**Notizen zu Künstlicher Intelligenz, Prolog, Rekursion, Listen, Graphen und OOP**

**1. Künstliche Intelligenz (KI)**

**1.1 Definitionen**

* Sie bezeichnen KI als ein System, das seine Umgebung wahrnehmen und Maßnahmen ergreifen kann, um die Chance zu maximieren, Ziele erfolgreich zu erreichen. Darüber hinaus ist dieses System in der Lage, Daten so zu interpretieren und zu analysieren, dass es lernt und sein Verhalten entsprechend anpasst.

**1.2 Turing-Test**

Ein Test, bei dem eine Maschine als intelligent gilt, wenn ein menschlicher Prüfer sie nicht von einem echten Menschen unterscheiden kann.

**1.3 Drei konstruktive Merkmale der Intelligenz**

1. **Lernen** aus Erfahrungen
2. **Problemlösung** durch Logik
3. **Anpassungsfähigkeit** an neue Situationen

**1.4 Vier Anwendungsgebiete der KI**

* **Expertensysteme**
* **Frage-Antwort-System**
* **Systeme zum maschinellen Lernen**
* **Roboter**
* **Tutorielle Systeme**

**1.5 Ethische Fragen zur KI**

* Verantwortung für KI-Entscheidungen
* Datenschutz und Überwachung
* Arbeitsplatzverlust durch Automatisierung
* Diskriminierung durch voreingenommene Algorithmen

**2. Definitionen**

**2.1 Term**

=> einzige in Prolog vorkommende Datenstruktur

ein Term ist ein Atom

*==Info.==*

Oder: Atom gefolgt von weiteren Termen, durch Kommata getrennt und in Klammern

eingeschlossen

*==male(simon).==*

*==father(simon, timo).==*

**2.2 Liste**

Puhh kein plan mehr

**2.3 Stelligkeit von Prädikaten**

Die Anzahl der Argumente eines Prädikats (bspw.: vater(X, Y) ist zweistellig).

**2.4 Closed-World-Assumption**

Die Annahme, dass alles, was nicht explizit in der Wissensbasis steht, falsch ist.

**3. Rekursion**

**3.1 Definition**

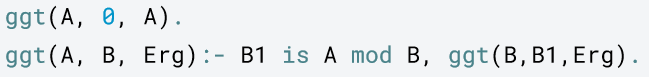
Eine Methode, bei der sich eine Funktion selbst aufruft.

**3.2 Voraussetzungen**

1. **Abbruchbedingung:** Beendet die Rekursion.
2. **Rekursiver Aufruf:** Zerlegt das Problem in kleinere Teile.

**Ein Bild, das Text, Schrift, Screenshot, Reihe enthält.

KI-generierte Inhalte können fehlerhaft sein.3.3 Rekursive Algorithmen (Beispiele)**

* **Addition**
* **ggT**
* **Fibonacci:** fib(0, 0). fib(1, 1). fib(N, R) :- N1 is N - 1, N2 is N - 2, fib(N1, R1), fib(N2, R2), R is R1 + R2.

**3.4 Nicht-mathematische Beispiele**

* **Vorfahr:**
  + vorfahr(X, Y) :- elternteil(X, Y).
  + vorfahr(X, Y) :- elternteil(X, Z), vorfahr(Z, Y).
* **Nachbar:**
  + nachbar(X, Y) :- links(X, Y).
  + nachbar(X, Y) :- rechts(X, Y).

**3.5 Darstellung mittels Treppe und Baum**

* **Treppe:** Zeigt iterative Reduktion.
* **Baum:** Veranschaulicht rekursive Verzweigung.

**4. Listen und Rekursion**

**4.1 Grundlegende Operationen**

* **Länge:** length([], 0). length([\_|R], N) :- length(R, N1), N is N1 + 1.
* **Summe:** sum([], 0). sum([H|T], S) :- sum(T, S1), S is H + S1.
* **Durchschnitt:** Summe/Länge kombinieren.
* **Letztes Element entfernen:**
  + remove\_last([\_], []).
  + remove\_last([H|T], [H|T1]) :- remove\_last(T, T1).

**4.2 Vordefinierte Prädikate**

* **member/2**: Prüft, ob ein Element in einer Liste ist.
* **length/2**: Bestimmt die Anzahl der Elemente.
* **append/3**: Hängt zwei Listen zusammen.
* **nth1/3**: Greift auf ein Element mit Index zu.
* **sort, msort**: Sortiert eine Liste.
* **flatten**: Entfernt Verschachtelung.
* **findall/3**: Sammelt alle Lösungen eines Prädikats.
* **is\_set/1, list\_to\_set/2**: Entfernt Duplikate.
* **intersection/3, subtract/3, union/3, subset/3**: Mengenoperationen.

**5. Graphen**

**5.1 Eingabe als Wissensbasis**

* Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

  KI-generierte Inhalte können fehlerhaft sein.Beispiel: kante(a, b). kante(b, c). kante(c, d).

**5.2 Suchalgorithmen**

* **Tiefensuche:**



* Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

  KI-generierte Inhalte können fehlerhaft sein.**Breitensuche:**



**6. DEA mit Prolog**

* (So wie Graphen)

**7. Objektorientierte Programmierung (OOP) (IN BEARBEITUNG)**

**7.1 Grundkonzepte**

* **Klasse:** Eine Klasse beschreibt die Variablen, Eigenschaften, Prozeduren und Ereignisse eines Objekts. Abstraktion von unwichtigen Informationen.
* **Objekt:** werden durch Klassen erzeugt, sie repräsentieren individuelle Instanzen einer Klasse **Vererbung:** Ableiten einer Klasse von einer anderen.
* **Polymorphismus:** Methoden können unterschiedliche Implementierungen haben.

**7.2 Klassendefinition**

class Auto {

private:

String marke;

int baujahr;

public:

Auto(String marke, int baujahr) {

this.marke = marke;

this.baujahr = baujahr;

}

}

**7.3 Objekt erstellen, manipulieren, ausgeben**

Auto meinAuto = new Auto("BMW", 2020);

Cout<<(meinAuto.marke);

**7.4 Konstruktoren und Destruktoren**

* **Konstruktor:** Initialisiert ein Objekt.
* **Destruktor:** Gibt Ressourcen frei

**7.5 Getter & Setter**

class Person {

private:

String name;

Public:

String getName() { return name; }

void setName(String name) { this.name = name; }

}