Künstliche Intelligenz

Ein moderner Ansatz

4., aktualisierte Auflage

Stuart Russell
Peter Norvig

Pearson

EXTRAS ONLINE

Prädikaten für diese eine Version des Ereigniskalküls[5] lautet:

$T(f, t_1, t_2)$	Fluent f ist wahr für alle Zeitpunkte zwischen t_1 und t_2
$Passiert(e, t_1, t_2)$	Ereignis e beginnt zum Zeitpunkt t_1 und endet zum Zeitpunkt t_2
$Initiieren(e, f, t)$	Ereignis e bewirkt, dass Fluent f zum Zeitpunkt t wahr ist
$Beenden(e, f, t)$	Ereignis e bewirkt, dass Fluent f zum Zeitpunkt t nicht mehr wahr ist
$Initiiert(f, t_1, t_2)$	Fluent f ist irgendwann zwischen t_1 und t_2 wahr
$Beendet(f, t_1, t_2)$	Fluent f hört irgendwann zwischen t_1 und t_2 auf, wahr zu sein
$t_1 < t_2$	Zeitpunkt t_1 liegt vor Zeitpunkt t_2

Wir können die Auswirkungen eines Flugereignisses beschreiben:

$$E = Flüge(a, hier, dort) \wedge Passiert(E, t_1, t_2) \Rightarrow Beenden(E, In(a, hier), t_1) \wedge Initiieren(E, In(a, dort), t_2)$$

Wir nehmen ein bestimmtes Ereignis an, *Start*, das den Anfangszustand beschreibt, indem es sagt, welche Fluents zum Startzeitpunkt wahr (mit *Initiieren*) oder falsch sind (mit *Beendet*). Wir können dann beschreiben, welche Fluents zu welchen Zeitpunkten wahr sind, und zwar mit einem Paar von Axiomen für T und $\neg T$, die das gleiche allgemeine Format wie die Nachfolgezustandsaxiome aufweisen: Nehmen wir an, dass zwischen den Zeitpunkten t_1 und t_3 ein Ereignis eintritt, das zum Zeitpunkt t_2 irgendwo in diesem Zeitintervall den Wert des Fluent f ändert, indem es ihn entweder initiiert (wahr macht) oder beendet (falsch macht). Wenn dann zum Zeitpunkt t_4 in der Zukunft kein anderes dazwischenliegendes Ereignis den Fluent verändert hat (entweder beendet oder initiiert), wird der Fluent seinen Wert behalten. Formal lauten die Axiome:

$$Passiert(e, t_1, t_3) \wedge Initiieren(e, f, t_2) \wedge \neg Beendet(f, t_2, t_4) \wedge t_1 \leq t_2 \leq t_3 \leq t_4 \Rightarrow T(f, t_2, t_4)$$
$$Passiert(e, t_1, t_3) \wedge Beenden(e, f, t_2) \wedge \neg Initiiert(f, t_2, t_4) \wedge t_1 \leq t_2 \leq t_3 \leq t_4 \Rightarrow \neg T(f, t_2, t_4) ,$$

wobei *Beendet* und *Initiiert* definiert sind durch:

$$Beendet(f, t_1, t_5) \Leftrightarrow \exists e, t_2, t_3, t_4 \; Passiert(e, t_2, t_4) \wedge Beenden(e, f, t_3) \wedge t_1 \leq t_2 \leq t_3 \leq t_4 \leq t_5$$
$$Initiiert(f, t_1, t_5) \Leftrightarrow \exists e, t_2, t_3, t_4 \; Passiert(e, t_2, t_4) \wedge Initiieren(e, f, t_3) \wedge t_1 \leq t_2 \leq t_3 \leq t_4 \leq t_5$$

Den Ereigniskalkül können wir erweitern, um simultane Ereignisse (z. B. zwei Personen, die zum Wippen benötigt werden), exogene Ereignisse (z. B. Wind, der ein Objekt bewegt), stetige Ereignisse (z. B. die ansteigende Flut), nichtdeterministische Ereignisse (z. B. Münzwurf mit Kopf oder Zahl) und andere Komplikationen darzustellen.

10.3.1 Zeit

Ereigniskalküle eröffnen uns die Möglichkeit, über Zeitpunkte und Zeitintervalle zu sprechen. Wir werden zwei Arten von Zeitintervallen betrachten: Momente und ausgedehnte Intervalle. Der Unterschied ist, dass nur Momente die Dauer null haben:

$$Partition(\{Momente, AusgedehnteIntervalle\}, Intervalle)$$
$$i \in Momente \Leftrightarrow Dauer(i) = Sekunden(0)$$

5 Unsere Version basiert auf Shanahan (1999), jedoch mit einigen Änderungen.

Als Nächstes führen wir eine Zeitskala ein und ordnen Momenten Punkte auf dieser Skala zu, sodass wir absolute Zeiten erhalten. Die Zeitskala ist willkürlich gewählt; wir messen in Sekunden und sagen, dass der Moment um Mitternacht (GMT) am 1. Januar 1900 die Zeit 0 hat. Die Funktionen *Anfang* und *Ende* wählen den ersten und den letzten Moment in einem Intervall aus, und die Funktion *Zeit* liefert für einen Moment den Punkt auf der Zeitskala zurück. Die Funktion *Dauer* gibt die Differenz zwischen der Endzeit und der Anfangszeit an.

$$\begin{aligned}
&Intervall(i) \Rightarrow Dauer(i) = (Zeit(Ende(i)) - Zeit(Anfang(i)))\\
&Zeit(Anfang(AD1900)) = Sekunden(0)\\
&Zeit(Anfang(AD2001)) = Sekunden(3187324800)\\
&Zeit(Ende(AD2001)) = Sekunden(3218860800)\\
&Dauer(AD2001) = Sekunden(31536000)
\end{aligned}$$

