Welche Wissensrepräsentationssysteme kennen Sie?
- Datenmodelle traditioneller Datenbanken
- objektorientiert (semantische Netze, Framerepräsentationen → zusammen Vererbungsnetze)
- regelorientiert (logikorientiert, grammatikalisch, Produktionsregelsysteme)
- lexikalisch (Computerlexika, Thesauri)

Die **semantischen Netze** sind sehr stark von den assoziativen der kognitiven Psychologie beeinflusst. Sie sind von besonderer Bedeutung für die Repräsentation natürlichsprachig gegebenen Wissens und werden u.a. in Frage-Antwort-Systemen eingesetzt.
Die **Framerepräsentationen** beschreiben Objekte innerhalb einer Hierarchie von Wissenselementen als Schemata, die durch Gruppen von Merkmals-/Wert-Paaren charakterisiert werden. Der Frame-Begriff hat Eingang in mehrere Wissensrepräsentationssprachen der KI und darauf aufbauende Systeme gefunden.

Bei den **regelorientierten WRS** spielen neben den rein **logischen Darstellungen (Abschn. 3.2)** vor allem die **Produktionsregelsysteme** eine wichtige Rolle. Sie sind insbesondere für die Wissensrepräsentation in Expertensystemen von Bedeutung **(s. Abschn. 7)**.
Grammatikalische WRS und **Computerlexika** dienen vorwiegend zur Beschreibung der strukturellen Eigenschaften der natürlichen Sprache **(s. Abschn. 8)**. Letztere markieren die Nahtstelle zwischen syntaktischen und semantischen Aspekten der automatischen Sprachverarbeitung **(s. Abschn. 9)**.

Was ist der Unterschied zwischen objekt- und regelorientiert?
Gruppierung der Informationen um Objekte, wodurch eine **objektorientierte Wissensrepräsentation** entsteht (im Gegensatz zur Logik, wo die Informationen über ein Objekt auf viele Aussagen verteilt sein können)

Welche Methoden der Wissensverarbeitung kennen Sie, die zur automatischen Verarbeitung geeignet sind?
Welche Untergruppen gibt es bei den objektorientierten WRS? Und bei den regelbasierten WRS?
Was ist ein Produktionsregelsystem, wie sehen die Regeln aus? Worauf basieren die Regeln? (Aussagenlogik bzw. Prädikatenlogik)
Gibt es KI-Programmiersprachen, die sich dafür eignen? (PROLOG)

Was ist ein Frame, Slots, Filler? Wie hängen Frames zusammen?
Wie werden Frames definiert? Wie repräsentieren sie Wissen?
(Hierarchie von Objekten, Vererbung, ...)
Ein Frame ist ein Schema zur Wissensrepräsentation, das eine bestimmte Entität (ein Objekt, einen Sachverhalt, ein Ereignis) oder eine Klasse von Entitäten innerhalb einer Hierarchie solcher Frames mit Hilfe von Merkmals-Wert-Paaren beschreibt. Es ist ein Beschreibungsmuster, das in stereotyper Weise in den verschiedensten Situationen wiederkehrt. Die Merkmale, auch Attribute oder **Slots** genannt, werden als offene Stellen mit Variablencharakter betrachtet, die je nach zu beschreibender Entität durch spezifische Merkmalswerte, die so genannten **Filler**, zu belegen sind.
- Frame Auto, Slott Motor, Filler Otto-Motor
- Frame Golf (erbt von Auto), Slott Leistung, Filler 44KW
- Frame Peters Auto (Golf), Slott Baujahr, Filler 1984

Die formale Beschreibung von Frames in KRL
KRL ist sowohl eine Sprache zur Beschreibung von Frames als auch ein System zur Wissensverwaltung. Es widerspiegelt die wesentlichen Eigenschaften von Frame-orientierten WRS, weshalb es hier in den Mittelpunkt der Betrachtungen gestellt wird. Es wird eine leicht abgewandelte Backus-Naur-Form als Metasprache verwendet. Grundelement und wichtigster Datentyp in KRL ist der „Unit", der zur Repräsentation von Frames dient. Jeder Unit wird durch einen eindeutigen Namen, das Kennzeichen UNIT, eine Typangabe sowie eine Menge von Slotts charakterisiert:

⟨Unit⟩ ::= [⟨Unit-Name⟩ UNIT ⟨Typ⟩ ⟨Slot⟩*]
⟨Typ⟩ ::= BASIC | SPEC | INDIV | ABSTR | REL | MANIF | PROPOS
⟨Slot⟩ ::= ⟨⟨Slot-Name⟩ ⟨Deskription⟩⟩ | ⟨SELF⟩ | ⟨⟨Slot-Name⟩ ⟨Deskription⟩ ; ⟨Feature⟩⟩| ⟨⟨Slot-Name⟩ ⟨Deskription⟩ ; ⟨Prozedur⟩⟩

Konkrete Frame-Hierarchie aufmalen; Instanzen bilden, (Default-) Attribute definieren
Eine konkrete Frame-Hierarchie aufmalen. Vererbung und Defaults erklären.
Was ist Frame-Repräsentation?
Was ist ein Frame und wie ist er aufgebaut?
Wie hängen die einzelnen Frames zusammen?
Slot und Filler erklären. Was sind Default-Annahmen?
Welche zwei verschiedenen Arten von Vererbung kennen Sie und was ist der Unterschied?

a)

AUTO:	ist ein FAHRZEUG
TEILE	- Motor, Chassis, Räder, Karosserie
FORTBEWEGUNGSART	- Rollen
BEWEGUNGSMEDIUM	- Land
ZWECK	- Transport von Gütern oder Personen
MOTORTYP	- Dieselmotor oder Viertakt-Otto-Motor
HERSTELLUNGSORT	- ⟨ eine Stadt ⟩
BAUJAHR	- ⟨ eine Jahreszahl ⟩
LEISTUNG	- ⟨ Maßangabe in Einheiten von Kilowatt ⟩

b)

GOLF:	ist ein AUTO mit
ZWECK	- Transport von Personen
MOTORTYP	- Viertakt-Otto-Motor; DEFAULT
HERSTELLUNGSORT	- Wolfsburg
LEISTUNG	- 44kW - 128kW
ZUBEHÖR	- Airbag; DEFAULT

c)

PETERS AUTO:	ist ein GOLF mit
BAUJAHR	- 1986
FARBE	- blau
NUMMER	- HA-KJ 34
ZUBEHÖR	- kein Airbag

Abbildung 3.6: Beschreibung von Objekten als Frames (informell)
a) Beschreibung eines begrifflichen Frames (Oberbegriff)
b) Instanz dieses Frames (zugehöriger Unterbegriff)
c) Individuum als am weitesten spezialisierte Instanz in der Hierarchie von Frames

Prädikatenlogik Da sowohl Prädikate als auch Relationen einem n-Tupel von Elementen Einen Wahrheitswert zuordnen, können beide unter einem verallgemeinerten Prädikatsbegriff zusammengefasst werden. In diesem Sinne sind die oben angeführten Prädikate als einstellige und die Relationen als mehrstellige Prädikate anzusehen. Das ist auch die Ursache dafür, dass eine mit diesen Begriffen aufgebaute Logik Prädikatenlogik genannt wird. Im folgenden wird der Terminus „Prädikat" in dieser allgemeinen Sprechweise verwendet.

Logikorientierte WRS:

Man verwendet für den formalen Aufbau der Logik (Abschn. 3.2.3) die Symbolik:
$\exists x\ P(x)$ mit dem Existenzquantor bzw. $\forall x\ P(x)$ mit dem Universalquantor oder Allquantor $\forall$

Verknüpfung	Inhaltlich-verbale Erklärung	Logische Operation
Negation	"Es ist nicht wahr, daß A gilt."	$\sim A$
Konjunktion	"Sowohl A als auch B sind wahr."	$A \wedge B$
Disjunktion	"A oder B oder beide sind wahr."	$A \vee B$
Implikation	"Wenn A gilt, dann gilt auch B."	$A \rightarrow B$
Äquivalenz	"A gilt genau dann, wenn B gilt."	$A \leftrightarrow B$

Tabelle 3.1: Verknüpfungsoperationen zwischen Aussagen

Die Variable x nennt man im ersten Fall „existenziell quantifiziert" und im zweiten Fall „universell quantifiziert".
Beispiele: „Es gibt grüne Blüten" wird dargestellt durch: $\exists x$ (BLUETE(x) ^ GRUEN(x))
„Alle Tiere sind Lebewesen" wird dargestellt durch: $\forall x$ (TIER(x) $\rightarrow$ LEBEWESEN(x))

Man achte auf die unterschiedlichen Konnektoren ^ bzw. $\rightarrow$ in Verbindung mit dem Existenz- bzw. Universalquantor. Das hängt mit der semantischen Interpretation dieser logischen Operatoren zusammen (s. Abschn. 3.2.4). Nach dem Studium dieses Abschnittes wird deutlich werden, dass bei Verwendung des Junktors $\rightarrow$ anstelle von ^ im ersten Beispiel der angegebene Ausdruck auch dann wahr wäre, wenn gar keine Blüten existierten. Der zweite Ausdruck mit dem Junktor ^ anstelle von $\rightarrow$ würde dagegen behaupten, dass alle Elemente des beschriebenen Gegenstandsbereiches Tiere sind. Beides ist nicht Inhalt der jeweiligen natürlichsprachigen Sätze.

Beispiel (Aktivsatz): „Jeder Schüler liest ein Buch."
Logische Darstellung (bei Betonung auf erstem Wort):
$\forall x(SCHUELER(x) \rightarrow \exists y(BUCH(y) \wedge LIEST(x, y)))$
Beispiel (Passivsatz): „Ein Buch wird von jedem Schüler gelesen."
Logische Darstellung (bei Betonung auf erstem Wort):
$\exists y(BUCH(y) \wedge \forall x(SCHUELER(x) \rightarrow LIEST(x, y)))$

Argumente		Wahrheitswertefunktion f(p,q)					
p	q	non(p)	vel(p,q)	et(p,q)	seq(p,q)	equ(p,q)	...
T	T	F	T	T	T	T	
T	F	F	T	F	F	F	
F	T	T	T	F	T	F	...
F	F	T	F	F	T	T	
Interpretation für:		$\sim A$ Negation	$A \vee B$ Disjunktion	$A \wedge B$ Konjunktion	$A \rightarrow B$ Implikation	$A \leftrightarrow B$ Äquivalenz	...

Tabelle 3.4: Tabelle der wichtigsten Wahrheitswertefunktionen

Implikation. Eine weitere Schwierigkeit bezüglich der Repräsentation natürlichsprachiger Aussagen mit Hilfe der Prädikatenlogik besteht darin, dass die Implikation $P \rightarrow K$ in ihrer extensionalen Deutung als Wahrheitswertefunktion eigentlich nicht dem entspricht, was man in der natürlichen Sprache unter konditionalen Beziehungen versteht (s. Abschn. 3.4.3). So ist es intuitiv unbefriedigend, dass ein semantisch nicht akzeptabler Satz wie: „Wenn Platin ein Gas ist, dann ist die Sonne ein Würfel." bei

Darstellung der „wenn-dann"-Beziehung mit Hilfe der Implikation einen logisch wahren Ausdruck ergibt:
GAS(Platin) → WUERFEL(Sonne).
Das liegt daran, dass die Konditionalbeziehung der natürlichen Sprache nicht einfach eine wahrheitsfunktionale Beziehung ist und im Gegensatz zur logischen Implikation P → K einen Sinnzusammenhang zwischen dem Vorderglied P (der Prämisse) und dem Hinterglied K (der Konklusion) verlangt.

