METHODEN DER KÜNSTLICHEN INTELLIGENZ

Ipke Wachsmuth, Universität Bielefeld

Kurztext zu Teil 3: Logik und Inferenz

Dieser Abschnitt befasst sich mit der Frage: *Wie lassen sich maschinell Schlussfolgerungen aus Annahmen ziehen?* Schlussfolgern (reasoning) bezeichnet hier kognitive Verarbeitungsprozesse, mit denen aus vorhandenem Wissen bzw. Annahmen oder Vermutungen neues Wissen bzw. Annahmen oder Vermutungen gewonnen werden; "neu" heißt dabei: jetzt verfügbar, vorher nicht unmittelbar verfügbar. Die formale Betrachtung des Schlussfolgerns oder des Bildens von *Inferenzen* ist ein Anliegen der traditionellen Logik. Im Zentrum stehen dort zunächst deduktive Inferenzen, welche legitime Schlüsse erlauben. Es gibt darüber hinaus abduktive Inferenzen, mit denen Hypothesen generiert werden können, und schließlich induktive Inferenzen, mit denen sich Verallgemeinerungen gewinnen lassen (kommt in Teil 4).

Stellen wir uns nun vor, dass die Wissensbasis (auch "Datenbasis" genannt) eines intelligenten Agenten eine organisierte Menge von symbolischen Datenstrukturen enthält, die die aktuellen "beliefs" (also die für wahr gehaltenen Aussagen) des Agenten darstellen. Eine Reasoning-Komponente oder *Inferenzmaschine* (auch "Gatekeeper" genannt) ist zuständig für das Errechnen von Schlussfolgerungen und das Hinzufügen – oder Löschen – von Einträgen in der Wissensbasis. Im Weiteren wird aufgezeigt, wie man von der Faktenebene, die sich auf darzustellende Sachverhalte eines Weltausschnitts bezieht, zu einer Beschreibung in symbolischer Form gelangt und schließlich zu konkreten Repräsentationen, welche maschinell verarbeitbare Ausdrücke bereitstellen. Die Prädikatenlogik (erster Ordnung), ihre Formeln und Inferenzregeln sind hier Bezugspunkt der Betrachtung.

In der Prädikatenlogik benutzt man Prädikate, deren Argumentstellen mit Termen gefüllt sind, um (atomare) Formeln zu bilden. Mit Junktoren (`and` für *und*, `or` für *oder*, `if` für *wenn-dann*) können atomare zu komplexen Formeln zusammengesetzt werden. Quantoren (`forall` für *für alle* und `exists` für *es existiert*) werden gebraucht, um Axiome zu formulieren wie:

```
(forall (x) (if (in x antarctica)(temperature x cold)))
(forall (x) (if (person x)(exists (y)(head-of x y))))
(forall (t1 t2 t3)(if (and (before t1 t2)(before t2 t3))
                      (before t1 t3)) )
```

Die Schlüsselidee ist (betrifft die Semantik): Jeder solche Ausdruck notiert (auch "denotiert") etwas über einen betrachteten Weltausschnitt. *Terme* notieren Individuen oder Klassen von Individuen; *Formeln* notieren Sachverhalte. Die in den Beispielen gewählten "sprechenden Namen" mögen eine Vorstellung von den notierten Sachverhalten vermitteln. Die Wahrheit einer Formel kann "weltabhängig" sein, nämlich abhängig davon, ob in der betrachteten Welt der mit der Formel notierte Sachverhalt faktisch wahr oder falsch ist.

Formeln, die zur Menge der "beliefs" einer Wissensbasis hinzugenommen werden, werden *Assertionen* genannt. In der Regel enthält eine KI-Wissensbasis Hunderte von Assertionen oder noch viel mehr. Im einfachsten Design enthält sie Datenstrukturen in Form prädikatenlogischer Formeln, und die Inferenzmaschine kann nur die gewöhnlichsten Inferenzregeln

(Schlussregeln) des Prädikatenkalküls ausführen. Als Basiskommandos zur Benutzung der Wissensbasis durch die Inferenzmaschine betrachten wir:

`assert:` fügt jeweils eine Proposition (Aussage-Formel) in die Wissensbasis ein
`retract:` nimmt Propositionen wieder heraus ("zieht sie zurück")
`query:` gibt eine Frage-Formel vor (als syntaktisches Muster) und versucht, darauf passende Antwort-Formeln aus den assertierten Formeln zu deduzieren

Beispiel

```
assert:   (forall(x) (if (inst x canary)(color x yellow)))
assert:   (inst tweety canary)      (notiert: Tweety ist ein Kanarienvogel)
query:    (color tweety yellow)     (Die Antwort sollte positiv sein.)
```

Mit der klassischen deduktiven Inferenzregel der Logik, dem Modus Ponens (I), kommt man hier aufgrund des Vorliegens einer Variablen (x) im zuerst assertierten Ausdruck nicht weiter:

(I) *Modus Ponens*
Gegeben, dass gilt: `(if` *p q* `)` und *p*
inferiere, dass gilt: *q*

