Gödel-Dummett counter-models through matrix computation

Dominique Larchey-Wendling

WoD 2004 Preliminary Version

Gödel-Dummett counter-models through matrix computation

Dominique Larchey-Wendling [1]

LORIA – CNRS
Vandœuvre-lès-Nancy, France

Abstract

We present a new method for deciding Gödel-Dummett logic. Starting from a formula, it proceeds in three steps. First build a conditional graph based on the decomposition tree of the formula. Then try to remove some cycles in this graph by instantiating these boolean conditions. In case this is possible, extract a counter-model from such an instance graph. Otherwise the initial formula is provable. We emphasize on cycle removal through matrix computation, boolean constraint solving and counter-model extraction.

Key words: Counter-models, conditional graphs and matrices.

1 Introduction

Gödel-Dummett logic LC is the intermediate logic (between classical logic and intuitionistic logic) characterized by linear Kripke models. It was introduced by Gödel in [10] and later axiomatized by Dummett in [6]. It is now one of the most studied intermediate logics for several reasons: among those, it is one of the *simplest "many-valued" logics,* whose semantics is captured by truth functions over the unit interval. It is one of the candidates (under the name "Gödel" logic) for use as a *fuzzy logic* [11]. With respect to decision procedures for intermediate logics, it witnesses some advantages of sequent calculi over hyper-sequent systems.

Proof-search in LC has benefited from the development of proof-search in intuitionistic logic IL with two important seeds: the *contraction-free calculus* of Dyckhoff [1,7,8] and the *hyper-sequent* calculus of Avron [2,14]. Two of the most recent contributions propose a similar approach based on a set of *local* and *strongly invertible* proof rules (for either sequent [13] or hyper-sequent [2] calculus,) and a semantic criterion to decide *irreducible (hyper)-sequents* and eventually build a counter-model.

[1] Email: larchey@loria.fr

We have recently proposed a combination of proof-search in sequent calculus and counter-model construction to provide a decision procedure for LC which is based on a new principle: we are able to gather all the useful information arising from all the proof-search branches into a semantic graph and then we use an efficient counter-model search algorithm based on cycle detection. We have reduced the decision problem in LC to a combination of boolean constraint solving and cycle detection. These results are presented in the upcoming paper [12], and the present paper comes as a complement to it. We will briefly recall the theoretical results, but we want to focus mainly on the description of the decision procedure with an emphasis on the counter-model generation algorithm.

Given a formula D of LC, the procedure proceeds in three steps at the end of which one obtains a counter-model (in case D is not provable.[2]) The first step which is described in full details in section 3 consists in building a particular *bi-colored graph* $\mathcal{G}_D$ based on the decomposition tree of D. The arrows of this graph may be indexed with *boolean conditions*. The second step consists in searching for an instantiation of the boolean conditions on arrows so that the *instance graph* has no remaining *r-cycle*.[3] This step is first described informally in section 3.3; then we provide a decision algorithm for this problem based on conditional matrix computation in section 4. For the third step, described in section 5, given a particular instance $\mathcal{G}_v$ with no r-cycle, we can extract a counter-model of D from this instance $\mathcal{G}_v$ by computing a *bi-height* for it.

2 The syntax and semantics of Gödel-Dummett logic

The set of propositional *formulae*, denoted Form is defined inductively, starting from a set of propositional *variables* denoted by Var and using the connectives $\wedge$, $\vee$ and $\supset$.[4] IL will denote the set of formulae that are provable in any intuitionistic propositional calculus (see [7]) and CL will denote the classically valid formulae. As usual an *intermediate propositional logic* [1] is a set of formulae $\mathcal{L}$ satisfying $\mathsf{IL} \subseteq \mathcal{L} \subseteq \mathsf{CL}$ and closed under the rule of modus ponens and under arbitrary substitution. LC is the smallest intermediate logic satisfying the axiom $(X \supset Y) \vee (Y \supset X)$.

On the semantic side, LC is characterized by linear Kripke models. In this paper, we will use the algebraic semantics characterization of LC [2] rather than Kripke semantics. The algebraic model is the set of natural numbers with its natural order $\leqslant$, augmented with a greatest element ∞. An interpretation of propositional variables $[\![\cdot]\!] : \mathsf{Var} \to \overline{\mathbb{N}}$ is inductively extended to formulae: the conjunction $\wedge$ is interpreted by the *minimum* function denoted

[2] As a decision procedure, it can also certify the validity of D in case it has a proof.

[3] A kind of cycle described later in section 5.

[4] We do not integrate the bottom $\bot$ constant. A specific treatment for $\bot$ is detailed in [13]. It can be easily integrated in the procedure described here.

$\wedge$, the disjunction $\vee$ by the *maximum* function $\vee$ and the implication $\supset$ by the operator $\twoheadrightarrow$ defined by $a \twoheadrightarrow b = $ if $a \leqslant b$ then ∞ else b. A formula D is *valid* for the interpretation $[\![\cdot]\!]$ if the equality $[\![D]\!] = \infty$ holds. This interpretation is complete for LC. A *counter-model* of a formula D is an interpretation $[\![\cdot]\!]$ such that $[\![D]\!] < \infty$.

3 A decision procedure for LC

In [12], we have described a procedure to decide the formulae of LC and to build a counter-model when a formula is not valid. The first step of this procedure is to build a graph with two kinds of arrows. Then the decision problem is reduced to the detection of particular cycles in this graph.

3.1 Conditional bi-colored graph construction

We introduce the exact notion of graph we use and then show how to build such a graph given a formula of LC.

Definition 3.1 A *bi-colored graph* is a (finite) directed graph with to kinds of arrows: *green* arrows denoted by $\rightarrow$ and *red* arrows denoted by $\Rightarrow$.

Definition 3.2 A *conditional bi-colored graph* is a bi-colored graph where arrows may be indexed with (propositional) boolean expressions.

We point out that we consider these boolean expressions up to classical equivalence, i.e. we consider them as representatives for boolean functions over atomic propositional variables. These variables can be instantiated by $\{0, 1\}$ with a valuation v and a boolean expression e gets a value $e_v \in \{0, 1\}$ computed in the obvious way. We thus obtain an instance graph: an arrow indexed with a boolean expression e belongs to this instance if and only if $e_v = 1$. The case of an unconditional (i.e. not indexed) arrow can be treated by considering that it has an implicit boolean conditional which is a tautology (and then always values 1) and non-existing arrows have an implicit boolean condition that always values 0.

Definition 3.3 Given a conditional bi-colored graph $\mathcal{G}$ and a valuation v of boolean variables in $\{0, 1\}$, we define the *instance graph* $\mathcal{G}_v$ as the bi-colored graph that one obtains when one evaluates boolean expressions indexing arrows and keeping exactly those whose valuation equals 1.

Given a LC formula D, we build a conditional bi-colored graph $\mathcal{G}_D$ by the following process. First, the nodes of $\mathcal{G}_D$ are obtained by considering the set of nodes of the decomposition tree of D, or equivalently, the set of occurrences of subformulae.

- If F is an occurrence of a subformula of D, we denote by $\mathcal{X}_F$ the corresponding node. Nodes are *signed* starting from $-$ at the root D^- and propagating

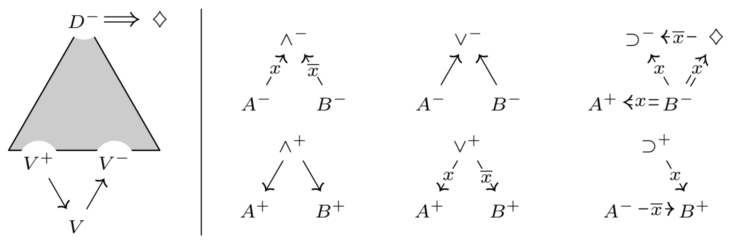


Fig. 1. Counter-model search system for LC

signs as usual.[5] We may write $\mathcal{X}_F^+$ or $\mathcal{X}_F^-$ to emphasize the sign.

- To this set of nodes, we add a node denoted V for each propositional variable V occurring in D. Hence, multiple occurrences of V only generate one node V but generate several $\mathcal{X}_V^+$ or $\mathcal{X}_V^-$ nodes.
- We add one new node denoted by $\Diamond$.

Then, the edges of $\mathcal{G}_D$ are obtained as follows: we describe the set of green and red arrows linking those nodes together and the boolean expressions indexing those arrows. We begin by unconditional arrows (i.e. arrows implicitly indexed with the tautology 1) introduced independently of the internal structure of the formula D:

- We add the (unconditional) red arrow $\mathcal{X}_D^- \Rightarrow \Diamond$ from the root node to $\Diamond$.
- For a negative occurrence V of a variable, we add the green arrow $V \to \mathcal{X}_V^-$.
- For a positive occurrence V of a variable, we add the green arrow $\mathcal{X}_V^+ \to V$.

These three rules are summarized on the left part of figure 1. Now we consider arrow introduction rules for internal nodes. First, the unconditional cases:

- For a positive occurrence $C \equiv A \wedge B$ of a subformula, we add the two following green arrows $\mathcal{X}_C^+ \to \mathcal{X}_A^+$ and $\mathcal{X}_C^+ \to \mathcal{X}_B^+$.
- For a negative occurrence $C \equiv A \vee B$ of a subformula, we add the two following green arrows $\mathcal{X}_A^- \to \mathcal{X}_C^-$ and $\mathcal{X}_B^- \to \mathcal{X}_C^-$.

We continue with conditional arrows. These arrows are indexed with *selectors*, i.e. boolean expressions of the form x or $\overline{x}$ where x is a boolean propositional variable. For each occurrence of subformula, we introduce a *new* boolean variable.[6]

- For a negative occurrence $C \equiv A \wedge B$ of a subformula, given a new boolean variable x, we introduce the two conditional green arrows $\mathcal{X}_A^- \to_x \mathcal{X}_C^-$ and $\mathcal{X}_B^- \to_{\overline{x}} \mathcal{X}_C^-$.
- For a positive occurrence $C \equiv A \vee B$ of a subformula, given a new boolean variable x, we introduce the two conditional green arrows $\mathcal{X}_C^+ \to_x \mathcal{X}_A^+$ and $\mathcal{X}_C^+ \to_{\overline{x}} \mathcal{X}_B^+$.

[5] The connectives $\wedge$ and $\vee$ preserve signs and $\supset$ preserves the sign on the right subformula and inverses the sign of the left subformula.

[6] Indexing these variables with the subformula occurrence is a way to ensure uniqueness.

- For a negative occurrence $C \equiv A \supset B$ of a subformula, given a new boolean variable x, we introduce the two following green arrows $\mathcal{X}_B^- \rightarrow_x \mathcal{X}_C^-$ and $\diamondsuit \rightarrow_{\overline{x}} \mathcal{X}_C^-$ and the two following red arrows $\mathcal{X}_B^- \Rightarrow_x \mathcal{X}_A^+$ and $\mathcal{X}_B^- \Rightarrow_x \diamondsuit$.
- For a positive occurrence $C \equiv A \supset B$ of a subformula, given a new boolean variable x, we introduce the two following green arrows $\mathcal{X}_C^+ \rightarrow_x \mathcal{X}_B^+$ and $\mathcal{X}_A^- \rightarrow_{\overline{x}} \mathcal{X}_B^+$.

All the rules introducing (un)conditional arrows for internal nodes (corresponding to subformulae of D that are not atomic) are summarized on the right part of figure 1.

Given this construction procedure, it should be clear that the construction of the graph $\mathcal{G}_D$ from a formula D take linear time as at most four arrows are introduced for each instance of a subformula of D. The validity of D is related to the existence of some particular cycles in instances of $\mathcal{G}_D$.

Definition 3.4 A *r-cycle* in a bi-colored graph is a cycle composed of either green ($\rightarrow$) or red ($\Rightarrow$) arrows, containing at least one red arrow. Equivalently, it is a chain of the form $l\,(\rightarrow + \Rightarrow)^\star \Rightarrow l$.

Theorem 3.5 *Let D be a formula of* LC *and $\mathcal{G}$ be its associated conditional bi-colored graph, built from the process previously described. Then D is provable in* LC *if and only if every instance graph $\mathcal{G}_v$ of $\mathcal{G}$ contains at least one r-cycle.*

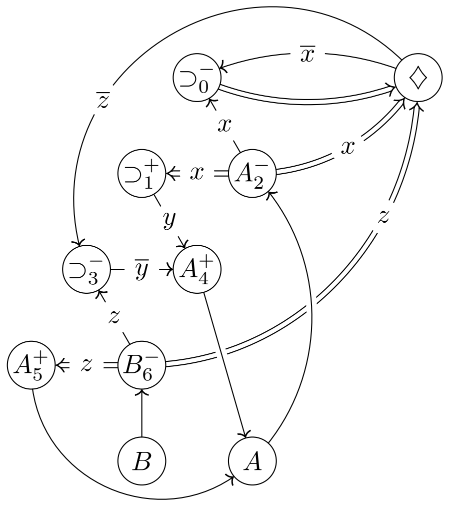
This result is proved in [12]. So in order to refute D, we have to find an instance graph $\mathcal{G}_v$ which does not contain any r-cycle. Let us proceed with an example.

3.2 Graph construction example

We consider the case of the classically valid Peirce's formula $((A \supset B) \supset A) \supset A$. It is not provable in any intermediate logic but classical logic, so in particular, it should have a counter-model in LC.

We index this formula as follows: $\big((A_5^+ \supset_3^- B_6^-) \supset_1^+ A_4^+\big) \supset_0^- A_2^-$. We construct its associated conditional graph:

- We add the arrow $\supset_0^- \Rightarrow \diamondsuit$.
- We have two variables A and B for 4 occurrences, so we add $A \rightarrow A_2^-$, $A_4^+ \rightarrow A$, $A_5^+ \rightarrow A$ and $B \rightarrow B_6^-$.
- For the internal node $\supset_0^-$, we choose a new boolean variable x and add the four conditional arrows $A_2^- \rightarrow_x \supset_0^-$, $\diamondsuit \rightarrow_{\overline{x}} \supset_0^-$, $A_2^- \Rightarrow_x \supset_1^+$ and $A_2^- \Rightarrow_x \diamondsuit$.



- For the internal node $\supset_1^+$, we choose a new boolean variable y and add the two conditional arrows $\supset_1^+ \rightarrow_y A_4^+$ and $\supset_3^- \rightarrow_{\overline{y}} A_4^+$.
- For the last internal node $\supset_3^-$, we choose a new boolean variable z and add the four conditional arrows $B_6^- \rightarrow_z \supset_3^-$, $\Diamond \rightarrow_{\overline{z}} \supset_3^-$, $B_6^- \Rightarrow_z A_5^+$ and $B_6^- \Rightarrow_z \Diamond$.

3.3 Naive elimination of r-cycles

We now have to find a valuation v_x, v_y and v_z in $\{0,1\}$ such that the corresponding instance graph has no r-cycle. For this we identify all the r-cycles and we try to find a valuation that simultaneously breaks each of the r-cycles. We only have to consider r-cycles that do not repeat nodes because any r-cycle contains at least one that does not repeat nodes. We find four such r-cycles:

$$\begin{gathered}
\supset_0^- \Rightarrow \Diamond \rightarrow_{\overline{x}} \supset_0^- \\
\supset_0^- \Rightarrow \Diamond \rightarrow_{\overline{z}} \supset_3^- \rightarrow_{\overline{y}} A_4^+ \rightarrow A \rightarrow A_2^- \rightarrow_x \supset_0^- \\
A_2^- \Rightarrow_x \Diamond \rightarrow_{\overline{z}} \supset_3^- \rightarrow_{\overline{y}} A_4^+ \rightarrow A \rightarrow A_2^- \\
A_2^- \Rightarrow_x \supset_1^+ \rightarrow_y A_4^+ \rightarrow A \rightarrow A_2^-
\end{gathered}$$

The first r-cycle is broken if and only if the condition $\overline{x} = 0$ is satisfied, which is equivalent to satisfy x. The second r-cycle is broken if and only if the condition $z + y + \overline{x}$ is satisfied.[7] The third r-cycle is broken just in case $\overline{x} + z + y$ is satisfied and the last r-cycle is broken when $\overline{x} + \overline{y}$ is satisfied.

In order to break these four r-cycles in one valuation, we look for a valuation v which satisfies $x \cdot (\overline{x} + y + z) \cdot (\overline{x} + y + z) \cdot (\overline{x} + \overline{y})$. This gives us a unique solution: $v_x = 1, v_y = 0, v_z = 1$. Then, the reader could verify that the instance graph $\mathcal{G}_v$ obtained from this valuation has no r-cycle. See section 5 for a representation of this graph and the associated counter-model of the Peirce's formula.

The naive procedure we have described for computing a valuation with no r-cycles consists of searching all the possible r-cycles (without repeating nodes) and solving a boolean constraint system associated with these cycles. Unfortunately, such a procedure would be highly inefficient because there might be exponentially many r-cycles for a given formula. This problem has also been addressed in [12]. In the next section, we give a description of one possible solution to the elimination of r-cycles.

4 Removing r-cycles in conditional bi-colored graphs

In section 3.1, we have introduced the notion of conditional bi-colored graph. A natural way to represent a directed graph is by considering the matrix of the underlying *incidence* relation. Usually, these matrices take there values

[7] We denote by + the boolean disjunction and by · the boolean conjunction.

$$\begin{array}{c|cccccccccc}
\to & 0 & 1 & 2 & 3 & 4 & 5 & 6 & A & B & \diamondsuit \\ \hline
0 & & & & & & & & & & \\
1 & & & & & y & & & & & \\
2 & x & & & & & & & & & \\
3 & & & & & \overline{y} & & & & & \\
4 & & & & & & & & 1 & & \\
5 & & & & & & & & 1 & & \\
6 & & & & z & & & & & & \\
A & & & 1 & & & & & & & \\
B & & & & & & & 1 & & & \\
\diamondsuit & \overline{x} & & & \overline{z} & & & & & &
\end{array}
\qquad
\begin{array}{c|cccccccccc}
\Rightarrow & 0 & 1 & 2 & 3 & 4 & 5 & 6 & A & B & \diamondsuit \\ \hline
0 & & & & & & & & & & 1 \\
1 & & & & & & & & & & \\
2 & & x & & & & & & & & x \\
3 & & & & & & & & & & \\
4 & & & & & & & & & & \\
5 & & & & & & & & & & \\
6 & & & & & & z & & & & z \\
A & & & & & & & & & & \\
B & & & & & & & & & & \\
\diamondsuit & & & & & & & & & &
\end{array}$$

Fig. 2. The conditional matrices for Peirce's formula.

in the boolean algebra $\{0, 1\}$ and a 1 in the cell (i, j) means that there is an arrow from the node i to the node j.

4.1 Conditional matrices

To represent conditional bi-colored graphs, we use *conditional matrices*: the cells of these matrices take their values from the set of boolean functions. These functions are represented by boolean expressions built from the boolean selectors introduced during the conditional graph construction.

Definition 4.1 A *conditional matrix* on set $\mathcal{S}$ of size k is a $k \times k$-array with values in the free boolean algebra over the set of selectors.

There are two incidence relations for a bi-colored graph corresponding to the green $(\to)$ and red $(\Rightarrow)$ arrows. So a conditional bi-colored graph is represented by a pair of conditional matrices. We use the same denotation for the (conditional) incidence relation and for its corresponding matrix. So $\mathcal{G}_D$ is represented by a pair $(\to, \Rightarrow)$ of conditional matrices. Figure 2 presents the two matrices corresponding to the graph $\mathcal{G}_D$ when D is the Peirce formula of section 3.2. We only write the cells whose values are different from 0: the matrices are *sparse* because the number of non-zero cells is linear whereas the total number of cell is quadratic.

4.2 R-cycle removal as a trace computation

The boolean operator of conjunction (or multiplication) $\cdot$ and disjunction (or sum) $+$ extend naturally to conditional matrices. So we may consider the sum $\to + \Rightarrow$, product $\to \cdot \Rightarrow$ of conditional matrices and the reflexive and transitive closure $\to^\star = \sum_{i \geqslant 0} \to^i$. We also introduce the trace of a matrix: $\mathrm{tr}(M) = \sum_x M_{x,x}$. When boolean selectors are instantiated inside a conditional matrix, we get a matrix with values in $\{0, 1\}$ which is the incidence matrix of the corresponding instance graph. Moreover, instantiation commutes with algebraic operations on matrices. This leads to the following result:

Theorem 4.2 *Let $\mathcal{G} = (\rightarrow, \Rightarrow)$ be a conditional bi-colored graph represented by a pair of conditional matrices. There exists a r-cycle in every instance $\mathcal{G}_v$ of $\mathcal{G}$ if and only if $tr\big((\rightarrow + \Rightarrow)^\star \Rightarrow\big) = 1$ holds.*

Moreover, when the boolean function $\mathrm{tr}\big((\rightarrow + \Rightarrow)^\star \Rightarrow\big)$ is not a tautology, there exists a valuation v on selectors in $\{0, 1\}$ such that this trace has value 0: $\mathrm{tr}\big((\rightarrow_v + \Rightarrow_v)^\star \Rightarrow_v\big) = 0$. Then, the corresponding instance graph $\mathcal{G}_v$ has no r-cycle.

Now, the problem is to compute this trace efficiently. Let us fix a size $k > 0$ of matrices. Let I denote the identity $k \times k$ matrix ($I_{x,x} = 1$ and $I_{x,y} = 0$ otherwise.) Let M be any conditional $k \times k$ matrix. Then $M^\star = (I + M)^k$ (because any path of size $k+1$ contains a sub-path of size k.) Moreover, as $I \leqslant I + M$ (cell-wise), $I, I + M, (I + M)^2 \ldots$ is a (point-wise) increasing sequence of conditional matrices which stabilizes in at most k steps.

In order to evaluate $(\rightarrow + \Rightarrow)^\star \Rightarrow$, we compute $\alpha = I + \rightarrow + \Rightarrow$ and $\beta = \Rightarrow$ and then the increasing sequence $\beta, \alpha\beta, \alpha^2\beta, \alpha^3\beta, \ldots$ until it stabilizes to $\alpha^\star\beta$. This can be done column by column on β. Let β_i denote the column i of β then $\alpha^\star\beta_i$ is the column i of $\alpha^\star\beta$. The computation of the column 1 for Peirce's formula is the following:

α	0	1	2	3	4	5	6	A	B	$\diamondsuit$	β_1	$\alpha\beta_1$	$\alpha^2\beta_1$	$\alpha^3\beta_1$	$\alpha^4\beta_1$	$\alpha^\star\beta_1$
0	1									1						$x.\overline{y}.\overline{z}$
1		1			y									$x.y$	$x.y$	$x.y$
2	x	x	1							x	x	x	x	x	x	x
3				1	$\overline{y}$									$x.\overline{y}$	$x.\overline{y}$	$x.\overline{y}$
4					1			1					x	x	x	x
5						1		1					x	x	x	x
6				z		z	1			z				$x.z$	$x.z$	$x.z$
A			1					1				x	x	x	x	x
B							1		1						$x.z$	$x.z$
$\diamondsuit$	$\overline{x}$			$\overline{z}$						1					$x.\overline{y}.\overline{z}$	$x.\overline{y}.\overline{z}$

Most of the columns of β contain only 0 in which case there is no need for computation: the fixpoint is this zero column. In the Peirce example, only column 1, 5 and $\diamondsuit$ contain values which are different from zero.

When evaluating the trace, it is possible to share computation between the columns of β. Let T be the column matrix composed of 1 on each cells. We consider the following sequence: $t_0 = 0$ and $t_i = \big[\alpha^\star(t_{i-1}T + \beta_i)\big]_i$ for $i = 1, \ldots, k$. Then $t_k = \mathrm{tr}(\alpha^\star\beta)$. For example, in the case of Peirce's formula, we get $t_0 = 0$ and then $t_1 = x.y$. Columns β_2, β_3 and β_4 are empty (i.e. contain only 0) so $t_2 = t_3 = t_4 = t_1 = x.y$. Then $t_5 = \big[\alpha^\star(x.y.T + \beta_5)\big]_5$ and we obtain

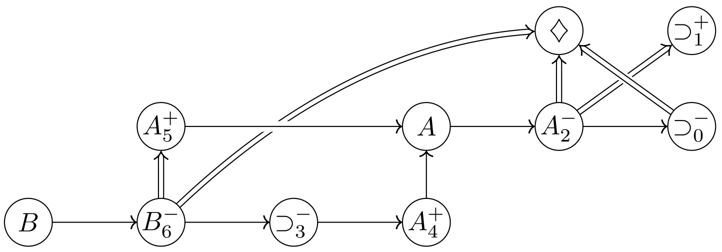
$t_5 = x.y$. Then columns β_6, β_A and β_B are empty and $t_6 = t_A = t_B = t_5 = x.y$. Finally we compute $t_\diamondsuit = \big[\alpha^\star(x.y.T + \beta_\diamondsuit)\big]_\diamondsuit$. Let $\gamma = x.y.T + \beta_\diamondsuit$:

α	0	1	2	3	4	5	6	A	B	$\diamondsuit$	γ	$\alpha\gamma$	$\alpha^2\gamma$	$\alpha^3\gamma$	$\alpha^\star\gamma$
0	1									1	1	1	1	1	1
1		1			y						$x.y$	$x.y$	$x.y$	$x.y$	$x.y$
2	x	x	1							x	x	x	x	x	x
3				1	$\overline{y}$						$x.y$	$x.y$	$x.y$	x	x
4					1			1			$x.y$	$x.y$	x	x	x
5						1		1			$x.y$	$x.y$	x	x	x
6				z		z	1			z	$x.y+z$	$x.y+z$	$x.y+z$	$x.y+z$	$x.y+z$
A			1					1			$x.y$	x	x	x	x
B							1		1		$x.y$	$x.y+z$	$x.y+z$	$x.y+z$	$x.y+z$
$\diamondsuit$	$\overline{x}$			$\overline{z}$						1	$x.y$	$\overline{x}+x.y$	$\overline{x}+x.y$	$\overline{x}+x.y$	$\overline{x}+x.y+x.\overline{z}$

and we obtain $t_\diamondsuit = \overline{x} + x.y + x.\overline{z}$. This the trace of $\alpha^\star\beta = (\rightarrow + \Rightarrow)^\star\Rightarrow$ and it is not a tautology. The only valuation that falsifies this trace is $v_x = 1, v_y = 0, v_z = 1$. This is of course the same valuation we obtained by hand (by looking up for r-cycles) in section 3.3.

5 Counter-model extraction

Now we explain how to extract a counter-model from the corresponding instance bi-colored graph $\mathcal{G}_v$. The reader can easily check that it can be represented by:



$$\begin{tikzcd}
 & & & & \diamondsuit & \supset_1^+ \\
 & A_5^+ \arrow[rr] & & A \arrow[r] & A_2^- \arrow[u, Rightarrow] \arrow[ur, Rightarrow] \arrow[r] & \supset_0^- \arrow[ul, Rightarrow] \\
B \arrow[r] & B_6^- \arrow[u, Rightarrow] \arrow[uurrr, Rightarrow, bend left] \arrow[r] & \supset_3^- \arrow[r] & A_4^+ \arrow[u] & &
\end{tikzcd}$$

In this graph, red arrows are always strictly climbing up and green arrows never go down so no r-cycle could exist. The counter-model is very easy to compute: give the variable A and B their height in this graph. So $[\![A]\!] = 1$ and $[\![B]\!] = 0$ is a counter-model to the Peirce's formula which can be checked by $[\![((A \supset B) \supset A) \supset A]\!] = ((1 \twoheadrightarrow 0) \twoheadrightarrow 1) \twoheadrightarrow 1 = (0 \twoheadrightarrow 1) \twoheadrightarrow 1 = \infty \twoheadrightarrow 1 = 1 < \infty$

Now we explain how to extract a counter-model out of an instance graph lacking r-cycles in the general case. We give a characterization of the lack of r-cycles based on the notion of *bi-height*:

Definition 5.1 Let $\mathcal{G}$ be a bi-colored graph. A *bi-height* is a function $h : \mathcal{G} \to \mathbb{N}$ such that for any $x, y \in \mathcal{G}$, if $x \rightarrow y \in \mathcal{G}$ then $h(x) \leqslant h(y)$ and if $x \Rightarrow y \in \mathcal{G}$ then $h(x) < h(y)$.

It is clear that the preceding graph has a bi-height given by $h(B) = h(B_6^-) = h(\supset_3^-) = h(A_4^+) = 0$, $h(A_5^+) = h(A) = h(A_2^-) = h(\supset_0^+) = 1$ and $h(\diamondsuit) = h(\supset_1^+) = 2$. In [12], you will find a constructive proof of the following result which states the existence of a bi-height whenever no r-cycle exist:

Theorem 5.2 *Let D be a formula of* LC*, $\mathcal{G}$ the corresponding conditional bi-colored graph and v a valuation such that the instance graph $\mathcal{G}_v$ does not contains any r-cycle. Then it is possible to compute a bi-height h for $\mathcal{G}_v$ in linear time.*[8] *Moreover, if we define $[\![\cdot]\!] : \mathsf{Var} \to \overline{\mathbb{N}}$ by $[\![V]\!] = h(V)$ for V variable of D then $[\![\cdot]\!]$ is a counter-model of D, i.e. $[\![D]\!] < \infty$.*

6 Implementation remarks and conclusion

The procedure described throughout this paper has been implemented completely in the Objective Caml language and is accessible at

http://www.loria.fr/~larchey/LC

The reader interested in the proofs of the results presented here can also find them there.

For the prototype implementation, we have chosen to represent conditional matrices by *sparse arrays*. The boolean functions which compose them are represented by the nodes of a shared BDD [5] for efficient boolean computations and extraction of boolean counter-models. The algorithm for the computation of bi-heights is a slightly modified version of a depth first search procedure.

In further work, we will deeper investigate the relationships between the notion of r-cycle and the G-cycles of [3] and analyze if our conditional graphs also fit in the hyper-sequent setting. We will also investigate the relationships between our parallel counter-model search and other approaches based for example on parallel dialogue games [4,9].

References

[1] Alessandro Avellone, Mauro Ferrari, and Pierangelo Miglioli. Duplication-Free Tableau Calculi and Related Cut-Free Sequent Calculi for the Interpolable Propositional Intermediate Logics. *Logic Journal of the IGPL*, 7(4):447–480, 1999.

[2] Arnon Avron. A Tableau System for Gödel-Dummett Logic Based on a Hypersequent Calculus. In *TABLEAUX 2000*, volume 1847 of *LNAI*, pages 98–111, 2000.

[3] Arnon Avron and Beata Konikowska. Decomposition Proof Systems for Gödel-Dummett Logics. *Studia Logica*, 69(2):197–219, 2001.

[8] Linearity is measured with respect to either the size of D or the number of nodes and arrows of $\mathcal{G}_v$.

[4] Matthias Baaz and Christian Fermüller. Analytic Calculi for Projective Logics. In *TABLEAUX'99*, volume 1617 of *LNCS*, pages 36–50, 1999.

[5] Randal E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, 1986.

[6] Michael Dummett. A Propositional Calculus with a Denumerable matrix. *Journal of Symbolic Logic*, 24:96–107, 1959.

[7] Roy Dyckhoff. Contraction-free Sequent Calculi for Intuitionistic Logic. *Journal of Symbolic Logic*, 57(3):795–807, 1992.

[8] Roy Dyckhoff. A Deterministic Terminating Sequent Calculus for Gödel-Dummett logic. *Logical Journal of the IGPL*, 7:319–326, 1999.

[9] Christian Fermüller. Parallel Dialogue Games and Hypersequents for Intermediate Logics. In *TABLEAUX 2003*, volume 2796 of *LNAI*, pages 48–64, 2003.

[10] Kurt Gödel. Zum intuitionistischen Aussagenkalkül. In *Anzeiger Akademie des Wissenschaften Wien*, volume 69, pages 65–66. 1932.

[11] Petr Hajek. *Metamathematics of Fuzzy Logic*. Kluwer Academic Publishers, 1998.

[12] Dominique Larchey-Wendling. Counter-model search in Gödel-Dummett logics. To be published in the proceedings of IJCAR 2004.

[13] Dominique Larchey-Wendling. Combining Proof-Search and Counter-Model Construction for Deciding Gödel-Dummett Logic. In *CADE-18*, volume 2392 of *LNAI*, pages 94–110, 2002.

[14] George Metcalfe, Nicolas Olivetti, and Dov Gabbay. Goal-Directed Calculi for Gödel-Dummett Logics. In *CSL*, volume 2803 of *LNCS*, pages 413–426, 2003.