Künstliche Intelligenz 1 - HS19

Pascal Brunner - brunnpa7

Inhaltsverzeichnis

1 Was ist AI?

AI ist die Möglichkeit wie man Maschine beibringen kann, ähnlich wie ein Mensch zu denken und zu handeln. Dabei unterscheidet man zwischen "menschlichemünd "rationalem". Im Wesentlichen unterscheidet man *denken* und *handeln*. Wobei das rationale eine Mischung aus Mathematik und Engineering ist. **mögliche Definition AI:** Ist die Diziplin für das Lösung von komplexen Problem mit dem Computer. komplexes Problem → konnte bis anhin nur der Mensch lösen

1.1 Lernziele

- tbd

1.2 Menschliches Handeln: Der Ansatz des Turing-Test

Der **Turing Test** wurde entwickelt, um eine zufriedenstellende betriebliche Definition von Intelligenz zu liefern. Dabei besteht ein Computer den Turing Test, wenn man die Antwort einer Person, nicht von einem Computer unterscheiden kann. Damit ein Computer diesen Test absolvieren kann, braucht er unterschiedliche Möglichkeiten:

- **natural language processing** um eine erfolgreiche Kommunikation auf Englisch zu führen
- **knowledge representation** um neues Wissen oder neu gehörtes abzuspeichern
- **automated reasoning** um bereits vorhandenes Wissen für die Beantwortung von Fragen zur verwenden oder neue Erkenntnisse zu gewinnen
- **machine learning** um sich an neue Gegebenheiten anzupassen und Muster zu erkennen und zu erweitern

Damit ein Computer den **total Turing Test** besteht ist zusätzlich folgendes notwendig

- **computer vision** um Objekte zu erkennen
- **robotics** um Objekte zu manipulieren und zu bewegen

⇒ Diese 6 Disziplinien vereinen das wichtigste zu AI.
Das Problem beim Turing-Test ist, dass er nicht produzierbar ist, nicht konstruktiv und sehr schwergänglich für eine klare mathematische Analyse.

1.3 Menschliches Denken: Der kognitive Modellierungsansatz

Wenn wir ein Programm wollen, dass wie ein Mensch denkt, so müssen wir verstehen wie der Mensch denkt. Dies kann man auf unterschiedliche Art & Weise tätigen.

- durch Selbstbeobachtung, man probiert seine eigene Gedankengänge zu analysieren und zu verstehen
- durch psychologische Experimente, man beobachte einen Person ïn action"
- durch Hirnuntersuchungen, man beobachtet das Gehirn ïn action"

Sobald man eine detaillierte Anwendung des menschlichen Denkens hat, kann man entsprechend ein Programm aufsetzen. Dabei ist die interdisziplinäre cognetive Wissenschaft sehr relevant. Sie beinhaltet im Wesentlichen die Computermodelle von AI, sowie den experimentellen Techniken der Psychologie.

1.4 Rationales Denken: Der Ansatz des Gedankengesetzes

Es beschreibt den logischen Schluss bei einer Anzahl von Informationen. Bspw: SSocrates ist ein Mann; Alle Männer sind sterblich; Socrates ist sterblich". Dieses Gesetz des Denkens wird auch als **Logik** genannt. Es gibt jedoch zwei Haupthindernisse zu diesem Ansatz:

1. Es ist nicht einfach informelles Wissen aufzubereiten, vor allem wenn das Wissen nicht 100
2. Es ist eine grosse Differenz zwischen einem Problem prinzipiell zu lösen und ein Problem in ëchtßu lösen

1.5 Rationales Handeln: Der Ansatz des rationalen Agenten

Ein Agent handelt einfach. Alle Computerprogramme sind Agenten, jedoch ist es nicht ein einfaches Handeln, sondern das Ziel ist es, von einem spezifischen Input den best möglichen Output zu liefern ("Das richtige tun"→ Zielerreichung maximieren). Dies wird auch als **rationalen Agenten** betitelt. Beim rationalen Denken war der Schwerpunkt für AI die richtige Schlussfolgerung zu ziehen. Die richtige Schlussfolgerung zu ziehen ist in gewissen Situation Teil des rationalen Handelns, jedoch ist die richtige Schlussfolgerung nicht immer die beste Entscheidung. Es gibt Situationen wo es nicht eine offensichtliche korrekte Lösung gibt und doch muss etwas gemacht werden. Dabei gelangt man mit der richtigen Schlussfolgerung nicht zu einem Ergebnis, aus diesem Grund ist das rationale Handeln in diesen Situation zu bevorzugen. Dabei gibt es im Wesentlichen zwei Vorteile:

1. Das rationale Handeln ist Allgemeiner gehalten als der Ansatz des rationalen Denkens, da die richtige Schlussfolgerung nur eine von vielen Möglichkeiten ist zu einer rationalen Entscheidung zu kommen.
2. Das rationale Handeln ist für die wissenschaftliche Entwicklung zugänglicher als Ansätze, die auf menschlichem Verhalten oder menschlichen Gedanken basieren.

