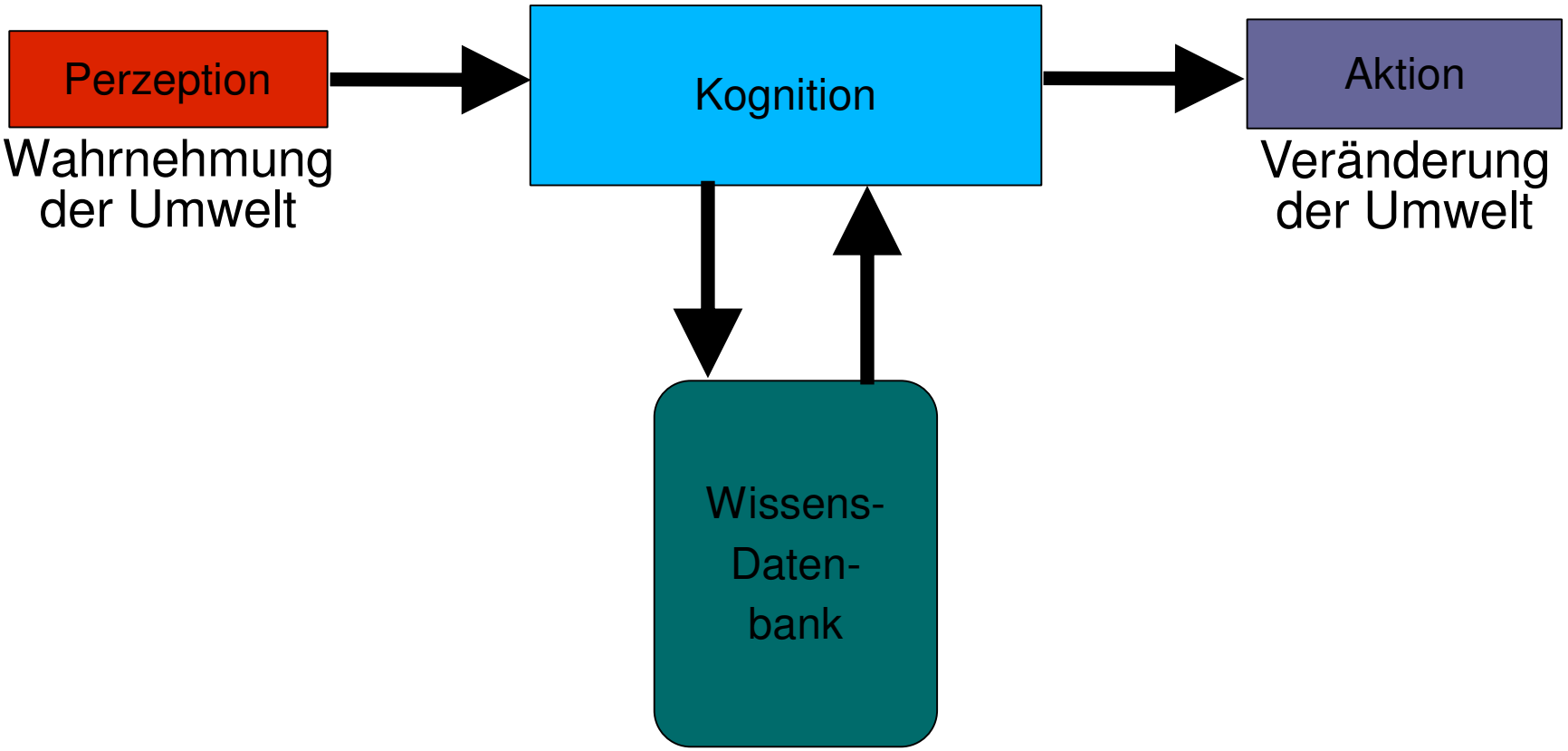
Kognitive Systeme

Wissen und Planung I

Montag, 5. Juli 2017

Elemente eines Kognitiven Systems

Ingenieursmäßiger Ansatz:



Gesichtserkennung

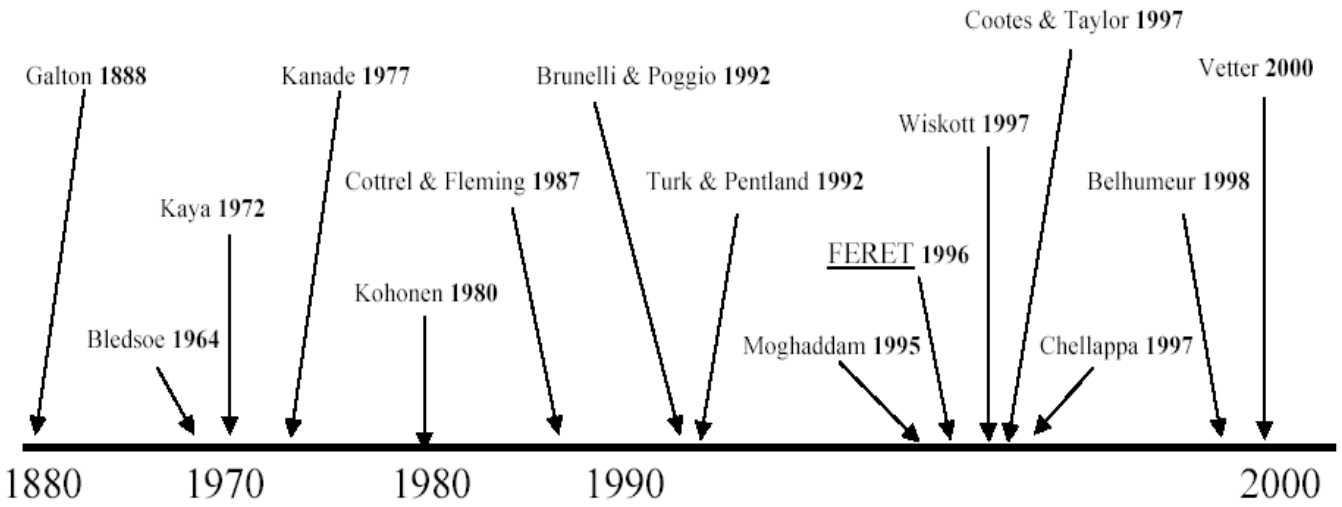
Gesichtserkennung

- Zu unterscheiden sind:
  - Gesichtsdetektion: Finden von Gesichtern (engl. *face detection*) (z.B. [Viola & Jones, 2001])
  - Identität der Person wird nicht bestimmt

- Im Folgenden:
  Gesichtserkennung = Identifikation von Gesichtern

  (die nachfolgenden Folien wurden von Prof. Stiefelhagen und Dr. Ekenel zur Verfügung gestellt)

Historischer Überblick



Features Period

Templates Period

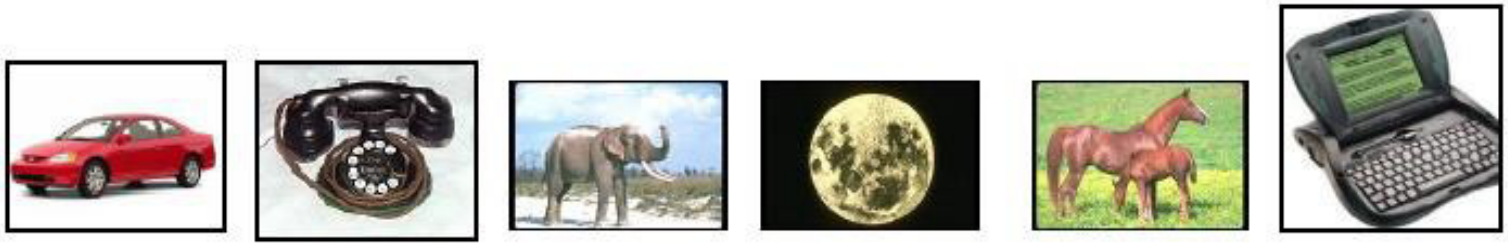
Neural Period

2D Subspace Models

3D Models

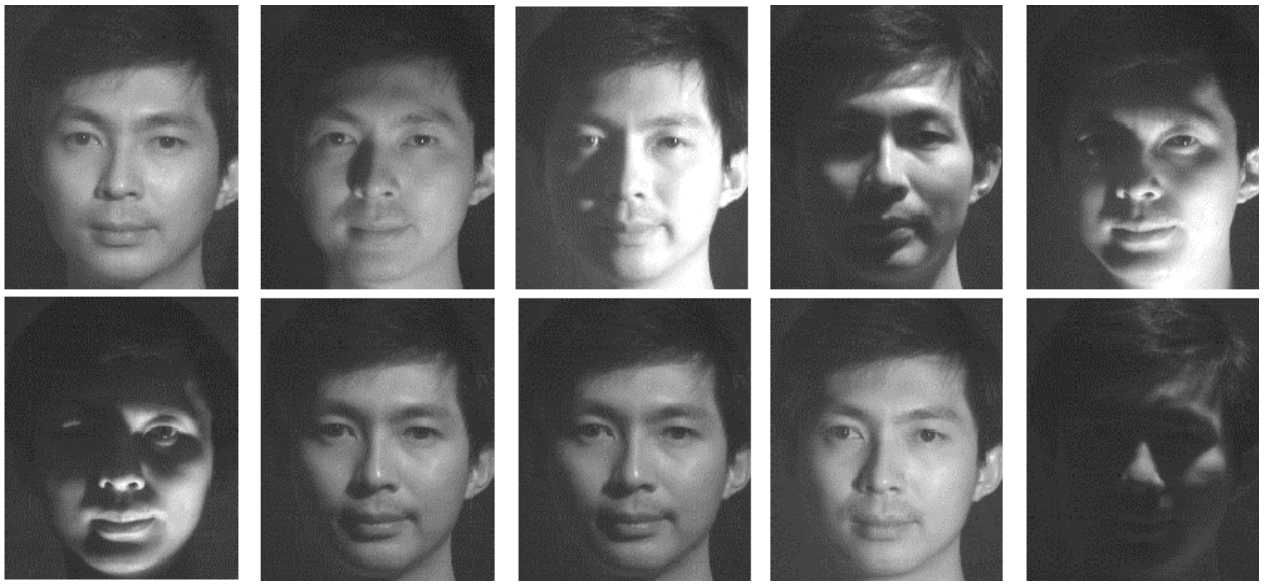
Gesichtserkennung vs. Objekterkennung

- Erkennung grundlegend unterschiedlicher Objekte und Objektklassen



- Erkennung verschiedener „Objekte“ innerhalb einer Klasse „Gesichter“

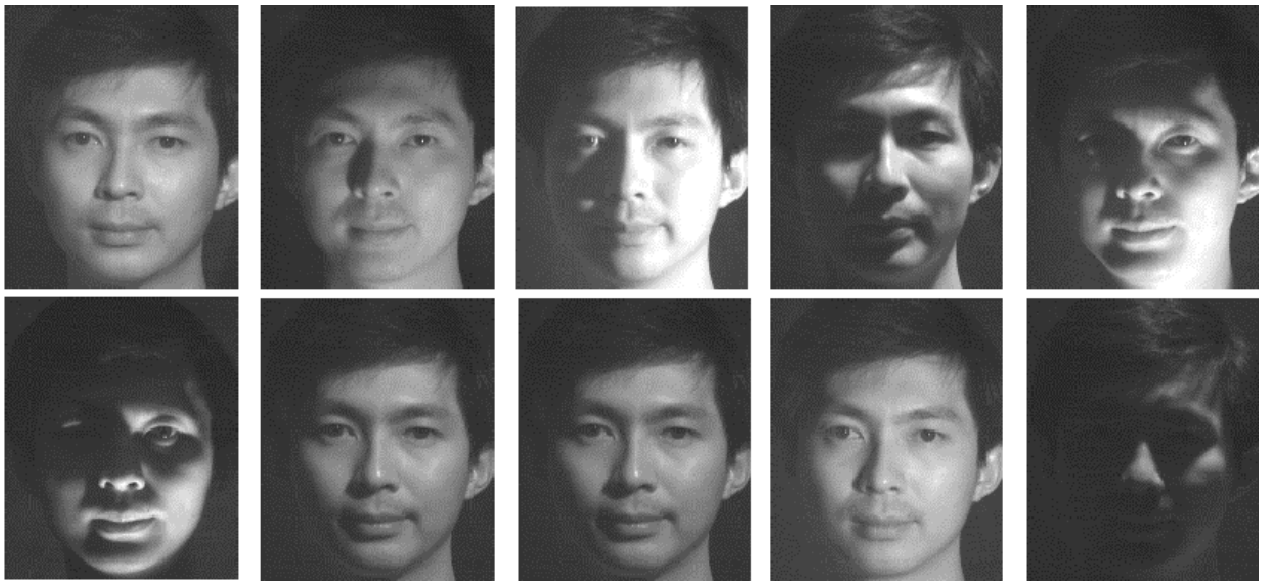
Hauptproblem



The variations between the images of the same face due to illumination and viewing direction are almost always larger than image variations due to change in face identity.

-- Moses, Adini, Ullman, ECCV‘94

Hauptproblem



The variations between the images of the same face due to illumination and viewing direction are almost always larger than image variations due to change in face identity.

-- Moses, Adini, Ullman, ECCV‘94

Ursachen für Variationen

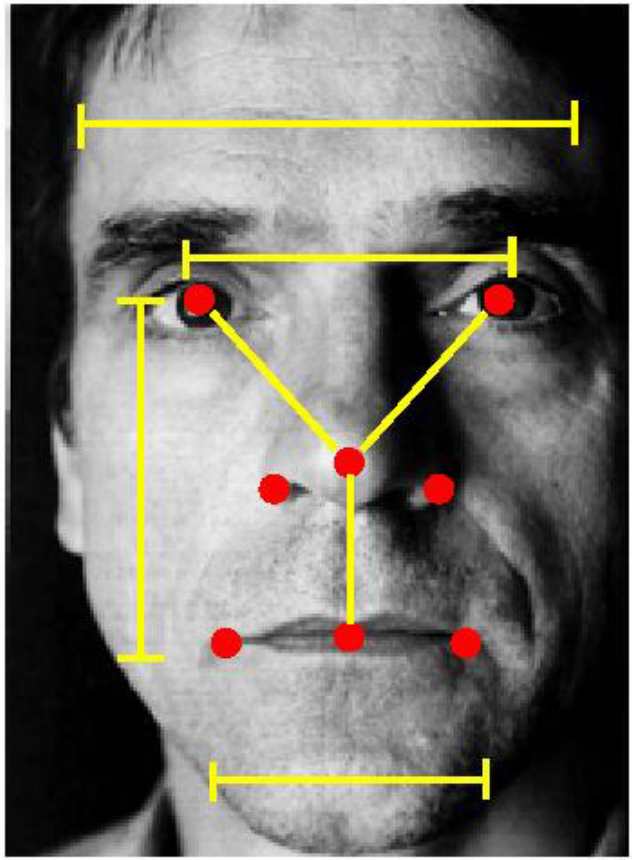
- Sicht / Blickrichtung
- Beleuchtung
- Verdeckung
- Brille
- Bart
- Makeup
- Altersunterschied

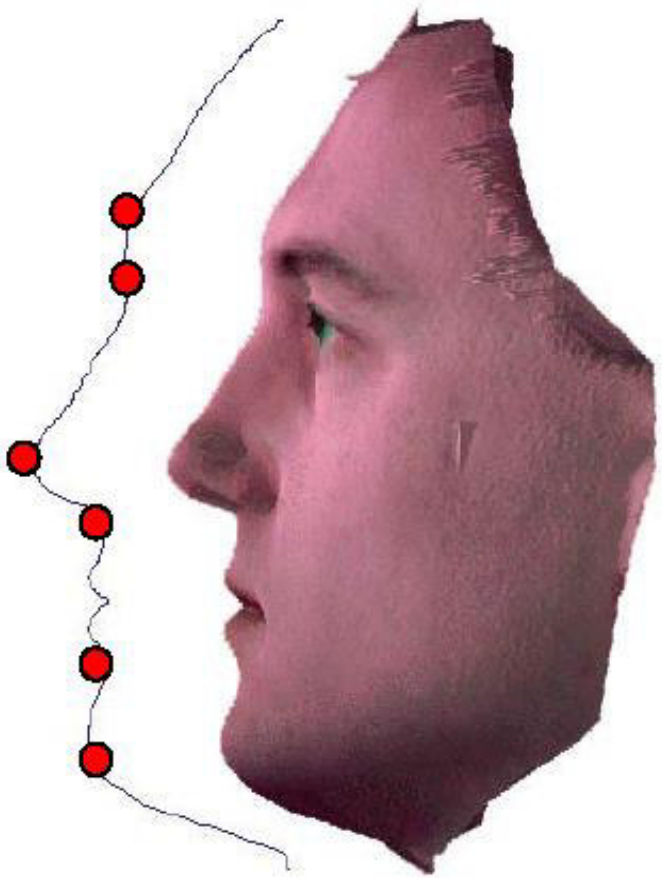
1+ Jahre Unterschied

Traditionelle Ansätze

- Merkmalsbasiert
  - Markante Punkte (engl. *fiducial points*)
  - Abstände, Winkel, Flächen, etc.
  - Geometrische Merkmale
- Ansichtsbasiert
  - Holistisch (global)
  - Markante Regionen (lokal) (engl. *fiducial regions*)
  - Statistisch

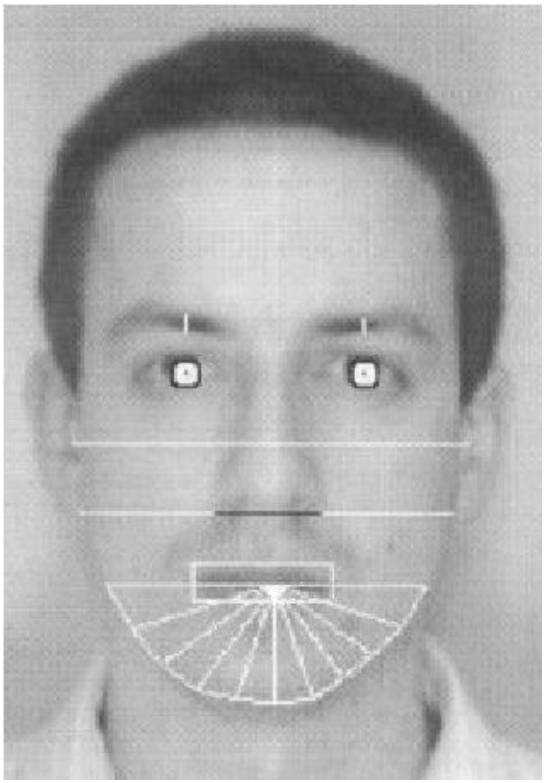
Merkmale: Frontal & Profil





Merkmalsbasierte Gesichtserkennung

- Dicke der Augenbrauen und vertikale Position relativ zum Auge
- Bogen der Augenbrauen
- Vertikale Position und Breite der Nase
- Vertikale Position des Mundes
- Breite und Höhe der Lippen
- Feste Anzahl (11) von Radien, welche die Form des Kinns beschreiben
- Breite des Gesichts auf Höhe der Nase
- Breite des Gesichts auf der mittleren Höhe zwischen Augen und Nase



R. Brunelli, T. Poggio, “Face Recognition: Features versus Templates”, *IEEE Trans. on PAMI,* Vol. 15, No. 10, pp. 1042-1052, Oct. 1993.

Beispielanwendung Gesichtserkennung

- System von Hazim Ekenel und Kai Nickel (CV:HCI Prof. Stiefelhagen)

Klassifikation

- Nächster Nachbar-Klassifikator mit der Mahalanobis-Distanz als Distanzmetrik:

$$\Delta_j(\mathbf{x}) = (\mathbf{x} - \mathbf{m}_j)^T \Sigma^{-1} (\mathbf{x} - \mathbf{m}_j)$$

$\boldsymbol{x}$: Anfragebild (Gesicht)

$\boldsymbol{m}_j$: Durchschnittsvektor der j-ten Person.

$\sum$: Kovarianzmatrix

- Eine Person wird nur durch ihren durchschnittlichen Merkmalsvektor beschrieben
- Für jede Anfrage (Gesicht) werden die Merkmalsvektoren aller in der Datenbank abgespeicherten Personen überprüft

Überblick

- Einführung
- Logik
- Aussagenlogik
- Aussagenlogische Deduktion
- Prädikatenlogik
- Planungssprachen
- Planungsstrategien

Einführung

Was ist Wissen? (naiv)

- Ich weiß, wie man Auto fährt.
- Ich weiß, wie man den Tisch deckt.
- Ich weiß, wie man englisch spricht.
- Ich weiß, wie man Differential-gleichungen löst.
- Ich weiß, was Schopenhauer über Religion geschrieben hat.
- Ich weiß, was Wissen ist.
- Ich weiß, wie man lernt.

Einführung

Was ist Wissen? (abstrakt)

Gespeicherte

- Beschreibungen, Modelle, Aktionsfolgen
- Vorschriften zur Reaktion auf Ereignisse
- Motorische, kognitive Fähigkeiten

(wieder ein wenig konkreter)

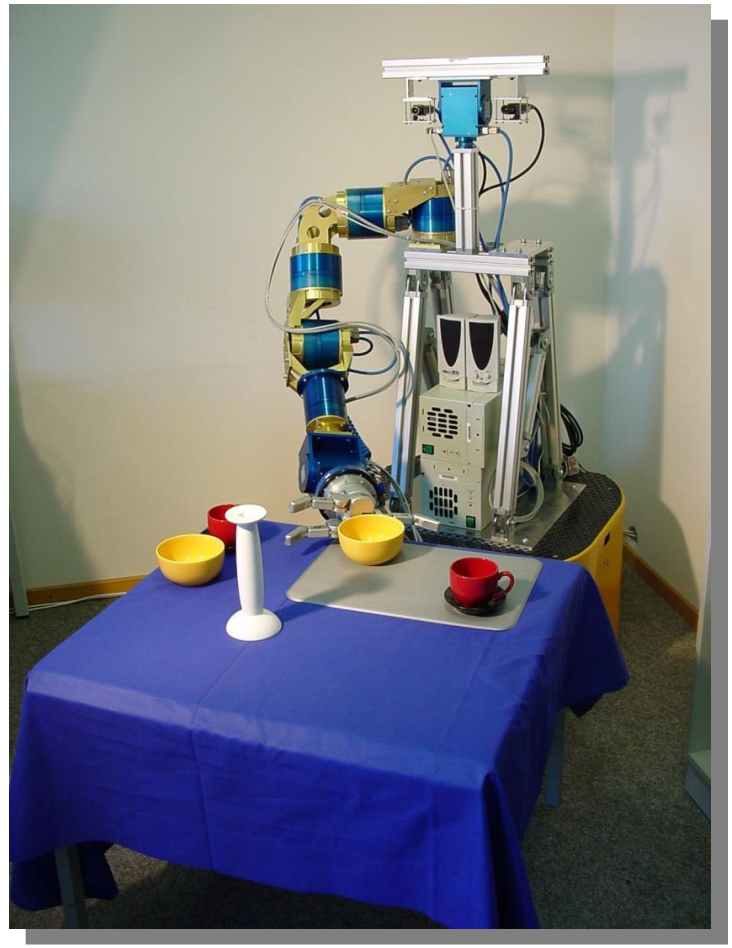
- Computerprogramme
- Regeln der Aussagenlogik
- Gewichte von Neuronalen Netzen, ...

Einführung

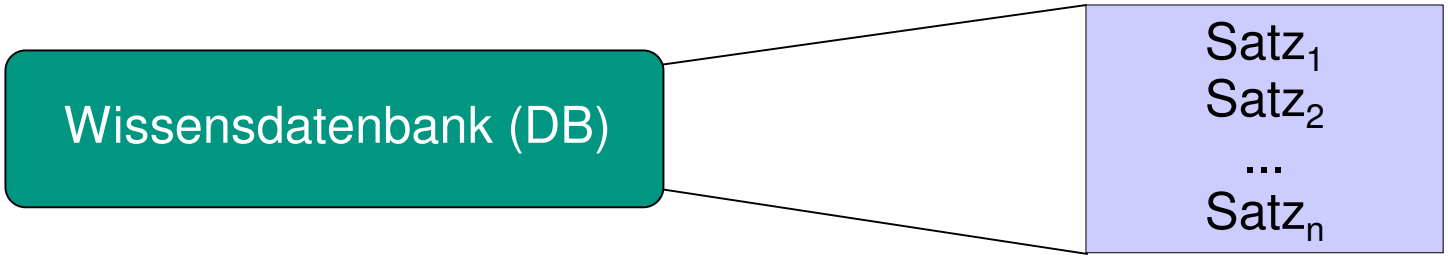
Beispiel: Tisch decken

Wissen:

- Wie bewege ich ein Objekt?
- Vorbedingung: Tisch leer
- Nachbedingungen:
  1. Teller auf Tisch
  2. Untertasse auf Tisch neben Teller
  3. Tasse auf Untertasse
  4. Kerzenleuchter egal



Wissensrepräsentation: Grundlagen



Eine **Wissensdatenbank** besteht aus **Sätzen**, die in einer **Wissensrepräsentationssprache** abgefasst sind.

Jeder **Satz** stellt eine Annahme über die Welt dar:

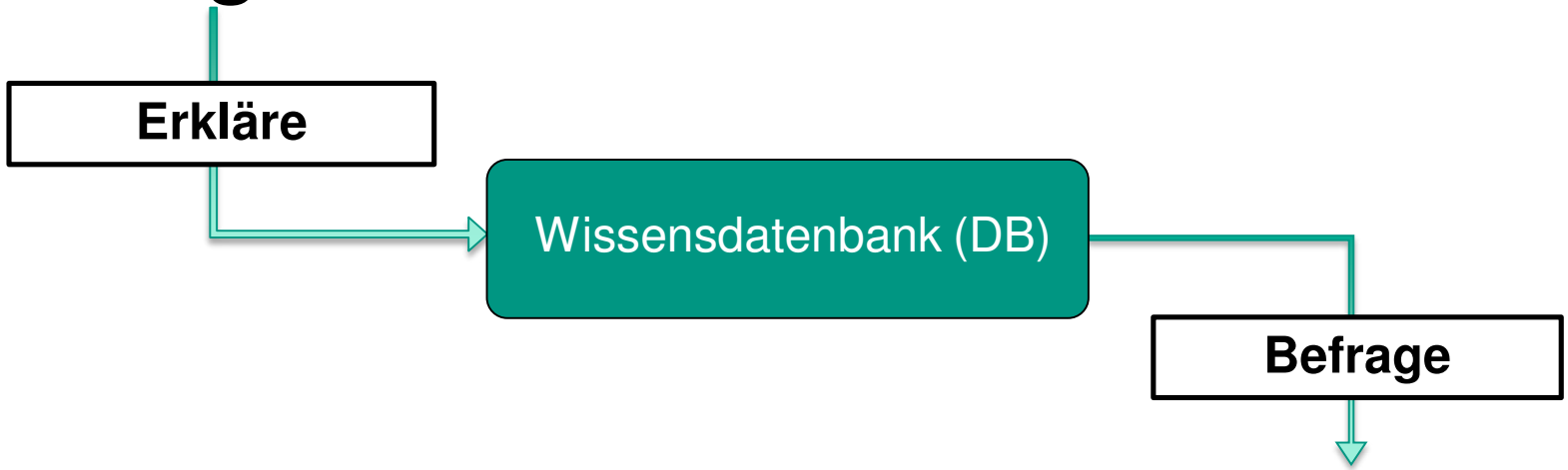
α: „**Ident1** IST-EIN Fahrrad.“

β: „**Ident2** IST-EIN Mensch.“

γ: „**Ident2** BESITZT Ident1.“ oder

δ: „ $r \in R^{\geq 0}$.“

Wissensrepräsentation: Grundlagen

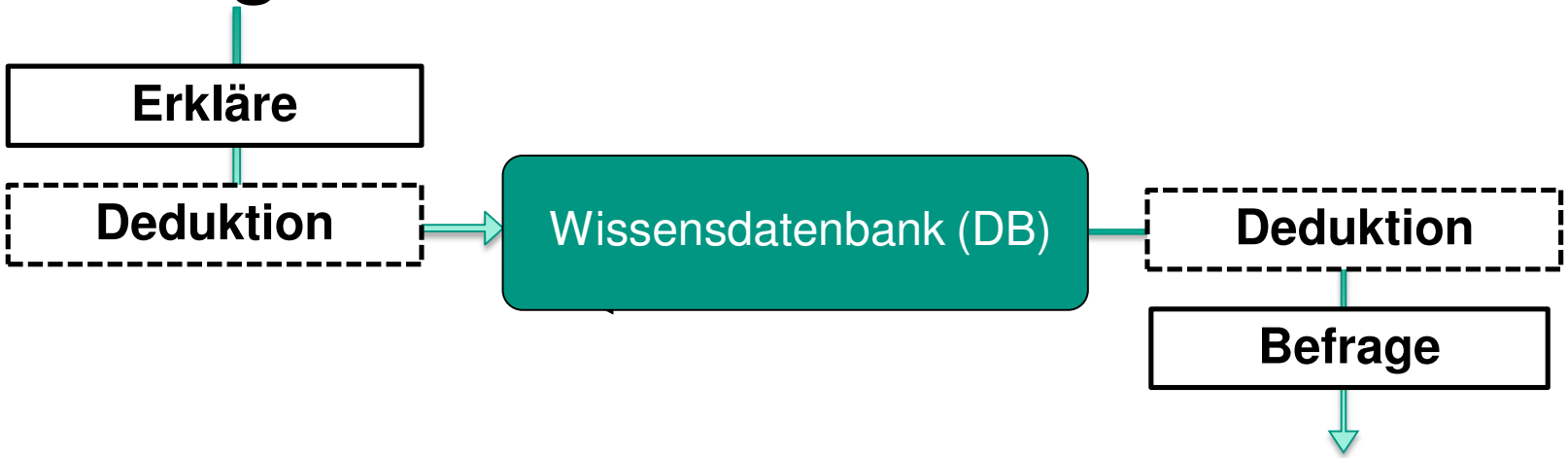


Erkläre: Füge der Datenbank (DB) Wissen hinzu.

Befrage: Frage Wissen aus der Datenbank ab.

Beispiel: **Erkläre**(DB, „**Ident1** BESITZT **Ident2.**“) ➔ OK.
Befrage(DB, „**Was** BESITZT **Ident2**?“) ➔ **Ident1**

Wissensrepräsentation: Grundlagen



Deduktion:

- Leite neue Sätze von bekannten ab
- Injektive Abbildung
- Abbildung: DB ➔ DB.
- Minimale Deduktion: Identität

Logik allgemein

Wissensrepräsentation: Logik

Logik: (**Symbolmenge**, **Belegungsmenge**, **Syntax**, **Semantik**, Folgerungsoperator $\models$)

- **Symbolmenge** *S* enthält alle Symbole, über die Aussagen gemacht werden können
- **Belegung**: Abbildung, die jedem Symbol einen Wert aus dem Wertebereich *D* zuordnet

$$M : S \rightarrow D(s_i)$$

- **Belegungsmenge** *M*: Menge aller *möglichen* Belegungen

Wissensrepräsentation: Logik

Beispiel Aussagenlogik:

$$
\begin{aligned}
S &= \{a, b\} \\
M &= \{\{(a \mapsto \text{wahr}), (b \mapsto \text{wahr})\}, \{(a \mapsto \text{wahr}), (b \mapsto \text{falsch})\}, \\
&\quad \{(a \mapsto \text{falsch}), (b \mapsto \text{wahr})\}, \{(a \mapsto \text{falsch}), (b \mapsto \text{falsch})\}\}
\end{aligned}
$$

Schreibweise

- Lateinische Buchstaben für Symbole
- Griechische Buchstaben für Sätze

Wissensrepräsentation: Logik

Logik: (**Symbolmenge**, **Belegungsmenge**, **Syntax**, **Semantik**,
Folgerungsoperator $\models$)

- **Syntax**: Legt fest, welche **Sätze** wohlgeformt sind
- Nur wohlgeformte **Sätze** sind gültig!

Beispiel:

$(a \vee b) \Rightarrow c$	wohlgeformt
$()a\vee \Rightarrow bc$	nicht wohlgeformt

Wissensrepräsentation: Logik

Logik: (**Symbolmenge**, **Belegungsmenge**, **Syntax**, **Semantik**, Folgerungsoperator $\models$)

- **Semantik** (Bedeutungslehre): Bestimmt den **Wahrheitswert** eines **Satzes** in Bezug auf eine **Belegung**.
- Wahrheitswerte für Aussagen: **wahr**, **falsch**
- Notation: $\cdot / M \rightarrow \{\text{wahr}, \text{falsch}\}$
  Ordnet einem Satz mit Bezug auf eine Belegung einen Wahrheitswert zu. „Ausgewertet in der Belegung *M*“
- Beispiel:

  Satz α: $(a \wedge b) \Rightarrow c$
  Belegung $M_1 = \{(a \mapsto \textbf{wahr}), (b \mapsto \textbf{wahr}), (c \mapsto \textbf{wahr})\}$
  $\Rightarrow \alpha / M_1$ ist **wahr,** „M_1 erfüllt α “
  Belegung $M_2 = \{(a \mapsto \textbf{wahr}), (b \mapsto \textbf{wahr}), (c \mapsto \textbf{falsch})\}$
  $\Rightarrow \alpha / M_2$ ist **falsch**

Wissensrepräsentation: Logik

Logik: (**Symbolmenge**, **Belegungsmenge**, **Syntax**, **Semantik**, Folgerungsoperator $\models$)

- $\alpha \models \beta$: „β folgt aus α“
- $\alpha \models \beta$ genau dann, wenn für alle Belegungen, in denen α wahr ist, β ebenfalls wahr ist.

Wissensrepräsentation: Logik

Zurück zur Wissensbasis *WB:*

- Kann als $\wedge$-verknüpfte Sequenz von **Sätzen** aufgefasst werden
  Beispiel:

$$\begin{aligned} WB &= \{\alpha_1 : x + y = 4, \alpha_2 : x/y = 1\} \\ &= \alpha_1 \wedge \alpha_2 \end{aligned}$$

- *WB* ist **wahr** für $M = \{(x \mapsto 2),(y \mapsto 2)\}$

$\Rightarrow$ *M* erfüllt *WB*

- Es gilt $WB \models \beta$ genau dann, wenn

$$\left(\bigwedge_{\alpha \in WB} \alpha \right) \Rightarrow \beta$$

allgemeingültig ist (d.h. wahr in allen WB erfüllenden Belegungen ist).

Logik: Prüfung der Ableitbarkeit

Ein **Algorithmus** zur Überprüfung der „$WB \models \alpha$“-Relation:

- Sei M die Menge aller Belegungen über S
- Sei $N \subseteq M$ die Menge aller Belegungen, für die gilt:
  $WB / M_i = \text{wahr}$
- $WB \models \alpha$ gilt genau dann, wenn
  $\forall M_i \in N : \alpha / M_i = \text{wahr}$

Logik: Deduktion

Deduktion (Ableitung):

- Wenn ein Algorithmus *i* existiert, der den Satz α aus der
  Wissensbasis *WB* ableiten kann, so schreiben wir

$$WB \vdash_i \alpha$$

- In Worten:
  - „α wird durch *i* aus *WB* abgeleitet“
  - „i leitet α aus WB ab“.

Logik: Deduktion

Eigenschaften von Deduktions-Algorithmen:

- Ein Algorithmus *i* ist **korrekt** genau dann, wenn er *nur* Sätze aus *WB* ableitet, die aus der *WB* folgen:

$$\forall \alpha : WB \vdash_i \alpha \Rightarrow WB \models \alpha$$

- Ein Algorithmus *i* ist **vollständig** genau dann, wenn er *alle* Sätze aus der WB ableitet, die aus der *WB* folgen:

$$\forall \alpha : WB \models \alpha \Rightarrow WB \vdash_i \alpha$$

Logik: Zusammenfassung

Eine **Logik**:

- besteht aus **Symbolmenge**, **Belegungsmenge**, **Syntax**, **Semantik** und **Folgerung**
- bestimmt den Wahrheitsgehalt von **Sätzen** in Bezug auf **Belegungen**

Eine **Wissensbasis**:

- besteht aus einer Menge von **Sätzen**
- passt zu einer Belegung *M* (oder auch nicht)
- Ist von einer Belegung *M* erfüllbar (oder nicht)

Ein **Deduktions**-Algorithmus:

- leitet **Sätze** aus einer Wissensbasis ab
- kann **korrekt** und / oder **vollständig** sein

Aussagenlogik

Aussagenlogik

- Aussagenlogik als Beispiel einer sehr einfachen Logik
- sehr alt (antikes Griechenland)
- bildet die Basis, auf der jede klassische Logik gründet
- eignet sich sehr „natürlich“ zur Illustration bestimmter Konzepte
- Erfüllbarkeit ist entscheidbar.

Aussagenlogik: Syntax

Satz	→	Atom | Komplex	
Atom	→	*true* | *false* | Symbol	
Symbol	→	*P* | *Q* | *R* | ...	
Komplex	→	¬ Satz	„nicht“
		(Satz ∧ Satz)	„und“
		| (Satz ∨ Satz)	„oder“
		| (Satz ⇒ Satz)	„impliziert“
		| (Satz ⇔ Satz)	„äquivalent“

Aussagenlogik: Syntax

- Symbole der AL-Formel werden durch Belegung mit „wahr“ oder „falsch“ belegt
- Der atomare Satz *true* ist immer wahr, *false* immer falsch
- Danach rekursive Auswertung der Formel mit einer Belegung M:
  - $\neg \alpha / M$ = wahr $\Leftrightarrow$ α / M = falsch
  - $(\alpha \wedge \beta) / M$ = wahr $\Leftrightarrow$ α / M = wahr und β / M = wahr
  - $(\alpha \vee \beta) / M$ = wahr $\Leftrightarrow$ α / M = wahr oder β / M = wahr
  - $(\alpha \Rightarrow \beta) / M$ = wahr $\Leftrightarrow$ α / M = falsch oder β / M = wahr
  - ...

Aussagenlogik : Begriffe

- **Literal**: Symbol oder negiertes Symbol
- **Disjunktion**: Menge von oder-verknüpften Literalen
- **Konjunktion**: Menge von und-verknüpften Literalen
- Aussage in **konjunktiver Normalform (KNF)** ⇔ Aussage ist eine Menge und-verknüpfter Disjunktionen von Literalen
- Aussage in **disjunktiver Normalform (DNF)** ⇔ Aussage ist eine Menge oder-verknüpfter Konjunktionen von Literalen $KNF: \quad (P \vee \neg Q \vee R) \wedge (\neg P \vee \neg R) \wedge Q$
- Beispiel: $DNF: \quad P \wedge Q \wedge \neg R \vee \neg P \wedge \neg Q \wedge R$

Aussagenlogik : Deduktion

Drei Algorithmen zur Deduktion

- Resolution
- Horn-Klauseln
- DPLL

Aussagenlogik: Muster

Korrekte Deduktionsregeln aus der klassischen Logik:

1. **Modus Ponens**:

   Aus den Sätzen $\alpha \Rightarrow \beta$ und α, kann β **inferiert (abgeleitet)** werden.

$$\frac{\alpha \Rightarrow \beta, \alpha}{\beta}$$

   Beispiel:
   Wenn ($FahrradDa \wedge \neg FahrradAbgeschl$) $\Rightarrow$ *FahreHeim* und *FahrradDa* $\wedge \neg$*FahrradAbgeschl* gegeben, dann kann *FahreHeim* inferiert werden.

Aussagenlogik: Muster

2. **Und-Elimination**:

   Aus einer Konjunktion kann jedes der Elemente inferiert werden.

$$\frac{\alpha \wedge \beta}{\alpha} \quad \text{und} \quad \frac{\alpha \wedge \beta}{\beta}$$

   Korrektheitsbeweise aus Wahrheitstabelle!

   Beispiel:
   Wenn (*FahrradDa* $\wedge$ *FahrradAbgeschl*) gegeben, können sowohl *FahrradDa,* als auch *Fahrradabgeschl* inferiert werden.

Aussagenlogik: Resolution

3. Einheits-Resolutionsregel:

- Folgt aus dem Modus Ponens
- Seien $P_1, \ldots, P_k$ und Q Literale mit $Q = \neg P_i$.
  Dann gilt:

$$\frac{(P_1 \vee \ldots \vee P_k), Q}{P_1 \vee \ldots \vee P_{i-1} \vee P_{i+1} \vee \ldots \vee P_k}$$

Aussagenlogik: Resolution

Allgemeine Form der Resolutionsregel:

- Seien $P_1,...,P_k$ und $Q_1,...,Q_n$ Literale, und es gelte $P_i = \neg Q_j$.

  Dann gilt:

$$\frac{(P_1 \vee ... \vee P_k), (Q_1 \vee ... \vee Q_n)}{P_1 \vee ... \vee \cancel{P_i} \vee ... \vee P_k \vee Q_1 \vee ... \vee \cancel{Q_j} \vee ... \vee Q_n}$$

Logik: Resolution

Resolutionsregel:

- Aus $(\neg p \vee \alpha)$ und $(p \vee \beta)$ kann $(\alpha \vee \beta)$ geschlossen werden. Es darf immer nur **genau ein** Literal resolviert werden.
- Die Wissensbasis *WB* muss in KNF vorliegen
  - Die einzelnen Disjunktionen werden als **Klauseln** geschrieben, also als Menge der darin vorkommenden Literale
  - Beispiel:

$$K = \{\{P, \neg Q, R\}, \{\neg P, R\}, \{Q\}\}$$

- Mit Hilfe der Resolutionsregel können weitere gültige Klauseln generiert und zur *WB* hinzugefügt werden.
  - Beispiel: $K = \{\{P, \neg Q, R\}, \{\neg P, R\}, \{\neg Q, R\}, \{Q\}\}$

Logik: Resolution

- Jeder vollständige **Suchalgorithmus**, kombiniert mit der **Resolutionsregel**, kann **jede Schlussfolgerung** ableiten, die aus **jeder Wissensbasis** der Aussagenlogik folgt.

- bildet die Basis für Familie von *vollständigen* Deduktionsalgorithmen

- **Widerspruchsbeweis**:
  beweist $WB \models \alpha$, indem $(WB \wedge \neg\alpha)$ widerlegt wird

$$WB \wedge \neg\alpha \vdash_{Resolution} \{\}$$

Logik: Resolutionsalgorithmus

Vorgehensweise:

- Wissensbasis in KNF bringen, falls nötig, und als Klauseln formulieren
  - $(P \vee Q) \wedge (\neg R \vee S)$ wird zu $\{P, Q\}, \{\neg R, S\}$
- Zu widerlegende Aussage: $(WB \wedge \neg\alpha)$
  - $\neg\alpha$ in KNF bringen
  - als Klausel schreiben
  - zu den Klauseln der WB hinzufügen
- Systematisch die Resolutionsregel anwenden, bis eine leere Klausel {} erzeugt wird
- Dann ist die Klauselmenge unerfüllbar

$$(WB \wedge \neg\alpha) \text{ unerfüllbar } \Rightarrow WB \models \alpha$$

Logik: Resolutionsalgorithmus

Beispiel:

Klauseln in WB

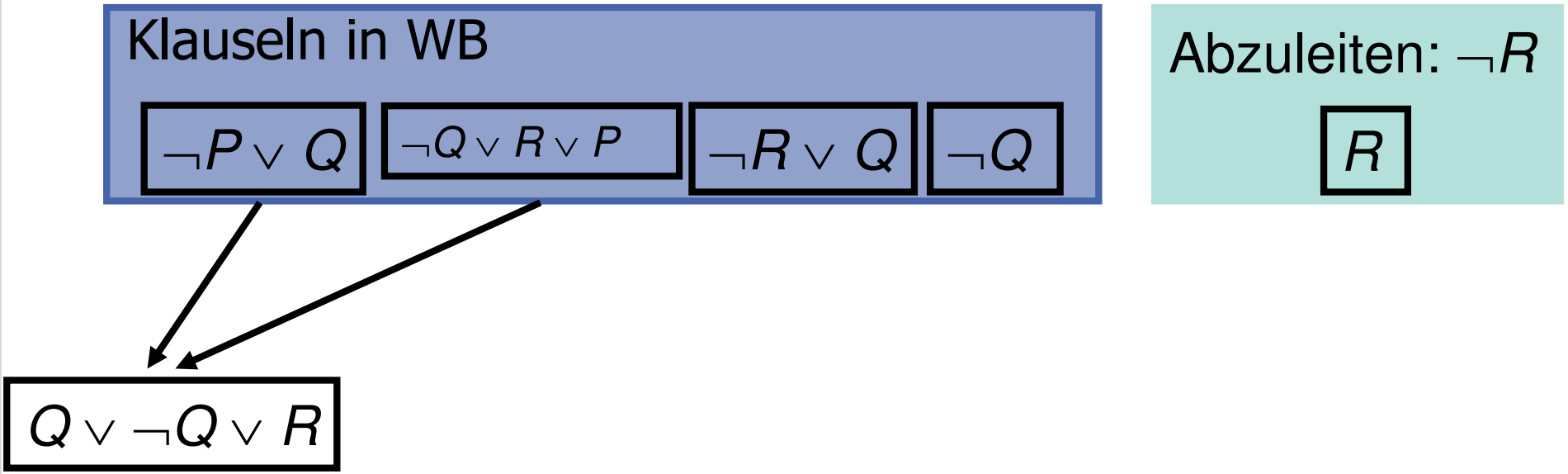
$\neg P \vee Q$ | $\neg Q \vee R \vee P$ | $\neg R \vee Q$ | $\neg Q$

Abzuleiten: $\neg R$

R

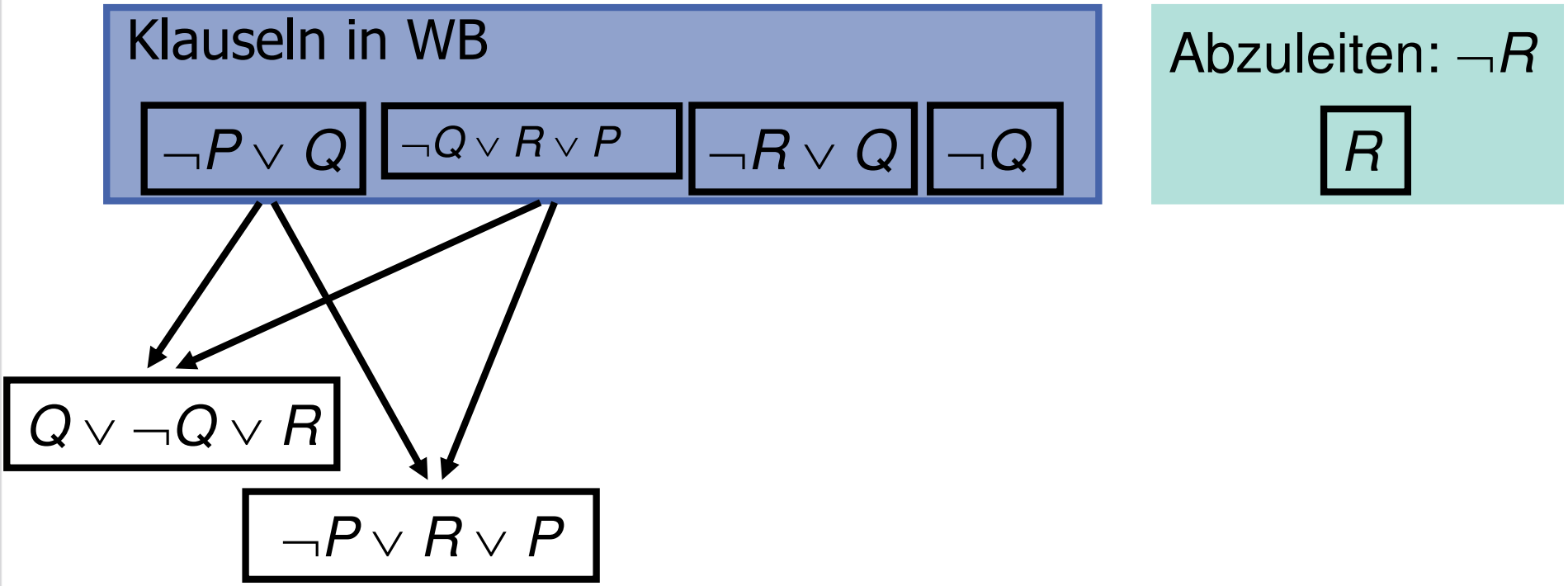
Logik: Resolutionsalgorithmus

Beispiel:



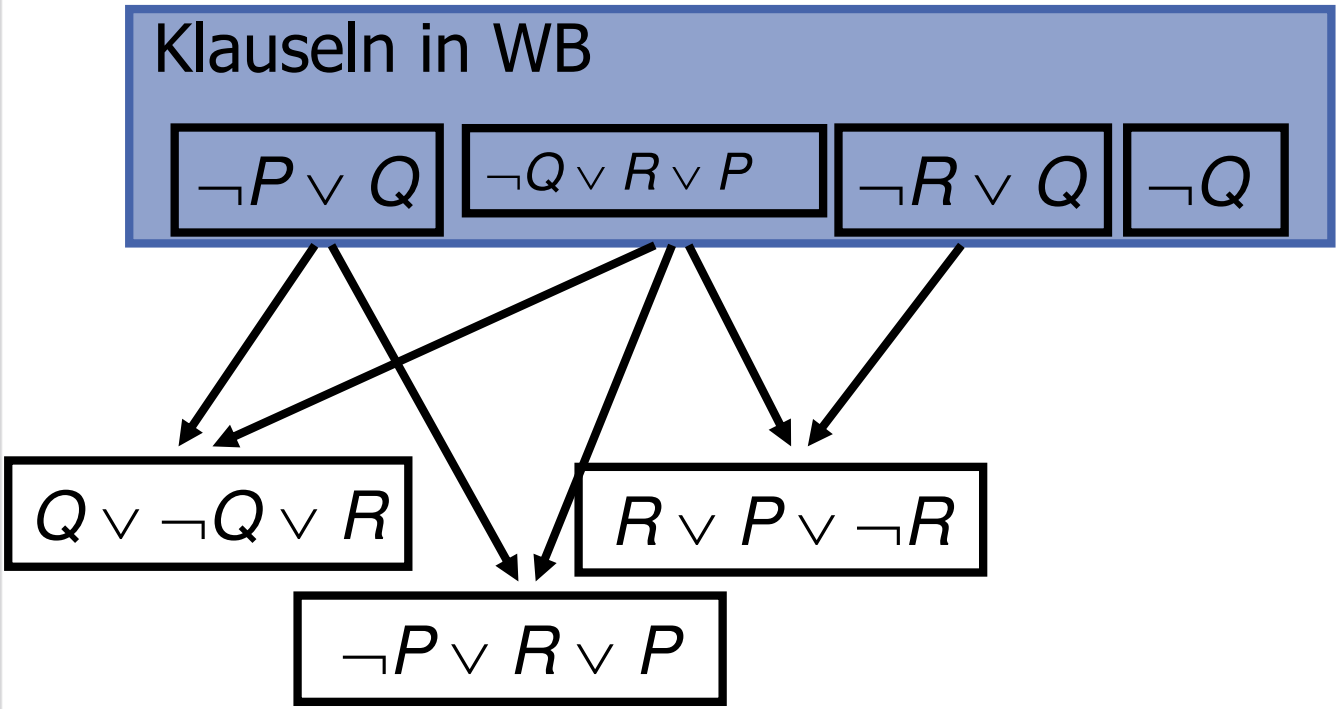
Logik: Resolutionsalgorithmus

Beispiel:



Logik: Resolutionsalgorithmus

Beispiel:

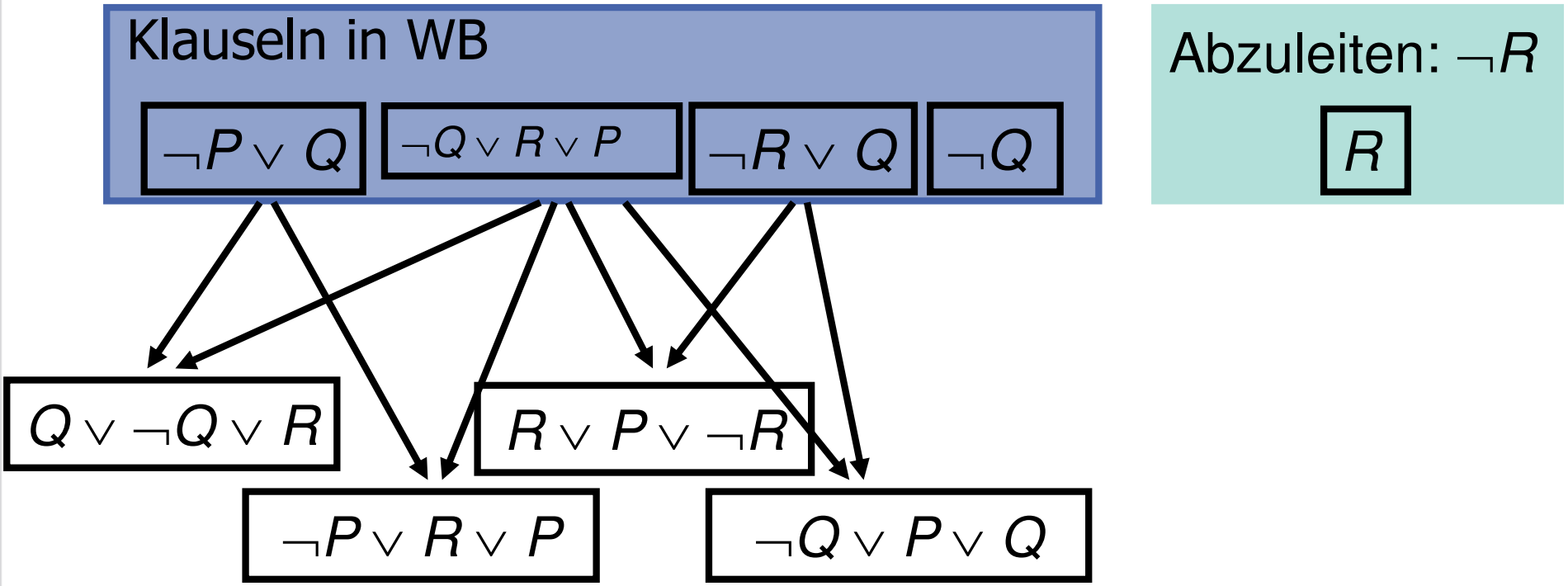


Abzuleiten: $\neg R$

R

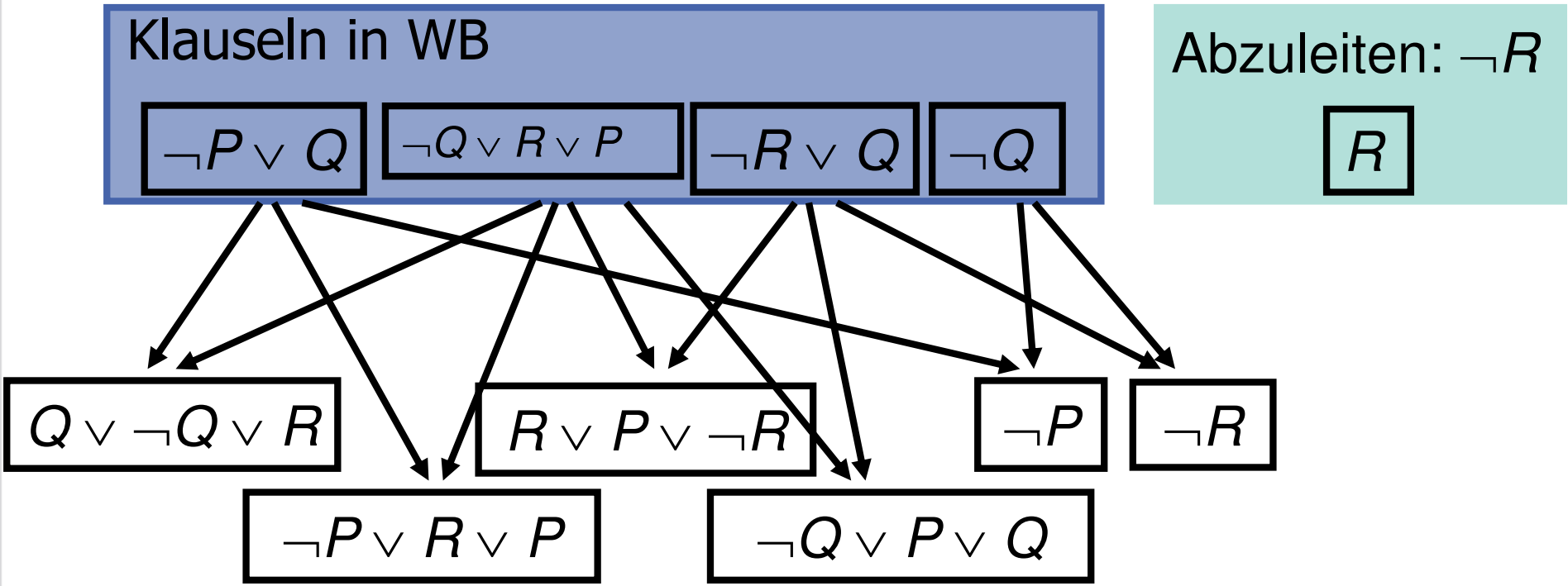
Logik: Resolutionsalgorithmus

Beispiel:



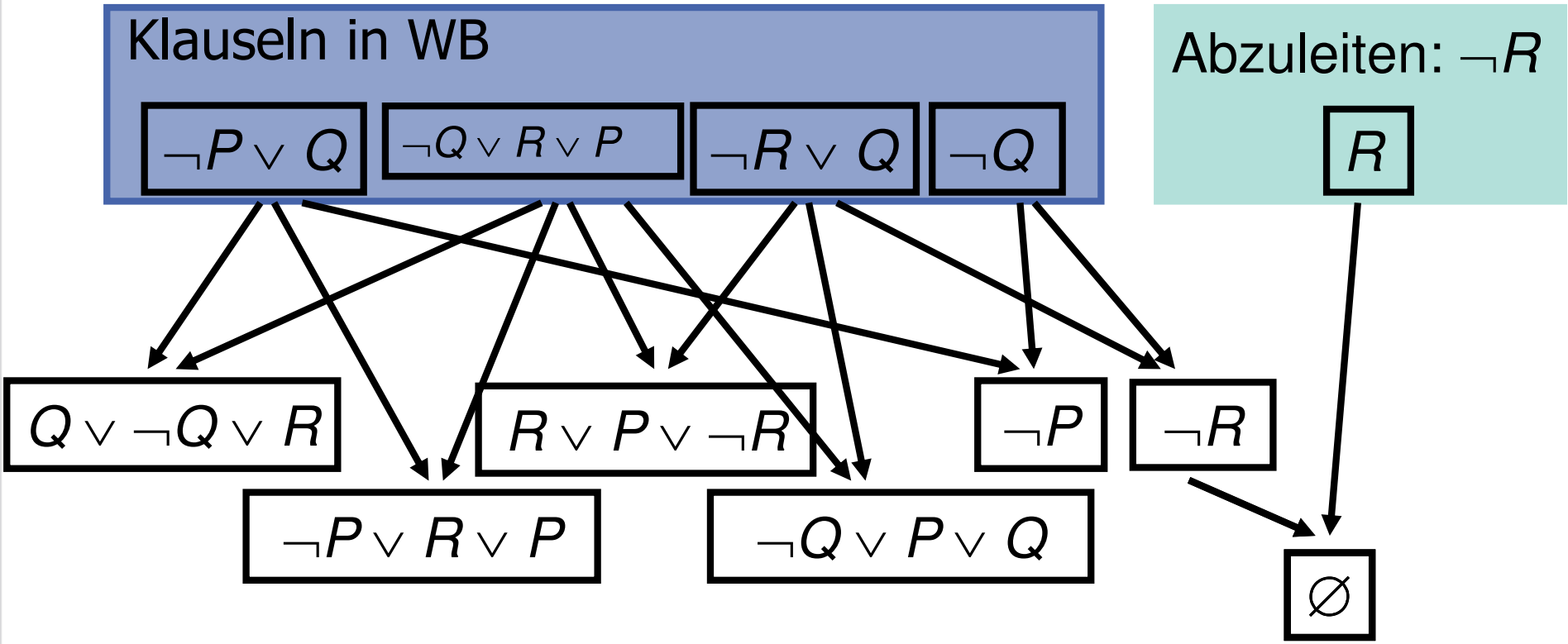
Logik: Resolutionsalgorithmus

Beispiel:



Logik: Resolutionsalgorithmus

Beispiel:



Problem: Erfüllbarkeitsproblem der Aussagenlogik (SAT) ist NP-vollständig (d.h. vermutlich exponentieller Worst-case-Aufwand)!

Logik: Horn-Klauseln

Horn-Klauseln:

- Disjunktion von Literalen, von denen *höchstens eins positiv* ist
- Beispiel:

$$(\neg P \vee \neg Q \vee R)$$

- Lässt sich schreiben als:

$$\underbrace{(P \wedge Q)}_{\text{Körper}} \Rightarrow \underbrace{R}_{\text{Kopf}}$$

Logik: Horn-Klauseln

- Typen von Horn-Klauseln:
  - keine negativen Literale: *Fakt, Axiom*
  - genau ein positives Literal: *Definition*
  - kein positives Literal: *Integritätseinschränkung*
- Eigenschaften von Horn-Klausel-basierten Wissensbasen:
  - Einfach visualisierbar: *Und-Oder-Graph*
  - Inferenz durch *Vorwärts- und Rückwärtsverkettung*,
  - sehr natürliche und leicht verständliche Algorithmen
  - Folgerungsentscheidung kann in *linearer Zeit* geschehen!
  - ABER: Horn-Formeln nur Teilmenge der prädikatenlogischen Formeln!

$$P$$

$$P \wedge Q \Rightarrow R$$

$$(\neg P \vee \neg Q) \Leftrightarrow$$

$$P \wedge Q \Rightarrow \text{false}$$

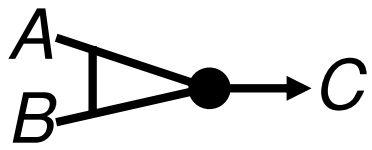
Logik: Horn-Klauseln

Beispiel:

Und-Oder-Graph: einfache Visualisierung eines Horn-Klausel-Systems

Legende:

$A \rightarrow B \quad A \Rightarrow B$



$A \wedge B \Rightarrow C$

$P \Rightarrow Q$

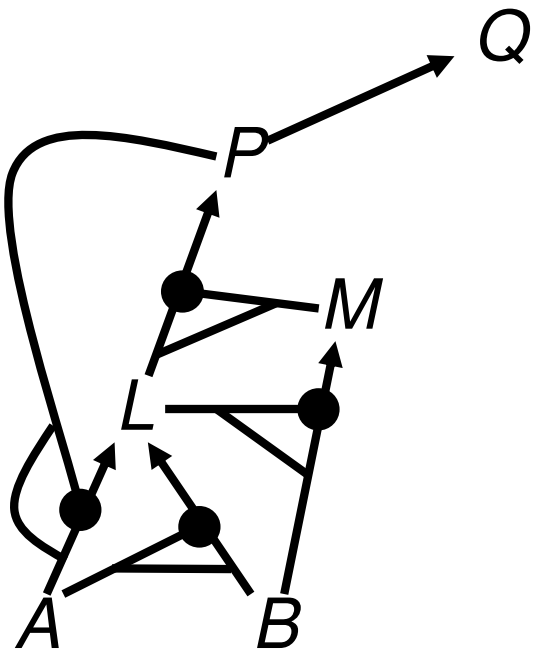
$A \wedge P \Rightarrow L$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

$A \wedge B \Rightarrow L$

$A \quad B$



Logik: Horn-Klauseln

- Vorwärtsverkettung
  - Gehe von Fakten aus; markiere bekannte Fakten als wahr
  - Arbeite Dich vorwärts durch den Baum
  - Ergebnis:
    Alle durchführbaren Folgerungsentscheidungen
- Rückwärtsverkettung:
  - Gehe von Anfrage aus
  - Arbeite Dich rückwärts durch den Baum
  - Stoppe bei bekannten Fakten (*Ground Truth,* Axiome)
  - Ergebnis:
    Wahrheitsgehalt der Anfrage

$$P \Rightarrow Q$$

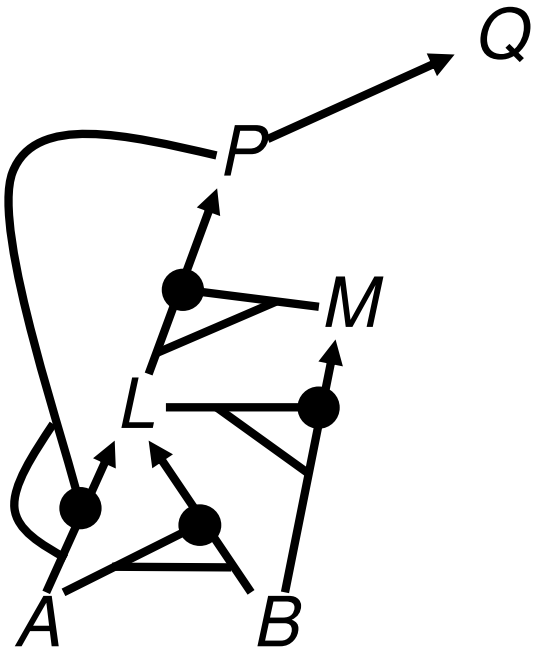
$$A \wedge P \Rightarrow L$$

$$L \wedge M \Rightarrow P$$

$$B \wedge L \Rightarrow M$$

$$A \wedge B \Rightarrow L$$

$$A \quad B$$



Logik: DPLL

Davis-Putnam-Logemann-Loveland (DPLL):

- Rekursiver Algorithmus zur Prüfung der Erfüllbarkeit
- Folgende Verbesserungen (zur Resolution):
  - **Früher Abbruch:**
    - Beispiel: $(P \vee Q) \wedge (P \vee R)$ / M = wahr, sobald P / M = wahr
  - **Einheitsklauseln** (Klauseln mit nur einem Literal):
    - Konjunktionen mit Einheitsklauseln sind nur wahr, wenn die Einheitsklauseln wahr sind
    - ➔ weitere Einschränkung des Suchraums
  - **reine Symbole:**
    - In $(P \vee \neg Q)$, $(\neg Q \vee \neg R)$, $(R \vee P)$ sind
      - P und $\neg Q$ rein
      - R unrein
    - Suche schränkt Raum möglicher Belegungen stark ein!

Logik: DPLL

- Rekursionsschritt:

```
function DPLL(k, s, m):
    # k Klauselmenge, s Symbolmenge, m (partielle) Belegung
    if alle Klauseln bereits wahr in m: return wahr
    if eine Klausel bereits falsch in m: return falsch
    P, wert := FINDE_EINHEITSKLAUSEL(k, s, m)
    if P existiert:
        return DPLL(k, s-P, SETZE_IN_BELEGUNG(P, wert, m))
    P, wert := FINDE_REINES_LITERAL(k, m)
    if P exisiert:
        return DPLL(k, s-P, SETZE_IN_BELEGUNG(P, wert, m))
    P := ERSTES(s); rest := REST(s)
    return DPLL(k, rest, SETZE_IN_BELEGUNG(P, true, m)) or
           DPLL(k, rest, SETZE_IN_BELEGUNG(P, false, m))
```

Logik: DPLL

- DPLL zur Überprüfung von Erfüllbarkeit:

```
function DPLL_ERFÜLLBAR(s):
    # s: Satz der Aussagenlogik
    klauseln := Klauselmenge der KNF von s
    symbole := Menge der Symbole aus s
    return DPLL(klauseln, symbole, {})
```

Beispiel

$$
\begin{array}{lllllllll}
P_1 & \vee & \neg P_2 & \vee & P_3 & \vee & \neg P_4 & \vee & \neg P_5 \\
P_1 & & & \vee & P_3 & \vee & P_4 & \vee & P_5 \\
P_1 & & & \vee & \neg P_3 & \vee & P_4 & & \\
P_1 & & & \vee & P_3 & \vee & \neg P_4 & \vee & P_5 \\
P_1 & & & \vee & \neg P_3 & \vee & \neg P_4 & & \\
\neg P_1 & \vee & \neg P_2 & & & & & & \\
 & & P_2 & & & & & & \\
 & & \neg P_2 & \vee & P_3 & \vee & P_4 & \vee & \neg P_5
\end{array}
$$

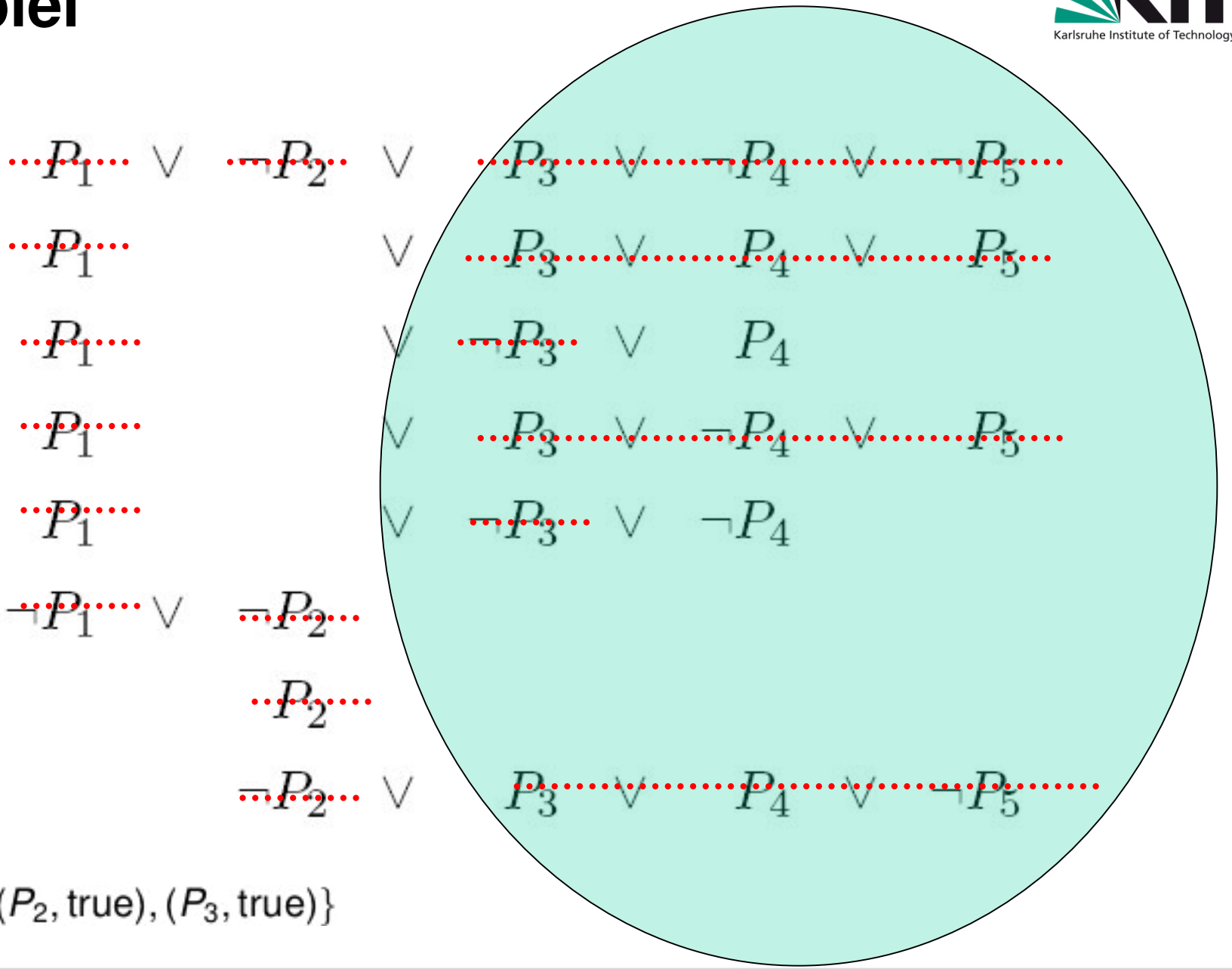
$\{(P_2, \text{wahr})\}$ (Einheitsklausel)

Beispiel

$$
\begin{array}{lllllllll}
P_1 & \vee & \neg P_2 & \vee & P_3 & \vee & \neg P_4 & \vee & \neg P_5 \\
P_1 & & & \vee & P_3 & \vee & P_4 & \vee & P_5 \\
P_1 & & & \vee & \neg P_3 & \vee & P_4 & & \\
P_1 & & & \vee & P_3 & \vee & \neg P_4 & \vee & P_5 \\
P_1 & & & \vee & \neg P_3 & \vee & \neg P_4 & & \\
\neg P_1 & \vee & \neg P_2 & & & & & & \\
 & & P_2 & & & & & & \\
 & & \neg P_2 & \vee & P_3 & \vee & P_4 & \vee & \neg P_5
\end{array}
$$

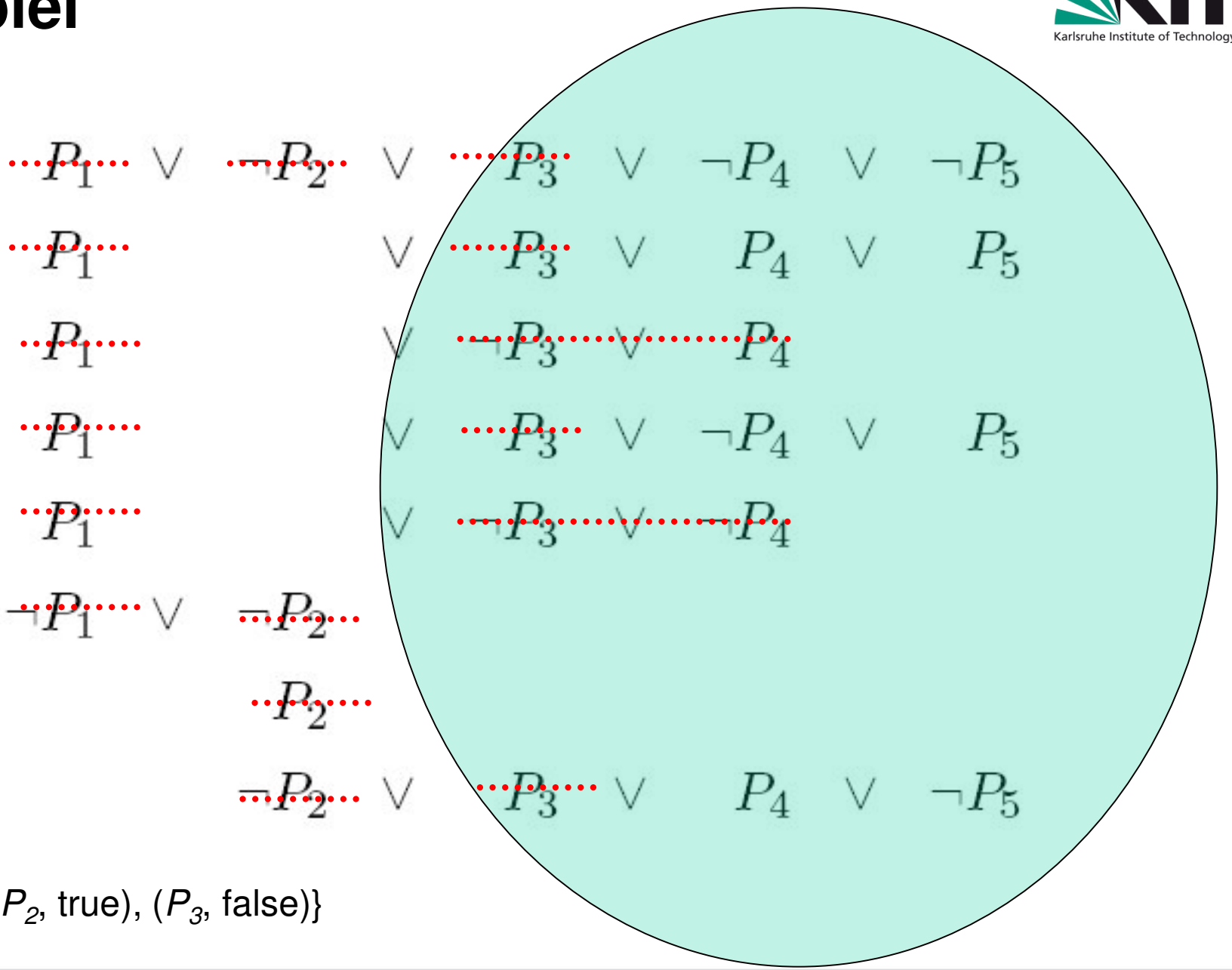
$\{(P_1, \text{false})(P_2, \text{true})\}$ (Einheitsklausel)

Beispiel



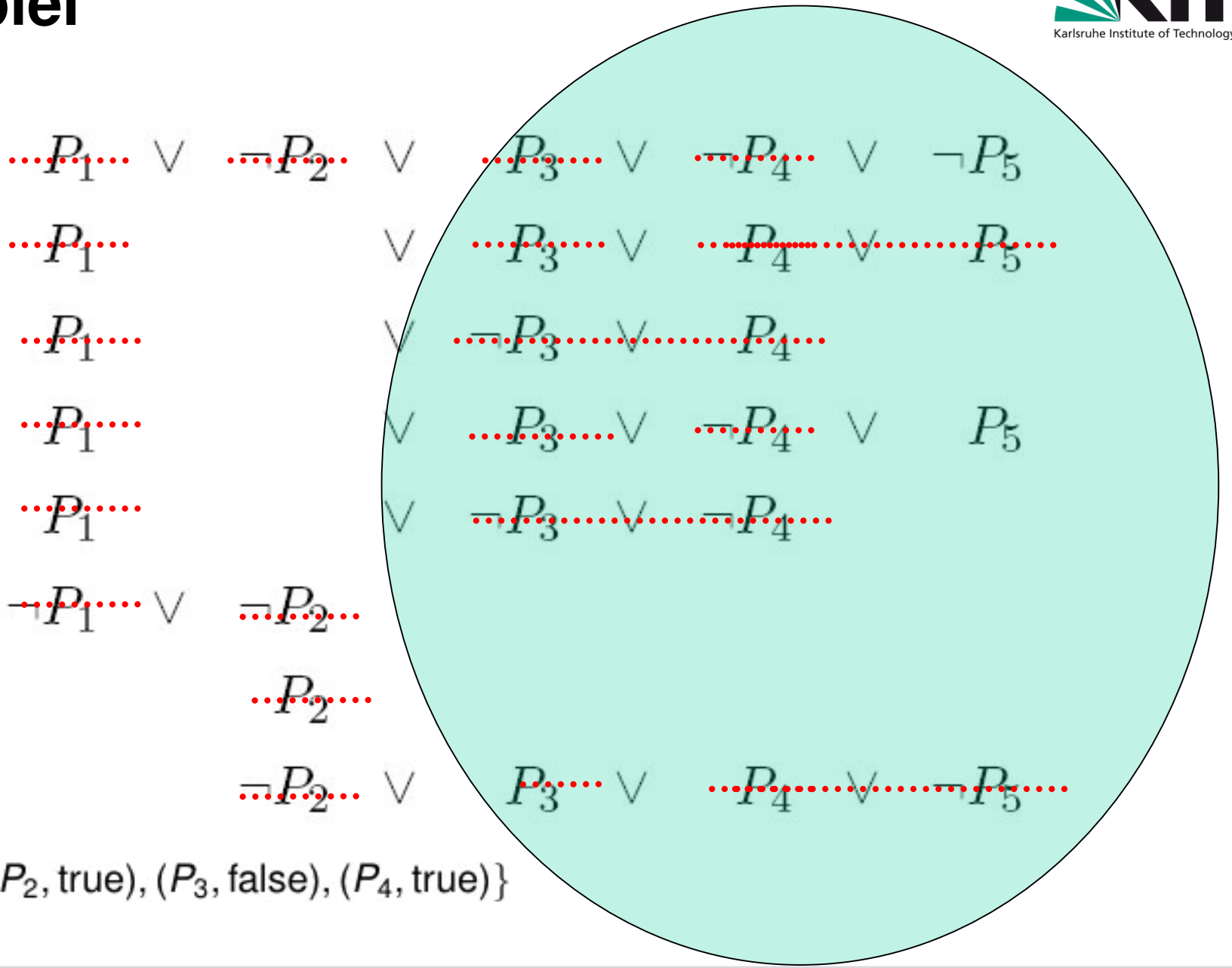
$\{(P_1, \text{false}), (P_2, \text{true}), (P_3, \text{true})\}$

Beispiel



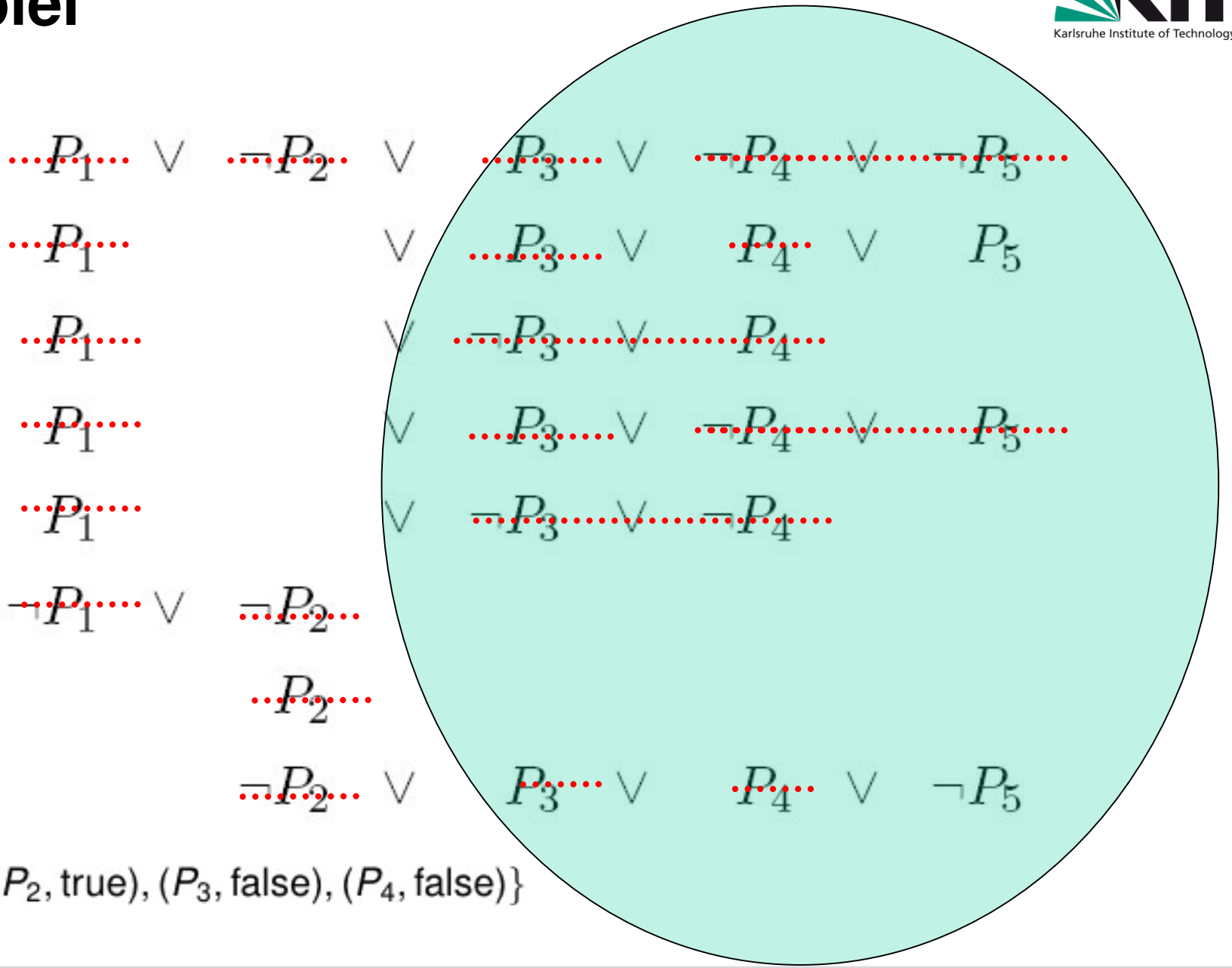
{(P_1, false), (P_2, true), (P_3, false)}

Beispiel



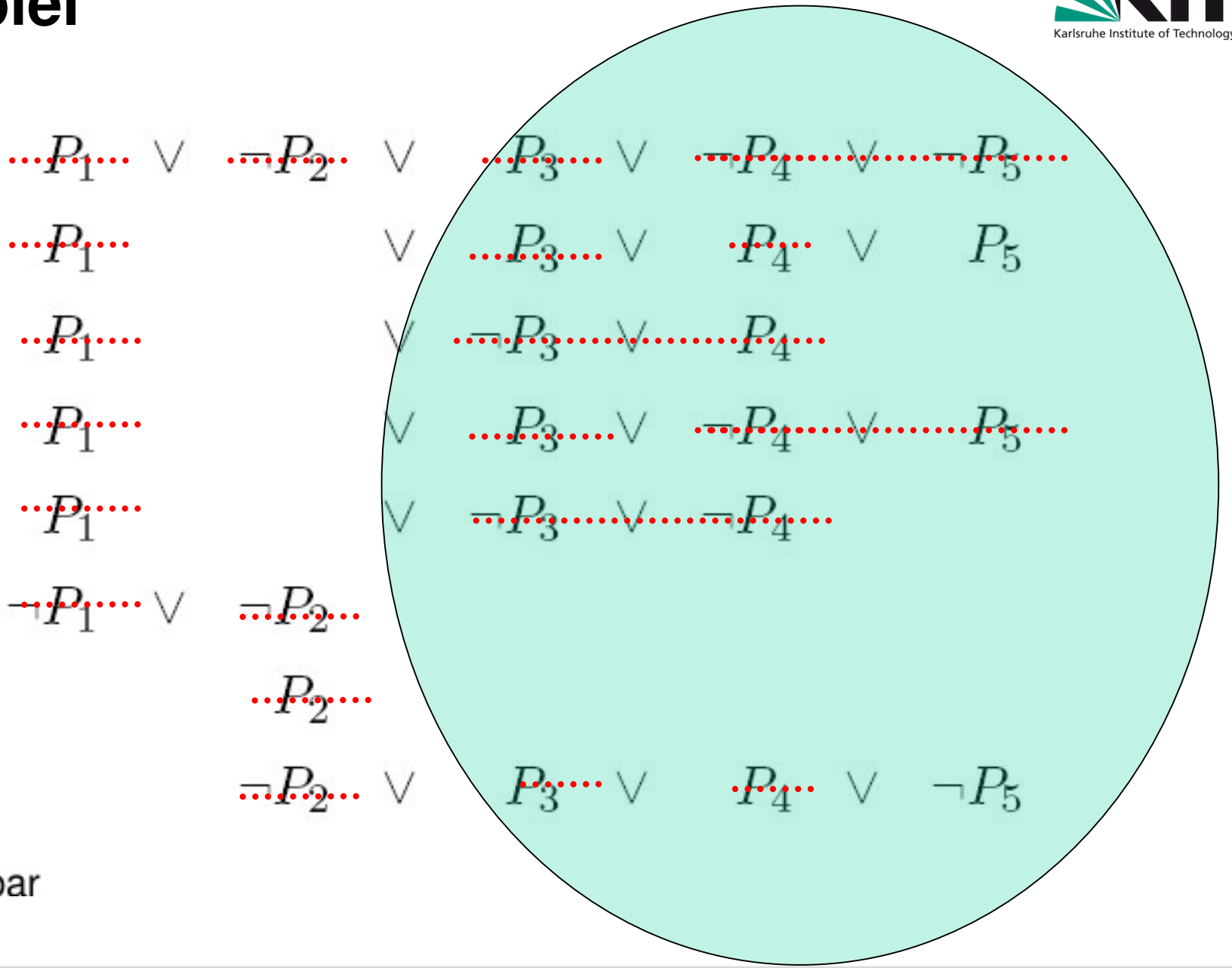
$\{(P_1, \text{false}), (P_2, \text{true}), (P_3, \text{false}), (P_4, \text{true})\}$

Beispiel



$\{(P_1, \text{false}), (P_2, \text{true}), (P_3, \text{false}), (P_4, \text{false})\}$

Beispiel



$\Rightarrow$ unerfüllbar

Prädikatenlogik

Prädikatenlogik

- Aussagenlogik nicht sehr mächtig
- Prädikatenlogik hinreichend, um fast alles zu beschreiben
- Aber wesentlich schwieriger zu behandeln:
  - (Un)erfüllbarkeit einer PL-Formel nur semientscheidbar
  - Unerfüllbarkeit: nachweisbar in endlicher Zeit
  - Erfüllbarkeit: Algorithmus terminiert evtl. nicht

Prädikatenlogik

- Neu im Vergleich zur Aussagenlogik: Funktions- und Prädikatszeichen
- PL-Formel wird über einem Universum *D* ausgewertet
- Konstanten *a,b,c* und Variablen *x,y,z* bezeichnen Werte aus *D*
- Funktionen *f(x,y), g(x)* werden als Funktionen in *D* interpretiert.
- Prädikate *P(x,y), Q(z)* werden abhängig von ihrer Belegung als *wahr* oder *falsch* ausgewertet

Prädikatenlogik: Syntax

$$
\begin{aligned}
Satz &\rightarrow Atom \mid \neg Satz \mid (Satz \wedge Satz) \\
&\mid (Satz \vee Satz) \mid (Satz \Rightarrow Satz) \mid (Satz \Leftrightarrow Satz) \\
&\mid \exists\, Variable\ Satz \mid \forall\, Variable\ Satz \\
Atom &\rightarrow true \mid false \mid LogVar \\
&\mid Prädikat \mid Term = Term \\
LogVar &\rightarrow P \mid Q \mid R \mid ... \\
Prädikat &\rightarrow P(Term, ...) \mid Q(Term, ...) \mid ... \\
Term &\rightarrow Konstante \mid Variable \mid Funktion \\
Funktion &\rightarrow f(Term, ...) \mid g(Term, ...) \mid ... \\
Konstante &\rightarrow a \mid b \mid c \mid ... \\
Variable &\rightarrow x \mid y \mid z \mid ...
\end{aligned}
$$

Prädikatenlogik: Semantik

- Belegung M ordnet jedem Symbol eine Bedeutung zu. Damit können Terme und Sätze ausgewertet werden.
- Der Satz $x = y$ ist genau dann *wahr*, wenn die Terme x und y identisch ausgewertet werden.
- Wahrheitsgehalt eines Prädikats ist
  - Abhängig von der Definition von P
  - Und der Belegung der Parameter: $P(x,y,...) / M := P(x/M, y/M ...)$
- $(\exists x\ \alpha)/M$ ist genau dann wahr, wenn es ein a in D gibt, so dass α / M wahr wird, wenn x = M(a) gilt.
- $(\forall x\ \alpha)/M$ ist genau dann wahr, wenn für jedes a in D gilt, dass α / M wahr wird, wenn x=M(a) gilt.
- Ansonsten wie bei AL

Prädikatenlogik

- Weiterführende Verfahren
  - Substitution
  - Resolution

  (werden im Skript vorgestellt)

Planungssprachen

- STRIPS
- ADL

Repräsentation von Plänen

Wie kann man Probleme so formulieren, dass

- ein Lösungsplan einfach zu erstellen ist?
- die Existenz einer Lösung bewiesen / widerlegt werden kann?

➔ Einschränkungsregeln für formelle Spezifikation von

- Zuständen
- Zielen
- Aktionen

- Beispiele solcher Regelsysteme: STRIPS, ADL

STRIPS

- „**ST**anford **R**esearch **I**nstitute **P**roblem **S**olver“
- Sprache für die Eingaben in das gleichnamige Planungssystem

- „Urvater“ vieler Planungssysteme (Stanford 1971)
- sehr einfach aufgebaut
- teilweise eingeschränkt

STRIPS: Zustände

Repräsentation von **Zuständen**:

- Konjunktion positiver aussagenlogischer Literale

$$Blau \wedge Rund$$

- Literale erster Ordnung (L1)

$$IstTasse(T_1) \wedge IstUntertasse(U_1) \wedge StehtAuf(T_1, U_1)$$

- L1 müssen funktions- und variablenfrei sein!

~~$IstTasse(x)$~~ ~~$IstTasse(ObjektAuf(U_1))$~~

- **Geschlossene Welt**:
  - Jeder Zustand ist vollständig bekannt.
  - Nicht im Zustand vorkommende Literale werden implizit als falsch angenommen!

STRIPS: Closed-World Assumption

- Keine Negationen im Zustand und Vorbedingungen, aber in Nachbedingung benötigt
- KEINE Negationen in **Vorbedingung**:

$b \Rightarrow$ *b* **muß** im Weltmodell vorkommen
a **kann** im Weltmodell vorkommen

- Negationen in **Nachbedingung**:

$\neg c \wedge d \Rightarrow$ **entferne** *c* aus dem Weltmodell
füge *d* dem Weltmodell **hinzu**

STRIPS: Ziele

Repräsentation von **Zielen**:

- Ziel: Teilweise spezifizierter Zustand
- Angegeben als Konjunktion von positiven Literalen

$$StehtAuf(T_1, U_1) \wedge StehtAuf(U_1, Tisch)$$

- Ein Zustand *S erfüllt* ein Ziel *Z*, wenn er alle Literale in *Z* enthält

$$StehtAuf(T_1, U_1) \wedge StehtAuf(U_1, Tisch) \wedge StehtAuf(Teller, Tisch)$$

erfüllt

$$StehtAuf(T_1, U_1) \wedge StehtAuf(U_1, Tisch)$$

STRIPS: Aktionen

Aktionen werden angegeben durch

- Aktionsname
- Parameter
- Vorbedingungen
- Effekte
- Beispiel:

$$A = (N_A, P_A, V_A, E_A)$$

$$\begin{aligned} N_A &= StelleAuf, \\ P_A &= (Obj_1, Obj_2), \\ V_A &= IstUntersatz(Obj_2) \wedge Auf(Obj_1, Tisch) \wedge (Obj_2, Tisch), \\ E_A &= \neg Auf(Obj_1, Tisch) \wedge Auf(Obj_1, Obj_2). \end{aligned}$$

STRIPS: Aktionen

- Alternative Schreibweise:

Aktion(*StelleAuf*(Obj_1, Obj_2),
  Vorbed: *IstUntersatz*(Obj_2) $\wedge$
  Auf(Obj_1, *Tisch*) $\wedge$
  Auf(Obj_2, *Tisch*),
  Effekt: $\neg$*Auf*(Obj_1, *Tisch*) $\wedge$
  Auf(Obj_1, Obj_2))

- Manchmal auch Effekt aufgeteilt in
  - *Additionsliste* (positive Literale)
  - *Deletionsliste* (negative Literale)

STRIPS: Semantik

Anwendbarkeit:

- Eine Aktion ist *anwendbar* auf allen Belegungen *M*, die die Vorbedingung V_A erfüllen

Ergebnis:

- Das *Ergebnis M´* der Ausführung einer Aktion *A* auf einer Belegung *M* erhält man durch
  - Entfernen aller negativen Literale des Effekts E_A aus *M*
  - Hinzufügen aller positiven Literale aus E_A zu *M*

STRIPS: Einschränkungen

Einschränkungen von STRIPS:

- Literale müssen *funktionsfrei* sein
  - endliche Grammatik
  - jede Aktion kann als endliche aussagenlogische Konjunktion dargestellt werden
  - Lösbarkeitsbeweis einfach
- Nur positive Literale / „*geschlossene Welt*“
  - einfachere Planung
  - aber: Falsch-Sachverhalte nur implizit
- Keine Quantisierung

➔ STRIPS-Aussagen oft lang und unübersichtlich

ADL

Action **D**escription **L**anguage:

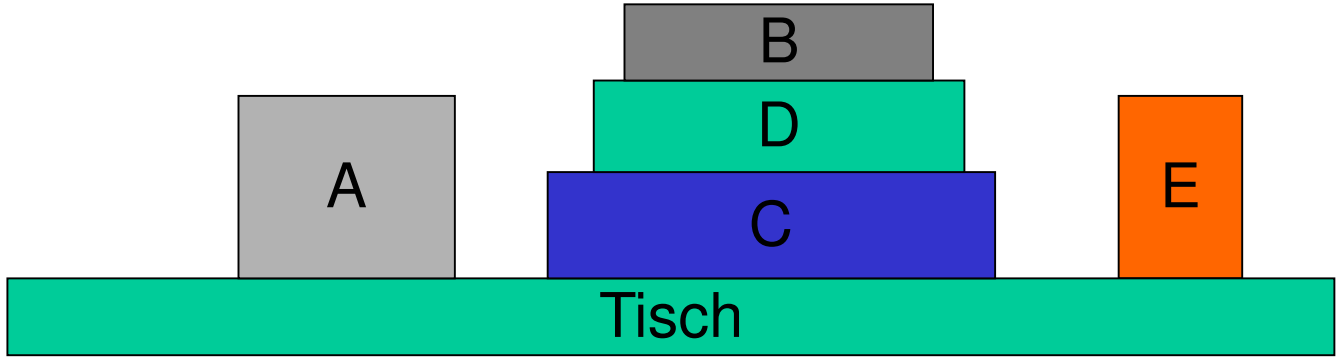
- Weiterentwicklung von STRIPS
- **„offene Welt“**: alle nicht angegebenen Literale gelten als unbekannt
- **negative** Literale und **Disjunktionen** erlaubt
- Effekt $\neg P \wedge Q$:
  - Füge $\neg P$ und Q hinzu
  - lösche P und $\neg Q$
- Gleichheits-Prädikat „=“ eingebaut
  - S*telleAuf*(T_1, T_1) nicht mehr möglich

Typüberprüfung

- Typüberprüfung nötig für Ausführbarkeit bestimmter Aktionen
- in STRIPS nur explizit (als Prädikat):
  IstUntersetzer(x)
  IstTasse(y)
- Häufig weggelassen (Schreibarbeit!), aber eigentlich nötig
- In ADL implizit möglich:

Aktion(*StelleAuf(*Obj_1*: Untersatz,* Obj_2*: Manipulierbar),*
Vorbed: *Auf(*Obj_1*, Tisch)* $\wedge$ *Auf(*Obj_2*, Tisch),*
Effekt: $\neg$*Auf(*Obj_2*, Tisch)* $\wedge$ *Auf(*Obj_2*,* Obj_1*))*

Beispiel: Blockwelt



- Klassisches Beispiel für Planungsprobleme
- Besteht aus
  - einem Tisch
  - *n* Quadern
- Je ein Quader kann auf einem anderen oder auf dem Tisch stehen
- Problem: Aktionsabfolge, um C auf E zu stellen

Beispiel: Blockwelt

- Verfügbare Prädikate (in STRIPS):
  - *Auf(b,x)*: Block *b* liegt auf *x* (Block oder Tisch)
  - *Frei(x)*: Nichts liegt auf *x* (Block oder Tisch)
- Verfügbare Aktionen (in STRIPS):

Aktion(*Bewege*(*b*,*x*,*y*),
Vorbed: $Auf(b,x) \wedge Frei(b) \wedge Frei(y)$,
Effekt: $Auf(b,y) \wedge Frei(x) \wedge \neg Auf(b,x) \wedge \neg Frei(y)$)

Aktion(*BewegeAufTisch*(*b*,*x*),
Vorbed: $Auf(b,x) \wedge Frei(b)$,
Effekt: $Auf(b,Tisch) \wedge Frei(x) \wedge \neg Auf(b,x)$)

Planungsstrategien

Suche im Zustandsraum - A*-Suche

Partial-Order-Planning

Planungsgraphen

Suche im Zustandsraum

- Intuitiver Ansatz: Von Startzustand ausgehend für alle ausführbaren Aktionen den daraus entstehenden Zustand ermitteln
- In den neuen Zuständen wiederum jeweils alle möglichen Aktionen ausführen
- Man erhält einen Baum, der alle erreichbaren Zustände enthält
- Solange ausführen, bis der gewünschte Zielzustand entsteht

➔ Aktionenfolge, die dorthin geführt hat, tatsächlich ausführen

Suche im Zustandsraum

- Suche im Zustandsraum ist **einfach**:
  - Keine Funktionssymbole
    - ➔ endlicher Zustandsraum
    - ➔ Standard-Algorithmen zur Baumsuche
  - Transformation der Vorbedingungen
    - ➔ Effekte bijektiv
    - ➔Suche auch rückwärts möglich

- **ABER:**
  - Zustandsraum oft *sehr* groß
  - Naive Suche sehr ineffizient
  - Benötigt:
    - gute Heuristik oder
    - Segmentierung in Teilprobleme

Suche beschleunigen

- Suchbaum hat schlimmstenfalls 2^n Knoten (n=Anzahl der Zustandsvariablen)
- Naive Suche nur für sehr kleine Instanzen praktikabel
- Suche muss für realistisch komplexe Situationen effizienter durchgeführt werden
- Lösungsansatz: gezielt suchen
- **Heuristik** verwenden: „welche Suchrichtung scheint am ehesten zum Ziel zu führen?“

Heuristikbasierte Suche

Heuristik:

- Eine Funktion, die für jeden Zustand einen **optimistischen** Schätzwert der „Entfernung“ zum Ziel liefert
- Darf Distanz unter-, aber nicht überschätzen!

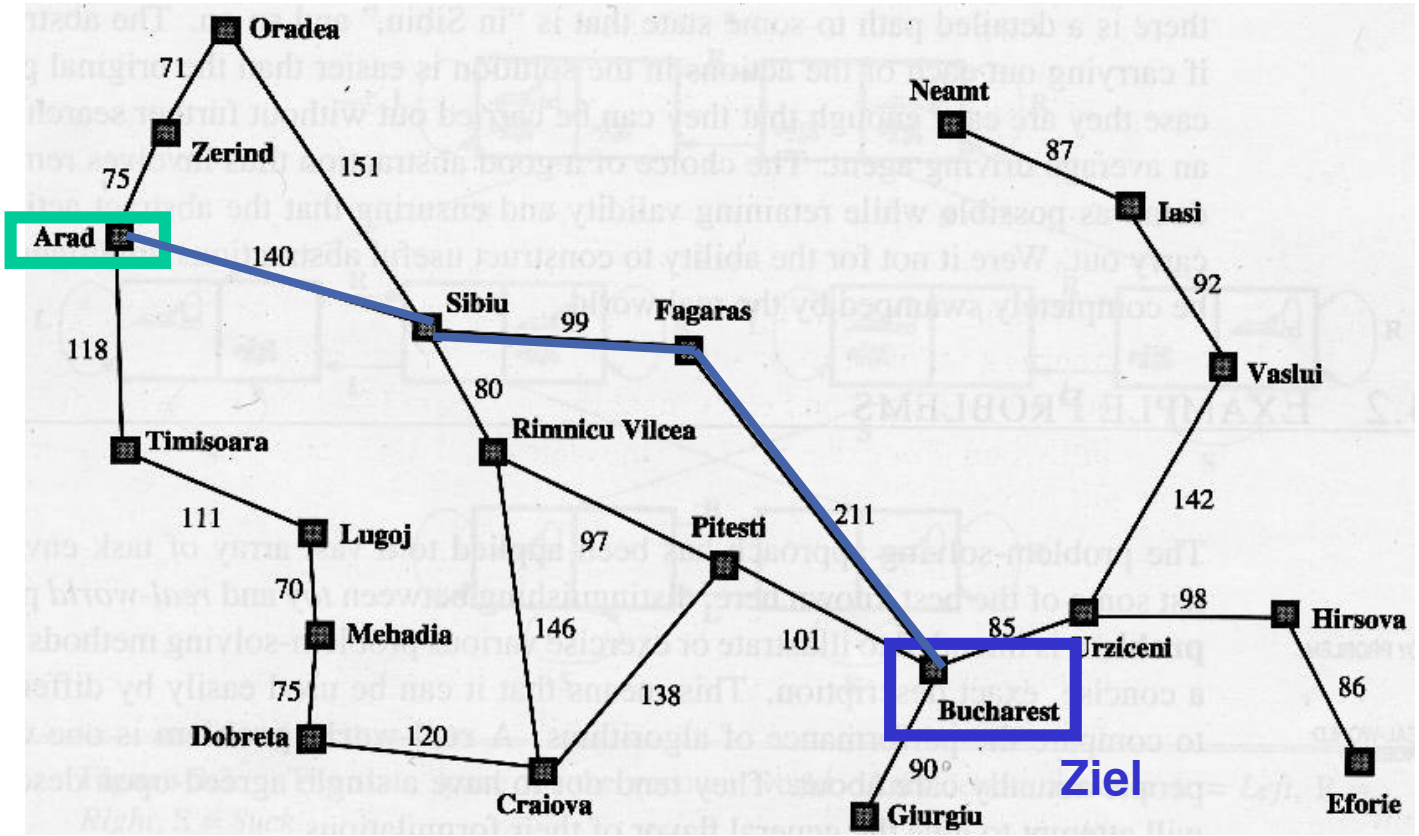
Einfacher Ansatz (**Greedy**-Algorithmus):

- wenn Heuristik zur Verfügung steht: immer von dem Zustand aus weitersuchen, der laut Heuristik dem Ziel am nächsten ist
- Kann Suche deutlich beschleunigen, wenn die Heuristik gut ist, kann aber auch zu großen Umwegen führen
- Liefert i. A. nicht den kürzesten Weg zum Ziel!

Beispiel: Greedy

Straßenkarte von Rumänien

Arad	366	Hirsova	151	Rimnicu Vilcea	193
Bukarest	0	Iasi	226	Sibiu	253
Craiova	160	Lugoj	244	Timisoara	329
Dobreta	242	Mehadia	241	Urziceni	80
Eforie	161	Neamt	234	Vaslui	199
Fagaras	176	Oradea	380	Zerind	374
Giurgiu	77	Pitesti	100		



A* - Suche

A*-Algorithmus

- Allgemeiner Algorithmus für die heuristikgestützte Suche in Graphen
- Liefert kürzesten Weg zum Ziel
- Normalerweise sehr effizient, abhängig von der Güte der Heuristik
- Expandiert den Knoten, für den die Summe

  (*geschätzte Entfernung zum Ziel*)

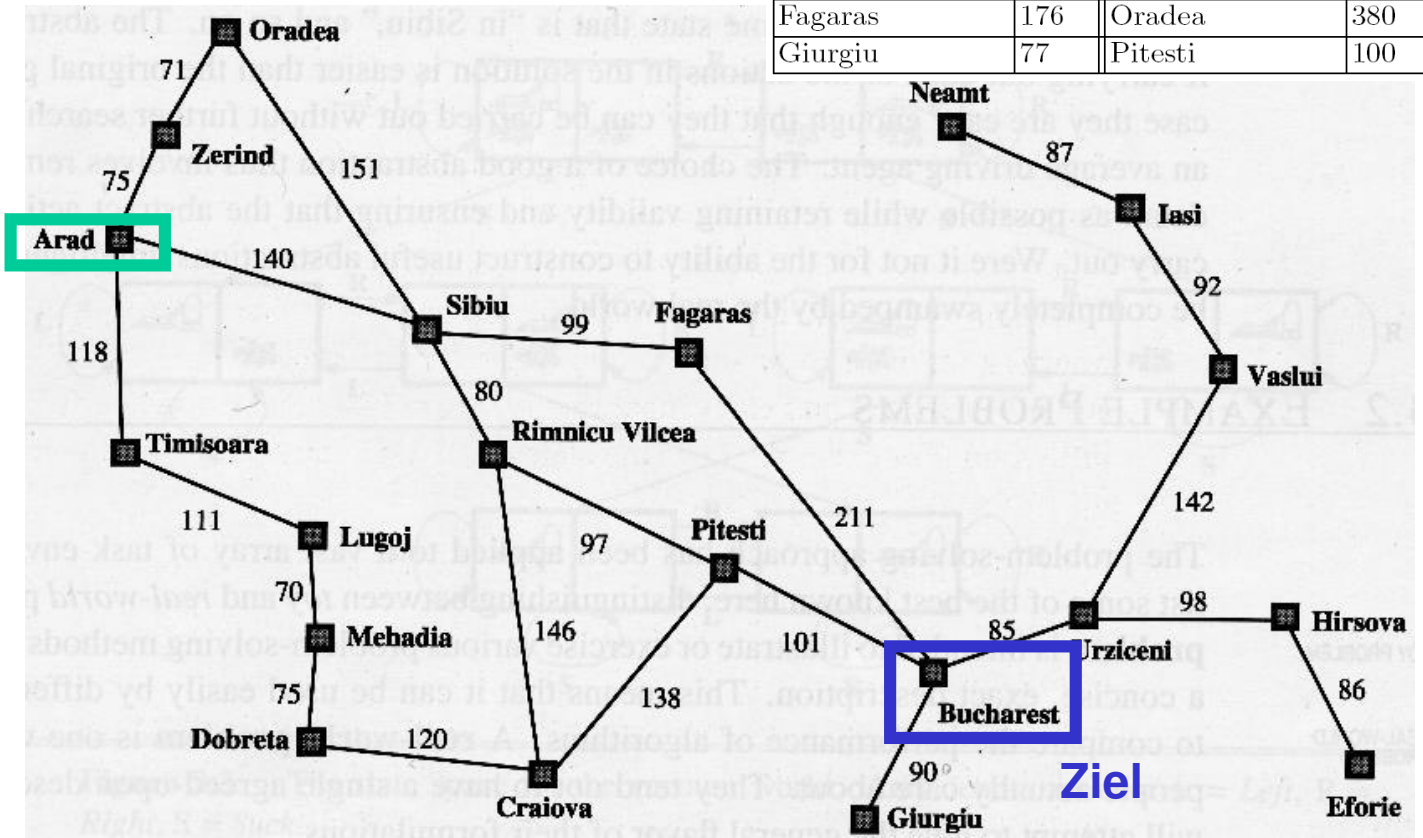
  + (*bisher zurückgelegte Strecke vom Start*)

  minimal ist.
- Es werden immer alle nicht-expandierten Knoten berücksichtigt (nicht nur der aktuelle Ast)

Beispiel A*-Algorithmus

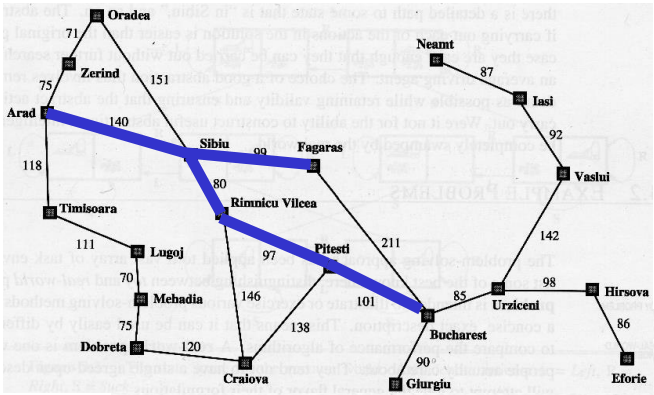
Straßenkarte von Rumänien

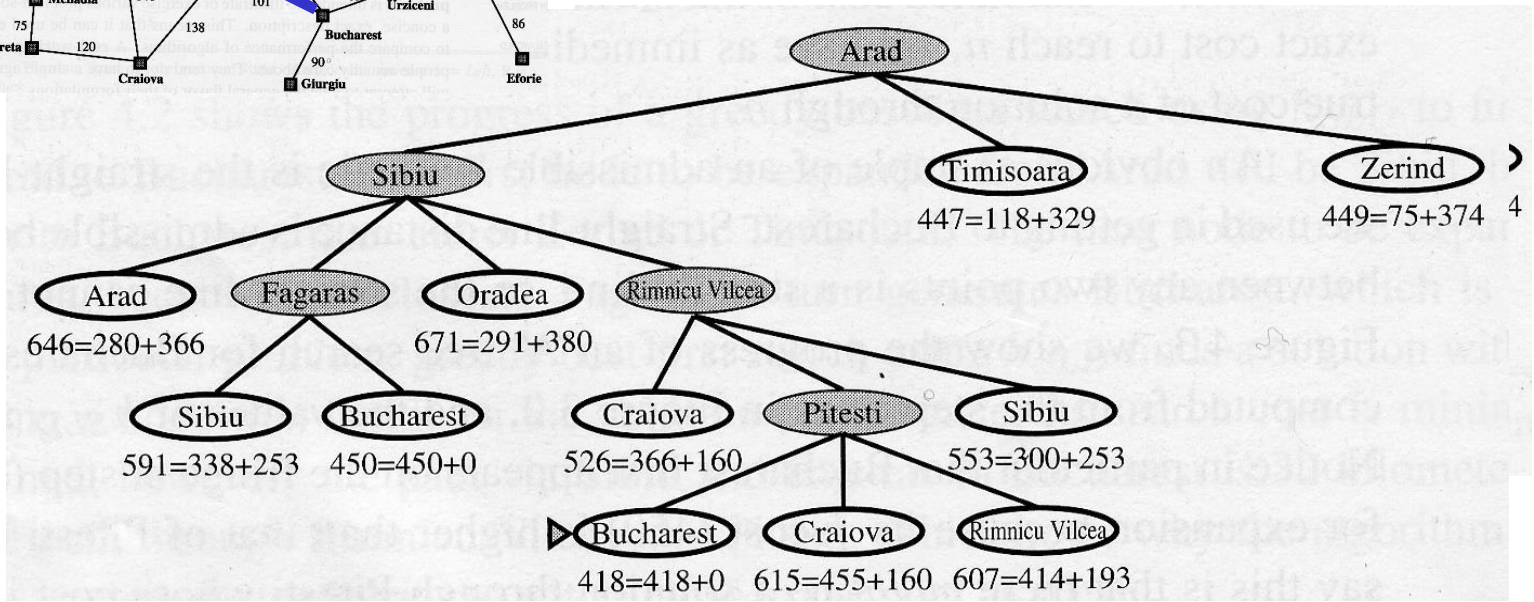
Arad	366	Hirsova	151	Rimnicu Vilcea	193
Bukarest	0	Iasi	226	Sibiu	253
Craiova	160	Lugoj	244	Timisoara	329
Dobreta	242	Mehadia	241	Urziceni	80
Eforie	161	Neamt	234	Vaslui	199
Fagaras	176	Oradea	380	Zerind	374
Giurgiu	77	Pitesti	100		



Beispiel A*-Algorithmus

Arad	366	Hirsova	151	Rimnicu Vilcea	193
Bukarest	0	Iasi	226	Sibiu	253
Craiova	160	Lugoj	244	Timisoara	329
Dobreta	242	Mehadia	241	Urziceni	80
Eforie	161	Neamt	234	Vaslui	199
Fagaras	176	Oradea	380	Zerind	374
Giurgiu	77	Pitesti	100		





Woher kommt die Heuristik?

- **Allgemeine** Heuristik für Planungsprobleme **existiert nicht**
- Für einzelne Aufgaben gibt es spezifische Heuristiken
  - z.B. bei der Routenplanung die euklidische Distanz der Knoten

Suche im Zustandsraum:

- Spontane Idee: Anzahl noch zu erfüllender Literale für Zielzustand
- Aber: Wenn eine Aktion mehrere Literale erfüllt, überschätzt die Heuristik evtl. den verbleibenden Aufwand

  ➔ ungültige Heuristik (darf unter-, aber nicht überschätzen)
- Der **Planungsgraph** liefert meist gute Heuristik (später mehr dazu)