Gödel-Dummett counter-models through matrix computation

Dominique Larchey

LORIA – CNRS

Nancy, France

Gödel-Dummett logic LC

- Intermediate logic: $\boxed{\mathsf{IL} \subset \mathsf{LC} \subset \cdots \subset \mathsf{LC}_n \subset \cdots \subset \mathsf{LC}_1 = \mathsf{CL}}$
- Syntactic characterization: $\mathsf{LC} = \mathsf{IL} + (X \supset Y) \vee (Y \supset X)$
- Semantic models:
  - Linear Kripke trees or the lattice $\overline{\mathbb{N}} = \mathbb{N} \cup \{\infty\}$
  - For finitary LC_n, $\overline{[0,n)} = [0,\ldots,n[\cup \{\infty\}$
  - Lattice structure: min, max, ...
- Complexity:
  - LC (and CL) are NP-complete
  - IL is PSPACE-complete

Deciding LC with proof-search

- **Proof search and counter-models combined**
  - Strongly invertible rules to reduce sequents
  - Semantic computation to decide irreducible sequents
- **Efficient** (duplication-free, loop-free) proof-search
  - IL (Dyckhoff & Hudelmair, Weich, Larchey & Galmiche)
  - Intermediate logics (Avellone et al. and Fiorino)
  - LC (Dyckhoff, Avron, Larchey)
- **Invertibility** and strong invertibility of logical rules
  - No backtracking in proof-search
  - Counter-model generation

Overview

- Bi-colored semantic graphs:
  - Implicational sequents
  - R-cycles
  - Height and counter-models
- The finitary case LC_n
- The proof system
  - Indexing, polarizing
  - Bi-colored graph construction

Overview (bis)

- Example of the Peirce's formula
- Systematic r-cycle detection:
  - Matrices of boolean functions
  - Provability and the trace operation
  - Sum operation and the minimal counter-model
- Empirical observations
- Conclusion

Deciding/refuting implicational sequents

- Associated **bi-colored semantic graph**

$$A \supset B, B \supset C, D \supset C \vdash C \supset A$$

$$\begin{tikzcd} B \arrow[r] & C \\ A \arrow[u] \arrow[ur, Rightarrow] & D \arrow[u] \end{tikzcd}$$

- **R-cycle**: cycle with at least one red arrow $\quad (x(\rightarrow + \Rightarrow)^{\star} \Rightarrow x)$
- $\boxed{\text{Implicational sequent valid in LC iff there exists one r-cycle}}$
- Otherwise draw the graph **by levels** (linear time) :
  - Red arrows $\Rightarrow$ go up (strictly)
  - Blue arrows $\rightarrow$ never go down
- $\boxed{\text{The counter model is given by the height}}$

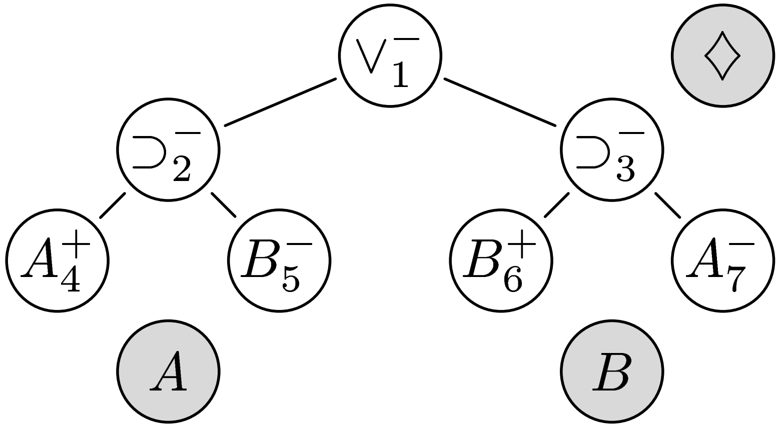
$\rightarrow$ $A = D = 0$ and $B = C = 1$

Refuting implicational sequents (bis)

- Implicational sequent and finitary LC_n, $1 \leq n < \infty$:
  - Necessary and sufficient condition for counter-models:
  - $\rightarrow$ $\boxed{\text{the graph contains no chain of the form } (\rightarrow^\star \Rightarrow)^{n+1}}$
  - Such n exists iff there is no r-cycle
  - Obvious bound: the number of red $\Rightarrow$ arrows
- Draw the graph with **at most $n + 1$ levels** (height $\leq n$)
- Counter-model in LC_n is given by the height
- Details of the supporting theory in IJCAR 2004

