So, hallo.

Was haben wir das letzte Mal gemacht?

Wolltest du sagen, oder? Mhm.

Genau, wir haben Problem lösen als Suchproblem gesagt, und es gibt die drei Strategien, die wir angeguckt haben: Breitensuche, Tiefensuche und iterative Tiefensuche. Was wolltest du noch sagen? Okay, gut. Wie funktioniert das?

Also, ich habe wir gucken uns ein Problem an, das wir selber schon angeschaut hatten, das war das hier, ja? Das ist das falsche. Ähm.

Ist das besser? Wir haben gesagt, wir können das wir haben es angewendet auf so Beispiele mehr aus diesen Quizzes und Puzzles, aber wir können es auch auf Routenplanung zum Beispiel anwenden, und das war das Beispiel, das wir das letzte Mal angeschaut hatten. Also, wir haben hier so einen Routenplan gehabt, sorry.

Oder eine Route, die man wo wir Städteverbindungen haben, die man auch grafisch darstellen kann, und jetzt wollen wir mal hingehen und werden wir diese Darstellung haben, und wir würden jetzt Breitensuche anwenden. Breitensuche baut ja den Suchbaum in einer gewissen Strategie auf und arbeitet den ab.

Wie in welcher Reihenfolge würde jetzt, wenn wir von St. Gallen nach Lausanne eine Städteverbindung suchen wollen, in welcher Reihenfolge würde der Suchbaum aufgebaut werden, untersucht werden? Ja?

Genau, und was wären die Ebenen in dem Fall? Also, wir können mal schauen, einfach in welcher Reihenfolge würden wir die nächsten Knoten anschauen. Also, wir starten mit St. Gallen, steht ja auch auf der Folie. So, was wäre der nächste Knoten, den wir anschauen, wenn wir Breitensuche machen?

Genau, wir würden als nächstes Zürich anschauen

und Bad Goida.

Wir hatten auch eine Datenstruktur angeschaut, mit der man die Knoten verwalten würde. Wisst ihr das noch? Nee?

So eine Abkürzung mit vier Buchstaben? FIFO.

First In, First Out. Und da gibt es noch einen anderen Namen dafür, für die Datenstruktur?

Eine Queue, nicht lange. Wir hängen immer etwas hinten an und bauen von vorne an. Okay, das habe ich jetzt gemacht. Dann die nächsten Schritte nach St. Gallen wäre Zürich und Bad Goida, das merken wir uns, hängt hinten an und bearbeitet es von oben ab. Okay?

So, wenn wir das auf der Liste haben, was wäre das nächste, was wir machen? Basel.

Also, wir gucken, also das haben wir abgehakt, wir haben den Nachfolger gesehen, das ist ja noch nicht das Ziel. Jetzt gucken wir uns den nächsten Knoten an, das ist Zürich. Ja? Zürich ist nicht das Ziel. Okay? Dann schieben wir Basel auf. Müssen wir dann anschauen und was noch? Genau.

Ich mache es jetzt in der Reihenfolge, wie es da steht, eure Luzern, aber das sind die nächsten Knoten, die wir auf

die auf unsere Liste setzen, auf unsere Queue, also hängt das hinten an. So, haben wir Zürich abgearbeitet, das nächste, was wir abarbeiten, ist Bad Goida, was kommt dann auf die Liste?

Die Nachfolger von Bad Goida? Genau. Eigentlich wieder Luzern, jetzt könnten wir schauen, wenn wir ein Verfahren haben, dass die Sachen, die schon auf der Liste sind, nicht nochmal macht, wenn wir das nicht hätten, sondern ganz normale Breitensuche, dann würden wir das halt noch ein zweites Mal da draufsetzen.

Okay, wie geht es weiter?

Also, Bad Goida haben wir abgehakt. Solothurn.

Ja? Solothurn, stimmt's?

Genau, wir gehen von oben durch, ja? Der nächste offene Punkt ist Basel. Und was passiert in Basel? Nichts.

Basel ist nicht das Zielort, wir haben es abgearbeitet, es gibt aber keine Nachfolge, also brauchen wir nichts draufzusetzen. Der nächste Ort, den wir anschauen, ist Olten. Okay, was passiert in Olten?

Ja, also es kommt nicht dann Solothurn, ne? Also, wir müssen durchgehen, wir gucken es Schritt für Schritt an. Also, Olten ist auch nicht Zielort, jetzt nehmen wir die Nachfolger von Olten, das ist Solothurn.

Und Bern. Okay?

Also, das ist Breitensuche, wir müssen jetzt nicht durchgehen, also wir fangen von oben an, setzen alles auf die Queue und so laufen wir in der Breite unseren Baum durch.

Ich habe den Baum auch auf einer Folie, den haben wir das letzte Mal entwickelt, also wir da wenn wir das so anschauen, wir sind einfach diesen Baum Zeile für Zeile durchgegangen. Und das macht man, indem man das FIFO-Prinzip wandert. Gut. Die andere Suche, die wir hatten, war Tiefensuche, wie funktioniert die?

Also, ich habe das hier auf der Reihenfolge, wie wir es jetzt gerade auch gehabt haben, auf der Folie, die gebe ich euch nachher auch. Tiefensuche, wenn wir das Gleiche machen würden.

So, fangen wir auch wieder mit St. Gallen an.

Sorry, was wolltest du sagen?

Was ist als nächstes drauf? Zürich? Okay, also St. Gallen kommen wir nach Zürich.

Okay, Zürich ist nicht Endstation, was kommt als nächstes drauf? Basel. Also, das haben wir abgearbeitet, das haben wir abgearbeitet, jetzt kommt Basel. So, sorry. Ist es nur Zürich? Wenn wir den Baum aufbauen.

Wie funktioniert die Tiefensuche?

Genau, was heißt bis zum Schluss? Das heißt, bis zum Beginn. Okay, wenn die ganze Strecke durch ist, jetzt in diesem Fall St. Gallen-Zürich-Basel, dann würde es zurückgehen und St. Gallen-Zürich-Olten würde in die nächste Position.

Also, man geht in die Tiefe, und wenn man irgendwo an einem Punkt ist, wo es nicht weitergeht, was macht man dann? Und man ist noch nicht am Ziel? Zurück.

Zurück. Wie nennt man das? Backtracking. Backtracking. Man geht zurück, wohin zurück?

Zu welchem Knoten? Zum letzten? Zum vorher war, damit man gucken kann, ob es da noch einen Knoten gibt, den man abarbeiten kann. Okay, zum davor und zum letzten offenen Knoten. Zum letzten offenen Knoten, wo wir noch eine Alternative haben.

Das heißt, wir müssen uns ja merken, an welchem Knoten wir noch eine Alternative haben. Okay? Das heißt, wenn ich jetzt hier von Zürich, St. Gallen sage, was ist der Nachfolger Zürich, müsste ich mir merken, dass es bei St. Gallen auch noch einen anderen Nachfolger gab. Okay?

Wo merke ich mir das?

Genau, das ist Last and First Out, aber wir müssten uns das auch auf unsere Datenstruktur schreiben, dass wir noch eine zweite Alternative haben. Und ich, was war anders? Ich fange von rechts nach links an, damit man links den Berg runtergeht, oder? Oder runter rechts, wir könnten auch rechts gehen.

Wenn ihr sagt, Zürich ist daserste, wir bauen einfach auf, also Zürich, und wir müssen es ab Goida eben auch merken. Wir müssen diese Reihenfolge machen. Was wäre der nächste Knoten, den wir expandieren? Sicher?

Also, das ist Last and First Out. Wir haben es St. Gallen draufgesetzt, von St. Gallen aus haben wir gesagt, zwei Nachfolger sind Zürich und Bad Goida. Welchen von diesen Knoten expandieren wir als nächstes? Warum?

Wir gucken uns den obersten Knoten an, den letzten Knoten, Last and First Out, ja? Der letzte Knoten, der ist auf unserer Liste, ist das Bad Goida. Das heißt, wenn wir den anschauen, schauen wir, ist das ein Ziel? Sagen wir nein.

Dann nutzen wir die Nachfolger von Bad Goida, machen wir auf unsere Liste. Oder auf unsere, wie heißt die Datenstruktur noch?

So ein Stack, ja? Bad Goida ist es nicht, das können wir runterschmeißen, dafür nehmen wir die Nachfolger von Bad Goida. Wie viele Nachfolger hat Bad Goida? Nur einen, Luzern, ja? So, jetzt ist Luzern der oberste. Wir nehmen immer den obersten Knoten, also den letzten zugefügten, und expandieren.

So, was passiert jetzt bei Luzern?

Genau, wir schauen, ist Luzern die Lösung? Nein, das ist es nicht, können wir es runterschmeißen und nehmen die Nachfolger drauf. Das ist jetzt Tun. So, jetzt wird es spannend. Was ist denn Tun?

Genau, ich habe jetzt hier mit einem Pfeil eingezeichnet, um zu zeigen, von Tun gibt es keinen Weg nach Bern, sondern zurück. Also, wenn ich keinen Pfeil hätte, könnte man jetzt Bern als Nachfolger drauf. Okay? So, das ist eine Einbahnstraße, die wir haben. So, was war bei Tun? Ist nichts erfolgreich, ist kein Ziel.

Was ist jetzt der oberste Knoten?

Zürich dann. Das ist das letzte, was noch da ist. Also gucken wir, was hat, wir expandieren Zürich. So, was passiert, wenn wir Zürich expandieren?

Was jetzt mal auf unseren Stack? Basel.

Nur Basel? Olten? Und?

Luzern.

Einfach stur ab. Alles, was als Nachfolgeknoten ist, kommt jetzt auf unseren Stack. So, was arbeiten wir als nächstes ab? Luzern. Und das kennt man schon, was passiert, Nachfolger ist Tun. Bringt auch nichts. Tun hat aber keine Nachfolger.

