Wissen beim Lernen

Beispiele und Hypothesen
Wissen im Lernen
Erklärungsbasiertes Lernen (EBL)
Lernen mit Relevanzinformation (RBL)
Induktive Logikprogrammierung (ILP)

Wissen beim Lernen

- Vorwissen über die Welt nutzt
- Kombination aus Wissensrepräsentation und Lernen
- Inkrementeller Aufbau der Hypothesen
- Logische Formulierung des Lernens

Beispiele & Hypothesen (1)

- Es soll eine Regel gelernt werden, die entscheidet, ob man auf einen Tisch wartet.
- Beispiele werden jetzt durch Attribute wie *Alternative, Bar, Frei/Sams* usw. beschrieben.

  $Alternative(X_1) \wedge \neg Bar(X_1) \wedge \neg Frei/Sams(X_1) \wedge \ldots$
- Die Klassifizierung ist durch den folgenden Satz gegeben:

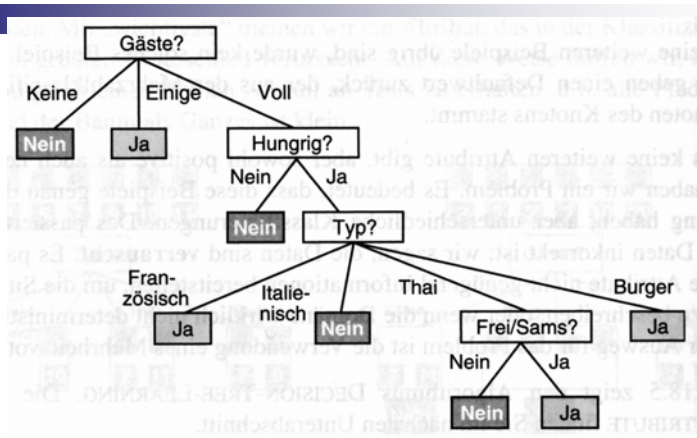
  $WerdenWarten(X_1)$

Beispiele & Hypothesen (1)

- Ziel des induktiven Lernens in der logischen Anordnung ist, einen äquivalenten Ausdruck für das Zielprädikat Q zu finden, um die Beispiele korrekt zu klassifizieren. Jede Hypothese schlägt einen solchen Ausdruck vor, den wir als Kandidatendefinition für das Zielprädikat bezeichnen. Wenn wir mit C_i die Kandidatendefinition bezeichnen, ist jede Hypothese H_i ein Satz der Form $\forall x\ Q(x) \Leftrightarrow C_i(x)$.

Extension

- Jede Hypothese sagt eine bestimmte Menge an Beispielen voraus. Diese Menge wird als Extension des Prädikats bezeichnet.
- Zwei Hypothesen mit unterschiedlichen Extensionen sind logisch inkonsistent zueinander, wenn sie in ihrer Vorhersage für mindestens ein Beispiel nicht übereinstimmen.
- Wenn sie dieselbe Extension haben, sind sie logisch äquivalent.



$$\forall r \quad WerdenWarten(r) \Leftrightarrow Gäste(r, Einige)$$
$$\vee\ Gäste(r, Voll) \wedge Hungrig(r) \wedge Typ(r, Französisch)$$
$$\vee\ Gäste(r, Voll) \wedge Hungrig(r) \wedge Typ(r, Thai)$$
$$\wedge\ Frei/Sams(r)$$
$$\vee\ Gäste(r, Voll) \wedge Hungrig(r) \wedge Typ(r, Burger)$$

Beispiele & Hypothesen (2)

- Ein Beispiel kann für die Hypothese **falsch negativ** sein, wenn die Hypothese sagt, es sollte negativ sein, tatsächlich positiv ist.
- Ein Beispiel kann für die Hypothese **falsch positiv** sein, wenn die Hypothese sagt, es sollte positiv sein, tatsächlich aber negativ ist.
- Ist eine Hypothese falsch negativ oder falsch positiv, ist das Beispiel und die Hypothese logisch inkonsistent zueinander. Dann kann diese Hypothese ausgeschlossen werden.

