PROOF COMPLEXITY AND INTERPOLATION

PROPOSITIONAL PROOF COMPLEXITY

Study of certificates of unsatisfiability for CNF formulas (and more, but this definition is good enough for today

PROOF SYSTEM for language L
Deterministic algorithm $P(x, \pi)$
Polynomial-time in $|x| + |\pi|$

Completeness If $x \in L$, then $\exists \pi$ s.t. $P(x,\pi) = 1$

Soundness If $x \notin L$, then $\forall \pi \; P(x,\pi) = 0$

A proof system P is **POLYNOMIALLY BOUNDED** if $\exists$ polynomial p s.t. $\forall x \in L \; \exists$ proof π s.t. $|\pi| \leq p(|x|)$ and $P(x,\pi) = 1$

A **PROPOSITIONAL PROOF SYSTEM** is a proof system for $UNSAT = \{$ Unsatisfiable CNF formulas $\}$

Believe that no polynomially bounded propositional proof system exist, because this would imply $NP = coNP$. Converse is also true: If $NP = coNP$, then $\exists$ polynomially bounded proof system for UNSAT (namely, standard NP verifier)

THEOREM 1.3 [Cook-Reckhow '79]
If there are no polynomially bounded propositional proof systems, then $P \neq NP$

Proof. UNSAT is in coNP. If no polynomially bounded proof system, then UNSAT $\notin$ NP (since NP is exactly family of languages with polynomially bounded proof systems), and so coNP $\neq$ NP. But then NP $\neq$ P, because P is closed under complement

"COOK'S PROGRAM"
Prove superpolynomial lower bounds for stronger and stronger proof systems until one day we can get general lower bound for all proof systems.

Has not worked out so well

But many other reasons to study proof complexity, e.g.:

- Compare strengths of different types of mathematical reasoning (formalized in different proof systems)
- Understand power and limitations of different algorithmic paradigms (formalized as different proof systems)

Today we will study the RESOLUTION proof system – arguably the most investigated proof system in all of proof complexity

Introduced in 1937 (?) by Blake
Started to get attention in 1960s in the context of Boolean satisfiability algorithms, so-called SAT solvers

[Davis – Putnam 1960]
[Davis – Logemann – Loveland 1962]
[Robinson 1965]

Resolution is still the basis of state-of-the-art solvers using CONFLICT-DRIVEN CLAUSE LEARNING invented by Marques-Silva & Sakallah 1996

PLAN FOR TODAY (AND NEXT LECTURE)

- Define resolution proof system
- Show how short resolution proofs can be turned into small circuits using INTERPOLATION
- Use the lower bound for monotone circuits computing CLIQUE$_{k,n}$ that we just proved in the last couple of lectures to get strong lower bounds for a related family of CNF formulas

RESOLUTION

- Start with clauses in formula F
- Derive new clauses that are semantically implied
- End by deriving contradiction
- Then original formula F must have been contradictory, i.e., unsatisfiable

DEF Resolution refutation π of unsatisfiable CNF formula F, denoted $\pi : F \vdash \bot$, is a sequence of clauses

$$\pi = (D_1, D_2, D_3, \ldots, D_{L-1}, D_L)$$

such that $D_L = \bot$ is the empty clause not containing any literals and each D_i is

(a) an AXIOM CLAUSE $D_i \in F$, or

(b) a clause on the form $D_i = B \vee C$ derived from clauses $D_j = B \vee x$ and $D_k = C \vee \bar{x}$ for $j, k < i$ using the RESOLUTION RULE

$$\frac{B \vee x \qquad C \vee \bar{x}}{B \vee C}$$

($B \vee x$ and $C \vee \bar{x}$ are RESOLVED OVER x and $B \vee C$ is the RESOLVENT)

View clauses as sets of literals: no repetitions, and order does not matter

Without loss of generality, all clauses are nontrivial — never $x \in C$ and $\bar{x} \in C$
(Not hard to show that trivial clauses can be removed from any resolution refutation)

LEMMA 2.2 Resolution is sound and complete, i.e., there exists a resolution refutation of F if and only if F is unsatisfiable.