Ein "Ding"welches seine Umgebung wahrnimmt und kennt und entsprechend darauf reagiert $\rightarrow f : P* \rightarrow A$ wobei P = Umgebung; * = gesamte Historie; A = Agent. Der Rationale Agente ist derjenige, der das gegebene Ziel unter den gegebenen Umständen und Ressourcen bestmöglich erreicht.

- ßchwach KI"→ Ein spezifisches Problem wird versucht optimal zu lösen.
- SStarke KI"→ Es wird nicht ein spezifisches Problem gelöst, sondern man möchte eine generelle Intelligenz schaffen

Diagramm von Slide P01 S.18

2 Intelligent Agents

Im Kapitel des rationalen Handelns haben wir den rationalen Agenten kennengelernt, welchen den Ansatz verfolgt immer aufgrund seiner Gegebenheiten und seiner Umwelt "das richtige zu tun".

Lernziele

- Definition von rationalen Agenten und PEAS kennen
- Ich kann erklären wieso man einen rationalen Agenten so nennt, obwohl er nicht allwissend und nicht in die Zukunft sehen kann
- Argue how expressiveness of an agent is a mixed blessing

2.1 Agenten und Umgebung (Environments)

Ein **Agent** alles, was seine **Umgebung** durch **Sensoren** aufnimmt und auf das Einwirken der Umgebung mittels Aktoren reagieren kann. Im Vergleich zum Mensch sind Sensoren beispielweise:

- Augen
- Ohren
- Hände
- uvm.

Merke. *Im Allgemeinen kann die Wahl des Agenten zu einem bestimmten Zeitpunkt von der gesamten bisher beobachteten Wahrnehmungssequenz abhängen, aber nicht von etwas, das er nicht wahrgenommen hat.*

Mathematisch gesprochen wird das Verhalten des Agenten durch eine **agent function** beschrieben, welche auf eine spezifische Aktion zeigt. Aus *externer Sicht* werden die Aktionen in einer Tabelle aufgelistet, welche man je nach Verhalten durchgehen kann. *interne Sicht* ist ein sogenanntes **Agent Program**, bei welchem die komplette Implementation stattfindet.

Merke. *Entscheidend ist nun, wie wir diese Tabelle ausfüllen. Was im Umkehrschluss darauf schliesst, was einen guten oder schlechten bzw. intelligenten oder dummen Agenten ausmacht.*

2.2 Gutes Verhalten: Das Konzept der Rationalität

Ein rationaler Agent macht immer das richtige, oder in andere Worte formuliert die Tabelle für die Agent function ist korrekt ausgefüllt.
Doch was bedeutet das Richtige tun? Ein möglicher Ansatz ist beim Betrachten der Konsequenz:
Wenn ein Agent in einer Umgebung niedergeschlagen wird, erzeugt er eine Abfolge von Aktionen entsprechend den empfangenen Wahrnehmungen. Diese Abfolge von Aktionen bewirkt, dass die Umgebung eine Abfolge von Zuständen durchläuft. Wenn die Reihenfolge korrekt ist, dann hat der Agent eine gute Leistung erbracht. Dieser Begriff der Erwünschtheit bzw. Korrektheit wird durch eine **performance measure** (Leistungskennzahl) erfasst, welche eine beliebige Folge von Umgebungszuständen auswertet.
Achtung: Man spricht von **environment states** und nicht von *agent states*.
Beim performance measurement ist dabei wichtig, dass man klar definiert nach was gemessen werden soll. Beim einem Staubsauger beispielweise sollte man die Auswertung auf das Ergebniss legen, sprich es befindet sich keinen Dreck mehr am Boden. Und **nicht** bei den Handlungen (Dreck erkennen, aufsaugen etc.), denn so könnte die Performance gesteigert werden, in dem der Staubsauger jedes Mal wenn er alles aufgesaugt hat, wieder den Dreck ausleert und neu aufsaugt. Daher gehen wir auf das Endergebniss (= keinen Dreck mehr).

