fully trimmed path made no difference to the proof of the corresponding NAND-clause. From that observation, we were able to generalize the concept of fully trimmed paths to the concept of oddly trimmed paths, while still having our original implication hold. In a similar manner, the following observation will further generalize our structure.

The direction of edges in a graph can often be changed individually while still keeping many axiomatic clauses unchanged, including all the NAND-clauses.

Consider the following graph:

[3] TODO: This notation is not explained

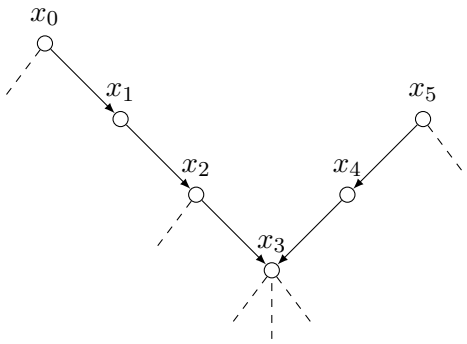


Figure 2.10

From the above graph, $\overline{x_0x_5}$ can be proven in the same way as in the proof of $\overline{x_0x_9}$ from earlier. Again, the only thing we are changing in terms of the axioms are the OR-clauses that are not used in the proof.

We will call this kind of graph structure an *odd vel* and define it formally in the following way: Two vertices a and b have an odd vel between them if there exists a vertex c such that there are oddly trimmed paths from a to c and from b to c, one of even length (possibly 0) and one of odd length. Notice that an oddly trimmed path of odd length is just an instance of an odd vel where the even path is of length 0.

The formal proof why all NAND-clauses from vertices connected by vels are provable in Neg can be found in Section 4.3.

There are obviously other ways of altering directions of individual edges in a path. The next example will however show that most of them will alter the OR-clauses in such a way that our implication no longer hold.

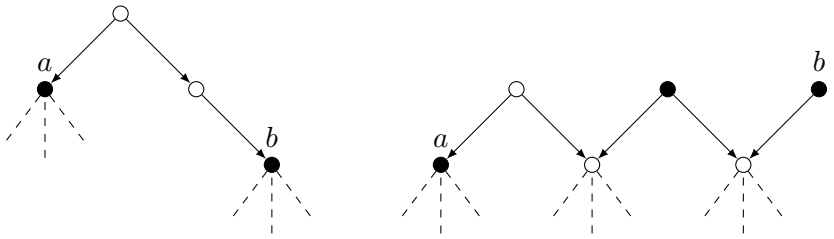


Figure 2.11

The figure above shows two odd paths that have had some of their edges flipped. In both examples, the shown kernels contain both a and b, showing that these constructions are not providing a provable NAND-clause $\overline{ab}$ in Neg.

2.5 Generalizing trimming

In this section, we will take a closer look at the current concept of trimming, and through some examples introduce a more general definition. So far, trimming has meant forcing certain vertices to point only at its successor in a path.

In Section 2.2 we motivated the concept of trimmed paths by presenting the Figure 2.7. The figure shows how a path that normally provides a provable NAND-clause may be "ruined" by adding out-edges to certain vertices in the path, making certain vital OR-clauses non-binary and thus making the NAND-clause unprovable.

Notice however that adding a loop on the vertex to which the path branches, leaves us with a completely different situation.

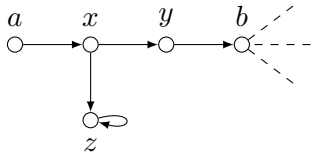


Figure 2.12

The addition of the loop adds $\overline{z}$ to the axioms. The NAND-clause $\overline{ab}$ can now easily be proven in the following way:

$$xyz\ \frac{\overline{ax} \quad \overline{yb} \quad \overline{z}}{\overline{ab}}$$

Figure 2.13

This shows the fact that vertices at odd positions in a path can indeed branch off to other vertices than their path successor and still contribute to a provable NAND-clause, just as long as the vertices they branch off to is provably false in Neg. If the vertex is provably false, we can use that unary NAND-clause, like we did in the above proof, to handle the non-binary OR-clauses. This observation lets us generalize our notion of trimmed vertices.

The big problem with this generalization is that it makes our vel-relation no longer a purely graph-structural one. With such a definition, whenever we want to check whether two vertices have a vel between them, we may have to work out the actual provability of certain unary NAND-clauses. This is exactly what we try to avoid with our vel-relation.

To make matters worse, consider the following two graphs:

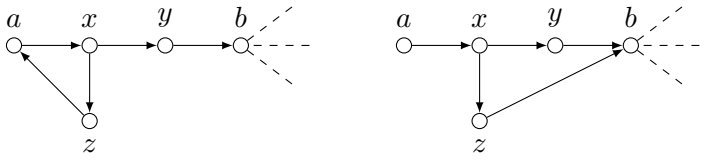


Figure 2.14

Like with the graph from Figure 2.12, the above graphs are both based on Figure 2.7, each with only one edge added. The left variant has $\overline{az}$ in its axioms, while the right variant has $\overline{bz}$, again making their respective proofs of $\overline{ab}$ almost trivial.

$$xyz\ \frac{\overline{ax} \quad \overline{yb} \quad \overline{za}}{\overline{ab}} \qquad xyz\ \frac{\overline{ax} \quad \overline{yb} \quad \overline{zb}}{\overline{ab}}$$

Figure 2.15

Not only can odd vertices on a path branch off to provably false vertices and still contribute to vel-connected vertices, the can also branch off to vertices that

Chapter 3

Proof power of binary NAND-clauses

In this chapter we will investigate the statement that whenever a theory is inconsistent, it can be proven in Neg using unary and binary NAND-clauses only. Such a result would be significant both because it would be a strong property for a proof system to have in general, but also because it could potentially help us to further characterize a kernel-free graph.

