Inferenz (Schlussfolgern)

wahrheitserhaltend

Deduktion

modus ponens (a->b, a=>b) modus pollens (a->b, !b=>!a)

Aussagenlogik

Formeln: IA: Buchstaben, IS: zwei Formeln mit ^, v, ->, !, <> verknüpft

Semantik: Fkt I ordnet jeder Variablen (Buchstabe) einen Wert aus 0, 1 zu. Wert I(Forme ergibt sich induktiv aus Wahrheitstabelle

- FOLGERN falls jedes Modell für F auch Modell für G ist, so folgt G aus F. (F |=
- falls I(F)=1 nennt man I ein MODELL für F
  - erfüllbar
  - allgemeingültig(Tautologie)
  - widersprüchlich(Kontradiktion)
- F, G semantisch äquiv. falls F|=G und G|=F

Folgerbarkeit prüfen:

- Wahrheitstabelle (Aufwand bei vielen Variablen zu groß)
- Resolution
  - falls W ^ !H widersprüchlich, so gilt W |= H
  - W ^ !H in KNF bringen
  - Resolutionsregel.....: (L1 v...vLiv A vLjv...vLn) ^ (L'1v...vL'kv !A vL'hv...vL'm) |= (L1 v...vLn v L'1 v....vL'm)
  - lässt sich die leere Klause resolvieren, war die Formel widersprüchlich, also galt W|=H

Prädikatenlogik

Formeln enthalten nicht nur Variablen sondern:

- Konstanten
- Funktionen
- Prädikate
- Quantoren

Terme: IA: Variable, Konstante; IS: Funktionsymb(Term)

Formel: IA: t1 und t2 Terme: Prädikatensymb(t1) oder t1=t2 ; IS: F, G Formel: ^, v, !, ->, <> ; ..... x Variable: forall x F, exists x F

Semantik Strukturen

- Universum U
- Interpretation I: ordnet jeder VariablenElement aus U zu, jedem FktSymb eine Funktion Ux..xU->U und jedem PrädSymb eine Fkt Ux..xU->[0, 1] zu
- Induktive Definition der Wahrheitswerte von Formeln wie immer

Eigenschaften

- nicht entscheidbar ob Formel erfüllbar
- widersprüchlichkeit jedoch beweisbar (Algorithmus terminiert nicht falls kein widerspruch)

Resolution funktioniert

- BEREINIGEN
  - freie u. gebundene Var: untersch. Namen
  - keine->, <>
- PRÄNEXFORM (Quantoren vorne)
  - !ExF = AxIF und umgekehrt
  - (QxF op G) = Qx(F op G) wenn bereinigt
- SKOLEMISIERUNG
  - (Existenzquantoren löschen, jeweilige Variable durch Funktion aller Allquantoren vor dem Existenzquantor ersetzen)
  - nicht semantisch äquivalent aber F erfüllbar gdw F' erfüllbar
- Allquantoren weglassen, in KNF bringen
- UNIFIKATION nötig
  - allgemeinster Unifikator: Subst* die Unifiziert, so dass alle andr Subst durch Subst* und eine weitere Subst dargestellt werden können
  - Algorithmus: Eingabe P(t1...tn); P(t1'...tm').
    - Suche i mit ti!=ti'. Suche Stelle in ti un ti' mit Unterschied.
      - Falls beides keine Var ->Fehler
      - Falls eins x und andres t
        - sehe ob x in t vorkommt (OccurCheck-> Fehler)
        - sonst ersetze x durch t und suche nach nächstem i mit ti!=ti'

nicht wahrheitserhaltend

Induktion

an Beispielen (8hall ging morgens die Sonne auf, kein Gegenbeispiel da, -> immer geh morgens die Sonne auf!)

Abduktion

a->b, b => a

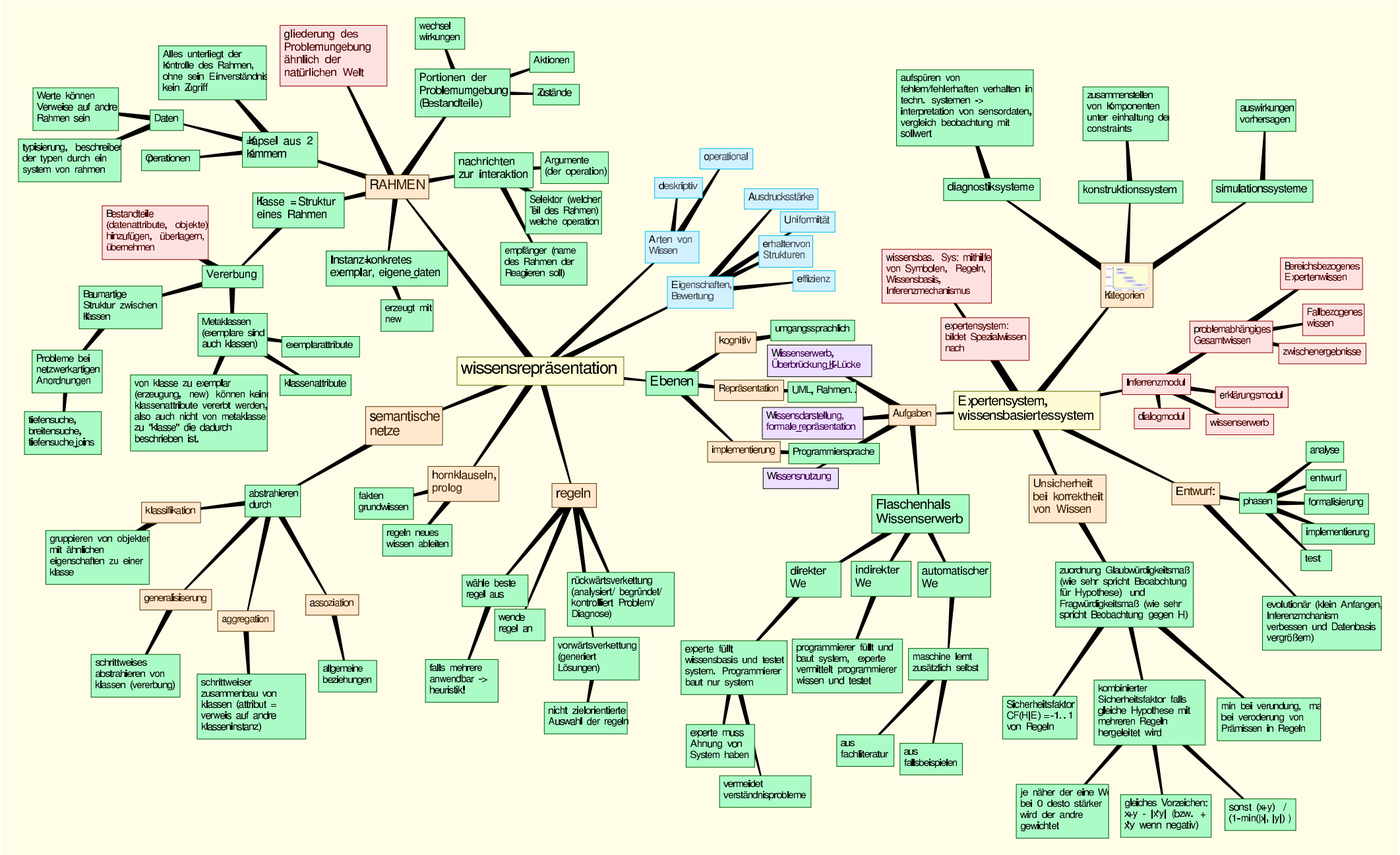
- Umkehrschluss auf plausible Ursache
- nicht zwingend richtig, geeignet falls keine "bessere" Hypothese gefunden werden kann

