Parameterized Algorithms

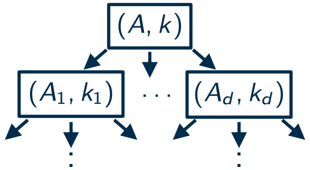
Parameterized Complexity and Databases

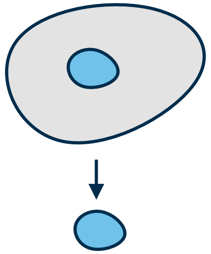
Thomas Bläsius

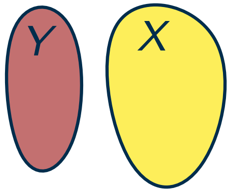
Content

Basic toolbox

- bounded search trees
- kernelization
- iterative compression

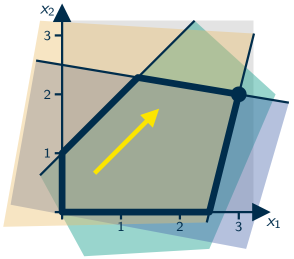


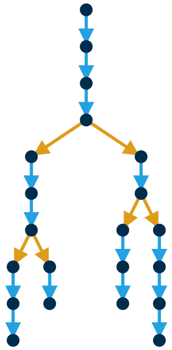


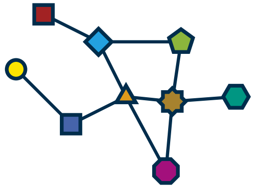


Extended toolbox

- linear programs
- branch-and-reduce
- color coding





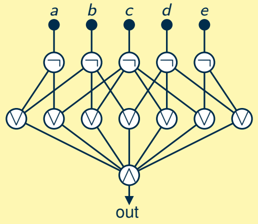


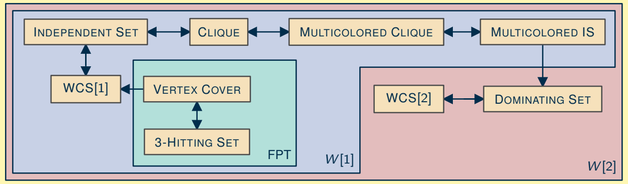
Tree width

- dynamic programming
- chordal and planar graphs
- Courcelle's theorem

Lower bonds

- kernel lower bounds
- parameterized reductions
- circuits and the W-hierarchy
- ETH and SETH





Reminder: Boolean Circuits

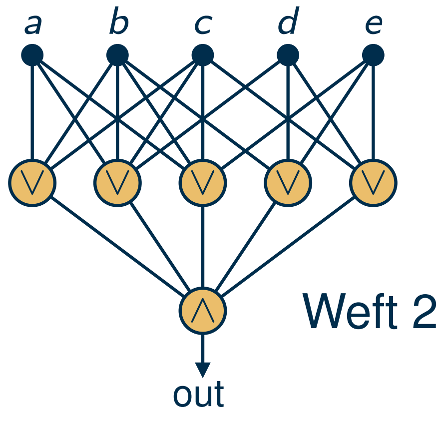
Definition
The largest number of nodes with in-degree > 2 on a directed path is called **weft** of the circuit.

Problem
WCS[t] restricts WCS to circuits of constant height and weft $\leq t$.

Definition
The class ***W*[t]** is the set of problems that have a parameterized reduction to WCS[t].

Remarks

- problem $\mathcal{L}$ is $W[t]$-complete if
  - $\mathcal{L} \in W[t]$ (parameterized reduction from $\mathcal{L}$ to WCS[T])
  - $\mathcal{L}$ is $W[t]$-hard (parameterized reduction from WCS[T] to $\mathcal{L}$)
- WCS[t] is $W[t]$-complete by definition
- we believe: FPT $\subset W[1] \subset W[2] \subset W[3] \subset \cdots$



Simple and difficult reductions

How do we prove $W[t]$-completeness?