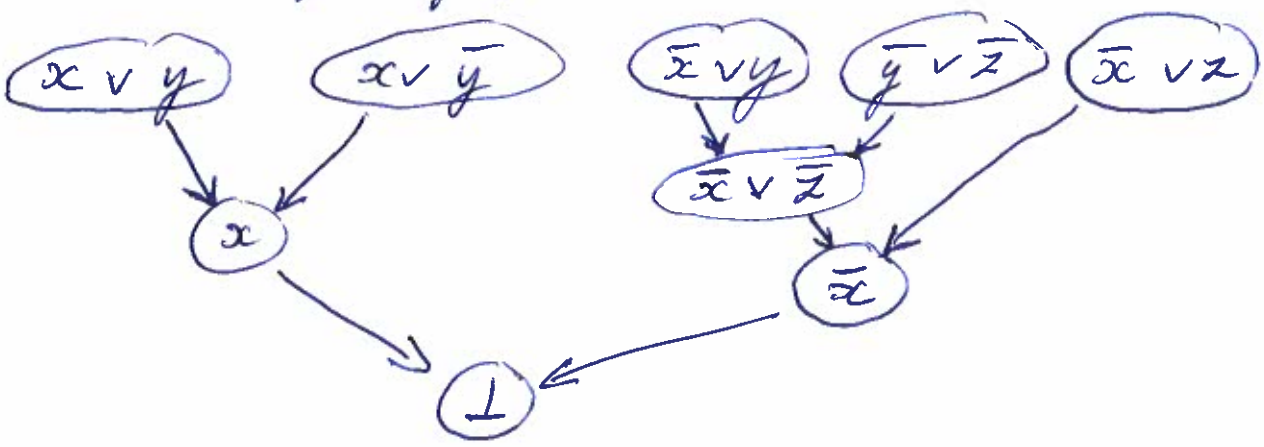
Proof sketch ($\Rightarrow$) Suppose α satisfies all clauses in F. Then α satisfies all resolvents by induction. But no assignment can satisfy the empty clause without literals.
($\Leftarrow$) Not hard, but requires an argument. Left as an exercise.

Example $F = (x \vee y) \wedge (x \vee \bar{y}) \wedge (\bar{x} \vee y) \wedge (\bar{x} \vee z) \wedge (\bar{y} \vee \bar{z})$

Present refutation as annotated list

1	$x \vee y$	axiom
2	$x \vee \bar{y}$	axiom
3	$\bar{x} \vee y$	axiom
4	$\bar{x} \vee z$	axiom
5	$\bar{y} \vee \bar{z}$	axiom
6	x	Res (1, 2)
7	$x \vee \bar{z}$	Res (3, 5)
8	$\bar{x}$	Res (4, 7)
9	$\perp$	Res (6, 8)

Can also represent resolution refutation π as directed acyclic graph (DAG) G_π



That is:
- One node per clause
- Axioms turn into source nodes
- Edges from resolved clauses to resolvents

LENGTH of resolution refutation π =
= # clauses in it $\quad L(\pi)$ [= 9 in our example]

LENGTH OF REFUTING F

$$L_R(F \vdash \bot) = \min_{\pi : F \vdash \bot} \{ L(\pi) \}$$

Often also called the size of a resolution refutation. Size measure should arguably count also the number of literals in each clause, but this is at most a linear factor difference, and we mainly care about difference between polynomial and superpolynomial

INTERPOLATION AND CLIQUE-COLOURING FORMULAS

PC / VII

Interpolation method introduced by Krajíček 1994

Used by Pudlák in 1997 to prove lower bounds on proof length for clique-colouring formulas in the CUTTING PLANES proof system (much stronger proof system).
To make our lives a little bit easier, we will do a version of Pudlák's result for resolution, which is a weaker proof system

Clique-colouring formulas
Parameters
- n : # vertices in graph
- m : size of clique

Formula says:
"There exists a graph on n vertices which has an m-clique and is also $(m-1)$-colourable"

Obviously absurd, so formula unsatisfiable
But resolution has a hard time understanding this...