Merke. *In der Regel ist es besser, Leistungskennzahlen nach dem zu gestalten, was man tatsächlich tut in der Umgebung, und nicht danach, wie man denkt, wie sich der Agent verhalten sollte.*

2.2.1 Rationalität

Rationalität ist jeweils von vier Dingen abhängig:

- Die performance measure, welche das Erfolgskriterium festlegt
- Die Vorkenntnisse zur Umgebung von dem Agenten
- Die Aktionen, welche der Agent ausführt
- Die bisherige Wahrnehmungssequenz des Agenten

Dies führt uns zu deiner **Definition für einen rationalen Agenten**

Definition 1. Für jede mögliche Wahrnehmungssequenz sollte ein rationaler Agent eine Aktion auswählen, von der erwartet wird, dass sie ihre Leistungskennzahl maximiert, wenn man die Beweise aus der Wahrnehmungssequenz und das eingebaute Wissen des Agenten berücksichtigt.

Beispiel: Rationaler Agent Ja / Nein? *Einleitende Frage: Stelle dir einen Staubsauger-Agent vor, welcher einen Raum reinigt, wenn es dreckig ist und zum nächsten Raum geht wenn er sauber ist. Ist das ein rationaler Agent?* ⇒ Kommt drauf an.
Zu erst müssen wir die performance measure, Umgebung, Sensoren und Aktoren definieren.

- Die performance measure erhält einen Punkt für jeden gereinigtes Raum, welcher innerhalb von 1000 Zeitschritten erfolgen konnte
- Die geografische Lage der Umgebung ist a priori bekannt, jedoch die Dreckverteilung sowie der Ausgangspunkt des Agenten nicht. Saubere Räume bleiben sauber und reinigen den aktuell Raum weiter. Links und Rechts bewegen den Staubsauger nach links und rechts, ausser wenn diese Bewegung das Verlassen der Umgebung veranlassen würde, dann würde der Agent dort bleiben wo er sich aktuell befindet

- Die einzigen Aktionen sind links, rechts und saugen
- Der Agent erkennt seine Position richtig und ob diese Schmutz enthält oder nicht

⇒ Unter diesen Umständen handelt es sich um einen rationalen Agenten

2.2.2 Allwissenheit, Lernen und Autonomie

Es gilt die beiden Begrifflichkeiten *Rationalität* und *Allwissenheit* klar zu unterscheiden.

- **Allwissendheit** - Der Agent kennt den aktuellen Outcome seiner Aktion und kann sofort darauf reagieren. → dies ist in der Realität jedoch unrealistisch
- **Rationaltität** - Bei der Rationalität geht es darum, seine erwartende Performance zu maximieren, dies während der aktuellen Performance. Diese lässt sich jedoch nicht perfektionieren. Die Rationalität setzt nicht Allwissenheit vorraus.

Ein wichtiges Ereignis bei der Rationalität ist das **information gathering**. Für unseren Agent ist jedoch nicht nur das Sammeln von Informationen elementar, sondern auch das **Lernen**. Er sollte so viel als mögliche von der erhaltenen Informationen lernen. Dabei können wir dem Agenten ein Basis-Wissen an Informationen liefern, welcher er selbstständig mit den gewonnen Informationen ausbauen kann.
In dem Masse, in dem sich ein Agent auf das Vorwissen seines Designers und nicht auf seine eigenen Vorstellungen verlässt, sagen wir, dass es dem Agenten an **Autonomie** mangelt. Ein rationaler Agent sollte jedoch in der Lage sein autonom zu sein - er sollte lernen er von dem bereits bestehenden Wissen kompensieren kann bzw. falsches Wissen zu korrgieren und dadurch sein Wissen weiteraufbauen. Nach ausreichender Erfahrung in seiner Umgebung kann das Verhalten eines rationalen Agenten effektiv unabhängig von seinem Vorwissen werden.

Nun steht die Definition der Rationalität und wird haben die Basis um einen rationalen Agent zu bauen. Zuerst müssen wir uns jedoch noch mit dem **task environment** befassen, welches das *essenzielle Problem* für den rationalen Agenten ist, welches er zu lösen hat.

2.2.3 Task environment spezifizieren

Die spezifizierten performance measure, the environment und der Agent's actuators und Sensoren, werden alle unter **task environment** zusammengefasst. Dies wird auch **PEAS** (**P**erformance, **E**nvironment,**A**ctuators,**S**ensors) genannt. Während dem Designen eines Agenten, muss zu aller erst immer das task environment so gut als möglich spezifiziert werden. Gerade auch als Ausblick für den Aufbau einer KI, sind die PEAS-Kriterien sehr wichtig. Analog dem Aufbau eine Business Cases, bevor man mit einem Unternehmen beginnnt.