- reduce to and from another complete problem
- INDEPENDENT SET or CLIQUE for $W[1]$ and DOMINATING SET or HITTING SET for $W[2]$
- for $t > 2$: here we only have WCS$[t]$ (so far)

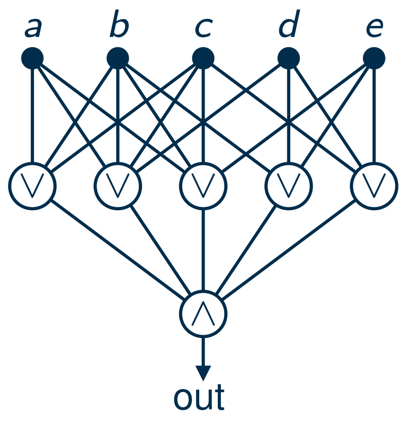
Reduction to WCS$[t]$ (containment in $W[t]$)

- WCS is a powerful modeling language $\rightarrow$ reduction usually rather simple

Reduction from WCS$[t]$ ($W[t]$-completeness)

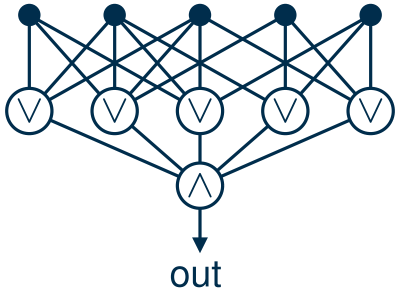
- this is usually more difficult
- one reason: the circuit might look really wild
- idea: forbid too wild circuits $\rightarrow$ normalization
- show $W[t]$-hardness for normalized circuits

not like this one $\rightarrow$



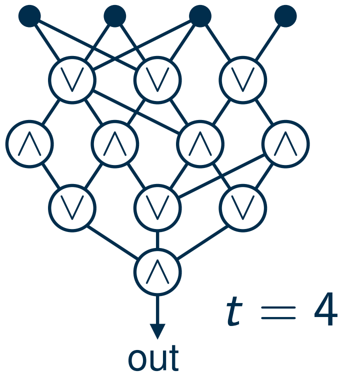
Normalized circuits

Properties of the circuit for DOMINATING SET



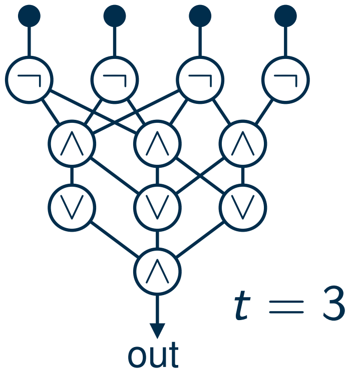
- on every path $\land$ and $\lor$ nodes alternate and output is $\land$ node
- no negations ($\neg$ node) $\Rightarrow$ circuit is **monoton**
- implication: superset of a solution is also a solution

WEIGHTED MONOTONE *t*-NORMALIZED SATISFIABILITY



- t layers of alternating $\land$ and $\lor$, output is $\land$, no $\neg$ nodes
- thus: $\underbrace{\text{conjunction of disjunctions of conjunctions} \dots \text{of}}_{t \text{ times ``of''}}$ positive literals

WEIGHTED ANTIMONOTONE *t*-NORMALIZED SATISFIABILITY

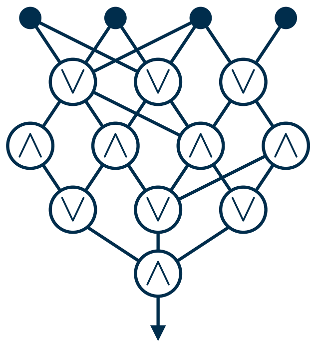


- same, but every input node must be negated
- thus: $\underbrace{\text{conjunction of disjunctions of conjunctions} \dots \text{of}}_{t \text{ times ``of''}}$ negative literals

Completeness of normalized circuits

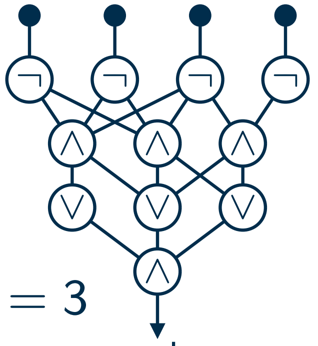
Theorem (without proof)
Weighted Monotone t-Normalized Satisfiability is $W[t]$-complete for every even t.