Negation. Eins der Hauptprobleme bei der Negation besteht darin, zwischen Satznegation (d.h. der Verneinung einer ganzen Aussage) und Konstituentennegetion (d.h. der Verneinung eines Teils einer Aussage) zu unterscheiden.
Beispiele: „Peter traf nicht die Zielscheibe." (Konstituentennegation)

$\exists x$ TREFFEN(Peter, x) ^ ~ZIELSCHEIBE(x)

„Peter sieht nichts" (Satznegation)

$\sim\exists x$ SEHEN(Peter, x)

Prädikatenlogik: Darstellen von "Jeder Student besitzt ein Fahrrad."
$\forall x(\text{STUDENT}(x) \rightarrow \exists y(\text{FAHRRAD}(y) \wedge \text{BESITZT}(x, y)))$
Warum haben sie Implikation und nicht Und zwischen Student und Fahrrad genommen?
Warum nicht ^ statt →

Kalkülbegriff allgemein formal definieren
Kalkül K = <W, AU, ABL, S>

- einem Gesamtwortvorrat W, der aus einem Alphabet bzw. Zeichenvorrat Z durch Bildung aller Wörter über dem Alphabet erzeugt wird
- einer Ausdrucksmenge AU mit AU ⊆ W; diese wird durch besondere syntaktische Vorschriften festgelegt
- einer Ableitungsrelation ABL; diese wird durch eine Abbildung AB definiert, die jeder Teilmenge von AU wieder eine solche zuordnet, so dass die Hüllenaxiome und das Endlichkeitsaxiom erfüllt sind (es gelte $X_1;X_2 \subseteq$ AU). Hüllenaxiome:
  - Monotonie: wenn $X_1 \subseteq X_2$, dann gilt auch $AB(X_1) \subseteq AB(X_2)$
  - Einbettung: $X \subseteq AB(X)$
  - Abgeschlossenheit: $AB(AB(X)) \subseteq AB(X)$
  - Endlichkeitsaxiom: Wenn $H \subseteq AB(X)$, dann gibt es ein endliches $X^* \subseteq X$, so dass $H \subseteq AB(X^*)$
- der Satzmenge S ⊂ AU, die aus einer vorgegebenen Ausdrucksmenge $S_0 \subset$ AU mit Hilfe der Ableitungsrelation definiert wird.

Wie widerspiegeln sich die eingeführten allgemeinen Charakteristika eines Kalküls im Formalismus der Aussagenlogik?
- Der Zeichenvorrat Z des Aussagenkalküls setzt sich wie folgt zusammen:
- A, B, C, ... als Aussagenvariable
- ∧, ~ als logische Konnektor
- (,) zur Strukturierung von Ausdrücken.

Es ist zu bemerken, dass die angegebenen Erläuterungen keinerlei Bedeutung für den Aufbau des Kalküls als solchen haben. Sie besitzen nur mnemonischen Wert. Der Gesamtwortvorrat W ist die Menge aller über Z bildbaren Zeichenketten.

- Die Ausdrucksmenge AU wird induktiv bestimmt:
  - alle Aussagenvariablen A, B, C, ... sind Ausdrücke
  - wenn U ein Ausdruck ist, dann ist auch ~U ein Ausdruck
  - wenn U und V Ausdrücke sind, dann ist auch (U ∨ V) einer
  - es gibt keine anderen Ausdrücke

- Ableitungsrelationen durch Ableitungsregeln definiert
- Die Satzmenge S wird so aufgebaut, dass man angibt, welche Ausdrücke
- Axiom aus AU per definitionem Sätze sind. Diese heißen in der Logik Axiome. Ihre Auswahl ist nur semantisch, also außerhalb des Kalküls, begründbar.

Alle übrigen Sätze, die man abgeleitete Sätze oder Theoreme nennt, werden als diejenigen Ausdrücke bestimmt, die entweder aus den Axiomen selbst oder aus anderen, bereits bewiesenen Theoremen ableitbar sind.
Eine wichtige Forderung an logische Kalküle bzw. deren Axiomensysteme ist die nach Vollständigkeit und Widerspruchsfreiheit.

Vorwärtsdeduktion: (Ableitungsstrategien im Aussagekalkül) Eine Methode des Vorgehens bzw. eine Ableitungsstrategie besteht darin, dass man ausgehend von den Axiomen oder bestimmten Annahmen (den Voraussetzungen) versucht, den gewünschten Ausdruck (die zu beweisende Formel) schrittweise abzuleiten. Wegen der Richtung vom Bekannten zum Gesuchten nennt man dieses Verfahren „Vorwärtsdeduktion". Dabei ist jedes Glied der Kette von Formeln entweder ein Axiom bzw. eine Voraussetzung (Beginn der Deduktion) oder es steht mit einem oder mehreren vorhergehenden Gliedern der Beweiskette in der Ableitungsrelation.
Rückwärtsdeduktion. Beim Ableiten einer Formel kann man auch den umgekehrten Weg gehen. Man beginnt mit dem zu beweisenden Ausdruck und versucht, rückwärts andere Ausdrücke zu finden, die mit ersterem in der Ableitungsrelation stehen. Dieses Verfahren wird so lange fortgesetzt, bis man auf Axiome oder bereits anderweitig bewiesene Ausdrücke trifft. In diesem Falle ist das Verfahren, das man „Rückwärtsdeduktion" nennt, beendet. Man braucht dann nur gedanklich alle Schritte in umgekehrter Richtung zu verfolgen und den Beweisgang analog zur Vorwärtsdeduktion zu bestätigen. Dieses Verfahren wird sehr häufig in der Mathematik verwendet.

Prädikatenkalkül allgemein definieren (3.2.3 Seite 115 ff)
Für viele Anwendungen genügt es nicht, elementare Aussagen in einem Ausdruck einfach durch ein Zeichen zu symbolisieren. Man möchte über den Aussagenkalkül hinaus zu Ausdrucksmöglichkeiten kommen, die es gestatten, auch *die innere Struktur der Aussagen zu repräsentieren*. Die dazu notwendige Erweiterung führt zum so genannten Prädikatenkalkül, in dem anstelle der Aussagenvariablen A, B, C usw. *prädikative Ausdrücke auftreten*, wie z.B. in Ausdrücken der Form:
$\exists x$ VOGEL(x) ^ FLUGUNFAEHIG(x)
$\forall x$ BLUME(x) $\rightarrow$ PFLANZE(x).

Die Ableitungsrelation wird wie im Aussagenkalkül durch Regeln definiert. Es gelten zunächst die gleichen Ableitungsregeln wie im Aussagenkalkül, nur mit dem Unterschied, dass *die Rolle der Aussagenvariablen durch prädikatenlogische Formeln übernommen wird.*
Die Satzmenge wird analog zum Aussagenkalkül mit Hilfe der Ableitungsrelation aus Axiomen bestimmt.

Monotonie erklären: Eine Ableitungsoperation Ab besitzt die Eigenschaft der Monotonie, wenn für zwei Ausdrucksmengen A und B mit $A \subseteq B$ gilt: $Ab(A) \subseteq Ab(B)$. Das bedeutet: die Menge der aus einer Ausdrucksmenge ableitbaren Ausdrücke kann bestenfalls wachsen (auf keinen Fall abnehmen), wenn die zugrunde liegende Ausdrucksmenge selbst wächst.

Was ist ein Semantisches Netz? Was wird konkret dargestellt?
Ein semantisches Netz (SN) ist das mathematische Modell einer Menge, von begrifflichen Entitäten und der zwischen diesen bestehenden kognitiven Beziehungen. Es wird in Form eines verallgemeinerten Graphen dargestellt. Als geeignete bildliche Darstellung eines einfachen SN bietet sich ein markierter, orientierter Graph an, der anschaulich aus einer Menge von benannten Punkten (den Knoten) und gerichteten, Kante benannten Verbindungen zwischen diesen Knoten (den Kanten) besteht.

Jeder Knoten des SN repräsentiert einen bestimmten Begriff, dessen intensionale Bedeutung wesentlich durch die am betreffenden Knoten aus- bzw. einlaufenden Kanten ausgedrückt wird. Beim Übersetzen eines natürlichsprachlichen Satzes S in eine semantische Netzdarstellung geht man zweckmäßigerweise so vor, dass man zunächst einen semantischen Repräsentanten sv (einen Knoten) für S festlegt, von dem dann markierte Kanten ausgehen, welche die semantischen Rollen der am Sachverhalt beteiligten Objekte repräsentieren. Hinsichtlich der Markierung der Knoten muss man darauf achten, dass man semantische Repräsentanten von generischen Begriffen (wie „Turm", „Mensch" usw.) i.a. über eine SUB-Beziehung (begriffliche Unterordnung) in das Netz einbinden muss. Bei Individualbegriffen (wie „Eiffelturm", „Peter" usw.) ist das nicht der Fall; diese werden unmittelbar (d.h. ohne Zwischenschalten einer SUB-Kante) als Markierung an die betreffenden Knoten angeschrieben. Ein generischer Begriff G wird nur dann nicht über eine SUB-Kante in das Netz eingefügt, wenn sich alle behaupteten Beziehungen (ausgedrückt durch die von G aus- bzw. einlaufenden Kanten des SN) auf den Begriff G selbst beziehen und ihn damit näher bestimmen.

Individualbegriff (einmalig identifizierbare Gegebenheit - Bauernkrieg, Oktoberrevolution), Eigenname (Heidi, Gerd), Individuum (Delphin Flipper), generischer Begriff (Klassen von Individuen - Delphin), Abstrakta (keine unmittelbare Anschauung - Inhalt, Entorphie)

Den Ausdruck "Peter schlägt sein Pferd mit einer Peitsche." als semantisches Netz darstellen
(Wichtig: Prädikate definieren, Instanzierung von "schlagen", "Pferd" und "Peitsche")
Warum haben Sie das hier gemacht (SUB, SUBA)? (SUB-Relationen erklären)
Was hieße es denn, wenn sie den generischen Begriff Peitsche direkt angebunden hätten?
Woher wissen Sie denn, wie sie diese (Instr) Beziehung einsetzen müssen (Relation…)?

Ein Problem ist allgemein durch das Spannungsfeld zwischen dem Ziel eines Individuums und einer bestimmten Ausgangssituation, die von diesem Ziel abweicht, gegeben.
Ein Problem ist ein Quadrupel P = <Z, A, L, O> aus

- einer Menge Z Problemzuständen
- einem Anfangszustand A $\in$ Z
- einer Menge L $\subseteq$ Z von Zielzuständen
- einer Menge O von Operationen oder Transformationen

Zug, Regel, Operation, Problemzustand, Problemraum, Lösungsweg

Graphen: Ein geeignetes Mittel zur Modellierung von Problemen mit hoher kombinatorischer Vielfalt sind Graphen. Ein Graph ist gegeben durch eine menge von Konten und einer Abbildung aus der Knotenmenge. Konten sind durch Kanten verbunden.
Einen Graphen, der nur gerichtete Kanten enthält, nennen wir gerichtet, andernfalls heiße er ungerichtet. Ein gerichteter Graph wird auch als orientierter Graph bezeichnet.

Graph (KE3): Mathematische Struktur, die bildlich als eine Menge von Punkten und zwischen diesen verlaufenden (gerichteten oder ungerichteten) Strecken dargestellt wird. Formal wird ein Graph als Tripel G = <N;K; Γ > charakterisiert, das aus einer Menge N von Knoten, einer zu N disjunkten Menge K von Kanten und einer Kantenzuordungsfunktion Γ besteht, die jeder Kante aus K ein Paar von Knoten aus N *x* N zuordnet.

Gerichtete, ungerichtete Graphen
Ein Baum ist ein gerichteter Graph, der eine Wurzel besitzt, von der aus Baum jeder andere Knoten des Graphen auf genau einem Wege erreichbar ist.

Ein **markierter Graph** ist ein Graph mit der Knotenmenge X und der Kantenmenge K, bei dem zusätzlich zwei Mengen M_X und M_K – so genannte Markierungsalphabete für die Knoten bzw. Kanten –

und zwei Abbildungen g_X: $X \rightarrow M_X$ bzw. g_K: $K \rightarrow M_K$ gegeben sind. g_X und g_K ordnen jedem Knoten bzw. jeder Kante des Graphen ein Element aus M_X bzw. M_K als Markierung zu.
Das kann z.B. ein Verkehrsnetz sein, bei dem die Knoten des zur Modellierung benutzten Graphen mit Ortsnamen und die Kanten mit den Namen der Straßen zwischen diesen Orten markiert sind.

Vorwärtssuche, Rückwärtssuche, bidirektionale Suche
Breite-Zuerst-Suche, Tiefe-Zuerst-Suche, Backtracking, Schnappschuss
Bewertete Suche – Wegkosten
Heuristische Verfahren
Problemminimierung - Teilziele
UND/ODER-Baum

Was sind antagonistische Probleme?
Es gibt eine Vielzahl von Problemen, bei denen die Schritte zu ihrer Lösung nicht nur von einer Person und deren Entscheidungen, sondern von mehreren beteiligten Seiten mit unterschiedlichen Interessen abhängen. Solche Probleme werden wir als antagonistische Probleme bezeichnen.