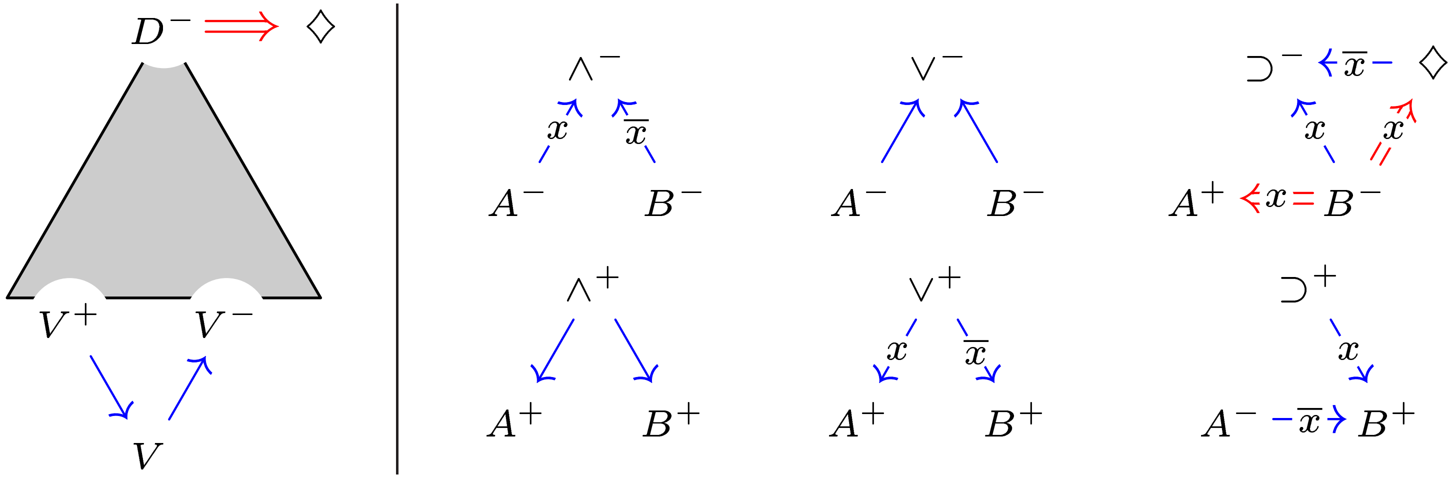
Indexing and polarizing

- Build a graph based on sub-formulae tree
- Each occurrence of a sub-formulae has an **index** and a **polarity**
- Example: $(A \supset B) \vee (B \supset A)$



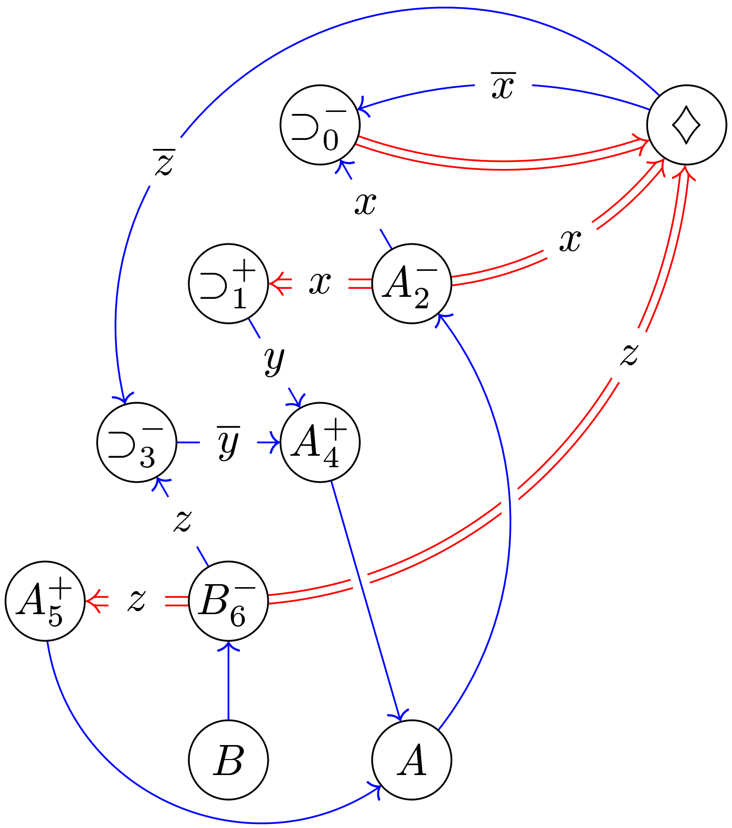
- Add one node $\Diamond$
- Add one further node for each variable V