Having that any inconsistency can be proven in Neg using unary and binary NAND-clauses only, might imply a similar property in kernel-free graphs, namely that inconsistencies can be described as a collection of pairwise structural relations between vertices.

We will check the statement first for non-graph-based theories and then for theories based on graphs with different properties.

3.1 Proving inconsistency in unrestricted theories

This section will show that there exist inconsistencies in non-graph-based theories that are not possible to prove in Neg using only unary and binary NAND-clauses. We will prove the pigeonhole principle to examplify this.

The pigeonhole principle states that whenever you have n pigeons and m holes such that $n > m$, then at least one hole must contain more than one pigeon. We can prove this principle in Neg by proving an inconsistency of the theory that (1) all n pigeons are contained in a hole and (2) each hole contains only one pigeon.

Letting the atom x_i denote the pigeon i occupying the hole x, the above theory, with n

pigeons and m holes, can be formalized in the following way:

$$(1): \quad \{\ 1_i 2_i 3_i \ldots m_i \mid i \leq n\ \} \tag{3.1}$$

$$(2): \quad \{\ \overline{x_i x_j} \mid i < j \leq n, x \leq m\ \} \tag{3.2}$$

3.1.1 Proof for 4 pigeons and 3 holes

The pigeonhole principle for 4 pigeons and 3 holes will have the following axioms:

$$\text{OR} = \{\ 1_1 2_1 3_1,\ 1_2 2_2 3_2,\ 1_3 2_3 3_3,\ 1_4 2_4 3_4\ \} \tag{3.3}$$

$$\text{NAND} = \left\{ \begin{array}{l} \overline{1_1 1_2},\ \overline{1_1 1_3},\ \overline{1_1 1_4},\ \overline{1_2 1_3},\ \overline{1_2 1_4},\ \overline{1_3 1_4}, \\ \overline{2_1 2_2},\ \overline{2_1 2_3},\ \overline{2_1 2_4},\ \overline{2_2 3_3},\ \overline{2_2 2_4},\ \overline{2_3 2_4}, \\ \overline{3_1 2_2},\ \overline{3_1 3_3},\ \overline{3_1 3_4},\ \overline{3_2 3_3},\ \overline{3_2 3_4},\ \overline{3_3 3_4} \end{array} \right\} \tag{3.4}$$

From the axioms above we can prove inconsistency in the following way:

Proving $\overline{1_4}$:

$$\dfrac{\overline{1_3 1_4} \quad \dfrac{\overline{1_2 1_4} \quad \overline{2_2 2_3} \qquad \dfrac{\overline{1_1 1_4} \quad \overline{2_1 2_3} \quad \overline{3_1 3_2}}{\overline{3_2 3_1 1_4}}\ 1_1 2_1 3_1}{\overline{2_3 1_4}}\ 1_2 2_2 3_2 \qquad \dfrac{\overline{1_2 1_4} \qquad \dfrac{\overline{1_1 1_4} \quad \overline{2_1 2_2} \quad \overline{3_1 3_3}}{\overline{2_2 3_3 1_4}}\ 1_1 2_1 3_1 \qquad \overline{3_2 3_3}}{\overline{3_3 1_4}}\ 1_2 2_2 3_2}{\overline{1_4}}\ 1_3 2_3 3_3$$

Proving $\overline{2_4}$:

$$\dfrac{\overline{1_2 1_3} \quad \overline{2_2 2_4} \qquad \dfrac{\overline{1_1 1_3} \quad \overline{2_1 2_4} \quad \overline{3_1 3_2}}{\overline{3_2 1_3 2_4}}\ 1_1 2_1 3_1}{\overline{1_3 2_4}}\ 1_2 2_2 3_2 \qquad 1_1 2_1 3_1\ \dfrac{\overline{1_1 1_2} \quad \overline{2_1 2_4} \quad \overline{3_1 3_3}}{\overline{1_2 3_3 2_4}}\ 1_2 2_2 3_2 \qquad \dfrac{\overline{2_2 2_4} \quad \overline{3_2 3_3}}{\overline{3_3 2_4}}}{\overline{2_4}}\ 1_3 2_3 3_3$$

Proving $\overline{3_4}$:

$$\dfrac{1_2 2_2 3_2\ \dfrac{\overline{1_2 1_3}}{\overline{1_3 3_4}}\ 1_1 2_1 3_1\ \dfrac{\overline{1_1 1_3} \quad \overline{2_1 2_2} \quad \overline{3_1 3_4}}{\overline{2_1 3_3 3_4}} \qquad \overline{3_2 3_4}}{1_3 2_3 3_3\ \dfrac{\overline{1_3 3_4}}{\overline{3_4}}}\ 1_1 2_1 3_1\ \dfrac{\overline{1_1 1_2} \quad \overline{2_1 2_3} \quad \overline{3_1 3_4}}{\overline{1_2 2_3 3_4}}\ 1_2 2_2 3_2\ \dfrac{\overline{2_2 2_3} \quad \overline{3_2 3_4}}{\overline{2_3 3_4}} \qquad \overline{3_3 3_4}$$

With these three unary clauses proven, we can now prove $\varnothing$ in one step:

$$1_4 2_4 3_4\ \dfrac{\overline{1_4} \quad \overline{2_4} \quad \overline{3_4}}{\varnothing}$$

Observe that NAND-clauses of length 3 appear several times in this proof. We will show that this is a necessity with the axiom set described above.