So, jetzt gehen wir zum nächsten zurück, der noch bearbeitet werden kann.

Das ist der letzte auf unserer Liste, was noch da was noch nicht bearbeitet ist. So, jetzt gucken wir an, was hat Olten für Nachfolger. Gut, Olten hat Nachfolger Solothurn.

Und Bern. Gut.

Seid ihr verwirrt oder ist alles okay? In diesem Fall, wenn es in Last and First Out geht, machen wir einen ganzen Knotenweg zuerst. Und so haben wir jetzt hier Zürich und Bad Goida direkt aufgeschrieben.

Wir müssen uns das merken, dass die Nachfolger von St. Gallen sind Zürich und Bad Goida. Und einen davon expandieren wir und gehen dann in die Tiefe. Aber den anderen müssen wir uns merken, um zu sagen, bei Zürich hätten wir noch Alternativen gehabt. Okay.

Ja, bei Zürich haben wir noch nicht expandiert gehabt zu dem Zeitpunkt. Also, wir sind zurückgegangen, also ich habe das vergessen zu streichen. Wir sind zu Zürich zurückgegangen und haben gesagt, jetzt machen wir Zürich. Was waren da die Alternativen? Dann haben wir Zürich expandiert zu Basel, Olten und Luzern.

Dann war Olten der höchste und von dort sind wir dann, jetzt was war dann? Genau, Basel, Olten, Luzern, Luzern war der höchste. Dann haben wir gesagt, das ist nicht das Ziel, wo gibt es Nachfolger?

Wir haben also immer alle Nachfolger, nicht nur einen, sonst wüssten wir ja nicht mehr, dass es noch Alternativen gab. Deshalb müssen wir immer beim Nachfolger alle draufsetzen. Also, auch bei Zürich haben wir dann Basel, Olten und Luzern als Nachfolger gehabt und nicht nur Basel.

Und dann von denen, die wir draufgesetzt haben, nehmen wir den letzten, der oben steht, und expandieren den. Okay, so garantieren wir, dass wir wirklich nichts vergessen.

Und das ist ein einfaches Prinzip, wir brauchen nur immer alle Nachfolger auf unsere Datenstruktur zu werfen. Einmal in eine Schlange, dann haben wir Breitensuche, einmal auf den Stack, dann haben wir Tiefensuche. Das ist eigentlich alles.

Bei allen Knoten eine Verzweigung, dann schreibe ich die mir auch im Stack und ich kann auch zufällig dann einen Knoten erweitern oder den letzten. Der letzte, den wir draufgeschrieben haben, ja.

Wir haben es jetzt so gerade jetzt so gemacht, dass wir von links nach rechts draufgeschrieben haben, dann wird der rechteste zuerst expandiert. Wenn wir es umgekehrt draufgeschrieben hätten, wäre der linkeste zuerst expandiert worden. Aber was nach der Datenstruktur am obersten ist, das wird als nächstes expandiert. Okay?

Genau, Olten haben wir abgearbeitet, eigentlich hätte ich das streichen müssen, habe ich vergessen, und haben das gesetzt durch Solothurn und Bern. Jetzt würden wir Bern expandieren und bei Bern würden wir dann Fribourg und Thun.

Würden Bern streichen, wären Fribourg und Thun.

Thun ist das oberste, das bringt uns nichts. Ja, dann expandieren wir Fribourg, weil Fribourg auch noch nicht das Ziel ist. Dann kommen wir nach Lausanne und Montreux.

Reicht mir mein Platz nicht, aber was reicht dann? Also, Montreux ist nicht das Ziel. Montreux hat Lausanne als Nachfolger. Würde man Montreux streichen, hätten wir Lausanne und damit wären wir am Ziel. Weil das oberste, was jetzt da ist, ist Lausanne, wenn wir das expandieren, okay, das ist unser Ziel.

Und ihr seht, Basel steht immer noch drauf, das haben wir gar nicht erst versucht, ob es Nachfolger gibt, weil das, was früher stand, waren Luzern und Olten, die wurden zuerst abgearbeitet. Wandelsfrage? Ja.

Also kann es auch in der Tiefensuche sein, dass sie einen Weg finden, aber nicht unbedingt den schnellsten. Genau. Ja, das haben wir das letzte Mal, glaube ich, schon noch gesagt. Also, Tiefensuche findet einen Weg, wenn es einen gibt, aber nicht unbedingt den schnellsten. Und es gibt auch ein Problem bei der Tiefensuche, das haben wir, glaube ich, auch noch gesagt.

Ich habe gesagt, wenn es einen gibt, wann?

Genau, wenn es nur endlos Schleifen geben würde, dann würden wir auch keine Lösung finden. Das ist das Problem bei der Tiefensuche. Also, wir könnten ja so eine Schleife haben.

Ich habe hier, wir könnten zum Beispiel, ja, St. Gallen, Zürich, Luzern, und wenn wir das in beide Richtungen hätten, dann würde Luzern, ginge auch nochmal nach Bad Goida, dann können wir wieder nach St. Gallen, wir könnten dort zum Beispiel in eine Schleife laufen. Und dann würden wir keine Lösung finden. Okay?

Klar soweit? Gut, das waren die Hauptlösungen, die wir hatten. Hier habe ich das nochmal aufgezeigt, das ist jetzt der Weg, wenn man links gegangen wäre. Gut. Breitensuche, Tiefensuche, was ist iterative Tiefensuche?

Wie funktioniert das?

Genau, wir begrenzen die Tiefensuche auf eine bestimmte Länge. Wenn wir da keine Lösung finden, dann gehen wir eine Ebene tiefer. Was hat das für einen Vorteil von den Problemen, die wir eben genannt haben?

Genau, wir kommen nicht so schnell in eine Schleife. Wenn es eine andere Lösung gibt, mit einem kürzeren Pfad, dann finden wir sie, weil wir ja Breitensuche bis zu einer bestimmten Tiefe machen.

Ja, also es ist nicht so, dass wir, wenn es eine Lösung gibt, finden wir es mit iterativer Tiefensuche, ohne dass wir der Gefahr laufen, in eine Schleife zu gehen.

Gut, okay.

Jetzt haben wir gesagt, wir finden eine Lösung, aber nicht unbedingt die beste. Anderes Problem ist, dass wir immer den ganzen Suchbaum durchlaufen.

Da haben wir gestern letztes Mal angefangen, ich habe es, glaube ich, nur kurz erwähnt, was eine alternative wäre.

Heuristisch ist so. Was ist noch Heuristik?

Eine Faustregel. Wie können wir uns das noch denken? Faustregel.

Common Sense, ja. Etwa was üblicherweise funktioniert. Ja, so. Gucken wir uns das an.

Wie haben wir zu Tiefensuche und Breitensuche noch gesagt?

Uninformierte Suche. Ah, ich habe gesagt, ich habe das un, wie darf ich jetzt gehört. Ist die uninformierte Suche. Okay, was ist damit gemeint?

Ohne Angaben, ja. Was wolltest du sagen? Genau, wir haben keine Information, welcher Pfad vielleicht irgendwie besser sein könnte. Wir haben sie nur die Nachfolger.

Und deshalb schmeißen wir sie einfach irgendwie auf den Stack oder in in die Queue und arbeiten sie ab. Weil wir keine Information haben, ob ein Knoten vielleicht besser sein könnte als ein anderer. Ja, das nennt man uninformierte Suche. Und das haben wir mit Tiefensuche und Breitensuche.

Sie suchen immer den ganzen Suchbaum ab, weil sie keine Ahnung haben, was jetzt vielleicht besser wäre. Und die heuristische Suche, die hat Informationen, welcher Pfad vielleicht besser sein könnte.

Und die Idee von der heuristischen Suche ist, dass wir vielleicht, dass wir eine gute Lösung finden, nicht unbedingt die optimale, aber nicht schnell eine gute Lösung finden.

Also Heuristiken sind Strategien, die in vielen Fällen zu einer guten Lösung führen, aber dafür müssten wir irgendwie berechnen können oder Angaben machen können, welcher Pfad könnte der beste sein, den wir jetzt nehmen.

Also, wenn wir unser Beispiel von vorhin anschauen, jetzt mal hier rüber.

Ja, wenn wir von St. Gallen loslegen, was könnte besser sein? Zürich oder Bad Goida? Wenn wir da eine Information drüber hätten, würden wir den Pfad nehmen, der typischerweise oder der üblicherweise sinnvoller ist. Also, wir brauchen irgendwelche zusätzlichen Informationen.

Was könnten das hier für Informationen sein, wenn man einen Routenplaner macht?

Himmelsrichtung, Entfernung, genau. Solche Informationen, wenn wir sowas hätten, könnten wir irgendwie Angaben machen, ja, wahrscheinlich ist dieser Pfad besser als ein anderer Pfad. Also, wir müssen es irgendwie bewerten können, welcher Weg da besser ist. Das nennen wir eine Bewertungsfunktion.

So, wir haben bisher eine ganz einfache Datenstruktur, FIFO, wir schieben immer hinten an und machen vorne weiter, also eine Schlange, oder LIFO, wir haben einen Stack.

Und bei der heuristischen Suche schieben wir neun Knoten nicht einfach hinten hin oder vorne hin, sondern wir ordnen die neun Knoten dort ein, je nachdem, wie hoch die Bewertungsfunktion ist.

Wenn es eher ein besserer Knoten wäre, wird er weiter hinten angehängt, wenn er eher ein schlechterer Knoten wird, dann weiter vorne angehängt. Gut. Also brauchen wir irgendwie eine Möglichkeit, um abzuschätzen, wie gut ein Knoten, ein nächster Knoten zur Lösung beiträgt.

