

# 1 AI101-02: AI Systems

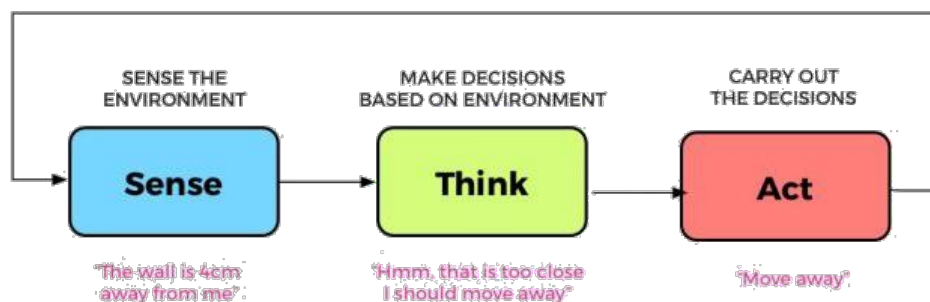
Dieses Kapitel behandelt die grundlegenden Bausteine von KI-Systemen: Agenten und ihre Umgebungen. Es definiert Rationalität und klassifiziert verschiedene Arten von Umgebungen und Agentenarchitekturen.

## 1.1 KI-Systeme und Agenten

Ein KI-System wird definiert durch die Interaktion zwischen einem Agenten und seiner Umgebung.

### Definition: Agent

Ein **Agent** ist eine Entität, die ihre Umgebung durch **Sensoren** wahrnimmt und auf diese Umgebung durch **Aktuatoren** (Effektoren) einwirkt.



- **Wahrnehmung (Percept)**: Der sensorische Input des Agenten zu einem bestimmten Zeitpunkt.
- **Wahrnehmungssequenz (Percept Sequence)**: Die vollständige Historie aller Wahrnehmungen, die der Agent bisher erhalten hat.
- **Agentenfunktion**: Eine mathematische Abbildung von Wahrnehmungssequenzen auf Aktionen:  $f : P^* \rightarrow A$ .
- **Agentenprogramm**: Die konkrete Implementierung der Agentenfunktion, die auf einer physischen Architektur läuft.

## 1.2 Rationalität

Ein rationaler Agent ist nicht zwangsläufig ein perfekter oder allwissender Agent. Rationalität bezieht sich auf die Qualität der Entscheidungsfindung basierend auf den verfügbaren Informationen.

### Rationaler Agent

Ein **rationaler Agent** wählt für jede mögliche Wahrnehmungssequenz diejenige Aktion aus, die erwartungsgemäß sein Leistungsmaß (*Performance Measure*) maximiert, unter Berücksichtigung der Wahrnehmungssequenz und des eingebauten Wissens.

Rationalität hängt von vier Faktoren ab:

1. Dem **Leistungsmaß** (Performance Measure), das den Erfolg definiert.
2. Dem **Wissen**, das der Agent bereits besitzt (Prior Knowledge).
3. Den **Aktionen**, die der Agent ausführen kann.
4. Der bisherigen **Wahrnehmungssequenz**.

**Wichtige Unterscheidung:**

- **Allwissenheit (Omniscience):** Kennt das tatsächliche Ergebnis jeder Aktion (in der Realität unmöglich).
- **Rationalität:** Maximiert das *erwartete* Ergebnis basierend auf dem aktuellen Wissen.

### 1.3 Task Environments (PEAS)

Um einen Agenten zu entwerfen, muss die Aufgabenstellung spezifiziert werden. Dies geschieht durch das PEAS-Modell.

#### PEAS

- **Performance Measure (Leistungsmaß):** Woran wird Erfolg gemessen?
- **Environment (Umgebung):** Wo operiert der Agent?
- **Actuators (Aktuatoren):** Womit handelt der Agent?
- **Sensors (Sensoren):** Womit nimmt der Agent wahr?

#### Beispiel: Autonomes Taxi

- **P:** Sicherheit, Schnelligkeit, Legalität, Komfort, Profit.
- **E:** Straßen, anderer Verkehr, Fußgänger, Wetter.
- **A:** Lenkung, Gaspedal, Bremse, Hupe, Blinker.
- **S:** Kameras, Radar, Tacho, GPS, Motorensensoren.

