Blatt 01 Search

Reimund Baron

Möglichkeiten und Grenzen sowie Auswirkungen der KI

1. Welche Probleme mittels Computer - Robotereinsatz können gelöst werden,
Welche sind aktuell noch ungelöst?

- Technische und wirtschaftliche Möglichkeiten:

Automatisierung von Routinearbeiten
Datenanalyse und Entscheidungsunterstützung
Robotik in gefährlichen oder monotonen Umgebungen
Personalisierung oder Serviceverbesserung
Kreative Unterstützung

- Aktuell ungelöste Probleme

Allgemeine KI die wie ein Mensch flexibel denken kann
KI Entscheidungen Erklärbarkeit
Ethische Standards und Regulierungen
Nachhaltigkeit mit der Reduzierung von Rechenleistung und Energieverbrauch

2. Auswirkungen auf die Gesellschaft durch die KI, etwa durch autonomes Fahren oder durch LLM's.

- LLM's haben Einfluss auf die gesprochene Sprache des Menschen, da durch etliche Daten

Von YouTube Diskussionen und Podcasts Wörter fallen die man als Mensch übernimmt wie im englischen Beispiel: delve, comprehend, boast, swift und meticulous.

- Kognitiver zerfall durch KI: Jegliche akademischen Arbeiten von der KI übernehmen lassen

Oder auch mittels Character.AI oder anderen Replikaten davon sich sozial abhängig von
Einem Chatbot machen.

- Nutzung von Land und Flächen für das Aufbauen von Datenzentren die Millionen, Billionen Kosten und nicht gerade Energieeffizient sind.

- Täuschung durch KI generierte Video-Inhalte. Je weiter man die Entwicklung vorantreibt, desto schwieriger wird es KI generierte Videos von echten zu unterscheiden.

- Fehler im Bereich des autonomen Fahrens bei dem unbeteiligte in Gefahr geraten können.

- Falschaussagen der KI im Bereich der Programmierung durch Veraltete Dokumentationen der genutzten Bibliotheken einer Programmiersprache.

Search.01:	**Problemformalisierung, Zustandsraum**

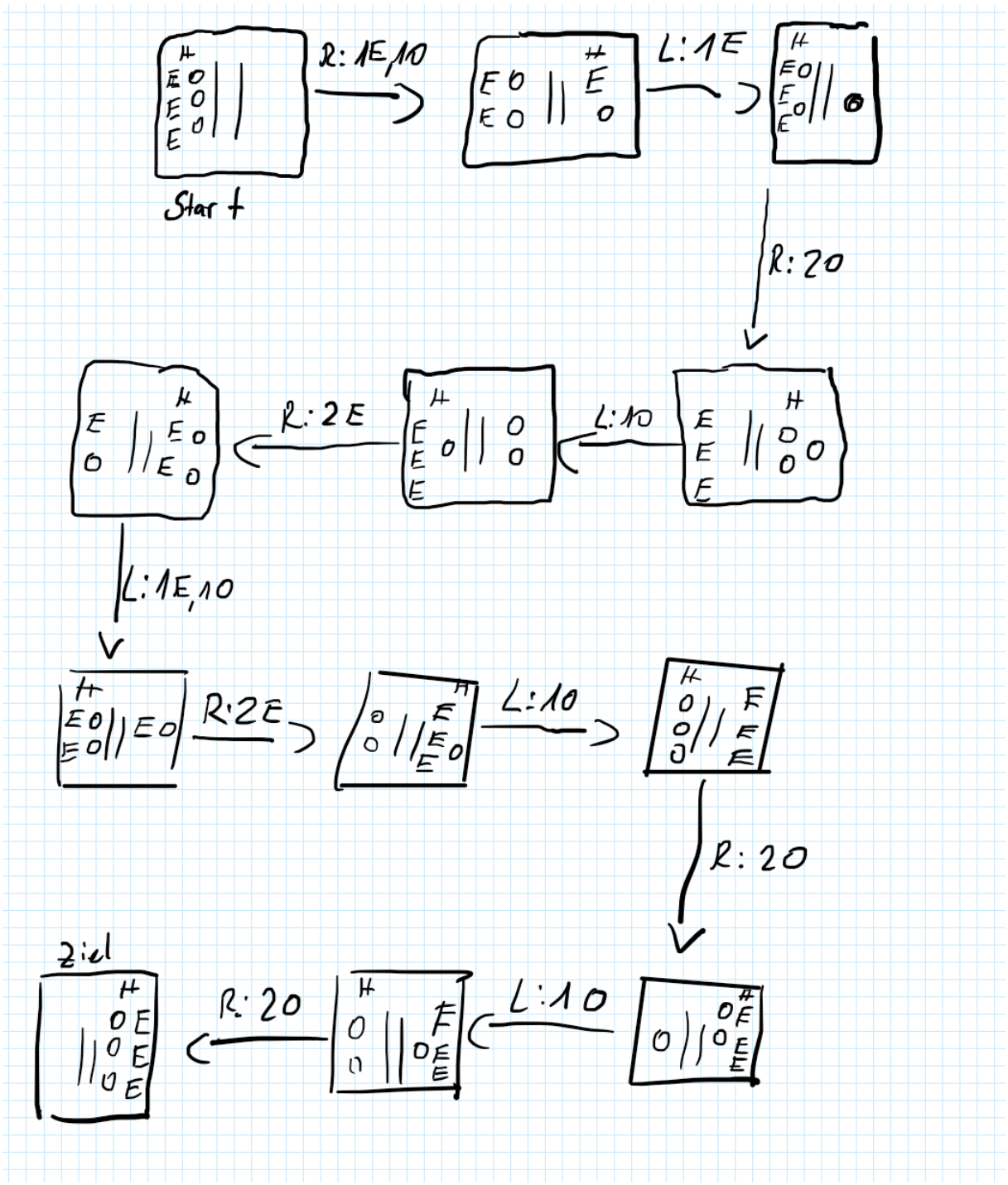
1. Formalisieren Sie das Problem (Zustände, Aktionen, Start- und Endzustand).