Und wie gesagt, das ist die Bewertungsfunktion. Jetzt macht es natürlich wenig Sinn, wenn die Bewertungsfunktion so aufwendig ist, als wenn wir den ganzen Graphen durchsuchen würden. Also, eine Bewertungsfunktion ist eine Funktion, die wir schnell berechnen können, einfach berechnen können und zu sagen, das könnte uns zum Ziel führen.

Und das ist der Algorithmus für die heuristische Suche. Wir brauchen es nicht im Detail durchzugehen. Was wichtig ist, bei FIFO haben wir immer vorne, sorry, hinten den Knoten angehängt.

Bei der Tiefensuche, bei LIFO haben wir immer vorne angehängt. Und jetzt steht hier einsortieren, und einsortieren heißt, wir nehmen die Knoten dorthin, wo sie nach der Bewertungsfunktion sortiert werden würden.

Damit wir immer als nächstes den haben, der die geringste Bewertungsfunktion hat, der uns schneller zum Ziel führen wird. Oder die höchste, wenn es darum geht, etwas mit einem möglichst hohen Wert zu haben.

Gut, also das Einsortieren, das ist eigentlich der wesentliche Punkt von der heuristischen Suche. Wir müssen eine Bewertung haben, sodass wir die Knoten sinnvoll sortieren können. Und diese Funktion, die besteht aus zwei Teilen.

Also Funktion nennt man F, besteht aus zwei Teilen, G und H. Irgendeine Idee, was das sein könnte?

Warum man das in zwei Teile aufspaltet?

Sie haben gesagt, wir haben eine Suchbahn.

Und den laufen wir durch. Okay, angenommen, das ist unser Zielknoten. Und wir sind jetzt durch den Suchbaum gelaufen und sind hier angelangt.

Jetzt wissen wir ja ganz sicher, wie viel Aufwand wir hatten oder wie gut es war, von hier nach hier zu kommen. Das kann man berechnen, das weiß man. Und von hier bis nach unten, das können wir nur schätzen.

Okay, also wie viel Kosten wir bisher verursacht haben, wissen wir, weil wir den Schritt ja gegangen sind. Also denkt immer an die Entfernung. Wenn wir von Zürich, von St. Gallen nach Zürich kommen, dann wissen wir, wie weit es ist. Was wir aber nicht wissen, ist, wie gut wir jetzt von Zürich nach Lausanne kommen, zum Beispiel.

Das können wir abschätzen. Also wir haben zwei Teile. Eins, was wir schon wissen, nämlich den Weg, den wir gegangen sind, den Aufwand da. Und was wir abschätzen können, ist der Rest vom Aufwand. Und das sind die beiden Funktionen. G ist der Aufwand, den wir schon wissen.

Und H ist die Heuristik, die Bewertungsfunktion für das, was wir nur abschätzen können.

Also die Abschätzung vom aktuellen Knoten zum Zielzustand.

Jetzt gibt es eine schöne Eigenschaft von dieser Schätzfunktion. Die heißt zulässig, wenn die Schätzfunktion nie höher ist als der reale Wert.

Soll schnell berechenbar sein und soll nicht höher sein als der Normalwert, weil wir dann garantieren können, dass wir eine optimale Lösung finden.

Also es gibt einen Algorithmus, den nennen wir A*, der findet eine optimale Lösung, wenn wir eine Schätzfunktion haben, die den aktuellen, den realen Wert nie überschätzt. Gut, und das können wir uns mal an einem Beispiel anschauen. Wir haben hier das Puzzle, das wir jetzt mal auch schon hatten.

Okay, und was wir, was wir unsere Kosten, wie viele Bewegungen wir brauchen, um zu einer Lösung zu kommen. Okay, so. Wie könnten wir das abschätzen, was eine gute Lösung ist?

Ohne dass wir es ausprobieren müssen.

Eine Idee?

Ja, wir wollen noch nicht verschieben, das wäre ja schon eine Aktion. Wir haben ja zwei Möglichkeiten, wir können die 5 nach rechts verschieben oder die 8 nach oben. Welche könnte besser sein? Das wollen wir abschätzen. Warum?

Weil wenn ich die 5 nach rechts schiebe, dann kommt dann rück die 4 weiter hoch und die 3 kann man automatisch weiter hoch setzen, oder? Und wenn ich das so weiterführe, dann muss ich irgendwann mal die 2 nach rechts verschieben und die 1 nach oben. Dann habe ich schon dieerste Reihe.

Ja, aber jetzt hast du schon im Kopf Sachen ausgeführt, oder? Ja. Das wollen wir nicht. Bevor wir irgendwas ausführen, wollen wir schon abschätzen können, wie viel Aufwand ist es. Und der Punkt ist, dass wir einfach mal zuerst wissen können, wie können wir Aufwand schätzen? Wie gut ist eine aktuelle Situation?

Wir können also von einer Situation die Nachfolgesituation schauen und welche der Nachfolgesituationen hat den besten Wert. Und das wollen wir nicht ausprobieren, sondern das wollen wir einfach berechnen können. Durch die Zahlen unten wollen wir die Position gerade liegen. Okay. Die Position gerade liegen.

Also von den verschiedenen Zahlen, zum Beispiel 3 ist ganz unten liegen. Dann die zwei höheren Zahlen 7 und 6 sind schon oben. 1 ist in der Mittellinie und 2 ist auch schon oben. Also haben wir eigentlich am meisten Schritte, die 3 auf die 1, also auf dieerste Linie zu bringen.

Also du berechnest ungefähr, wie weit die Plättchen von ihrer Zielsituation wechseln. Ja, das wäre eine Heuristik, die man machen könnte. Andere Ideen?

Es gibt noch eine andere, die auch relativ einfach ist. Man zählt einfach die Plättchen, die nicht an der richtigen Stelle liegen. Und die zweite, also wie wäre diese Funktion hier? Die Plättchen, die nicht an der richtigen Stelle liegen. Welche Plättchen liegen an der richtigen Stelle von unserem Ziel?

7 liegt an der richtigen Stelle. Alle anderen liegen nicht an der richtigen Stelle. Also von 8 Plättchen liegen 7 nicht an der richtigen Stelle. Okay, so, das wäre eine Zahl. Warum? Okay, tun wir das mal, kommen wir gleich nochmal drauf. Die zweite wäre, dass man horizontal und vertikal den Abstand zum Ziel, das wäre eine Lösung.

Okay, es ist ein bisschen aufwendiger zu berechnen, weil hier muss ich nur gucken, welche liegt an der richtigen Stelle und habe ich eine Zahl. Das wäre jetzt ein bisschen aufwendiger zu rechnen. Wir müssten für jedes Feld gucken, wie weit es von seinem Zielzustand weg ist. Wie weit ist die 2 von ihrem Zielzustand weg? 1. 1.

Okay, wie weit ist die 5 von ihrem Zielzustand weg? 2. Wo muss sie hin? 1, 2, 3, 4. 2, 5, ja. 1, dann. Wie ist? 1. 1, sind wir bei 3. G?

Sind alle beteiligt? Auch 1. 1, sind wir bei 4. Die 8? Soll sie hin, soll hier hin? Ja. Also müssen runter und rechts wäre eine Möglichkeit, hierher zu kommen, sind 2. Dann sind wir bei 6, habe ich mich richtig noch im Kopf. Die?

Ist in der richtigen Stelle, ist 0. Die? 2. Ja, ja, 3. 1, 2, 3, ne? Sind wir bei 9, wenn ich mich nicht vertan habe? Und das? 1. 1. Müsste also 10 rauskommen. Ja, haben wir richtig gemacht.

Ja, also hier, die hat den Punkt, wenn man den Abstand von der richtigen Position berechnet, hätte es den Wert 10. Und wenn man die andere Messfunktion nehmen würde, hätte es den Wert 7. Aber ihr seht, wir brauchen nichts auszuprobieren, wir können das einfach berechnen.

Und wenn wir jetzt das anwenden auf unser Beispiel? Also hier, das oben ist die Ausgangssituation. Ja. Und wir haben drei Nachfolgeknoten. Welchen von diesen Nachfolgeknoten sollte jetzt als nächstes angeschaut werden? Entschuldigung.

Nehmen wir die Heuristik 1, also die Anzahl der Plättchen, die an der richtigen, die nicht an der richtigen Stelle liegen.

Also bei Position B, wie viele Plättchen liegen nicht an der richtigen Stelle? 3?

Also liegt die 1 an der richtigen Stelle? Ja. Liegt die 5 an der richtigen Stelle? Gehen wir so, liegt die 5 an der richtigen Stelle? Nein, sogar 1. Liegt die 2 an der richtigen Stelle? 2. Liegt die 4 an der richtigen Stelle? Ja. Liegt die 3 an der richtigen Stelle?

Sind wir bei 3? Liegt die 7 an der richtigen Stelle? Ja. 8 liegt an der richtigen Stelle? Die 6? Nein. Nein, also sind wir bei 4. Also dieses hat den Wert 4. Bei dem hier?

Auch 4. Die 1 liegt falsch, die 2 liegt falsch, die 3 liegt falsch und die 6 liegt falsch. Ja. Beim rechten? 2.

Also 1 und 2 liegen richtig, 3 liegt falsch, 4 und 5 liegen richtig, 6 liegt falsch, 2 liegen falsch. Welcher Knoten soll als nächstes expandiert werden? B. B. Also wir gucken einfach, welcher Knoten hat den geringsten Wert.

In dem Fall, wir wollen ja möglichst wenig Züge machen. Ja, also ist der beste Nachfolgeknoten wert D. So, dann würden wir nach D gehen. So.

Dann können wir einen davon wählen, dann spielt es keine Rolle. Ja, man muss sich noch merken, wir sortieren hier ein, wir haben niemand nichts weggeworfen, wir sortieren es einfach an der Stelle ein, das 3 steht weiter vorne, die beiden mit der 4 kommen danach.