Induktives Lernen

- Angegeben ist der Hypothesenraum $H_1 \vee H_2 \vee H_3 \vee H_4$
- H_2und H_3sind inkonsistent
- Neuer Hypothesenraum

  $H_1 \vee H_4$
- Induktives Lernen durch schrittweise Eliminierung von Hypothesen

Aktuell beste Hypothese-Suche

Die Idee hinter der Aktuell-beste-Hypothesen-Suche ist, eine einzige Hypothese zu verwalten und sie anzupassen, wenn neue Beispiele eintreffen.

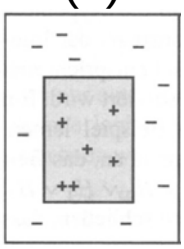
Wir haben eine Hypothese H_r, die mit allen Beispielen konsistent ist.

Was machen wir, wenn ein falsch negatives oder falsch positives Beispiel auftritt.

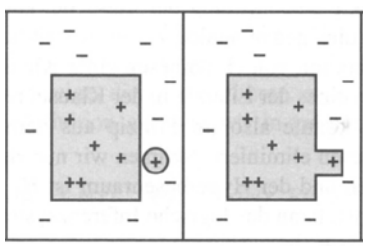
- Falsch negativ
  - Die Extension der Hypothese muss vergrößert werden. Man spricht von Verallgemeinerung
- Falsch positiv
  - Die Extension der Hypothese muss verkleinert werden. Man spricht von Spezialisierung

Aktuell beste Hypothese-Suche (2)

- Wir haben eine Hypothese H_r, von dem wir überzeugt sind.
- Alles innerhalb des Rechtecks gehört zur Extension von H_r.
- Die bisher aufgetretenen Beispiele werden als „+“ und „-“ dargestellt.

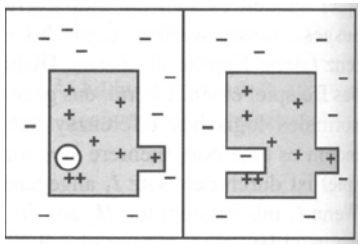


- Neues Beispiel ist falsch negativ.
- Die Hypothese besagt, dass es negativ sein muss, aber positiv ist.
- Eine Verallgemeinerung der Hypothese ist notwendig.



Aktuell beste Hypothese-Suche (3)

- Neues Beispiel ist falsch positiv.
- Die Hypothese besagt, dass es positiv sein muss, aber negativ ist.
- Eine Spezialisierung der Hypothese ist erforderlich



Current-Best-Learning

```
function CURRENT-BEST-LEARNING(examples) returns eine Hypothese

  H ← eine Hypothese, die konsistent mit dem ersten Beispiel
      in examples ist
  for each restliches Beispiel in examples do
     if e ist falsch positiv für H then
        H ← choose eine Spezialisierung von H, die mit
            examples konsistent ist
     else if e ist falsch negativ für H then
        H ← choose eine Verallgemeinerung von H, die mit
            examples konsistent ist
     if keine konsistente Verallgemeinerung/Spezialisierung gefunden
        then fail
  return H
```

Current-Best-Learning (2)

- Das erste Beispiel, X_1, ist positiv, Alternative(X_1) ist wahr
  $H_1 : \forall x\ WerdenWarten(x) \Leftrightarrow Alternative(x)$
- Das zweite Beispiel, X_2, ist negativ, H_1 sagt es als positiv voraus: Spezialisierung
  $H_2 : \forall x\ WerdenWarten(x) \Leftrightarrow Alternative(x) \wedge Gäste(x,Einige)$
- Das dritte Beispiel, X_3, ist positiv, H_2 sagt es als negativ voraus: Verallgemeinern
  $H_3: \forall x\ \ WerdenWarten(x) \Leftrightarrow Gäste(x,Einige)$
- Das vierte Beispiel, X_4 ist positiv, H_3 sagt es als negativ voraus: Verallgemeinern
  $H_4: \forall x\ WerdenWarten(x) \Leftrightarrow Gäste(x,Einige) \vee (Gäste(x,Voll) \wedge Frei/Sams(x))$