Kooperatives/nichtkooperatives Spiel
Nullsummenspiel, Maxmin-Prinzip

Man bezeichnet generell ein Verfahren, das nach einem bestimmten Prinzip Bewertungen von Knoten eines Baumes von unten nach oben (d.h. in Richtung zur Wurzel) weitergibt, als **Back-up-Methode**.
Die Ermittlung des ersten Zuges von A aufgrund eines back-up-Verfahrens, dem das MINIMAX-Prinzip zugrunde liegt, nennt man **MINIMAX-Verfahren**.
Ein Baum-Suchverfahren für Spiele, das im Rahmen des MINIMAX-Verfahrens mit zwei Schwellwerten ALPHA und BETA zur Baumbeschneidung arbeitet, nennt man **ALPHA-BETA-Technik**. Diese auch mit den griechischen Buchstaben α und β bezeichneten Schwellwerte haben folgende Bedeutung:
ALPHA ist die Bewertung, die der maximierende Spieler MAX in einer bestimmten Position mindestens erreichen kann; BETA ist die Bewertung, die der minimierende Spieler MIN in einer bestimmten Position für MAX höchstens zulässt.

Automatische Problemlösung durch Trail and Error funktioniert noch nicht.

Sprache
In der allgemeinen Zeichenlehre, der Semiotik, unterscheidet man drei verschiedene Zusammenhänge, die durch die Begriffe **Syntax, Semantik und Pragmatik** charakterisiert werden. Unter Syntax versteht man die Beziehungen der Zeichen eines Zeichensystems untereinander. Die Semantik charakterisiert die Beziehung zwischen den Zeichen und ihrer Bedeutung und die Pragmatik stellt die Verbindung her zwischen den Zeichen und den Sendern bzw. Empfängern dieser Zeichen (d.h. zu welchem Zweck wurden in einer gegebenen Situation bestimmte Zeichen gesendet und welche Wirkungen lösen sie aus?). Diese allgemeinen Begriffe werden für beliebige Zeichensysteme, nicht nur für die natürliche Sprache, verwendet.

Die äußere Erscheinungsform eines Satzes nennt man seine **Oberflächenstruktur** und die Repräsentationen auf den nachfolgenden Niveaus heißen Tiefstruktur. In Abbildung 8.5 sind die Oberflächenstruktur, die syntaktische **(Tiefen-)Struktur** und die semantische Tiefenstruktur eines natürlichsprachigen Satzes angegeben.

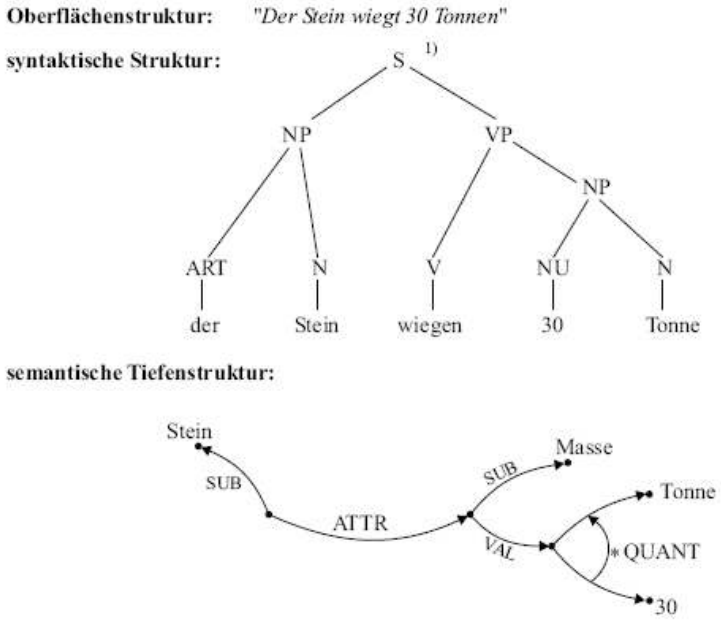


Abbildung 8.5: Satzrepräsentation in verschiedenen Sprachschichten
1) Symbolik: ART Artikel; N Nomen; NP Nominalphrase; NU Zahl; S Satz; V Verb; VP Verbphrase. (Jeder Knoten einer syntaktischen Struktur ist mit morpho-syntaktischen Merkmalen - sog. Grammatemen - verknüpft, die hier nicht angegeben wurden.)

Womit wird die syntaktische Tiefenstruktur dargestellt?
(Kontextfreie Grammatik - semantisches Netz?)

Konstituentenbeziehungen. Indem man sich den Satz als hierarchische Stufung von immer komplexeren Konstituenten aufgebaut denkt, entsteht ein Beziehungsgefüge von unmittelbaren und mittelbaren Konstituenten. So besteht die komplexe Konstituente „das schöne Mädchen" (eine NP) aus den unmittelbaren Konstituenten: Artikel, Adjektiv, Nomen. Diese NP kann selbst unmittelbare Konstituente eines Satzes S („das schöne Mädchen schläft") oder einer Präpositionalphrase PP sein („an das schöne Mädchen"). Dieser Gesichtspunkt wird besonders in den Konstituentengrammatiken und in den darauf aufbauenden Formalismen in den Vordergrund gestellt

Kongruenz. Dabei soll ausdrücklich auch die Übereinstimmung der von einer sprachlichen Einheit (z.B. einem Verb über seinen Kasusrahmen) ausgehenden semantischen Anforderungen mit den entsprechenden Merkmalen einer anderen Konstituente (z.B. eines Nomens) in den Kongruenzbegriff einbezogen werden. Wichtige Kongruenzbeziehungen im Deutschen, die eine Übereinstimmung in den paradigmatischen Kategorien beinhalten, sind:

- die Übereinstimmung von Artikel, (attributivem) Adjektiv und Substantiv in Genus, Kasus und Numerus
- die Übereinstimmung von Subjekt und finitem Verb in Person und Numerus
- die Übereinstimmung eines Pronomens mit dem Bezugsnomen bzw. einem anderem Pronomen (dem so genannten Antezedenten) in Genus und Numerus bzw. Person

Grammatik formal definieren
Unter einer Grammatik versteht man ein Regelwerk, das es gestattet, nach bestimmten Vorschriften die Sätze einer Sprache von den nicht zu dieser Sprache gehörigen Sätzen zu unterscheiden, und jeder Oberflächenstruktur eines akzeptierten Satzes eine entsprechende Tiefenstruktur zuzuordnen (analytische

Grammatik). Es gibt auch Grammatiken, die es erlauben, aus vorgegebenen Tiefenstrukturen bzw. aus einem Startsymbol entsprechende Oberflächenstrukturen abzuleiten (generative Grammatiken).
Im Idealfall realisiert eine Grammatik Γ eine Abbildung Γ: OF $\rightarrow$ SE aus der Menge OF der Oberflächenstrukturen in die Menge SE der semantischen Tiefenstrukturen, ergänzt um ein Element NIL für ungrammatische Sätze.

Unter einer **Sprache** L versteht man eine (endliche oder unendliche Menge) von Symbolketten, die über einem vorgegebenen endlichen Vokabular V gebildet sind. Bezeichnet man mit V* die Menge aller Symbolketten, die sich über V bilden lassen, so kann man diese Definition kurz durch $L \subseteq V^*$ ausdrücken.
Die Grammatiken dienen dazu, mit Hilfe eines endlichen Regelwerks die i.a. unendliche Menge L (nur dieser Fall ist interessant) zu beschreiben.

Eine **Grammatik G zur Beschreibung einer formalen Sprache** ist ein Quadrupel $G = <V_N;V_T;R; S>$, dabei ist

- V_N eine Menge nichtterminaler Symbole, V_T eine Menge terminaler Symbole mit $V_N \cap V_T = \varnothing$; $V_N \cup V_T = V$
- R eine Menge von geordneten Paaren (Regeln)
- $S \in V_N$ ist ein ausgezeichnetes Symbol (das Startsymbol).

Die Anwendung der Regeln einer Grammatik geschieht in folgender Weise:

Was für Grammatiken gibt es?
Uneingeschränkte, kontextabhängige, kontextfreie und reguläre Grammatik

Die **Konstituentengrammatiken** gehen von dem Gedanken aus, dass sich ein Satz aus mehreren Komponenten (auch „unmittelbare Konstituenten", „immediate constituents" oder ICs genannt) zusammensetzt, die wiederum aus anderen Komponenten bestehen; das setzt sich fort bis hinunter zur Wortebene. Danach lässt sich die syntaktische Struktur eines Satzes als Hierarchie von Konstituenten zunehmender Komplexität darstellen. Der formale Apparat der Konstituentengrammatiken basiert auf der Theorie der formalen Sprachen von Chomsky, wobei den kontextfreien Grammatiken eine zentrale Rolle zukommt.

Reguläre Grammatiken. Wir wollen uns zunächst der Frage zuwenden, inwieweit die natürliche Sprache durch eine reguläre Grammatik spezifiziert werden kann. Zu diesem Zweck entwerfen wir eine kleine Grammatik mit den terminalen Symbolen V_T = {art, adj, nom, verb} und den nichtterminalen Symbolen V_N = S, S1, S2, S3, S4. Als Regeln werden zugelassen:

S $\rightarrow$ art S1	S2 $\rightarrow$ verb S4
S $\rightarrow$ adj S1	S3 $\rightarrow$ art S4
S1 $\rightarrow$ nom S2	S4 $\rightarrow$ adj S4
S1 $\rightarrow$ adj S1	S4 $\rightarrow$ nom
S2 $\rightarrow$ verb S3	S2 $\rightarrow$ verb

Lexikon: art der, die, das, ...
adj große, grünes, kleine, ...
nom Fleisch, Geige, Gras, Ziege, ...
verb frisst, liegt, schläft, ...

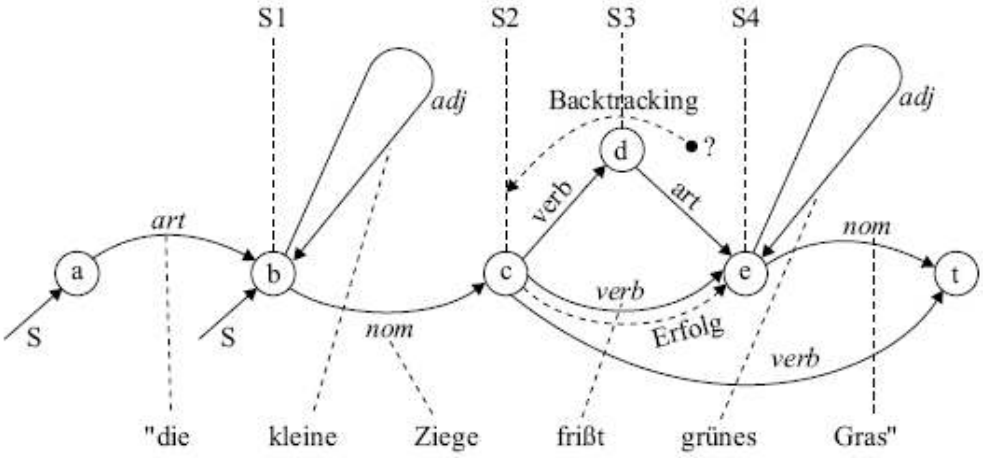
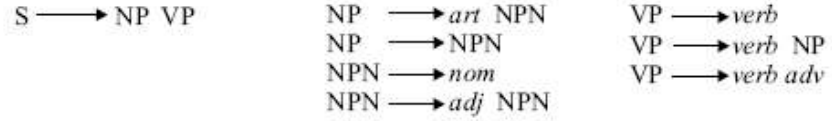


Abbildung 8.9: Einfaches Übergangsnetzwerk (indeterministisch)

Kontextfreie Grammatik aufschreiben, die einen gegebenen Satz erzeugt bzw. akzeptiert
Mit Hilfe einer kontextfreien Grammatik ist es möglich, den Fortgang der Analyse zu jedem Zeitpunkt nicht nur von einem einzelnen Wort, sondern von einer ganzen Wortgruppe abhängig zu machen (den Terminus „Wort" verwenden wir hier für terminale Symbole). In Abbildung 8.10 ist das Beispiel einer kontextfreien Grammatik angegeben, die ebenfalls an der natürlichen Sprache orientiert ist. Die erste Regel besagt z.B., dass sich ein Satz (Symbol: S) aus einer Nominalphrase NP (Substantivgruppe) und einer Verbphrase VP zusammensetzt. Die letztere besteht nach den angegebenen Regeln wiederum aus Verb und Nominalgruppe oder aus Verb und Adverb oder aus einem Verb allein.

a) **kontextfreie Grammatik**



b) **Strukturbaum**
(Konstituentenbaum):

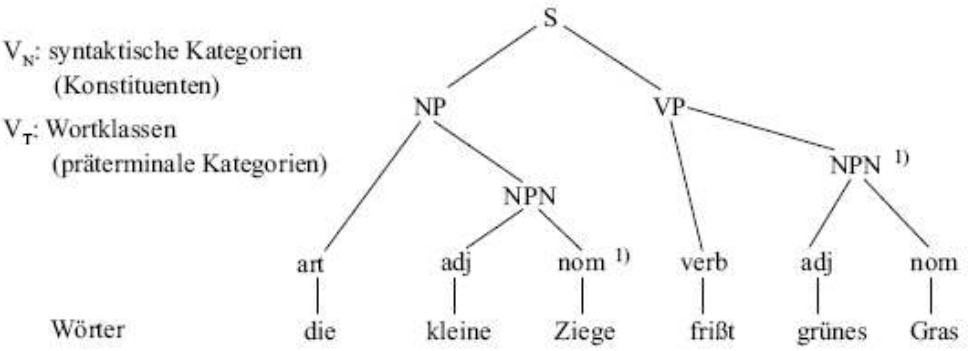


Abbildung 8.10: Kontextfreie Grammatik und Konstituentenbaum
a) Grammatik; b) syntaktische Struktur
1) Zwischenknoten, die durch die formale Ersetzung eines nichtterminalen Symbols durch ein anderes Symbol entstehen, wurden weggelassen.