Um diese Zahlen leichter lesbar zu machen, führen wir auch eine Funktion *Datum* ein, die sechs Argumente (Stunden, Minuten, Sekunden, Tag, Monat und Jahr) entgegennimmt und einen Zeitpunkt zurückgibt:

$$\begin{aligned}
&Zeit(Beginn(AD2001)) = Datum(0, 0, 0, 1, 1, 2001)\\
&Datum(0, 20, 21, 24, 1, 1995) = Sekunden(3000000000)
\end{aligned}$$

Zwei Intervalle treffen sich, wenn die Endzeit des ersten gleich der Anfangszeit des zweiten ist. Die vollständige Menge von Intervallbeziehungen (Allen, 1983) ist nachfolgend und in ► Abbildung 10.2 dargestellt:

$$\begin{aligned}
Treffen(i,j) &\Leftrightarrow Ende(i) = Anfang(j)\\
Vor(i,j) &\Leftrightarrow Ende(i) < Anfang(j)\\
Nach(j,i) &\Leftrightarrow Vor(i,j)\\
Während(i,j) &\Leftrightarrow Anfang(j) < Anfang(i) < Ende(i) < Ende(j)\\
Überlappen(i,j) &\Leftrightarrow Anfang(i) < Anfang(j) < Ende(i) < Ende(j)\\
Beginnen(i,j) &\Leftrightarrow Anfang(i) = Anfang(j)\\
Beenden(i,j) &\Leftrightarrow Ende(i) = Ende(j)\\
IstGleich(i,j) &\Leftrightarrow Anfang(i) = Anfang(j) \wedge Ende(i) = Ende(j)
\end{aligned}$$

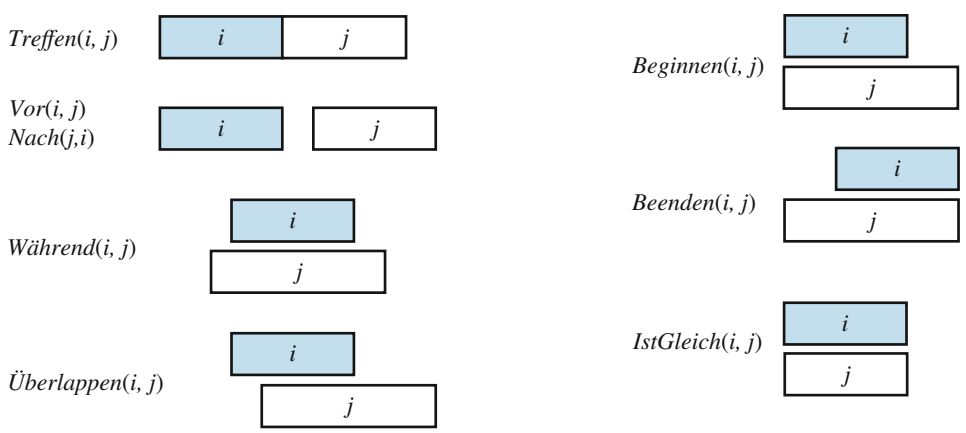

Abbildung 10.2: Prädikate für Zeitintervalle.

Diese haben alle ihre intuitive Bedeutung, mit Ausnahme von *Überlappen*: Wir neigen dazu, Überlappen als symmetrisch zu betrachten (wenn i j überlappt, dann überlappt j auch i), aber in dieser Definition ist $Überlappen(i, j)$ nur wahr, wenn i vor j beginnt. Die Praxis hat gezeigt, dass diese Definition zum Schreiben von Axiomen sehr nützlich ist. Um zu sagen, dass die Regentschaft von Elisabeth II. unmittelbar auf die von Georg VI. folgte und sich die Herrschaft von Elvis mit den 1950er Jahren überschnitt, können wir Folgendes schreiben:

$$Treffen(RegiertVon(GeorgeVI), RegiertVon(ElizabethII))$$
$$Überlappen(Fünfziger, RegiertVon(Elvis))$$
$$Anfang(Fünfziger) = Anfang(AD1950)$$
$$Ende(Fünfziger) = Ende(AD1959)$$

10.3.2 Fluents und Objekte

Physische Objekte können als verallgemeinerte Ereignisse betrachtet werden, da ein physisches Objekt ein Teil des Raums und der Zeit ist. Zum Beispiel kann man sich *USA* als ein Ereignis vorstellen, das 1776 als Zusammenschluss von 13 Staaten begann und heute als Zusammenschluss von 50 Staaten immer noch fortbesteht. Wir können die sich ändernden Eigenschaften von *USA* mithilfe von Zustandsfluents beschreiben, wie z. B. $Bevölkerung(USA)$. Eine Eigenschaft der USA, die sich alle vier oder acht Jahre ändert, es sei denn es geschieht ein Unglück, ist ihr Präsident. Man könnte vorschlagen, dass $Präsident(USA)$ ein logischer Begriff ist, der zu unterschiedlichen Zeiten ein jeweils unterschiedliches Objekt bezeichnet.

Leider ist dies nicht möglich, da ein Term genau ein Objekt in einer gegebenen Modellstruktur beschreibt. (Der Term $Präsident(USA, t)$ kann je nach dem Wert von t verschiedene Objekte darstellen, aber unsere Ontologie hält Zeitindizes getrennt von Fluents.) Die einzige Möglichkeit ist, dass $Präsident(USA)$ ein einzelnes Objekt bezeichnet, das zu verschiedenen Zeiten aus verschiedenen Personen besteht. Dieses Objekt ist von 1789 bis 1797 George Washington, von 1797 bis 1801 John Adams, und so weiter (► Abbildung 10.3). Um zu sagen, dass George Washington das ganze Jahr 1790 über Präsident war, können wir schreiben:

$$T(IstGleich(Präsident(USA), GeorgeWashington), Anfang(AD1790), Ende(AD1790))$$

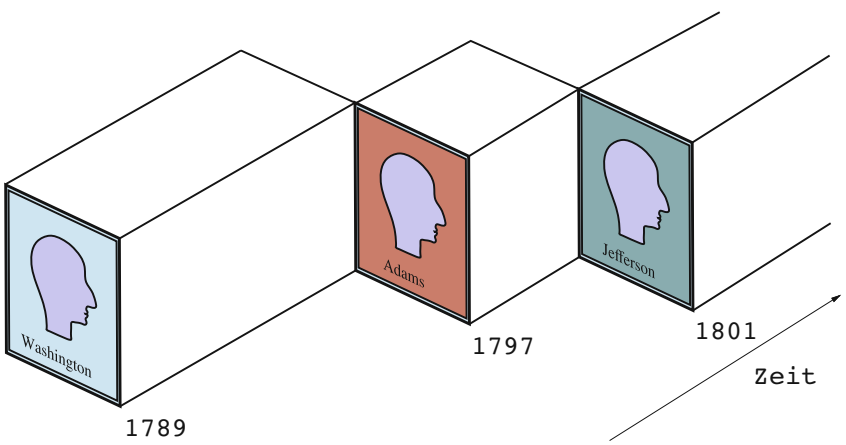

Abbildung 10.3: Eine schematische Darstellung des Objekts $Präsident(USA)$ für die ersten Jahre.

Wir verwenden das Funktionssymbol *IstGleich* anstelle des logischen Standardprädikats =, weil wir kein Prädikat als Argument für *T* angeben können und weil die Interpretation *nicht* lautet, dass *GeorgeWashington* und *Präsident*(*USA*) im Jahr 1790 logisch identisch sind; logische Identität kann sich nicht mit der Zeit ändern. Die Identität besteht zwischen den Untereignissen der Objekte *Präsident*(*USA*) und *GeorgeWashington*, die durch den Zeitraum 1790 definiert sind.

10.4 Mentale Objekte und Modallogik

Die Agenten, die wir bisher konstruiert haben, haben gewisse Überzeugungen und können neue Überzeugungen ableiten. Doch keiner von ihnen hat Wissen *über* Überzeugungen oder *über* Ableitung. Das Wissen über das eigene Wissen und die eigenen Schlussfolgerungsprozesse ist nützlich, um die Inferenz zu steuern. Angenommen, Alice fragt: „Was ist die Quadratwurzel aus 1.764?“ und Bob antwortet: „Ich weiß es nicht.“ Wenn Alice darauf besteht, dass Bob „intensiver nachdenkt“, sollte er erkennen, dass sich diese Frage mit etwas mehr Nachdenken tatsächlich beantworten lässt. Lautet die Frage hingegen „Sitzt der Präsident im Moment?“, dann sollte Bob erkennen, dass selbst intensiveres Nachdenken hier kaum hilft. Auch das Wissen über das Wissen anderer Agenten ist wichtig; Bob sollte erkennen, dass der Präsident es weiß.

Was wir brauchen, ist ein Modell der mentalen Objekte, die sich im Kopf von jemandem (oder in einer Wissensbasis) befinden, sowie der mentalen Prozesse, die diese mentalen Objekte manipulieren. Das Modell muss nicht detailliert sein. Wir müssen nicht in der Lage sein vorherzusagen, wie viele Millisekunden ein bestimmter Agent braucht, um einen Schluss zu ziehen. Wir sind schon zufrieden, wenn wir feststellen können, dass der Präsident weiß, ob er sitzt oder nicht.

Wir beginnen mit den **propositionalen Einstellungen**, die ein Agent gegenüber mentalen Objekten haben kann: Einstellungen wie *Überzeugt_sein*, *Wissen*, *Wollen* und *Informieren*. Die Schwierigkeit besteht darin, dass sich diese Einstellungen nicht wie „normale“ Prädikate verhalten. Nehmen wir zum Beispiel an, wir behaupten, dass Lois weiß, dass Superman fliegen kann:

$$Wissen(Lois, KannFliegen(Superman))$$

Dieser Ausdruck hat den kleinen Schönheitsfehler, dass wir uns *KannFliegen*(*Superman*) normalerweise als Satz vorstellen, während er hier als Term erscheint. Dieses Problem lässt sich beheben, indem man *KannFliegen*(*Superman*) reifiziert und damit zu einem Fluent macht. Schwerwiegender ist folgendes Problem: Wenn es stimmt, dass Clark Kent Superman ist, müssen wir daraus schließen, dass Lois weiß, dass Clark fliegen kann, was falsch ist, weil (in den meisten Versionen der Geschichte) Lois *nicht* weiß, dass Clark Superman ist.

$$(Superman = Clark) \wedge Wissen(Lois, KannFliegen(Superman)) \models Wissen(Lois, KannFliegen(Clark))$$

Dies ist eine Folge der Tatsache, dass Gleichheitsschlussfolgern in Logik integriert ist. Normalerweise ist das eine gute Sache, denn wenn unser Agent weiß, dass $2 + 2 = 4$ und $4 < 5$ ist, dann wollen wir, dass unser Agent weiß, dass $2 + 2 < 5$ ist. Diese Eigenschaft wird **referenzielle Transparenz** genannt – es spielt keine Rolle, welchen Term eine Logik verwendet, um auf ein Objekt zu verweisen, was zählt, ist das Objekt, das vom Term benannt wird. Aber für propositionale Einstellungen wie *glauben* und *wissen* hätten wir gerne referenzielle Opazität (Undurchsichtigkeit) – die verwendeten Terme spielen eine Rolle, weil nicht alle Agenten wissen, welche Terme koreferenziell sind.

Wir könnten es mit weiterem Reifizieren versuchen: beispielsweise ein Objekt, das Clark/Superman repräsentiert, ein weiteres Objekt, das die Person repräsentiert, die Lois als Clark

kennt, und noch ein weiteres für die Person, die Lois als Superman kennt. Diese aufgeblähte Anzahl von Objekten bedeutet jedoch, dass die Sätze, die wir schreiben wollen, schnell immer wortreicher und unhandlicher werden.

