Was bisher geschah

- KI-geschichte
- KI-Tests (Turing, Chinese Room)
- schwache / starke KI
- Daten, Information, Wissen, Intelligenz
- explizites und implizites Wissen

Wissen

- Was ist Wissen?
- Wie lässt es sich darstellen?
- Wie lässt es sich nutzen, um Probleme zu lösen?
- Wie lässt es sich erweitern / ändern?

Analogie zu Wissen von Experten auf einem Fachgebiet

Darstellung von Wissen

formale Repräsentation des Wissens in einer Wissensbasis:
spezielle Form der Daten in der Wissensbasis abhängig von

- Problembereich
- geplante Verwendung

Wissen in Wissensbasis ist immer Abstraktion, beschreibt Modelle der Realität

- Auswahl von (für den Anwendungsbereich) wichtigem Wissen
- Vernachlässigung unwichtiger Details

Beispiele:

- Liniennetzplan
- Grundriss
- Stundenplan
- Kostenplan

Wissensverarbeitung

- Problemlösen
  - algorithmische Suche in Zustandsräumen
  - logisches Schließen

  Beispiel: n-Damen-Problem, kürzeste Wege in Graphen
- Planen

  Finden einer Folge von Aktionen zum Erreichen eines Zieles
  Beispiel: morgens Anziehen, Fertigungsroboter
- Klassifikation

  Finden von Klassen (Diagnosen) anhand der Merkmalswerte (Symptome)
  Beispiel: Fahrzeuge, Fehlfunktionen

teilweise bekannt aus den Modulen

- Modellierung
- Algorithmen und Datenstrukturen

Anforderungen an Wissensbasen

Qualitätskriterien bei der Modellierung:

- für Problembereich geeignete Abstraktion
- effektiv, redundanzfrei
- vollständig
- erweiterbar
- verständlich

Beispiele für Wissensrepräsentation und Problemlösen

Suche / Planen:

- Kontext: Zustandsübergangssystem
- Aufgabe: Startzustand und Anforderungen an Zielzustände
- Lösung: Zielzustand (und evtl. Pfad dorthin)

Lösungsverfahren: Suche (vollständig oder heuristisch)

Logisches Schließen:

- Kontext: Menge logischer Formeln
- Aufgabe: Gilt die Behauptung (logische Formel) im Kontext?
- Lösung: ja / nein (evtl. mit Begründung)

Lösungsverfahren: logisches Folgern oder Schließen

Statistische Klassifikation:

- Kontext: klassifizierte Datenmenge (bekannte Fälle)
- Aufgabe: neuer Fall
- Lösung: Klassifikation (Zuordung zu einer Klasse)

Lösungsverfahren: statistische Verfahren (z.B. trainiertes KNN)

Programmierung und Wissensverarbeitung

Programmierung	Wissensverarbeitung
Entwurf eines Algorithmus zur Lösung des Problemes	Identifikation des zur Lösung des Problemes relevanten Wissens
Implementierung in einer geeigneten Programmiersprache	Darstellung des relevanten Wissens in einer geeigneten Repräsentationssprache
Problemlösung durch Ausführung des Programmes	Problemlösung durch Anwendung eines Standardverfahrens

Beispiel: n-Damen-Problem

Aufgabe: Setze n Damen ohne gegenseitige Bedrohungen auf ein $n \times n$-Spielfeld

Programmierung	Wissensrepräsentation
Entwurf geigneter Datenstrukturen und eines Algorithmus zur Lösungssuche	Identifikation der Bedingungen an Aufgabe und Lösung
Implementierung	Repräsentation von Spielfeld und Bedingungen an eine Lösung als logische Formeln (z.B. CNF)
Problemlösung durch Ausführung des Programmes	Problemlösung durch logisches Inferenzverfahren (z.B. Resolution, SAT-Solver, Prolog)

Programmierung und Wissensverarbeitung

Programmieren	Wissensverarbeitung

Erklärung der Lösung:

Verfolgen der Zustandsänderung bei Programmausführung (Debugging)	vom Inferenzverfahren verwendete Voraussetzungen

Fehlerbehandlung:

Debugging	fehlendes Wissen einfügen
Codeänderung	falsches Wissen löschen

Wissenserweiterung:

neuer Entwurf, Neuimplementierung	neues Wissen in Wissensbasis einfügen

Wissensverarbeitung

Teilaufgaben:

Repräsentation des Wissens

- geeignete Abstraktionsgrade
- Sprachen zur Wissensrepräsentation
- Modellierung

Verarbeitung des Wissens

- Erweiterung vorhandenen Wissens
- Herleitung neuen Wissens
- Verträglichkeit neuen Wissens mit vorhandenem

Anwendung des Wissens, z.B. zum

- Problemlösen
- Erklären
- Planen
- Klassifikation
- Diagnose
- Entscheidungsunterstützung

Beispiele wissensverarbeitender Systeme

- Expertensysteme
- Diagnosesysteme
- Schach- und andere Spielprogramme
- Datenanalyse
- Suchmaschinen
- Maschinelle Erkennung und Verarbeitung natürlicher Sprache
- Bild- und Zeichenerkennung (Klassifikation)
- Objekterkennung in digitalen Bildern
- Planungssysteme
- Steuerung autonomer Agenten,
  z.B. für Transport, Information, Unterhaltung, Rettung, Putzen