Beispiel: automatischer Taxi Fahrer Die einzelnen Punkte werden nachfolgend erläutert

Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe, fast, legal, comfortable trip, maximize profits	Roads, other traffic, pedestrians, customers	Steering, accelerator, brake, signal, horn, display	Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard

Abbildung 1: PEAS Beschreibung für task environment bei einem automatisierten Taxi-Fahrer

1. **performance measure** - *was streben wir an?*
   - Korrekte Destination
   - Minimaler Gebrauch von Benzin, so wie möglich geringe Abnutzung
   - Minimale Kosten bzw. Zeit pro Reise

- Möglichst keine Übertretungen der Strassengesetze
- Maximale Sicherheit und Komfort
- Maximaler Profit

⇒ Es gibt einige Zielkonflikte, wo durch Kompromisse eingegangen werden müssen

2. **Environment** - *Wie sieht die Umgebung des Taxi's aus?*
   - Unterschiedliche Strassenarten (von Kinderzonen bis zur mehrspurigen Autobahn)
   - Zusätzliche Verkehrsteilnehmer (Passanten, andere Fahrzeuge, priorisierte Fahrzeuge Polizei etc.)
   - Veränderungen der Gegebenheiten wie bspw. Wetter
   - Multi-Nationaler Einsatzmöglichkeiten (bspw. Linksfahren in England)

   ⇒ Je mehr wir die Umgebung eingrenzen, desto einfacher werden die Design-Probleme
3. **Actuators** - *Welche Möglichkeiten muss es geben?*
   - Kontrolle über den Motor, das Steuerrad, sowie den Bremsen
   - Interaktionsmöglichkeiten (Bspw. Sprachgenerator zum mit den Passanten zu kommunizieren oder Displays für andere Verkehrsteilnehmer)
4. **Sensors** - *Was muss das Taxi von seiner Umgebung aufnehmen können?*
   - Mehrere kontrollierbare Kameras für die Aufnahme der Strasseneinflüsse
   - Distanzmessungen zu anderen Objekten
   - Erkennung der aktuellen Geschwindigkeitsbegrenzung inkl. Anpassung seiner Geschwindigkeit
   - Beschleunigungssensor, welcher auch bei Kurven o.ä. eingreift
   - Kontrolle über den Zustand des Autos (Benzin, Öl, Reifen, Bremsen, uvm.)
   - GPS
   - Mikrofon und Tastatur für die Eingabe zum gewünschten Ziel

⇒ Es werden diverse Agenten gebraucht um dies umsetzen zu können. Dabei kommen oftmals die Diskussionen auf, ob es sich um einen rationalen oder einen intelligenten Agenten handelt. Tatsächlich geht es nicht um die Unterscheidung zwischen "realenünd "künstlichenÜmgebungen, sondern um die Komplexität der Beziehung zwischen dem Verhalten des Agenten, der von der Umgebung erzeugten Wahrnehmungssequenz und dem Leistungsmass.

2.2.4 Eigenschaften der task environments

Die Anwendungsbereich in welchen KI eine Aufgaben übernehmen können sind enorm breit gefächert und vielfältig. Nachfolgend sind einige Agenten-Typen aufgelistet inklusive deren Beschreibung für die Eingliederung im PEAS.

Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system	Healthy patient, reduced costs	Patient, hospital, staff	Display of questions, tests, diagnoses, treatments, referrals	Keyboard entry of symptoms, findings, patient's answers
Satellite image analysis system	Correct image categorization	Downlink from orbiting satellite	Display of scene categorization	Color pixel arrays
Part-picking robot	Percentage of parts in correct bins	Conveyor belt with parts; bins	Jointed arm and hand	Camera, joint angle sensors
Refinery controller	Purity, yield, safety	Refinery, operators	Valves, pumps, heaters, displays	Temperature, pressure, chemical sensors
Interactive English tutor	Student's score on test	Set of students, testing agency	Display of exercises, suggestions, corrections	Keyboard entry

Abbildung 2: Beispiele von AgentenTypen und deren PEAS Beschreibung

Fully observable vs. partially observable

- Wenn ein Sensor zu jedem Zeitpunkt sämtliche *relevanten* Informationen zurück liefert um die Auswahl der betreffenden Aktion auszuwählen, sprechen wir von einem **fully observable** Umgebung.
  ⇒ fully observable Umgebungen sind sehr praktisch, da keine interne Zustände gespeichert werden müssen. Bspw. ist Schach fully observerable, jedoch Poker nicht. Da man beim Poker verdeckte Karten hat.
- Sobald eine Informationen fehlen oder zu ungenau sind, so sprechen wir von einer **partially observable** Umgebung
- Wenn der Agent keine Sensoren hat, sprechen wir von einer **unobservable** Umgebung.