Zur Lösung dieses Problems wurde die **Modallogik** entwickelt. Reguläre Logik befasst sich mit nur einer Modalität, der Modalität der Wahrheit, die es uns erlaubt, „P ist wahr“ oder „P ist falsch“ auszudrücken. In der Modallogik gibt es spezielle **Modaloperatoren**, die Sätze (und nicht Terme) als Argumente übernehmen. Zum Beispiel wird „A weiß P“ durch die Notation $\mathbf{K}_A P$ dargestellt, wobei $\mathbf{K}$ der Modaloperator für Wissen ist. Er übernimmt zwei Argumente, einen Agenten (geschrieben als tiefgestelltes Zeichen) und einen Satz. Die Syntax der Modallogik ist die gleiche wie die der Prädikatenlogik, mit dem Unterschied, dass auch Sätze mit Modaloperatoren gebildet werden können.

Die Semantik der Modallogik ist etwas komplizierter. In der Prädikatenlogik enthält ein **Modell** eine Menge von Objekten sowie eine Interpretation, die jeden Namen dem entsprechenden Objekt, der Relation oder der Funktion zuordnet. In der Modallogik wollen wir sowohl die Möglichkeit in Betracht ziehen können, dass Supermans geheime Identität Clark ist, als auch die Möglichkeit, dass sie es nicht ist.

Daher benötigen wir ein komplizierteres Modell, das aus einer Sammlung **möglicher Welten** und nicht nur aus einer einzigen wahren Welt besteht. Die Welten sind in einem Graphen durch **Erreichbarkeitsrelationen** verbunden, und zwar eine Relation für jeden Modaloperator. Wir sagen, dass die Welt w_1 von der Welt w_0 aus in Bezug auf den Modaloperator $\mathbf{K}_A$ zugänglich ist, wenn alles in w_1 mit dem konsistent ist, was A in w_0 weiß. Betrachten wir ein Beispiel: In der realen Welt ist Bukarest die Hauptstadt von Rumänien, aber für einen Agenten, der das nicht weiß, ist eine Welt zugänglich, in der die Hauptstadt von Rumänien z. B. Sofia ist. Hoffentlich ist eine Welt, in der $22 + 2 = 5$ gilt, für keinen Agenten zugänglich.

Im Allgemeinen ist ein Wissensatom $\mathbf{K}_A P$ in der Welt w genau dann wahr, wenn P in jeder Welt wahr ist, die von w aus zugänglich ist. Der Wahrheitswert komplexerer Sätze wird durch rekursive Anwendung dieser Regel und der normalen Regeln der Prädikatenlogik abgeleitet. Das bedeutet, dass Modallogik verwendet werden kann, um über verschachtelte Wissenssätze zu schlussfolgern: Was ein Agent über das Wissen eines anderen Agenten weiß. Zum Beispiel können wir sagen, dass selbst wenn Lois nicht weiß, ob Supermans geheime Identität Clark Kent ist, so weiß sie zumindest, dass Clark es weiß:

$$\mathbf{K}_{Lois}[\mathbf{K}_{Clark} Identität(Superman, Clark) \vee \mathbf{K}_{Clark} \neg Identität(Superman, Clark)]$$

Die Modallogik löst einige knifflige Probleme beim Zusammenspiel von Quantoren und Wissen. Der Satz „Bond weiß, dass jemand ein Spion ist“ ist zweideutig. Die erste Lesart ist, dass es eine bestimmte Person gibt, von der Bond weiß, dass sie ein Spion ist; dies können wir folgendermaßen schreiben:

$$\exists\, x \;\; \mathbf{K}_{Bond} Spion(x)\,,$$

was in der Modallogik bedeutet, dass es ein x gibt, von dem Bond in allen zugänglichen Welten weiß, dass es ein Spion ist. Die zweite Lesart ist, dass Bond nur weiß, dass es mindestens einen Spion gibt:

$$\mathbf{K}_{Bond} \exists\, x \;\; Spion(x)$$

Der modallogischen Interpretation zufolge gibt es in jeder zugänglichen Welt ein x, das ein Spion ist, das aber nicht in jeder Welt das gleiche x sein muss.

Da wir nun einen Modaloperator für Wissen haben, können wir Axiome dafür schreiben. Zunächst können wir sagen, dass Agenten in der Lage sind, Schlussfolgerungen zu ziehen; wenn ein Agent P kennt und weiß, P impliziert Q, dann kennt der Agent Q:

$$(\mathbf{K}_a P \wedge \mathbf{K}_a (P \;\Rightarrow\; Q)) \;\Rightarrow\; \mathbf{K}_a Q$$

Aus dieser (und einigen anderen Regeln über logische Identitäten) können wir schließen, dass $\mathbf{K}_A(P \vee \neg P)$ eine Tautologie ist; jeder Agent weiß, dass jede Aussage P wahr oder falsch ist. Auf der anderen Seite ist $(\mathbf{K}_A P) \vee (\mathbf{K}_A \neg P)$ keine Tautologie; im Allgemeinen gibt es viele Aussagen, von denen ein Agent nicht weiß, ob sie wahr sind, und nicht weiß, ob sie falsch sind.

Es heißt (zurückgehend auf Plato), dass Wissen gerechtfertigter wahrer Glaube ist. Das heißt, wenn etwas wahr ist, wenn Sie es glauben und wenn Sie einen unanfechtbar guten Grund dafür haben, dann wissen Sie es. Wenn Sie also etwas wissen, muss es wahr sein. Damit haben wir das folgende Axiom:

$$\mathbf{K}_a P \Rightarrow P$$

Außerdem sind logische Agenten (im Gegensatz zu einigen Menschen) in der Lage, über ihr eigenes Wissen genau zu reflektieren – sie haben dann Introspektion. Wenn Sie etwas wissen, dann wissen Sie, dass Sie es wissen:

$$\mathbf{K}_a P \Rightarrow \mathbf{K}_a(\mathbf{K}_a P)$$

Ähnliche Axiome können wir für Glauben[6] (oft mit **B** für *Belief* bezeichnet) und andere Modalitäten definieren. Ein Problem mit dem Ansatz der Modallogik ist jedoch, dass er **logische Allwissenheit** auf Seiten der Agenten voraussetzt. Das heißt, wenn ein Agent eine Menge von Axiomen kennt, dann kennt er auch alle Konsequenzen dieser Axiome. Selbst für den etwas abstrakten Begriff des Wissens steht das Ganze schon auf wackligen Füßen, doch für den Glauben sieht es noch schlimmer aus, weil der Glaube eher mit dem Verweis auf Dinge assoziiert wird, die im Agenten physisch repräsentiert werden und nicht nur potenziell ableitbar sind.