3.1.2 Long NAND-clauses in pigeonhole proofs

As mentioned earlier, the only strategy in proving inconsistencies in Neg is to create new NAND-clauses until you have a set of unary NAND-clauses such that their union matches an OR-clause.

Now, what possible ways are there to create new NAND-clauses from our given pigeonhole axioms? The OR-clauses are what dictates the ways new NAND-clauses can, be created, and since all OR-clauses are of length 3, we get that any premise must consist of exactly 3 NAND-clauses. In addition, since all OR-clauses contain exactly one atom from each hole, each of the three NAND-clauses must contain atoms from different holes.

Looking at the NAND-clauses in the axiom set, we see that none of them contain atoms from two different holes, so the three axiomatic NAND-clauses eligible in a premise are mutually disjoint. Since all three are binary, we have a total of 6 different axioms in the premise, and with the OR-clause shaving of 3 of these, our conclusion must contain 3 different atoms. Any NAND-clause derived directly from axioms must therefore be of length 3.

$$1_i 2_i 3_i \ \frac{\overline{1_i 1_j} \quad \overline{2_i 2_k} \quad \overline{3_i 3_l}}{\overline{1_j 2_k 3_l}}$$

It is easy to see that this generalizes to any version of the pigeonhole principle. When you have n holes and $> n$ pigeons, the OR-clauses will be of length n, requiring n disjoint NAND-clauses in the premise, thus resulting in a NAND-clause of length n.

This means not only that we are unable to keep the clause length at 2, but also that the size of the clauses increase with the number of holes.

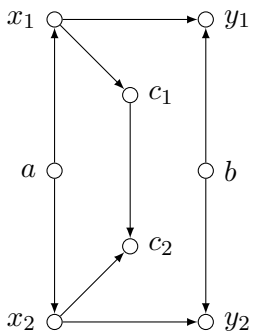
3.2 Proving inconsistency in graph theories

We have just shown that in the case of unrestricted theories, we have no guarantee that an inconsistency proof can be constructed using unary and binary NAND-clauses only, so what about graph theories? After all, it is these theories that motivated the search for this property in the first place. Unfortunately, it turns out that even for these kinds of theories, there are inconsistency proofs that require some longer NAND-clauses.

We will show this fact by presenting a graph containing a provable binary NAND-clause, and show that the only way to prove it is through using non-binary NAND-clauses. We will later extend the graph so that it becomes inconsistent, and show that the only way

to show this inconsistency is by using the mentioned binary NAND, thus showing that the inconsistency proof must contain a non-binary NAND-clause.

The graph in question is the following:



We will be treating the vertices y_1, y_2 and c_2 in the above graph as if each of them are initial vertices of (non-depicted) disjoint rays, and not sinks, as the figure suggests. This gives the following axiom set:

$$\text{OR} = \{ax_1x_2,\ x_1y_1c_1,\ x_2y_2c_2,\ by_1y_2,\ c_1c_2\} \tag{3.5}$$

$$\text{NAND} = \{\overline{ax_1},\ \overline{ax_2},\ \overline{x_1y_1},\ \overline{x_1c_1},\ \overline{x_2y_2},\ \overline{x_2c_2},\ \overline{by_1},\ \overline{by_2},\ \overline{c_1c_2}\} \tag{3.6}$$

Note that we do not include the clauses corresponding to the elements of the above-mentioned rays, since these will not contribute to the argument[1].

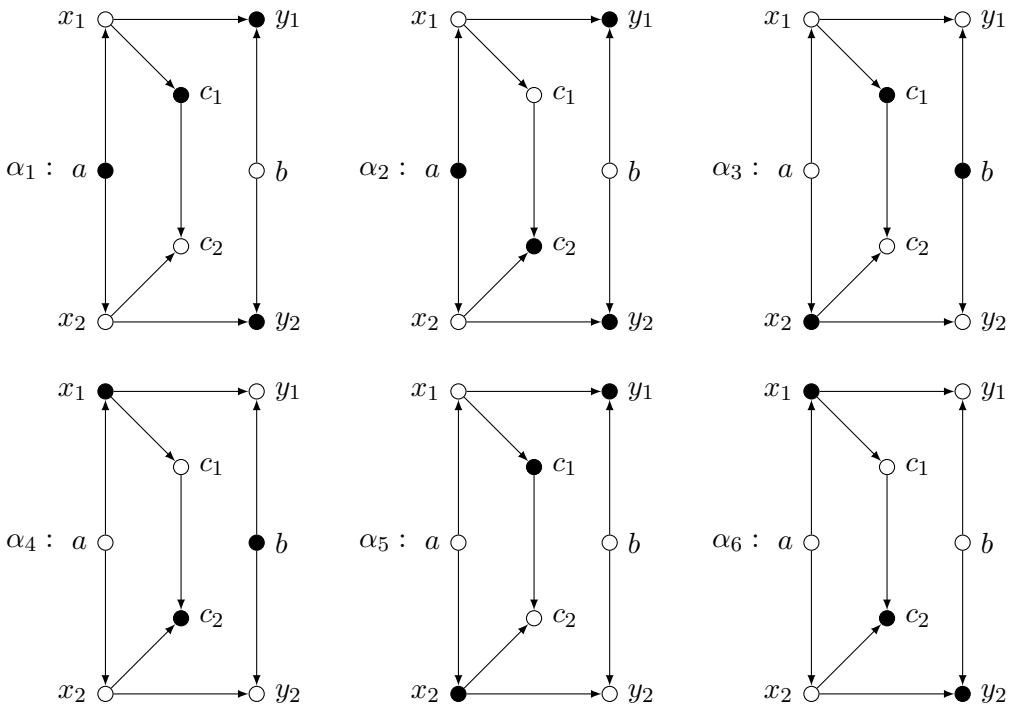
The NAND-clause that we want to prove is $\overline{ab}$, and this is one way to prove it in Neg:

$$\dfrac{x_2y_2c_2 \quad \dfrac{\overline{ax_2} \quad \overline{by_2} \quad \overline{c_1c_2}}{c_1c_2 \quad \overline{abc_1}} \qquad \dfrac{\overline{ax_1} \quad \overline{by_1} \quad \overline{c_1c_2}}{\overline{abc_2}} \quad x_1y_1c_1}{\overline{ab}}$$

In the premise of the last step, we have two non-binary NAND-clauses. How do we know that this cannot be avoided?

By taking a closer look at the graph, we can work out these 6 solutions, among others (black vertices being assigned 1, while white vertices being assigned 0):

[1] TODO: This needs an explanation



Based on these solutions, we can say a lot about the provability of various binary NAND-clauses. Any pair of vertices from the graph that can be 1 under the same assignment will not be a Neg-provable NAND. We get this from soundness of Neg, since when two vertices v and u are both 1 under some assignment, we get that $\nvDash \overline{uv}$ and by soundness of Neg that $\nvdash_{Neg} \overline{uv}$.

It is also worth noting that since all the vertices can be assigned 1 under at least one assignment, no unary NAND can be proven in Neg, by the same argument as above.

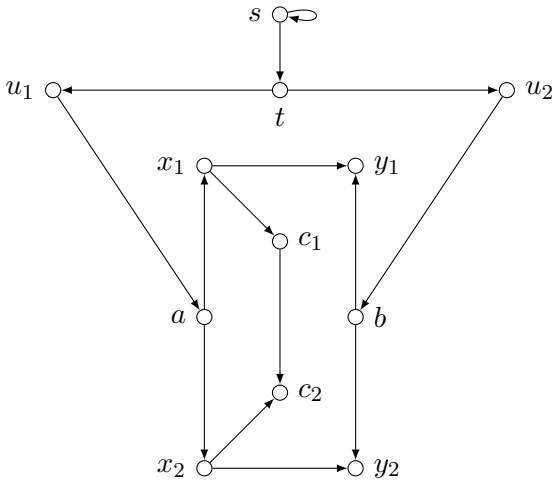
We can make the following table showing what pairs of vertices can be 1 simultaneously and thus consititute a binary NAND unprovable in Neg. A cell is filled with a reference to the assignment, if any, examplifying the case.

	a	b	x_1	x_2	y_1	y_2	c_1	c_2
a	$-$				α_1	α_1	α_1	α_2
b		$-$	α_4	α_3			α_3	α_4
x_1			$-$			α_6		α_6
x_2				$-$	α_5		α_5	
y_1					$-$	α_1	α_1	α_2
y_2						$-$	α_1	α_2
c_1							$-$	
c_2								$-$

Among the pairs above that are not shown to be unprovable, only two of them are not in the axiom set; $\overline{ab}$ and $\overline{x_1x_2}$. We will use this observation to show that it is impossible to prove $\overline{ab}$ with binary NAND-clauses only.

Let us assume that we have a rule application with all unary or binary NAND-clauses in the premise and with $\overline{ab}$ in the conclusion. Based on the (Rneg)-rule, we know that the premise must contain at least one binary NAND-clause containg a and at least one containing b. The table above tells us that the only provable binary NAND-clauses that contain a or b are the ones in the axiom set: $\overline{ax_1}$, $\overline{ax_2}$, $\overline{by_1}$ and $\overline{by_2}$. Since we don't want any x_i or y_i in the conclusion, these variables have to be removed by the OR-clause. The OR-clauses $x_1y_1c_1$ and $x_2y_2c_2$ are the only ones that contain both an x and a y, making them the only OR-clauses that can potentially conclude with $\overline{ab}$. This gives us the following information: since both the possible OR-clauses are of length 3, the rule application concluding with $\overline{ab}$ has 3 NAND-clauses in its premise; one containing an x, one containing a y and one containing a c. Looking at our table again, we see that the potentially provable NAND-clauses containing a c are again the axioms only. Since there are no provable NAND-clauses on the form ac_i or bc_i, we get that the conclusion of our rule cannot possibly be of length 2, thus contradicting our assumption. We can therefore conclude that $\overline{ab}$ cannot be proven in Neg using unary and binary NAND-clauses only.

Consider now the following extension of our original graph:



The extension provides us with 6 new NAND-clauses and 4 new OR-clauses in the axioms set:

$$\text{OR'} = \text{OR} \cup \{st,\ tu_1u_2,\ u_1a,\ u_2b\} \tag{3.7}$$

$$\text{NAND'} = \text{NAND} \cup \{\overline{s},\ \overline{st},\ \overline{tu_1},\ \overline{tu_2},\ \overline{u_1a},\ \overline{u_2b}\} \tag{3.8}$$

This extended graph can now be proven inconsistent using the newly provided clauses,

as shown in the following Neg proof:

$$\dfrac{\dfrac{\overline{s} \qquad \dfrac{\overline{tu_2} \qquad \dfrac{\overline{tu_1} \qquad \dfrac{\cdots}{\overline{\overline{ab}}}\, u_1 a}{\overline{tb}}\, u_2 b}{\overline{t}}\, st}{\varnothing}}{}$$

We will now show that this inconsistency proof has to use the $\overline{ab}$ clause.