Es lassen sich nämlich in den beiden oben assertierten Ausdrücken keine übereinstimmenden Muster für *p* auffinden. Ziehen wir nun folgende weitere deduktive Inferenzregel heran:

(II) *Universelle Einsetzung*
Gegeben, dass gilt: `(forall (-`*vars*`-)` *p* `)`
inferiere, dass gilt: *p* mit allen Vorkommen aller Variablen *-vars-* in *p* durch den gleichen Term eingesetzt

Dann folgt mit Regel (II) bei Einsetzung des Konstantenterms `tweety` für `x` in der ersten assertierten Formel `(if (inst tweety canary)(color tweety yellow))`, und nun lässt sich mit Modus Ponens (I) die Antwort `(color tweety yellow)` inferieren. Hier versteckt sich allerdings noch ein schwerer wiegendes Problem – durch die "universelle" Einsetzung wird nämlich *jeder* in der Wissensbasis vorhandene Term (z.B. jede Konstante wie `bill-22`, `block-1` etc.) eingesetzt, so dass man eine Menge unnützer Formeln erhält.

Quantorenelimination, Skolemisierung

Bevor wir uns gleich einem geschickteren Vorgehen, der *Unifikation*, zuwenden, widmen wir uns erst einem – auf dem Weg zur maschinenverarbeitbaren Formeldarstellung zu lösenden – Problem: der Quantorenelimination. Die Quantoren (`forall`, `exists`) stellen nämlich quasi globale Zeichen in einer Formel dar, deren Behandlung durch einen lokal muster-verarbeitenden Prozess vor Schwierigkeiten stellt. Jedoch gibt einen Weg, die Quantoren in einer konkreten Repräsentation implizit zu berücksichtigen (*implicit-quantifier form*):

1) Allquantifizierte Formeln werden, am Beispiel, wie folgt umgeschrieben:
`(forall(x)(if (inst x canary)(color x yellow)))`
wird zu `(if (inst ?x canary)(color ?x yellow))`

Die `?`-markierte "Matchvariable" bewirkt, dass die Inferenzmaschine jeden in der Wissensbasis gefundenen Term einsetzen darf: die Semantik des Allquantors wird operational rekonstruiert.

2) Existenzquantifizierte Formeln werden im einfachsten Fall wie folgt umgeschrieben:
`(exists(x)(and (nudist x)(party x uni-bielefeld)))`
wird zu `(and (nudist sk-1)(party sk-1 uni-bielefeld))`

Idee dabei: Wenn solch ein Nudist existiert (in Bielefelds Uni durchaus eine Realität), kann man ihn irgendwie benennen, und wenn mehrere existieren, ist der Benannte einer davon stellvertretend. Existenzquantifizierte Variablen, die im Einflussbereich eines Allquantors stehen, werden durch Funktionsausdrücke der Art `(neue-funktion ?x)` ersetzt. Das allgemeine Verfahren für die Elimination von Existenzquantoren aus prädikatenlogischen Formeln heißt *Skolemisierung*. Zusammen mit der Ersetzung von allquantifizierten Variablen durch Matchvariable erhält man (erfüllbarkeitsäquivalente) quantorenfreie Formeln, die von der Inferenzmaschine verarbeitet werden können. Es läuft etwa ab wie folgt (Varianten sind möglich):

- Zunächst werden die Formeln durch Äquivalenzumformungen in sog. Pränexform gebracht, d.h. alle Quantoren stehen vorn (dabei klärt sich der "wirkliche Typ eines Quantors")
- existenzquantifizierte Variable werden durch Skolemfunktionen ersetzt; die Argumente dieser Funktionen sind die allquantifizierten Variablen, in deren Skopus der Existenzquantor liegt
- falls zwei verschiedene allquantifizierte Variablen zufällig den gleichen Namen haben, werden neue Namen eingeführt (Standardisierung)
- dann wird jede allquantifizierte Variable durch eine `?`-markierte Variable ersetzt (die Allquantifizierung bleibt implizit dadurch gegeben, dass solche Variablen mit allen Termen "matchen").

Unifikation, Verkettung (Chaining)

Mit den obigen Vorbereitungen betrachten wir jetzt statt Modus Ponens (I) und universeller Einsetzung (II) die folgende Regel, die eine Verallgemeinerung des Modus Ponens darstellt:

(III) Aus *p'* und `(if p q )` inferiere *q'*
wobei *p* mit *p'* *unifizieren muss* und die resultierende Substitution θ auf *q* angewandt wird, wodurch man *q'* erhält
(d.h. $p\theta = p'$ und $q\theta = q'$).