Problemlösung durch Suche in Graphen – Beispiele

- Finden von Wegen in einem Graphen
  - Aufgabe:
    - gegeben: Graph G (Tafel)
    - gesucht: Weg (Pfad) in G von Knoten u zu Knoten v
  - Lösungsidee: Suche im Graphen
- Münzenstapelspiel (für eine Person)
  - Aufgabe:
    - gegeben: Stapel von n Münzen
    - gesucht: Zugfolge durch erlaubte Züge (zwei Münzen von einem Stapel nehmen und auf beide Nachbarn verteilen) bis zu einer Situation, in der kein Zug möglich ist
  - Lösungsidee:
    - Modellierung als Zustandsübergangssystem
    - Suche im Graphen
- 3 Krüge
  - Aufgabe:
    - gegeben: 3 volle Krüge mit Volumen 4l, 7l, 9l,
    - gesucht: genau 6l in einem der 3 Krüge
  - Lösungsidee: Zustände als Knoten eines Suchbaumes

Darstellung von Aufgabe und Lösung

Aufgabe:

gegeben:
- Menge V von Zuständen (evtl. unendlich) oft beschrieben durch Eigenschaften
- Startzustand $s \in V$
- Menge $Z \subseteq V$ von Zielzuständen (oder Eigenschaften der Zielzustände)
- mögliche Übergänge zwischen Zuständen Übergangsrelation $E \subseteq V \times V$

Lösung: Folge von Zuständen (Weg von einem Start- zu einem Zielzustand) (Mitunter interessiert nur der erreichte Zielzustand.)

Wissensrepräsentation: als Graph $G = (V, E)$ (Zustandsübergangssystem):
- Knotenmenge V: Zustände
- (gerichtete) Kanten: Zustandsübergänge

Entfaltung des Graphen zu einem Baum:
Pfade im Graphen = Knoten im Baum

Problemlösen durch Suchen

- formale Darstellung des Problemes
  als Graph (z.B. Baum, DAG)
- formale Beschreibung der Lösung als Eigenschaft von
  - Pfaden im Graphen
  - Knoten im Baum

Möglichkeiten zum Problemlösen:

- Pfadsuche im Graphen
- Knotensuche im Baum

Suche in Graphen

(schon bekannte) Verfahren zur Suche in Graphen (und Bäumen):

- Tiefensuche (depth-first search):
  Suche zuerst in Teilbäumen eines noch nicht besuchten Nachbarn des aktuellen Knotens
- Breitensuche (breadth-first search):
  Suche zuerst in Teilbäumen eines noch nicht besuchten Knotens mit der geringsten Tiefe

Allgemeines Suchverfahren

Daten:
- L_a Menge der noch zu expandierenden Knoten
- L_x Menge der expandierten Knoten
- s Startknoten
- φ Anforderungen an Lösung (Zielknoten)

Allgemeiner Suchalgorithmus:

1. $L_a = \{s\}, L_x = \emptyset$
2. solange $\neg\, L_a = \emptyset$:
   - 2.1 Verschiebe einen auf festgelegte Art ausgewählten Knoten u aus L_a in L_x
   - 2.2 Füge alle Nachbarn von u, die nicht in $L_a \cup L_x$ enthalten sind, auf eine festgelegte Art in L_a ein
     (Abbruch falls ein Nachbar v von u die Bedingung φ erfüllt, also eine Lösung repräsentiert)

prominente Spezialfälle:

Tiefensuche
- Verwaltung von L_a als Stack
- Einfügen der Nachbarn an den Anfang der Liste L_a
- festgelegter Knoten wurde zuletzt in L_a eingefügt

Breitensuche
- Verwaltung von L_a als Queue
- Einfügen der Nachbarn an das Ende der Liste L_a

Schrittweise Vertiefung

beschränkte Tiefensuche:

1. festgelegte Tiefenbeschränkung $m \in \mathbb{N}$
2. Tiefensuche auf allen Pfaden bis zur Tiefe m

nicht vollständig, weiter entfernte Lösungen werden nicht gefunden

Schrittweise Vertiefung(iterative deepening)
Kombination aus Breiten- und Tiefensuche durch Nacheinanderausführung der beschränkten Tiefensuche für alle $m \in \mathbb{N}$, solange keine Lösung gefunden wurde

vollständig, optimal
(asymptotischer) Zeit- und Platzbedarf wie Tiefensuche

Gleiche-Kosten-Suche (kleinste bisherige Kosten)

(uniform-cost-search)

bei Zustandsübergängen mit verschiedenen Kosten

Ziel: Lösung (Pfad vom Start- zu einem Lösungsknoten) mit möglichst geringen Pfadkosten
(Pfadkosten = Summe der Kosten aller Übergänge auf dem Pfad)

Bewertungsfunktion für Knoten $k : V \to \mathbb{R}_{\geq 0}$

$k(u)$ = minimale (bisher entdeckte) Pfadkosten vom Startknoten zu u

Datenstruktur zur Verwaltung von L_a: Priority Queue
Priorität eines Knotens u: $k(u)$

Beispiele:

- Breitensuche (Kosten = Tiefe des aktuellen Knotens u)
- kürzeste Wege (Kosten = minimale bisher bekannte Kosten vom Startknoten zum aktuellen Knoten u)
  Dijkstra-Algorithmus