Kann man mit kontextfreien Grammatiken die natürliche Sprache beschreiben?
Welche Probleme können nicht bewältigt werden?
Kongurrenz, diskontinuierliche Konstituenten, Ellipsen etc.

Welche Grammatiken eignen sich eher für die Beschreibung der natürlichen Sprache?

Die Bezeichnung **ATN-Grammatik** ist von dem englischen Terminus „Augmented transition network" abgeleitet, was soviel bedeutet wie erweitertes Übergangsnetzwerk. Dieser Grammatiktyp ist vor allem durch die Arbeit von Woods bekannt geworden. Zum besseren Verständnis der Zusammenhänge gehen wir von einem einfachen rekursiven Netzwerk (abgekürzt: RTN) aus, das aus mehreren Teilnetzen besteht. *Die Basis für ein RTN bildet eine kontextfreie Grammatik.* Jedes einzelne Teilnetz eines RTN unterscheidet sich von den im Zusammenhang mit den regulären Grammatiken eingeführten Übergangsnetzwerken zunächst einmal dadurch, dass als Kantenmarkierungen nicht nur einfache präterminale Symbole (art, adj, nom usw.) zugelassen sind, sondern auch nichtterminale Symbole, die komplexere syntaktische Konstituenten repräsentieren (wie z.B. NP für Substantivgruppe, PP für Präpositionalphrase, ADV für Adverbialgruppe usw.). Zusätzlich zu den bisher eingeführten Markierungen werden die Kanten noch mit einem Kantentyp: CAT, PUSH, POP oder JUMP gekennzeichnet.
Die ATN-Grammatiken gehen von einer kontextfreien Grammatik aus, die als rekursives Übergangsnetzwerk beschrieben wird und reichern diese durch Tests und Aktionen auf den Kanten des Netzes an.
Zur Überwindung dieser Nachteile entstanden im Grenzbereich zwischen KI und CL die ATN-Grammatiken, die kontextabhängige Elemente und transformationelle Zusammenhänge in der natürlichen Sprache durch Aufnahme von prozeduralen Anteilen (Tests und Registeroperationen) in die Grammatik darzustellen versuchen. Diese Grammatiken repräsentierten in den siebziger Jahren das dominierende Paradigma der Sprachbeschreibung in der KI. Sie sind in angewandten Systemen auch heute noch anzutreffen.

Die **lexikalisch funktionalen Grammatiken** benutzen Termgleichungen zur Attributierung der terminalen und nichtterminalen Symbole in kontextfreien Grammatiken, um die Kontextabhängigkeiten zwischen den Konstituenten eines Satzes auszudrücken. Die Analyse, der eine solche Grammatik zugrunde liegt, muss neben der Struktur des zu analysierenden Satzes auch die Lösung für die bei den Konstituenten angeschriebenen Gleichungssystems liefern. Da der wesentliche Mechanismus zur Lösung dieser Termgleichungen ein analoger Unifikationsmechanismus wie beim Theorembeweisen ist (s. Kap. 5.1.3), wird diese Art von Grammatiken auch Unifikationsgrammatik genannt.

LFG-Grammatik Eine lexikalisch-funktionale Grammatik ist ein Formalismus, der die grammatische Beschreibung eines Satzes aus zwei Komponenten aufbaut: einem Konstituentenbaum (C-Struktur) und einer so genannten Feature-Struktur (abgekürzt: F-Struktur), die beide miteinander in Beziehung stehen.
<F-Struktur> ::= [<aw-Element>*] → <aw-Element> ::= <Attribut> <Wert>

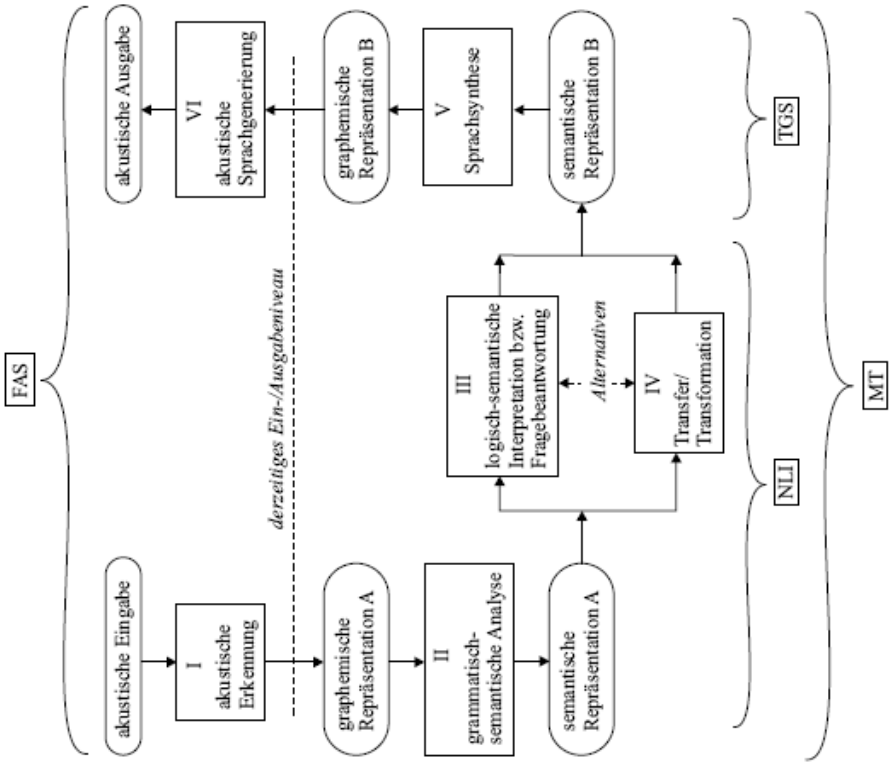


Abbildung 9.1: Komponenten von Systemen zur automatischen Sprachverarbeitung
FAS Frage-Antwort-System (Komponenten: I, II, III, V, VI); NLI natürlichsprachliches Interface (Komponenten: I, II, IV); MT maschinelle Übersetzung (Komponenten: I, II, IV, V, VI); TGS Textgenerierungssysteme (Komponenten: V und VI)

Gemeinsamkeiten der Systeme zur automatischen Sprachverarbeitung

Die Methoden und formalen Mittel der Computerlinguistik finden in den verschiedensten Anwendungssystemen ihren Niederschlag. Die wichtigsten dieser Systeme wurden bereits im Abschnitt 8.1 im Zusammenhang mit den einzelnen Ebenen der ASV angeführt. Wir wollen uns zunächst den gemeinsamen Grundzügen dieser Sprachverarbeitungssysteme zuwenden. Ein natürlichsprachliches Interface (NLI) zu einer Datenbank Interface, oder einem anderen Zielsystem enthält i.a. nur die Komponenten II und IV (im Idealfall, der gerade in den letzten Jahren erreicht worden ist, eventuell noch Komponente I). Die natürlichsprachlichen Interfaces gewährleisten auch für rechentechnisch nicht vorgebildete Nutzer einen Zugang zu bereits bestehenden und kommerziell genutzten Datenbanksystemen; sie bilden eine der Brücken zwischen KI und traditioneller Datenbanksoftware. Frage-Antwort- Frage-Antwort-Systeme (FAS) stellen das bisher vollständigste Paradigma der Simulation von Sprachverstehen auf dem Rechner dar. Sie umfassen alle Prozesse der Sprachverarbeitung (außer der als Alternative zu III anzusehenden Komponente IV). Sie wirken in der Phase der Wissenseingabe ähnlich wie ein NLI und bilden natürlichsprachig formulierte Ausdrücke in die semantische Repräsentationssprache der Wissensbasis (z.B. in ein semantisches Netz) ab. In der Phase der Auskunftserteilung, in der die Fragen zunächst ebenfalls wie in einem NLI behandelt werden, sind dann im Anschluss an die logische Antwortfindung (Komponente III) auch Prozesse der Sprachgenerierung bei der Antwortformulierung einbezogen. Den Aufbau eines FAS werden wir im Abschnitt 9.3 noch ausführlicher diskutieren. Die Frage-Antwort-Systeme bilden einen wichtigen Markstein beim Übergang von der Datenverwaltung zur Wissensverarbeitung. Dieser Trend macht sich von Seiten der Datenbankentwicklung durch immer reichhaltigere Datenmodelle und weitere Verbesserung der logischen Fähigkeiten der Systeme und von Seiten der KI durch stärkere Einbeziehung der bereits sehr weit ausgereiften Datenbanktechniken und Schaffung von Schnittstellen zwischen KI-Software und Datenbanken bemerkbar.

Welche Programmiersprachen sind in der KI wichtig?
LISP (imperative (Vorschrift, wie Problem gelöst wird), funktionale PS)
PROLOG (deklarative (primär die Beschreibung des Problems – Lösungsalgorithmus beim System), logikorientierte PS)
JAVA (deklarative, objektorientierte PS)

Was sind die Prinzipien der beiden?
LISP: Atome, Listen, S-Ausdruck (Blume . Rose), **(Indikator . Wert),**

S-Ausdrücke repräsentieren sehr allgemein verwendbare Datenstrukturen, besitzen aber dennoch eine einfache Syntax: <S-Ausdruck> ::= ( ) | (<Head> . <Tail>)
<Head> ::= <Atom> | <S-Ausdruck>
<Tail> ::= <Atom> | <S-Ausdruck>
Beispiele: (ROSE . BLUME), (A . (B . (C . D)))

Funktionen (DEFUN KREISFLAECHE (R) (TIMES (TIMES R R) PI)) bewirkt den Aufbau eines LAMBADA-Ausdrucks: (LAMBADA (R) (TIMES (TIMES R R) PI))

CAR (erstes Element): (CAR '(1 2 3 4)) ⇒ 1 oder (CAR (CAR '((A B) C))) ⇒ A
CDR (ohne erstes) (CDR (CDR '(A . (B . C)))) ⇒ C oder (CDR '(3)) ⇒ NIL oder (CDR '(1 2 3 4)) ⇒ (2 3 4)
Zusammengefasst: (CADADDR L) bedeutet dasselbe wie (CAR (CDR (CAR (CDR (CDR L)))))

CONS (Liste aufbauen): (CONS 'A 'B) ⇒ (A . B); (CONS 'X NIL) ⇒ (X); (CONS 1 '(2 3 4)) ⇒ (1 2 3 4)
LIST (CONS zusammengefasst): (LIST 'A 'B 'C) ⇒ (A B C); (LIST (LIST 'A 'B) 'C) ⇒ ((A B) C)
APPEND (anhängen): (APPEND ´(1 2 3) ´(4 5 6) ´( ) ´(7 8)) ⇒ (1 2 3 4 5 6 7 8)
REVERSE (umkehren): (REVERSE ´(D C B A)) ⇒ (A B C D)