Single Agents vs. Multiagent
Diese Unterscheidung fällt einem leicht zu treffen. Wenn nur ein Agent gebraucht wird um das Problem zu lösen, so wird von einem single Agent gesprochen. Bei den Mutliagents differenzieren wird noch zwischen verschiedenen Betrachtungen.

- competitive (konkurrierend)
- cooperative (kooperativ)

⇒ Beispielsweise im Eishockey spielt ein Team (Teamintern kooperativ) gegen ein anderes Team (konkurrierend).

Deterministisch vs. Stochastik vs. nondeterministic Wenn der nächste Zustand der Umgebung vollständig durch den aktuellen Zustand und die vom Agenten ausgeführte Aktion bestimmt wird, dann sagen wir, dass die Umgebung **deterministisch** ist, sonst nennen wir es **stochastisch**. Das Taxi-Beispiel ist klar stochastisch getrieben, da man nie eindeutig voraussagen kann wie das Verhalten des Verkehrs ist.
nondeterministic
Dinge die bspw. sehr schwer zum Berechnen, sind da die Berechnung sehr komplex ist, wird nicht-deterministisch (man kann keine genaue Wahrscheinlichkeit angeben)
Wir sagen, dass die Umgebung **unklar / unsicher** ist, wenn die Umgebung nicht fully observale oder nicht deterministisch ist.

episodisch vs. sequentiell
Bei einer **episodischen** Ausführung eines Tasks ist jeder Task in sich geschlossen (atomic). Dementsprechend ist die

nächste Aufgabe faktisch unabhängig der vorherigen. Bei der **sequentiellen** Ausführung kann eine Entscheidung im Task x, alle nachfolgenden Entscheidungen und Ausführunen beinflussen.

static vs. dynamic
Wenn die Umgebung sich ändern kann, während ein Agent 'nachdenklich' ist, so sprechen wir von einer **dynamischen** Umgebung. Sonst handelt es sich um eine **statische** Umgebung.

diskret vs. kontinuierlich
Die Unterscheidung zwischen diskret und kontinuierlich ist abhängig des *state* von der Umgebung, der *Zeit* in welcher es handelt, sowie der *Wahrnehmungen* und *Aktionen* des Agenten.

bekannt vs. unbekannt
In einer **bekannten** Umgebung, sind die Ergebnisse (oder die Ergebnisse für alle Möglichkeiten in einer stochastischen Umgebung) für alle Aktionen gegeben. Wobei in einer **unbekannten** Umgebung der Agent seine Umgebung zuerst noch kennenlernen muss. Es ist jedoch auch möglich, dass es sich um eine *bekannte, partially observable* Umgebung handelt, bspw. Solitaire. → Ich kenne die Regeln, bin jedoch nicht im Stande zu sehen, was als nächste Karte kommt. Des Weiteren gibt es auch *unbekannte, fully observable* Umgebungen, bspw. Video-Spiele. → Ich sehe den kompletten Bildschirm, aber ich weiss trotzdem nicht welche Auswirkungen die einzelne Knöpfe haben, bis ich diese ausprobiert habe.

Task Environment	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with a clock	Fully	Multi	Deterministic	Sequential	Semi	Discrete
Poker	Partially	Multi	Stochastic	Sequential	Static	Discrete
Backgammon	Fully	Multi	Stochastic	Sequential	Static	Discrete
Taxi driving	Partially	Multi	Stochastic	Sequential	Dynamic	Continuous
Medical diagnosis	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Image analysis	Fully	Single	Deterministic	Episodic	Semi	Continuous
Part-picking robot	Partially	Single	Stochastic	Episodic	Dynamic	Continuous
Refinery controller	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Interactive English tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete

Abbildung 3: Beispiele von Task Environsments mit deren charakteristik

2.3 Die Struktur der Agenten

Der Job von KI ist es, ein **agent program** zu designen, welches die agent function implementiert (Mapping von Wahrnehmung zu Aktion). Wir nehmen an, dass dieses Programm auf einem Computer-Gerät mit physikalischen Sensoren und Actuators läuft. Dies nennen wir **Architektur**:
$agent = architecture + program$ Natürlich müssen wir unser Programm so wählen, welches auf die entsprechende Architektur angepasst ist. Wenn wir eine Aktion ausführen *laufen*, so wäre es besser, wenn die Architektur Beine hat.