### 1.4 Eigenschaften von Umgebungen

Die Art der Umgebung bestimmt maßgeblich das Design des Agenten. Umgebungen werden anhand folgender Dimensionen klassifiziert:

1. **Fully Observable (Vollständig beobachtbar) vs. Partially Observable:**
  - *Fully:* Die Sensoren geben zu jedem Zeitpunkt den kompletten Zustand der Umgebung wieder (kein interner Speicher nötig).
  - *Partially:* Teile des Zustands sind verdeckt oder Sensoren sind ungenau (interner Zustand/Gedächtnis nötig).
2. **Single Agent vs. Multi-Agent:**
  - *Single:* Der Agent agiert allein (z.B. Kreuzworträtsel).
  - *Multi:* Es gibt andere Agenten, die kooperativ oder kompetitiv sein können (z.B. Schach, Straßenverkehr).
3. **Deterministic vs. Stochastic:**
  - *Deterministic:* Der nächste Zustand wird vollständig durch den aktuellen Zustand und die Aktion des Agenten bestimmt.
  - *Stochastic:* Es gibt Unsicherheiten/Zufall (z.B. Wetter, Würfelglück). Wenn Wahrscheinlichkeiten unbekannt sind, nennt man es *non-deterministic*.
4. **Episodic vs. Sequential:**
  - *Episodic:* Die Erfahrung ist in atomare Episoden unterteilt. Eine Aktion in Episode *A* hat keine Auswirkung auf Episode *B* (z.B. Bildklassifizierung).
  - *Sequential:* Die aktuelle Entscheidung beeinflusst alle zukünftigen Entscheidungen (z.B. Schach, Fahren).
5. **Static vs. Dynamic:**
  - *Static:* Die Umgebung ändert sich nicht, während der Agent "nachdenkt" (z.B. Kreuzworträtsel).
  - *Dynamic:* Die Umgebung ändert sich kontinuierlich (z.B. Taxi fahren).

- *Semi-dynamic*: Der Zustand ändert sich nicht, aber die Bewertung (Performance) ändert sich mit der Zeit (z.B. Schach mit Schachuhr).

#### 6. Discrete vs. Continuous:

- Bezieht sich auf die Zustände, Zeit oder Aktionen. Schach ist diskret (feste Felder), Taxi fahren ist kontinuierlich.

## 1.5 Agententypen

---

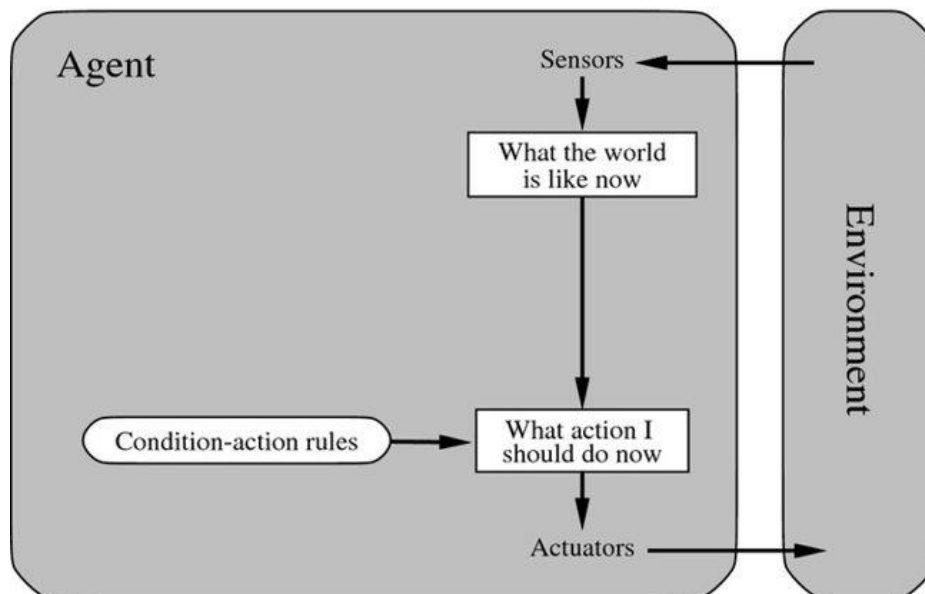
Es gibt vier grundlegende Typen von Agentenprogrammen, aufsteigend nach Komplexität und Fähigkeit.