ASSOC (das erste Element aus der Assoziationsliste <A-Liste> (s.u.), dessen Head mit dem angegebenen Objekt identisch ist): (ASSOC 3 ´((1 . 1) (2 . 4) (3 . 9) (4 . 16) (5 . 25))) ⇒ (3 . 9)
Assoziationslisten sind Listen, deren Elemente gepunktete Paare darstellen: ((AUTO . FAHRZEUG) (SOFA . SITZMOEBEL) (ZANGE .WERKZEUG))

(LISTP (CONS ´A NIL)) ⇒ T; (ATOM 'HANS) ⇒ T; (ATOM '(A B)) ⇒ NIL; (STRINGP „TEXT") ⇒ T

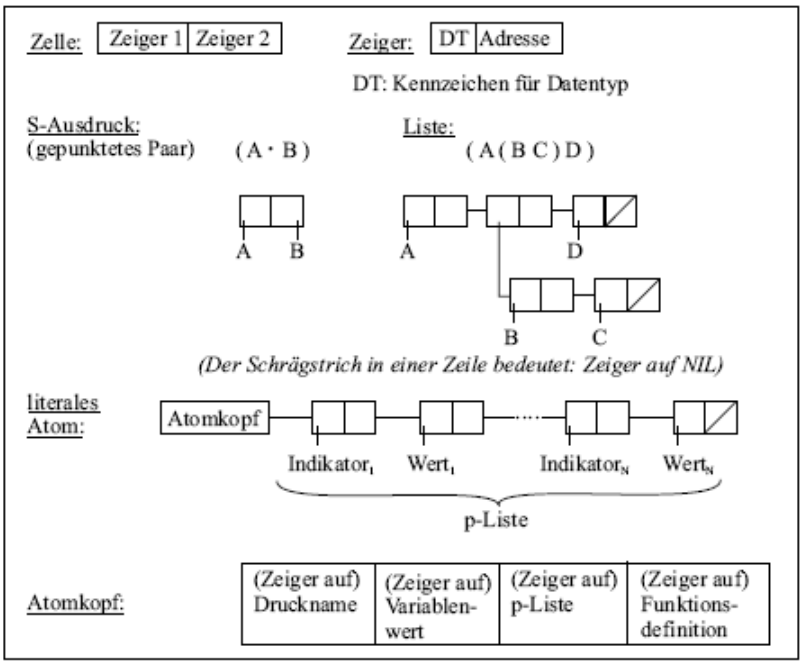
EQ (Vergleich, ob gleicher Zeiger): (EQ 'A (CAR '(A B))) ⇒ T; (EQ '(A B) '(A B)) ⇒ NIL; (EQ 'M 'M) ⇒ T

(EQUAL <Argument> <Argument>) ⇒ T, wenn beide Argumente identisch oder Kopien voneinander sind, andernfalls ist das Ergebnis NIL: (EQUAL '(A B) '(A B)) ⇒ T; (EQUAL T T) ⇒ T

Rekursives und iteratives Programmieren
Rekursive Programmierung. Ein typisches Merkmal rekursiver Programme ist das Auftreten des Namens der definierten Funktion in ihrem eigenen Funktionskörper. Rekursive Programme zeichnen sich in LISP meist durch Kompaktheit der Darstellung und eine gewisse Eleganz aus. Sie sind aber bei mehrfacher oder indirekter Anwendung rekursiver Funktionsaufrufe mitunter schwerer verständlich. Diese Art der Definition ist besonders zur Behandlung von Datenstrukturen geeignet, die selbst einen rekursiven Aufbau besitzen. Das trifft dann zu, wenn die Teile der zu verarbeitenden Struktur den gleichen syntaktischen Aufbau besitzen wie die Gesamtstruktur. Bei rekursiver Programmierung kommt man zwar meist mit weniger Funktionsaufrufen aus, muss aber dafür eine höhere Belastung des Kellerspeichers bei der Abarbeitung in Kauf nehmen. Diese Methode ist oft nicht so günstig anwendbar, wenn in Abhängigkeit von ein und demselben Zwischenergebnis in verschiedene Programmteile verzweigt werden muss (obwohl es dafür in vielen LISP-Systemen auch andere Mechanismen, wie die SELECT- oder CASE-Funktion, gibt).

Iterative Programmierung. Dieser Programmierstil ist den meisten Menschen zunächst vertrauter und die Programme sind mitunter auch leichter verständlich. Dafür ist aber gewöhnlich ein größerer Einsatz von Elementarfunktionen erforderlich sowie ein häufiges explizites Speichern von Zwischenergebnissen. Diese Programmierweise ist günstig einsetzbar, wenn ein und dasselbe Zwischenergebnis in verschiedenen Funktionen weiterverarbeitet werden soll.

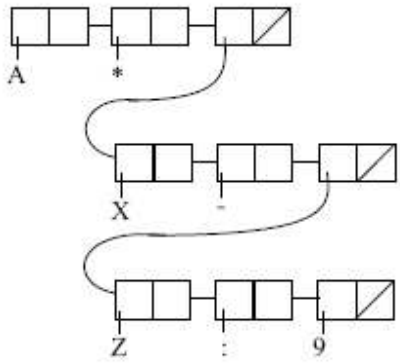
Zellenstruktur in LISP für gegebenen Ausdruck aufschreiben



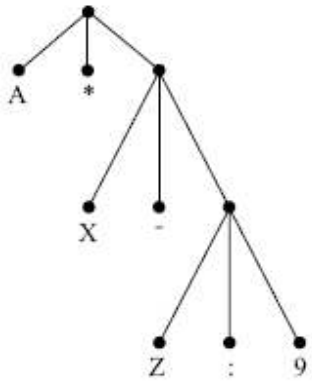
mathematische Formel: (a · (x - z : 9))

externe Darstellung als Liste: (A * (X - (Z : 9)))

interne Darstellung als Zellenstruktur:



Baumdarstellung der Liste:



Versuch Horn Clause zu erklären (Prolog):
Das Wesen der PROLOG-Programmierung besteht darin, eine Menge prädikatenlogischer Ausdrücke, die bestimmte Fakten und Regeln beschreiben, in der Gestalt von Horn-Clausen zu formulieren.

Clause, die höchstens ein positives Literal und, falls weitere Literale in der Clause vorkommen, ansonsten nur noch negative Literale enthält. Kann als Implikation, die in Clausenform dargestellt ist, aufgefasst werden: $A_1 \wedge A_2 \wedge ... \wedge A_n \rightarrow B$ entspricht $\{B; \neg A_1; \neg A_2; ...; \neg A_n\}$.

In den existierenden PROLOG-Systemen werden recht unterschiedliche Darstellungsweisen für Horn-Clausen verwendet:

Notation von *Kowalski*:	$C \rightarrow A_1, A_2, ... A_k$
Micro-PROLOG:	C if A_1 & A_2 & ... & A_k
Clocksin, Mellish:	C :- $A_1, A_2, ... A_k$

Aufbau eines Ausdruckes:

⟨PROLOG-Ausdruck⟩	::=	⟨Regel⟩. | ⟨Fakt⟩. | ⟨Anfrage⟩.
⟨Regel⟩	::=	⟨Kopf⟩ :- ⟨Körper⟩
⟨Fakt⟩	::=	⟨Kopf⟩
⟨Anfrage⟩	::=	⟨Körper⟩

Um eine **Datenbasis aufbauen** zu können, gibt es Prädikate, die Clausen zu einem vorhandenen PROLOG-Wissensbestand hinzufügen. Hier sind asserta(<Clause>) und assertz(<Clause>) zu nennen, die eine gegebene Clause an den Anfang bzw. an das Ende der Wissensbasis stellen. Genau genommen handelt es sich hierbei um Metaprädikate, da die Argumente von asserta und assertz prädikative Ausdrücke enthalten. Beispiele:

```
asserta(vater von(ernst; kurt))
asserta(vater von(kurt; gunter)).
assertz(mutter von(gunter; erna)).
assertz(mutter von(peter; jutta)).
asserta(mutter von(jutta; erna)).
assertz(elter(X; Y)) :- (vater von(X; Y);mutter von(X; Y)).
```

Ein weiteres nützliches Prädikat ist clause(P,Q). Es ist dann beweisbar, wenn in der Wissensbasis eine Regel existiert, deren linke Seite mit P und deren rechte Seite mit Q unifiziert werden kann. Fakten werden in diesem Fall als besondere Regeln aufgefasst, deren rechte Seite true ist. Beispiel:

```
?- asserta(teuer(auto)).          Antwort: yes
?- clause(teuer(auto), true).     Antwort: yes
```

Zum Entfernen von Clausen aus der Wissensbasis verwendet man das Prädikat retract(<P-Term>). Dieses Prädikat entfernt den ersten Ausdruck aus der Wissensbasis, der mit dem angegebenen P-Term unifiziert werden kann. Beispiele:

```
retract(vater von(ernst, X)).
retract(vater von( , )).
retractall(mutter von( , )).
```

Die letzten beiden Beispiele sind typische Einsatzfälle der so genannten anonymen Variablen '_'. Sie dient als Dummy-Parameter (Platzhalter), der mit einem beliebigen P-Term unifiziert werden kann.

Arithmetische Funktionen:

```
sum(X, Y, Z) :- Z is X + Y.
times(X, Y, Z) :- Z is X * Y.
remainder(X, P, Z) :- Z is X mod P.

?- integer(3).          Antwort: yes
?- integer(ernst).      Antwort: no
```

Zeichenketten: „String" ⇒ [83 , 116 , 114 , 105 , 110 , 103] „Liste" ⇒ [76 , 105 , 115 , 116 , 101]

X = Y X und Y sind mit der gleichen Zahl instantiiert

X \= Y X und Y sind mit verschiedenen Zahlen instantiiert
?- anton < arthur. Antwort: yes
?- arthur > paul. Antwort: no
?- anton \ = arthur. Antwort: yes

Beispielprogramm 1:

Fakten: struktur([3 , * , [5 , + , 7]]).
struktur([alpha , beta , gamma , delta]).
Frage: ?- struktur([X j Y]).
Antworten: X = 3, Y = [* , [5 , + , 7]]
X = alpha, Y = [beta , gamma , delta]
Frage: ?- struktur([3 , X , _]).
Antwort: X = *
Frage: ?- struktur([_ , _, [_| Z]]).
Antwort: Z = [+ , 7]

Beispielprogramm 2:

oldlist([a , b , c]).
newlist([X |j Y]) :- oldlist([X | _]), oldlist([_, _ | Y]).
?- newlist(Z).
Antwort: Z = [a , c]

PROLOG	LISP [1)]
member(X, [ ], [ ]). member(X, [X|Z], [X|Z]). member(X, [_ |Y], Z) :- member(X, Y, Z) append([ ], Y, Y). append([X|U], Y, [X|Z]) :- append(U,Y,Z) reverse([ ], [ ]). reverse([X|Y], Z) :- reverse(Y, U) append(U, [X], Z).	(DEFUN MEMBER (X L) (COND((NULL L) NIL) ((EQ X (CAR L)) L) (T (MEMBER X (CDR L⟩ (DEFUN APPEND (L Y) (COND ((NULL L) Y) (T (CONS (CAR L) (APPEND (CDR L) Y⟩ (DEFUN REVERSE (L) (COND ((NULL L) NIL) (T (APPEND (REVERSE (CDR L))(LIST (CAR L⟩

Tabelle 6.3: Listenmanipulation in PROLOG UND IN LISP
[1)] Die spitze Klammer stellt in LISP das Klammergleichgewicht bezüglich aller öffnenden runden Klammern her

append in PROLOG aufschreiben

?-append([1,2], [3,4,5], Z). → Z=[1,2,3,4,5] (normale Anwendung)
?-append(X, [3,4,5], [1,2,3,4,5]) → X=[1,2] (rekursive Anwendung)

Kann man denn in Prolog auch Objekte darstellen?

PROLOG: logikorientiert

Welche Programmiersprachen der KI kennen Sie?
Zu welcher Art der Programmiersprachen gehören Sie?
Wie ist LISP aufgebaut?
Wie ist Prolog aufgebaut?
Wie wird eine Funktion APPEND in LISP erstellt?

Expertensysteme:
Unter einem Expertensystem (abgekürzt: XS) versteht man ein wissensbasiertes System mit Fähigkeiten zur Problemlösung bzw. Inferenzausführung, das zur Lösung von Aufgaben eingesetzt wird, die im allgemeinen Spezialkenntnisse – d.h. eben einen Experten – verlangen.