Variables

$\vec{p} = \{p_{i,j} \mid 1 \le i < j \le n\}$

$p_{i,j}$ = "there is an edge between vertices i & j"

$\vec{q} = \{q_{k,i} \mid k \in [m],\ i \in [n]\}$

$q_{k,i}$ = "vertex i is kth member of clique"

$\vec{r} = \{r_{i,\ell} \mid i \in [n],\ \ell \in [m-1]\}$

$r_{i,\ell}$ = "vertex i gets colour ℓ"

Axiom clauses

$\bigvee_{i \in [n]} q_{k,i}$ "some vertex is kth clique member"

$\bar{q}_{k,i} \vee \bar{q}_{k',i}$ "clique vertices have unique member numbers" $(k \neq k')$

$p_{i,j} \vee \bar{q}_{k,i} \vee \bar{q}_{k',j}$ "clique members are connected by edges" $(i<j,\ k \neq k')$

$\bigvee_{\ell \in [m-1]} r_{i,\ell}$ "every vertex gets a colour"

$\bar{p}_{i,j} \vee \bar{r}_{i,\ell} \vee \bar{r}_{j,\ell}$ "neighbours have distinct colours"

Clique-colouring formula can be written as

$$A(\vec{p},\vec{q}) \wedge B(\vec{p},\vec{r}) \qquad (*)$$

where the variable sets $\vec{p},\vec{q},\vec{r}$ are disjoint.

Given partial assignment, or restriction, ρ to variables $Vars(F)$ of CNF formula F, we write $F\restriction_\rho$ for the restricted formula where variables $x \in Dom(\rho)$ are replaced by values $\rho(x)$ and formula is simplified by removing

- satisfied clauses
- falsified literals

Ex

$F = (x \vee y) \wedge (x \vee \bar{y}) \wedge (\bar{x} \vee y) \wedge (\bar{x} \vee z) \wedge (\bar{y} \vee \bar{z})$

$\rho = \{y \mapsto 1\}$

$F\restriction_\rho = x \wedge (\bar{x} \vee z) \wedge \bar{z}$

Suppose we have unsatisfiable formula $F = A(\vec{p},\vec{q}) \wedge B(\vec{p},\vec{r})$ as in $(*)$ and restriction ρ assigning all of $\vec{p}$

Then $F|_\rho = A(\vec{p},\vec{q})|_\rho \wedge B(\vec{p},\vec{r})|_\rho$ splits into two formulas on disjoint sets of variables.

PCI ☒

Write

$$A(\vec{p},\vec{q})|_\rho = A(\rho,\vec{q})$$

$$B(\vec{p},\vec{r})|_\rho = B(\rho,\vec{r})$$

- for notational convenience
- For any such ρ, either $A(\rho,\vec{q})$
- or $B(\rho,\vec{r})$ (or both) must be unsatisfiable.

A Boolean circuit $I(\vec{p})$ is an INTERPOLANT for $A(\vec{p},\vec{q}) \wedge B(\vec{p},\vec{r})$ if

$I(\rho) = 0 \Rightarrow A(\rho,\vec{q})$ unsatisfiable

- $I(\rho) = 1 \Rightarrow B(\rho,\vec{r})$ unsatisfiable

- Such interpolants always exist (and are not necessarily unique)

We are going to prove

If $A(\vec{p},\vec{q}) \wedge B(\vec{p},\vec{r})$ has a short resolution refutation, then it also has a small interpolant.

But suppose interpolant computes function for which we have a circuit lower bound — then this shows that there cannot exist any short resolution refutations

PROOF STRATEGY

1. Start with formula $F = A(\vec{p},\vec{q}) \wedge B(\vec{p},\vec{r})$
2. Assume towards contradiction that F has short resolution refutation
3. Deduce that there exist small interpolants
4. But argue now that interpolant computes something for which we have circuit lower bound
5. Contradiction! Hence refutation cannot be short

- Proof systems for which this proof strategy works are said to have FEASIBLE INTERPOLATION. Resolution (and cutting planes) has feasible interpolation.

Use this to show that clique-colouring formulas are hard for resolution

PROBLEM: We don't have good general circuit lower bounds!