Theorem (without proof)
Weighted Antimonotone t-Normalized Satisfiability is $W[t]$-complete for every odd $t \geq 3$.



$t = 4$

out



$t = 3$

out

Remarks

- WMNS[t] and WANS[t] are obviously in $W[t]$
- the difficult part is proving $W[t]$-hardness
- using the theorems, we can now reduce from WMNS[t] or WANS[t] to prove $W[t]$-hardness
- rule of thumb: choose even t for minimization and odd t for maximization problems

Candidate keys in databases

Definition
A column set A is a **unique column combination** if every tuple is unique with respect to A.

Problem: Unique
Given a database and a parameter k. Is there a unique column combination of size $\leq k$?

	first name	name	birthday	place
	John	Doe	1.6.	Karlsruhe
r_2	Bob	Doe	7.10.	Berlin
	Bob	Builder	4.3.	Hamburg
	Alice	Liddell	4.5.	Wonderland
r_5	Alice	Smith	7.10.	Berlin

Example

- no single column is unique
- {first name, name} is unique
- {birthday, place} is not

Slightly different perspective

- row pair (r_2, r_5) $\Rightarrow$ **first name** or **name** must be included
- every row pair (r_i, r_j) yields column set $A_{i,j}$ from which at least one element must be selected

Do you know this problem?

$\rightarrow$ Hitting Set

UNIQUE is $W[2]$-complete

Just seen

- param. reduction from UNIQUE to the $W[2]$-complete problem HITTING SET $\Rightarrow$ UNIQUE $\in W[2]$

Now: Reduction from HITTING SET to UNIQUE

- use one column per element
- every S_i must be represented by a row pair
- use same 0-row s_0 for each pair
- pair (r_0, r_i) guarantees that at least one element from S_i is selected
- thus: solution for UNIQUE $\Rightarrow$ solution for HITTING SET
- **careful:** other pairs also yield requirements
- but: (r_i, r_j) yields superset $S_i \cup S_j$
- r_i and r_0 different $\Rightarrow$ r_i and r_j different
- thus: solution for HITTING SET $\Rightarrow$ solution for UNIQUE

$\Rightarrow$ UNIQUE is $W[2]$-hard

instance of HITTING SET

$U = \{a, b, c, d, e\}$

$S_1 = \{a, b, c\}$

$S_2 = \{a, d, e\}$

$S_3 = \{b, d, e\}$

$S_4 = \{b, c\}$

instance of UNIQUE

	A	B	C	D	E
r_0	0	0	0	0	0
r_1	1	1	1	0	0
r_2	2	0	0	2	2
r_3	0	3	0	3	3
r_4	0	4	4	0	0

Functional dependencies

Observation

- {name} is not a unique column combination
- the name alone does not uniquely identify the row
- but: the name identifies the place

first name	name	birthday	place
John	Doe	7.10.	Karlsruhe
Bob	Doe	1.6.	Karlsruhe
Bob	Builder	7.10.	Karlsruhe
Alice	Liddell	4.5.	Wonderland
Alice	Smith	4.3.	Berlin
	X		A

Definition
For a column set X and a column $A \notin X$, $X \to A$ is a **functional dependency** if for every row pair (r_i, r_j), it holds that $r_i[A] \neq r_j[A] \Rightarrow r_i[X] \neq r_j[X]$.

Which of the following are functional dependencies?

- {first name} $\to$ birthday
- {birthday} $\to$ first name
- {birthday} $\to$ place
- {first name} $\to$ place
- {first name, birthday} $\to$ place
- {place, birthday} $\to$ first name

Complexity of functional dependencies

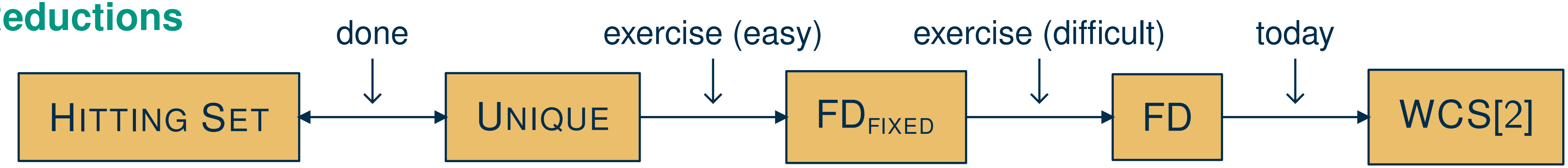
Problem: FD
Given a database and a parameter k. Is there a functional dependency $X \rightarrow A$ with $|X| \leq k$?

Prove $W[2]$-hardness for FD

- reduce from UNIQUE
- easier if we fix right-hand-side A

Problem: FD$_{\text{FIXED}}$
Given a database, a column A, and a parameter k. Is there a functional dependency $X \rightarrow A$ with $|X| \leq k$?

Reductions



- HITTING SET and WCS[2] are $W[2]$-complete $\Rightarrow$ FD$_{\text{FIXED}}$ and FD are $W[2]$-complete

FD $\to$ WCS[2]

Definition
For a column set X and a column $A \notin X$, $X \to A$ is a **functional dependency** if for every row pair (r_i, r_j), it holds that $r_i[A] \neq r_j[A] \Rightarrow r_i[X] \neq r_j[X]$.

Rough plan: model conditions of FD using Boolean formulas

- use only $\vee$ for the conditions and then one $\wedge$ in the end (CNF) $\Rightarrow$ weft 2

What are the conditions? What variables do we use?

- RHS consists of exactly one column
- LHS and RHS are disjoint column sets
- $A \in$ RHS and $r_i[A] \neq r_j[A]$ $\Rightarrow$ $\exists B$ with $B \in$ LHS and $r_i[B] \neq r_j[B]$

indicator variables x_A and y_A for $A \in$ LHS and $A \in$ RHS

RHS: exactly one column

- at least one: $\bigvee_{\text{column } A} y_A$
- not both A and B: $\neg y_A \vee \neg y_B$

LHS and RHS are disjoint

- for every column A: $\neg x_A \vee \neg y_A$

LHS and RHS form functional dependency

- for every column A and pair (r_i, r_j) with $r_i[A] \neq r_j[A]$:

$$\neg y_A \vee \bigvee_{\substack{\text{column } A \neq B \\ r_i[B] \neq r_j[B]}} x_B$$

Theorem: UNIQUE, FD$_{\text{FIXED}}$, and FD are $W[2]$-complete.

Inclusion dependencies

Goal: find data of one database in another database

- consider columns $X = \{A, C\} \subseteq R$
- and the map $\sigma : X \to S$ with $\sigma(A) = G$ and $\sigma(C) = E$
- tuple for X $\{11, 32, 23, 34\} \subseteq$ tuple for $\sigma(X)$ $\{11, 13, 23, 32, 34\}$

schema R

A	B	C
1	1	1
3	1	2
1	2	1
2	3	3
3	4	4

schema S

D	E	F	G
4	1	3	1
3	3	4	1
3	3	3	2
4	2	1	3
2	4	1	3
4	4	4	3

Definition
Consider two instances r and s of two schemata R and S. A subset $X \subseteq R$ together with an injective map $\sigma : X \to S$ is an **inclusion dependency**, if $r[X] \subseteq s[\sigma(X)]$.

Problem: IND
Given two databases r and s, and a parameter k. Is there an inclusion dependency of size k?

Problem: IND$_{\text{FIXED}}$
Given two databases r, s with the same schema and a parameter k. Is there an inclusion dependency of size k with σ being the identity?

Find the largest inclusion dependency

5	M	I	N
1	1	4	3
4	1	2	4
2	4	3	2
1	3	4	4
2	3	4	1
4	1	1	4
2	3	1	1

B	R	E	A	K
3	3	1	1	3
4	3	3	1	3
3	1	3	4	2
2	3	2	2	4
1	2	3	2	3
2	2	2	3	1
2	1	4	2	2
4	4	1	4	4

Definition
Consider two instances r and s of two schemata R and S. A subset $X \subseteq R$ together with an injective map $\sigma : X \to S$ is an **inclusion dependency**, if $r[X] \subseteq s[\sigma(X)]$.