logische Programmierung (Deklarativ)

- Algorithm=Logic+Control
- entspricht Resolution
  - f<-v1, v2 entspr. Klausel !v1 v !v2 v f
  - Resolution mit f: !v1, !v2 übrig
  - weiter nun v1 mit anderem Regelkopf resolvieren.
  - usw. bis leere Klausel
- Auf Anfrage hin wird Ziel gesucht (das in Regel vorne steht) das mit aktuellem Ziel/Anfrage unifiziert werden kann
- hornK. notwendig damit input-resolution funzt, also immer ein resolutionspartner aus urspr. Eingabe, nie 2 abgeleitete kombinieren
- Programme= Regeln (Folgerung :- Voraus1 Voraus2...)
  - Negative literale in Regeln nicht zulässig.
  - Modelle sind solche die alle Grundatome die sich aus Regeln folgern lassen mit 1 bewerten
  - Struktur die _alle_ Grunadatome mit _1_ bewertet ist immer Modell
  - Somit auch kein negatives literal beweisbar (mit deduktion)
- ?- Anfrage
- Prädikat das immer erfüllt ist

Prolog

- [a, b, c, d] liste mit Elementen abcd, [a|L] liste L folgt auf Element a
- probl. bei unendlichem Pfad -> unvollständig!! aber viel weniger Speicherplatz nötig
  - Tiefensuche
  - Regelabarbeitung der Reihenfolge nach
  - evtl. falsches YES aber viel schneller
  - kein Occur-Check
- META-Interpreter
- cut! Abschneiden der Suche
  - begegnet Prolog Cut, nutzt es danach keine andren Regeln mit dem gleichem Kopf mehr (beim Backtracking, falls die aktuelle Regel fehlschlägt.)
- Planung mit Prolog
  - Strips
  - Beispiel: Labyrinth durch Kämmern
    - für jede Verbindung i zwischen kammern a, b gilt uebergang(a, i, b)
    - übergänge mittels lösung(X,Y,[i|L]) :- uebergang(X,i,Z), lösung(Z,Y,L)
    - es gilt loesung(X,X,[]) (Um von X nach X zu kommen braucht man nix tun
- Negation
  - Approximation: negation by finite failure. Wird nach einer bestimmten Anzahl Schritte nicht das Gegenteil bewiesen, wird angenommen Aussage ist not p wahr
    - not(x) :-x, !, fa not(x).
    - gilt x, darf wegen des cut die Regel nicht angewendet werden. Gilt x nicht kommt man nicht b zum Cut, und 2. Regel wird verwendet
    - starke Variante der closed world Assumption
    - "falsche" Negation bei Variablen
      - es kann vorkommen das Antwort N ist (weil es eine Variablenbelegung gibt wo Ziel erfüllt, negation also nicht) obwohl es auch eine Belegung gibt wo das Ziel nicht erfüllt ist
      - implizite ALLQUANTIFIZIERUNG beachten!!


wissensrepräsentation
RAHMEN
gliederung des Problemungebung ähnlich der natürlichen Welt
Portionen der Problemumgebung (Bestandteile)
wechsel wirkungen
Aktionen
Zustände
Kapsel aus 2 Kammern
Alles unterliegt der Kontrolle des Rahmen, ohne sein Einverständnis kein Zugriff
Daten
Werte können Verweise auf andre Rahmen sein
typisierung, beschreiben der typen durch ein system von rahmen
Operationen
Klasse = Struktur eines Rahmen
Vererbung
Bestandteile (datenattribute, objekte) hinzufügen, überlagern, übernehmen
Baumartige Struktur zwischen Klassen
Probleme bei netzwerkartigen Anordnungen
tiefensuche, breitensuche, tiefensuche_joins
Metaklassen (exemplare sind auch klassen)
exemplarattribute
klassenattribute
von klasse zu exemplar (erzeugung, new) können keine klassenattribute vererbt werden, also auch nicht von metaklasse zu "klasse" die dadurch beschrieben ist.
Instanz-konkretes exemplar, eigene_daten
erzeugt mit new
nachrichten zur interaktion
Argumente (der operation)
Selektor (welcher Teil des Rahmen) welche operation
empfänger (name des Rahmen der Reagieren soll)
semantische netze
abstrahieren durch
klassifikation
gruppieren von objekten mit ähnlichen eigenschaften zu einer klasse
generalisiserung
schrittweises abstrahieren von klassen (vererbung)
aggregation
schrittweiser zusammenbau von klassen (attribut = verweis auf andre klasseninstanz)
assoziation
allgemeine beziehungen
hornklauseln, prolog
fakten grundwissen
regeln neues wissen ableiten
regeln
wähle beste regel aus
wende regel an
falls mehrere anwendbar -> heuristik!
rückwärtsverkettung (analysiert/ begründet/ kontrolliert Problem/ Diagnose)
vorwärtsverkettung (generiert Lösungen)
nicht zielorientierte Auswahl der regeln
Ebenen
Arten von Wissen
deskriptiv
operational
Eigenschaften, Bewertung
Ausdrucksstärke
Uniformität
erhalten von Strukturen
effizienz
kognitiv
umgangssprachlich
Wissenserwerb, Überbrückung K-Lücke
Repräsentation
UML, Rahmen...
Wissensdarstellung, formale_repräsentation
implementierung
Programmiersprache
Wissensnutzung
Aufgaben
Flaschenhals Wissenserwerb
direkter We
experte füllt wissensbasis und testet system. Programmierer baut nur system
experte muss Ahnung von System haben
vermeidet verständnisprobleme
indirekter We
programmierer füllt und baut system, experte vermittelt programmierer wissen und testet
automatischer We
maschine lernt zusätzlich selbst
aus fachliteratur
aus fallsbeispielen
Expertensystem, wissensbasiertessystem
wissensbas. Sys: mithilfe von Symbolen, Regeln, Wissensbasis, Inferenzmechanismus
expertensystem: bildet Spezialwissen nach
Kategorien
diagnostiksysteme
aufspüren von fehlern/fehlerhaften verhalten in techn. systemen -> interpretation von sensordaten, vergleich beobachtung mit sollwert
konstruktionssystem
zusammenstellen von Komponenten unter einhaltung der constraints
simulationssysteme
auswirkungen vorhersagen
Inferenzmodul
problemabhängiges Gesamtwissen
Bereichsbezogenes Expertenwissen
Fallbezogenes wissen
zwischenergebnisse
erklärungsmodul
dialogmodul
wissenserwerb
Unsicherheit bei korrektheit von Wissen
zuordnung Glaubwürdigkeitsmaß (wie sehr spricht Beoabchtung für Hypothese) und Fragwürdigkeitsmaß (wie sehr spricht Beobachtung gegen H)
Sicherheitsfaktor CF(H|E) =-1...1 von Regeln
kombinierter Sicherheitsfaktor falls gleiche Hypothese mit mehreren Regeln hergeleitet wird
je näher der eine W bei 0 desto stärker wird der andre gewichtet
gleiches Vorzeichen: x+y - |x*y| (bzw. + x*y wenn negativ)
sonst (x+y) / (1-min(|x|, |y|))
min bei verundung, ma bei verodernung von Prämissen in Regeln
Entwurf:
phasen
analyse
entwurf
formalisierung
implementierung
test
evolutionär (klein Anfangen, Inferenzmchanism verbessen und Datenbasis vergrößern)

