Master WIP

Kjetil Midtgarden Golid

February 17, 2016

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:

$$x \land \neg x \tag{1}$$

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 does, however, seem like a stretch to say that all inconsistent theories are paradoxes. Just consider the inconsistent statement, $x \land \neg x$ 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:

$$\text{"\textit{This sentence is false}".} \tag{2}$$

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. Note how the liar sentence is a statement about other statements (in this case itself). In order to study these kinds of meta-statements, we need a way to reference other statements within a statement. In propositional logic, we can do this by giving statements "names" in the form of adding fresh variables with equivalences to their correponding statements[1]. Consider the left statements below, together with their corresponding named statements on the right.

$$a \qquad\qquad x_1 \leftrightarrow a \tag{3}$$
$$a \land \neg a \qquad\qquad x_2 \leftrightarrow a \land \neg a \tag{4}$$
$$a \lor \neg a \qquad\qquad x_3 \leftrightarrow a \lor \neg a \tag{5}$$

By performing this naming-operation on these statements, one is obviously changing 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 in all the cases above can find a truth value for x_i that matches the one of the corresponding statement. This will not be the case for paradoxes, so our new named statements will be consistent if and only if they are not paradoxical.

Consider the liar sentence. It can be written as a named statement in the following way:

$$x \leftrightarrow \neg x \tag{6}$$

This statement is obviously inconsistent, making it a paradox by our newly acquired definition. The study of *discourses* takes this formalization a step further.

1.2 Discourses

A propositional theory is in **graph normal form (GNF)** if all its formulae have the following form:

$$x \leftrightarrow \bigwedge_{i \in I_x} \neg y_i \tag{7}$$

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 conjunctive normal from to graph normal form (shown in the end of this section), showing that any propositional theory has an equisatisfiable GNF theory. By interpreting the variable on the left as the name of the statement on the right, like shown earlier, one can start using GNF to model meta-statements.

We now formally define a **discourse** to be a theory in GNF, and a **paradox** to simply be an inconsistent discourse.

We will later show a very handy correspondence between these discourse theories and certain graphs. The correspondence lets us not only decide the satisfiability of a discourse

[1]bad wording?

theory (i.e. whether or not it is paradoxical) by looking at certain properties of the corresponding graph. The properties in the graph also provide us with the satisfying models, 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 predicates formally as follows:

$$N(x) := \{y \mid (x, y) \in N\} \tag{8}$$
$$N^{-}(x) := \{y \mid (y, x) \in N\} \tag{9}$$

A *path* is a sequence of unique vertices $x_1, x_2, \ldots, x_n$ such that for any consecutive pair x_i, x_{i+1} from the sequence, we have $\langle x_i, x_{i+1} \rangle \in N$. We say that two paths are *disjoint* if they do not share any vertices (possibly with the exception of their initial nodes). We will be using these notions later.

We can extend the predicates N and N^{-} to sets of vertices in the following way:

$$N(X) = \bigcup_{x \in X} N(x) \tag{10}$$
$$N^{-}(X) = \bigcup_{x \in X} N(x) \tag{11}$$

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

$$G \setminus K = N^{-}(K) \tag{12}$$

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

$G \setminus K \subseteq N^{-}(K)$, saying that each vertex outside the kernel has to have an edge into the kernel (K is absorbing).

$N^{-}(K) \subseteq G \setminus K$, 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 of great interest over several decades, mainly within the fields of game theory and economics[2]. In a graph representing some sort of a turn-based game, where vertices are states and edges are transitions, 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 inside the kernel, forcing the opponent to move out again.

Deciding the existence of kernels in a graph has been shown to be an NP-complete[3] problem. 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[2]), which we know is NP-complete.

We will get the correspondence between satisfying 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 $x \in G$:

$$\alpha(x) = 1 \iff \alpha(N(x)) = \{0\} \tag{13}$$

In simple words, this means that for any node x, if x is assigned 1, then all its successors has 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 to 1, since it vacuously does not point to any node assigned to 1. We use the notation $sol(\mathbf{G})$ to denote the set of all solutions of the graph $\mathbf{G}$.