First of all, since both the vertices s and t are provably false, as the proof above shows, they can by soundness of Neg not be assigned 1 under any assignment. This means that none of the binary NAND-clauses that contains either s or t can be ruled out as unprovable like we did above.

Chapter 4

Proofs

4.1 Translating CNF to GNF

Since any PL theory can be expressed in CNF, showing that any theory P in CNF can be translated to a theory R in GNF such that P and R are equisatisfiable gives us that any PL theory has an equisatisfiable GNF.

Given any CNF theory P, for each formula in it, follow the steps below to acquire its corresponding GNF theory.

Step 1: For each atom x_i in the formula, introduce a fresh variable x_i' and add the following two GNF formulae to P: $x_i' \leftrightarrow \neg x_i, x_i \leftrightarrow \neg x_i'$, (unless this has already been done while translating an earlier formula in the theory).

Step 2: In each clause of the formula, replace every negative literal $\neg x_i$ with its corresponding x_i' fom step 1. Every clause in the formula does now contain positive literals only.

Step 3: For every clause $(x_1 \vee x_2 \vee \cdots \vee x_n)$, replace it with the following formula, where y is fresh:

$$y \leftrightarrow (\neg x_1 \wedge \neg x_2 \wedge \cdots \wedge \neg x_n \wedge \neg y)$$

This formula is equisatisfiable with our original clause.

The combined set of all these acquired formulae will make up our GNF-theory. We have only added proper GNF formulae and all variables appear to the left in exactly one clause, so R will indeed be a GNF theory.

Step 1 is adding a bi-implication formula that is always satisfiable, since one of two variables in it is fresh. One can think of it as a labelling of an already existing variable. Thus, adding these formulae to a theory does not change its satisfiability/non-satisfiability.

Step 2 is replacing each negative literal with its label, also not changing the satisfiability.

Step 3 is replacing one formula with an equisatisfiable formula, thus not changing the overall satisfiability.

Since none of the steps above change the satisfiability of the theory, we can conclude that our acquired theory is equisatisfiable with to original CNF theory.

Below are some examples of CNF-theories with their corresponding GNF-theories.

Example 4.

$$\text{CNF:} \quad (a \vee b) \tag{4.1}$$
$$\text{GNF:} \quad (a \leftrightarrow \neg a'), (a' \leftrightarrow \neg a), (b \leftrightarrow \neg b'), (b' \leftrightarrow \neg b), (y_1 \leftrightarrow (\neg a \wedge \neg b \wedge \neg y_1)) \tag{4.2}$$
$$\tag{4.3}$$
$$\text{CNF:} \quad (\neg a) \tag{4.4}$$
$$\text{GNF:} \quad (a \leftrightarrow \neg a'), (a' \leftrightarrow \neg a), (y_1 \leftrightarrow (\neg a' \wedge \neg y_1)) \tag{4.5}$$
$$\tag{4.6}$$

4.2 Inconsistency of Stretched Yablo

One variation of the Yablo-graph is the Stretched Yablo-graph, presented by Walicki in [**michal-completeness**]. While the Yablo-graph is a ray where each vertex on it has direct edges to all the vertices after it, the Stretched Yablo-graph is a ray where each vertex has *disjoint paths* (with certain properties depending on the variation of Stretched Yablo) to all vertices in after it.

The variation on which we will be proving an inconsistency is one with a set of core vertices $\mathbb{N}$ where each vertex x has a disjoint path to every vertex y after it, such that the length of the path is $(2 \times (y - x) - 1)$.

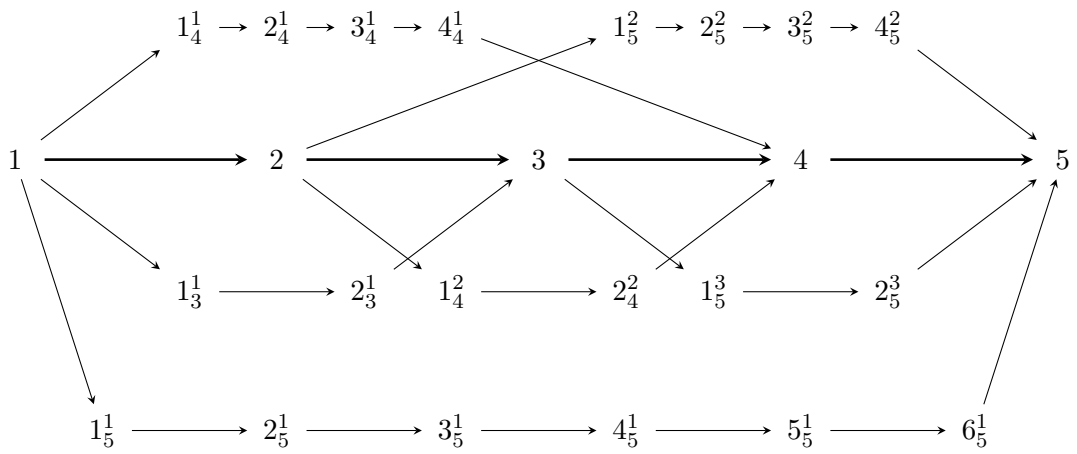


Figure 4.1: Stretched Yablo

Shown above are parts of the Stretched Yablo-graph described. We denote the core vertices by natural numbers. The peripheral vertices contained in each path is denoted by n_y^x where x and y is the source and target, respectively, of the path in which the vertex is contained. n denoted the relative position on the path.