Mittel-Zweck-Analyse

- welches Mittel verwenden um was zu erreichen
  - bis 1 km zufuss
  - ab 100km Bahn
  - ab 1000km flugzg
- Darstellung, welches Ziel unter welchen Vorraussetzungen mit welchen Mitteln erreicht werden kann
  - Regeln der Form: (Vorraussetzung) AKTION (Nachbedingung)
  - nicht aufgeführte Nachbedingungen bleiben so wie vor de Aktion
- MEA-Tabelle Stellt dar, welches Mittel für welchen Zweck geeignet ist
- Algorithmus: If Statrtzustand != Endzustand
  - bestimmte wichtigsten Unterschied zw. Start und Endzustand
  - wähle Operator der zur Überbrückung dieses Untersch. geeignet
  - prüfe ob von Startzust. Vorbedingungen erreichbar (rekusiver aufruf) und ob von Nachbedingungen Endzust. erreichbar (rekursiver aufruf)

Und - oder -Graphen

- manchmal müssen mehrere Teilprobleme gelöst werden, um eine Alternative umzusetzen
- Umsetzung von Regeln in Graphen
- Passe heuristische Suchverfahren an: bei "Und" werden die kosten beider teilbäume aufsumiert
- vorher nur KostenSchätzung bekannt, nach Durchsuchen eines Teilbaum: reiche richtige Kosten nach oben weiter
  - bei UND: SUMME der Söhne
  - bei ODER: MINIMUM der Söhne

Spielen

- Streichholz schachteln. .
- Wahl des "besten" Zuges
  - Zug mit bestem Ergebnis
    - Ergebnis aller möglichen Züge vorhersagen, oft unmöglich
  - heuristik (Schlage soviel wie mögl)
  - Lernen aus Erfahrung

Min-Max-Bäume

- von allen alternativen wenn A zieht, wird die maximal bewertete gewählt
  - Wenn B zieht, wird die minimal gewertete alternative gewählt
- Anzahl Züge die der Computer_vorhersagen muss: b^d
  - b=Suchbreite, alternativen
  - d=suchtiefe
- Bewertungsfunktion für Qualität des Spielzustandes
  - b1 für einen Spieler
  - b2 für andren
  - b1 = -b2 = b wird abwechselnd minimiert und maximiert
- Horizont-Effekt wenn Suchtiefe nicht ausreicht: wegen temporärer Verschlechterung der Bewertung des Spielzustand kurz vor dem Sieg
  - Praxis: dynamische Suchtiefe, z B. wenn s Spielwende abzeichnet: vergrößern

Optimierung_durch alpha-beta-strategie

- ALPHA-MAXimierender Spieler: UNTERE Schranke die bis jetzt gefunden wurde, sicher darunt bewertete Teilbäume werden nicht weiter durchsucht
  - alpha= größter Wert aller MAXVorfahren des MINknotens
  - in MINKnoten braucht, wenn unter Schranke alpha des drüberliegenden max knoten gelangt, nicht weiter gesucht werden
- BETA-MINimierender Spieler, OBERE Schranke, sicher darüb bewertete Teilbäume werden nicht weiter durchsucht
  - beta = kleinster Wert aller MINVorfahren des MAXKnoten
  - in MAXKnoten braucht, wenn ein Sohn über Schranke beta d drüberliegenden MINKnoten gelangt, nicht weitergesucht werden

Suchverfahren

- Problem
  - Zustandsmenge
  - Zielzustand
  - kosten
  - Anfangszust
  - Übergangs funktion
  - Repräsentation als Graph
  - Darstellung als Baumstruktur (Knoten_verdoppeln!)
    - Wenn jedoch ein Knoten expandiert wird, der schonmal Baum vorkommt, muss geprüft werden ob er ein Vorgänger "von sich selbst" ist -> ZYKEL -> ignorieren, nicht weitersuche
    - oder jede Kante die man geht markieren - kommt man zu einer schon markierten Kante -Zyklus -> unlösbar-gewicht für diesen weg
  - ZIEL: Aufbau des Suchbaumes zielorientiert gestalten, möglichs nur Teile des Pfades zum Endzust. aufbauen
- Bewertung:
  - Lsg gefunden
    - Zulässigkeit (Vollständigkeit)
  - Suchkosten
    - Zeitkomplexität
    - Speicherkomplexität
  - Pfadkosten
    - Optimalität
- Bsp Staubsauger(blind) : Starzust= alle Zustände, bei jedem übergang können einige Zust. ausgeschlossen werden b nur noch endzust. übrig

Tiefensuche

- zulässig, nicht optimal
  - viel zeit, backtracking
- viel viel speicher
  - zulässig, optima (kleinste zahl an übergängen)