SOLUTION: There is a monotone version of interpolation, and we are lucky enough that this will work

We have proven:

THEOREM 5.1 [Razborov '85, Alon-Boppana '87]
Let undirected graph G on n vertices be represented by $\binom{n}{2}$ edge indicator bits. Then for $m = \sqrt{n}$ there is no monotone circuit of size $2^{o(\sqrt{m})}$ that can distinguish between

- G has an m-clique
- G is $(m-1)$-colourable

But an interpolant for clique-colouring formula distinguishes precisely these two cases!

Define ternary SELECTOR function by

$$sel(x,y,z) = \begin{cases} y & \text{if } x=0 \\ z & \text{if } x=1 \end{cases}$$

This function is <u>not</u> monotone (why?)

We will build interpolating circuit using "gates" $\{\wedge, \vee, sel\}$.

Later, we will argue that we can replace sel by just $\wedge$- and $\vee$-gates to get monotone interpolating circuit.

Next lecture, we will prove following theorem

THEOREM 5.3 [Pudlák '97]
Suppose $A(\vec{p},\vec{q}) \wedge B(\vec{p},\vec{r})$ is unsatisfiable CNF formula over disjoint sets of variables $\vec{p},\vec{q},\vec{r}$ and that $\exists$ resolution refutation $\pi: A \wedge B \vdash \bot$ in length L.
Then the following holds:

(1) There is an interpolating circuit $I(\vec{p})$ over $\{\wedge, \vee, sel\}$ of size $O(L)$.

(2) From π we can construct resolution refutations
(a) $\pi_A : A(p,\vec{q}) \vdash \bot$ if $I(p) = 0$
(b) $\pi_B : B(p,\vec{r}) \vdash \bot$ if $I(p) = 1$
in both cases of length $\leq L$

(3) If $\vec{p}$ variables occur only positively in $A(\vec{p},\vec{q})$ or only negatively in $B(\vec{p},\vec{r})$, then sel-gates can be replaced by $\wedge$- and $\vee$-gates, yielding a MONOTONE circuit of size $O(L)$

If we can prove this theorem, then our proof strategy above yields an exponential lower bound for resolution refutations of clique-colouring formulas

DD2445 COMPLEXITY THEORY: LECTURE 22

RECAP OF LAST LECTURE

Proof complexity: How to prove that CNF formulas are unsatisfiable

Resolution refutation of F

Sequence $\pi = (C_1, C_2, \ldots, C_L)$ such that C_L is empty clause $\perp$ and for every $C_i \in \pi$ it holds that

a) $C_i \in F$ (axiom), or

b) C_i derived from C_j, C_k, $j,k < i$ by RESOLUTION RULE

$$\frac{B \vee x \qquad C \vee \bar{x}}{C \vee D}$$

Length/size of refutation = # clauses L

Our goal is to prove following theorem

THEOREM 1 (informal). Clique-colouring formulas expressing that there exist n-vertex graphs that are $(n-1)$-colourable but contain n-cliques are exponentially hard to refute for resolution.

Follows from:

1. Monotone circuit lower bound for clique
2. INTERPOLATION technique

MI II

THEOREM 2 [Pudlák '97]

Suppose $A(\vec{p}, \vec{q}) \wedge B(\vec{p}, \vec{r})$ is an unsatisfiable CNF formula over disjoint sets of variables $\vec{p}, \vec{q}, \vec{r}$.

Let $\pi : A \wedge B \vdash \bot$ be a resolution refutation in length L. Then the following holds:

(1) There is an INTERPOLATING CIRCUIT $I(p)$ of size $O(L)$ such that
   (a) $I(p) = 0 \Rightarrow A(p, \vec{q})$ unsatisfiable
   (b) $I(p) = 1 \Rightarrow B(p, \vec{r})$ unsatisfiable

(2) From π one can construct resolution refutation
   (a) $\pi_A : A(p, \vec{q}) \vdash \bot$ if $I(p) = 0$
   (b) $\pi_B : B(p, \vec{r}) \vdash \bot$ if $I(p) = 1$
   in both cases of length $\leq L$

(3) If $\vec{p}$-variables occur only positively in $A(\vec{p}, \vec{q})$ or only negatively in $B(\vec{q}, \vec{r})$, then the circuit $I(p)$ can be made monotone

---

The clique-colouring formula lower bound in Thm 1 follows immediately from Thm 2, as argued last time, so today we focus on Thm 2

Build circuits with $\wedge, \vee$, and sel-gates for simplicity.

$$\text{sel}(x,y,z) = \begin{cases} y & \text{if } x = 0 \\ z & \text{if } x = 1 \end{cases}$$

Proof plan

First do part ②.