Wir definieren:	Elben = E,	Orks = O,	Pferd = H,	Links = L,	Rechts = R,	Pickup = P,	Drop = D

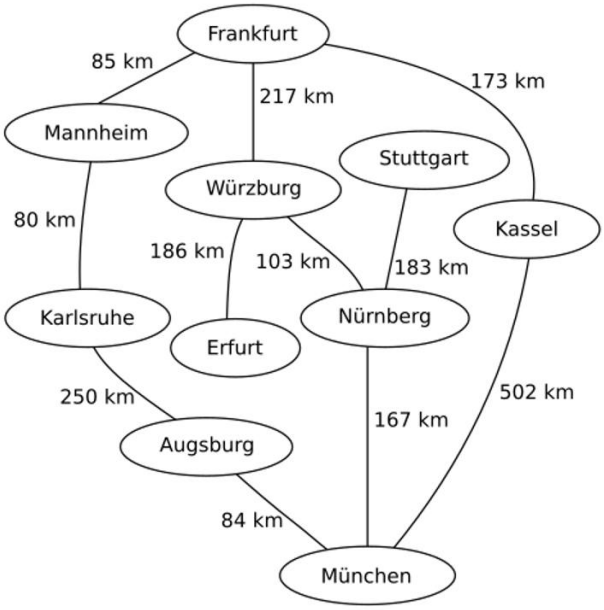
Startzustand = L (3E, 3O, P)
Endzustand = R (3E, 3O, P)

Zustandsmenge = {Startzustand: L(3E, 3O, P), …, Endzustand: R(3E, 3O, P)

Aktionen = L, R, P, D


R: 1E,10
L: 1E
R: 20
R: 2E
L: 10
L: 1E,10
R: 2E
L: 10
R: 20
Ziel
R: 20
L: 10
Start

Search.02:	Suchverfahren



1. Finden eines Weges von "Würzburg" nach "München"

Tiefensuche (Graph-Search): Als Datenstruktur wird ein Stack genommen (Frontier)

[]	Besucht: []
[Würzburg]	[]
[]	[Würzburg]
[Erfurt, Frankfurt, Nürnberg]	[Würzburg]
[Frankfurt, Nürnberg]	[Erfurt, Würzburg]
[Nürnberg]	[Frankfurt, Erfurt, Würzburg]
[Kassel, Mannheim, Nürnberg]	[Frankfurt, Erfurt, Würzburg]
[Mannheim, Nürnberg]	[Kassel, Frankfurt, Erfurt, Würzburg]
[München, Mannheim, Nürnberg]	[Kassel, Frankfurt, Erfurt, Würzburg]
[Mannheim, Nürnberg]	[München, Kassel, Frankfurt, Erfurt, Würzburg]

Breitensuche (Graph-Search): Als Datenstruktur wird eine Queue genommen

[]	Besucht: []
[Wü]	[]
[]	[Wü]
[Nü, Fr, Er]	[Wü]
[Nü, Fr]	[Er, Wü]
[Ma, Ks, Nü]	[Fr, Er, Wü]
[Mü, St, Ma, Ks]	[Nü, Fr, Er, Wü]
[Mü, St, Ma]	[Ks, Nü, Fr, Er, Wü]
[Ka, Mü, St]	[Ma, Ks, Nü, Fr, Er, Wü]
[Ka, Mü]	[St, Ma, Ks, Nü, Fr, Er, Wü]
[Au, Ka]	[Mü, St, Ma, Ks, Nü, Fr, Er, Wü]

A* (Tree-Search, "keine Zyklen"):

Kanten	g(n)
Würzburg-Nürnberg	103
Würzburg - Frankfurt	217
Würzburg - Erfurt	186
Frankfurt - Mannheim	85
Frankfurt - Kassel	173
Mannheim - Karlsruhe	80
Karlsruhe - Augsburg	250
Augsburg - München	84
Nürnberg - München	167
Nürnberg - Stuttgart	183
Kassel - München	502

Stadt	$h(n)$
Augsburg	0 km
Erfurt	400 km
Frankfurt	100 km
Karlsruhe	10 km
Kassel	460 km
Mannheim	200 km
München	0 km
Nürnberg	537 km
Stuttgart	300 km
Würzburg	170 km

Schätzungen der Restwegkosten für das Ziel *München*.