2.3.1 Agent programs

Agent Programs, welche wir hier designen haben alle dasselbe Grundgerüst: Sie nehmen die aktuelle Wahrnehmung als Input der Sensoren auf und wiedergeben diese mittels einer Aktion an die Aktoren.

function TABLE-DRIVEN-AGENT(*percept*) **returns** an action
  persistent: *percepts*, a sequence, initially empty
    table, a table of actions, indexed by percept sequences, initially fully specified

  append *percept* to the end of *percepts*
  action ← LOOKUP(*percepts*, *table*)
  return *action*

Abbildung 4: Beispieltabelle für Table Driven Agent

Wenn wir den Einsatz von einem Table-Driven Ansatz genauer anschauen, merken wir, dass dieser in der Praxis nicht verwertbar ist. Definieren wir P als mögliche Wahrnehmungen und T als Lifetime des Agenten (so viele Wahrnehmungen wird der Agent erhalten). Die LookUp Table wird mit folgender Grösse befüllt: $\sum_{t=1}^{T} |P|^T$. Wenn wir nun ausgehen, dass in bei einem Taxi-Fahrer eine Kamera pro Sekunde 27MegaBytes an informationen liefert, ergibt das eine Tabelle von $10^{250'000'000}$ Einträge. → keine Tabelle auf der Welt kann diese Menge an Datenspeichern.

Und doch, der Tabellen-Ansatz macht das was wir wollen → es implementiert die gewünschte agent function. Der Hauptherausforderung bei KI ist, dass wir einen guten Weg finden wie wir das Programm schreiben, welches alle Möglichkeiten abdeckt und nach rationalem Verhalten funktioniert.

2.3.2 simple reflex agents

Die einfachste Art eines Agenten ist der simple reflex agent. Dieser Agent wählt die Aktion auf Basis der *aktuellen* Wahrnehmung aus, und ignoriert dabei die komplette Historie. Wenn wir wieder Bezug nehmen auf den Staubsauger, war dieser ein simple reflex agent. Denn er entschied auf Basis der aktuellen Situation und ob diese Dreck enthalten hat oder nicht. Dabei ist die Reduzierung auf das Vernachlässigen der Historie mit der grössten Auswirkung betroffen. Denn die Anzahl von Möglichkeiten 4^T wurde auf 4 reduziert.
⇒ Reflex, es passiert etwas ohne darüber nachzudenken (Beim Menschen bspw. das Blinzeln, Husten etc.)
Simple reflex agent kommen jedoch auch in komplexeren Umgebungen vor. Wenn wir uns wieder an das Taxi-Beispiel erinnern, so wäre ein mögliches Szenario, dass wenn das Auto vor mir bremst und sein Bremslicht angeht, dass ich dies erkennen muss und ebenfalls bremsen muss.
if *car-in-front-is-breaking* ***then*** *initiate-breaking*

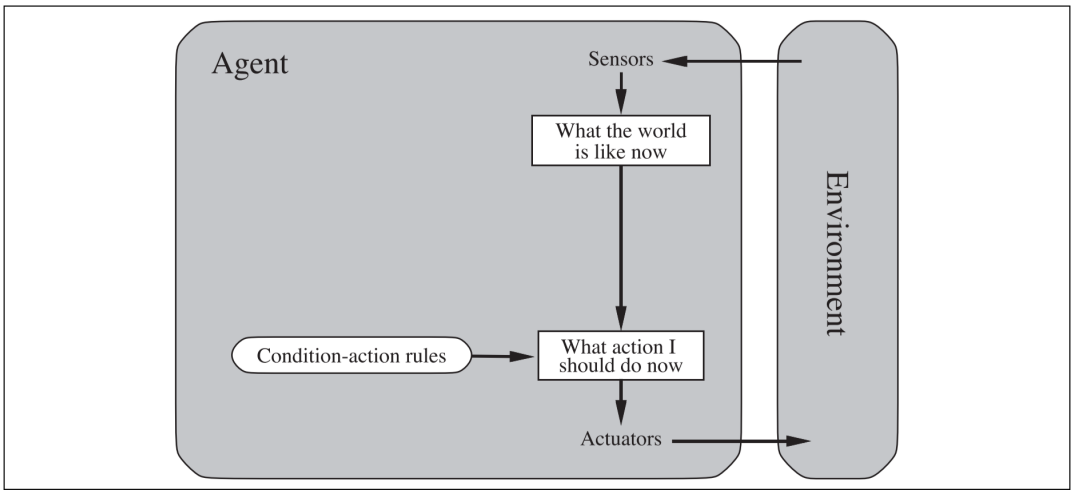


Abbildung 5: Schema eines Simple Reflex Agent

function SIMPLE-REFLEX-AGENT(*percept*) **returns** an action
persistent: *rules*, a set of condition–action rules

state ← INTERPRET-INPUT(*percept*)
rule ← RULE-MATCH(*state*, *rules*)
action ← *rule*.ACTION
return *action*