Also wir dürfen es nicht wegwerfen, sondern es könnte ja sein, dass wir auf dem Weg, den wir jetzt gehen, doch nicht zu einem Ziel kommen. So, wir würden jetzt D anwählen. So, was ist die Bewertungsfunktion von von I?

Und was ist die Funktion?

Der Funktionswert von F?

F war der Weg, die Kosten bis zur Stelle, wo wir hingelangt sind, plus das, was wir abschätzen fürs Ziel, oder?

Also wir haben hier einen Schritt gemacht, also der Wert von G wäre 1 und der Wert von H wäre auch 1, weil hier eine Stelle falsch ist, also als Bewertungsfunktion haben wir H1 genommen, eine Stelle ist falsch.

Also wir nehmen G, einen Schritt, den wir gemacht haben, und wir schätzen, dass wir noch einen Schritt brauchen. Ja, also der Wert hier wäre eine 2 für diesen Wert I. Weil wir bisher an der Stelle, wo wir waren, einen gebraucht haben und wir würden noch einen mindestens brauchen, um zum Ziel zu gelangen.

Okay, so berechnen wir G plus H.

Okay, wir würden also als nächsten Wert D nehmen, natürlich dann gehen wir I, weil das der einzige Nachfolger von D ist. Und dann würden wir bei den Nachfolgern von I, weil sie sind ja noch nicht fertig, würden wir jetzt hier berechnen und hier berechnen. Also hier sind wir mit einer Stelle, mit zwei Stellen falsch.

Ja, wir haben es schlechter gemacht, weil wir die 5 verschoben haben. Und hier sind wir richtig, dann wären wir hier am Ziel, das wäre auch das Beste, weil hier 2 plus 1 mit drei Schritten wären wir am Ziel.

So, müssen wir noch nicht einfach automatisch immer I haben, dann zu S gehen. Nein, wir haben zwei Nachfolger. Und dann schätzen wir, was das Beste ist, und dann sehen wir, dass das das Beste ist, weil es keinen mehr an der falschen Stelle hat. Damit können wir das ja abschätzen.

Es ist keiner mehr an der falschen Stelle, also H1 ist 0. Und damit sind wir, würden wir den nehmen und sind am Ziel. Wir können natürlich erstmal schauen, sind wir schon am Ziel, bevor wir die Bewertungsfunktion berechnen, dann hätten wir das auch gleich. Okay.

Wir schauen uns das jetzt mal für unser Beispiel an mit dem Problemlösen von der Navigation.

Was brauchen wir jetzt noch an Zusatzinformationen? Was könnten wir als Zusatzinformationen nehmen, damit wir eine informierte Suche haben?

Es geht um Entfernungen, oder? Wir wollen den kürzesten Weg haben.

An Städten und Stationen. Anzahl Stationen. Die Kilometer, das wäre eigentlich eine sinnvolle Variante. Wir können abschätzen die Entfernung in Kilometern und wir können wissen, wie weit es von einer Stadt zur anderen ist. Ja, aber den Gesamtweg, den müssen wir immer berechnen.

Das habe ich mal hier auf der Folie drauf. Also ich habe jetzt hier die gleichen Verbindungen wie vorher, habe aber noch die Kilometer hinzugefügt.

So, also das Ziel ist jetzt nicht einfach irgendeine Verbindung zu finden, sondern die kürzeste Verbindung. Und jetzt müssen wir uns wieder überlegen, was sind G und H?

Wenn wir an irgendeinem Knoten sind, zum Beispiel in Olten, ja, was wäre für Olten der Wert von G und was können, wie könnten wir den Wert von H abschätzen?

Also ich habe das, glaube ich, auf der nächsten Folie draufstehen. Also wie berechnet man G und was wäre eine geeignete Funktion, um die Entfernung abzuschätzen? Ja, wir wollen es nach Olten machen. Ja, wir wollen es nach Olten machen, wie immer. Und angenommen, wir wären jetzt über Zürich in Olten angelangt.

Ja. So, wie können wir für Olten den Wert F berechnen? Dafür brauchen wir zwei Teile. Wir brauchen G und wir brauchen H. Wir sind von Zangai nach Olten gekommen.

Was wäre ein Wert für G und was wäre ein potenzieller Wert für H?

Genau, das wissen wir. Also die Entfernung von Zangai nach Olten wissen wir. Das können wir einfach die Entfernung zusammenrechnen. Wenn ich mich nicht verrechne, ist das jetzt 152. Ja, so. Und jetzt müssen wir eine Abschätzung finden für den restlichen Weg. Wie können wir das abschätzen?

Das ist nur eine Abschätzung. Das können wir nur direkt hier berechnen. Also was wäre jetzt in so einem Beispiel eine sinnvolle Abschätzung?

Gut, Entfernung. Über welchen Weg?

Ja, genau. Luftlinie. Wäre Luftlinie eine zulässige Bewertungsfunktion?

Was ist zulässig?

Wollen wir auf einer Folie drauf?

Zulässig überschätzt nie die richtigen Werte. Also es ist immer geringer oder gleich dem richtigen Wert. Wäre die Luftlinie eine zulässige Bewertungsfunktion?

Ja, wir können keinen schnelleren Weg, keinen kürzeren Weg finden als Luftlinie, richtig? Deshalb ist das eine gute Funktion. Sie wird uns helfen. Was haben wir davon, wenn wir eine Bewertungsfunktion haben, die zulässig ist? Schneller?

Ja. Ich kann den Weg besser abschätzen. Wenn es näher an dem Wert ist, also wenn es Optimum, die Luftlinie ist, der Wert von der Linie, wo er am nächsten ist, wäre dann einfach. Ja, stimmt. Also wir wissen, dass der Algorithmus optimal ist. Genau.

Der Algorithmus ist optimal, er findet tatsächlich den kürzesten Weg. Wenn wir die Bewertungsfunktion haben, die die richtige Funktion nie überschätzt, dann finden wir auch den kürzesten Weg.

Und mit Luftlinie haben wir eine Bewertungsfunktion gefunden, die die richtige Entfernung nie überschätzt. Und damit würde der Stern-Algorithmus tatsächlich den kürzesten Weg nach Lausanne finden.

Gut, also zurückgelegte Wegstrecke ist G. Die Abschätzung per Luftlinie zum Ziel ist die Bewertungsfunktion H. Und wenn wir die anwenden in jedem Punkt, dann nehmen wir als nächsten Punkt tatsächlich den, der zum Ziel führt. Also wir finden auf jeden Fall die optimale Lösung zum Ziel.

Dann scheint es nochmal zurück müssen, aber wir finden die kürzeste Weg. Es ist nicht so, dass wir ihn auf dem direkten Weg finden, aber wir finden ihn mit relativ wenig Aufwand, mit einem definitiven optimalen Lösen. Das sagt der Stern-Algorithmus. Das, was wir jetzt gerade gesagt haben.

In jedem Knoten berechnet die Funktion F aus G und H, nimmt den kleinsten. Und wenn wir das immer machen, dann finden wir den optimalen Weg. Bitte?

Genau, wir können, wir berechnen ja hier immer die Funktion F. So, jetzt würden wir zum Beispiel, in Luftlinie wäre das der nächste, der Beste.

Ja, so, aber da geht es gar nicht weiter. So, dann wollen wir hier nicht weiterkommen, müssen wir zurückgehen und dann dort wieder schauen, was war der zweitbeste nach diesem Modell. Dann würden wir den Weg nehmen. Also es kann schon sein, dass wir auf einen Pfad gehen, wo wir nicht zum Ziel kommen. Oder wo plötzlich die Entfernung größer ist. Dass wir doch einen Umweg machen.

Ja, und dann plötzlich wird die Entfernung größer und dann würden wir auch zurückgehen. Entweder, weil wir auf dem Weg nicht weiterkommen oder weil wir jetzt irgendwo einen Umweg gemacht haben.

Das heißt, wenn es nicht weiterkommt, dann ist es ja wie in sich stehen, wird die Option dann gestrichen und fertig. Ja, genau. Also wir könnten eine Sackgasse fahren oder so, oder ja, wenn eine Baustelle ist da, also ich versuche es auch, reale Sachen anzuwenden.

Und dann würden wir zwar irgendwo hinfahren und sagen: "Oh ja, das geht nicht, das geht heute nicht oder geht generell nicht." Und dann müssten wir den Weg zurückgehen. Das wäre dann, es geht gar nicht weiter.

Oder wir finden einen Pfad, wo wir sagen: "Okay, wir sind, Basel liegt zwar luftliniennäher an Lausanne, oder sagen wir so, Luzern liegt vielleicht luftliniennäher an Lausanne, aber von Luzern hätten wir einen relativ langen Weg, um nach Lausanne zu

kommen." Dann würden wir vielleicht zuerst Luzern anschauen, aber dann gibt es keinen direkten Weg nach Lausanne, sondern nur auf Umwegen. Und dann sagen wir: "Nein, das war doch nicht ein guter Weg." Dann gehen wir zurück und sagen: "Okay, Olten war vielleicht besser." Das heißt, er würde verschiedene Parameter nehmen in seiner richtigen Sichtlinie, oder? Nein, er würde nur die zwei nehmen.

Er würde nur die tatsächlich zurückgelegte Entfernung und die abgeschätzte Entfernung. Mehr braucht er nicht.

Also, damit ich das richtig verstanden habe, würde er weiter richtig kursieren, wenn er jetzt von Olten Richtung Bern und dann, also von Olten nach Solothurn, dann nach Neuenburg gehen würde, würde er dann zurückgehen?

Ja, wenn er merkt, Neuenburg hat jetzt plötzlich einen schlechteren Wert als ein anderer, nur offener Punkt, und er sagt dann: "Ja, Neuenburg kann es nicht sein, ich gehe zurück und suche einen alternativen Pfad."

Gut, das war jetzt über die Entfernung. Ja, angenommen, wir wollen andere Sachen finden. Es geht uns nicht nur um die Entfernung, sondern es geht auch noch um die Fahrzeit und Energieverbrauch. Was würden wir dann machen?

Die Gerichtung, also Energieverbrauch oder Fahrzeit sind ja wieder zwei verschiedene Gerichtungen, und dann hinten dann Baustellen rein, Haltestopps und Zeit, alles verschiedene Parameter.

Wir können G zum Beispiel sagen, wir nehmen die Entfernung und die berechnen wir noch mit der Zeit, ja, und haben dann einen anderen Wert für G. Also wir können andere Sachen machen, nur die Zeit, nur die Entfernung, nur Energieverbrauch, oder wir können das auch miteinander kombinieren.

Dann müssen wir alles berechnen in G, und für alle drei müssten wir eine Abschätzung machen für den Restweg. Und das machen so Systeme wie die, die wir heute als Navigationssystem haben.

Da kann man ja wählen: Will ich den schnellsten Weg haben, will ich den kürzesten Weg haben, will ich den Weg haben mit dem wenigsten Energieverbrauch. Das sind drei Sachen, die man typischerweise wählen kann, und da wird einfach die Bewertungsfunktion, die heuristische Funktion, geändert.

Ich schätze den Energieverbrauch ab, oder ich schätze die Entfernung ab, oder ich schätze die Zeit ab.

Und Zeit kann man zum Beispiel abschätzen, dass man sagt: "Was ist die Höchstgeschwindigkeit, die das Auto hat?" Und dann sagt man: "Da ist man immer drunter, weil schneller kann man nicht fahren." Ja, und also man könnte auch die Höchstgeschwindigkeit auf der Straße nehmen, aber manche Leute fahren halt ein bisschen schneller. Dann hätte man keinen optimalen Weg.

Ja, dann wird es nicht immer korrekt abschätzen. Also muss man einen anderen Parameter finden, der zulässig ist, damit man immer den optimalen Weg findet. Und bei Energieverbrauch eben auch: Was hat ein Auto an Energieverbrauch? Nimmt man ein bisschen was drauf, und dann kann man den Energieverbrauch abschätzen.

Gut, das war Problemlösen, dieser Stern-Algorithmus. Wie gesagt, er findet den optimalen Weg, und wenn ich kein zusätzliches Wissen habe über mein Problem, wäre das eine Variante, immer eine optimale Lösung zu finden.

Machen wir eine kurze Pause bis 20 nach, und dann gehen wir über zur Logik und sagen, wie können wir noch zusätzliches Wissen verwenden, um Probleme zu lösen.

Entschieden, die Mail, die ich schon letztes Mal erwähnt habe, mit dem CTS-Punkt weiterleiten, weil Herr Weiss hat mir gesagt, ich soll es dem Professor Dr. Ivan Klöhl weiterleiten. Aber ich glaube nicht, dass der von meinem Studiengang ist. Okay, um was ging es nochmal?

Wegen den ECDS-Punkten oder vom Militär. Ah, ob die anrechenbar sind. Oh nee, das ist das.

Warum hat er das an Ivan schicken wollen? Vielleicht. Es kann sein, dass er der Beauftragte ist für diesen Militärdienstzeugs. Okay.

Hast du es mir auch geschickt? Nee, noch nicht. Dann schick mir es mal. Ich gucke mal an, ob ich das finde, aber ansonsten schicke ich es an Ivan weiter. Okay, super. Danke vielmals.

Okay, machen wir weiter. Wir haben jetzt angefangen mit den uninformierten Suchen, da hatten wir eigentlich keine Informationen, was jetzt uns helfen würde, um ein Problem zu lösen. Mit dem A-Stern haben wir zumindest eine heuristische Bewertungsfunktion gehabt, und ich will jetzt noch einen Schritt weitergehen.

Und zwar machen wir ein kleines Beispiel. Also hier habe ich ein Schachbrett, ja, und bei diesem Schachbrett sind zwei gegenüberliegende Ecken. Die habt ihr noch nicht, weil ich die Lösung noch nicht verraten will.

Bei dem Schachbrett sind zwei gegenüberliegende Ecken rausgeschnitten. So, und dann habe ich Dominosteine, die sind da oben.

Ich habe genügend Dominosteine, und so ein Dominostein besteckt zwei Felder, ja, und die Frage, die wir haben: Ist es möglich, mit dem Dominostein alle Felder des Schachbretts abzudecken?

Wir haben 31 Dominosteine, können wir mit 31 Dominosteinen alle Felder des Schachbretts abdecken?

Überlegt mal, und dann sagt mir, ob ihr eine Lösung habt.

Also es bleiben 62 Felder übrig, deshalb müssen 31 Dominosteine rein.

Okay, und du meinst, es funktioniert? Ja. Okay.

Den Reihen wird es ja dann irgendwo nicht so bei 17 Beispielen müsste auch. Ja, aber da kann man ja in die nächste Zeile gehen, oder? Genau. Also. Geht nicht. Geht nicht, warum nicht?

Weil, wenn man sie anordnen würde, auch wenn man sie so quer und dann wieder hochstellen würde, würde es bei einer Seite nicht aufgehen. Mhm.

Also man spaltet sie. Nee, das geht nicht. Ja. Also man darf das X nicht überschreiben, richtig? Genau. Das geht eben.

Wenn ich sie jetzt so anordnen würde, also ich bin jetzt kurz durchgegangen, aber wenn ich sie jetzt so anordnen würde und dann auf der linken Seite blöde ist, also wenn ich von unten anfangen würde und dann jetzt von der linken Seite, also das letzte so hochstellen würde, bis nach oben gehen würde, würde das linke Feld oben frei bleiben, oder ich müsste so einen Stein legen

und das würde dann auch gehen. Gut, wir können ja mal gucken, ich habe mal gesagt, wir machen das mal mit Tiefensuche, oder? Dann, wenn man das so macht, dann kommt man so hin. Ja.

Okay, man ist da. Ja. Dann macht man weiter und es funktioniert nicht. Aber wir haben ja Tiefensuche, es kann ja sein, dass wir einen Weg haben, der nicht funktioniert.

Jetzt können wir ja einen Schritt zurückgehen, ja, und versuchen eine Alternative, also wir gehen hier nach unten, das funktioniert auch nicht, aber es könnte ja sein, dass wir oben irgendwas nicht so gut gemacht haben, dass wir nicht bis ans Ende einer Zeile hätten gehen müssen, sondern vorher abbrechen und dann wären wir wieder zurückgekommen.

Das ist ja nur ein Weg. Der hat sich bisher nicht gefüllt. Wer glaubt denn insgesamt, dass es funktioniert? Könnt ihr hochheben, dass ihr es seht? Okay. Wir? Wer glaubt, dass es nicht funktioniert? Mehr, okay.

Können wir Argumente finden? Du hast jetzt so ein bisschen, okay, du hast ein bisschen ausprobiert und du hast noch nicht den Weg gefunden, aber es sind relativ viele Varianten, die man haben kann. Ja, man kann es ja auch so schräg setzen.

Übereinander hin, so. Ja, also nach unten, dann geht es nächstes nach unten und wieder rüber, also das ist dann so wie immer. So ein Treppenlauf. So, ja, genau. Genau, das wäre auch eine Variante.

Wird man bei der Tiefensuche irgendwann auch dort ankommen? Wenn es eine Möglichkeit gäbe, dann denke ich eher, dass es das wäre. Aber ich denke nicht, dass das aufgehen würde. Okay. Jetzt sind wir noch beim Glauben oder Denken. Können wir Argumente finden?

Könnten wir mal mit Wissen hingehen? Welches Wissen könnte uns helfen, das Problem schnell zu lösen? Und das war ausprobieren müssen.

Jeder Stein hat direkt zwei Felder. Mhm. Also Flächenabdeckung vielleicht. Ja? Ein bisschen größer. Mit Flächenabdeckung haben wir schon einen Teil davon, ja.

Einfädeln, schrauben und weiter. Wenn die geschraubt sind, werden. Cool. Und was hilft uns das? Und dann... Wenn die nebeneinander wären, zum Beispiel.

Ja, im Prinzip, was du sagst, ist richtig, es geht um schwarze und weiße Felder. Weil dann muss es ja ein schwarzes und ein weißes bedecken.

Genau, jetzt kommt deine Aussage, es sind mehr weiße als schwarze, aber wir haben am Ende zwei an den gegenüberliegenden Ecken zwei Felder weggenommen, das sind zwei schwarze.

Ein Dominostein kann immer nur zwei nebeneinander liegende Felder belegen, das ist immer ein schwarzes und ein weißes, also es funktioniert nur, wenn wir gleich viele schwarze wie weiße Felder hätten. Also wir können sicher sein, es funktioniert nicht. Gut.

Okay, das ist ein bisschen über das Problem. Also wir wollen entscheiden, ob das Brett mit dem Dominostein vollständig belegt werden kann und wir wissen, dass jeder Dominostein zwei benachbarte Felder belegt und benachbarte Felder unterschiedliche Farben haben.

Also lässt sich das Feld mit dem Dominostein nicht vollständig belegen. Dafür müssten die Felder gerade sein und die gleiche Anzahl Farben haben. Ich habe noch ein paar Beispiele hier. Wieso ist das doppelt? So. Das war unser Beispiel. Okay. Und wir haben gesagt, das funktioniert nicht.

Wir haben die beiden Felder weggenommen. Was bei diesem Beispiel? Wenn ich die Felder wegnehmen würde.

Nein, warum? Wenn ich es einfärben würde, hätten sie die gleichen Farben. Diagonal voneinander sind die gleichen Farben. Was bei dem hier? Ja.

Springer geht immer von einem Weißen zum Schwarzen, vom Schwarzen zum Weißen, ja. Also es sind zwei unterschiedliche Farben. Das könnte man lösen. Was mit dem hier?

Ja, was das dreht, der heißt ja einfach nicht, dass es nicht zwingend nicht gehen müsste, aber ich weiß ja nicht, dass es trotzdem geht. Okay, also hier können wir... Ja, stimmt. Es ist lösbar, aber wir... Ja, ich hätte es vielleicht anders formulieren sollen.

Es ist nicht lösbar und dann hätte man hier... Ja, stimmt. Aber es ist nicht, dass es sich verkehrt, sondern dass es... Genau, ich habe einen Zwischenschritt gemacht. Es ist lösbar, wenn es gleichfallende Schwarz wie Weiß hat.

Gucken wir mal das noch an. Ja.

Ja, es wäre lösbar, es sind zwei gleiche, zwei verschieden farbige Felder, wenn man es einfärben würde. Was mit dem Beispiel? Ja.

Ja, also ich habe jetzt die Größe des Feldes angepasst, ja, und auch dort gilt es. Es ist nicht nur bei 8 mal 8. Ja, was mit diesem Beispiel? Ja.

Alle ja? Nein.

Warum funktioniert es da nicht? Weil die Größenanordnung anders ist. Was heißt Größenanordnung? Oben ist weniger weiter, also oben unten hat es da weniger Felder als links und rechts. Es ist ein Quadrat. Was ist das?

Der Anzahl von Feldern, also sowieso schwarz und weiß. Genau, 9 mal 9 sind 81 Felder und damit geht es nicht. Also was haben wir jetzt zu sagen, ob es lösbar ist? Zwei Bedingungen, oder?

Müsste gerade Anzahl von Feldern sein und die Felder, die wir weggenommen haben, wenn wir es einfärben würden, müssten verschiedene Farben haben. Gut. Okay, so wie kann man jetzt Sachen aus dem Algorithmus formulieren, wenn man dieses Wissen hat?

Jetzt hört alle an, Python programmieren, oder?

Zwei Mems, also ein Mem mit einem Mems.

Ja, Punkt ist schon mal das Els. Also ich habe es mal hier gezeigt, das war ja die Bedingung, die wir haben, und der Algorithmus könnte es so zeigen. Ich zähle zuerst, wenn die Anzahl der Felder ungerade ist, dann ist es nicht lösbar.

Els, dann sind wir ja, Felder sind gerade, ja, und dann wird das fehlende Feld auf, wenn die fehlenden Felder auf einer Diagonale liegen, das ist jetzt der Algorithmus, wir müssen ja irgendwie checken, ob die beiden Felder die gleiche Farbe haben. Wenn sie auf einer Diagonale liegen, dann haben sie die gleiche Farbe.

Dann können wir sagen, es ist lösbar, oder es gibt zwei Diagonalen, die man anlegen kann, die sich in einem Feld schneiden. Dann wäre es auch lösbar. Also sie müssen nicht auf einer Diagonale liegen, aber die Diagonalen, die sie haben, die müssen ja gleiche Farben haben, unterschiedliche Farben haben.

Und wenn sie ein gemeinsames Feld haben, dann sind sie die gleiche Farbe.

Dann sollen die durch ein Fehlen Felder mit einem gemeinsamen Feld, ja, dann ist es nicht lösbar, sorry, das wäre nicht lösbar und ansonsten ist es lösbar. Okay.

Und ja. Sind da mit einem End, also wenn die zwei Funktionen, also wenn die zwei Werte korrekt sind, dann ja, dann geht es um solche. Also müssen wir beide Funktionen sicher sein. Ja, das haben wir ja.

Zuerst testen wir, ob es gerade oder ungerade ist, und für gerade machen wir dann den anderen Test, ob es gleiche Farben hätten. Wenn gerade und Beispielfelder, dann ja. Aber wir müssen auch testen, wann Felder gleich sind.

Und dafür gibt es zwei Bedingungen, nämlich dass es auf einer Diagonale liegt oder dass wir zwei Diagonalen, die sich schneiden, in einem Feld. Sie müssen ja nicht auf einer Diagonale liegen. Das sind die zwei Arten, wie man es berechnen kann. Deshalb ist es nicht einfach nur mit einem Und.

Warum ich das gezeigt habe, ist einerseits, wir haben einen Algorithmus, aber wir müssen uns da auch überlegen, in welcher Reihenfolge wir diese Tests machen. Es ist festgelegt. Aber der Algorithmus ist gut, weil er ist unabhängig von der Größe des Feldes.

Und wir müssen noch nicht mal gucken, ob es eingefärbt ist oder nicht. Wir müssen nur schauen, ob es auf der gleichen Diagonale liegt oder zwei Diagonalen, die sich schneiden. So, das ist das, was wir in der Programmierung machen.

Und was wir in der Wissensrepräsentation machen, ist eigentlich, ja, wir wollen nicht uns überlegen, wie man jetzt den Ablauf macht, was muss ich zuerst testen und dann. Also das ist jetzt, sorry, das war, was ich zeigen wollte. Wir können das als Regel formulieren.

Wenn die Anzahl der Felder gerade ist und fehlende Felder haben verschiedene Farben, dann ist das Problem lösbar. Jetzt müssen wir aber sagen, was sind verschiedene Farben? Felder auf einer Diagonale haben die gleiche Farbe, mehr eine Regel.

Die andere Regel sagt, Felder auf zwei Diagonalen und die Diagonalen haben ein gemeinsames Feld, ist auch gleiche Farbe. Und wenn es gleiche Farben hat, wenn es nicht gleiche Farben hat, dann sind es verschiedene Farben.

So, das sind vier Regeln, die wir aufschreiben können, die wir als eine Wissensbasis aufschreiben können, die können wir auf jedes Problem anwenden und wir müssen uns nicht einen Algorithmus überlegen, in welcher Reihenfolge wir irgendwas testen. Also so kommen wir auf den Weg zur Wissensrepräsentation.

Also wir haben Regeln, die wir auf beliebige Feldgrößen anwenden können, wo auch das Brett nicht eingefärbt sein muss, aber wir brauchen uns auch nicht zu überlegen, in welcher Reihenfolge wir diese Regeln testen. Das ist das, was wir eigentlich machen.

Und wenn man mal so historisch betrachtet, was eigentlich wir mit Programmierung machen. Wir haben angefangen mit der Programmierung, da hat man Maschinencode gehabt, ja.

Und wir wollen eigentlich von dem wegkommen, dass wir dem Computer sagen, was er machen soll, das möglichst nah an dem ist, wie wir denken. Ja, also das nächste, was man gemacht hat, war Assembler-Sprache. Kennt jemand Assembler? Schon mal davon gehört?

Du hast gehört? Gehört, aber ja. Da hat man gesagt, es gibt Register und man steckt Werte in Register, kann zwei nachfolgende Register miteinander verrechnen und so. Ich habe das noch gemacht in

meinem Studium und mein Prof hat mir damals schon gesagt, wir machen das Assembler nur, damit ihr wisst, wie schön es ist, dass wir höhere Programmiersprachen haben.

Das ist nämlich der nächste Schritt. Wir können höhere Programmiersprachen, da haben wir IFS und ELSE, wir haben Weitschleifen und so weiter, wir können damit programmieren. Und der nächste höhere Schritt war dann, wir haben eine objektorientierte Programmierung. Und eigentlich ist das hier auch etwas.

Wir wollen dem Computer sagen, was er tun soll, ohne dass wir wissen müssen, wie ein Computer funktioniert. So, also auf Basis von Logik, also wir haben hier vier Regeln, die wissen wir, die gelten, und jetzt soll der Computer schauen, wie er die Regeln abarbeitet. Ja, das ist das, was wir eigentlich unter Wissensrepräsentation verstehen.

Wir stellen das Wissen so dar, wie wir es einfach verstehen können, und wir müssen uns nicht darum kümmern, wie der Computer das abarbeitet. Das macht die Inferenzkomponente von dem Computer. Okay, aber wir sind deutlich weiter, als wenn wir nur mit Daten arbeiten müssten.

Egal, wir können jedes neue Problem mit diesen paar Regeln lösen. Und wenn ich das jetzt auf was anderes anwenden will, brauche ich nicht jede Menge Daten, mit denen ich das auch noch trainiere, sondern ich gebe jedem diese vier Regeln. Und das ist, wie wir auch lernen oder wie wir auch Wissen weitergeben.

Wir haben was verstanden, und weil wir es verstanden haben, können wir den anderen auch was erklären. Und der andere kann das auch anwenden. Okay, und jetzt, das ist der Grund, warum wir jetzt uns als nächstes mit Wissensrepräsentation und dort insbesondere mit Logik beschäftigen. Das ist die Art, wie wir das Wissen darstellen können. Gut.

Die Folien habt ihr jetzt wieder. Die nächsten.

Also wir beginnen jetzt, machen eine kurze Einführung in Logik. Logik ist die Basis, um Wissen abzubilden für die meisten Systeme. Die Frage ist jetzt, was ist Wissen?

Haben wir das schon mal gehabt? Diskussion? Glaube nicht, oder? Was ist Wissen? Daten?

Geschichte. Geschichte. Wie meinst du das mit Geschichte?

Ah, okay. Wir haben Wissen, weil wir Erfahrungen haben, etwas von der Vergangenheit, aufgrund dessen wir uns das Wissen angeeignet haben. Okay. Ja, das stimmt.

Das ist ja dann die Anwendung von Wissen, oder?

Was ist das Wissen, das uns die richtige Antwort liefert?

Also es gibt verschiedene Arten, wie man Wissen definieren kann. Also, aber es gibt zwei wichtige Unterscheidungen. Das eine in der Philosophie, da unterscheiden wir Wissen von Meinen oder Glauben. Also Wissen ist etwas, was man überprüfen kann, was wahr ist oder nicht wahr ist. Man kann es überprüfen.

Das ist so nach dem Satz, ja, weißt du es wirklich oder meinst du nur? Oder? Das ist in der Philosophie. Das ist nicht das Wissen, das wir interessiert sind. Wir sind an Alltagswissen interessiert. Warum? Weil wir wollen handeln, wir wollen etwas tun. Und das Wissen soll uns helfen, zum Beispiel ein Problem zu lösen.

Also es hat irgendwas mit Ursache-Wirkung-Zusammenhängen zu tun, Zusammenhängen, die wir erkennen, aufgrund derer wir dann Probleme lösen können. Das ist das Wissen, das uns interessiert. Das war gerade eben auch. Wir haben gewusst, wenn wir zwei schwarze Felder wegnehmen, dann können wir das Problem nicht mehr lösen. Das ist Wissen, dass wir das nicht mehr machen können.

Befähigt uns zum Handeln, befähigt uns, ein Problem zu lösen. Also wir haben ein Problem und jetzt haben wir Wissen und können mit diesem Wissen ein Problem lösen. Jetzt hast du noch gesagt, Wissen sind Daten. Was ist der Unterschied zwischen Daten und Wissen? Hast du das mal gemacht?

Vorhandenes Wissen kann man immer noch aneignen. Mhm, Daten ist das, was vorhanden ist, ja? Daten haben keine Bedeutung, das Wissen kann sich, darf man selbst nutzen. Okay, das ist richtig, Daten haben noch keine Bedeutung.

So, ich habe mir das mal so gemacht. Also, wenn ich das hier aufschreibe, was bedeutet das?

Oder mehr, dass die Notis mir nicht so hochgeraten ist, dass ich den extra darf.

Okay, gut, ja. Das könnte eins bedeuten, auf den Gedanken nicht umgangen gekommen. Vier und null ist eins. Was kann es noch bedeuten? Vier Grad. Vier Grad, ja. Wenn es vier Grad ist, was bedeutet das?

Vier Grad. Könnte Temperatur sein, oder? Ja, oder ein Winkel. Oder ein Winkel, richtig. Also, das sind Daten, hat noch keine Bedeutung. Oder was ist, wenn ich das hier schreibe? Datum.

Datum, ja. Was bedeutet dieses Datum? 5. August 2025. 5. August 2025. Muss aber nicht sein, kann auch im Englischen umgekehrt sein. Genau, es könnte auch der 8. Mai, 8. Mai 2025.

Daten sind die Sachen, die wir irgendwo aufnehmen können, die wir in Sensoren haben, zum Beispiel wie vier Grad, ja. Aus den Daten machen wir Information. Was sind Daten?

Informationen, indem wir die Daten interpretieren. Dann bekommt die Daten erst einen Sinn.

Ja, also wenn ich sage, das ist eine Temperatur, ja, also zum Beispiel sagen draußen, ist es vier Grad, ja, dann hat das Datum einen Sinn. Es ist eine Temperatur.

Hier kriege ich einen Sinn, indem ich sage, das ist die europäische Schreibweise oder es ist die amerikanische Schreibweise. Dann bekomme ich hier einen Sinn draus. Das ist der 5. Mai oder der 8. August. Sorry, der 5. Mai oder der 5. August.

Gut, und ein anderes Beispiel. Das habe ich auch noch nicht gemacht. Was bedeutet das? Engel. Oder Angel. Oder Angel. Die Interpretation kommt durch die Sprache. Ist das jetzt ein englisches Wort oder ist das ein deutsches Wort?

Oder Spanisch. Okay, ja, anscheinend, ja, aber da kann ich jetzt nicht. Aber die Bedeutung kommt durch die Sprache. Also von Daten zu Informationen kommen wir durch eine Interpretation.

Was ist jetzt Wissen?

Wissen wäre jetzt etwas Zusammenhänge, die mir helfen, irgendein Problem zu lösen. Ich könnte zum Beispiel als Wissen haben, dass vier Grad kalt ist.

So, das heißt, wenn es draußen vier Grad ist und ich weiß, dass vier Grad kalt ist, sollte ich mir vielleicht einen Jacken anziehen, wenn ich nach draußen gehe. Also das ist Wissen, das mir hilft zu handeln, damit ich nicht frieren muss. Also könnte man dann eigentlich auch eher davon ausgehen, dass es dann in englischer Schreibweise ist und nicht in europäischer Bildgesellschaft.

Genau, hier würde ich interpretieren, dass dasähnlich ist. Dann würde ich sagen, das ist der 8. Mai. Und dann kann ich hier vielleicht sagen, im Mai haben wir irgendwie Termine oder irgendwas. Ich kann mit dem dann handeln, wenn ich weiß, was das richtige Datum ist. Wenn ich das Datum nicht kenne, kann ich auch nicht handeln.

Aber ich meine jetzt, wenn ich jetzt weiß, wie viel Grad Celsius es ist, aber das Datum nicht genau interpretiert habe, dann kann ich da davon ausgehen, dass es im Mai gewesen ist. Also ich habe die beiden unabhängig voneinander gesehen. Aber sonst ja.

Damit ich weiß, an welchem Tag ich eine Jacke brauche, sollte ich dann auch auf das Datum schauen. Okay, also das ist ein wichtiger Punkt. Wir unterscheiden Daten von Wissen. Und wenn wir ein maschinelles Lernen machen, geben wir dem System Daten, und das kann daraus Zusammenhänge ableiten.

Und oft geben wir dem System auch wirklich nur Daten, und das muss auch sogar schon die Information ableiten, was die Bedeutung von diesen Daten ist. Das habe ich mal gemacht in einem Projekt. Damals gab es dort eine schweizerische Bank, und die haben Kreditvergabe gemacht, und die haben uns Daten gegeben und haben uns nicht gesagt, was diese Daten bedeuten.

Also sie haben alles anonymisiert. Also alle Attribute, die sie hatten, waren anonymisiert. Und wir sollten eine Klassifikation berechnen, ob man jemand in der Firma Kredit geben kann oder nicht, ob die kreditwürdig ist oder nicht.

Und es war verdammt schwierig, weil wir keine Information hatten, was diese Daten bedeuten. Und nachdem sie uns auch diese Information gegeben haben, was die Daten bedeuten, konnten wir eine bessere Klassifikation machen.

Unterschied zwischen Daten und und Information übrigens kennt man auch. Es geht ja um die Interpretation. Wenn ihr einen JPEG-File zum Beispiel mit Word öffnet, was passiert dann?

Oder mit einem Notebook irgend so einem anderen. Das ist irgendwie ein komischer Zeichensalat, oder?

Ein JPEG oder Ping, also ein Bild, eine Bilddatei mit einem Editor. Da kriegt er irgendeinen komischen Zeichensalat. Das ist die falsche Interpretation. Ja, weil Word erwartet, dass ihr einen Textfile habt, Notebook auch, aber JPEG ist kein Textfile. Kriegt er auch eine falsche Interpretation.

Also das zeigt den Unterschied zwischen Daten und Information. Und dann drüber haben wir Wissen, dass uns hilft, Probleme zu lösen. Und auch wenn wir Korrelationen haben zwischen Daten, was wir ja beim maschinellen Lernen haben, ist noch nicht unbedingt Wissen, weil Korrelation nicht unbedingt eine Kausalität ist.

Damit wir handeln können, müssen wir wissen, was ist eine Folge von der Tätigkeit. Und dafür müssen wir es eigentlich verstanden haben. Jetzt kommen wir wieder zu unserem Agenten. Wenn wir einen wissensbasierten Agenten haben, haben wir Eingabe-Ausgabe. Wir haben hier als Modell unsere Wissensbasis.

Da haben wir das letzte Mal gehabt. Ja, und damit wir die Wissensbasis verarbeiten können, haben wir noch eine Inferenzkomponente. Darauf wollen wir jetzt raus. Also wie, wenn ich so einen intelligenten Agenten habe, der eine Wissensbasis hat, muss ich erstmal die Wissensbasis erstellen.

Also ich muss das Wissen bekommen von irgendjemand, damit ich es repräsentieren kann. Wenn ich den Agenten dann anwende oder der Agent wirklich agiert, dann kriegt er eine Eingabe, wendet das

Wissen mit der Inferenzkomponente an, um eine Ausgabe zu generieren, und mit der kann er dann Aktionen auslösen.

Aber Wissen ermöglicht uns Aktionen, ermöglicht uns zu handeln. Die Frage ist, wie kann ich jetzt dieses Wissen darstellen? Und da setzen wir jetzt Logik ein, weil wir immer eine natürliche Sprache haben.

Da können wir einen Satz, einen kleinen Sachverhalt auf unterschiedliche Arten nennen. Also heute ist Dienstag, der heutige Tag ist Dienstag, Dienstag ist heute, heute ist Mittwoch, alles eigentlich die gleiche Aussage. Aber ganz anders ausgedrückt.

Und damit wir mit dem System gut arbeiten können und damit es automatisch verarbeitet werden kann, versuchen wir, eine einheitliche Darstellung von diesen Aussagen zu finden und Gemeinsamkeiten zwischen Aussagen zu finden.

Und Logik ist so eine Art, wie man das Wissen darstellen kann, ist eine Form der Wissensrepräsentation und ermöglicht uns, Verfahren zu entwickeln, mit denen wir aus dem Wissen wieder neues Wissen herleiten können, um zum Beispiel Probleme zu lösen. Und es gibt verschiedene Arten von Wissen.

Beziehungswissen, also zum Beispiel der Motor ist Teil von einem Auto oder zwei Personen, Peter und Maria sind verheiratet, Peter ist der Vater von Clara, das sind Beziehungen zwischen Objekten. Dann haben wir logisches Wissen.

Also wenn Peter der Vater von Clara ist und Clara weiblich ist, dann ist Clara die Tochter von Peter. Und wir können auch prozedurales Wissen haben. Also wenn der Tank leer ist, dann fahre ich zur Tankstelle. Das ist eine Prozedur, sagt, was eine Aktion ist, die wir ausführen wollen. Das sind verschiedene Arten von Wissen.

Und wir gucken uns das jetzt mal an, wie man dieses Wissen darstellen kann. Also Logik erlaubt uns, dieses Wissen darzustellen. Typischerweise verwenden wir das, um Regeln darzustellen oder Wissensgraphen. Und die Logik, die wir uns anschauen, ist Aussagenlogik und Prädikatenlogik.

Wer kennt das schon, Aussagenlogik? Habt ihr das irgendwann schon mal gemacht? In Mathe, irgendwann in der Schule. Du kennst das? In Mathe, ja. Logik ist ein Teilgebiet der Mathematik, aber viel spannender als viele andere Teile. Ja, das stimmt nicht, ich mag auch Mathematik sonst.

Aber was ist generell Logik? Wir verwenden den Begriff häufig, sagen, das ist logisch. Was ist denn eigentlich Logik? Rationales Denken. Genau, rationales, korrektes, folgerichtiges Denken, das ist Logik.

Und dafür gibt es einige Prinzipien, die gelten, und die wollen wir uns anschauen und schauen, wie man das für die Wissensrepräsentation verwenden kann. Ich habe gesagt, es gibt zwei Arten von von Logik, mit denen wir arbeiten. Es gibt die Aussagenlogik, die beschäftigt sich mit diesem Teil.

Es gibt und, oder, Implikation, nicht. Das sind Operatoren, mit denen wir logische Aussagen verknüpfen können. Und dann gibt es die Prädikatenlogik, das ist eine Erweiterung davon, sodass wir über einzelne Objekte reden können.

Also für alle Personen gilt, dass sie sterblich sind. Dann haben wir über einzelne Personen oder über alle Personen können wir etwas sagen, oder es gibt eine Person, die bestimmte Eigenschaften erfüllt.

Und das ist eine Form der Darstellung, die wir dann nachher erweitern, also Erweiterung von Aussagenlogik anschauen. Und wir gucken jetzt heute zuerst mal die Aussagenlogik.

Was ist Aussagenlogik? Aussagenlogik beschäftigt sich mit Aussagen. Und eine Aussage ist entweder wahr oder falsch. Es gibt nichts, was dazwischen liegt. Also es gibt keine Aussage, die sowohl wahr und falsch ist, und es gibt auch keine Aussage, die ein bisschen wahr und ein bisschen falsch ist.

Also Wahrscheinlichkeiten und sowas haben wir bei der Logik nicht. Logik hat nur Aussagen, und sie sagt ja oder nein, wahr oder falsch. Und dass es keine andere Sache gibt, ist das, was man als Satz von ausgeschlossenen Dritten bezeichnet, auch.

Es muss wahr sein oder es muss falsch sein. Und Aussagen sind sowas wie da, es regnet, die Firma hat Gewinne erwirtschaftet, der Schweizer Meister 2028 heißt FC Olten. Das heißt nicht, dass es wirklich, dass wir wissen, ob es wahr ist, aber entweder ist die Aussage wahr oder sie ist falsch.

Es gibt nichts, was dazwischen liegt.

Gut, und jetzt können wir mit den Aussagen arbeiten. Wir können sie verknüpfen. Zum Beispiel können wir eine Aussage mit und verknüpfen. Also wenn wir die Aussage haben, die Bieraktion ist um 10 Prozent gestiegen, dann ist das eine Aussage. Kann wahr sein oder falsch sein. Oder der Dollar ist um 1 Prozent gefallen.

Das ist auch eine Aussage. Und die beiden können wir verknüpfen. Wir können zum Beispiel sagen, die Aktie ist um 10 Prozent gestiegen und der Dollar ist um 1 Prozent gefallen. Und das ist wieder eine Aussage. Und die ist zusammengesetzt aus diesen beiden Aussagen vorher mit einem Verbindungswort, das heißt und.

Ja, und und dafür gibt es auch ein Symbol in der Logik, das heißt so. Ein Dreieck, das nach oben gerichtet ist.

Das heißt und, also wenn A dafür stehen würde, dass die BMW-Aktie ist um 10 Prozent gestiegen, B haben wir gesagt, ist die Aussage, der Dollar ist um 1 Prozent gefallen, dann ist A und B diese Aussage hier, die BMW-Aktie ist um 10 Prozent gestiegen und der Dollar ist um 1 Prozent gefallen.

Und da gibt es nicht nur das und, es gibt noch ein paar andere Operatoren, mit denen wir Sätze verbinden können. Es gibt die Negation. Also wenn A eine Aussage ist, dann bedeutet dieses Ding hier, nicht A ist auch eine Aussage. Das soll das Gegenteil ausdrücken.

Und können wir Aussagen verknüpfen, das ist oder. Das ist, wenn dann, wenn A wahr ist, dann ist auch B wahr. Oder genau dann, wenn A wahr ist, dann ist auch B wahr. Und so können wir die auch noch komplexer verknüpfen und das hier alles wären Aussagen.

A und B steht immer für irgendeine Aussage in diesem Fall.

Und das ist das, was ich jetzt gesagt habe, hier noch aufgeschrieben. X und Y, X ist wahr und Y ist wahr mit oder können wir es verknüpfen. Negation, wenn X wahr ist, dann ist die Negation von X falsch. Und wenn XY wahr ist, dann ist es auch X wahr.

Jetzt eine Formel zum Beispiel. Nicht wahr, sondern wir können es nicht, oder not. Ich habe es hier als Symbole gemacht, weil das ist international.

Und das ist so, wie man typischerweise diese Aussagen auch schreibt.

So, wie würde man diesen Satz ausdrücken? Petra, Max und wohl Äpfel als auch Birnen. Mit logischen Operatoren. Ja, aber ich will, dass es mit Petra ist, dass sie Äpfel und Birnen mag.

Und was sind die beiden Aussagen, die man mit dem und verknüpfen? Ja, ich weiß von Petra.

Also Petra mag Äpfel

und Birnen.

Einverstanden? Nee?

Warum nicht?

Nee, das ist schon, und aber hast du das Punkt? Das hier ist das Zeichen für den Durchschnitt. Ist sehrähnlich, weil Durchschnitt heißt, etwas muss in beiden Mengen drin sein und hier muss es für beide Aussagen wahr sein. Also es hat eine Analogie, aber es ist ein anderes Symbol.

Was steht links und rechts von diesem Zeichen?

Etwas, das wahr oder falsch sein kann. Eine Aussage. Petra mag Äpfel, ist eine Aussage. Die kann wahr oder falsch sein. Ist Birnen eine Aussage? Kann Birnen wahr sein? Genau.

Petra mag Birnen, ist eine Aussage. Birnen selbst ist keine Aussage. Kann ich einen Wahrheitswert annehmen. Aber das ist eine Aussage. Petra mag Birnen, das kann wahr oder falsch sein. Und wir sagen hier, Petra mag Birnen und Petra mag Äpfel und Petra mag Birnen.

Also in der natürlichen Sprache verwendet man sowas wie Petra mag sowohl Äpfel als auch Birnen. Logisch heißt das, wir verknüpfen zwei Aussagen, nämlich dass sie Birnen mag und dass sie Äpfel mag. Nächste, Petra mag weder Äpfel noch Birnen. Wie würden wir das in Logik ausdrücken?

Petra mag und dann dieses umgekehrte L.

Und dann ein bisschen Komma gesetzt und genau das Gleiche hätte ich hingeschrieben. Komma, Petra mag Birnen.

Einverstanden?

Oder man könnte auch auch die Äquivalent nehmen. Nein, nein, Äquivalent kommt irgendwann. Was meint ihr?

Was steht nach dem nicht dann dort? Ja, aber generell, was sollte da stehen?

Eine Aussage. Eine Aussage kann man negieren. Ist Äpfel eine Aussage? Nee. Das kann man auch nicht negieren. Wo müsste das not stehen? Ganz vorn.

Nicht, es gilt nicht, dass Petra mag Äpfel. Das ist eine Aussage. Also Petra mag Äpfel ist eine Aussage und wir sagen, sie gilt nicht, dann schreiben wir das nicht vorne dran. Also wir müssen nach dem nicht immer eine Aussage schreiben. Was ist jetzt mit dem hier? Äquivalent, ja.

So, es gilt nicht, dass Petra mag Birnen. So, was ist jetzt mit diesem Komma?

Komma muss ja auch was. Genau, das wird so eine Und-Zeitschrift. Petra mag keine Äpfel und Petra mag keine Birnen.

Also Petra mag weder Äpfel noch Birnen, das ist alles die gleiche Aussage, aber wenn wir es auf Logik runterziehen, müssen wir die Basisaussagen haben und eine Aussage, Petra mag Äpfel und wir sagen, das gilt nicht. Petra mag keine Birnen, das gilt auch nicht und beides zusammen gilt nicht, deshalb machen wir das mit dem.

Was mit dem nächsten? Klaus fährt nur Fahrrad, wenn die Sonne scheint.

Was werden die Aussagen? Klaus fährt mit dem Fahrrad. Klar, Klaus fährt mit dem Fahrrad und? Und die Sonne scheint. So, die beiden müssen wir jetzt verknüpfen. