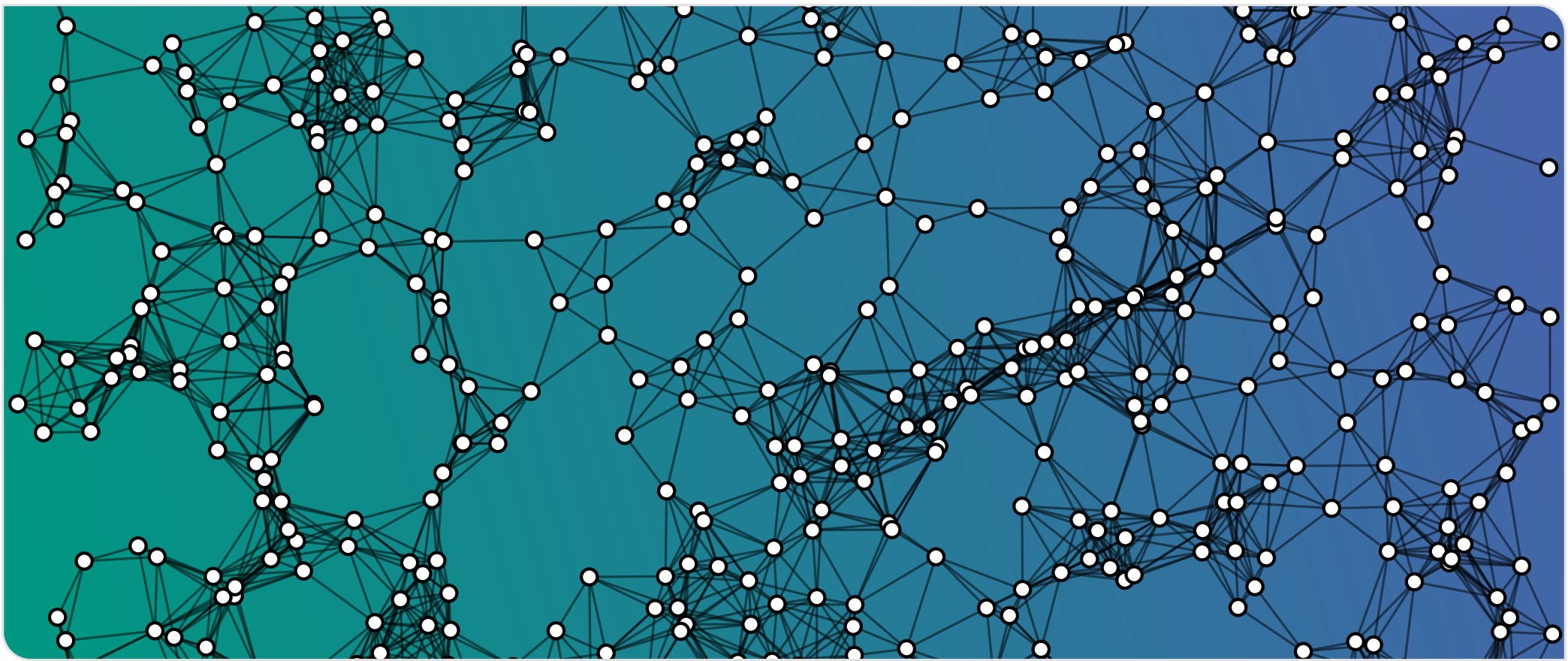
Karlsruher Institut für Technologie

# Parametrisierte Algorithmen

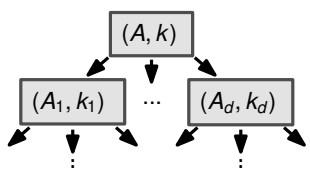
## Untere Schranken: parametrisierte Komplexität und Datenbanken

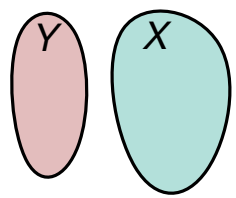


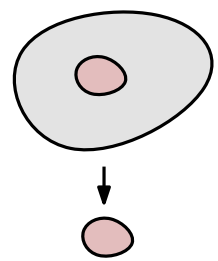
# Inhalt

## Basic Toolbox

- beschränkte Suchbäume
- iterative Kompression
- Kernbildung

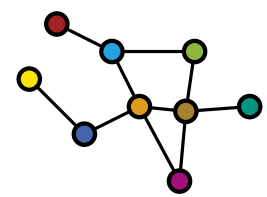


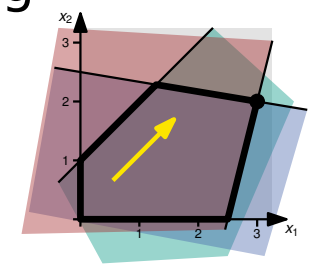



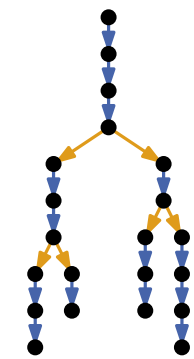



## Erweiterte Toolbox

- lineare Programme
- Branch-and-Reduce
- Color Coding





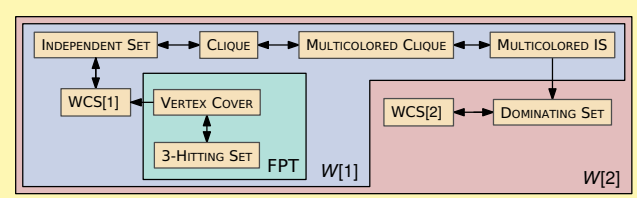



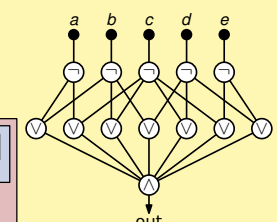
## Baumweite

- dynamische Programme
- chordale & planare Graphen
- Courcelles Theorem

## Untere Schranken

- parametrisierte Reduktionen
- boolesche Schaltkreise und die W-Hierarchie
- ETH und SETH



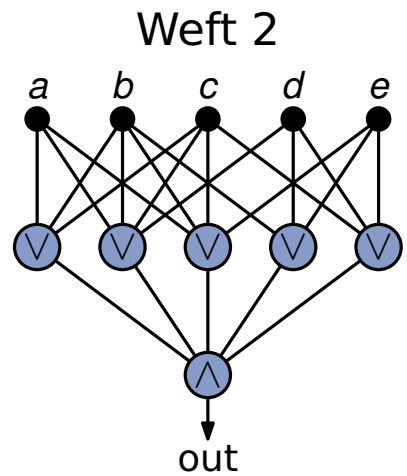



# Wiederholung: Boolesche Schaltkreise

**Definition**
Der **Weft** eines Booleschen Schaltkreises ist die maximale Anzahl an Knoten mit Eingangsgrad $> 2$ auf einem gerichteten Pfad.

**Problem**
**WCS**$[t]$ ist WCS eingeschränkt auf Schaltkreise mit konstanter Tiefe und Weft maximal $t$.

**Definition**
Die Klasse $\boldsymbol{W}[\boldsymbol{t}]$ enthält die Probleme, die eine parametrisierte Reduktion auf WCS$[t]$ zulassen.

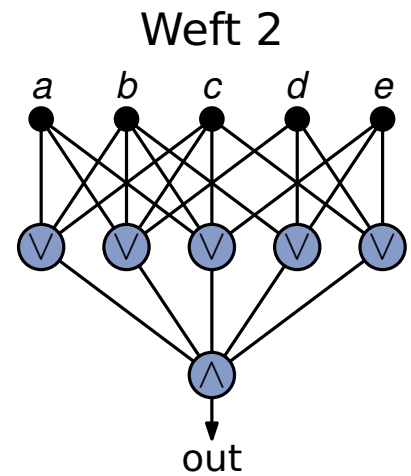


# Wiederholung: Boolesche Schaltkreise

**Definition**
Der **Weft** eines Booleschen Schaltkreises ist die maximale Anzahl an Knoten mit Eingangsgrad $> 2$ auf einem gerichteten Pfad.

**Problem**
**WCS**$[t]$ ist WCS eingeschränkt auf Schaltkreise mit konstanter Tiefe und Weft maximal $t$.

**Definition**
Die Klasse $\boldsymbol{W}[\boldsymbol{t}]$ enthält die Probleme, die eine parametrisierte Reduktion auf WCS$[t]$ zulassen.




## Bemerkungen

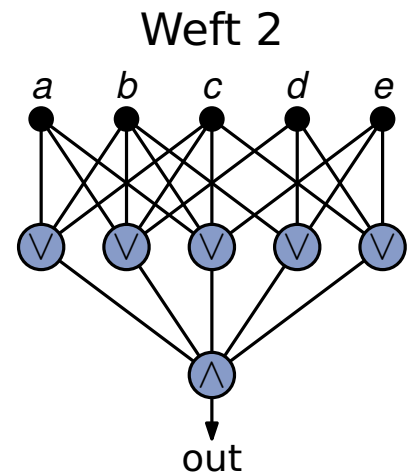
- Problem $\mathcal{L}$ ist $W[t]$-vollständig, wenn
  - $\mathcal{L} \in W[t]$
  - $\mathcal{L}$ $W[t]$-schwer (jedes $\mathcal{L}' \in W[t]$ lässt sich parametrisiert auf $\mathcal{L}$ reduzieren)

# Wiederholung: Boolesche Schaltkreise

**Definition**
Der **Weft** eines Booleschen Schaltkreises ist die maximale Anzahl an Knoten mit Eingangsgrad $> 2$ auf einem gerichteten Pfad.

**Problem**
**WCS**$[t]$ ist WCS eingeschränkt auf Schaltkreise mit konstanter Tiefe und Weft maximal $t$.

**Definition**
Die Klasse $\boldsymbol{W}[\boldsymbol{t}]$ enthält die Probleme, die eine parametrisierte Reduktion auf WCS$[t]$ zulassen.




## Bemerkungen

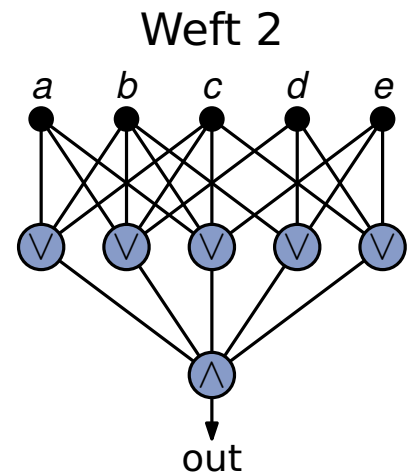
- Problem $\mathcal{L}$ ist $W[t]$-vollständig, wenn
  - $\mathcal{L} \in W[t]$
  - $\mathcal{L}$ $W[t]$-schwer (jedes $\mathcal{L}' \in W[t]$ lässt sich parametrisiert auf $\mathcal{L}$ reduzieren)
- WCS$[t]$ ist per Definition $W[t]$-vollständig

# Wiederholung: Boolesche Schaltkreise

**Definition**
Der **Weft** eines Booleschen Schaltkreises ist die maximale Anzahl an Knoten mit Eingangsgrad $> 2$ auf einem gerichteten Pfad.

**Problem**
**WCS**$[t]$ ist WCS eingeschränkt auf Schaltkreise mit konstanter Tiefe und Weft maximal $t$.

**Definition**
Die Klasse $\boldsymbol{W}[\boldsymbol{t}]$ enthält die Probleme, die eine parametrisierte Reduktion auf WCS$[t]$ zulassen.




## Bemerkungen

- Problem $\mathcal{L}$ ist $W[t]$-vollständig, wenn
  - $\mathcal{L} \in W[t]$
  - $\mathcal{L}$ $W[t]$-schwer (jedes $\mathcal{L}' \in W[t]$ lässt sich parametrisiert auf $\mathcal{L}$ reduzieren)
- WCS$[t]$ ist per Definition $W[t]$-vollständig
- Vermutung: FPT $\subset W[1] \subset W[2] \subset W[3] \subset \cdots$

# Leichte und schwere Reduktionen

## Wie zeigen wir $W[t]$-Vollständigkeit?

- reduziere auf und von anderem vollständigen Problem

# Leichte und schwere Reduktionen

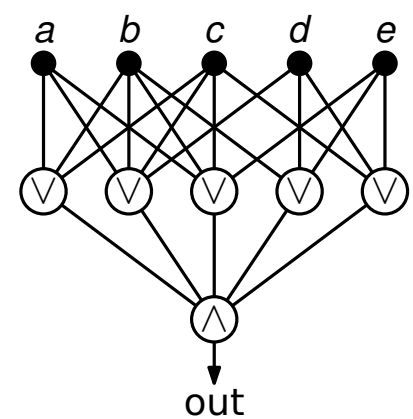
**Wie zeigen wir $W[t]$-Vollständigkeit?**

- reduziere auf und von anderem vollständigen Problem
- für $W[1]$: Independent Set oder Clique
- für $W[2]$: Dominating Set oder Hitting Set

# Leichte und schwere Reduktionen

**Wie zeigen wir $W[t]$-Vollständigkeit?**

- reduziere auf und von anderem vollständigen Problem
- für $W[1]$: Independent Set oder Clique
- für $W[2]$: Dominating Set oder Hitting Set
- für $t > 2$: hier haben wir nur WCS[$t$] zur Verfügung



# Leichte und schwere Reduktionen

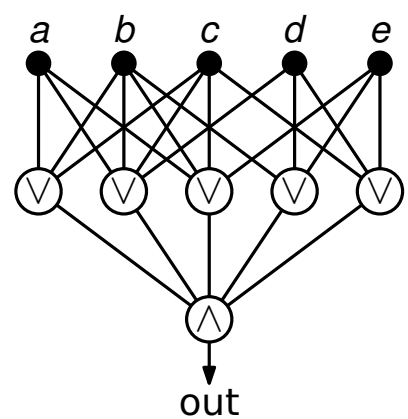
## Wie zeigen wir $W[t]$-Vollständigkeit?

- reduziere auf und von anderem vollständigen Problem
- für $W[1]$: Independent Set oder Clique
- für $W[2]$: Dominating Set oder Hitting Set
- für $t > 2$: hier haben wir nur WCS[$t$] zur Verfügung

## Reduktion nach WCS$[t]$

(Enthaltensein in $W[t]$)

- WCS ist eine mächtige Modellierungssprache
- Reduktion ist üblicherweise relativ einfach



# Leichte und schwere Reduktionen

## Wie zeigen wir $W[t]$-Vollständigkeit?

- reduziere auf und von anderem vollständigen Problem
- für $W[1]$: Independent Set oder Clique
- für $W[2]$: Dominating Set oder Hitting Set
- für $t > 2$: hier haben wir nur WCS[$t$] zur Verfügung

## Reduktion nach WCS$[t]$

(Enthaltensein in $W[t]$)

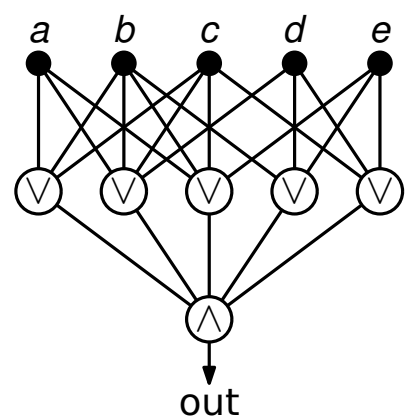
- WCS ist eine mächtige Modellierungssprache
- Reduktion ist üblicherweise relativ einfach

## Reduktion von WCS$[t]$

($W[t]$-Schwere)

- dieser Teil ist meist deutlich schwerer
- ein Grund: der Schaltkreis kann sehr wüst aussehen

(nicht so, wie der da $\rightarrow$)



# Leichte und schwere Reduktionen

## Wie zeigen wir $W[t]$-Vollständigkeit?

- reduziere auf und von anderem vollständigen Problem
- für $W[1]$: Independent Set oder Clique
- für $W[2]$: Dominating Set oder Hitting Set
- für $t > 2$: hier haben wir nur WCS[$t$] zur Verfügung

## Reduktion nach WCS$[t]$

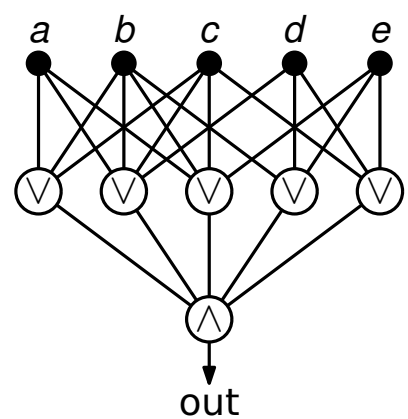
(Enthaltensein in $W[t]$)

- WCS ist eine mächtige Modellierungssprache
- Reduktion ist üblicherweise relativ einfach

## Reduktion von WCS$[t]$

($W[t]$-Schwere)

- dieser Teil ist meist deutlich schwerer
- ein Grund: der Schaltkreis kann sehr wüst aussehen
  (nicht so, wie der da $\rightarrow$)
- Idee: verbiete wüste Schaltkreise $\rightarrow$ Normalisierung



# Leichte und schwere Reduktionen

## Wie zeigen wir $W[t]$-Vollständigkeit?

- reduziere auf und von anderem vollständigen Problem
- für $W[1]$: Independent Set oder Clique
- für $W[2]$: Dominating Set oder Hitting Set
- für $t > 2$: hier haben wir nur WCS[$t$] zur Verfügung

## Reduktion nach WCS$[t]$

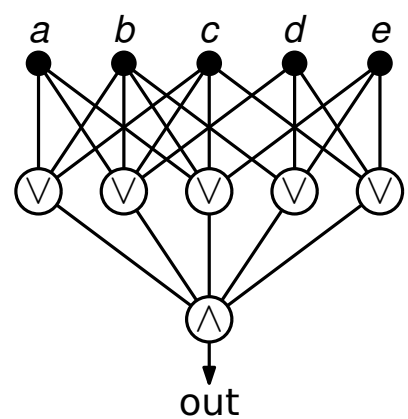
(Enthaltensein in $W[t]$)

- WCS ist eine mächtige Modellierungssprache
- Reduktion ist üblicherweise relativ einfach

## Reduktion von WCS$[t]$

($W[t]$-Schwere)

- dieser Teil ist meist deutlich schwerer
- ein Grund: der Schaltkreis kann sehr wüst aussehen
  (nicht so, wie der da $\rightarrow$)
- Idee: verbiete wüste Schaltkreise $\rightarrow$ Normalisierung
- zeige $W[t]$-Schwere für normalisierte Schaltkreise



# Leichte und schwere Reduktionen

## Wie zeigen wir $W[t]$-Vollständigkeit?

- reduziere auf und von anderem vollständigen Problem
- für $W[1]$: Independent Set oder Clique
- für $W[2]$: Dominating Set oder Hitting Set
- für $t > 2$: hier haben wir nur WCS[$t$] zur Verfügung

## Reduktion nach WCS$[t]$

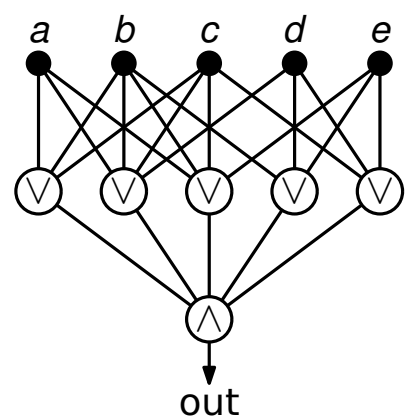
(Enthaltensein in $W[t]$)

- WCS ist eine mächtige Modellierungssprache
- Reduktion ist üblicherweise relativ einfach

## Reduktion von WCS$[t]$

($W[t]$-Schwere)

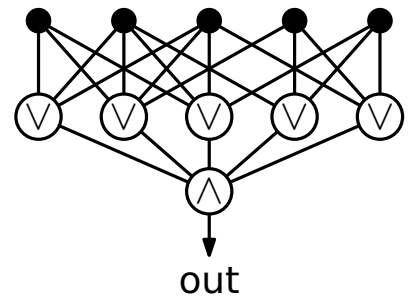
- dieser Teil ist meist deutlich schwerer
- ein Grund: der Schaltkreis kann sehr wüst aussehen
  (nicht so, wie der da $\rightarrow$)
- Idee: verbiete wüste Schaltkreise $\rightarrow$ Normalisierung
- zeige $W[t]$-Schwere für normalisierte Schaltkreise
- und reduziere dann von diesen weiter



# Normalisierte Schaltkreise

## Der Schaltkreis für Dominating Set

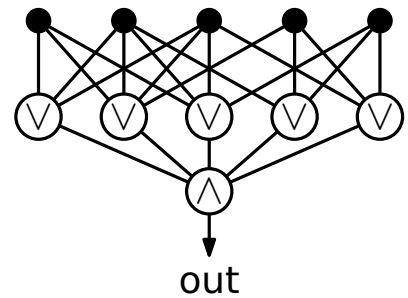
- auf jedem Pfad alternieren $\wedge$- und $\vee$-Knoten
- Ausgabeknoten (unterste Lage) ist ein $\wedge$-Knoten



# Normalisierte Schaltkreise

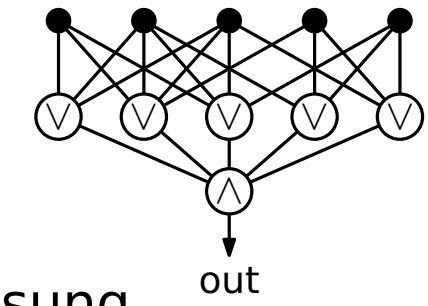
## Der Schaltkreis für Dominating Set

- auf jedem Pfad alternieren $\wedge$- und $\vee$-Knoten
- Ausgabeknoten (unterste Lage) ist ein $\wedge$-Knoten
- nur positive Variablen $\Rightarrow$ Schaltkreis ist **monoton**



# Normalisierte Schaltkreise

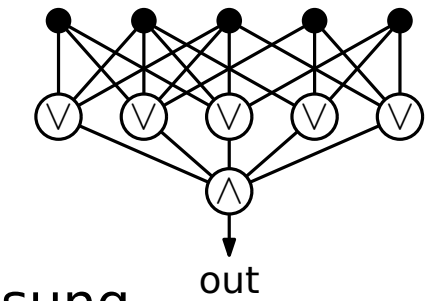
## Der Schaltkreis für Dominating Set




- auf jedem Pfad alternieren $\wedge$- und $\vee$-Knoten
- Ausgabeknoten (unterste Lage) ist ein $\wedge$-Knoten
- nur positive Variablen $\Rightarrow$ Schaltkreis ist **monoton**
- Implikation: Supermenge einer Lösung ist ebenfalls Lösung

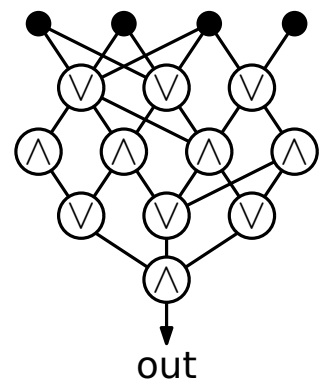
# Normalisierte Schaltkreise

## Der Schaltkreis für Dominating Set




- auf jedem Pfad alternieren $\wedge$- und $\vee$-Knoten
- Ausgabeknoten (unterste Lage) ist ein $\wedge$-Knoten
- nur positive Variablen $\Rightarrow$ Schaltkreis ist **monoton**
- Implikation: Supermenge einer Lösung ist ebenfalls Lösung

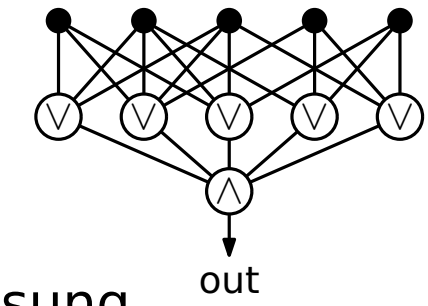
## Weighted Monotone *t*-Normalized Satisfiability




- $t$ Lagen alternierender $\wedge$- und $\vee$-Knoten
- $\wedge$-Konten als Ausgabeknoten
- keine $\neg$-Knoten erlaubt

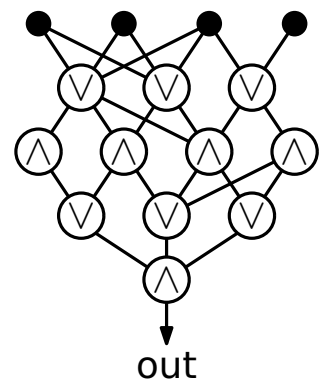
# Normalisierte Schaltkreise

## Der Schaltkreis für Dominating Set




- auf jedem Pfad alternieren $\wedge$- und $\vee$-Knoten
- Ausgabeknoten (unterste Lage) ist ein $\wedge$-Knoten
- nur positive Variablen $\Rightarrow$ Schaltkreis ist **monoton**
- Implikation: Supermenge einer Lösung ist ebenfalls Lösung

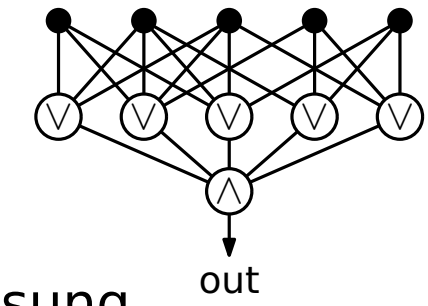
## Weighted Monotone $t$-Normalized Satisfiability




- $t$ Lagen alternierender $\wedge$- und $\vee$-Knoten
- $\wedge$-Konten als Ausgabeknoten
- keine $\neg$-Knoten erlaubt
- also: $\underbrace{\text{Konj. von Disj. von Konj. } \ldots \text{ von}}_{t \text{ „von“s}}$ positiven Literalen

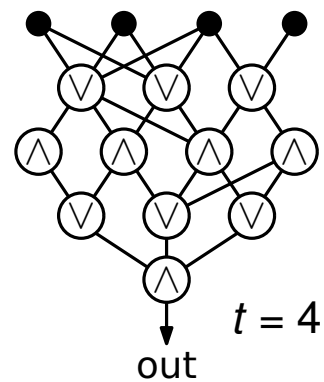
# Normalisierte Schaltkreise

## Der Schaltkreis für Dominating Set




- auf jedem Pfad alternieren $\wedge$- und $\vee$-Knoten
- Ausgabeknoten (unterste Lage) ist ein $\wedge$-Knoten
- nur positive Variablen $\Rightarrow$ Schaltkreis ist **monoton**
- Implikation: Supermenge einer Lösung ist ebenfalls Lösung

## Weighted Monotone $t$-Normalized Satisfiability

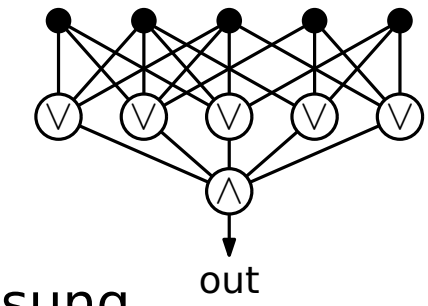



- $t$ Lagen alternierender $\wedge$- und $\vee$-Knoten
- $\wedge$-Konten als Ausgabeknoten
- keine $\neg$-Knoten erlaubt
- also: Konj. von Disj. von Konj. . . . von positiven Literalen

$t$ „von“s

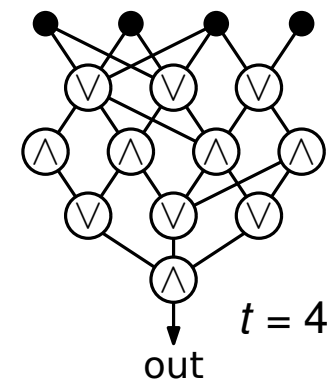
# Normalisierte Schaltkreise

## Der Schaltkreis für Dominating Set




- auf jedem Pfad alternieren $\wedge$- und $\vee$-Knoten
- Ausgabeknoten (unterste Lage) ist ein $\wedge$-Knoten
- nur positive Variablen $\Rightarrow$ Schaltkreis ist **monoton**
- Implikation: Supermenge einer Lösung ist ebenfalls Lösung

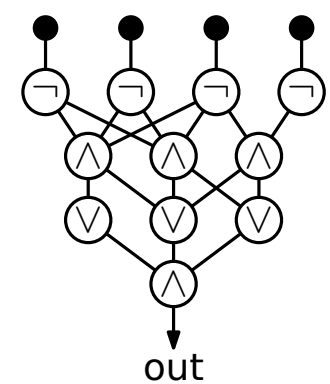
## Weighted Monotone $t$-Normalized Satisfiability




- $t$ Lagen alternierender $\wedge$- und $\vee$-Knoten
- $\wedge$-Konten als Ausgabeknoten
- keine $\neg$-Knoten erlaubt
- also: Konj. von Disj. von Konj. . . . von positiven Literalen

$t$ „von“s

## Weighted Antimonotone $t$-Normalized Satisfiability

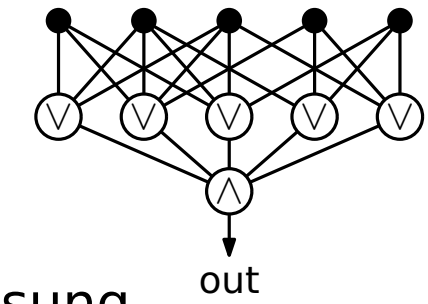



- jetzt muss jeder Eingangsknoten negiert werden
- also: Konj. von Disj. von Konj. . . . von negativen Literalen

$t$ „von“s

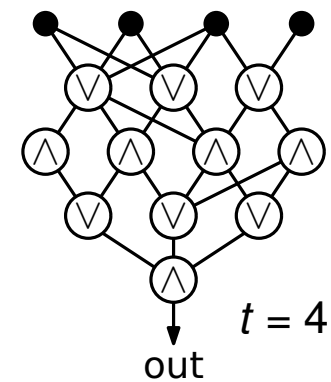
# Normalisierte Schaltkreise

## Der Schaltkreis für Dominating Set




- auf jedem Pfad alternieren $\wedge$- und $\vee$-Knoten
- Ausgabeknoten (unterste Lage) ist ein $\wedge$-Knoten
- nur positive Variablen $\Rightarrow$ Schaltkreis ist **monoton**
- Implikation: Supermenge einer Lösung ist ebenfalls Lösung

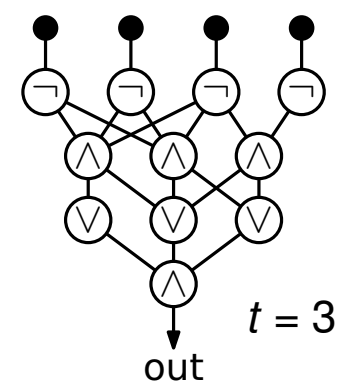
## Weighted Monotone $t$-Normalized Satisfiability




- $t$ Lagen alternierender $\wedge$- und $\vee$-Knoten
- $\wedge$-Konten als Ausgabeknoten
- keine $\neg$-Knoten erlaubt
- also: Konj. von Disj. von Konj. . . . von positiven Literalen

$t$ „von“s

## Weighted Antimonotone $t$-Normalized Satisfiability



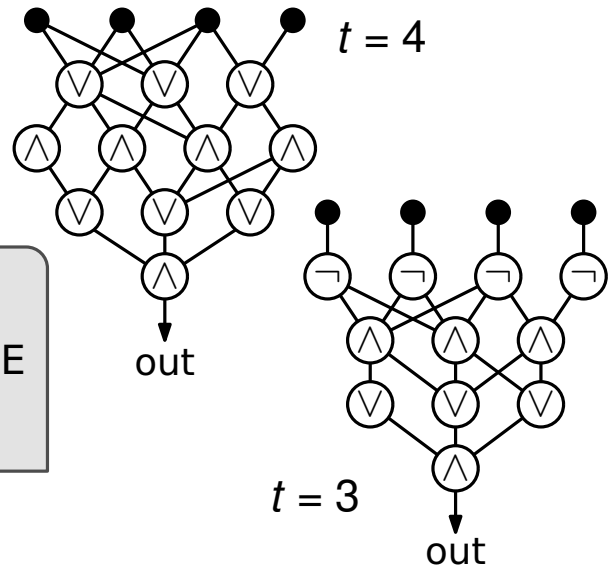

- jetzt muss jeder Eingangsknoten negiert werden
- also: Konj. von Disj. von Konj. . . . von negativen Literalen

$t$ „von“s

# Normalisierte Schaltkreise

**Theorem**
Für jedes gerade $t \geq 2$ ist WEIGHTED MONOTONE $t$-NORMALIZED SATISFIABILITY $W[t]$-vollständig.

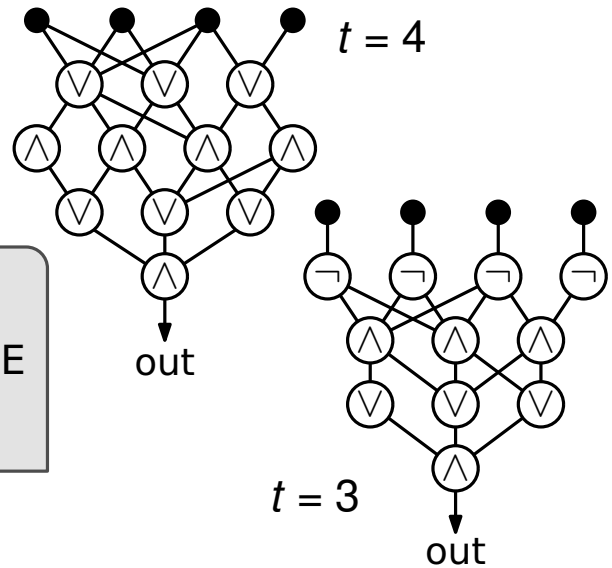
**Theorem**
Für jedes ungerade $t \geq 3$ ist WEIGHTED ANTIMONOTONE $t$-NORMALIZED SATISFIABILITY $W[t]$-vollständig.



# Normalisierte Schaltkreise

**Theorem**
Für jedes gerade $t \geq 2$ ist WEIGHTED MONOTONE $t$-NORMALIZED SATISFIABILITY $W[t]$-vollständig.

**Theorem**
Für jedes ungerade $t \geq 3$ ist WEIGHTED ANTIMONOTONE $t$-NORMALIZED SATISFIABILITY $W[t]$-vollständig.



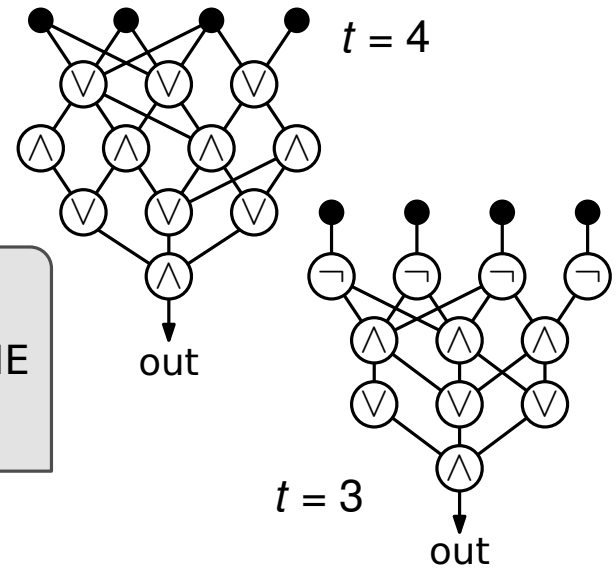

## Anmerkungen

- WMNS[$t$] und WANS[$t$] sind offensichtlich in $W[t]$ enthalten
- die Schwierigkeit besteht darin $W[t]$-Schwere zu zeigen

# Normalisierte Schaltkreise

**Theorem**
Für jedes gerade $t \geq 2$ ist WEIGHTED MONOTONE $t$-NORMALIZED SATISFIABILITY $W[t]$-vollständig.

**Theorem**
Für jedes ungerade $t \geq 3$ ist WEIGHTED ANTIMONOTONE $t$-NORMALIZED SATISFIABILITY $W[t]$-vollständig.



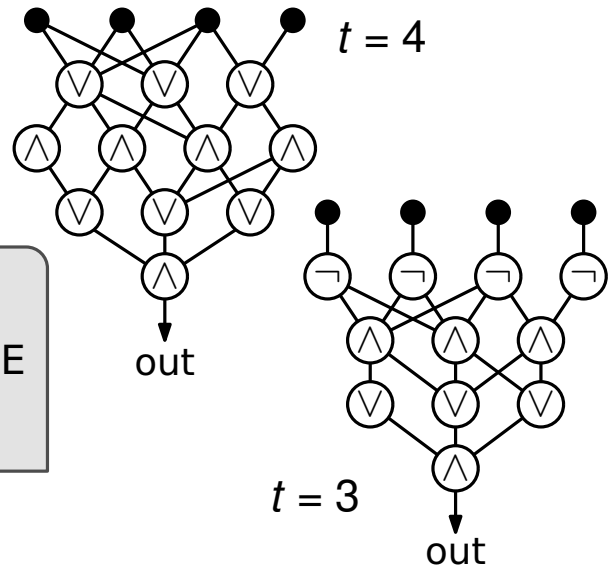

## Anmerkungen

- WMNS[$t$] und WANS[$t$] sind offensichtlich in $W[t]$ enthalten
- die Schwierigkeit besteht darin $W[t]$-Schwere zu zeigen
- wir können jetzt also von WMNS[$t$] bzw. WANS[$t$] reduzieren um $W[t]$-Schwere zu zeigen

# Normalisierte Schaltkreise

**Theorem**
Für jedes gerade $t \geq 2$ ist Weighted Monotone $t$-Normalized Satisfiability $W[t]$-vollständig.

**Theorem**
Für jedes ungerade $t \geq 3$ ist Weighted Antimonotone $t$-Normalized Satisfiability $W[t]$-vollständig.




## Anmerkungen

- WMNS[$t$] und WANS[$t$] sind offensichtlich in $W[t]$ enthalten
- die Schwierigkeit besteht darin $W[t]$-Schwere zu zeigen
- wir können jetzt also von WMNS[$t$] bzw. WANS[$t$] reduzieren um $W[t]$-Schwere zu zeigen
- Tendenz: wähle gerades $t$ für Minimierungs- und ungerades $t$ für Maximierungsprobleme

# Schlüsselkandidaten in Datenbanken

**Definition**
Eine Spaltenmenge $A$ ist eine **Unique Column Combination**, wenn die Einschränkung auf $A$ kein Tupel mehrfach enthält.

# Schlüsselkandidaten in Datenbanken

**Definition**
Eine Spaltenmenge $A$ ist eine **Unique Column Combination**, wenn die Einschränkung auf $A$ kein Tupel mehrfach enthält.

## Beispiel

| Vorname | Nachname | Geburtstag | Wohnort |
|---|---|---|---|
| Max | Mustermann | 1.6. | Karlsruhe |
| Bob | Mustermann | 7.10. | Berlin |
| Bob | Baumeister | 4.3. | Hamburg |
| Alice | Liddell | 4.5. | Wunderland |
| Alice | Musterfrau | 7.10. | Berlin |

# Schlüsselkandidaten in Datenbanken

**Definition**
Eine Spaltenmenge $A$ ist eine **Unique Column Combination**, wenn die Einschränkung auf $A$ kein Tupel mehrfach enthält.

## Beispiel

- keine einzelne Spalte ist unique

| Vorname | Nachname | Geburtstag | Wohnort |
|---|---|---|---|
| Max | Mustermann | 1.6. | Karlsruhe |
| Bob | Mustermann | 7.10. | Berlin |
| Bob | Baumeister | 4.3. | Hamburg |
| Alice | Liddell | 4.5. | Wunderland |
| Alice | Musterfrau | 7.10. | Berlin |

# Schlüsselkandidaten in Datenbanken

**Definition**
Eine Spaltenmenge $A$ ist eine **Unique Column Combination**, wenn die Einschränkung auf $A$ kein Tupel mehrfach enthält.

## Beispiel

- keine einzelne Spalte ist unique
- $\{$Vorname, Nachname$\}$ ist unique

| Vorname | Nachname | Geburtstag | Wohnort |
|---|---|---|---|
| Max | Mustermann | 1.6. | Karlsruhe |
| Bob | Mustermann | 7.10. | Berlin |
| Bob | Baumeister | 4.3. | Hamburg |
| Alice | Liddell | 4.5. | Wunderland |
| Alice | Musterfrau | 7.10. | Berlin |

# Schlüsselkandidaten in Datenbanken

**Definition**
Eine Spaltenmenge $A$ ist eine **Unique Column Combination**, wenn die Einschränkung auf $A$ kein Tupel mehrfach enthält.

## Beispiel

- keine einzelne Spalte ist unique
- {Vorname, Nachname} ist unique
- {Geburtstag, Wohnort} nicht

| Vorname | Nachname | Geburtstag | Wohnort |
|---|---|---|---|
| Max | Mustermann | 1.6. | Karlsruhe |
| Bob | Mustermann | 7.10. | Berlin |
| Bob | Baumeister | 4.3. | Hamburg |
| Alice | Liddell | 4.5. | Wunderland |
| Alice | Musterfrau | 7.10. | Berlin |

# Schlüsselkandidaten in Datenbanken

**Definition**
Eine Spaltenmenge $A$ ist eine **Unique Column Combination**, wenn die Einschränkung auf $A$ kein Tupel mehrfach enthält.

**Beispiel**

- keine einzelne Spalte ist unique
- {Vorname, Nachname} ist unique
- {Geburtstag, Wohnort} nicht

| Vorname | Nachname | Geburtstag | Wohnort |
|---|---|---|---|
| Max | Mustermann | 1.6. | Karlsruhe |
| Bob | Mustermann | 7.10. | Berlin |
| Bob | Baumeister | 4.3. | Hamburg |
| Alice | Liddell | 4.5. | Wunderland |
| Alice | Musterfrau | 7.10. | Berlin |

**Problem: UNIQUE**
Gegeben eine Datenbank und ein Parameter $k$. Gibt es eine Unique Column Combination der Größe maximal $k$?

# Schlüsselkandidaten in Datenbanken

**Definition**
Eine Spaltenmenge $A$ ist eine **Unique Column Combination**, wenn die Einschränkung auf $A$ kein Tupel mehrfach enthält.

## Beispiel

- keine einzelne Spalte ist unique
- {Vorname, Nachname} ist unique
- {Geburtstag, Wohnort} nicht

| | Vorname | Nachname | Geburtstag | Wohnort |
|---|---|---|---|---|
| | Max | Mustermann | 1.6. | Karlsruhe |
| $r_2$ | Bob | Mustermann | 7.10. | Berlin |
| | Bob | Baumeister | 4.3. | Hamburg |
| | Alice | Liddell | 4.5. | Wunderland |
| $r_5$ | Alice | Musterfrau | 7.10. | Berlin |

**Problem: UNIQUE**
Gegeben eine Datenbank und ein Parameter $k$. Gibt es eine Unique Column Combination der Größe maximal $k$?

## Leicht andere Sichtweise

- Zeilenpaar $(r_2, r_5)$ $\Rightarrow$ **Vor-** oder **Nachname** muss enthalten sein

# Schlüsselkandidaten in Datenbanken

**Definition**
Eine Spaltenmenge $A$ ist eine **Unique Column Combination**, wenn die Einschränkung auf $A$ kein Tupel mehrfach enthält.

## Beispiel

- keine einzelne Spalte ist unique
- {Vorname, Nachname} ist unique
- {Geburtstag, Wohnort} nicht

| | Vorname | Nachname | Geburtstag | Wohnort |
|---|---|---|---|---|
| | Max | Mustermann | 1.6. | Karlsruhe |
| $r_2$ | Bob | Mustermann | 7.10. | Berlin |
| | Bob | Baumeister | 4.3. | Hamburg |
| | Alice | Liddell | 4.5. | Wunderland |
| $r_5$ | Alice | Musterfrau | 7.10. | Berlin |

**Problem: UNIQUE**
Gegeben eine Datenbank und ein Parameter $k$. Gibt es eine Unique Column Combination der Größe maximal $k$?

## Leicht andere Sichtweise

- Zeilenpaar $(r_2, r_5) \Rightarrow$ **Vor-** oder **Nachname** muss enthalten sein
- jedes Zeilenpaar $(r_i, r_j)$ liefert Spaltenmenge $A_{i,j}$ aus der mindestens ein Element gewählt werden muss

# Schlüsselkandidaten in Datenbanken

**Definition**
Eine Spaltenmenge $A$ ist eine **Unique Column Combination**, wenn die Einschränkung auf $A$ kein Tupel mehrfach enthält.

**Beispiel**

- keine einzelne Spalte ist unique
- {Vorname, Nachname} ist unique
- {Geburtstag, Wohnort} nicht

| | Vorname | Nachname | Geburtstag | Wohnort |
|---|---|---|---|---|
| | Max | Mustermann | 1.6. | Karlsruhe |
| $r_2$ | Bob | Mustermann | 7.10. | Berlin |
| | Bob | Baumeister | 4.3. | Hamburg |
| | Alice | Liddell | 4.5. | Wunderland |
| $r_5$ | Alice | Musterfrau | 7.10. | Berlin |

**Problem: UNIQUE**
Gegeben eine Datenbank und ein Parameter $k$. Gibt es eine Unique Column Combination der Größe maximal $k$?

**Leicht andere Sichtweise**

- Zeilenpaar $(r_2, r_5) \Rightarrow$ **Vor-** oder **Nachname** muss enthalten sein
- jedes Zeilenpaar $(r_i, r_j)$ liefert Spaltenmenge $A_{i,j}$ aus der mindestens ein Element gewählt werden muss

Was ist das für ein Problem?

# Schlüsselkandidaten in Datenbanken

**Definition**
Eine Spaltenmenge $A$ ist eine **Unique Column Combination**, wenn die Einschränkung auf $A$ kein Tupel mehrfach enthält.

**Beispiel**

- keine einzelne Spalte ist unique
- {Vorname, Nachname} ist unique
- {Geburtstag, Wohnort} nicht

| | Vorname | Nachname | Geburtstag | Wohnort |
|---|---|---|---|---|
| | Max | Mustermann | 1.6. | Karlsruhe |
| $r_2$ | Bob | Mustermann | 7.10. | Berlin |
| | Bob | Baumeister | 4.3. | Hamburg |
| | Alice | Liddell | 4.5. | Wunderland |
| $r_5$ | Alice | Musterfrau | 7.10. | Berlin |

**Problem: UNIQUE**
Gegeben eine Datenbank und ein Parameter $k$. Gibt es eine Unique Column Combination der Größe maximal $k$?

**Leicht andere Sichtweise**

- Zeilenpaar $(r_2, r_5) \Rightarrow$ **Vor-** oder **Nachname** muss enthalten sein
- jedes Zeilenpaar $(r_i, r_j)$ liefert Spaltenmenge $A_{i,j}$ aus der mindestens ein Element gewählt werden muss

Was ist das für ein Problem? $\rightarrow$ HITTING SET

# Unique ist $W[2]$-vollständig

## Gerade gesehen

- parametrisierte Reduktion von Unique auf Hitting Set
- Hitting Set ist $W[2]$-vollständig $\Rightarrow$ Unique $\in W[2]$

# Unique ist $W[2]$-vollständig

## Gerade gesehen

- parametrisierte Reduktion von Unique auf Hitting Set
- Hitting Set ist $W[2]$-vollständig $\Rightarrow$ Unique $\in W[2]$

## Zeige: Reduktion von Hitting Set auf Unique

Instanz von Hitting Set

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

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

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

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

$S_4 = \{b, c\}$

# Unique ist $W[2]$-vollständig

## Gerade gesehen

- parametrisierte Reduktion von Unique auf Hitting Set
- Hitting Set ist $W[2]$-vollständig $\Rightarrow$ Unique $\in W[2]$

## Zeige: Reduktion von Hitting Set auf Unique

Instanz von Hitting Set

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

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

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

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

$S_4 = \{b, c\}$

Instanz von Unique

**A B C D E**

- verwende eine Spalte pro Element

# Unique ist $W[2]$-vollständig

## Gerade gesehen

- parametrisierte Reduktion von Unique auf Hitting Set
- Hitting Set ist $W[2]$-vollständig $\Rightarrow$ Unique $\in W[2]$

## Zeige: Reduktion von Hitting Set auf Unique

Instanz von Hitting Set

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

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

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

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

$S_4 = \{b, c\}$

Instanz von Unique

| | A | B | C | D | E |
|---|---|---|---|---|---|
| $r_0$ | 0 | 0 | 0 | 0 | 0 |
| $r_1$ | 1 | 1 | 1 | 0 | 0 |

- verwende eine Spalte pro Element
- jedes $S_i$ muss von einem Zeilenpaar repräsentiert werden

# Unique ist $W[2]$-vollständig

## Gerade gesehen

- parametrisierte Reduktion von Unique auf Hitting Set
- Hitting Set ist $W[2]$-vollständig $\Rightarrow$ Unique $\in W[2]$

## Zeige: Reduktion von Hitting Set auf Unique

Instanz von Hitting Set

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

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

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

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

$S_4 = \{b, c\}$

Instanz von 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 |

- verwende eine Spalte pro Element
- jedes $S_i$ muss von einem Zeilenpaar repräsentiert werden
- Benutze die 0-Zeile $s_0$ für jedes Paar

# Unique ist $W[2]$-vollständig

## Gerade gesehen

- parametrisierte Reduktion von Unique auf Hitting Set
- Hitting Set ist $W[2]$-vollständig $\Rightarrow$ Unique $\in W[2]$

## Zeige: Reduktion von Hitting Set auf Unique

Instanz von Hitting Set

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

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

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

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

$S_4 = \{b, c\}$

- verwende eine Spalte pro Element
- jedes $S_i$ muss von einem Zeilenpaar repräsentiert werden
- Benutze die 0-Zeile $s_0$ für jedes Paar

Instanz von 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 |

# Unique ist $W[2]$-vollständig

## Gerade gesehen

- parametrisierte Reduktion von Unique auf Hitting Set
- Hitting Set ist $W[2]$-vollständig $\Rightarrow$ Unique $\in W[2]$

## Zeige: Reduktion von Hitting Set auf Unique

Instanz von Hitting Set

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

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

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

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

$S_4 = \{b, c\}$

- verwende eine Spalte pro Element
- jedes $S_i$ muss von einem Zeilenpaar repräsentiert werden
- Benutze die 0-Zeile $s_0$ für jedes Paar

Instanz von 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 |

# Unique ist $W[2]$-vollständig

## Gerade gesehen

- parametrisierte Reduktion von Unique auf Hitting Set
- Hitting Set ist $W[2]$-vollständig $\Rightarrow$ Unique $\in W[2]$

## Zeige: Reduktion von Hitting Set auf Unique

Instanz von Hitting Set

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

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

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

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

$S_4 = \{b, c\}$

Instanz von 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 |

- verwende eine Spalte pro Element
- jedes $S_i$ muss von einem Zeilenpaar repräsentiert werden
- Benutze die 0-Zeile $s_0$ für jedes Paar
- Paar $(r_0, r_i)$ stellt sicher, dass ein Element aus $S_i$ gewählt wird
- also: Lösung für Unique $\Rightarrow$ Lösung für Hitting Set

# Unique ist $W[2]$-vollständig

## Gerade gesehen

- parametrisierte Reduktion von Unique auf Hitting Set
- Hitting Set ist $W[2]$-vollständig $\Rightarrow$ Unique $\in W[2]$

## Zeige: Reduktion von Hitting Set auf Unique

Instanz von Hitting Set

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

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

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

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

$S_4 = \{b, c\}$

Instanz von 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 |

- verwende eine Spalte pro Element
- jedes $S_i$ muss von einem Zeilenpaar repräsentiert werden
- Benutze die 0-Zeile $s_0$ für jedes Paar
- Paar $(r_0, r_i)$ stellt sicher, dass ein Element aus $S_i$ gewählt wird
- also: Lösung für Unique $\Rightarrow$ Lösung für Hitting Set
- **Achtung:** andere Paare liefern auch Bedingungen

# Unique ist $W[2]$-vollständig

## Gerade gesehen

- parametrisierte Reduktion von Unique auf Hitting Set
- Hitting Set ist $W[2]$-vollständig $\Rightarrow$ Unique $\in W[2]$

## Zeige: Reduktion von Hitting Set auf Unique

Instanz von Hitting Set

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

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

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

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

$S_4 = \{b, c\}$

Instanz von 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 |

- verwende eine Spalte pro Element
- jedes $S_i$ muss von einem Zeilenpaar repräsentiert werden
- Benutze die 0-Zeile $s_0$ für jedes Paar
- Paar $(r_0, r_i)$ stellt sicher, dass ein Element aus $S_i$ gewählt wird
- also: Lösung für Unique $\Rightarrow$ Lösung für Hitting Set
- **Achtung:** andere Paare liefern auch Bedingungen
- aber: $(r_i, r_j)$ liefert $S_i \cup S_j$

# Unique ist $W[2]$-vollständig

## Gerade gesehen

- parametrisierte Reduktion von Unique auf Hitting Set
- Hitting Set ist $W[2]$-vollständig $\Rightarrow$ Unique $\in W[2]$

## Zeige: Reduktion von Hitting Set auf Unique

Instanz von Hitting Set

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

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

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

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

$S_4 = \{b, c\}$

Instanz von 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 |

- verwende eine Spalte pro Element
- jedes $S_i$ muss von einem Zeilenpaar repräsentiert werden
- Benutze die 0-Zeile $s_0$ für jedes Paar
- Paar $(r_0, r_i)$ stellt sicher, dass ein Element aus $S_i$ gewählt wird
- also: Lösung für Unique $\Rightarrow$ Lösung für Hitting Set
- **Achtung:** andere Paare liefern auch Bedingungen
- aber: $(r_i, r_j)$ liefert $S_i \cup S_j$
- $r_i$ von $r_0$ unterscheidbar $\Rightarrow$ $r_i$ von $r_j$ unterscheidbar
- also: Lösung für Hitting Set $\Rightarrow$ Lösung für Unique

# Unique ist $W[2]$-vollständig

## Gerade gesehen

- parametrisierte Reduktion von Unique auf Hitting Set
- Hitting Set ist $W[2]$-vollständig $\Rightarrow$ Unique $\in W[2]$

## Zeige: Reduktion von Hitting Set auf Unique

Instanz von Hitting Set

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

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

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

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

$S_4 = \{b, c\}$

Instanz von 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 |

- verwende eine Spalte pro Element
- jedes $S_i$ muss von einem Zeilenpaar repräsentiert werden
- Benutze die 0-Zeile $s_0$ für jedes Paar
- Paar $(r_0, r_i)$ stellt sicher, dass ein Element aus $S_i$ gewählt wird
- also: Lösung für Unique $\Rightarrow$ Lösung für Hitting Set
- **Achtung:** andere Paare liefern auch Bedingungen
- aber: $(r_i, r_j)$ liefert $S_i \cup S_j$
- $r_i$ von $r_0$ unterscheidbar $\Rightarrow$ $r_i$ von $r_j$ unterscheidbar
- also: Lösung für Hitting Set $\Rightarrow$ Lösung für Unique

$\Rightarrow$ Unique ist $W[2]$-schwer

# Funktionale Abhängigkeiten

## Beobachtung

- {Nachname} ist kein Unique
- der Nachname identifiziert also nicht das gesamte Tupel
- aber: der Nachname identifiziert den Wohnort

| Vorname | Nachname | Geburtstag | Wohnort |
|---|---|---|---|
| Max | Mustermann | 7.10. | Karlsruhe |
| Bob | Mustermann | 1.6. | Karlsruhe |
| Bob | Baumeister | 7.10. | Karlsruhe |
| Alice | Liddell | 4.5. | Wunderland |
| Alice | Musterfrau | 4.3. | Berlin |

# Funktionale Abhängigkeiten

## Beobachtung

- {Nachname} ist kein Unique
- der Nachname identifiziert also nicht das gesamte Tupel
- aber: der Nachname identifiziert den Wohnort

| Vorname | Nachname | Geburtstag | Wohnort |
|---|---|---|---|
| Max | Mustermann | 7.10. | Karlsruhe |
| Bob | Mustermann | 1.6. | Karlsruhe |
| Bob | Baumeister | 7.10. | Karlsruhe |
| Alice | Liddell | 4.5. | Wunderland |
| Alice | Musterfrau | 4.3. | Berlin |
| | $X$ | | $A$ |

**Definition** (Grundsätzliche Annahme: $A \notin X$)

Für eine Spaltenmenge $X$ und eine Spalte $A$ ist $X \to A$ eine **Funktionale Abh.**, wenn $r_i[A] \neq r_j[A] \Rightarrow r_i[X] \neq r_j[X]$ für alle Zeilenpaare $(r_i, r_j)$.

# Funktionale Abhängigkeiten

## Beobachtung

- {Nachname} ist kein Unique
- der Nachname identifiziert also nicht das gesamte Tupel
- aber: der Nachname identifiziert den Wohnort

| Vorname | Nachname | Geburtstag | Wohnort |
|---|---|---|---|
| Max | Mustermann | 7.10. | Karlsruhe |
| Bob | Mustermann | 1.6. | Karlsruhe |
| Bob | Baumeister | 7.10. | Karlsruhe |
| Alice | Liddell | 4.5. | Wunderland |
| Alice | Musterfrau | 4.3. | Berlin |
| | $X$ | | $A$ |

**Definition** (Grundsätzliche Annahme: $A \notin X$)
Für eine Spaltenmenge $X$ und eine Spalte $A$ ist $X \to A$ eine **Funktionale Abh.**, wenn $r_i[A] \neq r_j[A] \Rightarrow r_i[X] \neq r_j[X]$ für alle Zeilenpaare $(r_i, r_j)$.

## Welche der folgenden Abhängigkeiten gelten?

- {Vorname} $\to$ {Geburtstag}
- {Geburtstag} $\to$ {Vorname}
- {Geburtstag} $\to$ {Wohnort}
- {Vorname} $\to$ {Wohnort}
- {Vorname, Geburtstag} $\to$ {Wohnort}
- {Wohnort, Geburtstag} $\to$ {Vorname}

# Funktionale Abhängigkeiten

**Problem: FD**
Gegeben eine Datenbank und ein Parameter $k$. Gibt es eine funktionale Abhängigkeit $X \to A$ mit $|X| \leq k$?

# Funktionale Abhängigkeiten

**Problem: FD**
Gegeben eine Datenbank und ein Parameter $k$. Gibt es eine funktionale Abhängigkeit $X \to A$ mit $|X| \leq k$?

## Zeige $W[2]$-Schwere für FD

- reduziere von UNIQUE

# Funktionale Abhängigkeiten

**Problem: FD**
Gegeben eine Datenbank und ein Parameter $k$. Gibt es eine funktionale Abhängigkeit $X \to A$ mit $|X| \leq k$?

## Zeige $W[2]$-Schwere für FD

- reduziere von UNIQUE
- leichter, wenn im Problem FD die rechte Seite $A$ feststeht

**Problem: $\text{FD}_{\text{FIXED}}$**
Gegeben eine Datenbank, eine Spalte $A$ und ein Parameter $k$. Gibt es eine funktionale Abhängigkeit $X \to A$ mit $|X| \leq k$?

# Funktionale Abhängigkeiten

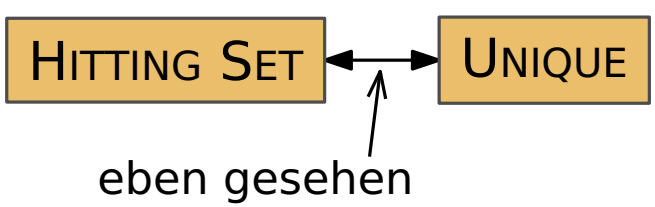
**Problem: FD**
Gegeben eine Datenbank und ein Parameter $k$. Gibt es eine funktionale Abhängigkeit $X \to A$ mit $|X| \leq k$?

## Zeige $W[2]$-Schwere für FD

- reduziere von UNIQUE
- leichter, wenn im Problem FD die rechte Seite $A$ feststeht

**Problem: $\text{FD}_{\text{FIXED}}$**
Gegeben eine Datenbank, eine Spalte $A$ und ein Parameter $k$. Gibt es eine funktionale Abhängigkeit $X \to A$ mit $|X| \leq k$?

## Reduktionen



# Funktionale Abhängigkeiten

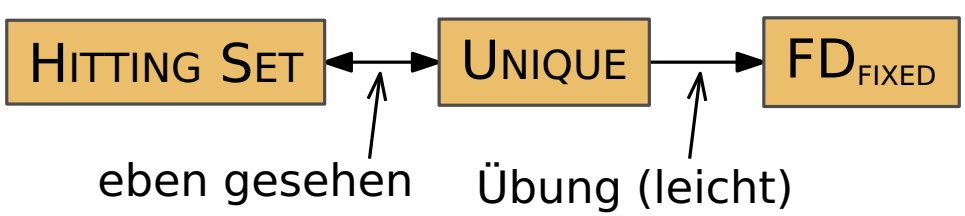
**Problem: FD**
Gegeben eine Datenbank und ein Parameter $k$. Gibt es eine funktionale Abhängigkeit $X \to A$ mit $|X| \leq k$?

## Zeige $W[2]$-Schwere für FD

- reduziere von UNIQUE
- leichter, wenn im Problem FD die rechte Seite $A$ feststeht

**Problem: $\text{FD}_{\text{FIXED}}$**
Gegeben eine Datenbank, eine Spalte $A$ und ein Parameter $k$. Gibt es eine funktionale Abhängigkeit $X \to A$ mit $|X| \leq k$?

## Reduktionen



# Funktionale Abhängigkeiten

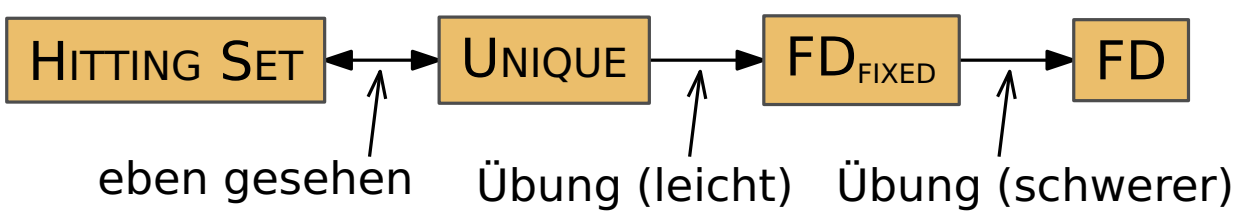
**Problem: FD**
Gegeben eine Datenbank und ein Parameter $k$. Gibt es eine funktionale Abhängigkeit $X \to A$ mit $|X| \leq k$?

## Zeige $W[2]$-Schwere für FD

- reduziere von UNIQUE
- leichter, wenn im Problem FD die rechte Seite $A$ feststeht

**Problem: $FD_{FIXED}$**
Gegeben eine Datenbank, eine Spalte $A$ und ein Parameter $k$. Gibt es eine funktionale Abhängigkeit $X \to A$ mit $|X| \leq k$?

## Reduktionen



# Funktionale Abhängigkeiten

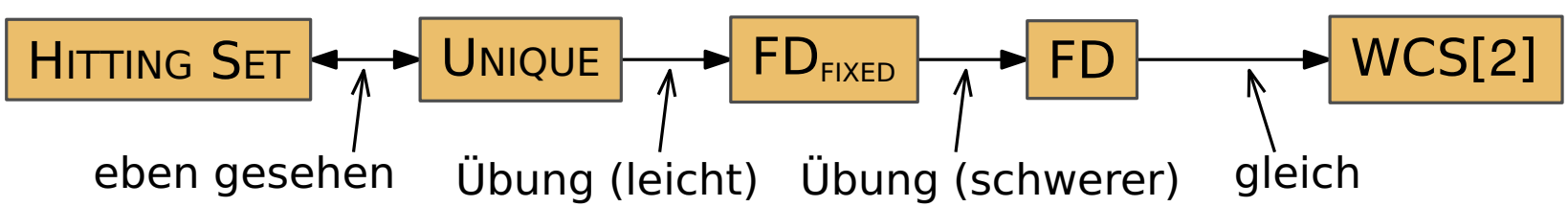
**Problem: FD**
Gegeben eine Datenbank und ein Parameter $k$. Gibt es eine funktionale Abhängigkeit $X \to A$ mit $|X| \leq k$?

## Zeige $W[2]$-Schwere für FD

- reduziere von UNIQUE
- leichter, wenn im Problem FD die rechte Seite $A$ feststeht

**Problem: $FD_{FIXED}$**
Gegeben eine Datenbank, eine Spalte $A$ und ein Parameter $k$. Gibt es eine funktionale Abhängigkeit $X \to A$ mit $|X| \leq k$?

## Reduktionen



# Funktionale Abhängigkeiten

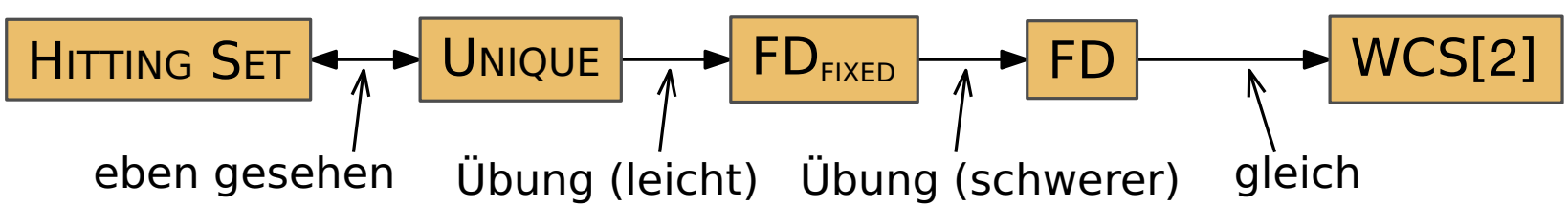
**Problem: FD**
Gegeben eine Datenbank und ein Parameter $k$. Gibt es eine funktionale Abhängigkeit $X \to A$ mit $|X| \leq k$?

## Zeige $W[2]$-Schwere für FD

- reduziere von UNIQUE
- leichter, wenn im Problem FD die rechte Seite $A$ feststeht

**Problem: $FD_{FIXED}$**
Gegeben eine Datenbank, eine Spalte $A$ und ein Parameter $k$. Gibt es eine funktionale Abhängigkeit $X \to A$ mit $|X| \leq k$?

## Reduktionen




- HITTING SET und WCS[2] sind $W[2]$-vollständig
  $\Rightarrow$ $FD_{FIXED}$ und FD sind $W[2]$-vollständig

# FD $\rightarrow$ WCS[2]

**Definition** (Grundsätzliche Annahme: $A \notin X$)
Für eine Spaltenmenge $X$ und eine Spalte $A$ ist $X \rightarrow A$ eine **Funktionale Abh.**, wenn $r_i[A] \neq r_j[A] \Rightarrow r_i[X] \neq r_j[X]$ für alle Zeilenpaare $(r_i, r_j)$.

## Grober Plan

- modelliere die Bedingungen von FD mittels Boolescher Formeln
- verwende für jede Bedingung nur $\vee$ und ein $\wedge$ am Ende (CNF)

  ($\Rightarrow$ Schaltkreis wird 2-normalisiert und damit Weft-2 (wenn auch nicht monoton))

# FD $\rightarrow$ WCS[2]

**Definition** (Grundsätzliche Annahme: $A \notin X$)
Für eine Spaltenmenge $X$ und eine Spalte $A$ ist $X \rightarrow A$ eine **Funktionale Abh.**, wenn $r_i[A] \neq r_j[A] \Rightarrow r_i[X] \neq r_j[X]$ für alle Zeilenpaare $(r_i, r_j)$.

## Grober Plan

- modelliere die Bedingungen von FD mittels Boolescher Formeln
- verwende für jede Bedingung nur $\vee$ und ein $\wedge$ am Ende (CNF)

  ($\Rightarrow$ Schaltkreis wird 2-normalisiert und damit Weft-2 (wenn auch nicht monoton))

## Was sind die Bedingungen? Welche Variablen benutzen wir?

# FD $\rightarrow$ WCS[2]

**Definition** (Grundsätzliche Annahme: $A \notin X$)
Für eine Spaltenmenge $X$ und eine Spalte $A$ ist $X \rightarrow A$ eine **Funktionale Abh.**, wenn $r_i[A] \neq r_j[A] \Rightarrow r_i[X] \neq r_j[X]$ für alle Zeilenpaare $(r_i, r_j)$.

## Grober Plan

- modelliere die Bedingungen von FD mittels Boolescher Formeln
- verwende für jede Bedingung nur $\vee$ und ein $\wedge$ am Ende (CNF)

  ($\Rightarrow$ Schaltkreis wird 2-normalisiert und damit Weft-2 (wenn auch nicht monoton))

## Was sind die Bedingungen? Welche Variablen benutzen wir?

- RHS besteht aus ein genau einer Spalte
- LHS und RHS sind disjunkte Spaltenmengen
- wenn $r_i[A] \neq r_j[A]$ für ein Zeilenpaar $(r_i, r_j)$ und $A \in$ RHS, dann:
  es gibt Spalte $B$ mit $r_i[B] \neq r_j[B]$ und $B \in$ LHS

# FD $\to$ WCS[2]

**Definition** (Grundsätzliche Annahme: $A \notin X$)
Für eine Spaltenmenge $X$ und eine Spalte $A$ ist $X \to A$ eine **Funktionale Abh.**, wenn $r_i[A] \neq r_j[A] \Rightarrow r_i[X] \neq r_j[X]$ für alle Zeilenpaare $(r_i, r_j)$.

## Grober Plan

- modelliere die Bedingungen von FD mittels Boolescher Formeln
- verwende für jede Bedingung nur $\vee$ und ein $\wedge$ am Ende (CNF)

  ($\Rightarrow$ Schaltkreis wird 2-normalisiert und damit Weft-2 (wenn auch nicht monoton))

## Was sind die Bedingungen? Welche Variablen benutzen wir?

- RHS besteht aus ein genau einer Spalte
- LHS und RHS sind disjunkte Spaltenmengen
- wenn $r_i[A] \neq r_j[A]$ für ein Zeilenpaar $(r_i, r_j)$ und $A \in$ RHS, dann:
  es gibt Spalte $B$ mit $r_i[B] \neq r_j[B]$ und $B \in$ LHS
- Indikatorvariablen $x_A$ und $x'_A$ für $A \in$ LHS bzw. $A \in$ RHS

# FD $\to$ WCS[2]

**Definition** (Grundsätzliche Annahme: $A \notin X$)
Für eine Spaltenmenge $X$ und eine Spalte $A$ ist $X \to A$ eine **Funktionale Abh.**, wenn $r_i[A] \neq r_j[A] \Rightarrow r_i[X] \neq r_j[X]$ für alle Zeilenpaare $(r_i, r_j)$.

## Grober Plan

- modelliere die Bedingungen von FD mittels Boolescher Formeln
- verwende für jede Bedingung nur $\vee$ und ein $\wedge$ am Ende (CNF)

  ($\Rightarrow$ Schaltkreis wird 2-normalisiert und damit Weft-2 (wenn auch nicht monoton))

## Was sind die Bedingungen? Welche Variablen benutzen wir?

- RHS besteht aus ein genau einer Spalte
- LHS und RHS sind disjunkte Spaltenmengen
- wenn $r_i[A] \neq r_j[A]$ für ein Zeilenpaar $(r_i, r_j)$ und $A \in$ RHS, dann:
  es gibt Spalte $B$ mit $r_i[B] \neq r_j[B]$ und $B \in$ LHS
- Indikatorvariablen $x_A$ und $x'_A$ für $A \in$ LHS bzw. $A \in$ RHS

**Wie verändert sich der Parameter?**

# FD $\rightarrow$ WCS[2]

**Definition** (Grundsätzliche Annahme: $A \notin X$)
Für eine Spaltenmenge $X$ und eine Spalte $A$ ist $X \rightarrow A$ eine **Funktionale Abh.**, wenn $r_i[A] \neq r_j[A] \Rightarrow r_i[X] \neq r_j[X]$ für alle Zeilenpaare $(r_i, r_j)$.

## Grober Plan

- modelliere die Bedingungen von FD mittels Boolescher Formeln
- verwende für jede Bedingung nur $\vee$ und ein $\wedge$ am Ende (CNF)

  ($\Rightarrow$ Schaltkreis wird 2-normalisiert und damit Weft-2 (wenn auch nicht monoton))

## Was sind die Bedingungen? Welche Variablen benutzen wir?

- RHS besteht aus ein genau einer Spalte
- LHS und RHS sind disjunkte Spaltenmengen
- wenn $r_i[A] \neq r_j[A]$ für ein Zeilenpaar $(r_i, r_j)$ und $A \in$ RHS, dann:
  es gibt Spalte $B$ mit $r_i[B] \neq r_j[B]$ und $B \in$ LHS
- Indikatorvariablen $x_A$ und $x'_A$ für $A \in$ LHS bzw. $A \in$ RHS

## RHS besteht aus genau einer Spalte

# FD $\to$ WCS[2]

**Definition** (Grundsätzliche Annahme: $A \notin X$)
Für eine Spaltenmenge $X$ und eine Spalte $A$ ist $X \to A$ eine **Funktionale Abh.**, wenn $r_i[A] \neq r_j[A] \Rightarrow r_i[X] \neq r_j[X]$ für alle Zeilenpaare $(r_i, r_j)$.

## Grober Plan

- modelliere die Bedingungen von FD mittels Boolescher Formeln
- verwende für jede Bedingung nur $\vee$ und ein $\wedge$ am Ende (CNF)

  ($\Rightarrow$ Schaltkreis wird 2-normalisiert und damit Weft-2 (wenn auch nicht monoton))

## Was sind die Bedingungen? Welche Variablen benutzen wir?

- RHS besteht aus ein genau einer Spalte
- LHS und RHS sind disjunkte Spaltenmengen
- wenn $r_i[A] \neq r_j[A]$ für ein Zeilenpaar $(r_i, r_j)$ und $A \in$ RHS, dann:
  es gibt Spalte $B$ mit $r_i[B] \neq r_j[B]$ und $B \in$ LHS
- Indikatorvariablen $x_A$ und $x'_A$ für $A \in$ LHS bzw. $A \in$ RHS

## RHS besteht aus genau einer Spalte

- mindestens eine Spalte: $\bigvee_{\text{Spalte } A} x'_A$

# FD $\to$ WCS[2]

**Definition** (Grundsätzliche Annahme: $A \notin X$)
Für eine Spaltenmenge $X$ und eine Spalte $A$ ist $X \to A$ eine **Funktionale Abh.**, wenn $r_i[A] \neq r_j[A] \Rightarrow r_i[X] \neq r_j[X]$ für alle Zeilenpaare $(r_i, r_j)$.

## Grober Plan

- modelliere die Bedingungen von FD mittels Boolescher Formeln
- verwende für jede Bedingung nur $\vee$ und ein $\wedge$ am Ende (CNF)

  ($\Rightarrow$ Schaltkreis wird 2-normalisiert und damit Weft-2 (wenn auch nicht monoton))

## Was sind die Bedingungen? Welche Variablen benutzen wir?

- RHS besteht aus ein genau einer Spalte
- LHS und RHS sind disjunkte Spaltenmengen
- wenn $r_i[A] \neq r_j[A]$ für ein Zeilenpaar $(r_i, r_j)$ und $A \in$ RHS, dann:
  es gibt Spalte $B$ mit $r_i[B] \neq r_j[B]$ und $B \in$ LHS
- Indikatorvariablen $x_A$ und $x'_A$ für $A \in$ LHS bzw. $A \in$ RHS

## RHS besteht aus genau einer Spalte

- mindestens eine Spalte: $\bigvee_{\text{Spalte } A} x'_A$
- nicht zwei Spalten $A$ und $B$ in der RHS: $\neg x'_A \vee \neg x'_B$

# FD $\to$ WCS[2]

**Definition** (Grundsätzliche Annahme: $A \notin X$)
Für eine Spaltenmenge $X$ und eine Spalte $A$ ist $X \to A$ eine **Funktionale Abh.**, wenn $r_i[A] \neq r_j[A] \Rightarrow r_i[X] \neq r_j[X]$ für alle Zeilenpaare $(r_i, r_j)$.

## Grober Plan

- modelliere die Bedingungen von FD mittels Boolescher Formeln
- verwende für jede Bedingung nur $\vee$ und ein $\wedge$ am Ende (CNF)

  ($\Rightarrow$ Schaltkreis wird 2-normalisiert und damit Weft-2 (wenn auch nicht monoton))

## Was sind die Bedingungen? Welche Variablen benutzen wir?

- RHS besteht aus ein genau einer Spalte
- LHS und RHS sind disjunkte Spaltenmengen
- wenn $r_i[A] \neq r_j[A]$ für ein Zeilenpaar $(r_i, r_j)$ und $A \in$ RHS, dann:
  es gibt Spalte $B$ mit $r_i[B] \neq r_j[B]$ und $B \in$ LHS
- Indikatorvariablen $x_A$ und $x'_A$ für $A \in$ LHS bzw. $A \in$ RHS

## LHS und RHS sind disjunkt

# FD $\to$ WCS[2]

**Definition** (Grundsätzliche Annahme: $A \notin X$)
Für eine Spaltenmenge $X$ und eine Spalte $A$ ist $X \to A$ eine **Funktionale Abh.**, wenn $r_i[A] \neq r_j[A] \Rightarrow r_i[X] \neq r_j[X]$ für alle Zeilenpaare $(r_i, r_j)$.

## Grober Plan

- modelliere die Bedingungen von FD mittels Boolescher Formeln
- verwende für jede Bedingung nur $\vee$ und ein $\wedge$ am Ende (CNF)

  ($\Rightarrow$ Schaltkreis wird 2-normalisiert und damit Weft-2 (wenn auch nicht monoton))

## Was sind die Bedingungen? Welche Variablen benutzen wir?

- RHS besteht aus ein genau einer Spalte
- LHS und RHS sind disjunkte Spaltenmengen
- wenn $r_i[A] \neq r_j[A]$ für ein Zeilenpaar $(r_i, r_j)$ und $A \in$ RHS, dann:
  es gibt Spalte $B$ mit $r_i[B] \neq r_j[B]$ und $B \in$ LHS
- Indikatorvariablen $x_A$ und $x'_A$ für $A \in$ LHS bzw. $A \in$ RHS

## LHS und RHS sind disjunkt

- für jede Spalte $A$: $\neg x_A \vee \neg x'_A$

# FD $\rightarrow$ WCS[2]

**Definition** (Grundsätzliche Annahme: $A \notin X$)
Für eine Spaltenmenge $X$ und eine Spalte $A$ ist $X \rightarrow A$ eine **Funktionale Abh.**, wenn $r_i[A] \neq r_j[A] \Rightarrow r_i[X] \neq r_j[X]$ für alle Zeilenpaare $(r_i, r_j)$.

## Grober Plan

- modelliere die Bedingungen von FD mittels Boolescher Formeln
- verwende für jede Bedingung nur $\vee$ und ein $\wedge$ am Ende (CNF)

  ($\Rightarrow$ Schaltkreis wird 2-normalisiert und damit Weft-2 (wenn auch nicht monoton))

## Was sind die Bedingungen? Welche Variablen benutzen wir?

- RHS besteht aus ein genau einer Spalte
- LHS und RHS sind disjunkte Spaltenmengen
- wenn $r_i[A] \neq r_j[A]$ für ein Zeilenpaar $(r_i, r_j)$ und $A \in$ RHS, dann:
  es gibt Spalte $B$ mit $r_i[B] \neq r_j[B]$ und $B \in$ LHS
- Indikatorvariablen $x_A$ und $x'_A$ für $A \in$ LHS bzw. $A \in$ RHS

## LHS und RHS bilden funktionale Abhängigkeit

# FD $\to$ WCS[2]

**Definition** (Grundsätzliche Annahme: $A \notin X$)
Für eine Spaltenmenge $X$ und eine Spalte $A$ ist $X \to A$ eine **Funktionale Abh.**, wenn $r_i[A] \neq r_j[A] \Rightarrow r_i[X] \neq r_j[X]$ für alle Zeilenpaare $(r_i, r_j)$.

## Grober Plan

- modelliere die Bedingungen von FD mittels Boolescher Formeln
- verwende für jede Bedingung nur $\vee$ und ein $\wedge$ am Ende (CNF)

  ($\Rightarrow$ Schaltkreis wird 2-normalisiert und damit Weft-2 (wenn auch nicht monoton))

## Was sind die Bedingungen? Welche Variablen benutzen wir?

- RHS besteht aus ein genau einer Spalte
- LHS und RHS sind disjunkte Spaltenmengen
- wenn $r_i[A] \neq r_j[A]$ für ein Zeilenpaar $(r_i, r_j)$ und $A \in$ RHS, dann:
  es gibt Spalte $B$ mit $r_i[B] \neq r_j[B]$ und $B \in$ LHS
- Indikatorvariablen $x_A$ und $x'_A$ für $A \in$ LHS bzw. $A \in$ RHS

## LHS und RHS bilden funktionale Abhängigkeit

- für jede Spalte $A$, und jedes Paar $(r_i, r_j)$ mit $r_i[A] \neq r_j[A]$:

$$\neg x'_A \vee \bigvee_{\substack{\text{Spalte } A \neq B \\ r_i[B] \neq r_j[B]}} x_B$$

# FD $\to$ WCS[2]

**Definition** (Grundsätzliche Annahme: $A \notin X$)
Für eine Spaltenmenge $X$ und eine Spalte $A$ ist $X \to A$ eine **Funktionale Abh.**, wenn $r_i[A] \neq r_j[A] \Rightarrow r_i[X] \neq r_j[X]$ für alle Zeilenpaare $(r_i, r_j)$.

## Grober Plan

- modelliere die Bedingungen von FD mittels Boolescher Formeln
- verwende für jede Bedingung nur $\vee$ und ein $\wedge$ am Ende (CNF)

  ($\Rightarrow$ Schaltkreis wird 2-normalisiert und damit Weft-2 (wenn auch nicht monoton))

## Was sind die Bedingungen? Welche Variablen benutzen wir?

- RHS besteht aus ein genau einer Spalte
- LHS und RHS sind disjunkte Spaltenmengen
- wenn $r_i[A] \neq r_j[A]$ für ein Zeilenpaar $(r_i, r_j)$ und $A \in$ RHS, dann:
  es gibt Spalte $B$ mit $r_i[B] \neq r_j[B]$ und $B \in$ LHS
- Indikatorvariablen $x_A$ und $x'_A$ für $A \in$ LHS bzw. $A \in$ RHS

**Theorem**
Die Probleme Unique, $\text{FD}_{\text{FIXED}}$ und FD sind $W[2]$-vollständig.

# Inclusion Dependency

**Ziel:** finde Daten in einer Datenbank, die in einer anderen enthalten sind

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 |

# Inclusion Dependency

**Ziel:** finde Daten in einer Datenbank, die in einer anderen enthalten sind

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 |

- betrachte Spalten $X = \{A, C\} \subseteq R$
- und Abbildung $\sigma : X \to S$ mit $\sigma(A) = G$ und $\sigma(C) = E$

# Inclusion Dependency

**Ziel:** finde Daten in einer Datenbank, die in einer anderen enthalten sind

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 |

- betrachte Spalten $X = \{A, C\} \subseteq R$
- und Abbildung $\sigma : X \to S$ mit $\sigma(A) = G$ und $\sigma(C) = E$
- Tupel für $X$: $\{11, 32, 23, 34\}$

$$\cap$$

- Tupel für $\sigma(X)$: $\{11, 13, 23, 32, 34\}$

# Inclusion Dependency

**Ziel:** finde Daten in einer Datenbank, die in einer anderen enthalten sind

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 |

- betrachte Spalten $X = \{A, C\} \subseteq R$
- und Abbildung $\sigma: X \to S$ mit $\sigma(A) = G$ und $\sigma(C) = E$
- Tupel für $X$: $\{11, 32, 23, 34\}$

$$\cap|$$

- Tupel für $\sigma(X)$: $\{11, 13, 23, 32, 34\}$

**Definition**
Betrachte die Instanzen $r$ und $s$ zweier Schemata $R$ bzw. $S$. Eine Teilmenge $X \subseteq R$ zusammen mit einer injektiven Abbildung $\sigma: X \to S$ ist eine **Inclusion Dependency**, wenn $r[X] \subseteq s[\sigma(X)]$.

# Inclusion Dependency

**Ziel:** finde Daten in einer Datenbank, die in einer anderen enthalten sind

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 |

- betrachte Spalten $X = \{A, C\} \subseteq R$
- und Abbildung $\sigma: X \to S$ mit $\sigma(A) = G$ und $\sigma(C) = E$
- Tupel für $X$: $\{11, 32, 23, 34\}$

$$\cap$$

- Tupel für $\sigma(X)$: $\{11, 13, 23, 32, 34\}$

**Definition**
Betrachte die Instanzen $r$ und $s$ zweier Schemata $R$ bzw. $S$. Eine Teilmenge $X \subseteq R$ zusammen mit einer injektiven Abbildung $\sigma: X \to S$ ist eine **Inclusion Dependency**, wenn $r[X] \subseteq s[\sigma(X)]$.

**Problem: IND**
Gegeben seien zwei Datenbanken $r$ und $s$ sowie ein Parameter $k$. Gibt es eine Inclusion Dependency der Größe $k$?

# Inclusion Dependency

**Ziel:** finde Daten in einer Datenbank, die in einer anderen enthalten sind

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 |

- betrachte Spalten $X = \{A, C\} \subseteq R$
- und Abbildung $\sigma: X \to S$ mit $\sigma(A) = G$ und $\sigma(C) = E$
- Tupel für $X$: $\{11, 32, 23, 34\}$

$$\cap|$$

- Tupel für $\sigma(X)$: $\{11, 13, 23, 32, 34\}$

**Definition**
Betrachte die Instanzen $r$ und $s$ zweier Schemata $R$ bzw. $S$. Eine Teilmenge $X \subseteq R$ zusammen mit einer injektiven Abbildung $\sigma: X \to S$ ist eine **Inclusion Dependency**, wenn $r[X] \subseteq s[\sigma(X)]$.

**Problem: IND**
Gegeben seien zwei Datenbanken $r$ und $s$ sowie ein Parameter $k$. Gibt es eine Inclusion Dependency der Größe $k$?

**Problem: $\text{IND}_{\text{FIXED}}$**
Gegeben seien $r$ und $s$ mit gleichem Schema und ein Parameter $k$. Gibt es eine Inclusion Dependency der Größe $k$ mit der Identität als $\sigma$?

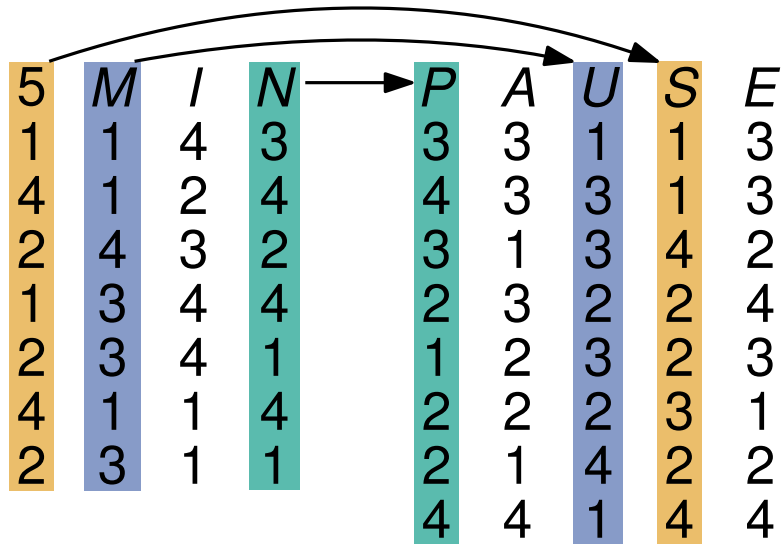
# Finde die größte Inclusion Dependency

| 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 |

| $P$ | $A$ | $U$ | $S$ | $E$ |
|---|---|---|---|---|
| 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**
Betrachte die Instanzen $r$ und $s$ zweier Schemata $R$ bzw. $S$. Eine Teilmenge $X \subseteq R$ zusammen mit einer injektiven Abbildung $\sigma\colon X \to S$ ist eine **Inclusion Dependency**, wenn $r[X] \subseteq s[\sigma(X)]$.

# Finde die größte Inclusion Dependency



| 5 | M | I | N | P | A | U | S | E |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 4 | 3 | 3 | 3 | 1 | 1 | 3 |
| 4 | 1 | 2 | 4 | 4 | 3 | 3 | 1 | 3 |
| 2 | 4 | 3 | 2 | 3 | 1 | 3 | 4 | 2 |
| 1 | 3 | 4 | 4 | 2 | 3 | 2 | 2 | 4 |
| 2 | 3 | 4 | 1 | 1 | 2 | 3 | 2 | 3 |
| 4 | 1 | 1 | 4 | 2 | 2 | 2 | 3 | 1 |
| 2 | 3 | 1 | 1 | 2 | 1 | 4 | 2 | 2 |
| | | | | 4 | 4 | 1 | 4 | 4 |

## Lösung

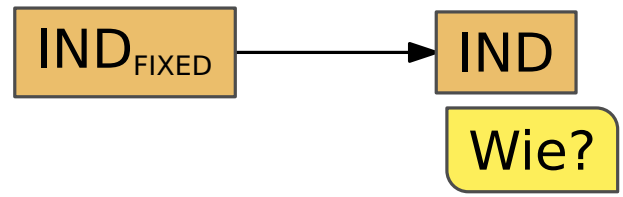
- $X = \{5, M, N\}$
- $\sigma(5) = S$, $\sigma(M) = U$ und $\sigma(N) = P$

**Definition**
Betrachte die Instanzen $r$ und $s$ zweier Schemata $R$ bzw. $S$. Eine Teilmenge $X \subseteq R$ zusammen mit einer injektiven Abbildung $\sigma\colon X \to S$ ist eine **Inclusion Dependency**, wenn $r[X] \subseteq s[\sigma(X)]$.

# IND ist $W[3]$-Vollständig

## $\text{IND}_{\text{FIXED}}$ und IND

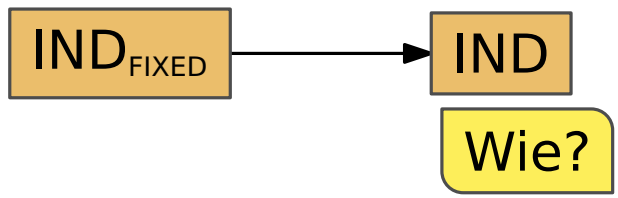
- Reduktion von $\text{IND}_{\text{FIXED}}$ auf IND ist einfach



# IND ist $W$[3]-Vollständig

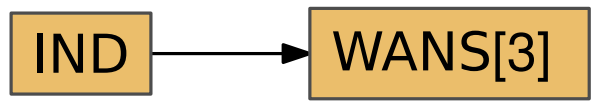
## $\mathbf{IND_{FIXED}}$ und IND

- Reduktion von $IND_{FIXED}$ auf IND ist einfach




## IND und WANS[3]

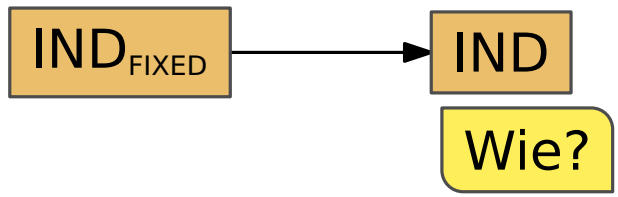
- Reduktion von IND auf WANS[3]



# IND ist $W[3]$-Vollständig

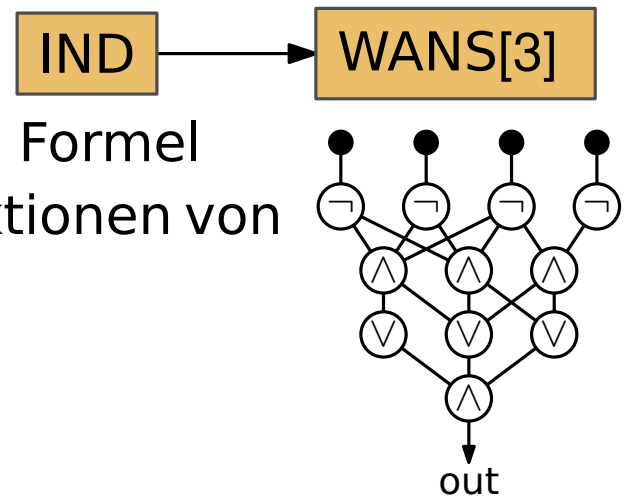
## $\text{IND}_{\text{FIXED}}$ und IND

- Reduktion von $\text{IND}_{\text{FIXED}}$ auf IND ist einfach




## IND und WANS[3]

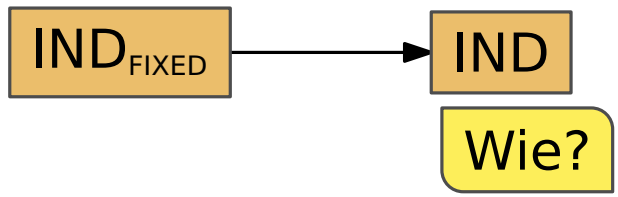
- Reduktion von IND auf WANS[3]
- modelliere die IND-Bedingungen als Boolesche Formel
- stelle sicher: es ist eine Konjunktion von Disjunktionen von Konjunktionen von negierten Variablen



# IND ist $W[3]$-Vollständig

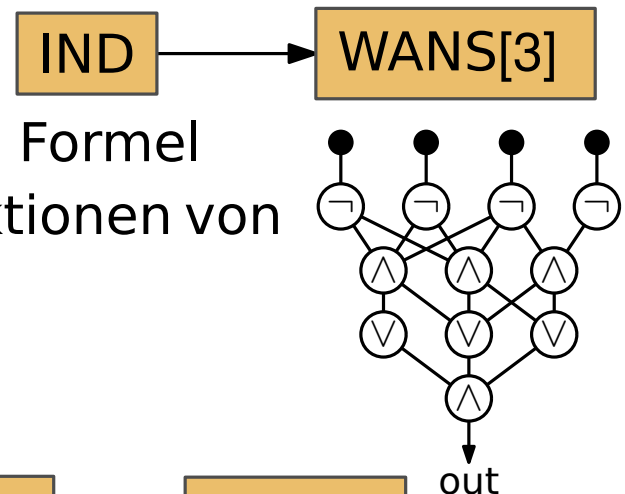
## $\text{IND}_{\text{FIXED}}$ und IND

- Reduktion von $\text{IND}_{\text{FIXED}}$ auf IND ist einfach



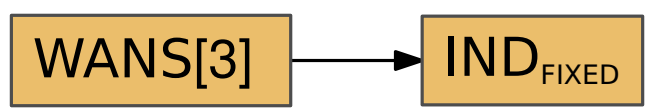

## IND und WANS[3]

- Reduktion von IND auf WANS[3]
- modelliere die IND-Bedingungen als Boolesche Formel
- stelle sicher: es ist eine Konjunktion von Disjunktionen von Konjunktionen von negierten Variablen




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

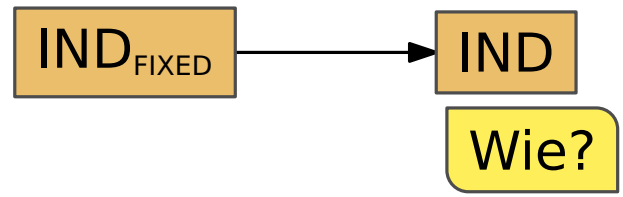
- Reduktion von WANS[3] auf $\text{IND}_{\text{FIXED}}$



# IND ist $W[3]$-Vollständig

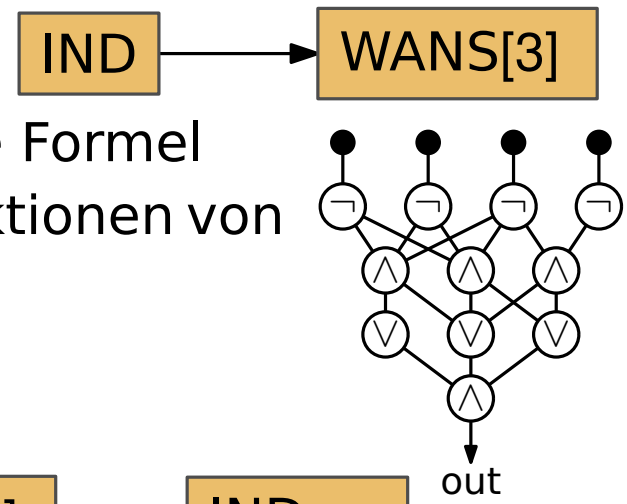
## $\text{IND}_{\text{FIXED}}$ und IND

- Reduktion von $\text{IND}_{\text{FIXED}}$ auf IND ist einfach



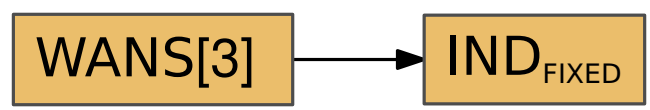

## IND und WANS[3]

- Reduktion von IND auf WANS[3]
- modelliere die IND-Bedingungen als Boolesche Formel
- stelle sicher: es ist eine Konjunktion von Disjunktionen von Konjunktionen von negierten Variablen




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

- Reduktion von WANS[3] auf $\text{IND}_{\text{FIXED}}$

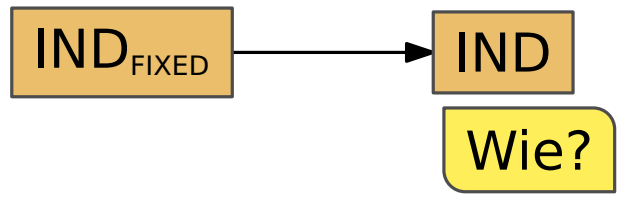



- modelliere antimonotone 3-normalisierte Boolesche Formeln mittels zweier Datenbanken

# IND ist $W[3]$-Vollständig

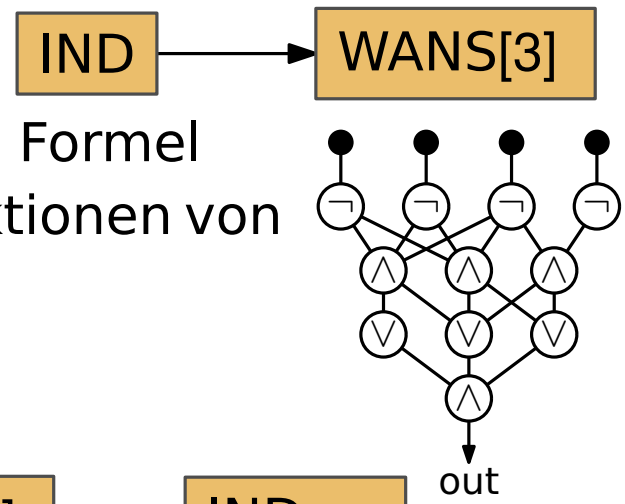
## $\text{IND}_{\text{FIXED}}$ und IND

- Reduktion von $\text{IND}_{\text{FIXED}}$ auf IND ist einfach



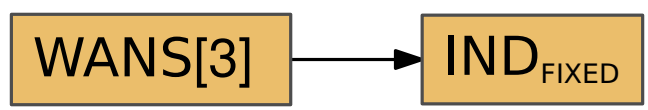

## IND und WANS[3]

- Reduktion von IND auf WANS[3]
- modelliere die IND-Bedingungen als Boolesche Formel
- stelle sicher: es ist eine Konjunktion von Disjunktionen von Konjunktionen von negierten Variablen



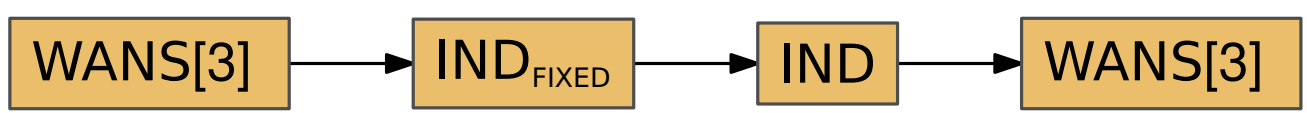

## $\text{WANS[3]}$ und $\text{IND}_{\text{FIXED}}$

- Reduktion von WANS[3] auf $\text{IND}_{\text{FIXED}}$




- modelliere antimonotone 3-normalisierte Boolesche Formeln mittels zweier Datenbanken

**Gesamtplan:**



# WANS[3] $\rightarrow$ $\mathbf{IND_{FIXED}}$

## $\mathbf{IND_{FIXED}}$ aufgefasst als Boolesche Funktion

- fasse eine Spaltenmenge als 01-String auf

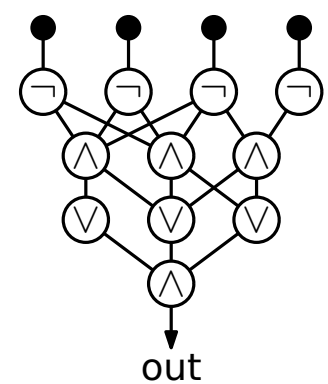
(Bsp: $\{A, C\} \widehat{=} 101$)

Datenbank $r$

| $A$ | $B$ | $C$ |
|---|---|---|
| 1 | 1 | 1 |
| 3 | 1 | 2 |
| 1 | 2 | 1 |
| 2 | 3 | 3 |
| 3 | 4 | 4 |

Datenbank $s$

| $A$ | $B$ | $C$ |
|---|---|---|
| 1 | 3 | 1 |
| 1 | 4 | 3 |
| 2 | 3 | 3 |
| 3 | 1 | 2 |
| 3 | 1 | 4 |
| 3 | 4 | 4 |



# WANS[3] $\rightarrow$ $\mathbf{IND_{FIXED}}$

## $\mathbf{IND_{FIXED}}$ aufgefasst als Boolesche Funktion

- fasse eine Spaltenmenge als 01-String auf

  (Bsp: $\{A, C\} \widehat{=} 101$)

- betrachte $f\colon \{0,1\}^n \rightarrow \{0,1\}$, mit $f(x) = 1 \Leftrightarrow x$ ist Inclusion Dependency für $(r, s)$

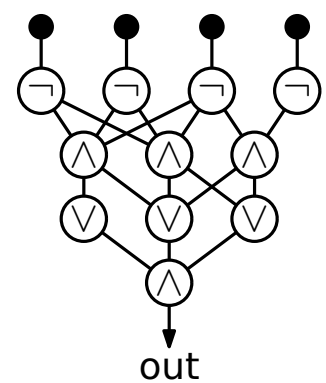
  (Bsp: $f(101) = 1$, $f(110) = 0$)

Datenbank $r$

| $A$ | $B$ | $C$ |
|---|---|---|
| 1 | 1 | 1 |
| 3 | 1 | 2 |
| 1 | 2 | 1 |
| 2 | 3 | 3 |
| 3 | 4 | 4 |

Datenbank $s$

| $A$ | $B$ | $C$ |
|---|---|---|
| 1 | 3 | 1 |
| 1 | 4 | 3 |
| 2 | 3 | 3 |
| 3 | 1 | 2 |
| 3 | 1 | 4 |
| 3 | 4 | 4 |



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

## $\text{IND}_{\text{FIXED}}$ aufgefasst als Boolesche Funktion

- fasse eine Spaltenmenge als 01-String auf

  (Bsp: $\{A, C\} \widehat{=} 101$)

- betrachte $f : \{0,1\}^n \rightarrow \{0,1\}$, mit $f(x) = 1 \Leftrightarrow x$ ist Inclusion Dependency für $(r, s)$

  (Bsp: $f(101) = 1$, $f(110) = 0$)

Datenbank $r$

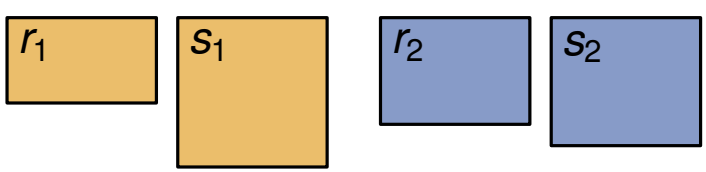
| $A$ | $B$ | $C$ |
|---|---|---|
| 1 | 1 | 1 |
| 3 | 1 | 2 |
| 1 | 2 | 1 |
| 2 | 3 | 3 |
| 3 | 4 | 4 |

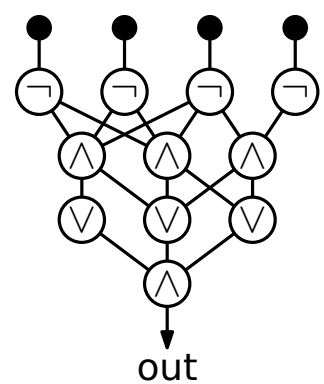
Datenbank $s$

| $A$ | $B$ | $C$ |
|---|---|---|
| 1 | 3 | 1 |
| 1 | 4 | 3 |
| 2 | 3 | 3 |
| 3 | 1 | 2 |
| 3 | 1 | 4 |
| 3 | 4 | 4 |

## Verundung von zwei Instanzen (über dem gleichen Schema)

- betrachte Instanzen $(r_1, s_1)$ und $(r_2, s_2)$ mit Funktionen $f_1$ bzw. $f_2$



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

## $\text{IND}_{\text{FIXED}}$ aufgefasst als Boolesche Funktion

- fasse eine Spaltenmenge als 01-String auf

  (Bsp: $\{A, C\} \widehat{=} 101$)
- betrachte $f\colon \{0,1\}^n \rightarrow \{0,1\}$, mit $f(x) = 1 \Leftrightarrow x$ ist Inclusion Dependency für $(r, s)$

  (Bsp: $f(101) = 1$, $f(110) = 0$)

Datenbank $r$

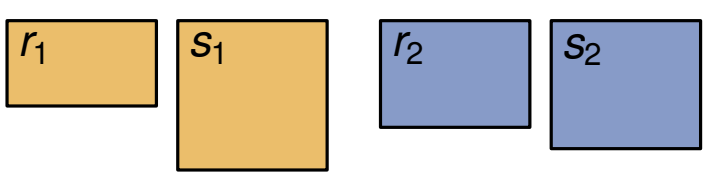
| $A$ | $B$ | $C$ |
|---|---|---|
| 1 | 1 | 1 |
| 3 | 1 | 2 |
| 1 | 2 | 1 |
| 2 | 3 | 3 |
| 3 | 4 | 4 |

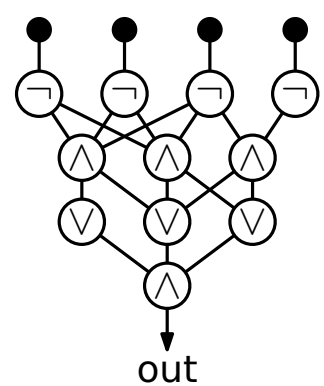
Datenbank $s$

| $A$ | $B$ | $C$ |
|---|---|---|
| 1 | 3 | 1 |
| 1 | 4 | 3 |
| 2 | 3 | 3 |
| 3 | 1 | 2 |
| 3 | 1 | 4 |
| 3 | 4 | 4 |

## Verundung von zwei Instanzen (über dem gleichen Schema)

- betrachte Instanzen $(r_1, s_1)$ und $(r_2, s_2)$ mit Funktionen $f_1$ bzw. $f_2$
- git es Instanz $(r, s)$, sodass $f = f_1 \wedge f_2$?



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

## $\text{IND}_{\text{FIXED}}$ aufgefasst als Boolesche Funktion

- fasse eine Spaltenmenge als 01-String auf

  (Bsp: $\{A, C\} \widehat{=} 101$)
- betrachte $f\colon \{0,1\}^n \rightarrow \{0,1\}$, mit $f(x) = 1 \Leftrightarrow x$ ist Inclusion Dependency für $(r, s)$

  (Bsp: $f(101) = 1$, $f(110) = 0$)

Datenbank $r$

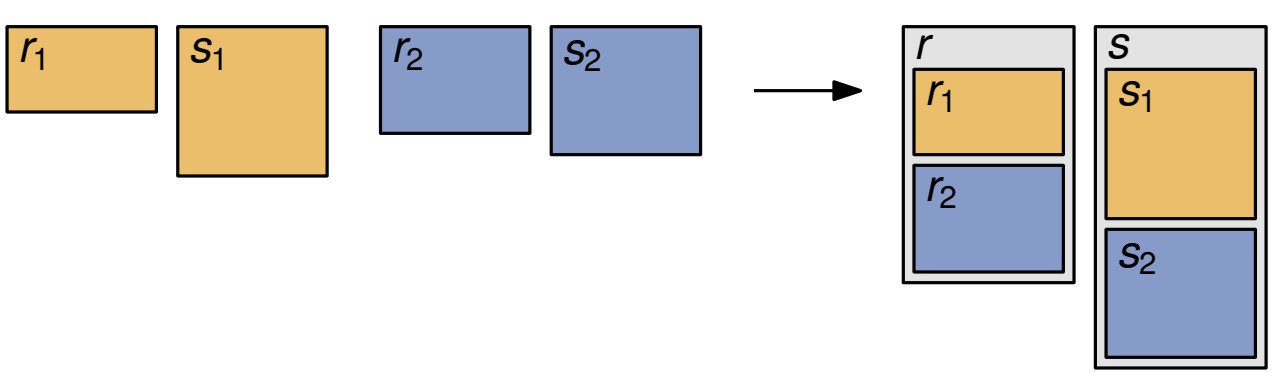
| $A$ | $B$ | $C$ |
|---|---|---|
| 1 | 1 | 1 |
| 3 | 1 | 2 |
| 1 | 2 | 1 |
| 2 | 3 | 3 |
| 3 | 4 | 4 |

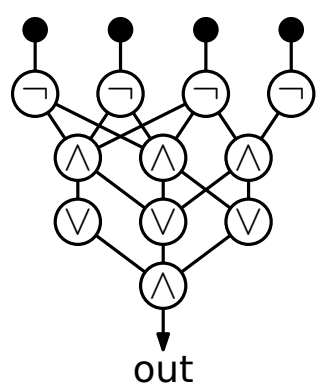
Datenbank $s$

| $A$ | $B$ | $C$ |
|---|---|---|
| 1 | 3 | 1 |
| 1 | 4 | 3 |
| 2 | 3 | 3 |
| 3 | 1 | 2 |
| 3 | 1 | 4 |
| 3 | 4 | 4 |

## Verundung von zwei Instanzen (über dem gleichen Schema)

- betrachte Instanzen $(r_1, s_1)$ und $(r_2, s_2)$ mit Funktionen $f_1$ bzw. $f_2$
- git es Instanz $(r, s)$, sodass $f = f_1 \wedge f_2$?
- Idee: sorge dafür, dass $(r_1, s_1)$ und $(r_2, s_2)$ disjunkte Werte haben und schreibe die Instanzen einfach untereinander



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

## $\text{IND}_{\text{FIXED}}$ aufgefasst als Boolesche Funktion

- fasse eine Spaltenmenge als 01-String auf
  (Bsp: $\{A, C\} \widehat{=} 101$)
- betrachte $f\colon \{0,1\}^n \rightarrow \{0,1\}$, mit $f(x) = 1 \Leftrightarrow x$ ist Inclusion Dependency für $(r, s)$
  (Bsp: $f(101) = 1$, $f(110) = 0$)

Datenbank $r$

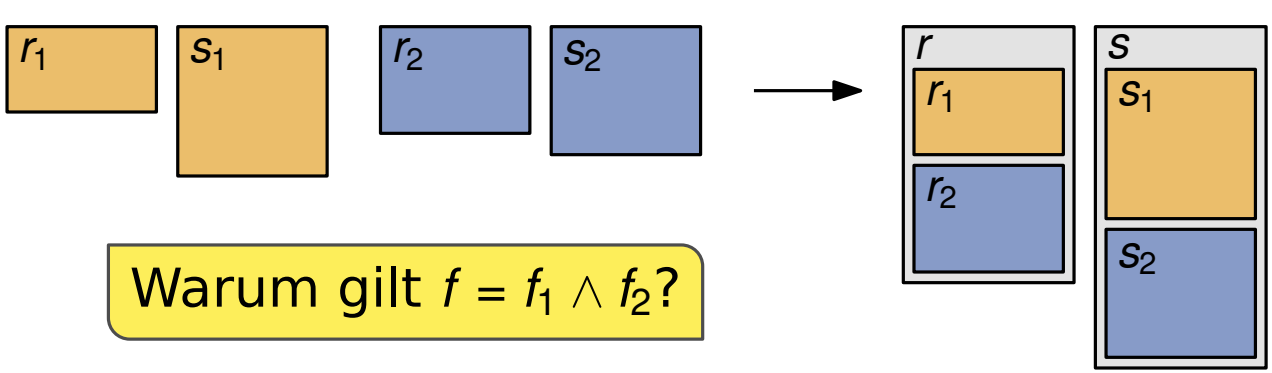
| $A$ | $B$ | $C$ |
|---|---|---|
| 1 | 1 | 1 |
| 3 | 1 | 2 |
| 1 | 2 | 1 |
| 2 | 3 | 3 |
| 3 | 4 | 4 |

Datenbank $s$

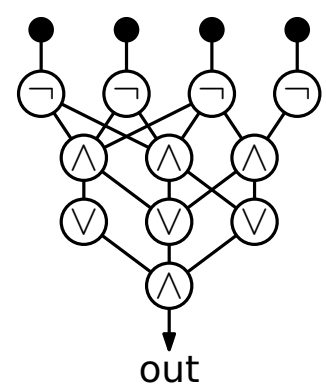
| $A$ | $B$ | $C$ |
|---|---|---|
| 1 | 3 | 1 |
| 1 | 4 | 3 |
| 2 | 3 | 3 |
| 3 | 1 | 2 |
| 3 | 1 | 4 |
| 3 | 4 | 4 |

## Verundung von zwei Instanzen (über dem gleichen Schema)

- betrachte Instanzen $(r_1, s_1)$ und $(r_2, s_2)$ mit Funktionen $f_1$ bzw. $f_2$
- git es Instanz $(r, s)$, sodass $f = f_1 \wedge f_2$?
- Idee: sorge dafür, dass $(r_1, s_1)$ und $(r_2, s_2)$ disjunkte Werte haben und schreibe die Instanzen einfach untereinander

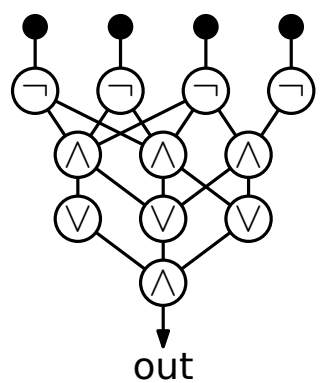



Warum gilt $f = f_1 \wedge f_2$?



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

**Lemma**
Gegeben zwei $\text{IND}_{\text{FIXED}}$-Instanzen mit Indikatorfunktionen $f_1$ und $f_2$, dann gibt es eine (nicht zu große) $\text{IND}_{\text{FIXED}}$-Instanz mit Indikatorfunktion $f_1 \wedge f_2$.

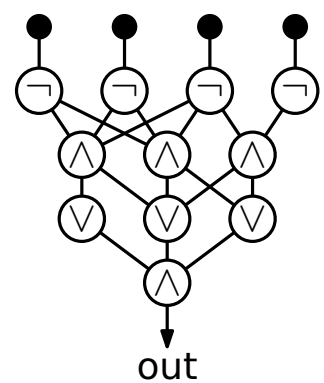


# WANS[3] $\rightarrow$ $\text{IND}_{\text{FIXED}}$

**Lemma**
Gegeben zwei $\text{IND}_{\text{FIXED}}$-Instanzen mit Indikatorfunktionen $f_1$ und $f_2$, dann gibt es eine (nicht zu große) $\text{IND}_{\text{FIXED}}$-Instanz mit Indikatorfunktion $f_1 \wedge f_2$.

## Zwischenstand

- Konjunktionen können wir also schon modellieren
- können wir auf ähnliche Art auch Disjunktionen modellieren?

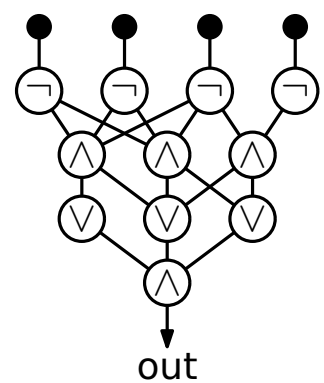


# WANS[3] $\rightarrow$ $\text{IND}_{\text{FIXED}}$

**Lemma**
Gegeben zwei $\text{IND}_{\text{FIXED}}$-Instanzen mit Indikatorfunktionen $f_1$ und $f_2$, dann gibt es eine (nicht zu große) $\text{IND}_{\text{FIXED}}$-Instanz mit Indikatorfunktion $f_1 \wedge f_2$.

## Zwischenstand

- Konjunktionen können wir also schon modellieren
- können wir auf ähnliche Art auch Disjunktionen modellieren?
- falls ja, dann wäre $\text{IND}_{\text{FIXED}}$ sogar $W[t]$-schwer, für beliebiges $t$



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

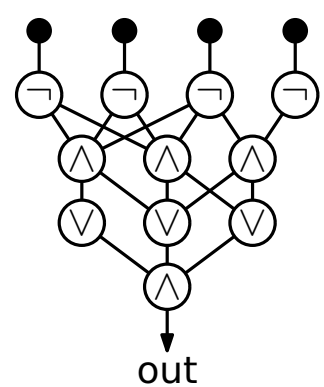
**Lemma**
Gegeben zwei $\text{IND}_{\text{FIXED}}$-Instanzen mit Indikatorfunktionen $f_1$ und $f_2$, dann gibt es eine (nicht zu große) $\text{IND}_{\text{FIXED}}$-Instanz mit Indikatorfunktion $f_1 \wedge f_2$.

## Zwischenstand

- Konjunktionen können wir also schon modellieren
- können wir auf ähnliche Art auch Disjunktionen modellieren?
- falls ja, dann wäre $\text{IND}_{\text{FIXED}}$ sogar $W[t]$-schwer, für beliebiges $t$

**Ziel:** modelliere Disjunktion von Konjunktionen von negierten Variablen

$\varphi = c_1 \vee c_2 \vee c_3$
$c_1 = (\neg x_1 \wedge \neg x_2 \wedge \neg x_3)$
$c_2 = (\neg x_2 \wedge \neg x_4 \wedge \neg x_5)$
$c_3 = (\neg x_1 \wedge \neg x_3 \wedge \neg x_4 \wedge \neg x_6)$



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

**Lemma**
Gegeben zwei $\text{IND}_{\text{FIXED}}$-Instanzen mit Indikatorfunktionen $f_1$ und $f_2$, dann gibt es eine (nicht zu große) $\text{IND}_{\text{FIXED}}$-Instanz mit Indikatorfunktion $f_1 \wedge f_2$.

## Zwischenstand

- Konjunktionen können wir also schon modellieren
- können wir auf ähnliche Art auch Disjunktionen modellieren?
- falls ja, dann wäre $\text{IND}_{\text{FIXED}}$ sogar $W[t]$-schwer, für beliebiges $t$

**Ziel:** modelliere Disjunktion von Konjunktionen von negierten Variablen

$\varphi = c_1 \vee c_2 \vee c_3$
$c_1 = (\neg x_1 \wedge \neg x_2 \wedge \neg x_3)$
$c_2 = (\neg x_2 \wedge \neg x_4 \wedge \neg x_5)$
$c_3 = (\neg x_1 \wedge \neg x_3 \wedge \neg x_4 \wedge \neg x_6)$

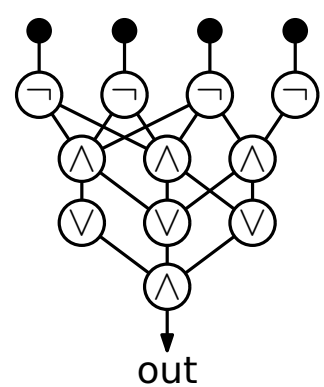
$r$:

| $A_1$ | $A_2$ | $A_3$ | $A_4$ | $A_5$ | $A_6$ |
|---|---|---|---|---|---|
| 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 |

- ein einzelnes $c_i$ darstellen ist leicht



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

**Lemma**
Gegeben zwei $\text{IND}_{\text{FIXED}}$-Instanzen mit Indikatorfunktionen $f_1$ und $f_2$, dann gibt es eine (nicht zu große) $\text{IND}_{\text{FIXED}}$-Instanz mit Indikatorfunktion $f_1 \wedge f_2$.

## Zwischenstand

- Konjunktionen können wir also schon modellieren
- können wir auf ähnliche Art auch Disjunktionen modellieren?
- falls ja, dann wäre $\text{IND}_{\text{FIXED}}$ sogar $W[t]$-schwer, für beliebiges $t$

**Ziel:** modelliere Disjunktion von Konjunktionen von negierten Variablen

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

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

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

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

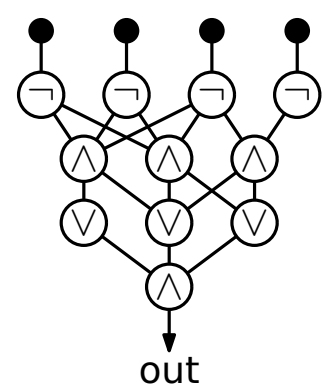
$r$:

| $A_1$ | $A_2$ | $A_3$ | $A_4$ | $A_5$ | $A_6$ |
|---|---|---|---|---|---|
| 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 |

- ein einzelnes $c_i$ darstellen ist leicht
- Veroderung der $c_i$: biete in $s$ jedes der $c_i$ einmal an



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

**Lemma**
Gegeben zwei $\text{IND}_{\text{FIXED}}$-Instanzen mit Indikatorfunktionen $f_1$ und $f_2$, dann gibt es eine (nicht zu große) $\text{IND}_{\text{FIXED}}$-Instanz mit Indikatorfunktion $f_1 \wedge f_2$.

## Zwischenstand

- Konjunktionen können wir also schon modellieren
- können wir auf ähnliche Art auch Disjunktionen modellieren?
- falls ja, dann wäre $\text{IND}_{\text{FIXED}}$ sogar $W[t]$-schwer, für beliebiges $t$

**Ziel:** modelliere Disjunktion von Konjunktionen von negierten Variablen

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

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

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

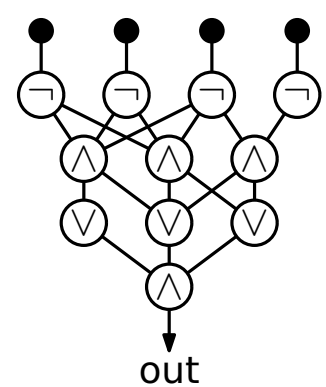
$c_3 = (\neg x_1 \wedge \neg x_3 \wedge \neg x_4 \wedge \neg x_6)$

$r$:

| $A_1$ | $A_2$ | $A_3$ | $A_4$ | $A_5$ | $A_6$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |

$s$:

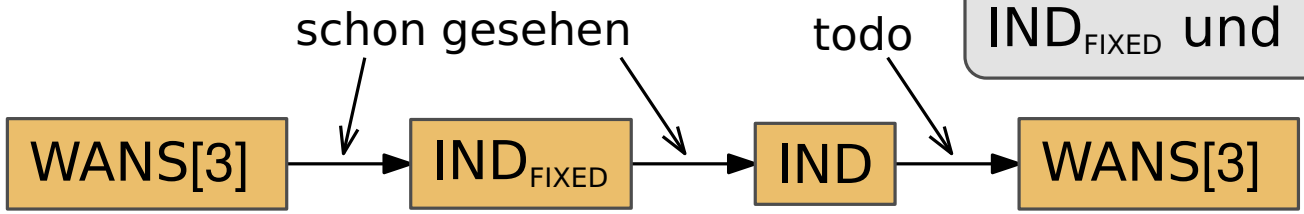
| $A_1$ | $A_2$ | $A_3$ | $A_4$ | $A_5$ | $A_6$ |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 |



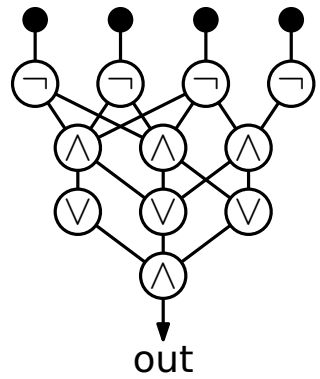

- ein einzelnes $c_i$ darstellen ist leicht
- Veroderung der $c_i$: biete in $s$ jedes der $c_i$ einmal an

**Check:** Lösung für $\varphi$ liefert Lösung für $(r, s)$ und umgekehrt

# Zwischenstand

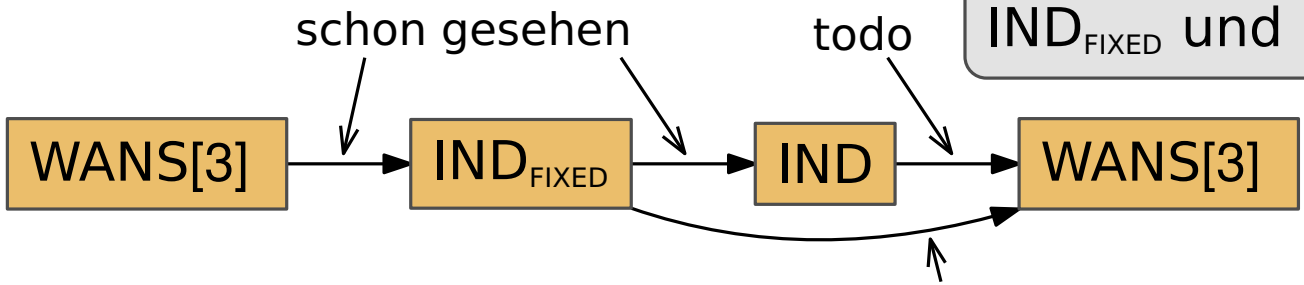



**Theorem**
IND$_{FIXED}$ und IND sind $W[3]$-schwer.

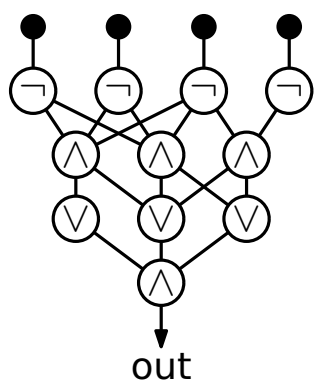


# Zwischenstand

**Theorem**
$\mathrm{IND}_{\mathrm{FIXED}}$ und IND sind $W[3]$-schwer.

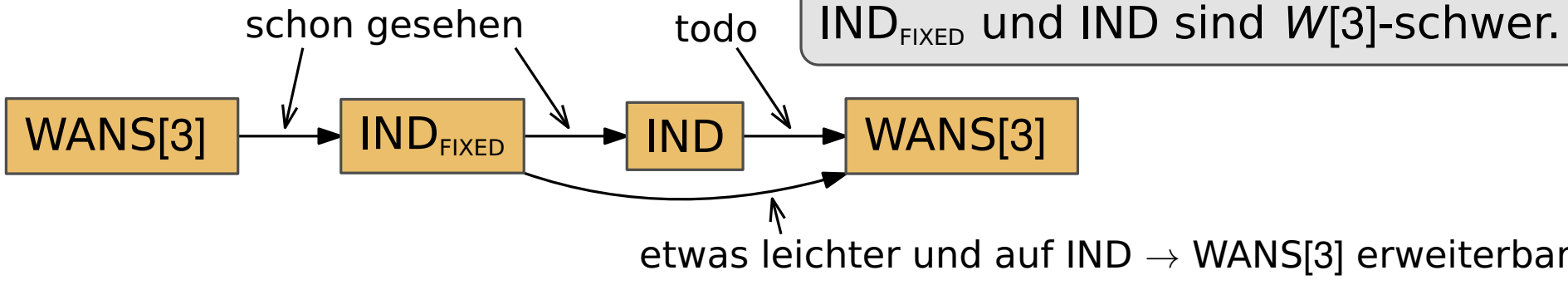



etwas leichter und auf IND $\rightarrow$ WANS[3] erweiterbar

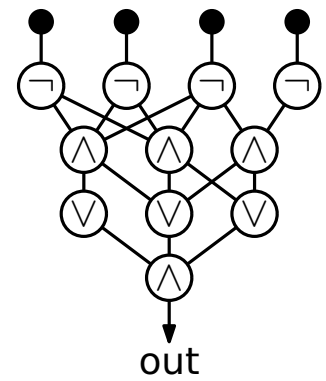


# Zwischenstand

**Theorem**
$\mathrm{IND}_{\mathrm{FIXED}}$ und IND sind $W[3]$-schwer.

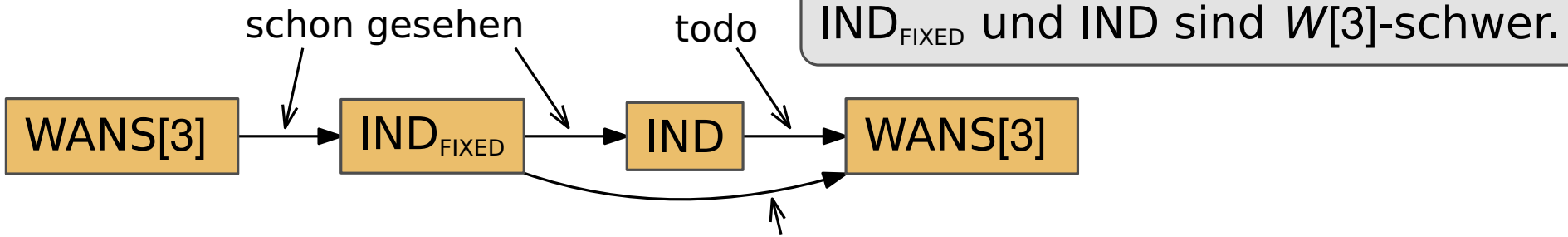



**Welche Bedingungen müssen wir modellieren?**



# Zwischenstand

**Theorem**
$\text{IND}_{\text{FIXED}}$ und IND sind $W[3]$-schwer.



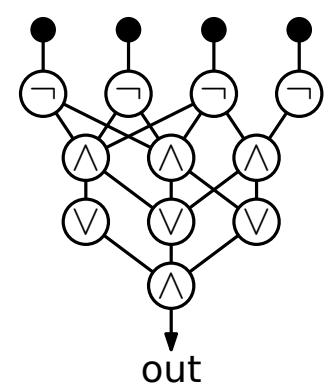

## Welche Bedingungen müssen wir modellieren?

- einzelnes Zeilenpaar $(r_i, s_j)$: wähle gewisse Elemente nicht

| | $A_1$ | $A_2$ | $A_3$ | $A_4$ |
|---|---|---|---|---|
| $r_j$ | 1 | 0 | 1 | 0 |

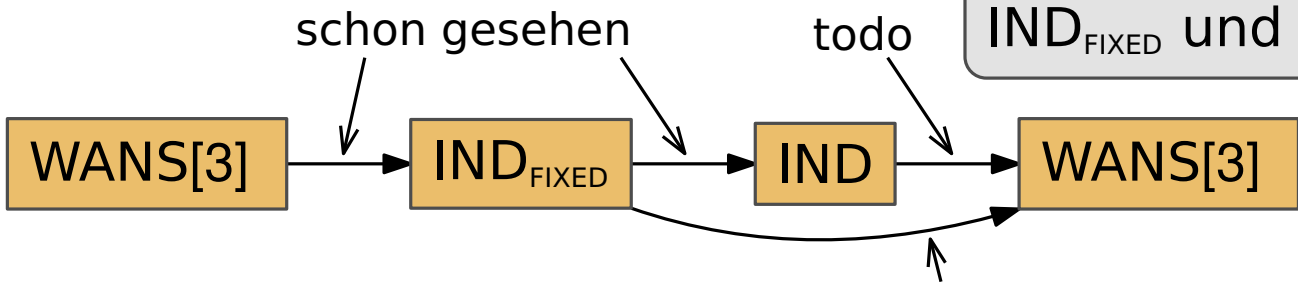
| | $A_1$ | $A_2$ | $A_3$ | $A_4$ |
|---|---|---|---|---|
| $s_j$ | 1 | 1 | 0 | 0 |

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



# Zwischenstand

**Theorem**
$\text{IND}_{\text{FIXED}}$ und IND sind $W[3]$-schwer.




etwas leichter und auf IND $\rightarrow$ WANS[3] erweiterbar

## Welche Bedingungen müssen wir modellieren?

- einzelnes Zeilenpaar $(r_i, s_j)$: wähle gewisse Elemente nicht

| | $A_1$ | $A_2$ | $A_3$ | $A_4$ |
|---|---|---|---|---|
| $r_j$ | 1 | 0 | 1 | 0 |

| | $A_1$ | $A_2$ | $A_3$ | $A_4$ |
|---|---|---|---|---|
| $s_j$ | 1 | 1 | 0 | 0 |

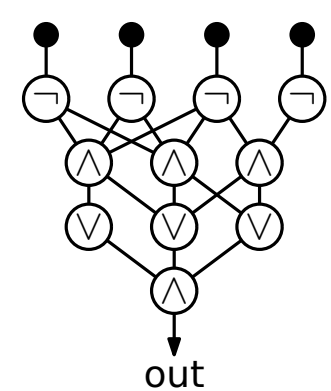
$\longrightarrow \quad \neg x_2 \wedge \neg x_3$

- $r_i$ und mehrere Zeilen in $s$: erfülle eine der vorherigen Bedingungen

| | $A_1$ | $A_2$ | $A_3$ | $A_4$ |
|---|---|---|---|---|
| $r_j$ | 1 | 0 | 1 | 0 |

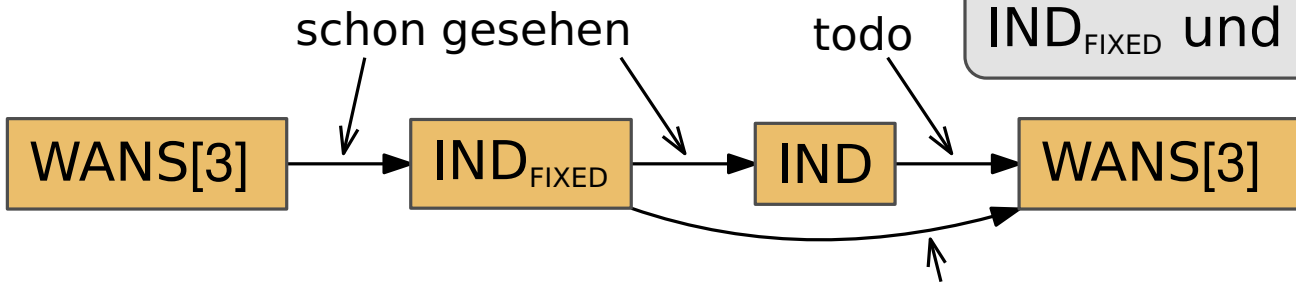
| | $A_1$ | $A_2$ | $A_3$ | $A_4$ |
|---|---|---|---|---|
| $s_1$ | 1 | 1 | 0 | 0 |
| $s_2$ | 0 | 1 | 1 | 0 |

$\longrightarrow \quad \overbrace{\neg x_2 \wedge \neg x_3}^{(r_j \mapsto s_1)} \vee \overbrace{\neg x_1 \wedge \neg x_2}^{(r_j \mapsto s_2)}$



# Zwischenstand

**Theorem**
$\text{IND}_{\text{FIXED}}$ und IND sind $W[3]$-schwer.




## Welche Bedingungen müssen wir modellieren?

- einzelnes Zeilenpaar $(r_i, s_j)$: wähle gewisse Elemente nicht

| | $A_1$ | $A_2$ | $A_3$ | $A_4$ |
|---|---|---|---|---|
| $r_j$ | 1 | 0 | 1 | 0 |

| | $A_1$ | $A_2$ | $A_3$ | $A_4$ |
|---|---|---|---|---|
| $s_j$ | 1 | 1 | 0 | 0 |

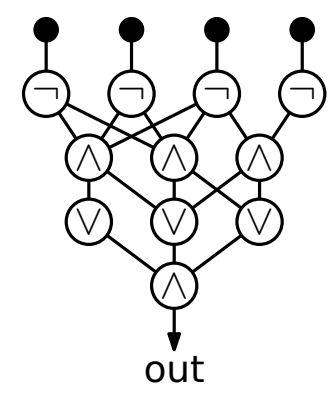
$\longrightarrow \quad \neg x_2 \wedge \neg x_3$

- $r_i$ und mehrere Zeilen in $s$: erfülle eine der vorherigen Bedingungen

| | $A_1$ | $A_2$ | $A_3$ | $A_4$ |
|---|---|---|---|---|
| $r_j$ | 1 | 0 | 1 | 0 |

| | $A_1$ | $A_2$ | $A_3$ | $A_4$ |
|---|---|---|---|---|
| $s_1$ | 1 | 1 | 0 | 0 |
| $s_2$ | 0 | 1 | 1 | 0 |

$\longrightarrow \quad \overbrace{\neg x_2 \wedge \neg x_3}^{(r_j \mapsto s_1)} \vee \overbrace{\neg x_1 \wedge \neg x_2}^{(r_j \mapsto s_2)}$




- mehrere Zeilen in $r$: erfülle Bedingungen für jede Zeile

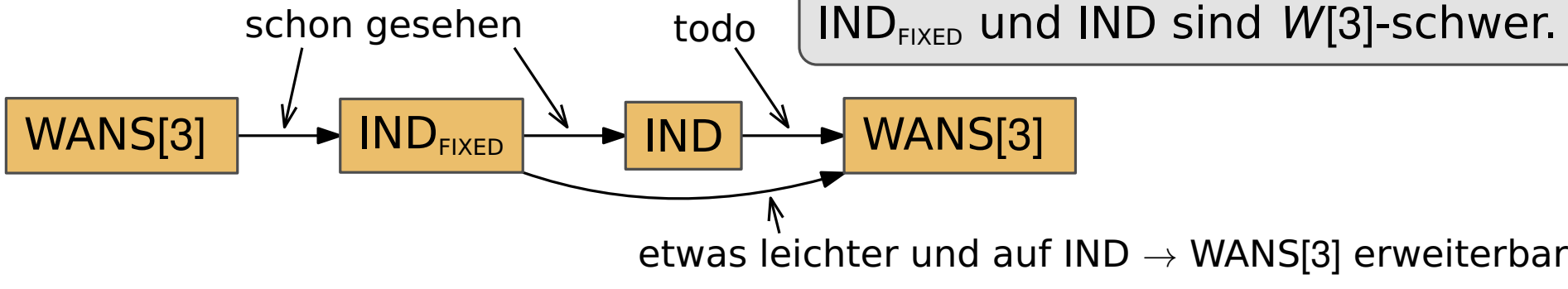
| | $A_1$ | $A_2$ | $A_3$ | $A_4$ |
|---|---|---|---|---|
| $r_1$ | 1 | 0 | 1 | 0 |
| $r_2$ | 0 | 1 | 0 | 1 |

| | $A_1$ | $A_2$ | $A_3$ | $A_4$ |
|---|---|---|---|---|
| $s_1$ | 1 | 1 | 0 | 0 |
| $s_2$ | 0 | 1 | 1 | 0 |

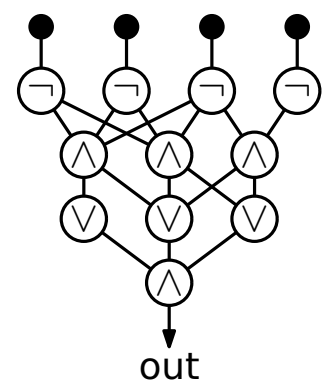
$$\longrightarrow \quad \Big(\overbrace{\neg x_2 \wedge \neg x_3}^{(r_1 \mapsto s_1)} \vee \overbrace{\neg x_1 \wedge \neg x_2}^{(r_1 \mapsto s_2)}\Big) \wedge \Big(\underbrace{\neg x_1 \wedge \neg x_4}_{(r_2 \mapsto s_1)} \vee \underbrace{\neg x_3 \wedge \neg x_4}_{(r_2 \mapsto s_2)}\Big)$$

# Zwischenstand

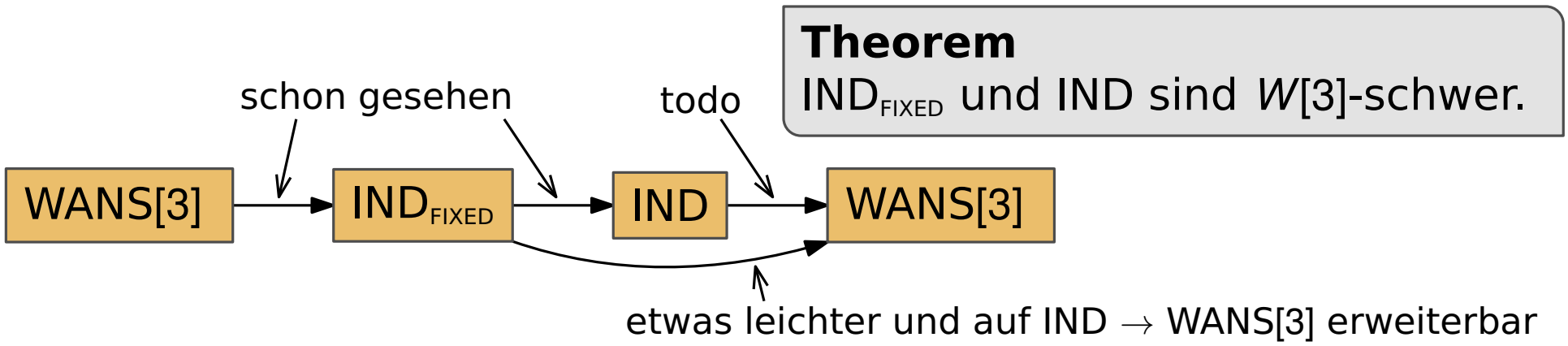
**Theorem**
$\mathrm{IND}_{\mathrm{FIXED}}$ und IND sind $W[3]$-schwer.




**Erweiterung auf IND → WANS[3]**

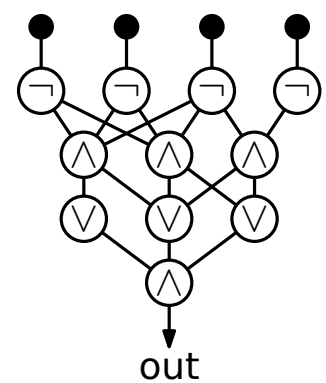


# Zwischenstand

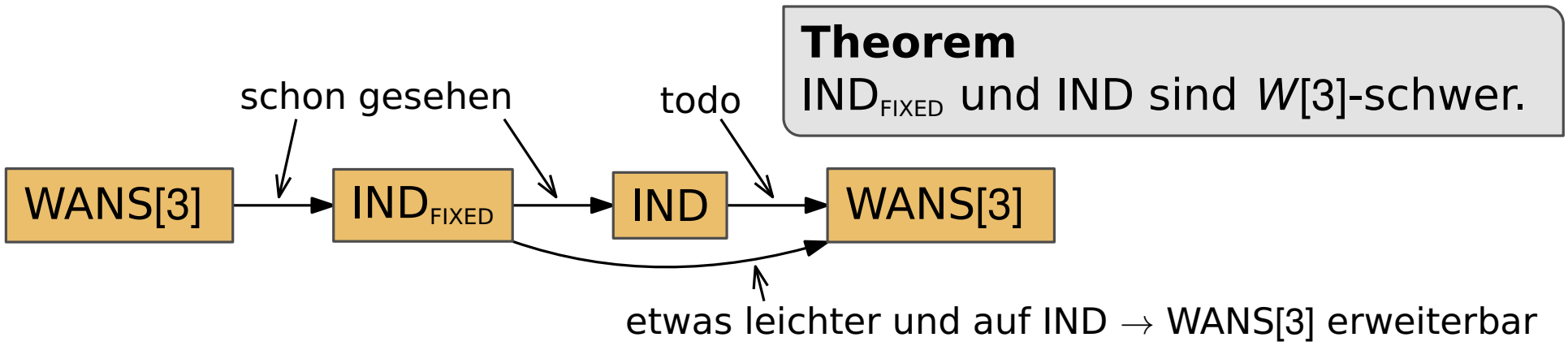



## Erweiterung auf IND → WANS[3]

- benutze Variablen $x_{i,j}$ mit der Bedeutung, dass $x_{i,j} = 1 \Leftrightarrow$ Spalte $i$ in $R$ wird ausgewählt und auf Spalte $j$ in $S$ abgebildet
- sorge dafür, dass man eine Bijektion erhält

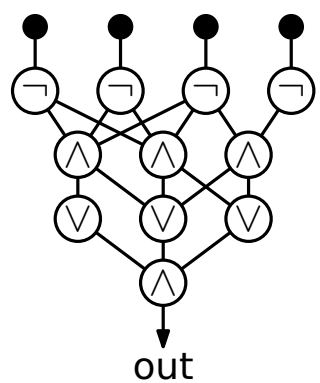


# Zwischenstand

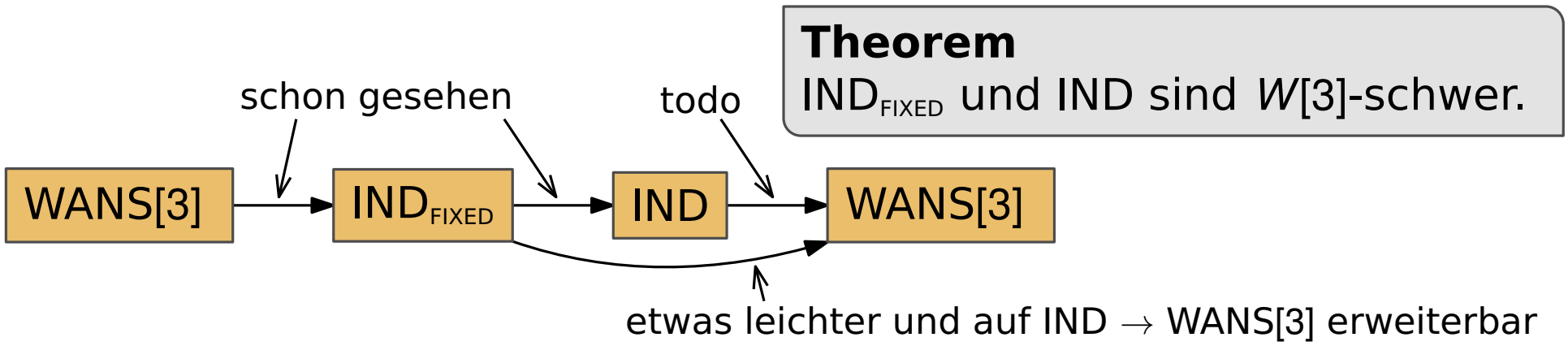



## Erweiterung auf IND → WANS[3]

- benutze Variablen $x_{i,j}$ mit der Bedeutung, dass $x_{i,j} = 1 \Leftrightarrow$ Spalte $i$ in $R$ wird ausgewählt und auf Spalte $j$ in $S$ abgebildet
- sorge dafür, dass man eine Bijektion erhält
- kombiniere das, mit den Bedingungen von vorher



# Zwischenstand

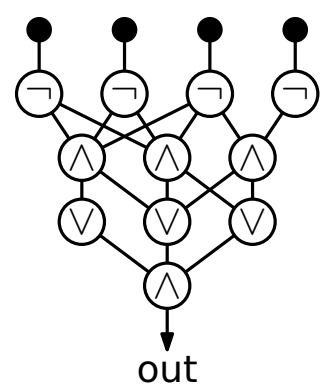



**Theorem**
IND$_{FIXED}$ und IND sind $W[3]$-schwer.

## Erweiterung auf IND → WANS[3]

- benutze Variablen $x_{i,j}$ mit der Bedeutung, dass $x_{i,j} = 1 \Leftrightarrow$ Spalte $i$ in $R$ wird ausgewählt und auf Spalte $j$ in $S$ abgebildet
- sorge dafür, dass man eine Bijektion erhält
- kombiniere das, mit den Bedingungen von vorher

**Theorem**
IND$_{FIXED}$ und IND sind $W[3]$-vollständig.



# Zusammenfassung

## Normalisierte Schaltkreise

- bei Reduktionen von WCS können wir mit recht aufgeräumten Schaltkreisen starten
- brauchen uns dann nicht mehr um Weft $t$ kümmern (stattdessen: Anzahl alternierender Ebenen)

# Zusammenfassung

## Normalisierte Schaltkreise

- bei Reduktionen von WCS können wir mit recht aufgeräumten Schaltkreisen starten
- brauchen uns dann nicht mehr um Weft $t$ kümmern (stattdessen: Anzahl alternierender Ebenen)
- gerades $t$: monoton; ungerades $t$: antimonoton

# Zusammenfassung

## Normalisierte Schaltkreise

- bei Reduktionen von WCS können wir mit recht aufgeräumten Schaltkreisen starten
- brauchen uns dann nicht mehr um Weft $t$ kümmern (stattdessen: Anzahl alternierender Ebenen)
- gerades $t$: monoton; ungerades $t$: antimonoton

## Dependency Detection

- es müssen nicht immer nur Graphen sein
- es gibt auch natürliche $W[3]$-vollständige Probleme

# Zusammenfassung

## Normalisierte Schaltkreise

- bei Reduktionen von WCS können wir mit recht aufgeräumten Schaltkreisen starten
- brauchen uns dann nicht mehr um Weft $t$ kümmern (stattdessen: Anzahl alternierender Ebenen)
- gerades $t$: monoton; ungerades $t$: antimonoton

## Dependency Detection

- es müssen nicht immer nur Graphen sein
- es gibt auch natürliche $W[3]$-vollständige Probleme
- erklärt nicht, warum funktionale Abhängigkeiten in der Praxis schnell ausgerechnet werden können

  (tatsächlich können sogar alle inklusionsminimalen UCCs/FDs aufgezählt werden)

# Literaturhinweise


**The Parameterized Complexity of Dependency Detection in Relational Databases**

- Thomas Bläsius, Tobias Friedrich, Martin Schirneck [2016]
- enthält die eben gesehenen Reduktionen

doi.org/10.4230/LIPIcs.IPEC.2016.6

**Profiling relational data: a survey**

- Ziawasch Abedjan, Lukasz Golab, Felix Naumann [2015]
- Überblick zum Finden von Abhängigkeiten in Datenbanken

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]
- in der Praxis effizienter Algo zum Aufzählen von Unique Colum 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]
- in der Praxis effizienter Algo zum Aufzählen von funktionalen Abhängigkeiten

doi.org/10.1145/3639298