Es hat Versuche gegeben, eine Form der begrenzten Rationalität für Agenten zu definieren – zu sagen, dass Agenten nur die Behauptungen glauben, die sich durch Anwendung von nicht mehr als k Schlussfolgerungsschritten oder nicht mehr als s Sekunden Rechenzeit ableiten lassen. Diese Versuche haben sich im Allgemeinen als unbefriedigend erwiesen.

10.4.1 Andere Modallogiken

Es wurden viele Modallogiken nicht nur für Wissen, sondern auch für andere Modalitäten vorgeschlagen. Ein Vorschlag bestand darin, Modaloperatoren für *Möglichkeit* und *Notwendigkeit* hinzuzufügen: Es ist möglicherweise wahr, dass einer der Autoren dieses Buchs gerade sitzt, und es ist notwendigerweise wahr, dass $2 + 2 = 4$ ist.

Wie in Abschnitt 8.1.2 erwähnt, favorisieren einige Logiker Modalitäten, die sich auf die Zeit beziehen. In der **linearen Temporallogik** fügen wir die folgenden Modaloperatoren hinzu:

- **X** P: „P wird im nächsten (*Ne**X**t*) Zeitschritt wahr sein“
- **F** P: „P wird irgendwann (***F**inally*) in einem zukünftigen Zeitschritt wahr sein“
- **G** P: „P ist immer (***G**lobally*) wahr“
- P **U** Q: „P bleibt wahr, bis (***U**ntil*) Q eintritt“

Manchmal können hiervon weitere Operatoren abgeleitet werden. Das Hinzufügen dieser Modaloperatoren macht die Logik selbst komplexer (mit der Folge, dass es für logische Inferenzalgorithmen schwerer ist, einen Beweis zu finden). Andererseits können wir dank der Operatoren bestimmte Fakten in einer prägnanteren Form angeben (was die logische Inferenz wiederum schneller macht). Die Wahl der anzuwendenden Logik ist vergleichbar mit der Wahl der geeignetsten Programmiersprache: Wählen Sie eine, die für Ihre Aufgabe geeignet

6 Anm. zur Übersetzung: Beachten Sie, dass „Glauben“ hier im Sinne von „überzeugt sein, dass“ benutzt wird.

ist, die Ihnen und den anderen, die mit Ihnen zusammenarbeiten, vertraut ist und die für Ihre Zwecke effizient genug ist.

10.5 Schlussfolgerungssysteme für Kategorien

Kategorien sind die primären Bausteine groß angelegter Wissensrepräsentationsschemas. Dieser Abschnitt beschreibt Systeme, die speziell für das Organisieren und Schlussfolgern mit Kategorien entwickelt wurden. Es gibt zwei eng verwandte Systemfamilien: **Semantische Netze** bieten grafische Hilfsmittel zum Visualisieren einer Wissensbasis und effiziente Algorithmen zum Ableiten von Eigenschaften eines Objekts auf der Basis seiner Kategoriezugehörigkeit. **Beschreibungslogiken** bieten eine formale Sprache zum Konstruieren und Kombinieren von Kategoriendefinitionen sowie effiziente Algorithmen zum Ermitteln von Teilmengen- und Obermengenbeziehungen zwischen Kategorien.

10.5.1 Semantische Netze

Charles S. Peirce schlug 1909 eine grafische Notation von Knoten und Kanten vor, sogenannte **Existenzgraphen**, die er als „die Logik der Zukunft“ bezeichnete. Damit begann eine lang andauernde Debatte zwischen den Verfechtern der „Logik“ und den Verfechtern der „semantischen Netze“. Leider verdeckte die Debatte die Tatsache, dass semantische Netze eine Form der Logik *sind*. Die Notation, die von semantischen Netzen für bestimmte Arten von Sätzen bereitgestellt wird, ist oft bequemer, aber wenn wir die Probleme mit der „menschlichen Schnittstelle“ außer Acht lassen, sind die zugrunde liegenden Konzepte – Objekte, Relationen, Quantifizierung usw. – die gleichen.

Es gibt viele Varianten von semantischen Netzen, aber alle sind in der Lage, einzelne Objekte, Kategorien von Objekten und Relationen zwischen Objekten zu repräsentieren. Eine typische grafische Notation zeigt Objekt- oder Kategorienamen in Ovalen oder Kästen an und verbindet sie mit beschrifteten Kanten. In ► Abbildung 10.4 gibt es z. B. eine *ElementVon*-Verknüpfung zwischen *Mary* und *WeiblichePersonen*, die der logischen Behauptung $Mary \in WeiblichePersonen$ entspricht; analog dazu entspricht die *SchwesterVon*-Verknüpfung zwischen *Mary* und *John* der Behauptung *SchwesterVon*(*Mary*, *John*). Wir können Kategorien auch mit *TeilmengeVon*-Verknüpfungen verbinden, und so weiter. Doch achten Sie bei dem ganzen Spaß, den es macht, Kästchen, Ovale und Pfeile zu zeichnen, auch darauf, es nicht zu übertreiben. Wir wissen zum Beispiel, dass Personen weibliche Personen als Müt-