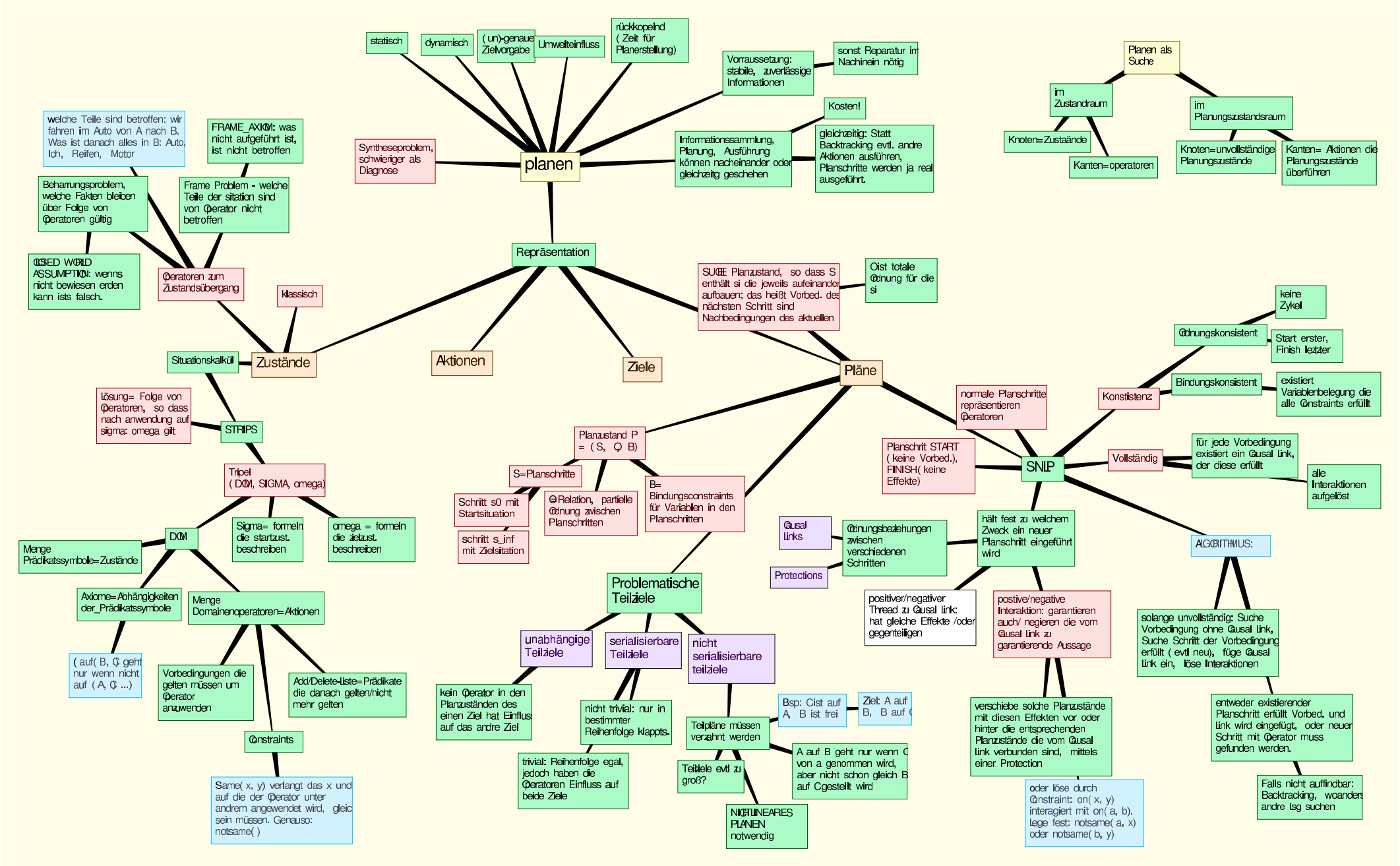
Breitensuche

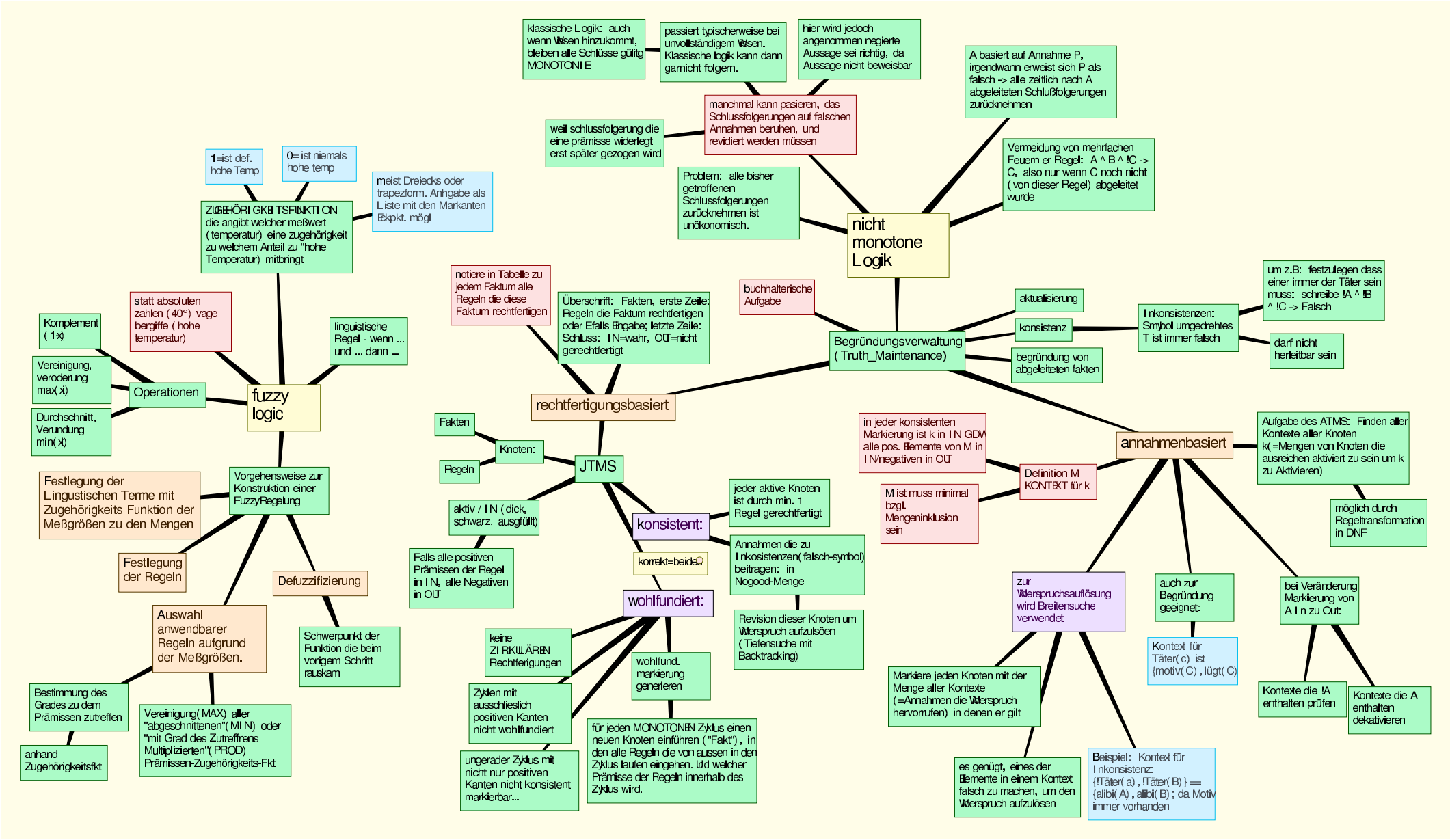
heuristische Suche

- Hill Climbing
  - "Beste Fortsetzung"; Tiefensuche + lokale Messungen
    - garnicht optimal, läuft immer auf grad, auch wenn der sich schlängelt
- Best First
  - Speichert Kosten aller bisher gefundenen Alternativen und nimmt von allen immer die beste
    - nicht optimal
- A*
  - falls alle Zust. übergänge gleich viel kosten und h(x) =0 => Breitensuche
  - Schätzfunktion für voraus. Kosten
    - soll monoton sein (<= tatsächliche Kosten)
  - wie Best First aber wähle Wg mit geringsten voraus. Gesamtkosten
    - optimal, falls Schätzfkt. monoton

zuordnungsprobleme

- Menge von Objekten denen Werte aus Bereich V zugewiesen werden sollen (und dabei best. Bedingungen erfüllt sein sollen)
  - bsp: färbeproblem, landkarte, benachbarte verschieden (binary constraints)
  - n-stellige Constraints
  - suchbaum hat tiefe |V|
  - ZustandsÜbergang: durch Belegung des Objektes m Wert; Reihenfolge d. Belegung egal
- Darstellung von binary constraints in Matrix oder Constraint-Graph
- knontenkonsistenz
  - jede Markierung erfüllt einstellige constraints
- kantenkonsistenz
  - jede Markierung lässt sich zu einer Markierung erweitern, sodass zweistellige Constraints erfüllt sind (Und Knotenkonsistent!)
  - kantenkonsistente Probleme Backtrackingfrei lösbar mit Algorithmus MWD
- vorgehensweise: durch das Belegen mit Werten beschränken wir die Möglichkeiten die andren Objekte mit Werten zu belegen
  - heuristik: wähle variable mit kleinstem (verbleibenden) Wertebereich
  - heuristik: wähle Var. mit meisten Beschränkungen (die in den meisten Constraints vorkommt)
    - so das möglichst viele Beschränkungsrelationen für die andren Werte "Feuern": Constraint Propagation
  - Bsp 4 Zahlen zuordnen, so das eine größer als andre
    - Suchraum vorher beschneiden (wertebereiche aufgrund constraints einschränken)
    - größte Variable die größer als andre sein soll laut constraint kann nicht mit 1 belegt werden
  - Beispiel n-Damen-Problem
    - Alternative, die dazu führt das für andren Wert keine Lsg vorhanden -> verwerfen
    - Wähle die Variable die nur einen möglichen Wert hat auf jeden Fall
    - Wert der Variablen: Minimaler Konflikt, Wertebereiche der andren Var. möglichst wenig einschränken


planen
statisch
dynamisch
(un)-genaue Zielvorgabe
Umwelteinfluss
rückkopelnd (Zeit für Planerstellung)
Vorraussetzung: stabile, zuverlässige Informationen
sonst Reparatur im Nachinein nötig
Informationssammlung, Planung, Ausführung können nacheinander oder gleichzeitig geschehen
Kosten!
gleichzeitig: Statt Backtracking evtl. andre Aktionen ausführen, Planschritte werden ja real ausgeführt.
Syntheseproblem, schwieriger als Diagnose
Planen als Suche
im Zustandraum
Knoten=Zustaände
Kanten=operatoren
im Planungszustandsraum
Knoten=unvollständige Planungszustände
Kanten= Aktionen die Planungszustände überführen
welche Teile sind betroffen: wir fahren im Auto von A nach B. Was ist danach alles in B: Auto, Ich, Reifen, Motor
FRAME_AXIOM: was nicht aufgeführt ist, ist nicht betroffen