A new proof system



- Given D^-, compute its graph with boolean selectors (linear)
- Instance graphs obtained by setting selectors ($x = 0$ or 1)
- Result: $\boxed{D \text{ is provable iff every instance graph has a r-cycle}}$
- If an instance has no r-cycle, its height is a counter-model

Example: Peirce's formula



$$((A_5^+ \supset_3^- B_6^-) \supset_1^+ A_4^+) \supset_0^- A_2^-$$

- Build the bi-colored graph
- R-cycles and condition to "break" it

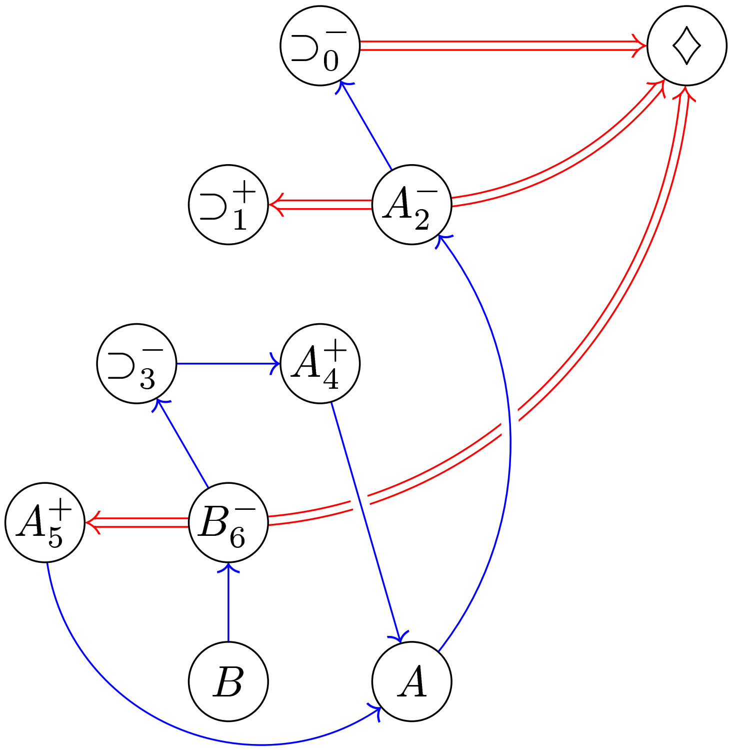
$0 \Rightarrow \diamondsuit \rightarrow 0$	x
$0 \Rightarrow \diamondsuit \rightarrow 3 \rightarrow 4 \rightarrow A \rightarrow 2 \rightarrow 0$	$z + y + \overline{x}$
$2 \Rightarrow \diamondsuit \rightarrow 3 \rightarrow 4 \rightarrow A \rightarrow 2$	$\overline{x} + z + y$
$2 \Rightarrow 1 \rightarrow 4 \rightarrow A \rightarrow 2$	$\overline{x} + \overline{y}$

- Constraint for no r-cycle:

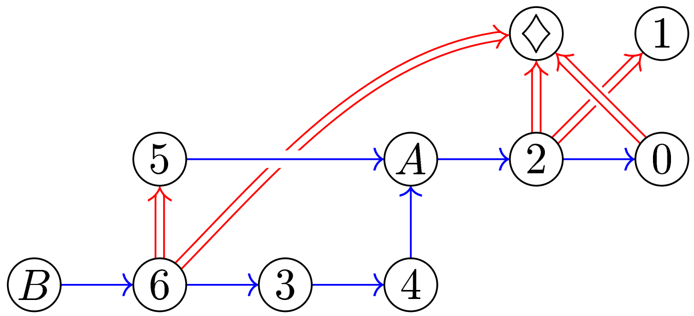
$$x.(z + y + \overline{x}).(\overline{x} + z + y).(\overline{x} + \overline{y})$$

- Solution: $x = z = 1$ and $y = 0$

Example (bis): instance graph



- Intanciate with $x = z = 1$, $y = 0$
- Of course, there is no r-cycle!
- Redraw the graph by levels:



- Counter-model: $[\![A]\!] = 1$, $[\![B]\!] = 0$
- In LC and LC_2 but not in $CL = LC_1$

R-cycle detection and matrix computation

- Bi-colored graph = sparse **matrices of boolean functions**

$\rightarrow$	0	1	2	3	4	5	6	A	B	$\diamondsuit$
0										
1					y					
2	x									
3					$\overline{y}$					
4								1		
5								1		
6				z						
A			1							
B							1			
$\diamondsuit$	$\overline{x}$			$\overline{z}$						