### 1.5.1 1. Simple Reflex Agents (Einfache Reflex-Agenten)

---

Diese Agenten entscheiden nur basierend auf der *aktuellen* Wahrnehmung. Sie ignorieren die Wahrnehmungshistorie.

- **Funktionsweise**: Condition-Action Rules (Wenn-Dann-Regeln).
- **Formel**: Aktion = Funktion(Aktuelle Wahrnehmung).
- **Einschränkung**: Funktionieren nur zuverlässig in *fully observable* Umgebungen. Drohen in unendliche Schleifen zu geraten, wenn die Umgebung nur teilweise beobachtbar ist.

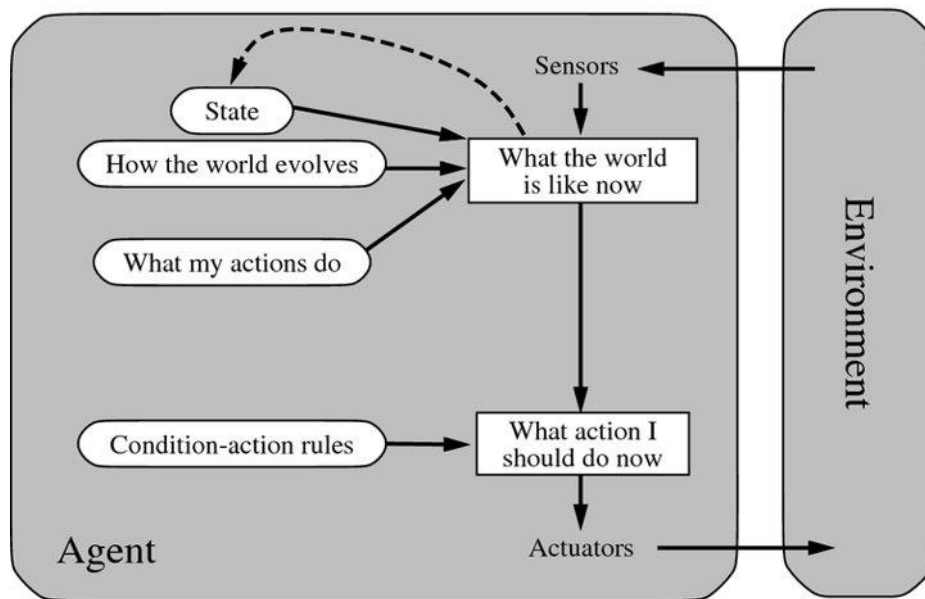


### 1.5.2 2. Model-based Reflex Agents (Modellbasierte Reflex-Agenten)

---

Diese Agenten besitzen einen internen Zustand (*Internal State*), um mit partieller Beobachtbarkeit umzugehen.

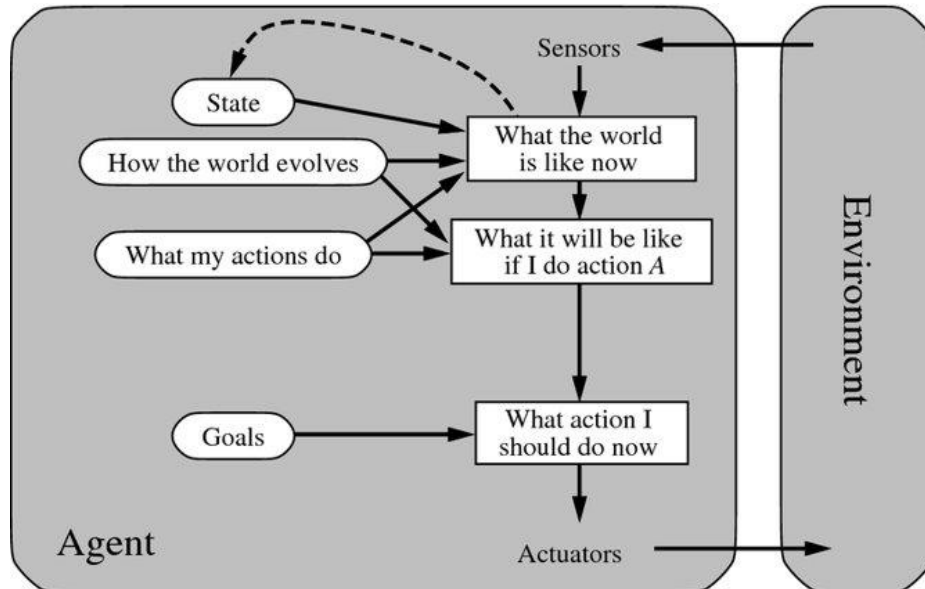
- **Interner Zustand**: Repräsentiert Wissen über die Welt, das aktuell nicht sichtbar ist (Gedächtnis).
- **Modell der Welt**: Wissen darüber, wie die Welt funktioniert (Übergangsmodelle) und wie Aktionen die Welt verändern.
- **Ablauf**: Update State → Wähle Regel → Aktion.