"Unifizieren" heißt dabei, dass Variablen derart substituiert werden, dass zwei Ausdrücke gleich werden. Allgemein ist eine Substitution eine Menge von Variable-Wert-Paaren; jedes solche Paar heißt Variablenbindung. Jede Variable wird "durch die Substitution gebunden" genannt. Die Anwendung einer Substitution auf eine Formel heißt, jedes Vorkommen einer dadurch gebundenen Variablen durch ihren Wert zu ersetzen; Werte können auch andere Variablen sein. Betrachten wir noch einmal das frühere Tweety-Beispiel, hier bereits in Skolemform assertiert:

```
assert:   (if (inst ?x canary)(color ?x yellow))
assert:   (inst tweety canary)
```

Dann unifiziert `(inst ?x canary)` mit `(inst tweety canary)` bei Substitution: θ = {x = tweety}. Durch Anwendung dieser Substitution auf `(color ?x yellow)` erhält man, ohne Umstände wie oben bei (II): `(color tweety yellow)`.

Die unifizierende Substitution θ wird ein Unifikator genannt, und als einen *allgemeinsten Unifikator* (most general unifier, MGU) bezeichnet man die unifizierende Substitution, die am wenigsten Spezialisierungen vornimmt. Für jede unifizierbare Formelmenge existiert ein solcher MGU (er ist eindeutig bis auf Varianten, d.h. Formeln, die einander subsumieren).

Vorwärts- und Rückwärtsverkettung (Chaining). In allgemeiner Form lautet obige Regel (III): Aus `p'` und `(if p q)` inferiere `q'`, wobei `p` mit `p'` unifiziert mit MGU θ und `q'`$= q\theta$; sie erlaubt "vorwärtsverkettende" Inferenzen: von Fakten zu Schlussfolgerungen. Im Normalfall werden die meisten Inferenzen in deduktiven Systemen aber zur Query Time vorgenommen. Das heißt, die Inferenzmaschine wird eine Implikation `(if p q)` zwar assertieren, aber nichts damit anstellen, bis eine Frage der Form `q'` gestellt wird. Wird nach `q'` gefragt (als "goal"), und `q,q'` haben den MGU θ, so wird `p'`$= p\theta$ als subgoal aufgeworfen, usf. Das derartige Aufwerfen von subgoals, sub-subgoals etc. wird als backward chaining bezeichnet. Der Basismechanismus des Backward Chaining bezieht sich auf *nichtkonjunktive goals* und verkettet Inferenzen rückwärts. Zentral für backward chaining sind Antwortsubstitutionen, z.B.

```
assert:   (inst tweety canary)
assert:   (if (inst ?x canary)(color ?x yellow))
```

Das goal `(Show: (color ?y yellow))` unifiziert mit der zweiten Assertion zu
`(Show: (inst ?y canary))` mit θ = `{x = ?y}`; und mit der ersten
dann zu `(inst tweety canary)` mit ψ = `{y = tweety}`

D.h. die Frage "*Existiert etwas, das gelb ist?*" führt zu der Antwort "*Ja,* `tweety` *ist gelb*", und zwar auf Basis der Antwortsubstitution `{x = tweety}`. (Zur Kennzeichnung von goal-Formeln wird der "Pseudo-Junktor" `Show:` verwendet; weitere Details müssen hier fortfallen.) Als Suchstruktur bei *konjunktiven goals* von der Form `(if (and p1 p2) q)` werden Goal Trees eingesetzt (siehe Teil 2); die UND-Knoten enthalten dabei als *constraints,* dass die Variablenbindungen gleichnamiger Variablen in konjunktiven Teillösungen auch tatsächlich gleich sind.

Bemerkungen. Die hier einblickhaft beschriebenen Verfahren sind Spezialfälle des umfassenderen Resolutionsverfahrens, auf dem viele Algorithmen für das rechnerische Schlussfolgern – sog. Theorembeweiser – beruhen; sie werden im Gebiet "Automatisches Beweisen" untersucht. Sie alle sind aus dem allgemeinen Resolutionsverfahren ("complete resolution") hervorgegangen, das J.A. Robinson 1965 als "eine Maschinen-orientierte Logik" erfunden hat.

Indexing; Inferenz durch Graphsuche

Eine wichtige Frage bei der konkreten Repräsentation maschinenverarbeitbarer Symbolstrukturen ist es, wie man den Zugriff auf Einträge der Wissensbasis geschickt organisiert. Wie gesagt kann eine KI-Wissensbasis Hunderte oder noch viel mehr Assertionen enthalten, die nicht allesamt durchmustert werden sollen, wenn Inferenzprozesse ablaufen; mit zunehmender Größe der Wissensbasis würde das Schließen immer langsamer. Hierfür sind viele Verfahren entwickelt worden, von denen nur Basisansätze betrachtet seien. Beim *Indexieren prädikatenlogischer Formeln* werden Formelausdrücke mit einem Schlüssel versehen, der ausdrückt, an welcher Position der Formel ein bestimmter Ausdruck steht (in der Regel wird jede einzelne Formel mehrfach indexiert). Ein grundsätzlich anderer Ansatz ist es, die Wissensbasis in Form eines semantischen Netzes aufzubauen, das über Pointer-Strukturen den Wissenszugriff unterstützt und Inferenzverfahren über *Graphsuchalgorithmen* abwickelt.