W: 0 + 170 = 170	Günstigster:
W - E: 186 + 400 = 586, **W - F: 217 + 100 = 317**, W - N: 103 + 537 = 640	W – F (317)
W - F - Ks: 390 + 460 = 850 , **W - F - Ma: 302 + 200 = 502**, W - E = 586, W - N = 640	W - F – Ma (502)
W - F - Ma - Ka: 382 + 10 = 392, W - F - Ks = 850, W - E = 586, W - N = 640	W - F - Ma – Ka (392)
W - F - Ma - Ka - A: 632 + 0 = 632, W - F - Ks = 850, **W - E = 586**, W - N = 640	W – E (586) -Keine weiteren Knoten, W - F - Ma - Ka – A (632)
W - F - Ma - Ka - A - Mu: 716 + 0 = 716, W - F - Ks = 850, **W - N = 640**	W – N (640)
W - N - Mu: 270 + 0 = 270, W - N - S: 286 + 300 = 586, W - F -Ks = 850	W - N – Mu (270)

München im Suchgraph vorhanden also günstigsten Weg nach München gefunden!
Während der Auflistung der Kosten wurde darauf geachtet bereits erschienene Städte zu überspringen.

2. Dürfen die oben gegebenen Restkostenabschätzungen in A* verwendet werden?

Die angegebenen Heuristiken sollten nicht für die A* Suche im Tree-Search verfahren verwendet werden da die Heuristiken an manchen Stellen nicht zulässig ist z.B.: Nürnberg mit 537km, obwohl die tatsächlichen Kosten 167 betragen, somit wurde die Zulässigkeit verletzt da eine Heuristik h(n) <= h*(n).

Kanten	g(n)
Würzburg-Nürnberg	103
Würzburg - Frankfurt	217
Würzburg - Erfurt	186
Frankfurt - Mannheim	85
Frankfurt - Kassel	173
Mannheim - Karlsruhe	80
Karlsruhe - Augsburg	250
Augsburg - München	84
Nürnberg - München	167
Nürnberg - Stuttgart	183
Kassel - München	502

Korrigierte Abschätzung	
Stadt	h(n)
München	0
Augsburg	80
Karlsruhe	300
Mannheim	350
Frankfurt	450
Stuttgart	220
Nürnberg	150
Würzburg	230
Kassel	450
Erfurt	400

W: 0 + 230 = 230	Günstigster:
W - E(186 + 400 = 586), W - F(217 + 450 = 667), **W - N(103 + 150 = 253)**	W - N (253)
W - N - Mu(270 + 0 = 270), W - E(586), W - F(667)	W - N - Mu(270)

Bei korrigierter Heuristik wurde bereits nach wenigen Durchläufen der kürzeste Weg gefunden.

Search.03:	**Dominanz**

Was bedeutet "Eine Heuristik h1(n) dominiert eine Heuristik h2(n)?

Wenn für alle Knoten n gilt:
h1(n) >= h2(n)

Und beide zulässig sind

Wenn h1 immer mindestens so große, nicht überschätzende Werte liefert wie h2, dann ist h1 eine bessere Schätzung der tatsächlichen Restkosten.

So wie in der Aufgabe "Suche den kürzesten Weg von Würzburg nach München"

Wir nehmen h2(n) = 0 für alle Städte -> Der Dijkstra Uniform Cost Search
Bei dem A* alles expandieren wird, da die Heuristik nicht hilft.

Für h1(n) nehmen wir die Werte aus der neuen Tabelle (Luftlinienentfernung. Geschätzter Weg etwas kleiner als echter Weg)

Beide führen zum kürzesten Weg, aber h2(n) braucht mehr Durchläufe da erst alles expandiert werden muss.

Search.04:	**Beweis der Optimalität von A***

A* wählt immer Knoten mit den kleinsten Kosten:

f(n) = g(n) + h(n)

g(n) = bisherige Kosten vom Start bis zu aktuellem Knoten
h(n) = geschätzte Restkosten bis zum Ziel

Wenn h zulässig ist dann ist h(n) nie zu groß, unterschätzt höchstens den Rückweg

A* durchsucht alle Wege in Reihenfolge von f(n)
Jeder Knoten auf dem optimalen Pfad hat einen f – Wert, der nicht größer ist als die tatsächlichen optimalen Gesamtkosten.
f(n) <= optimale Kosten

Wenn A* ziel erreicht, Preis = C

Annahme C -> nicht günstigster Preis, also irgendwo gibt es einen günstigeren Weg C*,

Aber dann müsste noch ein Knoten in der Warteschlange liegen damit C* günstiger sein kann.

Aber A* wählt immer den günstigsten Weg zuerst also kann kein Knoten mehr in der Warteschlange liegen Wiederspruch!