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 [4]. We get the following definitions from [5]

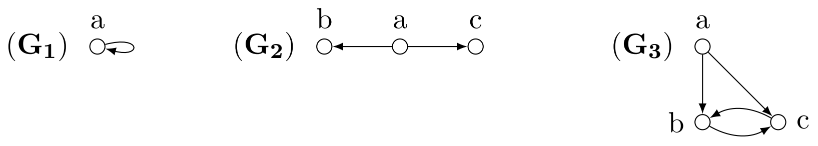
$\mathcal{T}$: translating a digraph $\mathbf{G}$ into a corresponding theory $\mathcal{T}(\mathbf{G})$ such that $sol(\mathbf{G}) = mod(\mathcal{T}(G))$.

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

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

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.



Using the graphs from above, we get the following theories using $\mathcal{T}$:

$$\mathcal{T}(\mathbf{G_1}) = \{a \leftrightarrow \neg a\} \tag{14}$$
$$\mathcal{T}(\mathbf{G_2}) = \{a \leftrightarrow (\neg b \wedge \neg c), b, c\} \tag{15}$$
$$\mathcal{T}(\mathbf{G_3}) = \{a \leftrightarrow (\neg b \wedge \neg c), b \leftrightarrow \neg c, c \leftrightarrow \neg b\} \tag{16}$$

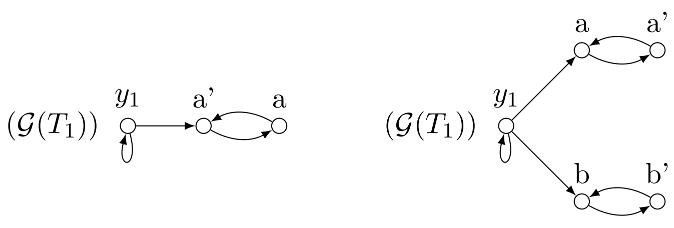
The fact that $sol(\mathbf{G}) = mod(\mathcal{T}(G))$ is shown in [5]. Allthough not proving it, we can observe that $\mathbf{G_1}$ has no solution, just like its corresponding theory $\mathcal{T}(\mathbf{G_1})$ has no satisfying models. $\mathbf{G_2}$ has one solution, where one assigns $a = 0, b = 1, c = 1$. This assignment also works as a satisfying model for $\mathcal{T}(\mathbf{G_2})$. In $\mathbf{G_3}$, we get two solutions, both with a assigned to 0, but with 0 and 1 distributed on b and c. These are also the only two models satisfying $\mathcal{T}(\mathbf{G_3})$.

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 $x \leftrightarrow \bigwedge_{i \in I_x} y_i$ make a directed edge $\langle x, y_i \rangle$ for each $i \in I_x$.

Example 2.

$$T_1 = \neg a \iff (a \leftrightarrow \neg a'), (a' \leftrightarrow \neg a), (y_1 \leftrightarrow (\neg a' \wedge \neg y_1)) \tag{17}$$
$$T_2 = a \vee b \iff (a \leftrightarrow \neg a'), (a' \leftrightarrow \neg a), (b \leftrightarrow \neg b'), (b' \leftrightarrow \neg b), (y_1 \leftrightarrow (\neg a \wedge \neg b \wedge \neg y_1)) \tag{18}$$

Using $\mathcal{G}$ on the above theories – translated to GNF – gives us the following graphs:



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 to 1. This reflects onto the graph since y_1 has to be assigned to 0, thus forcing a or b to be assigned to 1. The fact that $\mathcal{G}$ gives us the correspondence we are looking for is shown in [5].

With the problem of solutions in the graph being equivalent with SAT, we get our final equivalence between kernels in the graph and satisfying models of the theory. This equivalence connects the fields of logic with the fields of 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 explore some of the various findings within Kernel Theory, and especially the findings related to infinitary graphs.