Find the largest inclusion dependency

5	M	I	N
1	1	4	3
4	1	2	4
2	4	3	2
1	3	4	4
2	3	4	1
4	1	1	4
2	3	1	1

B	R	E	A	K
3	3	1	1	3
4	3	3	1	3
3	1	3	4	2
2	3	2	2	4
1	2	3	2	3
2	2	2	3	1
2	1	4	2	2
4	4	1	4	4

Solution

- $X = \{5, M, N\}$
- $\sigma(5) = A$, $\sigma(M) = E$, and $\sigma(N) = B$

Definition
Consider two instances r and s of two schemata R and S. A subset $X \subseteq R$ together with an injective map $\sigma\colon X \to S$ is an **inclusion dependency**, if $r[X] \subseteq s[\sigma(X)]$.

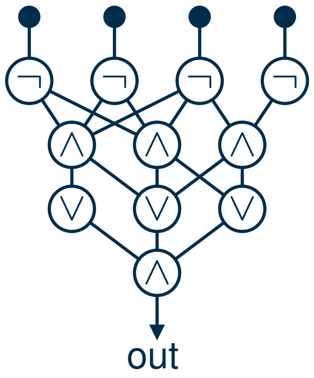
IND is $W[3]$-complete

Reduction from $\mathrm{IND}_{\mathrm{FIXED}}$ to IND
- easy

How?

$\mathrm{IND}_{\mathrm{FIXED}} \rightarrow \mathrm{IND}$

Reduction from IND to WANS[3]
- model IND-conditions as Boolean formulas
- ensure: conj. of disj. of conj. of negative literals



$\mathrm{IND} \rightarrow \mathrm{WANS}[3]$

Reduction from WANS[3] to $\mathrm{IND}_{\mathrm{FIXED}}$
- model antimonotone 3-normalized Boolean formula with two databases

$\mathrm{WANS}[3] \rightarrow \mathrm{IND}_{\mathrm{FIXED}}$

Overall plan:

$\mathrm{WANS}[3] \rightarrow \mathrm{IND}_{\mathrm{FIXED}} \rightarrow \mathrm{IND} \rightarrow \mathrm{WANS}[3]$

WANS[3] $\rightarrow$ IND$_{\text{FIXED}}$: the final “and”

Viewing IND$_{\text{FIXED}}$ as Boolean function

- formalize set of columns as 01-string (e.g., $\{A, C\} \equiv 101$)
- Boolean function f: $f(X) = 1 \Leftrightarrow X$ is IND (e.g., $f(101) = 1$, $f(110) = 0$)

database r

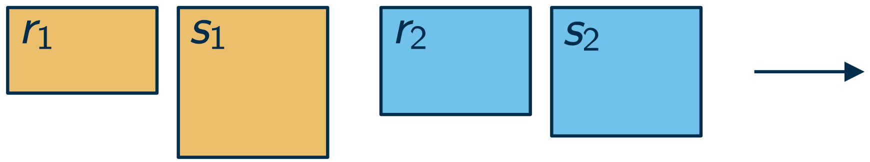
A	B	C
1	1	1
3	1	2
1	2	1
2	3	3
3	4	4

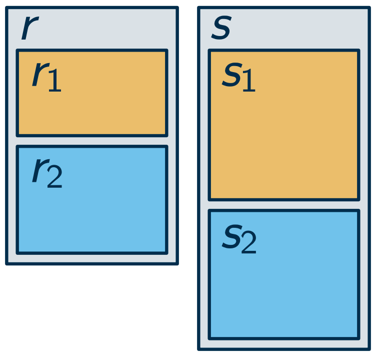
database s

A	B	C
1	3	1
1	4	3
2	3	3
3	1	2
3	1	4
3	4	4

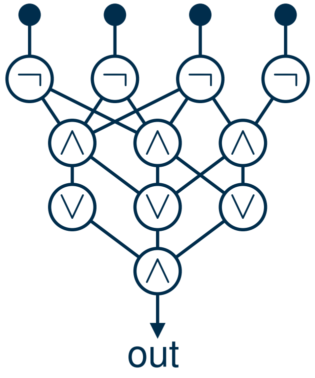
“And”ing two instances (on the same Schema)

- consider two instances (r_1, s_1) and (r_2, s_2) with functions f_1 and f_2
- Is there an instance (r, s) such that $f = f_1 \wedge f_2$?
- make sure that (r_1, s_1) and (r_2, s_2) have disjoint values and concatenate the databases





Why does $f = f_1 \wedge f_2$ hold?