Note that since two core vertices x and y have a peripheral path between them of length $L_y^x = 2(y-x)-1$ (from the definition above), each path consists of $L_y^x - 1 = 2(y-x)-2$ peripheral vertices. The vertex pointing back onto the core ray will thus alway be the vertex $(2(y-x)-2)_y^x$.

Based on the given definition of Stretched Yablo and the notation from above, we get

the following sets of axioms:

$$N1a = \{\ \overline{x(x+1)} \mid x \in \mathbb{N}\ \} \tag{4.7}$$
$$N1b = \{\ \overline{x1^x_y} \mid x, y \in \mathbb{N}, x+1 < y\ \} \tag{4.8}$$
$$N2 = \{\ \overline{n^x_y(n+1)^x_y} \mid n, x, y \in \mathbb{N}, x+1 < y, n < 2(y-x)-2\ \} \tag{4.9}$$
$$N3 = \{\ \overline{n^x_y y} \mid n, x, y \in \mathbb{N}, x+1 < y, n = 2(y-x)-2\ \} \tag{4.10}$$
$$O1 = \{\ x(x+1)1^x_{x+2}1^x_{x+3}\ldots \mid x \in \mathbb{N}\ \} \tag{4.11}$$
$$O2 = \{\ n^x_y(n+1)^x_y \mid n, x, y \in \mathbb{N}, x+1 < y, n < 2(y-x)-2\ \} \tag{4.12}$$
$$O3 = \{\ (n^x_y y \mid n, x, y \in \mathbb{N}, x+1 < y, n = 2(y-x)-2\ \} \tag{4.13}$$

$$\text{NAND} = N1a \cup N1b \cup N2 \cup N3 \qquad \text{OR} = O1 \cup O2 \cup O3 \tag{4.14}$$

One can identify the different axioms by looking at the figure above[1]: $N1a$ are the edges making up the core ray of the graph, $N1b$ are the edges going from a core vertex onto a peripheral vertex, $N2$ are the edges between two peripheral vertices and $N3$ are the edges going from a peripheral vertex back onto a core vertex. $O1$ are the vertices on the core ray, $O2$ and $O3$ are the peripheral vertices, $O3$ being the ones that points back to a core vertex.

In order to prove $\varnothing$ from these axiom, we first prove three sets of intermediate clauses:

$$D1 = \{\ \overline{xy} \mid x, y \in \mathbb{N}, x \neq y\ \} \tag{4.15}$$
$$D1b = \{\ \overline{1^x_y n^x_y} \mid x, y \in \mathbb{N}, x+1 < y, n = 2(y-x)-2\ \} \tag{4.16}$$
$$D2 = \{\ \overline{x1^y_z} \mid x, y, z \in \mathbb{N}, x \neq z, y+1 < z\ \} \tag{4.17}$$
$$D3 = \{\ \overline{1^1_x 1^y_z} \mid x, y, z \in \mathbb{N}, x \neq z, y+1 < z\ \} \tag{4.18}$$

4.2.1 Proof, D1 and D1b

We already have $\overline{xy}$ from our axioms whenever $y = x \pm 1$, so let us assume that either $y < x-1$ or $x+1 < y$. In both these cases, we get that there exists a peripheral path between x and y. Let us – without loss in generality – assume that $x+1 < y$. We thus have a path from x to y containing $n = 2 \times (y-x) - 2$ peripheral vertices. Consider now the following proof:

[1] recall how NANDs corresponds to edges, while ORs corresponds to vertices

$$\dfrac{\text{(O2) } 1^x_y 2^x_y \quad \dfrac{(N1b)\ \overline{x1^x_y} \qquad (N2)\ \overline{2^x_y 3^x_y}}{\overline{x3^x_y}} \quad \dfrac{(N2)\ \overline{4^x_y 5^x_y}}{\overline{x5^x_y}} \quad \dfrac{(N2)\ \overline{6^x_y 7^x_y}}{\overline{x7^x_y}} \quad \vdots \quad \dfrac{(N3)\ \overline{n^x_y y}}{\overline{x(n-1)^x_y}}}{\overline{xy}}$$

Intuitively, this proof follows along the peripheral path between the two core vertices. It is important to notice that since all our peripheral paths are odd in length, $(n-1)$ is always odd. This guarantees that we are able to prove $\overline{x(n-1)^x_y}$ in $n/2$ steps using the strategy above.

Using x, y, n from above, the proof of D1b follows a similar pattern as D1:

$$\dfrac{\text{(O2) } 2^x_y 3^x_y \quad \dfrac{(N2)\ \overline{1^x_y 2^x_y} \qquad (N2)\ \overline{3^x_y 4^x_y}}{\overline{1^x_y 4^x_y}} \quad \dfrac{(N2)\ \overline{5^x_y 6^x_y}}{\overline{1^x_y 6^x_y}} \quad \dfrac{(N2)\ \overline{7^x_y 8^x_y}}{\overline{1^x_y 8^x_y}} \quad \vdots \quad \dfrac{(N2)\ \overline{(n-1)^x_y n^x_y}}{\overline{1^x_y (n-2)^x_y}}}{\overline{1^x_y n^x_y}}$$

Just like with D1, it is crucial that the path is odd in order for this proof strategy to work.