Then use ② to get ①

Maybe skip details on ③ (but they are in the LaTeX'ed lecture notes)

Key definitions (for proof):

g-clause: Clause C over variables $\vec{g}$ derivable from $A(g, \vec{g})$

r-clause: Clause C over variables $\vec{r}$ derivable from $B(g, \vec{r})$

Trivially true clause $\underline{1}$ considered to be both g-clause and r-clause.

Inductive proof of part ② From $\pi = (C_1, C_2, \ldots, C_L)$ construct sequence of clauses $\tilde{\pi} = (\tilde{C}_1, \tilde{C}_2, \ldots, \tilde{C}_L)$ which will contain candidate derivations π_A and π_B

Inductive hypothesis

P.1. $\tilde{C}_i$ is a g-clause or an r-clause

Write type $(\tilde{C}_i) = g / r$

P2. $\tilde{C}_i = 1$ only if $C_i \wedge \rho = 1$

If $\tilde{C}_i \neq 1$, then $\tilde{C}_i \subseteq C_i$

P3. If $\tilde{C}_i = 1$ then there is an associated axiom clause $\mathcal{E}_i$ such that

- $\mathcal{E}_i$ satisfied by $\rho(a) = 1$ for $a \in C_i \wedge \mathcal{E}_i$
- $\mathcal{E}_i \in A(\vec{p}, \vec{q})$ if type $(\tilde{C}_i) = q$
- $\mathcal{E}_i \in B(\vec{p}, \vec{r})$ if type $(\tilde{C}_i) = r$

Think of $\mathcal{E}_i$ as justification or excuse why we choose $\tilde{C}_i = 1$

This book-keeping is important for the monotonicity in part 3 (but we won't have time to discuss this in detail, so will not pay too much attention to $\mathcal{E}_i$)

Base case $C_i \in A(\vec{p}, \vec{q}) \wedge B(\vec{p}, \vec{r})$

Set $\tilde{C}_i = C_i \wedge \rho$

$\mathcal{E}_i = C_i$ if justification needed

type $(\tilde{C}_i) = q$ if $C_i \in A$, $= r$ if $C_i \in B$.

Inductive step

$C_i = C \vee D$ derived by resolution rule

$$\frac{C \vee x \qquad D \vee \bar{x}}{C \vee D}$$

from $C_j = C \vee x$

$C_k = D \vee \bar{x}$ $\quad j, k < i$

Have already constructed $\tilde{C}_j$ and $\tilde{C}_k$

$x \in \vec{p} \cup \vec{q} \cup \vec{r}$. Case analysis: | MIU

Case 1 ($x \in \vec{p}$)

If $\rho(x) = 0$, set $\tilde{C}_i = \tilde{C}_j$

$type(\tilde{C}_i) = type(\tilde{C}_j)$

$\mathcal{E}_i = \mathcal{E}_j$ if needed

P1 clearly OK

If $\tilde{C}_i \not\models \rho \neq 1$, then

$\tilde{C}_i \not\models \subseteq C_j \setminus \{a, \bar{a} \mid a \in \rho\}$

$\subseteq C \setminus \{a, \bar{a} \mid a \in \rho\}$

$\subseteq (C \vee D) \setminus \{a, \bar{a} \mid a \in \rho\}$

$= C_i \setminus \{a, \bar{a} \mid a \in \rho\}$

If $\tilde{C}_i = 1$, then $C \wedge \rho = 1$ since $\rho(x) = 0$, so $C_i \wedge \rho = (C \vee D) \wedge \rho = 1$ and $\mathcal{E}_i = \mathcal{E}_j$ works as justification axiom.