### 1.5.3 3. Goal-based Agents (Zielbasierte Agenten)

Wissen über den Zustand reicht nicht aus; der Agent benötigt ein Ziel (*Goal*).

- **Planung/Suche:** Der Agent simuliert verschiedene Sequenzen von Aktionen, um zu sehen, ob sie zum Ziel führen.
- **Unterschied zum Reflex:** Der Reflex-Agent reagiert, der Goal-based Agent plant in die Zukunft.
- **Flexibilität:** Ziele können sich ändern, der Agent passt sein Verhalten an.

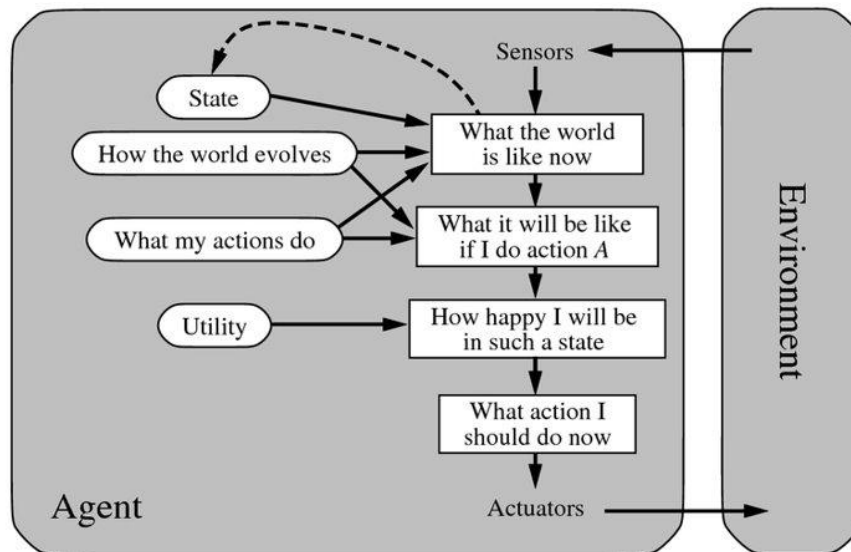


### 1.5.4 4. Utility-based Agents (Nutzenbasierte Agenten)

Ziele sind oft binär (erreicht/nicht erreicht). Nutzen (*Utility*) erlaubt eine feinere Unterscheidung.

- **Utility Function:** Bildet einen Zustand auf eine reelle Zahl ab, die den Grad der "Zufriedenheit" des Agenten angibt.
- **Vorteil:** Ermöglicht rationale Entscheidungen bei
  - Zielkonflikten (z.B. Geschwindigkeit vs. Sicherheit).

- Unsicherheit (Abwägung von Erfolgswahrscheinlichkeit und Nutzen).



### 1.5.5 5. Learning Agents (Lernende Agenten)

Jeder der obigen Agententypen kann lernfähig gemacht werden. Dies erlaubt dem Agenten, in unbekannten Umgebungen zu operieren und kompetenter zu werden.

#### Komponenten eines lernenden Agenten

- **Performance Element:** Der eigentliche Agent (einer der 4 obigen Typen), der Entscheidungen trifft.
- **Critic (Kritiker):** Bewertet, wie gut der Agent ist, basierend auf einem festen Leistungsstandard.
- **Learning Element:** Nutzt das Feedback des Kritikers, um das Performance Element zu verbessern.
- **Problem Generator:** Schlägt Aktionen vor, die zu neuen Erfahrungen führen (Exploration), auch wenn diese kurzfristig suboptimal sind.