4.2.2 Proof, D2

Again, we have a trivial case where $x = y$ making the clause simply an instance of $N1b$. We therefore continue under the assumption that $x \neq y$, together with the other restrictions from the definition of $D2$. Using the same notation as above, where n is the number of vertices in the peripheral path between y and z, we now get the following short proof:

$$\dfrac{\text{(O3) } n^y_z z \quad \dfrac{(D1)\ \overline{xz} \qquad (D1b)\ \overline{1^y_z n^y_z}}{\overline{x1^y_z}}}{}$$

Using $D1$ and $D1b$ in the above proof, restricts us to cases where $x \neq z$ and where $y + 1 < z$, but these are restrictions we have already assumed.

4.2.3 Proof, D3

n now denotes the number of vertices in the peripheral path between y and z.

$$\text{(O3) } n_z^y z \quad \frac{\dfrac{(D2)}{\overline{1_x^1 z}} \qquad \dfrac{(D1b)}{\overline{1_z^y n_z^y}}}{\overline{1_x^1 1_z^y}}$$

We are again restricting our cases by using $D3$ and $D1b$, but these restrictions are already assumed.

4.2.4 Proof, $\varnothing$

We will prove the inconsistency of Stretched Yablo by first proving $\overline{1}, \overline{2}, \overline{1_3^1}, \overline{1_4^1 1_5^1} \ldots$.

Proving $\overline{1}$:

$$\text{(O1) } 231_4^2 1_5^2 1_6^2 \ldots \quad \frac{\dfrac{(N1)}{\overline{12}} \quad \dfrac{(D1)}{\overline{13}} \quad \dfrac{(D2)}{\overline{11_4^2}} \quad \dfrac{(D2)}{\overline{11_5^2}} \quad \dfrac{(D2)}{\overline{11_6^2}} \quad \ldots}{\overline{1}}$$

Proving $\overline{2}$:

$$\text{(O1) } 341_5^3 1_6^3 1_7^3 \ldots \quad \frac{\dfrac{(N1)}{\overline{23}} \quad \dfrac{(D1)}{\overline{24}} \quad \dfrac{(D2)}{\overline{11_5^3}} \quad \dfrac{(D2)}{\overline{11_6^3}} \quad \dfrac{(D2)}{\overline{11_7^3}} \quad \ldots}{\overline{2}}$$

Proving $\overline{1_3^1}$:

$$\text{(O1) } 451_6^4 1_7^4 1_8^4 \ldots \quad \frac{\dfrac{(D2)}{\overline{1_3^1 4}} \quad \dfrac{(D2)}{\overline{1_3^1 5}} \quad \dfrac{(D3)}{\overline{1_3^1 1_6^4}} \quad \dfrac{(D3)}{\overline{1_3^1 1_7^4}} \quad \dfrac{(D3)}{\overline{1_3^1 1_8^4}} \quad \ldots}{\overline{1_3^1}}$$

Proving $\overline{1_4^1}$:

$$\text{(O1) } 561_7^5 1_8^5 1_9^5 \ldots \quad \frac{\dfrac{(D2)}{\overline{1_4^1 5}} \quad \dfrac{(D2)}{\overline{1_4^1 6}} \quad \dfrac{(D3)}{\overline{1_3^1 1_7^5}} \quad \dfrac{(D3)}{\overline{1_3^1 1_8^5}} \quad \dfrac{(D3)}{\overline{1_3^1 1_9^6}} \quad \ldots}{\overline{1_4^1}}$$

The clauses $\overline{1_4^1}$, $\overline{1_5^1}$, $\overline{1_5^1}$, $\overline{1_7^1}$, ... can be proven in exactly the same manner as above. With the results from above we can finally prove $\varnothing$:

$$\text{(O1) } 121_3^1 1_4^1 1_5^1 \ldots \quad \frac{\overline{1} \quad \overline{2} \quad \overline{1_3^1} \quad \overline{1_4^1} \quad \overline{1_5^1} \quad \ldots}{\varnothing}$$

Having proved $\varnothing$ in Neg, we get from soundness of Neg that our Stretched Yablo-graph is indeed inconsistent.

4.3 Provability of NAND-clauses from vels

This section will prove the following statement: Given a graph where two vertices a and b are connected by a vel, as defined in Section 2.4, the binary NAND-clause $\overline{ab}$ is provable in Neg.

Let $\mathbf{G} = (G, N)$ be a graph containing the vertices a, b such that they have a vel between them. By definition, this means that there exists a vertex c such that there is an oddly trimmed path from a to c and from b to c, one of odd length and one of even length.

Let P and Q be the two sets containing the vertices in the path from a to c and from b to c, respectively. We will denote each element of P as p_i where $i \in \mathbb{N}$ is the position of that vertex in the path, starting at 0. The elements of Q will be named q_i by the same rule. We immediately have that $a = p_0$ and $b = q_0$. As long as the trimming restrictions are met, P and Q might overlap, i.e. we might have cases where $p_j = q_k$ from some i and j.

We assume, without any loss of generality, that the path from a to c is of odd length, making the path from b to c. We denote the lengths of the two paths by the numbers n and m, giving us that $c = p_n = q_m$. The path from b to c might also be empty, making $b = p_0 = c$.

This general variant of a vel can be illustrated in the following way, where the possibly branching vertices are shown with dashed edges out of them.

