

# Artificial Intelligence

Christoph Benz Müller and Raul Rojas

Freie Universität Berlin

Block Lecture, SS 2014

# (Propositional) Hornlogic

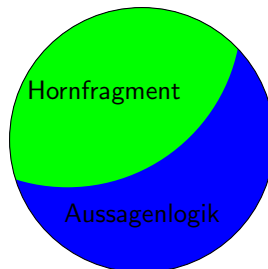
## Erfüllbarkeit/Gültigkeit

- ▶ entscheidbar
- ▶ bisher kein “effektiver” Algorithmus bekannt (NP-vollständig)



Erfüllbarkeit/Gültigkeit

- ▶ entscheidbar
- ▶ “effektiver”  
Algorithmus (P)

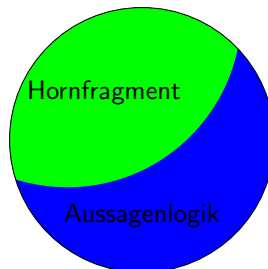


Erfüllbarkeit/Gültigkeit

- ▶ entscheidbar
- ▶ “effektiver”  
Algorithmus (P)

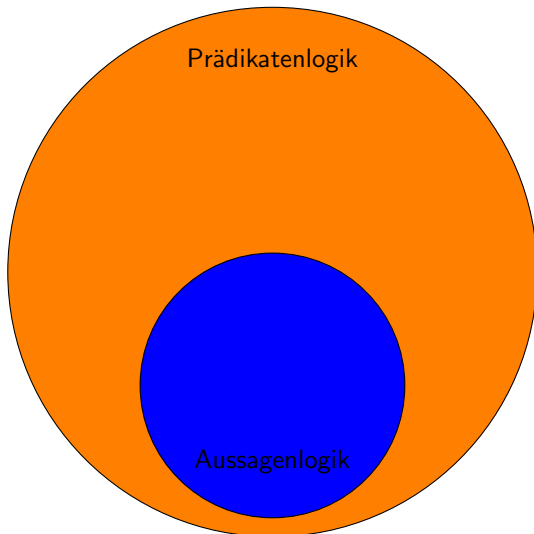
$$P \stackrel{?}{=} NP$$

(one million dollar question)



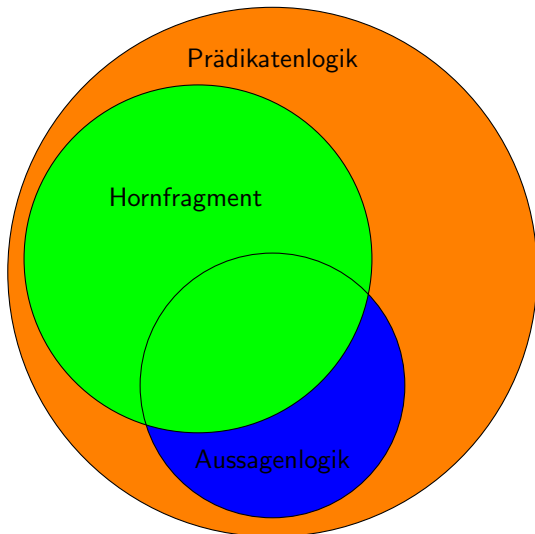
Erfüllbarkeit/Gültigkeit

- ▶ **unentscheidbar**  
(semi-entscheidbar)
- ▶ **ineffektive**  
Suchalgorithmen



Erfüllbarkeit/Gültigkeit

- ▶ **unentscheidbar**  
(semi-entscheidbar)
- ▶ effektive  
Suchalgorithmen  
(Prolog)



## Syntax der Aussagenlogik (AL)

$$s, t ::= P \mid \neg s \mid s \vee t \mid s \wedge t \mid s \Rightarrow t \mid \perp \mid \top$$

## Semantik

### ► Interpretationsfunktion

$$I : AL \longrightarrow \{T, F\}$$

- atomare Aussagen  $P$ :

wähle  $I(P) \in \{T, F\}$

-  $I(\neg s), I(s \vee t), I(s \wedge t), I(s \Rightarrow t), I(\perp), I(\top)$  festgelegt wie folgt