Beharrungsproblem, welche Fakten bleiben über Folge von Operatoren gültig
Frame Problem - welche Teile der sitation sind von Operator nicht betroffen
CLOSED WORLD ASSUMPTION: wenns nicht bewiesen erden kann ists falsch.
Operatoren zum Zustandsübergang
Repräsentation
klassisch
Zustände
Situationskalkül
Aktionen
Ziele
Pläne
SUCHE Planzustand, so dass S enthält si die jeweils aufeinander aufbauen; das heißt Vorbed. des nächsten Schritt sind Nachbedingungen des aktuellen
O ist totale Ordnung für die si
Lösung= Folge von Operatoren, so dass nach anwendung auf sigma: omega gilt
STRIPS
Tripel (DOM, SIGMA, omega)
DOM
Menge Prädikatssymbole=Zustände
Sigma= formeln die startzust. beschreiben
omega = formeln die zielzust. beschreiben
Axiome=Abhängigkeiten der_Prädikatssymbole
Menge Domainenoperatoren=Aktionen
(auf(B, C) geht nur wenn nicht auf (A, C) ...)
Vorbedingungen die gelten müssen um Operator anzuwenden
Add/Delete-Liste=Prädikate die danach gelten/nicht mehr gelten
Constraints
Same(x, y) verlangt das x und auf die der Operator unter andrem angewendet wird, gleic sein müssen. Genauso: notsame( )
Planzustand P = (S, O, B)
S=Planschritte
Schritt s0 mit Startsituation
schritt s_inf mit Zielsitation
O=Relation, partielle Ordnung zwischen Planschritten
B= Bindungsconstraints für Variablen in den Planschritten
Problematische Teilziele
unabhängige Teilziele
kein Operator in den Planzuständen des einen Ziel hat Einfluss auf das andre Ziel
serialisierbare Teilziele
nicht trivial: nur in bestimmter Reihenfolge klappts.
trivial: Reihenfolge egal, jedoch haben die Operatoren Einfluss auf beide Ziele
nicht serialisierbare teilziele
Teilpläne müssen verzahnt werden
Bsp: C ist auf A, B ist frei
Ziel: A auf B, B auf C
A auf B geht nur wenn C von a genommen wird, aber nicht schon gleich B auf C gestellt wird
Teilziele evtl zu groß?
NICHTLINEARES PLANEN notwendig
normale Planschritte repräsentieren Operatoren
Planschrit START (keine Vorbed.), FINISH(keine Effekte)
SNLP
Konstistenz
Ordnungskonsistent
keine Zykel
Start erster, Finish letzter
Bindungskonsistent
existiert Variablenbelegung die alle Constraints erfüllt
Vollständig
für jede Vorbedingung existiert ein Causal link, der diese erfüllt
alle Interaktionen aufgelöst
Causal links
Ordnungsbeziehungen zwischen verschiedenen Schritten
Protections
hält fest zu welchem Zweck ein neuer Planschritt eingeführt wird
positiver/negativer Thread zu Causal link: hat gleiche Effekte /oder gegenteiligen
postive/negative Interaktion: garantieren auch/ negieren die vom Causal Link zu garantierende Aussage
verschiebe solche Planzustände mit diesen Effekten vor oder hinter die entsprechenden Planzustände die vom Causal link verbunden sind, mittels einer Protection
oder löse durch Constraint: on(x, y) interagiert mit on(a, b). lege fest: notsame(a, x) oder notsame(b, y)
ALGORITHMUS:
solange unvollständig: Suche Vorbedingung ohne Causal link, Suche Schritt der Vorbedingung erfüllt (evtl neu), füge Causal link ein, löse Interaktionen
entweder existierender Planschritt erfüllt Vorbed. und link wird eingefügt, oder neuer Schritt mit Operator muss gefunden werden.
Falls nicht auffindbar: Backtracking, woanders andre lsg suchen