X1 Fähigkeit zur Wissensrepräsentation bzw. -manipulation
X2 Fähigkeit zur effektiven Problemlösung bzw. Inferenzausführung
X3 Selbsterklärungsfähigkeit (setzt Reflexion über eigene Inferenzprozesse voraus)
X4 Fähigkeit zum Wissenserwerb oder zumindest zur Unterstützung desselben
X5 nutzerfreundliche Kommunikation mit dem Anwender
X6 Erreichen von Expertenniveau bezüglich seiner Leistungsfähigkeit
X7 Einsetzbarkeit für spezifische, hinreichend komplexe Aufgaben auf einem Gebiet, das folgende Merkmale hat: Problemraum groß; Spezialwissen erforderlich; Wissen oft heterogen oder unscharf (d.h. mit Unsicherheiten behaftet) oder sich zeitlich ändernd; Praxisrelevanz des Anwendungsfalles.

Produktionsregelsysteme: Aufbauend auf bestimmten Basisprogrammiersprachen – meist sind es die der KI – wurden spezielle sprachliche Mittel und Methodenbanken geschaffen, die den Bau von Expertensystemen unterstützen. Sie werden unter der Bezeichnung XS-Werkzeuge (oder auch Tool bzw. Tool kits) zusammengefasst. Man kann zwei Gruppen unterscheiden: reine Produktionsregelsysteme (hierzu gehört OPS5 Produktionsregel) und hybride Systeme, die neben Regeln auch Konzepte der objektorientierten Programmierung, Frames u.a. umfassen (hierzu gehören LOOPS, System, hybrides KEE, BABYLON, OPS83).

OPS5 Klassendeklaration:
<C-Deklaration> ::= (literalize <Klassenname> <Attributname>*)
Beispiele: (literalize Mitarbeiter Name Vorname Geburtsjahr Wohnort Beruf)
(literalize Spezialist Name Vorname Kategorie)
Durch diese Deklaration werden die Klassen „Mitarbeiter" und „Spezialist" mit den entsprechenden Attributen vordefiniert, ohne dass bereits konkrete Elemente dieser Klassen bekannt sind.

OPS5 Fakten:
<av-Element> ::= (<Klassenname> <AW-Paar>*) mit <AW-Paar> ::= "<Attributname> <Wert>
Beispiel: (Mitarbeiter ↑Name Bayer ↑Vorname Peter ↑Geburtsjahr 1950 ↑Wohnort Hagen
↑Beruf [Dreher Schlosser])

Atome (KE1): Elementare Einheiten, wie Zahlen, symbolische Bezeichner u.a. (auch atomare Ausdrücke genannt), aus denen sich komplexere Ausdrücke nach bestimmten syntaktischen Vorschriften zusammensetzen.

Boolesche Algebra (KE1): Algebra von Aussageelementen A, B, C ..., die zweier Werte („true" oder „false" bzw. 0 oder 1) fähig sind. Als Operationen über diesen Aussageelementen werden als einstellige Operation die Negation $\neg$ und als zweistellige Operationen die Disjunktion *v* bzw. die Konjunktion ^ verwendet. Die übrigen zweistelligen Operationen dieser Algebra lassen sich mit Hilfe der Negation und einer der beiden anderen Operationen definieren.

Built-in-Funktion (KE1): Funktion, die von einem LISP-System als elementares Konstrukt (vordefinierte Funktion) bereitgestellt wird, auf das bei der Programmierung vor allem mit den Mitteln der Funktionskomposition aufgebaut werden kann.

Chinesisches Zimmer (KE1): Von Searle vorgeschlagenes Gedankenexperiment, das eigentlich den Turing-Test widerlegen sollte. Der Grundgedanke besteht darin, dass man sich einen Menschen M vorstellt, der nicht Chinesisch beherrscht und in einem Zimmer sitzt, in das ihm Fragen in chinesischer Sprache hereingereicht werden. M verfüge über ein umfassendes Regelwerk zur Verarbeitung der chinesischen Fragen, das in einer ihm vertrauten Sprache (z.B. in Englisch) verfasst ist. Mit Hilfe dieser Regeln sei er in der Lage, zu jeder beliebigen Frage eine sinnvolle Antwort in Chinesisch zu erzeugen, die er wieder aus dem Zimmer herausreicht. Das Problem besteht nun darin zu entscheiden, ob der Mensch allein oder eventuell das ganze Chinesische Zimmer (d.h. Mensch plus Regelwerk), die ja den Turing-Test erfüllen, Chinesisch „versteht" oder nicht.

Compilierende Verarbeitung (KE1): Zweistufige Verarbeitungsweise, bei der durch einen Compiler C_Q ein Programm $p \in P$ (P Menge der Programme in einer Quellsprache L_Q) in ein selbständig ausführbares Programm $o \in O$ (O Menge der Programme in einer Maschinensprache/Objektsprache L_O) übersetzt wird. Der Compiler realisiert also zunächst eine Abbildung $C_Q : P \rightarrow O$. Bei der Abarbeitung eines Programms $o \in O$ wird dann eine zweite Abbildung $o : D \rightarrow E$ aus der Menge der Eingabedaten D in die Menge der Ausgabedaten E realisiert.

Freges Begriffsschrift (KE1): Von Frege entwickelte und an die Ausdrucksmittel der Arithmetik angelehnte Formelsprache zur Formalisierung „des Denkens". Sie bildete den ersten leistungsfähigen Formalismus zur Darstellung des Prädikatenkalküls (s. KE-2).

Funktionale Programmiersprache (KE1): Imperative Programmiersprache, die wesentlich auf dem Funktionskonzept der Mathematik beruht und deren wesentliches Konstruktionsprinzip der Aufbau komplexer funktionaler Terme aus Konstanten, Variablen oder einfacheren Termen ist (rekursiver Aufbau).

Interpretative Verarbeitung (KE1): Verarbeitungsmethode von Programmen, bei der Daten (D) und Programme (P), die in einer Quellsprache L_Q geschrieben sind, zu gleicher Zeit durch ein Verarbeitungssystem, den Interpreter I_Q, in Ergebnisdaten (E) umgewandelt werden. Ein Interpreter I realisiert also eine Abbildung: $I_Q : D \; x \; P \rightarrow E$.

Liste (KE1): Rechentechnische Darstellung einer Folge von Elementen, die entweder Atome oder wieder Listen sein können. Entsprechend dieser Definition gehören Listen zu den rekursiven Datenstrukturen.

Logische Programmiersprache (KE1): Deklarative Programmiersprache, die sich an den Konzepten der Logik orientiert und deren wesentliche Konstrukte Fakten bzw. Regeln sind. Das Analogon zu den Built-in-Funktionen der funktionalen Programmiersprachen sind hier die Built-in-Prädikate (s. KE-6).

Produktionsregel (KE1): Regel vom Typ <Bedingung> → <Aktion>, die in Abhängigkeit vom Erfülltsein einer Bedingung eine Aktion auslöst und die besonders in Expertensystemen Anwendung findet.

Rekursive Datenstrukturen (KE1): Datenstrukturen, deren Teile prinzipiell den gleichen strukturellen Aufbau besitzen, wie die Gesamtstruktur. Sie werden demgemäß durch rekursive Schemata definiert (Beispiele: Listen, Bäume konstanten Verzweigungsgrades u.a.).

Roboter (KE1): Autonomes intelligentes technisches System, das über Rezeptoren bzw. Effektoren verfügt und sich ein Modell der äußeren umgebenden Welt aufbauen kann, in der es zielgerichtet und planvoll handelt.

S-Ausdruck (KE1): Komplexe Datenstruktur, die u.a. die Listen umfasst. Ein S-Ausdruck (auch symbolischer Ausdruck genannt) ist ein gepunktetes Paar von Elementen, die entweder Atome oder wiederum S-Ausdrücke sind (rekursive Definition).

Symbolmanipulation (KE1): Art der Datenverarbeitung, in der aus Nutzersicht im Gegensatz zur Zahlmanipulation beliebige Symbole – also insbesondere nicht-numerische Symbole – und deren Kombination zu Ausdrücken im Vordergrund stehen.

Turing-Test (KE1): Ein von dem englischen Mathematiker Turing vorgeschlagener Test, der feststellen soll, ob ein technisches System ebenso intelligent
wie ein menschliches Wesen ist.

Ableitung (KE2): Gewinnung von neuen Ausdrücken aus einer Menge vorgegebener Ausdrücke nach bestimmten formalen Regeln, die sich nur auf die syntaktische Struktur – nicht aber auf den Inhalt – der Ausdrücke beziehen.

Deduktion (KE2): Besondere Form der Ableitung in logischen Kalkülen, die eng mit dem Begriff der Implikation verknüpft ist (formaler Aspekt des Schließens). Die Deduktion stellt eine Formalisierung des inhaltlichen Folgerungsbegriffes dar (s. dort).

Faktizität (KE2): Merkmal eines Sachverhalts, das aussagt, ob dieser Sachverhalt prinzipiell einen Wahrheitswert hat oder nicht, oder ob er nur hypothetisch gesetzt ist. Dieses Merkmal läst sich auch auf Objekte bzw. deren Existenz übertragen: ein hypothetisches Objekt ist demnach ein Objekt, über dessen Existenz bzw. Nicht-Existenz (zumindest in einem bestimmten Stadium der Erkenntnis) noch keine endgültige Entscheidung getroffen werden kann (Beispiel: Quarks).

Filler (KE2): Komponente eines Frames; datentechnische Darstellung des Wertes eines Merkmals in einer Frame-Repräsentation.

Folgerung (KE2): Semantischer Aspekt des Schließens eines Ausdrucks B aus einem Ausdruck A, der beinhaltet, dass immer dann, wenn A in einem bestimmten Modellbereich gilt, auch B im gleichen Modellbereich gilt.

Frame (KE2): Stereotypes Schema, das in den verschiedensten Situationen wiederkehrt und durch Slots und deren potentielle bzw. aktuelle Filler charakterisiert wird.

Implikation (KE2): Aussagenlogische Verknüpfung der Art: „Wenn A gilt, dann gilt auch B".

Inferenzmethoden (KE2): Methoden zur Automatisierung des vernünftigen Schließens in der KI; der Begriff „Inferenz" ist vom Englischen „to infer" („Schließen/Schlußfolgern") abgeleitet.

Junktor (KE2): Logischer Operator, der zwei Ausdrücke wahrheitsfunktionell miteinander verknüpft; dazu wird auch der Negator als einstelliger logischer Operator gezählt.

Kognitive Psychologie (KE2): Zweig der Psychologie, der sich mit den menschlichen Denkleistungen befasst.

Konklusion: Hinterglied/rechter Teil einer Implikation oder einer Folgerungsbeziehung.

Modalität (KE2): Stellungnahme desjenigen, der eine Aussage formuliert, zur Gültigkeit der Aussage, sowie die Formalisierung dieses Konzepts in der Logik.

Monotonie (KE2): Eine Ableitungsoperation Ab besitzt die Eigenschaft der Monotonie, wenn für zwei Ausdrucksmengen A und B mit $A \subseteq B$ gilt: $Ab(A) \subseteq Ab(B)$. Das bedeutet: die Menge der aus einer Ausdrucksmenge ableitbaren Ausdrücke kann bestenfalls wachsen (auf keinen Fall abnehmen), wenn die zugrunde liegende Ausdrucksmenge selbst wächst.

Nicht-Monotonie (KE2): Negation der Eigenschaft der Monotonie. Das bedeutet: die Menge der aus einer Ausdrucksmenge ableitbaren Ausdrücke kann durchaus abnehmen, wenn die zugrunde liegende Ausdrucksmenge selbst wächst. Oder: bei Hinzukommen neuer Informationen müssen sehr oft alte Annahmen revidiert werden.

Pattern (KE2): Datenstruktur (auch „Muster" genannt) mit bestimmten, als Variable gedeuteten Strukturelementen, die zum Vergleich mit anderen Datenstrukturen dient. Vertritt eine ganze Klasse von Daten.

Pattern matching (KE2): Mustervergleich; Überprüfung, ob ein gegebenes Pattern (Muster) mit einem bestimmten Datum übereinstimmt bzw. durch geeignete Variablensubstitution mit diesem zur Übereinstimmung gebracht werden kann.

Prämisse (KE2): Vorderglied/Voraussetzung einer Folgerung oder einer logischen Implikation.