WANS[3] $\to$ IND$_{\text{FIXED}}$

Lemma
Given two IND$_{\text{FIXED}}$-instances with indicator functions f_1 and f_2, then there is a (not too large) IND$_{\text{FIXED}}$-Instanz with indicator function $f_1 \wedge f_2$.

Achievement so far

- we can model conjunctions ($\wedge$)
- Can we model disjunctions ($\vee$) similarly?
- would show $W[t]$-hardness for every t

Disjunction of conjunction of negative literals

$\varphi = c_1 \vee c_2 \vee c_3$

$c_1 = (\neg x_1 \wedge \neg x_2 \wedge \neg x_3)$

$c_2 = (\neg x_2 \wedge \neg x_4 \wedge \neg x_5)$

$c_3 = (\neg x_1 \wedge \neg x_3 \wedge \neg x_4 \wedge \neg x_6)$

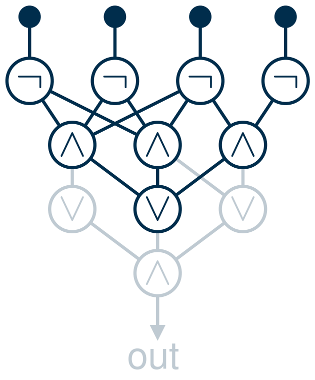
r:

A_1	A_2	A_3	A_4	A_5	A_6
0	0	0	0	0	0

s:

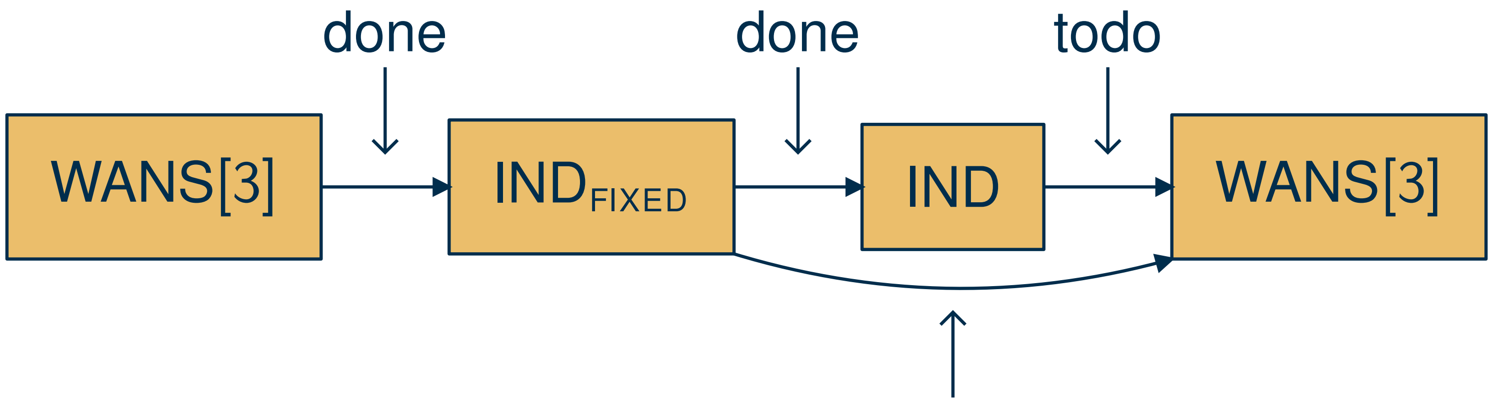
A_1	A_2	A_3	A_4	A_5	A_6
1	1	1	0	0	0
0	1	0	1	1	0
1	0	1	1	0	1