fuzzy logic
Operationen
Komplement (1-x)
Vereinigung, veroderung max(xi)
Durchschnitt, Verundung min(xi)
statt absoluten zahlen (40°) vage bergiffe (hohe temperatur)
ZUGEHÖRIGKEITSFUNKTION die angibt welcher meßwert (temperatur) eine zugehörigkeit zu welchem Anteil zu "hohe Temperatur) mitbringt
1=ist def. hohe Temp
0= ist niemals hohe temp
meist Dreiecks oder trapezform. Anhgabe als Liste mit den Markanten Ekpkt. mögl
linguistische Regel - wenn ... und ... dann ...
Vorgehensweise zur Konstruktion einer FuzzyRegelung
Festlegung der Lingustischen Terme mit Zugehörigkeits Funktion der Meßgrößen zu den Mengen
Festlegung der Regeln
Auswahl anwendbarer Regeln aufgrund der Meßgrößen.
Bestimmung des Grades zu dem Prämissen zutreffen
anhand Zugehörigkeitsfkt
Vereinigung(MAX) aller "abgeschnittenen"(MIN) oder "mit Grad des Zutreffrens Multiplizierten"(PROD) Prämissen-Zugehörigkeits-Fkt
Defuzzifizierung
Schwerpunkt der Funktion die beim vorigem Schritt rauskam
nicht monotone Logik
Klassische Logik: auch wenn Wissen hinzukommt, bleiben alle Schlüsse gültig MONOTONIE
passiert typischerweise bei unvollständigem Wissen. Klassische logik kann dann garnicht folgern.
hier wird jedoch angenommen negierte Aussage sei richtig, da Aussage nicht beweisbar
manchmal kann pasieren, das Schlussfolgerungen auf falschen Annahmen beruhen, und revidiert werden müssen
weil schlussfolgerung die eine prämisse widerlegt erst später gezogen wird
Problem: alle bisher getroffenen Schlussfolgerungen zurücknehmen ist unökonomisch.
A basiert auf Annahme P, irgendwann erweist sich P als falsch -> alle zeitlich nach A abgeleiteten Schlußfolgerungen zurücknehmen
Vermeidung von mehrfachen Feuern er Regel: A ^ B ^ !C -> C, also nur wenn C noch nicht (von dieser Regel) abgeleitet wurde
Begründungsverwaltung (Truth_Maintenance)
buchhalterische Aufgabe
aktualisierung
konsistenz
Inkonsistenzen: Smybol umgedrehtes T ist immer falsch
um z.B: festzulegen dass einer immer der Täter sein muss: schreibe !A ^ !B ^ !C -> Falsch
darf nicht herleitbar sein
begründung von abgeleiteten fakten
rechtfertigungsbasiert
notiere in Tabelle zu jedem Faktum alle Regeln die diese Faktum rechtfertigen
Überschrift: Fakten, erste Zeile: Regeln die Faktum rechtfertigen oder Efalls Eingabe; letzte Zeile: Schluss: IN=wahr, OUT=nicht gerechtfertigt
JTMS
Knoten:
Fakten
Regeln
aktiv / IN (dick, schwarz, ausgfüllt)
Falls alle positiven Prämissen der Regel in IN, alle Negativen in OUT
konsistent:
jeder aktive Knoten ist durch min. 1 Regel gerechtfertigt
Annahmen die zu Inkosistenzen(falsch-symbol) beitragen: in Nogood-Menge
Revision dieser Knoten um Widerspruch aufzulsöen (Tiefensuche mit Backtracking)
korrekt=beide☺
wohlfundiert:
keine ZIRKULÄREN Rechtferigungen
Zyklen mit ausschlieslich positiven Kanten nicht wohlfundiert
ungerader Zyklus mit nicht nur positiven Kanten nicht konsistent markierbar...
wohlfund. markierung generieren
für jeden MONOTONEN Zyklus einen neuen Knoten einführen ("Fakt"), in den alle Regeln die von aussen in den Zyklus laufen eingehen. Und welcher Prämisse der Regeln innerhalb des Zyklus wird.
annahmenbasiert
Aufgabe des ATMS: Finden aller Kontexte aller Knoten k(=Mengen von Knoten die ausreichen aktiviert zu sein um k zu Aktivieren)
möglich durch Regeltransformation in DNF
Definition M KONTEXT für k
in jeder konsistenten Markierung ist k in IN GDW alle pos. Elemente von M in IN/negativen in OUT
M ist muss minimal bzgl. Mengeninklusion sein
zur Widerspruchsauflösung wird Breitensuche verwendet
Markiere jeden Knoten mit der Menge aller Kontexte (=Annahmen die Widerspruch hervorrufen) in denen er gilt
es genügt, eines der Elemente in einem Kontext falsch zu machen, um den Widerspruch aufzulösen
Beispiel: Kontext für Inkonsistenz: {!Täter(a), !Täter(B)} == {alibi(A), alibi(B); da Motiv immer vorhanden
auch zur Begründung geeignet:
Kontext für Täter(c) ist {motiv(C), lügt(C)
bei Veränderung Markierung von A I n zu Out:
Kontexte die !A enthalten prüfen
Kontexte die A enthalten dekativieren