Slot (KE2): Komponente eines Frames; datentechnische Darstellung des Merkmals eines Objekts.

Reasoning (KE2): Auch in der deutschsprachigen Literatur oft verwendete und nicht übersetzte Bezeichnung für das rationale/vernünftige Denken (Engl.: to reason –vernünftig denken, argumentieren).

Vererbung (KE2): Übertragung von Eigenschaften von einem übergeordneten auf einen untergeordneten Begriff. Dieser Vorgang wird insbesondere in Frame-orientierten und Objekt-orientierten Wissensrepräsentationssystemen automatisch unterstützt.

UML (KE2): Die Unified Modeling Language („vereinheitlichte Modellierungssprache"), häufig mit UML abgekürzt, ist eine von der Object Management Group (OMG) entwickelte und

standardisierte Beschreibungssprache, um Strukturen und Abläufe in objektorientierten Softwaresystemen darzustellen. Im Sinne einer Sprache definiert die UML dabei Bezeichner für die meisten Begriffe, die im Rahmen der Objektorientierung entstanden sind, und legt mögliche Beziehungen zwischen diesen Begriffen fest. Die UML definiert des weiteren grafische Notationsweisen für diese Begriffe und für Modelle von Software oder anderen Abläufen, die man in diesen Begriffen formulieren kann. Damit wurde die UML zur de-Facto-Norm für \technische Zeichnungen" von Programmen und Abläufen.

Allgemeinbegriff (KE3): Begriff, der im Gegensatz zum Individualbegriff auf eine Gesamtheit von Gegebenheiten (Entitäten) zutrifft; auch *generischer Begriff* genannt.

CDT-Theorie (KE3): Conceptual Dependency Theory; objektorientierte Wissensrepräsentation in der Form eines semantischen Netzes, bei der alle Sachverhalte auf einige wenige universelle Fundamental-Konzepte zurückgeführt werden.

Computer-Lexikographie (KE3): Zweig der Computerlinguistik, der sich mit dem Inhalt und mit dem Aufbau von Wörterbüchern befasst, die für die automatische Sprachverarbeitung eingesetzt werden.

Datenmodell (KE3): Logische Struktur der Wissensbasis einer traditionellen Datenbank.

Extension (KE3): Bedeutungsaspekt, der den Umfang eines Begriffes, seine Beziehung zur realen Welt oder auch zu einer gedachten (möglichen) Welt zum Ausdruck bringt (vgl. Intension).

Generischer Begriff (KE3): s. *Allgemeinbegriff*

Graph (KE3): Mathematische Struktur, die bildlich als eine Menge von Punkten und zwischen diesen verlaufenden (gerichteten oder ungerichteten) Strecken dargestellt wird. Formal wird ein Graph als Tripel $G = <N;K; \Gamma >$ charakterisiert, das aus einer Menge N von Knoten, einer zu N disjunkten Menge K von Kanten und einer Kantenzuordungsfunktion Γ besteht, die jeder Kante aus K ein Paar von Knoten aus N *x* N zuordnet.

Individualbegriff (KE3): Begriff, dessen Begriffsumfang (Extension) sich auf ein einziges Individuum eines Gegenstandsbereichs bezieht. Gegensatz: Allgemeinbegriff.

Intension (KE3): Bedeutungsaspekt, der den Inhalt eines Begriffes, seine Beziehung zu anderen Begriffen zum Ausdruck bringt (vgl. Extension).

Kardinalität (KE3): Mächtigkeit einer Menge; wird für endliche Mengen durch die Anzahl der Elemente bestimmt (auch Kardinalzahl genannt). Kanonische Bedeutungsdarstellung: Darstellungsform des Inhalts von natürlichsprachlichen Ausdrücken, bei welcher bedeutungsgleiche natürlichsprachliche Sätze auf dieselbe Tiefenstruktur zurückgeführt werden.

Oberflächenstruktur (KE3): Darstellungsform eines Satzes oder Textes, so wie er dem Hörer/Leser dargeboten wird.

Pragmatik (KE3): Zweig der Zeichenlehre (Semiotik), der sich mit dem Verhältnis von Sender (beabsichtigte Wirkung) und Empfänger (tatsächlich erzielte Wirkung) bei einer Informationsübermittlung durch Zeichen befasst.

Sachverhalt (KE3): Der Begriff „Sachverhalt" wird im Basistext in einem verallgemeinerten Sinn für komplexe situative Gegebenheiten (Entitäten) verwendet, denen entweder eine reale Entsprechung zukommen kann (Fakten, reale Sachverhalte) oder die nur hypothetisch gesetzt/gedacht sind (so genannte hypothetische Sachverhalte oder Pseudosachverhalte).

Semantik (KE3): Der Zweig der Zeichenlehre (Semiotik), der sich mit dem Inhalt von symbolischen Ausdrücken befasst; gleichzeitig Bezeichnung für die Bedeutung natürlichsprachlicher Ausdrücke.

Semantisches Netz (KE3): Repräsentation der Bedeutung von (vorwiegend natürlichsprachlich gegebenen) Informationen in Form eines markierten orientierten Graphen. Dabei entsprechen den Knoten des Graphen Begriffe der natürlichen Sprache und den Kanten Beziehungen zwischen diesen Begriffen.

Syntax (KE3): Zweig der Zeichenlehre (Semiotik), der sich mit dem formalen Aufbau von symbolischen Ausdrücken befasst; gleichzeitig Bezeichnung für die strukturellen Beziehungen der Wörter eines natürlichsprachlichen Konstrukts zueinander.

Tiefenkasus (KE3): Verallgemeinerter gemeinsamer Inhalt, der jeweils verschiedenen grammatischen Kasus in den einzelnen Sprachen mit ausgeprägtem Kasussystem auf semantischer Ebene zugrunde liegt (ein auf den amerikanischen Sprachwissenschaftler Fillmore zurückgehender Begriff [92]).

Tiefenstruktur (KE3): Darstellung der syntaktischen Struktur oder des semantischen Gehalts eines natürlichsprachlichen Konstrukts bei der automatischen Verarbeitung natürlicher Sprache (syntaktische bzw. semantische Tiefenstruktur).

Agent (KE4): Selbständig handelnder Problemlösemodul mit eigener Wissensbasis und eigenen Problemlösetechniken, der im Rahmen einer Gesamtproblemlösung mit anderen Agenten kooperiert oder ihnen antagonistisch gegenübersteht.

Antagonismus (KE4): Gegensatz zwischen verschiedenen Gruppen oder Interessen.

ALPHA-BETA-Technik (KE4): eine mit geeignet gewählten oberen und unteren Schranken für die Zustands-Bewertung arbeitende Problemlösemethode, die auf dem Minimax-Prinzip aufbaut.

Heuristik (KE4): Eine empirisch gestützte Findehilfe, die eine Problemlösemethode i.a. stark effektiviert, aber dafür das Risiko in sich birgt, dass bei ihrem Einsatz Lösungen verloren gehen. Eine Heuristik verkörpert problemspezifisches Wissen, das zur Lösungsfindung eingesetzt wird.

Means-ends-analysis (KE4): Methode zur Problemlösung, bei der in Abhängigkeit von der Differenz zwischen augenblicklich erreichtem Problemzustand und Zielzustand die geeignete Operation zur Fortsetzung des Verfahrens ausgesucht wird.

MINIMAX-Prinzip (KE4): Verarbeitungsprinzip, das der Problemlösung für antagonistische Probleme zugrunde liegt. Nach diesem Prinzip versucht jeder beteiligte Problemlöser (Spieler), seine eigenen Zustandsbewertungen (Gewinnaussichten) zu maximieren und die des Gegners zu minimieren.

MINIMAX-Verfahren (KE4): Anwendung des MINIMAX-Prinzips bei der Lösung antagonistischer Probleme.

Planung: Entwicklung einer Konzeption für eine Folge von Handlungen, mit denen ein bestimmtes Ziel erreicht werden soll.

Problem (allgemein) (KE4): Spannungsfeld zwischen den Zielen eines Individuums und der augenblicklichen Lage bzw. der augenblicklichen Situation, in der es sich befindet.

Problem (formal) (KE4): Quadrupel aus Zustandsmenge Z, Anfangszustand $a \in Z$, Menge von Zielzuständen $ZZ \subseteq Z$ (die Menge der Lösungen) und einer Operatormenge O, wobei gilt: $o: Z \rightarrow Z$ für alle $o \in O$.

Problemgraph (KE4): Darstellung eines formalen Problems als Graph in der Weise, dass den Problemzuständen die Knoten des Graphen entsprechen und den Kanten Übergänge zwischen den Zuständen, die durch eine Operatoranwendung vermittelt werden. Dem Anfangszustand entspricht eine Wurzel des Graphen und den Zielzuständen entsprechen ausgezeichnete terminale Knoten, von denen keine Knoten ausgehen.

Problemzustand (KE4): Repräsentant der Situation, die in einem Problemlöseverfahren nach Ausführung einer bestimmten Anzahl von Suchschritten erreicht wurde.

Rochade (KE4): Einziger Zug im Schachspiel, bei dem gleichzeitig zwei Figuren (und zwar König und Turm) der gleichen Partei bewegt werden. Setzt voraus, dass der König und der beteiligte Turm noch nicht gezogen wurden.

Suchbaum (KE4): Spezielle Form eines Problemgraphen, die dann entsteht, wenn alle Problemzustände voneinander verschieden und vom Ausgangszustand auf genau einem Weg erreichbar sind. Im Prinzip lässt sich jeder Problemgraph in einen Suchbaum entwickeln, wenn man die Entstehungsgeschichte (d.h. die zur Generierung erforderliche Folge von Operatoranwendungen) zur Charakterisierung eines Problemzustandes hinzunimmt.

Strategie (KE4): Plan eines Agenten oder Spielers im Rahmen eines Problemlöseprozesses, der dem betreffenden Agenten oder Spieler in jeder Situation vorschreibt, welche Aktion als nächste auszuführen ist.

Wert (eines Zweipersonenspiels) (KE4): Bewertung derjenigen Spielsituation (oder Strategiekombination), die sich ergibt, wenn beide Spieler ihre optimale Strategie verfolgen; vgl. hierzu: MINIMAX-Prinzip.

Abduktion (KE5): Umkehrung der Deduktion mit Hilfe des Modus ponens, bei der aus einer angenommenen Implikation $A \rightarrow B$ und einem angenommenen Hinterglied B auf das Vorderglied A geschlossen wird.

Atomare Formel (KE5): Elementarer prädikatenlogischer Ausdruck, der entweder die Prädikation P(t) eines einstelligen Prädikats P über einen Term t (also im einfachsten Fall auch über eine Konstante oder eine Variable) oder das Inbeziehungsetzen von n Termen $t_1, ..., t_n$ mit Hilfe eines n-stelligen Relators R_n in der Form $R_n(t_1, ..., t_n)$ zum Ausdruck bringt.

Approximatives Schließen (KE5): Oberbegriff für alle Schlußweisen, die nicht exakt, sondern nur mit einem bestimmten Gewißheitsgrad gelten (als Maß für die Sicherheit des Schlusses kommen z.B. subjektive Gewißheitsgrade oder Wahrscheinlichkeitsmaße in Frage)

Clause (KE5): Menge von disjunktiv verknüpften Literalen (in der Literatur häufig auch als Klausel bezeichnet).

Disjunktive Normalform (KE5): Darstellung von logischen Ausdrücken als Disjunktion von konjunktiv verknüpften Literalen.

Herbrand-Universum (KE5): Aufzählbare Menge von Grundtermen, die konstruktiv aus den Konstanten- und Funktionssymbolen einer Ausdrucksmenge in Skolemnormalform definiert werden und zur Feststellung der Erfüllbarkeit bzw. Nicht-Erfüllbarkeit dieser Ausdrucksmenge dient.

Homomorphismus (KE5): Relationserhaltende Abbildung f: A $\rightarrow$ B aus einer Menge A in eine Menge B. Wenn über A die Verknüpfung (Operation bzw. Relation) $\circ$ und über B die Verknüpfung $\bullet$ definiert ist, dann gilt: f (x $\circ$ y) = f (x) $\bullet$ f (y) für alle x, y $\in$ A

Konjunktive Normalform (KE5): Darstellung von logischen Ausdrücken als Konjunktion von disjunktiv verknüpften Literalen.

Literal (KE5): Negierte oder unnegierte atomare Formel der Logik.