$s$	$t$	$\neg s$	$s \vee t$	$s \wedge t$	$s \Rightarrow t$	$\perp$	$\top$
$T$	$T$	$F$	$T$	$T$	$T$	$F$	$T$
$T$	$F$	$F$	$T$	$F$	$F$	$F$	$T$
$F$	$T$	$T$	$T$	$F$	$T$	$F$	$T$
$F$	$F$	$T$	$F$	$F$	$F$	$F$	$T$



## Syntax der Aussagenlogik (AL)

$$s, t ::= P \mid \neg s \mid s \vee t \mid s \wedge t \mid s \Rightarrow t \mid \perp \mid \top$$

## Semantik

### ► Interpretationsfunktion

$$I : AL \longrightarrow \{T, F\}$$

- atomare Aussagen  $P$ :

wähle  $I(P) \in \{T, F\}$

-  $I(\neg s), I(s \vee t), I(s \wedge t), I(s \Rightarrow t), I(\perp), I(\top)$  festgelegt wie folgt

$s$	$t$	$\neg s$	$s \vee t$	$s \wedge t$	$s \Rightarrow t$	$\perp$	$\top$
$T$	$T$	$F$	$T$	$T$	$T$	$F$	$T$
$T$	$F$	$F$	$T$	$F$	$F$	$F$	$T$
$F$	$T$	$T$	$T$	$F$	$T$	$F$	$T$
$F$	$F$	$T$	$F$	$F$	$F$	$F$	$T$

definiert als

$\neg s \vee t$



## Syntax der Aussagenlogik (AL)

$$s, t ::= P \mid \neg s \mid s \vee t \mid s \wedge t \mid s \Rightarrow t \mid \perp \mid \top$$

## Semantik

### ► Interpretationsfunktion

$$I : AL \longrightarrow \{T, F\}$$

- atomare Aussagen  $P$ :

wähle  $I(P) \in \{T, F\}$

-  $I(\neg s), I(s \vee t), I(s \wedge t), I(s \Rightarrow t), I(\perp), I(\top)$  festgelegt wie folgt

$s$	$t$	$\neg s$	$s \vee t$	$s \wedge t$	$s \Rightarrow t$	$\perp$	$\top$
$T$	$T$	$F$	$T$	$T$	$T$	$F$	$T$
$T$	$F$	$F$	$T$	$F$	$F$	$F$	$T$
$F$	$T$	$T$	$T$	$F$	$T$	$F$	$T$
$F$	$F$	$T$	$F$	$F$	$F$	$F$	$T$

Definition Hornklausel:

(alternative Sichtweise)

$$P \vee \neg Q_1 \vee \dots \vee \neg Q_n$$

$$P \Leftarrow Q_1 \wedge \dots \wedge Q_n$$

Definition Hornklausel:

(alternative Sichtweise)

Bedingungsteil  $n \geq 0$

$$P \vee \neg Q_1 \vee \dots \vee \neg Q_n$$

max. ein positives Literal (Konklusion)

Bedingungsteil  $n \geq 0$

$$P \Leftarrow Q_1 \wedge \dots \wedge Q_n$$

Definition Hornklausel:

(alternative Sichtweise)

$$P \vee \overbrace{\neg Q_1 \vee \dots \vee \neg Q_n}^{\substack{\text{Bedingungsteil} \\ n \geq 0}} \quad \quad \quad P \Leftarrow \overbrace{Q_1 \wedge \dots \wedge Q_n}^{\substack{\text{Bedingungsteil} \\ n \geq 0}}$$

max. ein positives Literal (Konklusion)

Es gibt drei Typen von Hornklauseln (Beispiele)

$C$

*Fakt* : '  $C$  gilt '

$C \Leftarrow \top$

$A \vee \neg B \vee \neg D$

*Regel* : '  $A$  gilt falls  $B$  und  $D$  gelten '

$A \Leftarrow B \wedge D$

$\neg B \vee \neg D$

*Ziel* : ' Gelten  $B$  und  $D$  ? '

$\perp \Leftarrow B \wedge D$

Definition Hornklausel:

(alternative Sichtweise)

$$P \vee \underbrace{\neg Q_1 \vee \dots \vee \neg Q_n}_{\substack{\text{Bedingungsteil} \\ n \geq 0}}$$

max. ein positives Literal (Konklusion)

$$P \Leftarrow \underbrace{Q_1 \wedge \dots \wedge Q_n}_{\substack{\text{Bedingungsteil} \\ n \geq 0}}$$

Es gibt drei Typen von Hornklauseln (Beispiele)

$C$

*Fakt* : 'C gilt'

$C \Leftarrow \top$

$A \vee \neg B \vee \neg D$

*Regel* : 'A gilt falls B und D gelten'

$A \Leftarrow B \wedge D$

$\neg B \vee \neg D$

*Ziel* : 'Gelten B und D?'

$\perp \Leftarrow B \wedge D$

$A \vee C \vee \neg B \vee \neg D$

$A \vee C \Leftarrow B \wedge D$

Definition Hornklausel:

(alternative Sichtweise)



Es gibt drei Typen von Hornklauseln (Beispiele)

$C$

*Fakt* : '  $C$  gilt '

$C \Leftarrow \top$

$A \vee \neg B \vee \neg D$

*Regel* : '  $A$  gilt falls  $B$  und  $D$  gelten '

$A \Leftarrow B \wedge D$

$\neg B \vee \neg D$

*Ziel* : ' Gelten  $B$  und  $D$ ? '

$\perp \Leftarrow B \wedge D$

~~$A \vee D \vee \neg B \vee \neg C$~~

*verboten*

~~$A \vee D \Leftarrow B \wedge C$~~

## Definition Hornformel:

Hornformel = Konjunktionen von Hornklauseln



Definition Hornformel:

Hornformel = Konjunktionen von Hornklauseln

Beispiel

$$\begin{aligned} & (C) \\ \wedge & (D) \\ \wedge & (B \vee \neg C) \\ \wedge & (A \vee \neg B \vee \neg D) \\ \wedge & (\neg A \vee \neg D) \end{aligned}$$

## Definition Hornformel:

Hornformel = Konjunktionen von Hornklauseln

## Beispiel

$$\begin{aligned} & \{C\} \\ \wedge & \{D\} \\ \wedge & \{B, \neg C\} \\ \wedge & \{A, \neg B, \neg D\} \\ \wedge & \{\neg A, \neg D\} \end{aligned}$$

## Definition Hornformel:

Hornformel = Konjunktionen von Hornklauseln

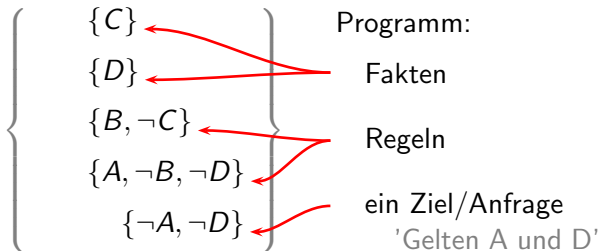
Beispiel

$$\left\{ \begin{array}{l} \{C\} \\ \{D\} \\ \{B, \neg C\} \\ \{A, \neg B, \neg D\} \\ \{\neg A, \neg D\} \end{array} \right\}$$

## Definition Hornformel:

Hornformel = Konjunktionen von Hornklauseln

Beispiel



Definition Resolution:  $\{\neg P, N_1, \dots, N_n\} \quad \{P, M_1, \dots, M_m\}$

Definition Resolution:



$$\frac{\{\neg P, N_1, \dots, N_n\} \quad \{P, M_1, \dots, M_m\}}{\text{komplementär}}$$

Definition Resolution:

$$\frac{\{\neg P, N_1, \dots, N_n\} \quad \{P, M_1, \dots, M_m\}}{\{N_1, \dots, N_n, M_1, \dots, M_m\}}$$

Definition Resolution: 
$$\frac{\{\neg P, N_1, \dots, N_n\} \quad \{P, M_1, \dots, M_m\}}{\{N_1, \dots, N_n, M_1, \dots, M_m\}}$$

Beispiele und (generelle) Beobachtungen:

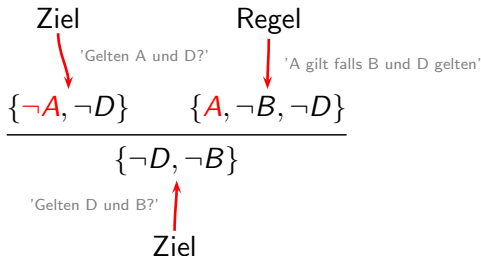
Ziel	Regel
 'Gelten A und D?'	 'A gilt falls B und D gelten'
$\{\neg A, \neg D\}$	$\{A, \neg B, \neg D\}$
<hr/>	



Definition Resolution:

$$\frac{\{\neg P, N_1, \dots, N_n\} \quad \{P, M_1, \dots, M_m\}}{\{N_1, \dots, N_n, M_1, \dots, M_m\}}$$

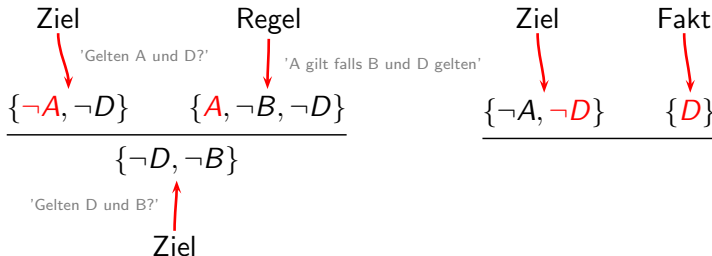
Beispiele und (generelle) Beobachtungen:



Definition Resolution:

$$\frac{\{\neg P, N_1, \dots, N_n\} \quad \{P, M_1, \dots, M_m\}}{\{N_1, \dots, N_n, M_1, \dots, M_m\}}$$

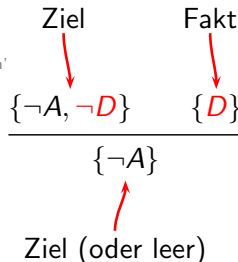
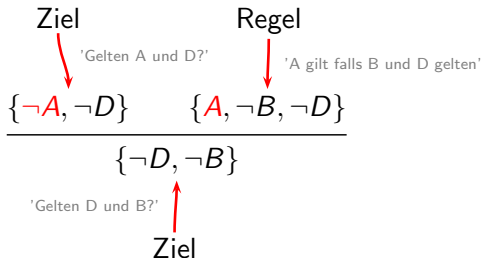
Beispiele und (generelle) Beobachtungen:



Definition Resolution:

$$\frac{\{\neg P, N_1, \dots, N_n\} \quad \{P, M_1, \dots, M_m\}}{\{N_1, \dots, N_n, M_1, \dots, M_m\}}$$



Beispiele und (generelle) Beobachtungen:



Definition Resolution:

$$\frac{\{\neg P, N_1, \dots, N_n\} \quad \{P, M_1, \dots, M_m\}}{\{N_1, \dots, N_n, M_1, \dots, M_m\}}$$

Beispiele und (generelle) Beobachtungen:

Regel	Regel
	
$\{A, \neg B, \neg D\}$	$\{B, \neg C\}$
<hr/>	

Definition Resolution: 
$$\frac{\{\neg P, N_1, \dots, N_n\} \quad \{P, M_1, \dots, M_m\}}{\{N_1, \dots, N_n, M_1, \dots, M_m\}}$$

Beispiele und (generelle) Beobachtungen:

Regel  $\swarrow$

$\{A, \neg B, \neg D\}$        $\{B, \neg C\}$

---

$\{A, \neg D, \neg C\}$

$\nwarrow$  Regel

Definition Resolution: 
$$\frac{\{\neg P, N_1, \dots, N_n\} \quad \{P, M_1, \dots, M_m\}}{\{N_1, \dots, N_n, M_1, \dots, M_m\}}$$

Beispiele und (generelle) Beobachtungen:

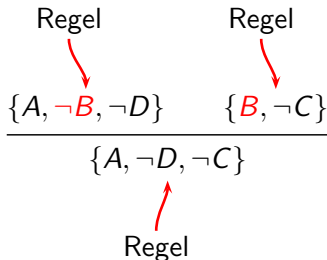
Regel                      Regel

$\{A, \neg B, \neg D\}$        $\{B, \neg C\}$

---

$\{A, \neg D, \neg C\}$

Regel



Regel                      Fakt

$\{A, \neg B, \neg D\}$        $\{D\}$

---

Definition Resolution: 
$$\frac{\{\neg P, N_1, \dots, N_n\} \quad \{P, M_1, \dots, M_m\}}{\{N_1, \dots, N_n, M_1, \dots, M_m\}}$$

Beispiele und (generelle) Beobachtungen:

Regel  $\swarrow$   
 $\{A, \neg B, \neg D\}$   
 Regel  $\swarrow$   
 $\{B, \neg C\}$   


---

 $\{A, \neg D, \neg C\}$   
 $\nwarrow$   
 Regel

Regel  $\swarrow$   
 $\{A, \neg B, \neg D\}$   
 Fakt  $\swarrow$   
 $\{D\}$   


---

 $\{A, \neg B\}$   
 $\nwarrow$   
 Regel (oder Fakt)

## Algorithmus: SLD-Resolution

```
while Ziel  $\neq \emptyset$  do
  — wähle Literal L und komplementäre
    Regel/Fakt K
  — if kein K then backtrack/'No'
  — else Ziel := resolviere(Ziel,K)
done
return 'Yes';
```

Programm:

```
{C}
{D}
{B,  $\neg C$ }
{A,  $\neg B$ ,  $\neg D$ }
{ $\neg A$ ,  $\neg D$ }
```

{ $\neg A$ ,  $\neg D$ }

**S** Selektion (& Backtracking)

**L** Linear

**D** Definite Klauseln

(haben genau ein pos. Literal  
→ in jedem Schritt beteiligt)



## Algorithmus: SLD-Resolution

```
while Ziel  $\neq \emptyset$  do
  — wähle Literal L und komplementäre
    Regel/Fakt K
  — if kein K then backtrack/'No'
  — else Ziel := resolviere(Ziel,K)
done
return 'Yes';
```

Programm:

```
{C}
{D}
{B,  $\neg C$ }
{A,  $\neg B$ ,  $\neg D$ }
{ $\neg A$ ,  $\neg D$ }
```

{ $\neg$ **A**,  $\neg D$ }

**S** Selektion (& Backtracking)

**L** Linear

**D** Definite Klauseln

(haben genau ein pos. Literal  
→ in jedem Schritt beteiligt)

## Algorithmus: SLD-Resolution

```
while Ziel  $\neq \emptyset$  do
  – wähle Literal L und komplementäre
    Regel/Fakt K
  — if kein K then backtrack/'No'
  — else Ziel := resolviere(Ziel,K)
done
return 'Yes';
```

Programm:

```
{C}
{D}
{B,  $\neg C$ }
{A,  $\neg B$ ,  $\neg D$ }
{ $\neg A$ ,  $\neg D$ }
```

{ $\neg$ **A**,  $\neg D$ }    {**A**,  $\neg B$ ,  $\neg D$ }

**S** Selektion (& Backtracking)

**L** Linear

**D** Definite Klauseln

(haben genau ein pos. Literal  
→ in jedem Schritt beteiligt)


## Algorithmus: SLD-Resolution

```
while Ziel  $\neq \emptyset$  do
  – wähle Literal L und komplementäre
    Regel/Fakt K
  — if kein K then backtrack/'No'
  — else Ziel := resolviere(Ziel,K)
done
return 'Yes';
```

Programm:

```
{C}
{D}
{B,  $\neg C$ }
{A,  $\neg B$ ,  $\neg D$ }
{ $\neg A$ ,  $\neg D$ }
```

$\{\neg A, \neg D\}$      $\{A, \neg B, \neg D\}$



$\{\neg D, \neg B\}$

**S** Selektion (& Backtracking)

**L** Linear

**D** Definite Klauseln

(haben genau ein pos. Literal  
→ in jedem Schritt beteiligt)


## Algorithmus: SLD-Resolution

```
while Ziel  $\neq \emptyset$  do
  – wähle Literal L und komplementäre
    Regel/Fakt K
  — if kein K then backtrack/'No'
  — else Ziel := resolviere(Ziel,K)
done
return 'Yes';
```

Programm:

```
{C}
{D}
{B,  $\neg C$ }
{A,  $\neg B$ ,  $\neg D$ }
{ $\neg A$ ,  $\neg D$ }
```

$\{\neg A, \neg D\}$      $\{A, \neg B, \neg D\}$



$\{\neg D, \neg B\}$

**S** Selektion (& Backtracking)

**L** Linear

**D** Definite Klauseln

(haben genau ein pos. Literal  
→ in jedem Schritt beteiligt)


## Algorithmus: SLD-Resolution

```
while Ziel  $\neq \emptyset$  do
  – wähle Literal L und komplementäre
    Regel/Fakt K
  — if kein K then backtrack/'No'
  — else Ziel := resolviere(Ziel,K)
done
return 'Yes';
```

Programm:

```
{C}
{D}
{B,  $\neg C$ }
{A,  $\neg B$ ,  $\neg D$ }
{ $\neg A$ ,  $\neg D$ }
```

$\{\neg A, \neg D\}$      $\{A, \neg B, \neg D\}$   
↓  
 $\{\neg D, \neg B\}$      $\{D\}$



**S** Selektion (& Backtracking)

**L** Linear

**D** Definite Klauseln  
(haben genau ein pos. Literal  
→ in jedem Schritt beteiligt)

## Algorithmus: SLD-Resolution

```
while Ziel  $\neq \emptyset$  do
  – wähle Literal L und komplementäre
    Regel/Fakt K
  — if kein K then backtrack/'No'
  — else Ziel := resolviere(Ziel,K)
done
return 'Yes';
```

**S** Selektion (& Backtracking)

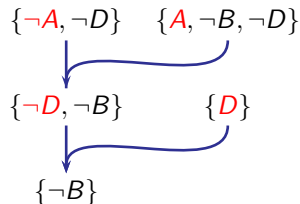
**L** Linear

**D** Definite Klauseln

(haben genau ein pos. Literal  
→ in jedem Schritt beteiligt)

Programm:

```
{C}
{D}
{B,  $\neg C$ }
{A,  $\neg B$ ,  $\neg D$ }
{ $\neg A$ ,  $\neg D$ }
```



## Algorithmus: SLD-Resolution

```
while Ziel  $\neq \emptyset$  do
  – wähle Literal L und komplementäre
    Regel/Fakt K
  — if kein K then backtrack/'No'
  — else Ziel := resolviere(Ziel,K)
done
return 'Yes';
```

**S** Selektion (& Backtracking)

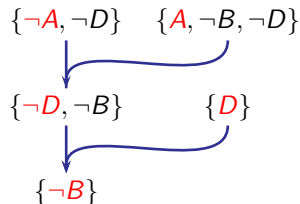
**L** Linear

**D** Definite Klauseln

(haben genau ein pos. Literal  
→ in jedem Schritt beteiligt)

Programm:

```
{C}
{D}
{B,  $\neg C$ }
{A,  $\neg B$ ,  $\neg D$ }
{ $\neg A$ ,  $\neg D$ }
```



## Algorithmus: SLD-Resolution

```
while Ziel  $\neq \emptyset$  do
  — wähle Literal L und komplementäre
    Regel/Fakt K
  — if kein K then backtrack/'No'
  — else Ziel := resolviere(Ziel,K)
done
return 'Yes';
```

**S** Selektion (& Backtracking)

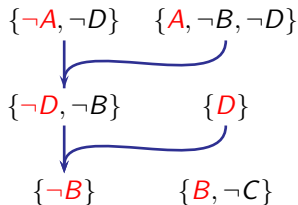
**L** Linear

**D** Definite Klauseln

(haben genau ein pos. Literal  
→ in jedem Schritt beteiligt)

Programm:

```
{C}
{D}
{B, ¬C}
{A, ¬B, ¬D}
{¬A, ¬D}
```





## Algorithmus: SLD-Resolution

```
while Ziel  $\neq \emptyset$  do
  — wähle Literal L und komplementäre
    Regel/Fakt K
  — if kein K then backtrack/'No'
  — else Ziel := resolviere(Ziel,K)
done
return 'Yes';
```

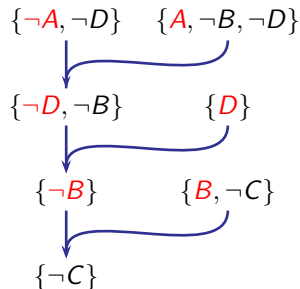
**S** Selektion (& Backtracking)

**L** Linear

**D** Definite Klauseln  
(haben genau ein pos. Literal  
→ in jedem Schritt beteiligt)

Programm:

```
{C}
{D}
{B, ¬C}
{A, ¬B, ¬D}
{¬A, ¬D}
```



## Algorithmus: SLD-Resolution

```
while Ziel  $\neq \emptyset$  do
  — wähle Literal L und komplementäre
    Regel/Fakt K
  — if kein K then backtrack/'No'
  — else Ziel := resolviere(Ziel,K)
done
return 'Yes';
```

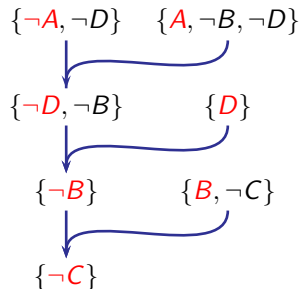
**S** Selektion (& Backtracking)

**L** Linear

**D** Definite Klauseln  
(haben genau ein pos. Literal  
→ in jedem Schritt beteiligt)

Programm:

```
{C}
{D}
{B, ¬C}
{A, ¬B, ¬D}
{¬A, ¬D}
```



## Algorithmus: SLD-Resolution

```
while Ziel  $\neq \emptyset$  do
  — wähle Literal L und komplementäre
    Regel/Fakt K
  — if kein K then backtrack/'No'
  — else Ziel := resolviere(Ziel,K)
done
return 'Yes';
```

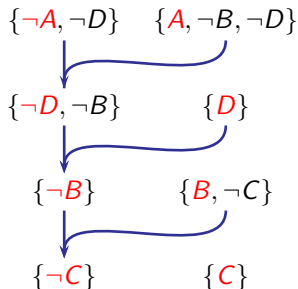
**S** Selektion (& Backtracking)

**L** Linear

**D** Definite Klauseln  
(haben genau ein pos. Literal  
→ in jedem Schritt beteiligt)

Programm:

```
{C}
{D}
{B, ¬C}
{A, ¬B, ¬D}
{¬A, ¬D}
```



## Algorithmus: SLD-Resolution

```
while Ziel  $\neq \emptyset$  do
  – wähle Literal L und komplementäre
    Regel/Fakt K
  — if kein K then backtrack/'No'
  — else Ziel := resolviere(Ziel,K)
done
return 'Yes';
```

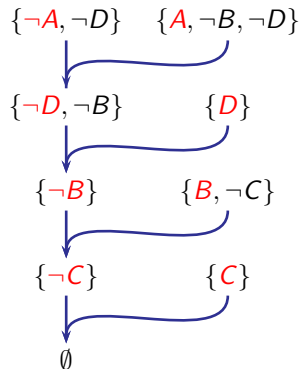
**S** Selektion (& Backtracking)

**L** Linear

**D** Definite Klauseln  
(haben genau ein pos. Literal  
→ in jedem Schritt beteiligt)

Programm:

```
{C}
{D}
{B, ¬C}
{A, ¬B, ¬D}
{¬A, ¬D}
```



## Algorithmus: SLD-Resolution

```
while Ziel  $\neq \emptyset$  do
  – wähle Literal L und komplementäre
    Regel/Fakt K
  — if kein K then backtrack/'No'
  — else Ziel := resolviere(Ziel,K)
done
return 'Yes';
```

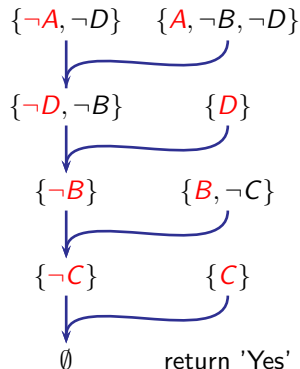
**S** Selektion (& Backtracking)

**L** Linear

**D** Definite Klauseln  
(haben genau ein pos. Literal  
→ in jedem Schritt beteiligt)

Programm:

```
{C}
{D}
{B, ¬C}
{A, ¬B, ¬D}
{¬A, ¬D}
```



## Algorithmus: SLD-Resolution

```
while Ziel  $\neq \emptyset$  do
  — wähle Literal L und komplementäre
    Regel/Fakt K
  — if kein K then backtrack/'No'
  — else Ziel := resolviere(Ziel,K)
done
return 'Yes';
```

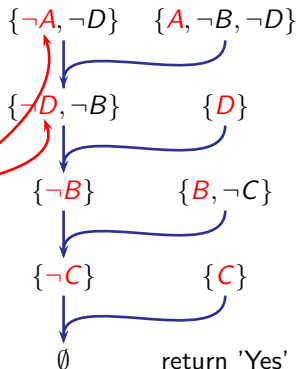
**S** Selektion (& Backtracking)

**L** Linear

**D** Definite Klauseln  
(haben genau ein pos. Literal  
→ in jedem Schritt beteiligt)

Programm:

```
{C}
{D}
{B,  $\neg C$ }
{A,  $\neg B, \neg D$ }
{ $\neg A, \neg D$ }
```



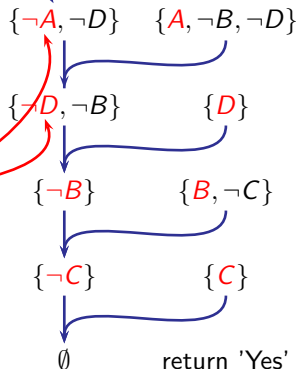
## Algorithmus: SLD-Resolution

```
while Ziel  $\neq \emptyset$  do
  — wähle Literal L und komplementäre
    Regel/Fakt K
  — if kein K then backtrack/'No'
  — else Ziel := resolviere(Ziel,K)
done
return 'Yes';
```

- S** Selektion (& Backtracking)
- L** Linear
- D** Definite Klauseln  
(haben genau ein pos. Literal  
→ in jedem Schritt beteiligt)

Programm:

```
{C}
{D}
{B,  $\neg C$ }
{A,  $\neg B, \neg D$ }
{ $\neg A, \neg D$ }
```



## Algorithmus: SLD-Resolution

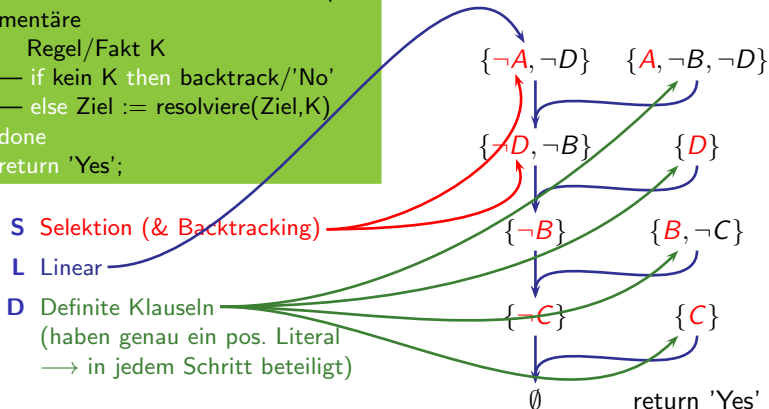
```

while Ziel  $\neq \emptyset$  do
  — wähle Literal L und komplementäre
    Regel/Fakt K
  — if kein K then backtrack/'No'
  — else Ziel := resolviere(Ziel,K)
done
return 'Yes';
    
```

Programm:

```

{C}
{D}
{B,  $\neg C$ }
{A,  $\neg B, \neg D$ }
{ $\neg A, \neg D$ }
    
```





- ▶ Hornlogik-Fragment der Prädikatenlogik
  - ▶ andere Algorithmen: Markierungsalgorithmus, Gentzenkalkül,  
...
  - ▶ Vollständigkeit & Korrektheit der Verfahren
  - ▶ Komplexität der Verfahren
- ▶ Hornlogik-Fragment der Prädikatenlogik erster Stufe
  - ▶ ...
  - ▶ PROLOG
  - ▶ ...
- ▶ Hornlogik-Fragment der Logik höherer Stufe
  - ▶ ...
  - ▶  $\lambda$ -PROLOG
  - ▶ ...