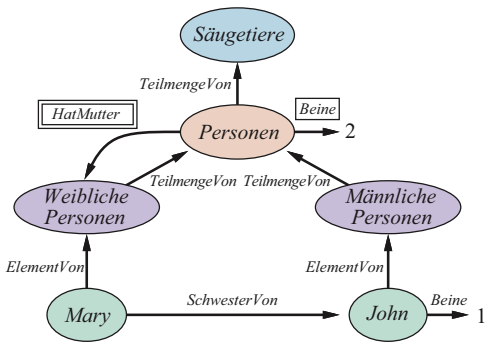

Abbildung 10.4: Ein semantisches Netz mit vier Objekten (John, Mary, 1 und 2) und vier Kategorien. Die Relationen werden durch beschriftete Verknüpfungen dargestellt.

ter haben. Können wir also eine *HatMutter*-Verknüpfung von *Personen* zu *WeiblichePersonen* zeichnen? Die Antwort ist nein, denn *HatMutter* ist eine Beziehung zwischen einer Person und ihrer Mutter, und Kategorien haben keine Mütter.[7]

Aus diesem Grund haben wir in ► Abbildung 10.4 eine spezielle Notation verwendet – die Verknüpfung mit doppelt umrandeter Beschriftung. Diese Verknüpfung behauptet, dass

$$\forall x \;\; x \in Personen \;\Rightarrow\; [\forall y \;\; HatMutter(x, y) \;\Rightarrow\; y \in WeiblichePersonen]$$

gilt. Wir wollen vielleicht auch behaupten, dass Personen zwei Beine haben – das heißt:

$$\forall x \;\; x \in Personen \;\Rightarrow\; Beine(x, 2)$$

Wie zuvor müssen wir darauf achten, dass wir nicht behaupten, dass eine Kategorie Beine hat; die Verknüpfung mit einfach umrandeter Beschriftung in ► Abbildung 10.4 behauptet Eigenschaften jedes Elements einer Kategorie.

Die Notation des semantischen Netzes macht es leichter, Schlussfolgerungen durch **Vererbung** durchzuführen, wie in Abschnitt 10.2 beschrieben. Zum Beispiel erbt Mary dank der Tatsache, dass sie eine Person ist, die Eigenschaft, zwei Beine zu haben. Um also herauszufinden, wie viele Beine Mary hat, folgt der Vererbungsalgorithmus der *ElementVon*-Verknüpfung von *Mary* zu der Kategorie, zu der sie gehört, und von dort aus den *TeilmengeVon*-Verknüpfungen in der Hierarchie, bis er eine Kategorie findet, für die es eine Verknüpfung mit einer einfach umrandeten Beschriftung *Beine* gibt – in diesem Fall die Kategorie *Personen*. Die Einfachheit und Effizienz dieses Inferenzmechanismus im Vergleich zu semientscheidbaren logischen Theorembeweisern war einer der Hauptvorteile von semantischen Netzen.

Die Vererbung wird kompliziert, wenn ein Objekt zu mehr als einer Kategorie gehört oder eine Kategorie eine Teilmenge von mehr als einer anderen Kategorie ist. Dies wird als **Mehrfachvererbung** bezeichnet. In solchen Fällen findet der Vererbungsalgorithmus möglicherweise zwei oder mehr widersprüchliche Werte als Antwort auf die Abfrage. Aus diesem Grund ist Mehrfachvererbung in einigen **objektorientierten Programmiersprachen** (OOP) verboten, z. B. in Java, das Vererbung in seinen Klassenhierarchien verwendet. In semantischen Netzen ist sie normalerweise erlaubt, doch die Diskussion darüber wollen wir uns für Abschnitt 10.6 vorbehalten.

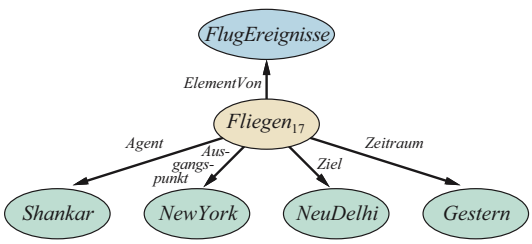

Abbildung 10.5: Ausschnitt eines semantischen Netzes, das die Repräsentation der logischen Behauptung *Fliegen*(*Shankar*, *NewYork*, *NewDelhi*, *Gestern*) zeigt.

7 Mehrere frühe Systeme konnten nicht zwischen den Eigenschaften von Elementen einer Kategorie und den Eigenschaften der Kategorie als Ganzes unterscheiden. Dies kann direkt zu Inkonsistenzen führen, wie Drew McDermott (1976) in seinem Artikel „Artificial Intelligence Meets Natural Stupidity“ aufzeigte. Ein weiteres häufiges Problem war die Verwendung von *IstEin*-Verknüpfungen sowohl für Teilmengen- als auch für Zugehörigkeitsrelationen, entsprechend dem natürlichen Sprachgebrauch: „Eine Katze ist ein Säugetier“ und „Fifi ist eine Katze“ (zur Vertiefung siehe Übung 12.27 auf der Website).

Dem Leser ist vielleicht ein offensichtlicher Nachteil der Notation semantischer Netze im Vergleich zur Prädikatenlogik aufgefallen, nämlich dass die Verknüpfungen zwischen den Ovalen nur *binäre* Relationen darstellen. Zum Beispiel kann der Satz *Fliegen*(*Shankar*, *NewYork*, *NewDelhi*, *Gestern*) in einem semantischen Netz nicht direkt ausgedrückt werden. Dennoch *können* wir den Effekt von *n*-stelligen Behauptungen erzielen, indem wir den eigentlichen Satz als Ereignis reifizieren, das zu einer entsprechenden Ereigniskategorie gehört. ► Abbildung 10.5 zeigt die semantische Netzstruktur für dieses spezielle Ereignis. Beachten Sie, dass die Beschränkung auf binäre Relationen die Erstellung einer umfangreichen Ontologie reifizierter Konzepte erzwingt.