Abbildung 6: Function eines Simple Reflex Agent

3 Problemlösung mittels Suche

Die Suche ist ein klassicher KI-Ansatz, welcher schon seit einiger Zeit im Einsatz ist und trotzdem hat es seine starke Berechtigung.

3.1 Lernziel

- Kennen der klassischen Such-Algorithmen und wie sie sich hier in der Komplexität (Speicher und Zeit) unterscheiden
- Verstehen wie ein intelligentes Verhalten funktioniert
- kennen wie man eine Suche mittels einer Heuristik verbessern kann.
- Modellierung auf ein Real-World Beispiel anwenden kann

3.2 Suche als Problemlösung

Beispielsweise das GPS-Problem mit einer Anfangsstadt (**Initial State**) und einer finalen Destination (**formulate goal**). Das Problem wird mittels dem Formulieren des Problems, mittels verschiedenen **States** (Städte) und **actions** (Fahrt zwischen den Städten).

Dabei beginnt man zuerst mit der **Formulierung des Problems**.

1. initial state → man befindet sich in Arad
2. successor function → gibt uns die verschiedenen Möglichkeiten an
3. goal test → explizit (Zielstadt ist Bukarest) oder implizit (nicht mehr dreckig)
4. path cost → aufsummierte Kosten der Distanz oder die Anzahl der Aktionen etc.

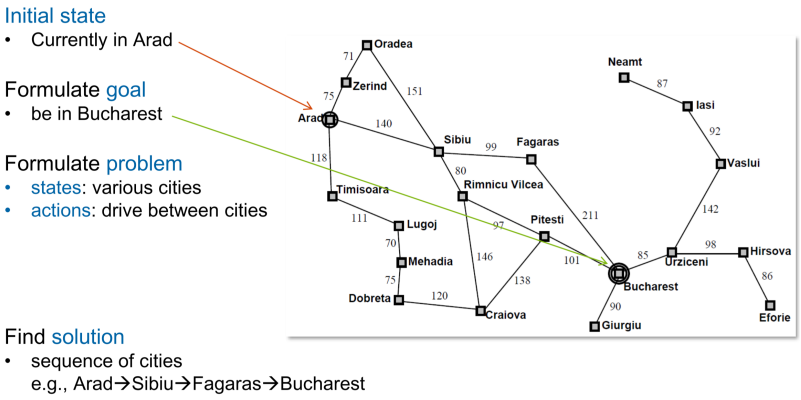
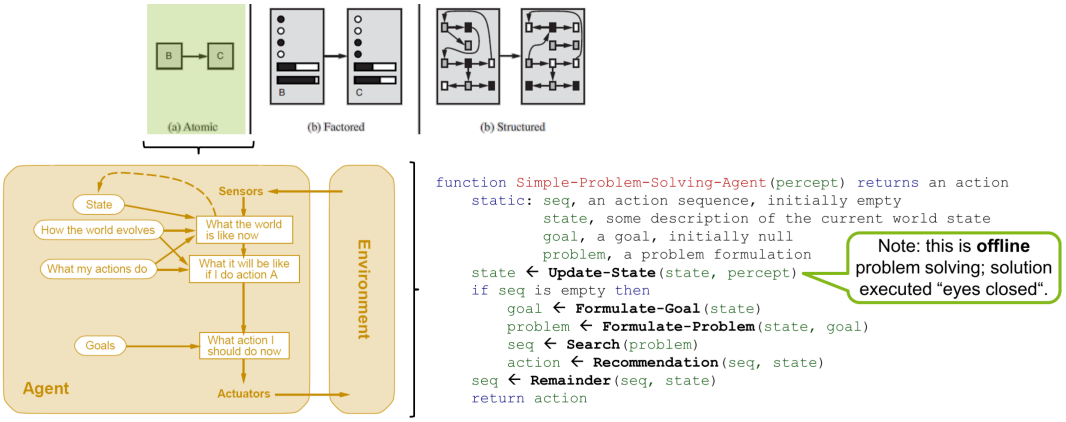