1.5 Results in Kernel Theory

Ultimately, the end goal of Kernel Theory would be to have an easy way to answer the question "Does this digraph have a kernel?" no matter the graph, and no matter the answer. 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 Richardson's Theorem, worked out by Moses Richardson in 1953:

Theorem 3. *[6] If D is a finitary[3] digraph without odd cycles, then D has a kernel.*

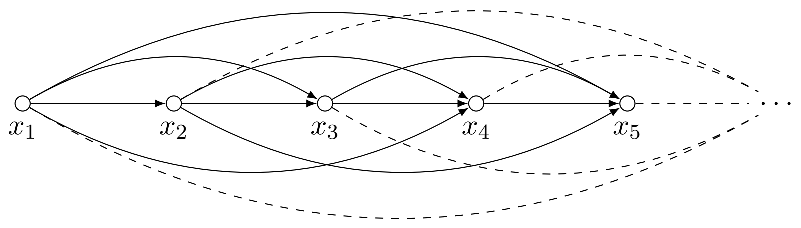
This theorem gives us the confirmation that whenever dealing with finitary dags, for instance, one can be certain that its corresponding theory is consistent.

Intuitively, one is tempted to believe that *all* digraphs without odd cycles have kernels, but this is not the case. Until now, our paradoxes have always 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.

The **Yablo Graph** [7] is an example of an acyclic graph with no kernel. It is constructed with an infinite set of vertices $\{x_i | i \in \mathbb{N}\}$ and a set of edges N such that $\langle x_i, x_j \rangle \in N$ iff

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

$i < j$.

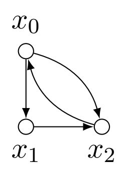


Since there exist no two numbers $x, y \in \mathbb{N}$ such that $x < y$ and $y < x$, we get that the Yablo graph indeed is acyclic[4]. Furthermore, since any natural number has infinitely many numbers strictly larger than it, we get that all the vertices are infinitely branching, making the Yablo graph infinitary (not finitary).

The corresponding discourse theory of 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 that this graph is indeed without a kernel, but for now we will take Yablo's word [7] for it.

One thing should be mentioned at this point; neither odd cycles nor infinitely branching vertices *entails* that their respective graphs are paradoxical. The two following graphs illustrate this point:



(19)

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

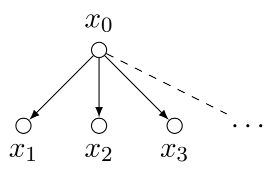


Figure 1

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

[4]This is a bit over-simplified

It is shown that every digraph (with at least one edge) can be transformed into a infinitary dag[5] such that α is a solution to the created dag if and only if it is a solution to the original digraph [5]. 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. Furthermore, if one is trying to find ways to identify paradoxical graphs, one does only need to look at dags. The reason for this is simple; say you have a method of identifying paradoxical dags. Then given any graph, you will be able to decide whether or not it is paradoxical. If it is a dag, you use your method. If it is cyclic, translate it, then use your method. Since the translation preserves and reflects the solutions, one can be sure that the cyclic graph is paradoxical if and only if the resulting dag was paradoxical.

This result will be of great importance to us, enabling us to look only at dags when searching for paradoxes.

1.6 Dags without kernels

Knowing that any graph can translate 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. (20)

Some terminology: A *ray* is an semi-infinite path, i.e. a path containing an initial vertex and an infinite number of other vertices.

A vertex x_0 *dominates* a set of vertices Y if there exists an infinite number of disjoint paths from x_0 to some vertex is Y.

The contrapositive of Walicki's thesis suggests a weaker condition for a kernel, 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 will be presenting an inference system introduced by Walicki in [] which handles clausal theories induced from GNF-theories.

[5]Directed acyclic graph

Recall that a thoeory written in GNF have formulae on the following form:

$$x \leftrightarrow \bigwedge_{i \in I_x} \neg y_i \tag{21}$$

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:

$$x \rightarrow \bigwedge_{i \in I_x} \neg y_i \qquad \text{and} \qquad x \leftarrow \bigwedge_{i \in I_x} \neg y_i \tag{22}$$

The first implication can be rewritten in the following way:

$$x \rightarrow \bigwedge_{i \in I_x} \neg y_i \quad = \quad \neg x \vee \bigwedge_{i \in I_x} \neg y_i \quad = \quad \bigwedge_{i \in I_x} (\neg x \vee \neg y_i) \quad = \quad \bigwedge_{i \in I_x} \neg(x \wedge y_i) \tag{23}$$

The second implication can be rewritten in the following way:

$$x \leftarrow \bigwedge_{i \in I_x} \neg y_i \quad = \quad x \vee \neg \left(\bigwedge_{i \in I_x} \neg y_i \right) \quad = \quad x \vee \bigvee_{i \in I_x} y_i \tag{24}$$