Die Reifikation von Aussagen macht es möglich, jeden grundierten, funktionsfreien atomaren Satz der Prädikatenlogik in der Notation semantischer Netze darzustellen. Bestimmte Arten von allquantifizierten Sätzen können mithilfe von inversen Verknüpfungen und Anwendung der einfach und doppelt gerahmten Kantenbeschriftungen auf Kategorien ausgedrückt werden, aber dennoch sind wir weit von einer vollständigen Prädikatenlogik entfernt. Es fehlen Negation, Disjunktion, verschachtelte Funktionssymbole und Existenzquantifizierung. Es ist jedoch *möglich*, die Notation zu erweitern, sodass sie äquivalent zur Prädikatenlogik ist – wie in den Existenzgraphen von Peirce –, aber dadurch wird einer der Hauptvorteile semantischer Netze negiert, nämlich die Einfachheit und Transparenz der Inferenzprozesse. Die Entwickler können ein großes Netz aufbauen und trotzdem eine gute Vorstellung davon haben, welche Abfragen effizient sind, weil es (a) einfach ist, die Schritte der Inferenzprozedur zu visualisieren, und (b) in einigen Fällen die Abfragesprache so einfach ist, dass schwierige Abfragen nicht gestellt werden können.

In Fällen, in denen sich die Ausdrucksstärke als zu einschränkend erweist, sehen viele semantische Netzsysteme ein **prozedurales Beiwerk** vor, um die Lücken zu füllen. Das prozedurale Beiwerk ist eine Technik, bei der eine Abfrage über eine bestimmte Relation (oder manchmal eine Aussage zu derselbigen) zu einem Aufruf einer speziellen Prozedur führt, die für diese Relation entwickelt wurde, und nicht zur Ausführung eines allgemeinen Inferenzalgorithmus.

Einer der wichtigsten Aspekte von semantischen Netzen ist ihre Fähigkeit, **Default-Werte** (Vorbelegungen) für Kategorien darzustellen. Wenn wir uns ► Abbildung 10.4 genau anschauen, stellen wir fest, dass John ein Bein hat, obwohl er eine Person ist und alle Personen zwei Beine haben. In einer streng logischen Wissensbasis wäre dies ein Widerspruch, aber in einem semantischen Netz hat die Behauptung, dass alle Personen zwei Beine haben, nur den Status einer Voreinstellung, d. h., es wird angenommen, dass eine Person zwei Beine hat, solange dies nicht durch spezifischere Informationen widerlegt wird. Die Default-Semantik wird auf natürliche Weise durch den Vererbungsalgorithmus erzwungen, da dieser den Verknüpfungen vom Objekt selbst (in diesem Fall John) nach oben folgt und abbricht, sobald er einen Wert gefunden hat. Wir sagen, dass der Default-Wert durch den spezifischeren Wert **überschrieben** wird. Beachten Sie, dass wir die Standardanzahl von Beinen auch außer Kraft setzen könnten, indem wir eine Kategorie *EinbeinigePersonen* einführen, die eine Teilmenge von *Personen* ist und die *John* als Element hat.

Wir können eine streng logische Semantik für das Netzwerk beibehalten, wenn wir sagen, dass die Aussage *Beine* für Personen eine Ausnahme für John enthält:

$$\forall x \;\; x \in Personen \wedge x \neq John \;\Rightarrow\; Beine(x, 2)$$

Für ein *festes* Netz ist dies semantisch adäquat, aber für den Fall, dass es viele Ausnahmen gibt, bei Weitem nicht so prägnant wie die Netzwerknotation selbst. Bei Netzen, die mit weiteren Behauptungen aktualisiert werden, schlägt dieser Ansatz jedoch fehl – wir wollen eigentlich sagen, dass alle noch unbekannten Personen mit einem Bein ebenfalls Ausnahmen sind. Abschnitt 10.6 beschäftigt sich näher mit diesem Thema und dem Schlussfolgern mit Default-Informationen im Allgemeinen.

10.5.2 Beschreibungslogkien

Die Syntax der Prädikatenlogik wurde entwickelt, um Aussagen über Objekte zu vereinfachen. **Beschreibungslogiken** sind Notationen, die es einfacher machen sollen, Definitionen und Eigenschaften von Kategorien zu beschreiben. Systeme der Beschreibungslogik entwickelten sich aus semantischen Netzen. Sie waren eine Antwort auf den Druck, zu formalisieren, was die Netze bedeuten, und zwar möglichst unter Beibehaltung der taxonomischen Struktur als Organisationsprinzip.

Die wichtigsten Inferenzaufgaben für Beschreibungslogiken sind **Subsumption** (Überprüfen, ob eine Kategorie eine Teilmenge einer anderen ist, durch Vergleich ihrer Definitionen) und **Klassifikation** (Überprüfen, ob ein Objekt zu einer Kategorie gehört). Einige Systeme beinhalten auch die **Konsistenz** einer Kategoriendefinition – ob die Zugehörigkeitskriterien logisch erfüllbar sind.

Die Sprache CLASSIC (Borgida *et al.*, 1989) ist eine typische Beschreibungslogik. Die Syntax von CLASSIC-Beschreibungen ist in ► Abbildung 10.6 dargestellt.[8] Um beispielsweise zu sagen, dass Junggesellen unverheiratete männliche Erwachsene sind, würden wir schreiben:

$$Junggeselle = And(Unverheiratet, Erwachsen, Männlich)$$

Das Äquivalent in der Prädikatenlogik würde wie folgt aussehen:

$$Junggeselle(x) \Leftrightarrow Unverheiratet(x) \wedge Erwachsen(x) \wedge Männlich(x)$$

Beachten Sie, dass zu einer Beschreibungslogik eine Algebra von Operationen auf Prädikaten gehört, was in der Prädikatenlogik natürlich nicht möglich ist. Jede Beschreibung in CLASSIC kann in einen äquivalenten prädikatenlogischen Satz übersetzt werden, aber einige Beschreibungen sind in CLASSIC einfacher zu handhaben. Um beispielsweise die Menge der Männer mit mindestens drei Söhnen zu beschreiben, die alle arbeitslos und mit Ärztinnen verheiratet sind und höchstens zwei Töchter haben, die alle eine Professur in Physik oder Mathematik