Entscheidungsbäume

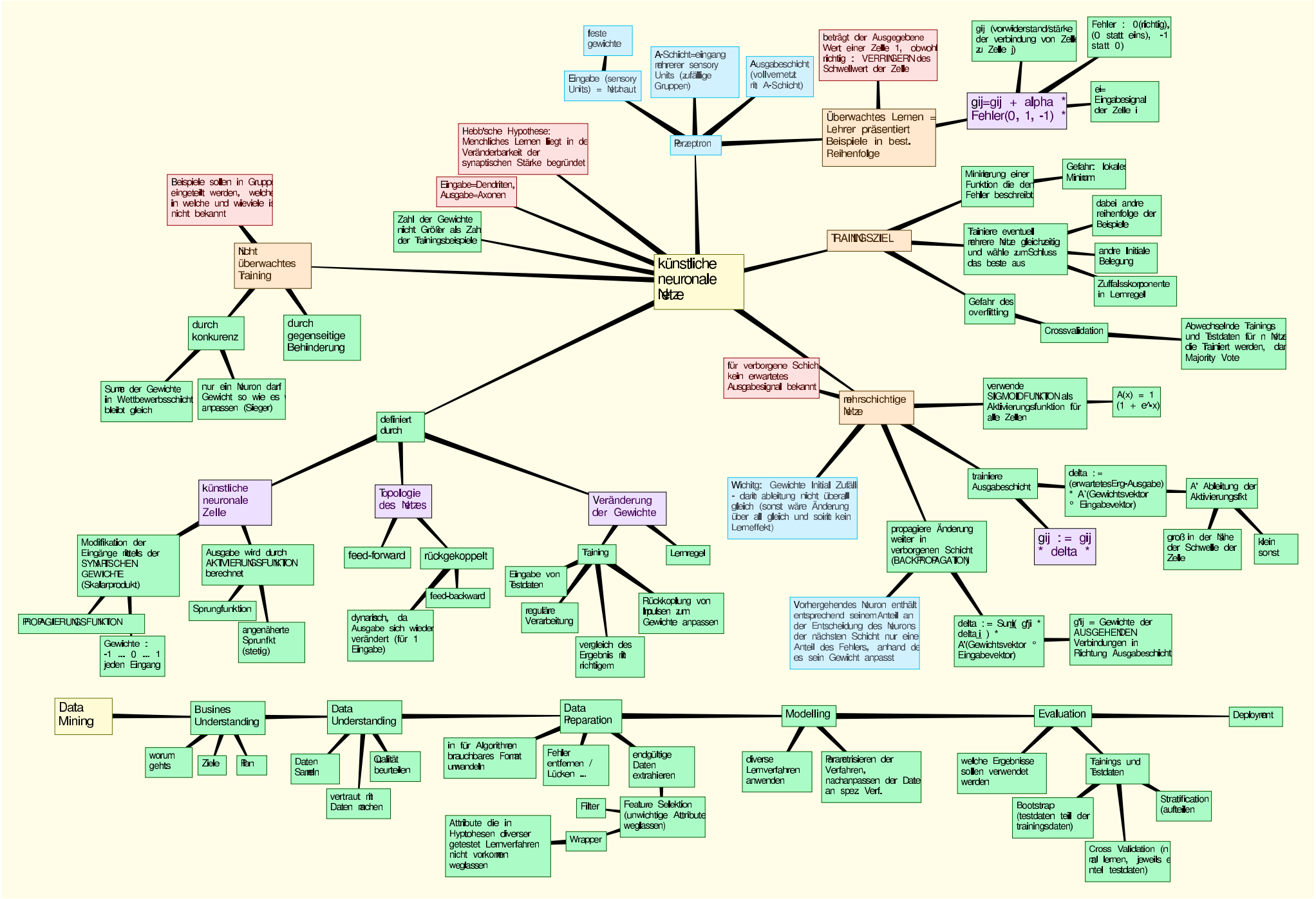
- Definition: -jeder innere Knoten ist mit Attribut Ai markiert und hat für jedes a in Val(Ai) eine mit a markierte ausgehende Kante; -jedes Blatt ist mit einem Wert aus Val(B) markiert
  - nicht nominale, numerische Werte:
    - Betrachte nur Werte die: in Beispielen vorkommen
      - w o das Beispiel mit nächsthöherem/ kleinerem Wert anders klassifiziert ist
        - von den verbleibenden das mit maximalen Gain für größer/kleiner Test auswählen
    - zusätzliche Tests: größer/ kleiner
- BspDaten: Attribute(Ai), Klassifizierung(B)
  - nomiales Attribut (Val(A) endlich)
  - klassifiziertes Bsp= tupel (a1,...,an,b)
- Eigenschaften
  - Jede boolsche Fkt kann durch Entsch. baum repräsentiert werden
  - Es gibt nicht immer einfache Entscheidungsbäume
  - Es gibt nicht immer kleine Entscheidungsbäume
- Lernaufgabe: finde kleinsten Entscheidungsbaum T für Beispiele S
  - GREEDY: Alle s in S gleich Klassifiziert -> Blatt; Sonst wähle geignetes Attribut und Teile S entsprechend Attribut auf und fahre rekursiv fort
    - geeignet?
      - So, das nachfolgende Teilbäume möglichst klein bleiben
      - Attribut, das am meisten Information liefert wählen!
        - maximiere GAIN(S,A) = info(A unbek) - info(A bek)
        - Gain(S,A) = H(S) - a1*H(A1) - ... - an*H(An)
          - ai anteilige Größe

- Entropie und Info
  - je inwahrscheinlicher das Auftreten eines Zeichens ist, desto mehr information erhält man durch sein Auftreten
    - abhängig von Auftritswahrsch. p
    - info(p)>=0
    - stetig
    - info(pq)=info(p)+info(q)
  - info(p)=log(1/p) in BIT
  - Entropie= Durchschnittsinformation von k Symbolen
    - H(p1..pk) = p1*info(p1)+... +pk*info(pk)
      - H(S) = H(P(s in S=ja) , P(s in S=nein))

