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?

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 |= 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 |= T, wenn **jedes** Modell der Menge A auch ein Modell der Menge T ist.

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

  $P_1, P_2, ..., P_n$

  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

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.)

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 Deduktionsysteme

  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

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

Klauselform der Aussagenlogik

- konjunktive Normalform einer aussagenlogischen Formel: Konjunktion von Disjunktionen von atomaren und negierten atomaren Aussagen

  $D_1 \wedge D_2 \wedge ... \wedge D_n$ (D_i heissen Klauseln)

  $D_i = L_{i1} \vee L_{i2} \vee ... \vee L_{im}$ (L_{ij} heissen Literale)

- systematische Umwandlung in Klauselform

  Beispiel

  $A \rightarrow \neg(B \rightarrow C)$

  1. Schritt: Elimination von $\rightarrow$

  $A \rightarrow \neg (B \rightarrow C) \equiv$ [Regel: $P \rightarrow Q \equiv \neg P \vee Q$]

  $\neg A \vee \neg (\neg B \vee C)$

  2. Schritt: Verteilung von $\neg$ auf atomare Ausdrücke

  $\neg A \vee \neg (\neg B \vee C) \equiv$ [Regel: $\neg (P \vee Q) \equiv \neg P \wedge \neg Q$]

  $\neg A \vee (\neg \neg B \wedge \neg C) \equiv$ [Regel: $\neg \neg P \equiv P$]

  $\neg A \vee (B \wedge \neg C)$

  3. Schritt: Umwandlung in eine Konjunktion von Disjunktionen

  $\neg A \vee (B \wedge \neg C) \equiv$ [Regel: $P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$]

  $(\neg A \vee B) \wedge (\neg A \vee \neg C)$

  4. Schritt: Darstellung als Menge von Klauseln

  $\{(\neg A \vee B) , (\neg A \vee \neg C)\}$

Klauselform der Aussagenlogik

- Klausel ist eine Aussage der Form

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

- äquivalente Formen

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

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

- Notation der logischen Programmierung

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

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

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.

$$\frac{K_1 \qquad\qquad K_2}{\{K_1 - \{L\}\} \cup \{ K_2 - \{\neg L\}\}}$$

- Beispiel

$$\frac{p \vee q \vee \neg r \qquad\qquad s \vee \neg p}{q \vee \neg r \vee s}$$

Andere Schlussregeln als Resolution

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

- modus ponens

$$\frac{P \rightarrow Q \qquad P}{Q}$$

als Resolution

$$\frac{\neg P \vee Q \qquad P}{Q}$$

- modus tollens

$$\frac{P \rightarrow Q \qquad \neg Q}{\neg P}$$

als Resolution

$$\frac{\neg P \vee Q \qquad \neg Q}{\neg P}$$

Leere Klausel

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

- Wie kann man das verstehen?

  Es gilt

  $A \vee \bot \equiv A$

  $A \wedge \neg \bot \equiv A$

  Dann

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

  $\equiv (N_1 \wedge N_2 \wedge ... \wedge N_m \wedge \neg \bot) \rightarrow (P_1 \vee P_2 \vee ... \vee P_n \vee \bot)$

  leere Klausel enthält keine N und keine P

  $\equiv \neg \bot \rightarrow \bot$

  $\equiv \neg \neg \bot \vee \bot$

  $\equiv \bot \vee \bot$

  $\equiv \bot$

Leere Klausel

- Beispiel

  Klauselmenge M = {¬p ∨ q, p, ¬ q}

  Resolutionen

  ¬p ∨ q p
  ---
  q ¬ q

  {}

  d.h. die Klauselmenge M ist inkonsistent

  andere Ableitung der leeren Klausel

  ¬p ∨ q ¬q

  ¬ p p

  {}

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

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 $B = \bot$

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

Deduktion mit Resolution

- Deduktion mit Resolution verwendet Beweise durch Widerspruch (Refutation)

- Vorgehensweise

  M |– P genau dann, wenn M ∪ {¬P} inkonsistent ist

  Um nachzuweisen, dass M ∪ {¬P} inkonsistent ist, transformiert man M ∪ {¬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 ∪ {¬P} entstandenen Klauseln durch Resolution die leere Klausel ableiten kann, dann ist diese Klauselmenge inkonsistent. Damit ist auch M ∪ {¬P} inkonsistent, und somit gilt M |– P.

  Falls man die leere Klausel nicht ableiten kann, dann ist M ∪ {¬P} konsistent und somit kann P nicht aus M abgeleitet werden.

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 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

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

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

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

  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.

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.

Resolution in der Prädikatenlogik

- Resolution ist eine der wichtigsten Deduktions-methoden der Prädikatenlogik

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

Klauselform der Prädikatenlogik

- Pränexform

  Ein prädikatenlogischer Ausdruck ist in Pränexform, wenn er die Form Q_1X_1 (Q_2X_2 ....Q_nX_n A) ...) 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 $\forall X_1$ ($\forall X_2$ ....$\forall X_n$ A) ...) 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

  $\forall X$ (mensch(X) $\rightarrow$ sterblich(X))

  Pränexform

  wie gegeben

  Klauselform

  $\forall X$ ($\neg$ mensch(X) $\vee$ sterblich(X))

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

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

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.

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

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.

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

Klauselform der Prädikatenlogik

- Klausel ist Aussage der Form

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

  alle Variablen sind implizit vor der Klausel universell quantifiziert

- äquivalente Formen

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

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

- Notation der logischen Programmierung

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

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

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.

- allgemeinster Unifikator (**m**ost **g**eneral **u**nificator, 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 Variablen-umbenennungen eindeutig.

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} ≡

  [p(X, b, Z)] {X / a, Y / b, Z / c} ≡

  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} ≡

  [p(X, b, Z)] {X / a, Y / b} ≡

  p(a, b, Z)

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

Resolution mit Unifikation

- Gegeben

  Literal L1 in einer Klausel K1

  Literal ¬ 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 ¬ L2, auf die u angewandt wurde

  $$\frac{K1 \qquad\qquad K2}{[\{K1 - \{L1\}\} \cup \{K2 - \{\neg L2\}\}]\ u}$$

- Beispiel

  $$\frac{p(X,f(2)) \vee q(X) \vee \neg\, r(X,Y) \qquad\qquad s(V) \vee \neg\, p(4,V)}{q(4) \vee \neg\, r(4,Y) \vee s(f(2))}$$

  allgemeinster Unifikator ist $\{X/4, V/f(2)\}$ denn es gilt

  $p(X,f(2))\ \{X/4, V/f(2)\} = p(4,V)\ \{X/4, V/f(2)\}$

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

  add(X, 0, X)

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

  und das Theorem

  add(ss0, ss0, ssss0)

- Umwandlung in Klauselform

  (1) add(X, 0, X)

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

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

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) $\neg$ add(X, Y, Z) $\vee$ add(X, sY, sZ)

  (3) $\neg$ add(ss0, ss0, ssss0)

  mit dem mgu = {X / ss0, Y / s0, Z / sss0} erhalten wir die Resolvente

  (4) $\neg$ add(ss0, s0, sss0)

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

  (4) $\neg$ add(ss0, s0, sss0)

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

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

  (5) $\neg$ add(ss0, 0, ss0)

Beispiel (Truss p. 313)

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

  (5) $\neg$ add(ss0, 0, ss0)

  (1) add(X, 0, X)

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

  (6) {}

- Die leere Resolvente bedeutet, dass die Klauselmenge {(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 – d.h. 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

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) $add(X, 0, X)$

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

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

- Erster Resolutionsschritt:

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

  (3) $\neg\, add(W, ss0, ssss0)$

  $mgu = \{X/W, Y/s0, Z/sss0\}$

  (4) $\neg\, add(W, s0, sss0)$

- Zweiter Resolutionsschritt:

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

  (4) $\neg\, add(W, s0, sss0)$

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

  (5) $\neg\, add(W, 0, ss0)$

Berechnung als Beweis

- Dritter Resolutionsschritt:

  (1) add(X, 0, X)

  (5) ¬ add(W, 0, ss0)

  mgu = {X / W, X / ss0}

  (6) {}

  X wurde an ss0 gebunden, W an X, d.h. W an ss0.

  Damit haben wir W de facto als ss0 berechnet.

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.

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

Definite Klauseln

- *definite Klauseln* enthalten genau ein positives Literal

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

  werden meistens als nach links gerichtete Implikation geschrieben

  $H \leftarrow B_1, B_2, ... , B_n.$ $(n \geq 0)$ (Prolog: :- statt $\leftarrow$)

  Atom H heisst Kopf

  Konjunktion $B_1, B_2, ... , B_n$ 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

Definite Programme

- definites Programm: endliche Menge von definiten Klauseln

  pfad(X, Y) ‹– verbunden(X, Y).

  pfad(X, Y) ‹– verbunden(X, Z), pfad(Z, Y).

  verbunden(a, b).

  verbunden(b, c).

- Prolog-Programme sind definite Programme

Zielklauseln, Anfragen

- Klauseln, die nur negative Literale enthalten

  $\forall X_1 \ldots X_k (\neg G_1 \vee \neg G_2 \vee \ldots \vee \neg G_m)$

  können zu

  $\neg\exists X_1 \ldots X_k (G_1 \wedge G_2 \wedge \ldots \wedge G_m)$

  umgeformt werden, wobei $X_1 \ldots X_k$ die Variablen der Klausel sind

- Bedeutung: es gibt keine Instanzen der Variablen $X_1 \ldots X_k$, sodass die Konjunktion der Ziele G_i wahr ist

- $\forall(\neg G_1 \vee \neg G_2 \vee \ldots \vee \neg G_m)$ wird definite Zielklausel genannt und als Anfrage verstanden, die durch Widerspruch beantwortet werden kann

- meistens geschrieben

  $\leftarrow G_1, G_2, \ldots, G_n$ (Prolog: ?- $G_1, G_2, \ldots, G_n$.)

- Beispiel

  ‹– pfad(a, b).

- definite Klauseln und definite Zielklauseln werden zusammen *Horn-Klauseln* genannt

SLD Resolution

- gegeben sei das Prolog Programm

  pfad(X, Y) ‹– verbunden(X, Y).

  pfad(X, Y) ‹– verbunden(X, Z), pfad(Z, Y).

  verbunden(a, b).

  verbunden(b, c).

  und die Anfrage

  ‹– pfad(a, Wohin).

- Anfrage ‹– pfad(a, 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 nichtleeere Resolventen, für die kein weiterer Resolutionsschritt möglich ist

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

  ‹– pfad(a, 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