P2 & P3 OK

If $\rho(x) = 1$, set $\tilde{C}_i = \tilde{C}_k$

$type(\tilde{C}_i) = type(\tilde{C}_k)$

$\mathcal{E}_i = \mathcal{E}_k$ if needed

Argument same as for $\rho(x) = 1$.

Case 2 ($x \in \vec{q}$)

Divide into subcases depending on types of $\tilde{C}_j$ and $\tilde{C}_k$

(a) If one of $\tilde{C}_j$ and $\tilde{C}_k$ is r-clause, set $\tilde{C}_i$ to that clause (choose arbitrarily if both are r-clauses).
Set type($\tilde{C}_i$) = r — P1 clearly OK
Copy justification clause to $\tilde{C}_i$ if needed

Observe:
- $\tilde{C}_i$ does not contain any $\vec{q}$-variables (by IH)
- $x \in \vec{q}$ only variable that disappears in resolution step

therefore P2 & P3 OK.

(b) If $\tilde{C}_j$ or $\tilde{C}_k$ is q-clause not containing x (e.g., if $= 1$) Let $\tilde{C}_i$ = such q-clause without x (choose arbitrarily if both qualify)
Copy justification clause to $\tilde{C}_i$ if needed.

Note that since no variable in $\vec{p}$ disappears in resolution step, my (needed is) justification clause still OK.

(c) If $\tilde{C}_j$ or $\tilde{C}_k$ q-clause not containing $\bar{x}$ let $\tilde{C}_i$ be such q-clause. Argue as in (b).

(d) If none of previous cases apply, then we have g-clauses

$\tilde{C}_j = \tilde{C}_j' \vee x \qquad \tilde{C}_k = \tilde{C}_k' \vee \bar{x}$

both nontrivial (i.e., $\neq 1$)

Let $\tilde{C}_i$ resolvent of these clauses.

with type $(\tilde{C}_i) = g$

P1 Resolvent of two g-clauses $\Rightarrow$ g-clause

P2 $\tilde{C}_i = \tilde{C}_j \vee \tilde{C}_k \setminus \{x, \bar{x}\}$

$\subseteq (C_j \vee C_k \setminus \{a, \bar{a} \mid a \in g\}) \setminus \{x, \bar{x}\}$

$= C_i \setminus \{a, \bar{a} \mid a \in g\}$

P3 Vacuously OK since $\tilde{C}_i \neq 1$.

Case 3 ($x \in \vec{r}$)

Analogous.

If one of $\tilde{C}_j$ and $\tilde{C}_k$ is g-clause, copy that clause. Otherwise copy or construct r-clause. Details are left as an exercise.

Part 2 Now follows by induction principle. Final clause $C_L = \bot$ gets classified as g-clause or r-clause $\tilde{C}_L \subseteq C_L = \bot$ means $\tilde{C}_L = \bot$. Derived only from $A(g, \vec{g})$ if g-clause or $B(g, \vec{r})$ if r-clause

Proof of part (1)

Go over $\pi = (C_1, C_2, \ldots, C_L = 1)$

Look at construction of $\tilde{\pi} = (\tilde{C}_1, \tilde{C}_2, \ldots, \tilde{C}_L = 1)$

For each C_i construct subcircuit with output gate $\tilde{v}_i$

computing

$$type(\tilde{C}_i) = \begin{cases} 0 & \text{if } \tilde{C}_i \text{ } q\text{-clause} \\ 1 & \text{if } \tilde{C}_i \text{ } r\text{-clause} \end{cases}$$

Just inspect proof of part (2)

Base case

$C_i \in A(\vec{p}, \vec{q}) \Rightarrow \tilde{v}_i$ constant 0

$C_i \in B(\vec{p}, \vec{r}) \Rightarrow \tilde{v}_i$ constant 1

Induction step

Again case analysis over resolution variable $x \in \vec{p} \cup \vec{q} \cup \vec{r}$

Case 1 ($x \in \vec{p}$)

$type(\tilde{C}_i) = sel(x, type(\tilde{C}_j), type(\tilde{C}_k))$

$= sel(x, v_j, v_k)$

Case 2 ($x \in \vec{q}$)

