

## Deduktion

- Menge von Ausdrücken der Aussagenlogik oder der Prädikatenlogik beschreibt einen bestimmten Sachverhalt, quasi eine "Theorie" des Anwendungsbereiches. Was folgt logisch aus dieser Theorie?
- Deduktion: aus wahren Aussagen andere notwendigerweise wahre Aussagen ableiten
- Beispiele
  - Wenn es regnet, dann wird die Strasse nass.*  
*Es regnet.*  
-----  
*Die Strasse wird nass.*
  - Vögel können fliegen.*  
*Tweetie ist ein Vogel.*  
-----  
*Tweetie kann fliegen.*
- verallgemeinert zur universellen Schlussregel "modus ponens"
  - Wenn A gilt, dann gilt B.*  
*A gilt.*  
-----  
*B gilt.*
- Wie kann man diese und andere Schlussregeln formalisieren?

Deduktion: Resolution 1

## Syntaktische Folgerungsbeziehung

- nach festen syntaktischen Regeln – Deduktionsregeln oder Inferenzregeln – werden rein mechanisch aus Aussagen andere Aussagen hergeleitet
- Aussagen T sind aus einer Menge von Aussagen A *herleitbar*, wenn man T nach endlichen vielen Anwendungen der Regeln auf A erhält
  - $A \vdash T$
  - Ausgangsaussagen A werden Axiome genannt, hergeleitete Aussagen T Theoreme
  - Ableitung wird auch Beweis genannt
- Beweis ist endliche Folge von Aussagen
  - $P_1, P_2, \dots, P_n$
  - wobei jedes  $P_i$  ein Axiom oder eine nach den Regeln hergeleitete Aussage ist
  - Beweis gelingt, wenn die letzte Aussage  $P_n$  das zu beweisende Theorem ist
- Beweisstrategie
  - effizient – möglichst kurze Beweise
  - zielgerichtet – möglichst wenig fehlschlagende Beweisversuche

Deduktion: Resolution 3

## Semantische Folgerungsbeziehung

- Wenn in einer bestimmten Situation Aussagen A wahr sind, sind dann notwendigerweise auch andere Aussagen T wahr?
- Folgerung sollte für alle Fälle gelten, in denen A und T wahr sind
- Aussagenlogik
  - A und T sind Mengen aussagenlogischer Ausdrücke
  - Es gilt die logische Konsequenz  $A \models T$ , wenn **jede** Zuordnung von Wahrheitswerten, die A wahr macht, auch T wahr macht.
- Prädikatenlogik
  - A und T sind Mengen prädikatenlogischer Ausdrücke
  - Es gilt die logische Konsequenz  $A \models T$ , wenn **jedes** Modell der Menge A auch ein Modell der Menge T ist.

Deduktion: Resolution 2

## Zusammenhang zwischen semantischer und syntaktischer Folgerung

- Deduktionssystem muss korrekt sein, d.h. jedes hergeleitete Theorem T muss logische Konsequenz der Axiome A sein
  - $A \vdash T \rightarrow A \models T$
- Deduktionssystem sollte vollständig sein, d.h. jede logische Konsequenz T der Axiome A sollte auch hergeleitet werden können
  - $A \models T \rightarrow A \vdash T$
- wenn ein Deduktionssystem korrekt und vollständig ist, dann kann man die logische Konsequenz durch einen Beweis ersetzen
  - $A \models T \equiv A \vdash T$
- Entscheidbarkeit eines Deduktionssystems: Gibt es ein **allgemeines** Verfahren, das uns erlaubt festzustellen, ob ein Theorem aus den Axiomen hergeleitet werden kann oder nicht?
  - Aussagenlogik ist entscheidbar
  - Prädikatenlogik ist semi-entscheidbar, d.h. man kann feststellen, dass ein Theorem aus den Axiomen folgt, aber nicht, dass ein Theorem nicht aus den Axiomen folgt. (Das Verfahren kann in diesem Fall in eine unendliche Schleife geraten.)

Deduktion: Resolution 4

## Deduktionssysteme

- Deduktionssystem besteht aus Axiomen, Deduktionsregeln und – was oft vergessen wird – einer Beweisstrategie
- axiomatische Deduktionssysteme
  - enthalten meistens mehrere Deduktionsregeln
  - zusätzlich zu den Axiomen, die den Diskursbereich beschreiben, werden sogenannte logische Axiome verwendet, d.h. Tautologien zur Umformung von Ausdrücken
  - axiomatische Systeme eignen sich in erster Linie für theoretische Untersuchungen
  - sie sind für praktische und speziell für automatische Beweise schlecht geeignet
- natürliche Deduktionssysteme
  - enthalten Deduktionsregeln, die das menschliche Beweisen widerspiegeln sollen
  - gut für das Arbeiten mit Papier und Bleistift
  - basieren auf menschlicher Intuition, sind daher schlecht zu automatisieren
- Resolutionssysteme
  - Grundlage für automatisches Beweisen und für logische Programmiersprachen
- Tableaux-Systeme
  - können manuell wie automatisch verwendet werden

Deduktion: Resolution 5

## Klauselform der Aussagenlogik

- konjunktive Normalform einer aussagenlogischen Formel: Konjunktion von Disjunktionen von atomaren und negierten atomaren Aussagen
 
$$D_1 \wedge D_2 \wedge \dots \wedge D_n \quad (D_i \text{ heißen Klauseln})$$

$$D_i = L_{i1} \vee L_{i2} \vee \dots \vee L_{im} \quad (L_{ij} \text{ heißen Literale})$$
- systematische Umwandlung in Klauselform
  - Beispiel
 
$$A \rightarrow \neg(B \rightarrow C)$$
  - 1. Schritt: Elimination von  $\rightarrow$ 

$$A \rightarrow \neg(B \rightarrow C) \equiv \neg A \vee \neg(\neg B \vee C) \quad [\text{Regel: } P \rightarrow Q \equiv \neg P \vee Q]$$
  - 2. Schritt: Verteilung von  $\neg$  auf atomare Ausdrücke
 
$$\neg A \vee \neg(\neg B \vee C) \equiv \neg A \vee (\neg\neg B \wedge \neg C) \equiv \neg A \vee (B \wedge \neg C) \quad [\text{Regel: } \neg(P \vee Q) \equiv \neg P \wedge \neg Q]$$
  - 3. Schritt: Umwandlung in eine Konjunktion von Disjunktionen
 
$$\neg A \vee (B \wedge \neg C) \equiv (\neg A \vee B) \wedge (\neg A \vee \neg C) \quad [\text{Regel: } P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)]$$
  - 4. Schritt: Darstellung als Menge von Klauseln
 
$$\{(\neg A \vee B), (\neg A \vee \neg C)\}$$

Deduktion: Resolution 7

## Resolution

- Resolution ist ein Deduktionssystem mit einer einzigen Deduktionsregel
- Resolution entstand beim Versuch, Deduktion zu automatisieren, d.h. Beweise durch den Computer automatisch ausführen zu lassen
- Grundidee der Resolution: aus den beiden wahren Aussagen
 
$$P \vee Q$$

$$\neg P \vee R$$
 kann man die wahre Aussage
 
$$Q \vee R$$
 schliessen, denn entweder ist P wahr, dann muss R wahr sein, oder P ist falsch, dann muss Q wahr sein; in jedem Fall ist dann  $Q \vee R$  wahr
- Resolution setzt voraus, dass Aussagen in der Klauselform, d.h. als konjunktive Normalform, dargestellt werden

Deduktion: Resolution 6

## Klauselform der Aussagenlogik

- Klausel ist eine Aussage der Form
 
$$P_1 \vee P_2 \vee \dots \vee P_n \vee \neg N_1 \vee \neg N_2 \vee \dots \vee \neg N_m$$
- äquivalente Formen
 
$$\equiv P_1 \vee P_2 \vee \dots \vee P_n \vee \neg(N_1 \wedge N_2 \wedge \dots \wedge N_m)$$

$$\equiv (N_1 \wedge N_2 \wedge \dots \wedge N_m) \rightarrow (P_1 \vee P_2 \vee \dots \vee P_n)$$
- Notation der logischen Programmierung
 
$$\equiv P_1, P_2, \dots, P_n \leftarrow N_1, N_2, \dots, N_m$$
 (Kommata auf der linken Seite bedeuten Disjunktion, auf der rechten Seite Konjunktion)

Deduktion: Resolution 8

## Schlussregel Resolution

- Resolutionsregel (Robinson)

Klausel  $K_1$  mit dem positiven Literal  $L$

Klausel  $K_2$  mit dem negativen Literal  $\neg L$

Aus den Klauseln  $K_1$  und  $K_2$  leitet man die Resolvente, d.h. die Klausel  $\{K_1 - \{L\}\} \cup \{K_2 - \{\neg L\}\}$  ab.

$$\begin{array}{r} K_1 \qquad \qquad K_2 \\ \hline \{K_1 - \{L\}\} \cup \{K_2 - \{\neg L\}\} \end{array}$$

- Beispiel

$$\begin{array}{r} p \vee q \vee \neg r \qquad s \vee \neg p \\ \hline q \vee \neg r \vee s \end{array}$$

Deduktion: Resolution 9

## Leere Klausel

- leere Klausel  $\{\}$ , d.h. Klausel ohne positive und negative Literale, steht für die widersprüchliche Aussage  $\perp$ , d.h. für Inkonsistenz

- Wie kann man das verstehen?

Es gilt

$$A \vee \perp \equiv A$$

$$A \wedge \neg \perp \equiv A$$

Dann

$$(N_1 \wedge N_2 \wedge \dots \wedge N_m) \rightarrow (P_1 \vee P_2 \vee \dots \vee P_n)$$

$$\equiv (N_1 \wedge N_2 \wedge \dots \wedge N_m \wedge \neg \perp) \rightarrow (P_1 \vee P_2 \vee \dots \vee P_n \vee \perp)$$

leere Klausel enthält keine  $N$  und keine  $P$

$$\equiv \neg \perp \rightarrow \perp$$

$$\equiv \neg \neg \perp \vee \perp$$

$$\equiv \perp \vee \perp$$

$$\equiv \perp$$

Deduktion: Resolution 11

## Andere Schlussregeln als Resolution

- Resolution ist eine mächtige Schlussregel, die andere Schlussregeln als Spezialfälle enthält

- modus ponens

$$\begin{array}{r} P \rightarrow Q \qquad P \\ \hline Q \end{array}$$

als Resolution

$$\begin{array}{r} \neg P \vee Q \qquad P \\ \hline Q \end{array}$$

- modus tollens

$$\begin{array}{r} P \rightarrow Q \qquad \neg Q \\ \hline \neg P \end{array}$$

als Resolution

$$\begin{array}{r} \neg P \vee Q \qquad \neg Q \\ \hline \neg P \end{array}$$

Deduktion: Resolution 10

## Leere Klausel

- Beispiel

$$\text{Klauselmenge } M = \{\neg p \vee q, p, \neg q\}$$

Resolutionen

$$\begin{array}{r} \neg p \vee q \qquad p \\ \hline q \qquad \neg q \\ \hline \{\} \end{array}$$

d.h. die Klauselmenge  $M$  ist inkonsistent

andere Ableitung der leeren Klausel

$$\begin{array}{r} \neg p \vee q \qquad \neg q \\ \hline \neg p \qquad p \\ \hline \{\} \end{array}$$

- Eine Klauselmenge ist inkonsistent, wenn man durch Resolution auf irgendeine Weise die leere Klausel  $\{\}$  ableiten kann.

Deduktion: Resolution 12

## Deduktionstheorem

- Deduktionstheorem

$M \cup \{A\} \vdash B$  genau dann, wenn  $M \vdash (A \rightarrow B)$

- $M \vdash P$  genau dann, wenn  $M \cup \{\neg P\}$  inkonsistent ist

Spezialfall des Deduktionstheorems

$M \cup \{A\} \vdash B$  genau dann, wenn  $M \vdash (A \rightarrow B)$

setze  $A = \neg P$  und  $B = \perp$

$M \cup \{\neg P\} \vdash \perp$  genau dann, wenn  $M \vdash (\neg P \rightarrow \perp)$ , d.h. wenn  $M \vdash P$

Deduktion: Resolution 13

## Deduktion mit Resolution: Beispiele

- Beispiel

Beweis von  $(p \rightarrow q) \rightarrow r \vdash p \rightarrow (q \rightarrow r)$

Ist

$\{(p \rightarrow q) \rightarrow r, \neg(p \rightarrow (q \rightarrow r))\}$

inkonsistent?

Umwandlung in Klauselform

$\equiv \{\neg(\neg p \vee q) \vee r, \neg(\neg p \vee (\neg q \vee r))\}$

$\equiv \{(p \wedge \neg q) \vee r, p \wedge q \wedge \neg r\}$

$\equiv \{(p \vee r) \wedge (\neg q \vee r), p \wedge q \wedge \neg r\}$

$\equiv \{p \vee r, \neg q \vee r, p, q, \neg r\}$

Resolution

$\neg q \vee r \quad \neg r$

-----  
 $\neg q \qquad q$   
 -----  
 $\{\}$

Beweis wurde erbracht

Deduktion: Resolution 15

## Deduktion mit Resolution

- Deduktion mit Resolution verwendet Beweise durch Widerspruch (Refutation)

- Vorgehensweise

$M \vdash P$  genau dann, wenn  $M \cup \{\neg P\}$  inkonsistent ist

Um nachzuweisen, dass  $M \cup \{\neg P\}$  inkonsistent ist, transformiert man  $M \cup \{\neg P\}$  in Klauselform.

Dabei verwendet man den Satz: Eine Menge  $M$  von aussagenlogischen Formeln kann in eine Menge von Klauseln transformiert werden, die genau dann konsistent ist, wenn  $M$  konsistent ist.

Wenn man aus den aus  $M \cup \{\neg P\}$  entstandenen Klauseln durch Resolution die leere Klausel ableiten kann, dann ist diese Klauselmengung inkonsistent. Damit ist auch  $M \cup \{\neg P\}$  inkonsistent, und somit gilt  $M \vdash P$ .

Falls man die leere Klausel nicht ableiten kann, dann ist  $M \cup \{\neg P\}$  konsistent und somit kann  $P$  nicht aus  $M$  abgeleitet werden.

Deduktion: Resolution 14

## Deduktion mit Resolution: Beispiele

- Beispiel (Umkehrung der Aussagen)

Beweis von  $p \rightarrow (q \rightarrow r) \vdash (p \rightarrow q) \rightarrow r$

Ist

$\{p \rightarrow (q \rightarrow r), \neg((p \rightarrow q) \rightarrow r)\}$

inkonsistent?

Umwandlung in Klauselform

$\equiv \{\neg p \vee (\neg q \vee r), \neg(\neg(p \rightarrow q) \vee r)\}$

$\equiv \{\neg p \vee \neg q \vee r, (\neg p \vee q) \wedge \neg r\}$

$\equiv \{\neg p \vee \neg q \vee r, \neg p \vee q, \neg r\}$

Resolution führt in diesem Fall nicht zur leeren Klausel; man kann sich leicht überzeugen, dass immer eine Resolvente  $\neg p$  übrigbleibt

D.h. die obige Ableitung gilt nicht.

Deduktion: Resolution 16

## Korrektheit und Vollständigkeit der Resolution in der Aussagenlogik

- Korrektheit der Resolution in der Aussagenlogik

Wenn eine Klausel  $K$  aus einer Menge  $M$  von Klauseln durch Resolution abgeleitet werden kann, dann ist  $K$  eine logische Konsequenz von  $M$ .

- Korollar

Wenn die leere Klausel aus einer Menge  $M$  von Klauseln durch Resolution abgeleitet werden kann, dann ist die leere Klausel eine logische Konsequenz von  $M$ , d.h.  $M$  inkonsistent.

- Vollständigkeit der Resolution bezüglich Refutation in der Aussagenlogik

Wenn eine Menge von Klauseln inkonsistent ist, dann kann man aus ihr durch Resolution die leere Klausel ableiten.

Deduktion: Resolution 17

## Klauselform der Prädikatenlogik

- Pränexform

Ein prädikatenlogischer Ausdruck ist in Pränexform, wenn er die Form  $Q_1X_1 (Q_2X_2 \dots Q_nX_n A) \dots$  hat, wobei  $Q_i$  Quantoren sind,  $X_i$  Variable und  $A$  keine Quantoren enthält.

- Klauselform

Ein geschlossener prädikatenlogischer Ausdruck ist in Klauselform, wenn er die Form  $\forall X_1 (\forall X_2 \dots \forall X_n A) \dots$  hat, wobei  $X_i$  alle Variablen sind, die in  $A$  auftauchen,  $A$  keine Quantoren enthält und die Form einer Disjunktion von atomaren Ausdrücken oder deren Negation hat.

- Beispiel

Ausdruck

$$\forall X (\text{mensch}(X) \rightarrow \text{sterblich}(X))$$

Pränexform

wie gegeben

Klauselform

$$\forall X (\neg \text{mensch}(X) \vee \text{sterblich}(X))$$

Deduktion: Resolution 19

## Resolution in der Prädikatenlogik

- Resolution ist eine der wichtigsten Deduktionsmethoden der Prädikatenlogik
- Um Resolution in der Prädikatenlogik verwenden zu können, müssen wir jedoch einige Erweiterungen einführen, insbesondere bei der Behandlung von Variablen.

Deduktion: Resolution 18

## Umwandlung in Klauselform

- Eine Menge von prädikatenlogischen Ausdrücken kann in eine Menge von Klauseln transformiert werden, die genau dann konsistent ist, wenn die Ausgangsmenge konsistent ist.
- Die Umwandlung eines Ausdrucks in Klauselform geschieht in 8 Schritten

- Beispiel

$$\forall X (\forall Y p(X, Y) \rightarrow \neg (\forall Y (q(X, Y) \rightarrow r(X, Y))))$$

1. Wir eliminieren  $\rightarrow$  und erhalten

$$\forall X (\neg \forall Y p(X, Y) \vee \neg (\forall Y (\neg q(X, Y) \vee r(X, Y))))$$

2. Wir verteilen die Negationen, sodass jede Negation nur auf ein Atom wirkt, und erhalten

$$\forall X (\exists Y \neg p(X, Y) \vee \exists Y (q(X, Y) \wedge \neg r(X, Y)))$$

3. Wir benennen die Variablen um, sodass jede Variable nur einmal quantifiziert wird, und erhalten

$$\forall X (\exists Y \neg p(X, Y) \vee \exists Z (q(X, Z) \wedge \neg r(X, Z)))$$

Deduktion: Resolution 20

## Umwandlung in Klauselform

### 4. Wir eliminieren alle existentiellen Quantoren.

Wenn ein existentieller Quantor nicht im Gültigkeitsbereich eines universellen Quantors vorkommt, ersetzen wir jedes Auftauchen der quantifizierten Variablen durch eine bisher nicht verwendete Konstante.

$$\exists X p(X) \rightarrow p(a)$$

Wenn ein existentieller Quantor im Gültigkeitsbereich universeller Quantoren vorkommt, dann ist es möglich, dass die existentiell quantifizierte Variable von den universell quantifizierten abhängt. Wir ersetzen sie daher durch eine bisher nicht verwendete Funktion der universell quantifizierten Variablen.

$$\forall X (\exists Y p(X, Y)) \rightarrow \forall X p(X, f(X))$$

Ergebnis für das Beispiel:

$$\forall X (\neg p(X, f1(X)) \vee (q(X, f2(X)) \wedge \neg r(X, f2(X))))$$

5. Alle verbleibenden Variablen sind nun universell quantifiziert. Wir können die universellen Quantoren daher auch fortlassen. Alle Variablen werden implizit als universell quantifiziert betrachtet.

$$\neg p(X, f1(X)) \vee (q(X, f2(X)) \wedge \neg r(X, f2(X)))$$

Deduktion: Resolution 21

Deduktion: Resolution 22

## Klauselform der Prädikatenlogik

- Klausel ist Aussage der Form

$$P_1 \vee P_2 \vee \dots \vee P_n \vee \neg N_1 \vee \neg N_2 \vee \dots \vee \neg N_m$$

alle Variablen sind implizit vor der Klausel universell quantifiziert

- äquivalente Formen

$$\equiv P_1 \vee P_2 \vee \dots \vee P_n \vee \neg (N_1 \wedge N_2 \wedge \dots \wedge N_m)$$

$$\equiv N_1 \wedge N_2 \wedge \dots \wedge N_m \rightarrow P_1 \vee P_2 \vee \dots \vee P_n$$

- Notation der logischen Programmierung

$$\equiv P_1, P_2, \dots, P_n \leftarrow N_1, N_2, \dots, N_m$$

(Kommata auf der linken Seite bedeuten Disjunktion, auf der rechten Seite Konjunktion)

Deduktion: Resolution 23

Deduktion: Resolution 24

## Umwandlung in Klauselform

6. Wir bringen den Satz in konjunktive Normalform d.h. eine Konjunktion von Disjunktionen.

$$(\neg p(X, f1(X)) \vee q(X, f2(X))) \wedge (\neg p(X, f1(X)) \vee \neg r(X, f2(X)))$$

7. Wir eliminieren alle konjunktiven Konnektoren und schreiben die Konjunktion als eine Menge von Klauseln.

$$\{\neg p(X, f1(X)) \vee q(X, f2(X)), \neg p(X, f1(X)) \vee \neg r(X, f2(X))\}$$

8. Zum Schluss benennen wir die Variablen noch einmal um, sodass keine Variable in mehr als einer Klausel auftaucht.

$$\{\neg p(X, f1(X)) \vee q(X, f2(X)), \neg p(Y, f1(Y)) \vee \neg r(Y, f2(Y))\}$$

## Resolution mit Variablen

- gegeben zwei Klauseln

$$p(X1) \vee q(X1)$$

$$\neg p(a) \vee r(X2, Y2)$$

- Resolvente?

- Da alle Variablen universell quantifiziert sind, kann man jede Variable durch einen Term aus dem Wertebereich der Variablen substituieren.

- Im Beispiel ergibt die Substitution  $\{X1/a\}$  die Klauselinstanzen

$$p(a) \vee q(a)$$

$$\neg p(a) \vee r(X2, Y2)$$

- Resolvente

$$q(a) \vee r(X2, Y2)$$

## Substitutionen, Unifikatoren

- Begriffe
  - [L] s bedeutet Anwendung der Substitution s auf das Literal L
  - Literal [L] s = L' ist eine Instanz von L
  - Literal L subsumiert seine Instanz L'
- Unifikator
  - Wenn es für zwei Literale L1 und L2 eine Substitution s gibt, sodass
    - [L1] s = [L2] s
 dann nennen wir s einen Unifikator von L1 und L2.
- allgemeinsten Unifikator (**most general unificator**, mgu)
  - Der Unifikator s ist genau dann der allgemeinste Unifikator für L1 und L2, wenn für jeden anderen Unifikator s' gilt, dass das Literal [L1] s das Literal [L1] s' subsumiert.
  - Der allgemeinste Unifikator ist bis auf Variablenumbenennungen eindeutig.

Deduktion: Resolution 25

## Wann unifizieren zwei Literale?

- Relationszeichen (Prädikatsname) und Arität müssen übereinstimmen
- alle Argumente müssen unifiziert werden können
  - Unifikation gelingt, wenn ein oder beide Argumente Variable sind
  - Unifikation gelingt, wenn beide Argumente die gleiche Konstante sind
  - Unifikation gelingt, wenn beide Argumente zusammengesetzte Terme mit dem gleichen Funktionszeichen (Funktionsnamen, Funktor) und der gleichen Arität sind und alle Argumente unifizieren

Deduktion: Resolution 27

## Beispiel für allgemeinsten Unifikator

- Literale
  - $p(a, Y, Z)$
  - $p(X, b, Z)$
 werden durch die Substitution s
  - $\{X/a, Y/b, Z/c\}$
 unifiziert
  - $[p(a, Y, Z)] \{X/a, Y/b, Z/c\} \equiv$
  - $[p(X, b, Z)] \{X/a, Y/b, Z/c\} \equiv$
  - $p(a, b, c)$
 Aber s ist nicht der allgemeinste Unifikator. Der allgemeinste Unifikator ist in diesem Fall
  - $\{X/a, Y/b\}$
 mit der Instanz
  - $[p(a, Y, Z)] \{X/a, Y/b\} \equiv$
  - $[p(X, b, Z)] \{X/a, Y/b\} \equiv$
  - $p(a, b, Z)$

Deduktion: Resolution 26

## Resolution mit Unifikation

- Gegeben
  - Literal L1 in einer Klausel K1
  - Literal  $\neg L2$  in einer Klausel K2
  - L1 und L2 haben allgemeinsten Unifikator u
- Resolvente besteht aus der Vereinigungsmenge aller Literale von K1 und K2 abzüglich L1 und  $\neg L2$ , auf die u angewandt wurde
  - |   |    |
|---|----|
| K1  | K2 |
| -----   |    |
| $\{[K1 - \{L1\}] \cup [K2 - \{\neg L2\}]\} u$ |    |
- Beispiel
  - |  |                          |
|--|--------------------------|
| $p(X, f(2)) \vee q(X) \vee \neg r(X, Y)$ | $s(V) \vee \neg p(4, V)$ |
| -----                                    |                          |
| $q(4) \vee \neg r(4, Y) \vee s(f(2))$    |                          |
  - allgemeinsten Unifikator ist  $\{X/4, V/f(2)\}$  denn es gilt
  - $p(X, f(2)) \{X/4, V/f(2)\} = p(4, V) \{X/4, V/f(2)\}$

Deduktion: Resolution 28

**Beispiel (Truss p. 313)**

- Beweis der Addition  $2 + 2 = 4$  in Peano Arithmetik durch Resolution

Axiome

$$X + 0 = X$$

$$X + s(Y) = s(X + Y)$$

Theorem

$$s(s(0)) + s(s(0)) = s(s(s(s(0))))$$

- Mit den Abkürzungen

sX für s(X)

add(X, Y, Z) für  $X+Y=Z$ 

lauten die Axiome

$$\text{add}(X, 0, X)$$

$$\text{add}(X, Y, Z) \rightarrow \text{add}(X, sY, sZ)$$

und das Theorem

$$\text{add}(ss0, ss0, ssss0)$$

- Umwandlung in Klauselform

$$(1) \text{ add}(X, 0, X)$$

$$(2) \neg \text{add}(X, Y, Z) \vee \text{add}(X, sY, sZ)$$

$$(3) \neg \text{add}(ss0, ss0, ssss0) \quad (\text{negiertes Theorem})$$

Deduktion: Resolution 29

**Beispiel (Truss p. 313)**

- Für den dritten Resolutionsschritt verwenden wir die Resolvente (5) und Klausel (1).

$$(5) \neg \text{add}(ss0, 0, ss0)$$

$$(1) \text{ add}(X, 0, X)$$

Mit dem mgu =  $\{X/ss0\}$  erhalten wir die Resolvente

$$(6) \{\}$$

- Die leere Resolvente bedeutet, dass die Klauselmengemenge  $\{(1), (2), (3)\}$  inkonsistent ist und dass das Theorem aus den Axiomen folgt, d.h. dass  $2 + 2 = 4$  in der Peano Arithmetik gilt.

- Der Resolutionsbeweis hat eine Kombination von zwei Beweisstrategien benutzt, um effizient – dh. in möglichst wenigen Schritten – und effektiv – dh. ohne Umwege und ohne Sackgassen – zur leeren Klausel zu gelangen:

*set-of-support*

man benutzt das negierte Theorem als eine der beiden Klauseln beim ersten Resolutionsschritt

*lineare Resolution*

man verwendet ab dem zweiten Resolutionsschritt immer die Resolvente des vorherigen Schrittes als eine der beiden Klauseln; die andere Klausel ist eines der Axiome oder eine vorherige Resolvente

Deduktion: Resolution 31

**Beispiel (Truss p. 313)**

- Für den ersten Resolutionsschritt verwenden wir das negierte Theorem (3) und die einzige passende Klausel (2). (Das Literal der Klausel (3) unifiziert mit dem Literal der Klausel (1) nicht.)

$$(2) \neg \text{add}(X, Y, Z) \vee \text{add}(X, sY, sZ)$$

$$(3) \neg \text{add}(ss0, ss0, ssss0)$$

mit dem mgu =  $\{X/ss0, Y/s0, Z/ss0\}$  erhalten wir die Resolvente

$$(4) \neg \text{add}(ss0, s0, sss0)$$

- Für den zweiten Resolutionsschritt verwenden wir die Resolvente (4) und Klausel (2). Klausel (1) kommt wieder nicht in Frage.

$$(4) \neg \text{add}(ss0, s0, sss0)$$

$$(2) \neg \text{add}(X, Y, Z) \vee \text{add}(X, sY, sZ)$$

Mit dem mgu =  $\{X/ss0, Y/0, Z/ss0\}$  erhalten wir die Resolvente

$$(5) \neg \text{add}(ss0, 0, ss0)$$

Deduktion: Resolution 30

**Berechnung als Beweis**

- Im Beispiel wurde die Addition  $2+2=4$  überprüft. Nun wollen wir den Resolutionsbeweis konstruktiv verwenden, d.h. einen Wert "durch Beweis berechnen".

Welchen Wert muss W haben, damit  $W + 2 = 4$ ?

- Klauselform

$$(1) \text{ add}(X, 0, X)$$

$$(2) \neg \text{add}(X, Y, Z) \vee \text{add}(X, sY, sZ)$$

$$(3) \neg \text{add}(W, ss0, ssss0) \quad (\text{negiertes Theorem})$$

- Erster Resolutionsschritt:

$$(2) \neg \text{add}(X, Y, Z) \vee \text{add}(X, sY, sZ)$$

$$(3) \neg \text{add}(W, ss0, ssss0)$$

$$\text{mgu} = \{X/W, Y/s0, Z/ss0\}$$

$$(4) \neg \text{add}(W, s0, sss0)$$

- Zweiter Resolutionsschritt:

$$(2) \neg \text{add}(X, Y, Z) \vee \text{add}(X, sY, sZ)$$

$$(4) \neg \text{add}(W, s0, sss0)$$

$$\text{mgu} = \{X/W, Y/0, Z/ss0\}$$

$$(5) \neg \text{add}(W, 0, ss0)$$

Deduktion: Resolution 32



## Berechnung als Beweis

- Dritter Resolutionsschritt:

(1)  $\text{add}(X, 0, X)$

(5)  $\neg \text{add}(W, 0, \text{ss}0)$

$\text{mgu} = \{X/W, X/\text{ss}0\}$

(6)  $\{\}$

X wurde an  $\text{ss}0$  gebunden, W an X, d.h. W an  $\text{ss}0$ .

Damit haben wir W de facto als  $\text{ss}0$  berechnet.

Deduktion: Resolution 33

## Implementierung der Resolution in Prolog

- Programmiersprache Prolog: effiziente und effektive Implementierung von Resolution
- Einschränkungen der Sprache auf Hornklauseln
- Verwendung der SLD Resolutionsstrategie
- Gewinn in Effizienz und Effektivität wird durch Unvollständigkeit erkauft, d.h. es können nicht mehr alle Lösungen durch Resolution gefunden werden

Deduktion: Resolution 35

## Korrektheit und Vollständigkeit der Resolution in der Prädikatenlogik

- Korrektheit der Resolution in der Prädikatenlogik

Wenn eine Klausel K aus einer Menge M von Klauseln durch Resolution abgeleitet werden kann, dann ist K eine logische Konsequenz von M.

- Korollar

Wenn die leere Klausel aus einer Menge M von Klauseln durch Resolution abgeleitet werden kann, dann ist M inkonsistent.

- Vollständigkeit der Resolution bezüglich Refutation in der Prädikatenlogik

Wenn eine Menge von Klauseln inkonsistent ist, dann kann man aus ihr durch Resolution die leere Klausel ableiten.

Deduktion: Resolution 34

## Definite Klauseln

- *definite Klauseln* enthalten genau ein positives Literal

$$\forall (H \vee \neg B_1 \vee \neg B_2 \vee \dots \vee \neg B_n)$$

werden meistens als nach links gerichtete Implikation geschrieben

$$H \leftarrow B_1, B_2, \dots, B_n. \quad (n \geq 0) \quad (\text{Prolog: } :- \text{ statt } \leftarrow)$$

Atom H heisst Kopf

Konjunktion  $B_1, B_2, \dots, B_n$  Körper

Kommata stehen für die logische Konjunktion

alle Variablen sind implizit vor der Klausel universell quantifiziert

Klausel heisst Regel, wenn  $n > 0$

Klausel heisst Fakt, wenn  $n = 0$  ( $\leftarrow$  wird dann oft fortgelassen)

Fakt kann als Regel mit dem Körper *true* betrachtet werden

Deduktion: Resolution 36

## Definite Programme

- definites Programm: endliche Menge von definiten Klauseln
  - $\text{pfad}(X, Y) \leftarrow \text{verbunden}(X, Y).$
  - $\text{pfad}(X, Y) \leftarrow \text{verbunden}(X, Z), \text{pfad}(Z, Y).$
  - $\text{verbunden}(a, b).$
  - $\text{verbunden}(b, c).$
- Prolog-Programme sind definite Programme

Deduktion: Resolution 37

## SLD Resolution

- gegeben sei das Prolog Programm
  - $\text{pfad}(X, Y) \leftarrow \text{verbunden}(X, Y).$
  - $\text{pfad}(X, Y) \leftarrow \text{verbunden}(X, Z), \text{pfad}(Z, Y).$
  - $\text{verbunden}(a, b).$
  - $\text{verbunden}(b, c).$
 und die Anfrage
  - $\leftarrow \text{pfad}(a, \text{Wohin}).$
- Anfrage  $\leftarrow \text{pfad}(a, \text{Wohin})$  wird durch Resolution beantwortet und die Variable Wohin wird während des Resolutionsbeweises an die Konstante b gebunden – der Wert von Wohin wird als b berechnet
- Prolog verwendet eine Kombination von set-of-support Resolution mit linearer Resolution
- wenn eine Resolvente aus mehr als einem Literal besteht, wird immer das am weitesten links stehende Literal für den nächsten Resolutionsschritt verwendet
- Resolventen bilden den SLD Baum; Wurzel des Baumes ist die ursprüngliche Anfrage, Blätter des Baumes sind entweder leere Resolventen – Refutationsbeweis ist gelungen – oder nichtleere Resolventen, für die kein weiterer Resolutionsschritt möglich ist

Deduktion: Resolution 39

## Zielklauseln, Anfragen

- Klauseln, die nur negative Literale enthalten
  - $\forall X_1 \dots X_k (\neg G_1 \vee \neg G_2 \vee \dots \vee \neg G_m)$
 können zu
  - $\neg \exists X_1 \dots X_k (G_1 \wedge G_2 \wedge \dots \wedge G_m)$
 umgeformt werden, wobei  $X_1 \dots X_k$  die Variablen der Klausel sind
- Bedeutung: es gibt keine Instanzen der Variablen  $X_1 \dots X_k$ , sodass die Konjunktion der Ziele  $G_i$  wahr ist
- $\forall (\neg G_1 \vee \neg G_2 \vee \dots \vee \neg G_m)$  wird definite Zielklausel genannt und als Anfrage verstanden, die durch Widerspruch beantwortet werden kann
- meistens geschrieben
  - $\leftarrow G_1, G_2, \dots, G_n$  (Prolog: ?-  $G_1, G_2, \dots, G_n$ )
- Beispiel
  - $\leftarrow \text{pfad}(a, b).$
- definite Klauseln und definite Zielklauseln werden zusammen *Horn-Klauseln* genannt

Deduktion: Resolution 38

## SLD Resolution

- Prolog verwendet Tiefensuche, um den SLD Baum abzusuchen, bzw. zu generieren
- wenn ein Blatt eine nichtleere Resolvente ist, dann wird automatisch *Backtracking* ausgelöst, d.h. der Beweis wird am nächsten vorangehenden Knoten des Baumes fortgesetzt, der alternative Zweige hat; gibt es keine alternativen Zweige, dann ist der Refutationsbeweis misslungen
- Prolog ist nichtdeterministisch, d.h. Anfragen können mehr als eine Lösung generieren; konkret hat die Anfrage
  - $\leftarrow \text{pfad}(a, \text{Wohin}).$
 die Lösungen
  - Wohin = b
  - Wohin = c
- wenn im SLD Baum unendliche Zweige auftauchen, dann verhindert die Tiefensuche, dass Zweige abgesucht werden, die rechts vom unendlichen Zweig sind; d.h. mögliche Lösungen werden nicht gefunden; Prolog ist also potentiell unvollständig

Deduktion: Resolution 40