- Patternsprachen
  - beliebige Pattern
    - algorithmisch unbeherschbar, w in L(p)? NP-vollst; L(p)teilm.L(q) unlösbar
  - reguläre Pattern
    - Probleme schnell lösbar
    - keine Variable mehrfach

Erweiterte Ansätze:

- Vorverarbeitung
  - Alphabetreduktion
    - lernen von Pattern-Sprachen
    - A Alphabet, P,N teilmenge A* =postitive/negative Bsp; phi:A->B, es gilt phi(P) geschnitten phi(N) ist leer
      - bestimmung von phi aus A,B,P,N ist NP-vollständig
- Pruning
  - postpruning
    - Modifikation des vollständigen Baumes im Nachhinein
    - Varante 1: direkt Baum
      - für jeden Knoten k
        - Welche Klassifizierung bk haben die meisten Testdaten die k erreichen
          - Ist ACCURACY besser oder genausogut wenn man k mit Blatt bk ersetzt
    - Variante2: in Regeln umwandeln und Regeln modifiziere
      - für jedes Blatt eine Regel (if x1==1 && x2==0 &&...)
        - vereinfachung der Regeln, für jede vereinfachung ACCURACY berechnen
          - Klassifikation durch Majority-Vote der Regeln deren Vorbedinungen von konkretem Bsp. erfüllt sind
  - prepruning
    - schrittweise überprüfung mit Testdaten während Konstruktion des Baums
  - vermeidung von zu speziellen Hypothesen
    - fehlerhafte Bsp
    - unrepräsentative bsp
    - verhält sich auf Trainigsdaten besser als auf Testdaten
      - Maß: accuracy
- Attribut-Auswahl
  - Attribute die sehr viele Werte annhemen können zerlegen in viele kleine Mengen, hoher Gain aber kleine Mengen nicht repräsentativ
    - grad der Aufteilung: splitinfo = H(|S1|/S,...,|Sn|/S)
      - hohe Werte, wenn viele gleich häufige Ausprädungen
    - Maximiere gainratio=gain/splitinfo


künstliche neuronale Netze
Perzeptron
feste gewichte
Eingabe (sensory Units) = Netzhaut
A-Schicht=eingang mehrerer sensory Units (zufällige Gruppen)
Ausgabeschicht (vollvernetzt mit A-Schicht)
Überwachtes Lernen = Lehrer präsentiert Beispiele in best. Reihenfolge
beträgt der Ausgegebene Wert einer Zelle 1, obwohl richtig : VERRINGERN des Schwellwert der Zelle
gij=gij + alpha * Fehler(0, 1, -1) *
gij (vorwiderstand/stärke der verbindung von Zelle i zu Zelle j)
Fehler : 0(richtig), (0 statt eins), -1 statt 0)
ei= Eingabesignal der Zelle i
Hebb'sche Hypothese: Menchliches Lernen liegt in der Veränderbarkeit der synaptischen Stärke begründet
Eingabe=Dendriten, Ausgabe=Axonen
Zahl der Gewichte nicht Größer als Zahl der Trainingsbeispiele
Beispiele sollen in Gruppen eingeteilt werden, welche in welche und wieviele ist nicht bekannt
Nicht überwachtes Training
durch konkurenz
durch gegenseitige Behinderung
Summe der Gewichte in Wettbewerbsschicht bleibt gleich
nur ein Neuron darf Gewicht so wie es anpassen (Sieger)
TRAINGSZIEL
Minimierung einer Funktion die den Fehler beschreibt
Gefahr: lokales Minimum
Trainiere eventuell mehrere Netze gleichzeitig und wähle zum Schluss das beste aus
dabei andre reihenfolge der Beispiele
andre Initiale Belegung
Zuffalsskorponente in Lernregel
Gefahr des overfitting
Crossvalidation
Abwechselnde Trainings und Testdaten für n Netze die Trainiert werden, dann Majority Vote
für verborgene Schichten kein erwartetes Ausgabesignal bekannt
mehrschichtige Netze
verwende SIGMOIDFUNKTION als Aktivierungsfunktion für alle Zellen
A(x) = 1 (1 + e^-x)
Wichtig: Gewichte Initiall Zufällig - darit ableitung nicht überall gleich (sonst wäre Änderung über all gleich und somit kein Lerneffekt)
trainiere Ausgabeschicht
delta := (erwartetesErg-Ausgabe) * A'(Gewichtsvektor ° Eingabevektor)
A' Ableitung der Aktivierungsfkt
groß in der Nähe der Schwelle der Zelle
klein sonst
gij := gij * delta *
propagiere Änderung weiter in verborgenen Schicht (BACKPROPAGATION)
Vorhergehendes Neuron enthält entsprechend seinem Anteil an der Entscheidung des Neurons der nächsten Schicht nur eine Anteil des Fehlers, anhand des es sein Gewicht anpasst
delta := Sum(gji * delta_i) * A'(Gewichtsvektor ° Eingabevektor)
gij = Gewichte der AUSGEHENDEN Verbindungen in Richtung Ausgabeschicht
definiert durch
künstliche neuronale Zelle
Modifikation der Eingänge mittels der SYNAPTISCHEN GEWICHTE (Skalarprodukt)
PROPAGIERUNGSFUNKTION
Gewichte : -1 ... 0 ... 1 jeden Eingang
Ausgabe wird durch AKTIVIERUNGSFUNKTION berechnet
Sprungfunktion
angenäherte Sprunfkt (stetig)
Topologie des Netzes
feed-forward
rückgekoppelt
feed-backward
dynamisch, da Ausgabe sich wieder verändert (für 1 Eingabe)
Veränderung der Gewichte
Training
Eingabe von Testdaten
reguläre Verarbeitung
vergleich des Ergebnis mit richtigem
Rückkopplung von Impulsen zum Gewichte anpassen
Lernregel
Data Mining
Busines Understanding
worum gehts
Ziele
Plan
Data Understanding
Daten Sammeln
Qualität beurteilen
vertraut mit Daten machen
Data Preparation
in für Algorithmen brauchbares Format umwandeln
Fehler entfernen / Lücken ...
endgültige Daten extrahieren
Feature Selektion (unwichtige Attribute weglassen)
Filter
Wrapper
Attribute die in Hyptohesen diverser getestet Lernverfahren nicht vorkommen weglassen
Modelling
diverse Lernverfahren anwenden
Parametrisieren der Verfahren, nachanpassen der Daten an spez. Verf.
Evaluation
welche Ergebnisse sollen verwendet werden
Trainings und Testdaten
Bootstrap (testdaten teil der trainingsdaten)
Stratification (aufteilen)
Cross Validation (normal lernen, jeweils ein Anteil testdaten)
Deployment