8 Beachten Sie, dass es die Sprache *nicht* erlaubt, einfach zu sagen, dass ein Begriff oder eine Kategorie eine Teilmenge einer anderen ist. Dies ist eine bewusste Vorgehensweise: Die Subsumption zwischen Kategorien muss aus einigen Aspekten der Kategoriebeschreibungen ableitbar sein. Wenn nicht, dann fehlt etwas in den Beschreibungen.

$$
\begin{aligned}
Konzept &\rightarrow \textbf{Thing} \mid KonzeptName \\
&\mid \textbf{And}(Konzept, \ldots) \\
&\mid \textbf{All}(RollenName, Konzept) \\
&\mid \textbf{AtLeast}(Integer, RollenName) \\
&\mid \textbf{AtMost}(Integer, RollenName) \\
&\mid \textbf{Fills}(RollenName, IndividualName, \ldots) \\
&\mid \textbf{SameAs}(Pfad, Pfad) \\
&\mid \textbf{OneOf}(IndividualName, \ldots) \\
Pfad &\rightarrow [RollenName, \ldots] \\
KonzeptName &\rightarrow Erwachsen \mid Weiblich \mid Männlich \mid \ldots \\
RollenName &\rightarrow Ehepartner \mid Tochter \mid Sohn \mid \ldots
\end{aligned}
$$

Abbildung 10.6: Die Syntax von Beschreibungen in einer Teilmenge der Sprache CLASSIC.

vorweisen können, würden wir schreiben:

$$And(Mann, AtLeast(3, Sohn), AtMost(2, Tochter), All(Sohn, And(Arbeitslos, Verheiratet, All(Ehepartner, Arzt))), All(Tochter, And(Professor, Fills(Lehrstuhl, Physik, Math))))$$

Wir überlassen es dem Leser als Übung, dies in Prädikatenlogik zu übersetzen.

Der vielleicht wichtigste Aspekt von Beschreibungslogiken ist, wie stark sie die effiziente Machbarkeit der Inferenz in den Mittelpunkt stellen. Eine Probleminstanz wird gelöst, indem sie zuerst beschrieben wird. Dann wird gefragt, ob sie unter eine von mehreren möglichen Lösungskategorien fällt. In Standardsystemen der Prädikatenlogik ist es oft nicht möglich, die Lösungszeit vorherzusagen. Es bleibt häufig dem Benutzer überlassen, die Repräsentation so zu entwerfen, dass sie Sätze umgeht, die das System anscheinend wochenlang mit der Lösung des Problems beschäftigen. In der Beschreibungslogik geht es dagegen darum, dass die Subsumptionsprüfung in einer Zeit gelöst werden kann, die polynomiell zur Größe der Beschreibungen ist.[9]

Das klingt im Prinzip wunderbar, bis man merkt, dass das nur eine von zwei Konsequenzen haben kann: Entweder können schwere Probleme überhaupt nicht ausgedrückt werden oder sie erfordern exponentiell große Beschreibungen! Die Ergebnisse bezüglich der effizienten Machbarkeit geben jedoch Aufschluss darüber, welche Arten von Konstrukten Probleme verursachen, und helfen so dem Benutzer zu verstehen, wie sich verschiedene Repräsentationen verhalten. Zum Beispiel fehlen in Beschreibungslogiken normalerweise *Negation* und *Disjunktion*. Beide zwingen prädikatenlogische Systeme dazu, eine potenziell exponentielle Fallanalyse zu durchlaufen, um Vollständigkeit zu gewährleisten. CLASSIC erlaubt nur eine begrenzte Form der Disjunktion in den *Fills*- und *OneOf*-Konstrukten, die eine Disjunktion über explizit aufgelistete Individuen, aber nicht über Beschreibungen erlauben. Bei disjunkten Beschreibungen können verschachtelte Definitionen leicht zu einer exponentiellen Anzahl alternativer Routen führen, durch die eine Kategorie eine andere subsumieren kann.

10.6 Schlussfolgern mit Default-Informationen

Im vorherigen Abschnitt haben wir ein einfaches Beispiel für eine Behauptung mit Default-Status gesehen: Menschen haben zwei Beine. Dieser Default-Wert kann durch spezifischere Informationen überschrieben werden, z. B. dass Long John Silver ein Bein hat. Wir haben gesehen, dass der Vererbungsmechanismus in semantischen Netzen das Überschreiben von Default-Werten auf einfache und natürliche Weise implementiert. In diesem Abschnitt betrachten wir Default-Werte ganz allgemein, wobei unser Schwerpunkt darauf liegt, ihre *Semantik* zu verstehen und nicht nur einen prozeduralen Mechanismus zu beschreiben.

10.6.1 Zirkumskription und Default-Logik

Wir haben zwei Beispiele für Schlussfolgerungsprozesse vorgestellt, die die **Monotonieeigenschaft** der Logik verletzen, die in Kapitel 7 bewiesen wurde.[10] In diesem Kapitel haben wir gesehen, dass eine Eigenschaft, die von allen Elementen einer Kategorie in einem semantischen Netz geerbt wird, durch spezifischere Informationen für eine Unterkategorie überschrieben werden kann. In Abschnitt 9.4.4 haben wir gesehen, dass unter der Closed-World Assumption Folgendes gilt: wenn eine Aussage α nicht in der Wissensbasis KB erwähnt wird, dann $KB \models \neg\alpha$, aber $KB \wedge \alpha \models \alpha$.

9 CLASSIC bietet in der Praxis effiziente Subsumptionstests, aber die Worst-Case-Laufzeit ist exponentiell.

10 Wir wissen, dass gemäß der Monotonie alle Sätze, die mittels semantischer Folgerung hergeleitet wurden, korrekt bleiben müssen, nachdem neue Sätze zur Wissensbasis hinzugefügt wurden. Das heißt, wenn $KB \models \alpha$ dann $KB \wedge \beta \models \alpha$.

Copyright