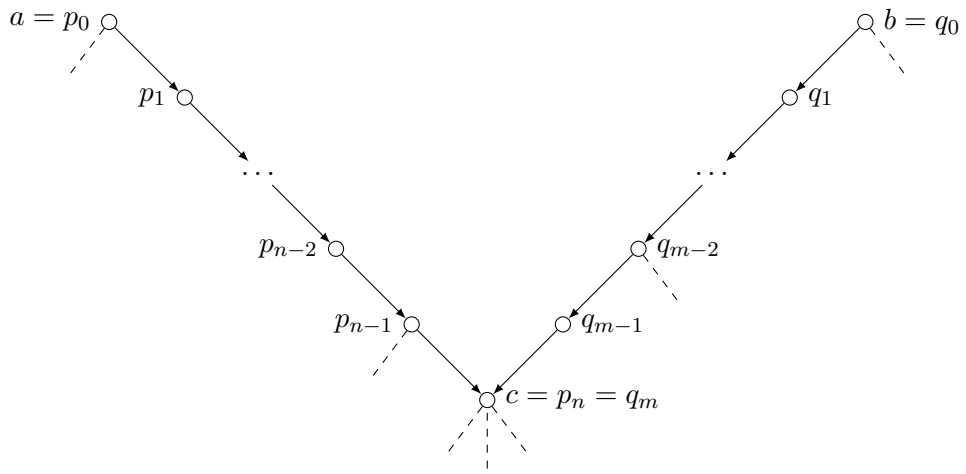


Figure 4.2: A general vel between a and b

Let NAND_G and OR_G denote the axiomatic NAND- and OR-clauses we get from our graph. We can work out the following subset of NAND_G:

$$\text{NAND}_V = \{\ \overline{p_i p_{i+1}} \mid 0 \leq i < n\ \} \cup \{\ \overline{q_j q_{j+1}} \mid 0 \leq j < m\ \} \tag{4.19}$$

Since both paths are oddly trimmed, every vertex at an odd position in its path will only have one out-edge, resulting in a binary OR-clause. This makes us able to work out the following subset of NAND_G:

$$\mathrm{OR}_V = \{\, p_i p_{i+1} \mid 0 \leq i < n, i \text{ is odd} \,\} \cup \{\, q_j q_{j+1} \mid 0 \leq j < m, j \text{ is odd} \,\} \tag{4.20}$$

The proof of $\overline{ab}$ can now be worked out, based only on the axioms $\Gamma_V = \mathrm{NAND}_V \cup \mathrm{OR}_V$:

$$\dfrac{p_1 p_2 \quad \dfrac{p_3 p_4 \quad \dfrac{\dfrac{\dfrac{}{\overline{ap_1}} \quad \dfrac{}{\overline{p_2 p_3}}}{\overline{ap_3}} \quad \dfrac{}{\overline{p_4 p_5}}}{\overline{ap_5}} \quad \vdots \quad \dfrac{\dfrac{}{\overline{ap_{n-2}}} \quad \dfrac{}{\overline{p_{n-1} c}}}{\overline{ac}}}{\overline{ab}} \quad \dfrac{q_1 q_2 \quad \dfrac{q_3 q_4 \quad \dfrac{\dfrac{\dfrac{}{\overline{bq_1}} \quad \dfrac{}{\overline{q_2 q_3}}}{\overline{bq_3}} \quad \dfrac{}{\overline{q_4 q_5}}}{\overline{bq_5}} \quad \vdots \quad \dfrac{\dfrac{}{\overline{bq_{m-3}}} \quad \dfrac{}{\overline{q_{m-2} q_{m-1}}}}{\overline{bq_{m-1}}}}{\overline{ab}} \quad q_{m-1} c}{\overline{ab}}$$

Figure 4.3: Proof that a and b are vel-connected

All the NAND-clauses used as axioms in the above proof are on the form $\overline{x_i x_{i+1}}$ and thus elements of NAND_V. All the OR-clauses used as axioms are also on the form $x_i x_{i+1}$, and since n is odd and m is even, we see that all the OR-clauses used in the proof are indeed from OR_V.

Observe that the case where $b = c$ is unproblematic, since the above proof also proves $\overline{ac}$. The case where some $p_i = q_j$ does not make any of the axioms change, making it too unproblematic.

All the axioms used are thus in Γ_V, which is a subset of Γ_G, and since all the rule applications are correct, our proof is valid.

Bibliography

[1] G. Priest and F. Berto, "Dialetheism," in *The Stanford Encyclopedia of Philosophy*, E. N. Zalta, Ed., Summer 2013, 2013. [Online]. Available: `http://plato.stanford.edu/archives/sum2013/entries/dialetheism/`.

[2] S. Dyrkolbotn and M. Walicki, "Propositional discourse logic," *Synthese*, vol. 191, pp. 863–899, 2014.

[3] M. Bezem, C. Grabmeyer, and M. Walicki, "Expressive power of digraph solvability," *Annals of Pure and Applied Logic*, vol. 163, pp. 200–213, 2012.

[4] J. von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior*. Princeton University Press, 1944 (1947).

[5] V. Chvátal, "On the computational complexity of finding a kernel. technical report crm-300," Centre de Recherches Mathématiques, Univeristé de Montréal, Tech. Rep., 1973.

[6] R. T. Cook, "Patterns of paradox," *The Journal of Symbolic Logic*, vol. 69(3), pp. 767–774, 2004.

[7] M. Richardson, "Solutions of irreflexive relations," *The Annals of Mathematics, Second Series*, vol. 58(3), pp. 573–590, 1953.

[8] S. Yablo, "Paradox without self-reference," *Analysis*, vol. 53(4), pp. 251–252, 1993.