Restaurantdomäne

Bei-spiel	Attribute										Ziel
	Alt	Bar	Fr/Sa	Hung	Gäste	Preis	Regen	Reser.	Typ	Wart	Werden-Warten
X1	Ja	Nein	Nein	Ja	Eingie	€€€	Nein	Ja	Franz.	0-10	Ja
X2	Ja	Nein	Nein	Ja	Voll	€	Nein	Nein	Thai	30-60	Nein
X3	Nein	Ja	Nein	Nein	Einige	€	Nein	Nein	Burger	0-10	Ja
X4	Ja	Nein	Ja	Ja	Voll	€	Ja	Nein	Thai	10-30	Ja
X5	Ja	Nein	Ja	Nein	Voll	€€€€	Nein	Ja	Franz.	>60	Nein
X6	Nein	Ja	Nein	Ja	Einige	€€	Ja	Ja	Ital.	0-10	Ja
X7	Nein	Ja	Nein	Nein	Keine	€	Ja	Nein	Burger	0-10	Nein
X8	Nein	Nein	Nein	Ja	Einige	€€	Ja	Ja	Thai	0-10	Ja
X9	Nein	Ja	Ja	Nein	Voll	€	Ja	Nein	Burger	>60	Nein
X10	Ja	Ja	Ja	Ja	Voll	€€€	Nein	Ja	Ital.	10-30	Nein
X11	Nein	Nein	Nein	Nein	Keine	€	Nein	Nein	Thai	0-10	Nein
X12	Ja	Ja	Ja	Ja	Voll	€	Nein	Nein	Burger	30-60	Ja

Geringste Verpflichtung

- Anstatt eine Hypothese als die beste zu betrachten, werden die Hypothesen beibehalten, die mit allen bisherigen Daten konsistent sind.
- Wird bei einer neuen Instanz festgestellt, das verschiedene Hypothesen inkonsistent sind mit den Beispielen, so werden nur die Hypothesen beibehalten, die nicht ausgeschlossen wurden.
- Wenn der ursprüngliche Hypothesenraum die richtige Antwort enthalten hat, so muss die reduzierte Disjunktion ebenfalls die richtige Antwort enthalten.
- Die Menge der verbleibenden Hypothesen wird als Versionsraum bezeichnet.

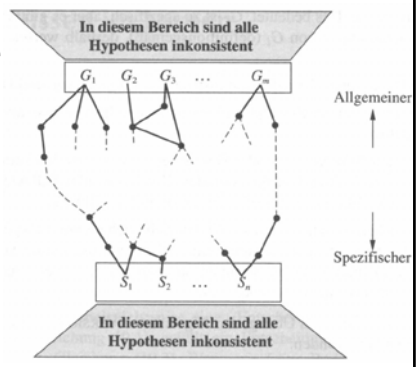
Grenzmenge

Der Hypothesenraum ist riesig. Wie können wir diese gigantische Disjunktion aufschreiben?

- Überlegung wie stellt man reelle Zahlen zwischen 1 und 2 dar?
  - Intervalldarstellung mit Angabe der Grenzen: [1,2]
- Übertragen: der Hypothesenraum ist ebenfalls geordnet, durch Verallgemeinerung/Spezialisierung. Grenzen anstelle von Punkten durch eine Menge von Hypothesen setzen.
- Der Versionsraum wird durch zwei Grenzmengen bestimmt.
  - G-Menge der allgemeinsten Grenze
  - S-Menge der spezifischsten Grenze

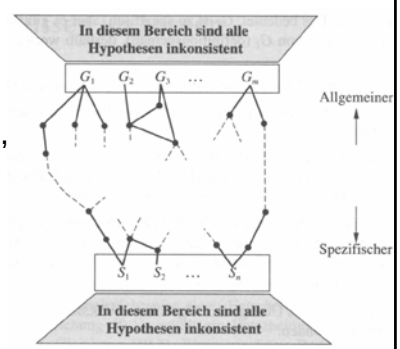
S-Menge

- Falsch positiv für S_i: S_i ist zu allgemein. Da keine konsistente Spezialisierung von S_i existiert, werfen wir S_i aus der S-Menge.
- Falsch negativ für S_i: S_i ist zu spezifisch. Wir ersetzen es durch alle unmittelbaren Verallgemeinerungen, vorausgesetzt, sie sind spezifischer als ein Element von G.



G-Menge

- Falsch positiv für G_i: G_i ist zu allgemein, deshalb ersetzen wir es durch alle unmittelbaren Spezialisierungen, vorausgesetzt, sie sind allgemeiner als ein Element von S.
- Falsch negativ für G_i: G_i ist zu spezifisch, da keine Verallgemeinerung von G_i existiert, werfen wir es aus der Menge G.



Grenzmenge

Diese Operationen werden so lange für jede neue Instanz ausgeführt, bis eines der drei Dinge passiert:

- Es gibt nur noch genau ein Konzept im Versionsraum, das wir dann als einzige Hypothese zurückgeben.
- Der Versionsraum kollabiert – *S* und *G* wird leer, d.h. es wurde keine konsistente Hypothese für die Trainingsmenge gefunden.
- Die Beispiele gehen uns aus, aber es befinden sich noch mehrere Hypothesen im Versionsraum. Das bedeutet, der Versionsraum stellt eine Disjunktion von Hypothesen dar. Wenn alle Disjunkte übereinstimmen, können wir für jedes neue Beispiel ihre Klassifizierung des Beispiels zurückgeben.

Erklärungsbasiertes Lernen (1)

- Allgemeine Regeln aus Beobachtungen extrahieren
- Beispiel: Ein Höhlenbewohner, brät eine Eidechse am Ende eines Stocks über einem Feuer. Er wird von einer begeisterten Menge seiner weniger intelligenten Zeitgenossen beobachtet, die ihre Beute mit bloßen Händen über dem Feuer halten. Diese Erfahrung ist ausreichend, um die Beobachter von einem allgemeinen Prinzip schmerzfreien Kochens zu überzeugen.
- Dieser Verallgemeinerungsprozess wird als erklärungsbasiertes Lernen (EBL) bezeichnet.
- Der Agent lernt faktisch nichts neues.
- Er hätte auch selber mit viel Rechenaufwand auf die Lösung kommen können.

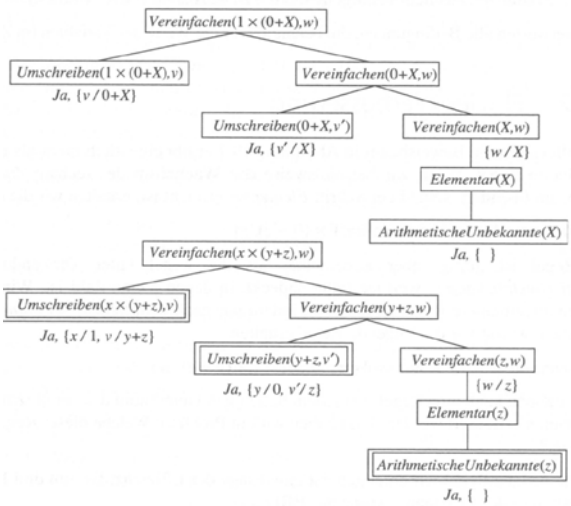
Erklärungsbasiertes Lernen (2)

- Angenommen, wir wollen 1x(0+X) vereinfachen.
- Die Wissensbasis enthält die folgende Regeln:

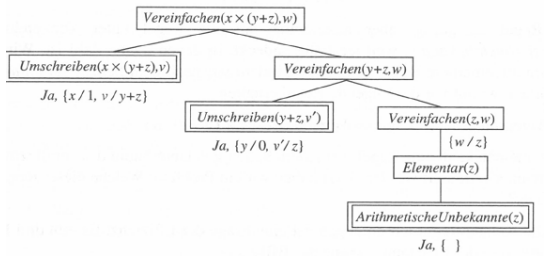
$$Umschreibung(u,v) \wedge Vereinfachen(v,w) \Rightarrow Vereinfachen(u,v)$$
$$Elementar(u) \Rightarrow Vereinachen(u,u)$$
$$ArithmetischeUnbekannte(u) \Rightarrow Elementar(u)$$
$$Zahl(u) \Rightarrow Elementar(u)$$
$$Umschreiben(1 \times u, u)$$
$$Umschreiben(0 + u, u)$$

Beweisbaum

- Die EBL-Methode erzeugt zwei Beweisbäume
  - Der erste Baum zeigt den Beweis für das ursprüngliche Problem
  - Der zweite Baum zeigt den Beweis für ein Problem, wobei alle Konstanten durch Variablen ersetzt wurden.



Allgemeine Regel ableiten



- Wir bauen eine neue Regel auf, deren linke Seite aus den Blättern des Beweisbaumes bestehen und deren rechte Seite das variablenunterlegte Ziel ist.

  Umschreiben(1x(0+z),0+z) $\wedge$ Umschreiben(0+z,z) $\wedge$ ArithmetischeUnbekannte(z) $\Rightarrow$ Vereinfachen(1x0(0+z),z)

- Verwerfen alle Bedingungen, die belanglos für die Werte der Variablen im Ziel sind.

  ArithmetischeUnbekannte(z) $\Rightarrow$ Vereinfachen(1x(0+z),z)

Lernen mit Relevanzinformation

- Bei diesem Lernschema beeinflusst das Vorwissen *Hintergrund* die **Relevanz** einer Menge von Funktionsmerkmalen für das Zielprädikat.
- Dieses Wissen erlaubt dem Agenten zusammen mit den Beobachtungen eine neue, allgemeine Regel abzuleiten.
- Diese Art der Verallgemeinerung wird als relevanzbasiertes Lernen (RBL) bezeichnet.
- RBL benutzt den Inhalt der Beobachtung, aber erzeugt keine neue Hypothese, die über den logischen Inhalt des Hintergrundwissens und der Beobachtung hinausgehen.

RBL Beispiel

- Beispiel:
  Ein Brasilienreisender trifft seinen ersten Brasilianer. Nachdem er gehört hat, dass dieser portugiesisch spricht, schließt er, dass alle Brasilianer portugiesisch sprechen. Aber obwohl sein Name Fernando war schließt er nicht, dass alle Brasilianer Fernando heißen.
- Der Brasilienreisende konnte auf Grund seines Vorwissens, dass Menschen in einem Land die selbe Sprache sprechen, darauf schließen, dass alle Brasilianer portugiesisch sprechen.

  $Nationalität(x,n) \wedge Nationalität(y,n) \wedge Sprache(x,l) \Rightarrow Sprache(y,l)$
- Der Logische Schluss

  $Nationalität(Fernando,Brasilianisch) \wedge Sprache(Fernando,Portugiesisch) \wedge Nationaliät(x,Brasilianisch) \Rightarrow Spache(x,Portugiesisch)$

Hypothesenraum festlegen

- Obwohl die Bestimmtheit einen allgemeinen Schluss unterstützt, können wir keine allgemeine Theorie aus einem Beispiel erzeugen.
- Wir können aber den Hypothesenraum, den der lernende Agent auswerten muss, reduzieren.
- Wir spezifizieren ein ausreichendes Grundvokabular, aus dem Hypothesen generiert werden können, die das Zielprädikat beschreiben.
- Eine Reduzierung der Hypothesenraumgröße macht es dem Agenten einfacher, das Zielprädikat zu lernen.

Lernen und Verwenden von Relevanzinformation

- Wir brauchen einen Lernalgorithmus, der die einfachste Bestimmtheit ermittelt, die mit den Beobachtungen konsistent sind.
- Eine Bestimmtheit $P \succ Q$ sagt, wenn Beispiele mit P übereinstimmen, dann müssen sie auch mit Q übereinstimmen.

Stichprobe	Masse	Temp.	Material	Größe	Leitfähigkeit
S1	12	26	Kupfer	3	0,59
S1	12	100	Kupfer	3	0,57
S2	24	26	Kupfer	6	0,59
S3	12	26	Blei	2	0,05
S3	12	100	Blei	2	0,04
S4	24	26	Blei	4	0,05

- Minimale konsistente Bestimmtheit

  $Material \wedge Temperatur \succ Leitfähigkeit$

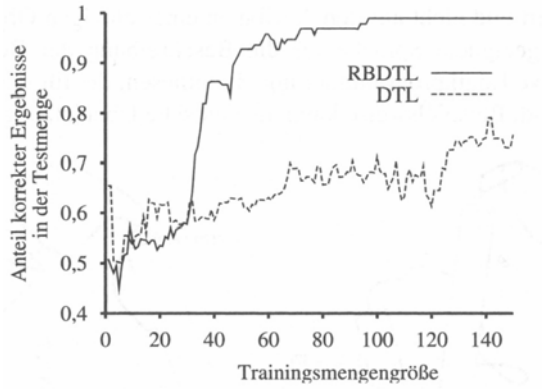
Minimal-Consitent-Det

```
function MINIMAL-CONSISTENT-DET(E, A) returns eine Attributmenge
   inputs: E, eine Beispielmenge
           A, eine Attributmenge der Größe n

   for i ← 0, …, n do
      for each Untermenge A_i von A der Größe i do
         if CONSISTENT-DET?(A_i, E) then return A_i
```

```
function CONSISTENT-DET?(A, E) returns einen Wahrheitswert
   inputs: A, eine Attributmenge
           E, eine Beispielmenge
   local variables: H, eine Hash-Tabelle

   for each Beispiel e in E do
      if ein Beispiel in H hat dieselben Werte wie e für die Attribute A
          aber eine andere Klassifizierung then return false
      Speichere die Klasse von e in H, indiziert durch die Werte für die
          Attribute A des Beispiels e
   return true
```

Leistungsvergleich

Leistungsvergleich zwischen RBDTL und Deccision-Tree-Learning für zufällig erzeugte Daten für eine Zielfunktion, die von nur fünf aus 16 Attributen abhängig ist.



Induktive Logikprogrammierung

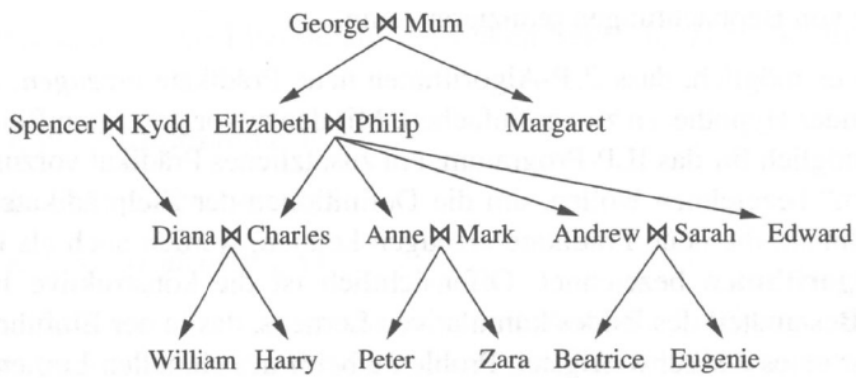
- Induktive Logikprogrammierung (ILP) kombiniert das Hintergrundwissen und die Hypothese, um ein Beispiel zu erklären.
- Hypothesen können in allgemeiner Logik erster Stufe formuliert werden. So kann auch in Umgebungen gelernt werden, die von einfacheren Systemen nicht verstanden werden.
- Die Hypothesen, die ein ILP-System erzeugt, sind relativ einfach für den Menschen zu lesen und können so auch leichter überprüft und kritisiert werden.

Beispiel Familienverhältnis

- Wir Beschreiben den Stammbaum mit Hilfe der Relationen Mutter, Vater und Verheiratet sowie den Eigenschaften Männlich und Weiblich. Beschreibung:

Vater(Philip, Charles)	*Vater(Philip, Anne)*	…
Mutter(Mum, Margaret)	*Mutter(Mum, Elizabeth)*	…
Verheiratet(Diana, Charles)	*Verheiratet(Elizabeth, Philip)*	…
Männlich(Philip)	*Männlich(Charles)*	…
Weiblich(Beatrice)	*Weiblich(Margaret)*	…

Stammbaum



Klassifizierungen

- Die Sätze der Klassifizierungen sind von dem zu lernenden Zielkonzept abhängig. Wir könnten beispielsweise *Großeltern*, *Schwager* oder *Vorfahre* lernen. Für *Großeltern* enthält die vollständige Menge für *Klassifizierungen* 20x20 = 400 Konjunkte der Form:

$$\textit{Großeltern}(\textit{Mum}, \textit{Charles}) \quad \textit{Großeltern}(\textit{Elizabeth}, \textit{Beatrice}) \quad \ldots$$
$$\neg\textit{Großeltern}(\textit{Mum}, \textit{Harry}) \quad \neg\textit{Großeltern}(\textit{Spencer}, \textit{Peter}) \quad \ldots$$

Hypothese ohne Vorwissen

- Nehmen wir an, der Agent hat kein Hintergrundwissen: *Hintergrund* ist leer. Sieht die Hypothese wie folgt aus:

$$\begin{aligned} \textit{Großeltern}(x, y) \Leftrightarrow\ & [\exists z\ \textit{Mutter}(x, z) \wedge \textit{Mutter}(z, y)] \\ \vee\ & [\exists z\ \textit{Mutter}(x, z) \wedge \textit{Vater}(z, y)] \\ \vee\ & [\exists z\ \textit{Vater}(x, z) \wedge \textit{Mutter}(z, y)] \\ \vee\ & [\exists z\ \textit{Vater}(x, z) \wedge \textit{Vater}(z, y)] \end{aligned}$$

- Attributbasierte Lernalgorithmen sind nicht in der Lage, relationale Prädikate zu lernen.