- modeling a single c_i is easy
- for IND it suffices to find the data from r in one row of s
  multiple rows in s $\to$ naturally models "or" between them



Check: solution for φ yields solution for (r, s) and vice versa

Seen so far



easier and can be extended to IND $\rightarrow$ WANS[3]

IND$_{(\text{FIXED})}$ $\to$ WANS[3]

IND$_{\text{FIXED}}$ as Boolean formula

- single pair of rows (r_i, s_j):
  exclude certain columns
- now multiple rows in s:
  find r_i in a row of s $\to$ "or"
- also multiple rows in r:
  find each row of r in s $\to$ "and"

	A_1	A_2	A_3	A_4
r_i	1	0	1	0

	A_1	A_2	A_3	A_4
s_j	1	1	0	0

$\longrightarrow \neg x_2 \wedge \neg x_3$

	A_1	A_2	A_3	A_4
r_i	1	0	1	0

	A_1	A_2	A_3	A_4
s_1	1	1	0	0
s_2	0	1	1	0

$\longrightarrow \underbrace{\neg x_2 \wedge \neg x_3}_{(r_j \mapsto s_1)} \vee \underbrace{\neg x_1 \wedge \neg x_2}_{(r_j \mapsto s_2)}$

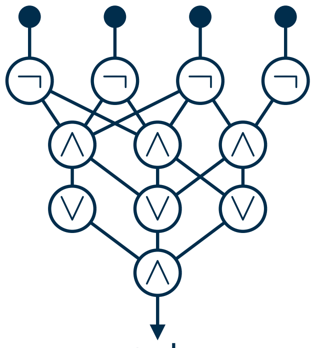
	A_1	A_2	A_3	A_4
r_1	1	0	1	0
r_2	0	1	0	1

	A_1	A_2	A_3	A_4
s_1	1	1	0	0
s_2	0	1	1	0

$$\longrightarrow \Big(\overbrace{\neg x_2 \wedge \neg x_3}^{(r_1 \mapsto s_1)} \vee \overbrace{\neg x_1 \wedge \neg x_2}^{(r_1 \mapsto s_2)}\Big) \wedge \Big(\underbrace{\neg x_1 \wedge \neg x_4}_{(r_2 \mapsto s_1)} \vee \underbrace{\neg x_3 \wedge \neg x_4}_{(r_2 \mapsto s_2)}\Big)$$

Extension to IND $\to$ WANS[3]

- variables $x_{i,j}$ with: $x_{i,j} = 1 \;\Leftrightarrow\;$ col i selected in R and mapped to col j in S
- make sure this gives a bijection
- combine with previous conditions



> **Theorem**
> IND$_{\text{FIXED}}$ and IND are $W[3]$-complete.

Wrap-Up

Normalized circuits

- we can start reductions from WCS with rather clean circuits
- no need to look at weft; just count the layers
- even t: monotone; odd t: antimonotone

Dependency detection

- rare case of a non-graph problem
- there are natural $W[3]$-complete problems
- does not explain why functional dependencies can be computed efficiently in practice

  (one can actually enumerate all inclusion-wise minimal UCCs/FDs rather quickly)

Literature

The Parameterized Complexity of Dependency Detection in Relational Databases

- Thomas Bläsius, Tobias Friedrich, Martin Schirneck [2016]
- contains the reduction from today doi.org/10.4230/LIPIcs.IPEC.2016.6

Profiling relational data: a survey

- Ziawasch Abedjan, Lukasz Golab, Felix Naumann [2015]
- overview on finding dependencies in databases doi.org/10.1007/s00778-015-0389-y

Hitting Set Enumeration with Partial Information for Unique Column Combination Discovery

- Johann Birnick, Thomas Bläsius, Tobias Friedrich, Felix Naumann, Thorsten Papenbrock, Martin Schirneck [2020]
- practically efficient algorithm for finding unique column combinations doi.org/10.14778/3407790.3407824

Discovering Functional Dependencies through Hitting Set Enumeration

- Tobias Bleifuß, Thorsten Papenbrock, Thomas Bläsius, Martin Schirneck, Felix Naumann [2024]
- practically efficient algorithm for finding functional dependencies doi.org/10.1145/3639298