Abbildung 7: Real-World Beispiel für eine Baumsuche

geeigneten Suchagenten strukturieren
Dabei soll der Agent so aufgebaut werden, dass sich "die Welt"darin wieder findet.
Offline problem solving bedeutet, dass man das Problem simulieren kann. Dagegen ist die Online Problemlösung bspw. das effektive Abfahren der Strecke.



centering

Abbildung 8: Möglicher Aufbau eines modellbasierten Agenten

3.2.1 Verschiedene Ansätze der Suche

- **uninformed (blind) search** Man ist blind, man kann nicht abschätzen, welcher Ansatz der besser ist. Dementsprechend muss man alle Ansätze durchgehen und am Schluss entscheiden, was der bessere isch. Vor allem geeignet bei: fully observable, deterministisch und diskret
- **heuristic** Kennt den nächsten Status. Vor allem geeignet bei: identisch wie uninformed, jedoch grösser
- **Online Search** für Umgebungen, welche dynamisch sind
- **local search** fokusiert sich auf das Finden des Ziels und nicht auf den optimalen Pfad
- **adversial search** ist die Suche wenn man einen Gegner hat, bspw. Schach

3.3 Uninformed / blinde Suche

Ist eine Baumsuche, welche alle Knoten des Baums gesucht wird. Evaluationkriterien für die Strategie folgende:

- complet findet der Algorithmus immer eien Lösung
- optimal, findet der Algorithmus jeweils die Lösung mit den geringsten Kosten
- Zeit-Komplexität, Anzahl der Knoten (generated/expaneded
- Speicherplatz-Komplexität, maximale Anzahl an Knoten im Speicher

Zeit und Speicher Komplexität sind gemessen in

- *b:* maximum branching factor of the search tree
- *d:* Tiefe der Kostengünstigste Lösung
- *m:* maximum depth of the state space (may be infinity

Growth of time and memory requirements

- Algorithm: breadth-first search (→ ADS: exponential time & space complexity $O(b^d)$)
  Assumptions: $b = 10$, 1 mio nodes/sec, 1 kB/node
  Question: what d is easily manageable?

centering

Depth	Nodes	Time	Memory
2	110	.11 milliseconds	107 kilobytes
4	11,110	11 milliseconds	10.6 megabytes
6	10^6	1.1 seconds	1 gigabyte
8	10^8	2 minutes	103 gigabytes
10	10^{10}	3 hours	10 terabytes
12	10^{12}	13 days	1 petabyte
14	10^{14}	3.5 years	99 petabytes
16	10^{16}	350 years	10 exabytes

Abbildung 9: Live Beispiel einer blinden Suche

Strategien für die blinde Suche

- *Tiefensuche* tbd
- *Breitensuche* tbd
- *Uniform-Cost* tbd
- *limitierte Tiefensuche* tbd
- *Iterative Deepening* Mix aus Tiefen- und Breitesuche, da es auch die Vollständigkeit und Optimalität widerherstellt

centering

Expand the shallowest unexpanded node | Expand node with lowest path cost $g(n)$ | Expand deepest node | DFS only up to level l | Try DLS with $l = 1$, $l = 2$, ... until goal is reached

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening
Complete?	Yes*	Yes*	No	Yes, if $l \geq d$	Yes
Time	b^{d+1}	$b^{\lceil C^*/\epsilon \rceil}$	b^m	b^l	b^d
Space	b^{d+1}	$b^{\lceil C^*/\epsilon \rceil}$	bm	bl	bd
Optimal?	Yes*	Yes	No	No	Yes*

Abbildung 10: unterschiedliche Strategien für die blinde Suche

3.4 heuristische Suche

Man hat eine Möglichkeit einen Zustand zu bewerten und abzuschätzen, ob man mit diesem Zustand näher am Ziel ist als ein anderer Zustand oder einen vorherigen Zustand.
Ein möglicher Ansatz ist die **best-first-Suche**:

- Wähle den nächsten Node aus, basierend auf einer definierten Funktion $f(node)$
- Typischerweise ist f, als Heuristik $h(node)$ implementiert
- $h(node)$ facillitates pruning of the search tree

3.4.1 typische Implementationen

Greedy Search

- immer den günstigen Weg nehmen zum nächsten Node
- tbd

A

- A $\rightarrow$ man nimmt den günstigen Weg, berücksichtigt jedoch noch den bereits zurückgelegten Wert
- h(n) needs to be admissible: **tbd**
- A ist optimal und komplett
- A hat die Zeit-Kompexität $O(2'erroofh) * d$

SMA - simplified memory-bounded A

- A braucht meistens zu viel Speicherplatz, da kommt dann der SMA ins Spiel
- tbd