Modus ponens (KE5): Schlussegel, bei der aus der Gültigkeit einer Aussage A und der Gültigkeit der Implikation A $\rightarrow$ B auf die Gültigkeit von B geschlossen wird.

Normalform (KE5): Nach bestimmten Strukturprinzipen aufgebaute Standardform von Ausdrücken, die sich durch besondere Einfachheit auszeichnet (Beispiel: Konjunktive Normalform).

Pränexnormalform (KE5): Normalform, bei der alle Quantoren am Anfang der Formel stehen. Die Quantoren zusammen mit den durch sie gebundenen Variablen bilden das Präfix, der übrige (quantorfreie) Teil der Formel wird Matrix genannt.

Resolutionsprinzip (KE5): Ableitungsregel für Ausdrücke in Skolemnormalform bzw. Clausenmengen (Verallgemeinerung der Schnittregel).

Schnittregel (KE5): Ableitungsregel der Aussagenlogik, mit der durch Herausschneiden eines negierten Literals :A bzw. unnegierten Literals A aus zwei Formeln F_1 bzw. F_2 und anschließende Vereinigung der Ergebnisformeln eine neue Formel F (die abgeleitete Formel) gewonnen wird.

Skolemfunktion (KE5): Funktion y = sk(x_1, ..., x_n) zur Kennzeichnung der Abhängigkeit zwischen einer existenziell quantifizierten Variablen y eines prädikatenlogischen Ausdrucks A und allen universell quantifizierten Variablen x_1, ..., x_n, die im Präfix von A vor der Variablen y stehen (s. Pränexnormalform). Die Benennung der Funktion sk ist willkürlich (bis auf die Forderung nach Namensverschiedenheit zu anderen Funktionen des Formalismus, zu dem A gehört).

Skolemnormalform (KE5): Pränexnormalform, bei der die Abhängigkeiten von existentiell quantifizierten Variablen eines Ausdrucks A von den in A vorkommenden universell quantifizierten Variablen durch Skolemfunktionen ausgedrückt sind.

Struktur (KE5): Unter einer Struktur $S_G = <G, F, P>$ versteht man einen *Gegenstandsbereich* G, zusammen mit einem System F von *Funktionen* und einem System P von *Attributen* bzw. *Relationen*, die über G definiert sind.

Term (KE5): Ausdruck, der entweder ein Konstantensymbol, ein Variablensymbol oder die Anwendung eines Operators f auf eine der Stelligkeit von f entsprechende Anzahl von Termen darstellt (rekursive Definition).

Agent (KE6): Im Kontext „Expertensysteme" $\rightarrow$ Handelnder in einem Problemlöseprozess.

Backtracking (KE6): Zurückgehen im Suchbaum während des Problemlösevorgangs, und zwar bis zu einem Punkt, an dem noch alternative Suchmöglichkeiten offen sind. Wird ausgelöst, wenn die Suche in eine Sackgasse geraten ist.

Blackboard-Modell (KE6): Steuerungsmodell, das die Arbeit mehrerer kooperierender Agenten bei der Problemlösung und deren Kommunikation über eine zentrale Datenstruktur (das Blackboard) überwacht und regelt.

Deadlock (KE6): Situation in einem von mehreren Agenten ausgeführten Problemlöseprozess, in der alle Agenten wechselseitig auf die Ergebnisse der anderen Agenten warten, um ihre eigen Arbeit fortsetzen zu können (Stillstand, der insbesondere in *opportunistischen* Problemlöseverfahren auftreten
kann).

Hierarchisches Problemlösen (KE6): Methode der sukzessiven Verfeinerung von Problemlösungen durch Betrachtung des Problems auf unterschiedlichen Generalisierungsebenen. Dabei werden Operatoren einer niedrigeren Ebene als Spezialisierungen von Operatoren einer höheren Problemslösungsebene aufgefasst.

Horn-Clause (KE6): Clause, die höchstens ein positives Literal und, falls weitere Literale in der Clause vorkommen, ansonsten nur noch negative Literale enthält. Kann als Implikation, die in Clausenform dargestellt ist, aufgefasst werden: $A_1 \wedge A_2 \wedge ... \wedge A_n \rightarrow B$ entspricht $\{B; \neg A_1; \neg A_2; ...; \neg A_n\}$.

Inferenz (KE6): Einzelschritt in einem Verfahren des intelligenten Schließens / Schlussfolgerns, wie z.B. deduktives oder induktives Schließen (abgeleitet von „to infer" - schlussfolgern).

Konsistenz (KE6): Widerspruchsfreiheit einer Wissensbasis, die in wissensbasierten Systemen vor allem nach der Eingabe neuen Wissens und nach der Regel-Ausführung immer wieder neu überprüft bzw. gesichert werden muss (Wird von einem so genannten Konsistenz-Checker durchgeführt).

Knowledge acquisition (KE6): Breites Spektrum von Techniken des Wissenserwerbs für Expertensysteme, das von psychologischen Methoden des Brainstorming oder Fragebogen-Techniken bis hin zur rechnergestützten Abfrage von Experten mit Hilfe der automatischen Verarbeitung natürlicher Sprache reicht.

Knowledge engineer (KE6): Tätigkeitsbezeichnung für einen Spezialisten (Wissensingenieur), der die Entwicklung wissensbasierter Systeme, insbesondere von Expertensystemen, betreibt;

eine seiner Aufgaben ist die Unterstützung des Wissenserwerbs und der Extraktion von Wissen aus dem Erfahrungsschatz eines menschlichen Experten.

Knowledge enterer (KE6): Spezialist, der die Eingabe von Wissen in große Wissensbasen (oft mit Hilfe besonderer Tools) vornimmt.

λ-Kalkül (KE6): Von Church eingeführter Kalkül zur Formalisierung des Begriffs der Berechenbarkeit, dessen Basisausdrucksmittel Terme sind. Als Terme gelten Variable, λ-Terme der Gestalt (λx. T), wobei T wieder ein Term ist, und Termapplikationen (T_1 T_2), mit T_1 und T_2 als Termen. Als Operationen des Kalküls werden Substitutionen und Termreduktionsregeln definiert. Mit diesen Ausdrucksmitteln lässt sich auch die Semantik von LISP-Programmen spezifizieren.

Least commitment (KE6): Prinzip der geringstmöglichen Festlegung in einem Problemlöseprozess, das eine Balance zwischen zu früher Festlegung auf Lösungsalternativen (Gefahr von Sackgassen) und zu später Entscheidung (Gefahr von Deadlocks) herstellen soll.

Opportunistische Problemlösung (KE6): Methode des Problemlösens, bei der einzelne Agenten, Spezialisten oder Moduln eines Problemlöseprozesses jeweils dann selbständig ihre Arbeit beginnen, wenn ihre Einsatzbedingungen erfüllt sind. Sie werden dabei nicht von einem zentralen Steuermechanismus überwacht bzw. eingesetzt.

Scheduler (KE6): Komponente eines Expertensystems oder eines Problemlösers, die die Steuerstrategie realisiert und die Regelauswahl vornimmt.

Schnappschuß (KE6): Festhalten der alternativen Fortsetzungsmöglichkeiten in einem Verzweigungspunkt eines Problemlöseprozesses zum Zweck der Wiederaufnahme der Verarbeitung an diesem Punkt bei einem späteren Backtracking.

Unsicheres Schließen (KE6): Schlussweise, die im Gegensatz zur Aussagenlogik nicht mit diskreten Wahrheitswerten, sondern mit kontinuierlichen Gewissheitsgraden (Evidenzfaktoren) bzw. Unsicherheitsfaktoren arbeitet.

Computerlinguistik (KE7): Interdisziplinärer Wissenschaftszweig, der sich mit der Theorie und Praxis der automatischen Sprachverarbeitung befasst und der zwischen Informatik und Linguistik angesiedelt ist. Im weitesten Sinne werden hierzu alle Methoden gezählt, die den Computer zur Bearbeitung linguistischer Fragestellungen benutzen (z.B. auch textstatistische Untersuchungen). Im engeren Sinne – diese Sicht wird in der KI eingenommen – gehören zu diesem Gebiet nur die Methoden, die sich mit der Automatisierung des Sprachverstehens befassen.

ATN-Grammatik (KE7): Abkürzung für Augmented Transition Network Grammar; Typ einer Grammatik, die als Übergangsnetzwerk dargestellt ist, mit dessen Kanten prozedurale Elemente (wie Tests und Registeroperationen) verknüpft werden, die zur Behandlung kontextabhängiger Spracherscheinungen, wie z.B. von Kongruenzen, und zum Aufbau von Ergebnisstrukturen dienen.

CFG (KE7): Abkürzung für Context Free Grammar; von Chomsky eingeführter Grammatiktyp, dessen Regeln $B \rightarrow C_1 \ldots C_n$ sich dadurch auszeichnen, dass die Ersetzung eines nicht-terminalen Symbols B (linke Regelseite) in einer Ausgangszeichenkette Z durch die auf der

rechten Regelseite stehenden Symbole C_1 ... C_n nicht von den in Z links und rechts von B stehenden Symbolen abhängt.

Featurestruktur (KE7): Merkmals-Wert-Schema, das grammatischen Merkmalen (wie GENUS bzw. NUMERUS) bestimmte Werte (z.B. FEMININUM bzw. PLURAL) zuordnet und als Argument-Wert-Zuordnungstabelle einer Funktion mit endlichem Definitions- und Wertebereich aufgefasst werden kann.

Frage-Antwort-System (KE7): KI-System, das Sprachverstehen im umfassendsten Sinne modelliert und sowohl die Eingabe von Informationsbeständen als auch die Abfrage der Wissensbasis in natürlicher Sprache vorsieht.

Grammatische Struktur (KE7): Formale, meist hierarchisch gegliederte Beschreibungsform, die den Aufbau und die innere Struktur eines natürlichsprachlichen Ausdrucks oder eines Textes widerspiegelt (hängt stark von der zugrunde liegenden Grammatikauffassung ab).

Kongruenz (KE7): Übereinstimmung von Konstituenten natürlichsprachlicher Ausdrücke in bestimmten grammatischen oder semantischen Merkmalen (diese bestimmen wesentlich die Kontextabhängigkeit der natürlichen Sprache mit).

Konstituente (KE7): Relativ geschlossene Einheit eines Satzes/Textes, die aus ein oder mehreren Wörtern besteht und einen inneren Zusammenhang aufweist. Sie kann durch Weglass-, Umstellungs- und Ersetzungstests ermittelt werden.

Kontextabhängige Grammatik (KE7): Grammatiktyp in der von Chomsky eingeführten Hierarchie von formalen Grammatiken, der dadurch gekennzeichnet ist, dass die Ersetzung eines nicht-terminalen Symbols B in einer Zeichenkette Z nach Regeln vom Typ: $\varphi\ B\ \psi \rightarrow \varphi\ C_1$ ... $C_n\ \psi$ erfolgt. Danach ist die Ersetzbarkeit von B durch die Symbole C_1 ... C_n vom umgebenden Kontext φ, ψ abhängig.

Kontextfreie Grammatik (KE7): s. CFG

LFG (KE7): Abkürzung für *Lexikalisch Funktionale Grammatik*; Grundtyp einer Grammatik zur Beschreibung natürlicher Sprachen, bei der eine im Kern kontextfreie Grammatik durch Termgleichungen attributiert wird, die Beziehungen zwischen Feature-Strukturen beschreiben.

NLI (KE7): Natural language interface (natürlichsprachliches Interface); Nutzerschnittstelle zu einem traditionellen Softwareprodukt, zum Internet oder einem KI-System, das eine Kommunikation des Nutzers mit diesem System in natürlicher Sprache ermöglicht.

Parsing (KE7): Prozess der Analyse eines natürlichsprachlichen Konstrukts (eines Satzes, eines Textes usw.), der sich auf eine vorgegebene Grammatik stützt. Dabei wird festgestellt, ob dieses Konstrukt in dem von der Grammatik beschriebenen Sprachausschnitt liegt oder nicht. Im ersteren Fall wird im Ergebnis des Parsing zusätzlich eine vom verwendeten Grammatiktyp bestimmte grammatische Struktur des zu analysierenden Konstrukts abgeleitet.

Reguläre Grammatik (KE7): Spezieller Typ einer kontextfreien Grammatik mit Regeln der Gestalt $B \rightarrow C_1\ C_2$, wobei C_1 ein terminales (nicht weiter ersetzbares) Symbol und C_2 entweder leer oder ein nicht-terminales Symbolist.