Hypothese mit Vorwissen

- Wie kann Vorwissen bei der Repräsentation der Großeltern-Definition helfen?
- Wird Beispielsweise *Hintergrund* in den Satz aufgenommen:

  $Eltern(x,y) \Leftrightarrow [Mutter(x,y) \vee Vater(x,y)]$
- Die Definition von Großeltern reduziert sich auf:

  $Großeltern(x,y) \Leftrightarrow [\exists z\ Eltern(x,z) \wedge Eltern(z,y)]$
- Hintergrundwissen kann die Größe von Hypothesen für die Erklärung von Beobachtungen stark reduzieren.

Konstruktive Induktionsalgorithmen

- Es ist möglich, das ILP-Algorithmen neue Prädikate erzeugen, um den Ausdruck erklärender Hypothesen zu vereinfachen.
- Es ist durchaus möglich, für das ILP-Programm ein zusätzliches Prädikat vorzuschlagen, das wir als „Eltern“ bezeichnen, um die Definition der Zielprädikate zu vereinfachen.
- Algorithmen, die neue Prädikate erzeugen können, werden auch als **konstruktive Induktionsalgorithmen** bezeichnet.

Induktive „Top-down“ -Lernmethode

- Wir beginnen mit einer sehr allgemeinen Regel und spezialisieren sie schrittweise, so dass sie mit den Daten übereinstimmt. Analog zum Entscheidungsbaumlernen.
- Wir verwenden Literale erster Stufe statt Attributen.
- Hypothesen bilden eine Menge von Klauseln statt einen Entscheidungsbaum.

Beispiel: FOIL

- FOIL, eines der ersten ILP-Programme
- Wir wollen die Definition des Prädikates *Großvater*(*x*,*y*) lernen.
- Wir unterscheiden positive und negative Beispiele.
  - Positive Beispiele:
    <*Georg*,*Anne*>,<*Philip*,*Peter*>,<*Spencer*,*Harry*>
  - Negative Beispiele:
    <*Georg*,*Elizabeth*>,<*Harry*,*Zara*>,<*Charles*,*Philip*>
- Da *Großvater* ein binäres Prädikat ist, stellt jedes Beispiel ein *Paar* von Objekten dar.
- Insgesamt gibt es 12 positive und 388 negative Paare.

FIOL (2)

- FOIL konstruiert eine Menge von Klauseln, die die 12 positiven Beispiele als Instanzen der *Großvater*(*x*,*y*)-Beziehung klassifizieren und gleichzeitig die 388 negativen Beispiele ausschließt.
  $\Rightarrow$ *Großvater*(*x*,*y*)
- Diese Klausel klassifiziert jedes Beispiel als positiv. Weiter spezialisieren.
  Vater(*x*,*y*) $\Rightarrow$ *Großvater*(*x*,*y*)
  Eltern(*x*,*z*) $\Rightarrow$ *Großvater*(*x*,*y*)
  Vater(*x*,*z*) $\Rightarrow$ *Großvater*(*x*,*y*)
- Wir bevorzugen die dritte Klausel. Sie stimmt mit allen 12 positiven Beispielen überein.

FIOL (3)