$\Rightarrow$	0	1	2	3	4	5	6	A	B	$\diamondsuit$
0										1
1										
2		x								x
3										
4										
5										
6						z				z
A										
B										
$\diamondsuit$										

- Shared ROBDDs, and matrix operations:
  - $+ = \vee$, $\times = \wedge$, $M^\star = I + M^2 + M^3 + \cdots$
  - $\mathrm{tr}(M) = \sum_x M_{x,x}$, $\sum(M) = \sum_{x,y} M_{x,y}$

R-cycle detection and matrix computation (bis)

- Result: provable in LC iff $\boxed{\mathrm{tr}\big((\rightarrow + \Rightarrow)^{\star}\Rightarrow\big) = 1}$
- If the trace not a tautology:
  - Extract a instance with no r-cycle
  - Draw the instance by levels
  - Counter-model given by height
- Result: invalid in LC_n iff $\boxed{\sum\big(\rightarrow^{\star}\Rightarrow\big)^{n+1} < 1}$
- Compute this sequence:
  - n bounded by number of $\supset^{-}$+1
  - Search the first non-tautology
  - Obtain the minimal counter-model

Matrix computation (continued)

- $\alpha = I + \rightarrow + \Rightarrow$, $\beta = (\beta_0, \beta_1, \ldots, \beta_\diamondsuit) = \Rightarrow$ by columns
- $\beta, \alpha\beta, \alpha^2\beta, \alpha^3\beta, \ldots$ until it stabilizes to $\alpha^\star\beta = (\rightarrow + \Rightarrow)^\star \Rightarrow$
- Compute column by column: $\alpha^\star\beta_0$, $\alpha^\star\beta_1$, ...
- Peirce formula (remark, $\beta_0 = (0)$ so $\alpha^\star\beta_0 = (0)$)

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					$x.\overline{y}.\overline{z}$	$x.\overline{y}.\overline{z}$

Optimisations for the trace computation

- $t_0 = 0$, $t_i = \left[\alpha^\star(t_{i-1}(1) + \beta_i)\right]_i$ and the limit $t_n = \mathrm{tr}(\alpha^\star\beta)$
  - $t_0 = 0, t_1 = x.y = t_2 = \cdots = t_4$, $t_5 = x.y = t_6 = t_A = t_B$
  - $t_\diamondsuit = \left[\alpha^\star(t_B(1) + \beta_\diamondsuit)\right]_\diamondsuit$. Let $\gamma = x.y.(1) + \beta_\diamondsuit$

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	$x.y$	$\overline{x}+x.y$	$\overline{x}+x.y$	$\overline{x}+x.y$	$\overline{x}+x.y+x.\overline{z}$

- Result: $t_\diamondsuit = \overline{x} + x.y + x.\overline{z}$, refuted by $x = z = 1, y = 0$

Peirce's formula in pratice

```
Shell No. 3 - Konsole <2>
# compute_t ~log:2 "((A->B)->A)->A";;
This graph has 11 nodes
---------------------------------
 0 : (=>,-,1,6) : 0=>10,
 1 : (=>,+,2,5) : 1-(1+)>5,
 2 : (=>,-,3,4) : 2-(1-)>5,
 3 : {A,+} : 3->7,
 4 : {B,-} : 4-(2+)>2, 4=(2+)>3, 4=(2+)>10,
 5 : {A,+} : 5->7,
 6 : {A,-} : 6-(0+)>0, 6=(0+)>1, 6=(0+)>10,
 7 : [A] : 7->6, 7->10, 7->10,
 8 : [B] : 8->4,
 9 : [.] : 9->7, 9->8,
10 : [%] : 10-(0-)>0, 10-(2-)>2,
---------------------------------
Skipping empty column 0
Computing column 1 (trace=0)
**********************************************************
   0    0    0    0    0    0    2    0    0    0    0
   0    0    0    0    0    0    2    2    0    0    0 [1]
   0    0    0    2    0    2    2    2    0    2    0 [3]
   0    8    9    2   10    2    2    2    0    2    0 [3]
   0    8    9    2   10    2    2    2   10    2   14 [2]
  14    8    9    2   10    2    2    2   10    2   14 [1]
  14    8    9    2   10    2    2    2   10    2   14 [0]
**********************************************************
Skipping empty column 2
Computing column 3 (trace=8)
**********************************************************
   8    8    8    8   10    8    8    8    8    8    8
   8    8    8    8   10    8    8    8   10    8    8 [1]
   8    8    8    8   10    8    8    8   10   10    8 [1]
   8    8    8    8   10    8    8    8   10   10    8 [0]
**********************************************************
Skipping empty column 4
Skipping empty column 5
Skipping empty column 6
Skipping empty column 7
Skipping empty column 8
Skipping empty column 9
Computing column 10 (trace=8)
**********************************************************
   1    8    8    8   10    8    2    8    8    8    8
   1    8    8    8   10    8    2    2   10    8   13 [3]
   1    8    8    2   10    2    2    1   10   17   13 [4]
   1    8    2    1   10    1    2    1   10    1   13 [4]
   1    5   20    1   10    1    2    1   10    1   23 [3]
   1    5   20    1   10    1    2    1   10    1   23 [0]
**********************************************************
Result trace=23
- : (string * int) list option =
Some [("A", 1); ("B", 0); ("ff", 0); ("tt", 2)]
#
```

```
Shell No. 3 - Konsole <2>
   1    8    2    1   10    1    2    1   10    1   13 [4]
   1    5   20    1   10    1    2    1   10    1   23 [3]
   1    5   20    1   10    1    2    1   10    1   23 [0]
**********************************************************
Result trace=23
- : (string * int) list option =
Some [("A", 1); ("B", 0); ("ff", 0); ("tt", 2)]
# compute_s ~log:2 "((A->B)->A)->A";;
This graph has 11 nodes
---------------------------------
 0 : (=>,-,1,6) : 0=>10,
 1 : (=>,+,2,5) : 1-(1+)>5,
 2 : (=>,-,3,4) : 2-(1-)>5,
 3 : {A,+} : 3->7,
 4 : {B,-} : 4-(2+)>2, 4=(2+)>3, 4=(2+)>10,
 5 : {A,+} : 5->7,
 6 : {A,-} : 6-(0+)>0, 6=(0+)>1, 6=(0+)>10,
 7 : [A] : 7->6, 7->10, 7->10,
 8 : [B] : 8->4,
 9 : [.] : 9->7, 9->8,
10 : [%] : 10-(0-)>0, 10-(2-)>2,
---------------------------------
**********************************************************
   1    0    0    0    6    0    2    0    0    0    0
   1    0    0    0    6    0    2    2    6    0    3 [3]
   1    0    0    2    6    2    2    1    6    8    3 [4]
   1    9   10    1    6    1    2    1    6    1    3 [5]
   1    5    4    1    6    1    2    1    6    1   15 [3]
   1    5    4    1    6    1    2    1    6    1   15 [0]
**********************************************************
Result sum=1
**********************************************************
   9    0    0    0    6    0   14    0    0    0    0
   9    0    0    0    6    0   14   14    6    0    3 [3]
   9    0    0   14    6   14   14   15    6   16    3 [4]
   9   11   12   15    6   15   14   15    6    1    3 [5]
   9    5   17   15    6   15   14   15    6    1    9 [3]
   9    5   17   15    6   15   14   15    6    1    9 [0]
**********************************************************
Result sum=1
**********************************************************
   9    0    0    0   15    0   13    0    0    0    0
   9    0    0    0   15    0   13   13   15    0    3 [3]
   9    0    0   13   15   13   13   12   15   19    3 [4]
   9   17   18   12   15   12   13   12   15   12    3 [5]
   9    5   10   12   15   12   13   12   15   12    9 [3]
   9    5   10   12   15   12   13   12   15   12    9 [0]
**********************************************************
Result sum=12
Sum is <top after 3 iterations
- : (string * int) list option =
Some [("A", 1); ("B", 0); ("ff", 0); ("tt", 2)]
#
```

Empirical observations

- Concise implementation (`wc *.ml --> 2017 lines`)
- Fast with chains $(X_1 \supset X_2) \vee (X_2 \supset X_3) \vee (X_3 \supset X_4)$ or cycles
- $(X_0 \wedge (X_0 \supset (X_1 \vee X_1)) \wedge \cdots \wedge (X_{n-1} \supset (X_n \vee X_n))) \supset X_n$
  - Untrackable for proof-search based methods
  - Easily solved for $n = 100$ (early constraint solving)
- No complete benchmark tests for the moment
  - Compare with proof-search based provers ?
  - Compare with classical provers ?

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

Conclusion

- A new calculus to decide/build counter-models for LC and LC_n
- Proof-search space in a shared graph theoretic structure
  - Decision amounts to (parallel) r-cycle detection
  - Extraction of counter-models
- Systematic r-cycle detection with sparse matrices and BDDs
  - Computing the trace and sum efficiently
  - Minimal counter-model extraction
  - Empirical remarks