$type(\tilde{C}_i) = 1$ if one of $\tilde{C}_j, \tilde{C}_k$ has type r

otherwise $= 0$

$type(\tilde{C}_i) = type(\tilde{C}_j) \vee type(\tilde{C}_k)$

$= v_j \vee v_k$

Case 3 ($x \in \vec{r}$)

type $(\tilde{C}_i) = v_j \wedge v_k$

Details left as exercise

By the induction principle, final gate v_L will compute type$(\tilde{C}_L)$ = type$(\perp)$

Yields correct interpolating circuit

Proof of part (3)

Selector gates are non-monotone. Need to get rid of them

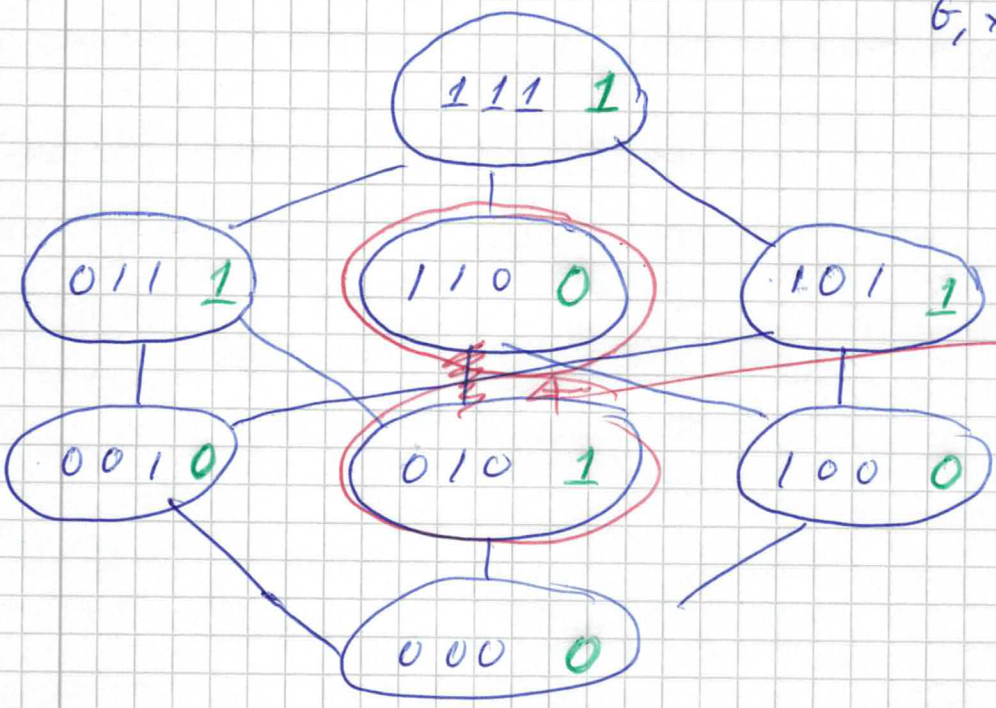
If $\vec{p}$-variables appear only positively in $A(\vec{p}, \vec{q})$, solution is described in LaTeXed notes.

$\vec{p}$-variables only negative in $B(\vec{p}, \vec{r})$: Left as exercise.

This is where we use the "justification axioms" $\mathcal{E}_i$

///

But the circuit is not monotone, because $sel(b,x,y)$ is not monotone



Suppose w.l.o.g. $\vec{p}$ only appears positively in $A(\vec{p},\vec{q})$
(Other case analogous.)
Replace $sel(b,x,y)$ by $(b \vee x) \wedge y$

Only difference for $b=0, x=1, y=0$ ↦ 0

$\alpha(p_k) = 0$ so we should have picked type from $\tilde{C}$, which was an r-clause, but instead we are picking q-clause $\tilde{D}$

WRONG if $\tilde{D}$ contains $\bar{p}_k$, because then $\tilde{D}$ partial but $C \vee D\restriction_{\vec{\alpha}}$ not, and condition $\tilde{D} \leq C \vee D\restriction_{\vec{\alpha}}$ is violated.

But $\tilde{D}$ cannot contain negative literal $\bar{p}_k$ since it is derived from $A(\vec{p},\vec{q})$. So this replacement is OK