- Die Klausel muss weiter spezialisiert werden, um Fälle auszuschließen, wo *x* der Vater von einem *z* ist, *z* aber nicht Elternteil von *y*.
  Vater(*x*,*z*) $\wedge$ *Eltern*(*z*,*y*) $\Rightarrow$ *Großvater*(*x*,*y*)
- Damit werden alle Klauseln korrekt klassifiziert.
- Der Algorithmus erzeugt wiederholt Klauseln, wobei jeweils einzelne Literale eingefügt werden, bis die Klausel mit einer Untermenge der positiven und keinem der negativen Beispielen übereinstimmt.
- Die von der Klausel abgedeckten positiven Beispiele werden aus der Trainingsmenge entfernt.

```
function FOIL(examples, target) returns eine Menge von Horn-Klauseln
  inputs: examples, Beispielmenge
          target, ein Literal für das Zielprädikat
  local variables: clauses, eine Klauselmenge, anfänglich leer

  while examples enthält positive Beispiele do
    clause ← NEW-CLAUSE(examples, target)
    von clause abgedeckte Beispiele aus examples entfernen
    clause in clauses einfügen
  return clauses
```

```
function NEW-CLAUSE(examples, target) returns eine Horn-Klausel
  local variables: clause, eine Klausel mit target als Kopf und
                           leerem Rumpf
                   l, ein Literal, das der Klausel hinzugefügt werden
                      soll
                   extended_examples, eine Beispielmenge mit Werten
                      für neue Variablen

  extended_examples ← examples
  while extended_examples negative Beispiele enthält do
    l ← CHOOSE-LITERAL(NEW-LITERALS(clause), extended_examples)
    l dem Rumpf von clause hinzufügen
    extended_examples ← Beispielmenge, die durch Anwendung von
                        EXTEND-EXAMPLES auf jedes Beispiel in
                        extended_examples erzeugt wurde
  return clause
```

```
function EXTEND-EXAMPLE(example, literal) returns
  if example erfüllt literal
    then return die Beispielmenge, die durch Extension von example
                mit jedem möglichen Konstantenwert für jede neue
                Variable in literal erzeugt wurde
  else return die leere Menge
```


Induktives Lernen mit inversem Schließen

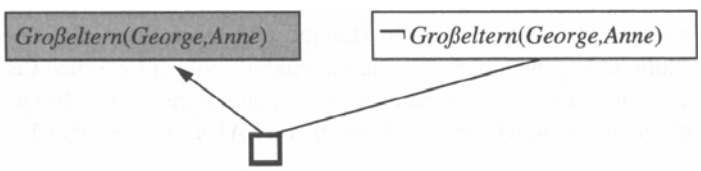
- Dieser Ansatz invertiert den normalen deduktiven Beweisprozess.
- Die inverse Resolution basiert auf der Beobachtung, dass, wenn die Beispiel-Klassifizierung aus Hintergrund $\wedge$ Hypothese $\wedge$ Beschreibung folgen, man auch in der Lage sein muss, diese Tatsache durch Resolution zu beweisen.
- Wir führen einen Rückwärtsbeweis aus, so dass wir eine Hypothese finden die durchgeht.

Inverse Resolution

- Normaler Resolutionsschritt: Zwei Klauseln, C_1 und C_2 werden resolviert und erzeugen einen Resolventen C
- Ein inverser Resolutionsschritt nimmt einen Resolventen C und erzeugt zwei Klauseln C_1 und C_2
- Alternativ kann auch ein Resolvent C und eine Klausel C_1 genommen werden und eine Klausel C_2 erzeugen.

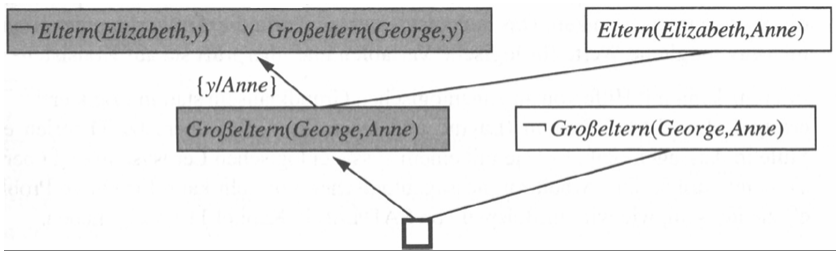
Beispiel (1)

- Führen wir einen Inversen Resolutionsprozess auf das positive Beispiel *Großeltern*(*Georg*,*Anne*) aus.
- Der Resolvent C ist die leere Klausel(ein Wiederspruch).
- Die Klausel C_2 ¬*Großeltern*(*Georg*,*Anne*)
- Die Klausel *Gorßeltern*(*Gerog*,*Anne*) für C_1 wird erzeugt.



Beispiel (2)

- Als Nächstes wird die Klausel *Großeltern*(*Georg*,*Anne*) als *C* und die Klausel *Eltern*(*Elizabeth*,*Anne*) als C_2 genommen und erzeugen die Klausel ¬*Eltern*(*Elizabeth*,*y*) ∨ *Großeltern*(*Gerog*,*y*).



Beispiel (3)

- Der letzte Schritt behandelt die Klausel ¬*Eltern*(*Elizabeth*,*y*) ∨ *Großeltern*(*Gerog*,*y*) als Resolvente.