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:

$$\text{OR-clause:} \quad x \vee \bigvee_{i \in I_x} y_i \tag{25}$$

$$\text{NAND-clauses:} \quad \neg(x \wedge y_i), \text{ for every } i \in I_x \tag{26}$$

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 thins like $\overline{xy} \subset \overline{xyz}$. A theory will – as expected – be a set of 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:

$$\mathcal{T}(\mathbf{G_1}) = \{a, \overline{a}\} \tag{27}$$

$$\mathcal{T}(\mathbf{G_2}) = \{abc, b, c, \overline{ab}, \overline{ac}\} \tag{28}$$

$$\mathcal{T}(\mathbf{G_3}) = \{abc, bc, \overline{ab}, \overline{bc}\} \tag{29}$$

Further notation: $A \subseteq G$ denotes an OR-clause while $\overline{A} \subseteq$ 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.

$$\text{(Ax)} \quad \Gamma \vdash C, \quad \text{for } C \in \Gamma \tag{30}$$

$$\text{(Rneg)} \quad \frac{\{\Gamma \vdash \overline{a_i A_i} \mid i \in I\} \quad \Gamma \vdash \{a_i \mid i \in I\}}{\Gamma \vdash \overline{\bigcup_{i \in I} A_i}} \tag{31}$$

$$\text{(Rpos)} \quad \frac{\Gamma \vdash A \quad \{\Gamma \vdash B_i K_i \mid i \in I\} \quad \{\Gamma \vdash \overline{a_i k} \mid i \in I, k \in K_i\}}{\Gamma \vdash (A \setminus \{a_i \mid i \in I\}) \cup \bigcup_{i \in I} B_i} \tag{32}$$

(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.

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

$$(1)\ \frac{\overline{ax} \quad \overline{by} \quad \overline{cz}}{\overline{xyz}} abx \qquad (3)\ \frac{\overline{ax} \quad \overline{by}}{\overline{xy}} abx \qquad (2)\ \frac{\overline{ax} \quad \overline{by} \quad \overline{bz}}{\overline{xyz}} abx \tag{33}$$

(1) is incorrect because the NAND $\overline{cz}$ contains no atoms from the OR ab. (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 such that the above requirements are met.

$$(4)\ \frac{\overline{ax} \quad \overline{by} \quad \overline{cz}}{\overline{xyz}} abc \qquad (5)\ \frac{\overline{ax} \quad \overline{b}}{\overline{x}} ab \qquad (6)\ \frac{\overline{ax} \quad \overline{by} \quad \overline{xyz}}{\overline{xyz}} abx \tag{34}$$

The three above applications are all unproblematic, since all the atoms in each OR-clause gets 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 [] that Neg is sound and refutational complete for theories with only a finite 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 $\emptyset$ in Neg.

Having these properties, we are hoping that some proof structural patterns found in Neg proofs could ultimately help us in further weakening the conditions for a kernel. The remainder of this paper will be focused mainly on exploring such patterns.

1.8 Proofs

1.8.1 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, start with an empty theory R and for each formula in P, follow the steps below to acquire its corresponding GNF formulae.

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

Step 2: In each clause, replace every negative literal $\neg x_i$ with its corresponding x'_i fom step 1. Every clause does now contain all positive literals. For every clause $(x_1 \vee x_2 \vee \cdots \vee x_n)$, create a fresh variable y and add the following GNF formula to R:

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

The combined set of all these acquired formulae will make up the the corresponding theory R. 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.

Example 4.

$$\text{CNF:} \quad (a \vee b) \tag{35}$$

$$\text{GNF:} \quad (a \leftrightarrow \neg a'), (a' \leftrightarrow \neg a), (b \leftrightarrow \neg b'), (b' \leftrightarrow \neg b), (y_1 \leftrightarrow (\neg a \wedge \neg b \wedge \neg y_1)) \tag{36}$$

$$\tag{37}$$

$$\text{CNF:} \quad (\neg a) \tag{38}$$

$$\text{GNF:} \quad (a \leftrightarrow \neg a'), (a' \leftrightarrow \neg a), (y_1 \leftrightarrow (\neg a' \wedge \neg y_1)) \tag{39}$$

$$\tag{40}$$

References

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

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

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

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

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

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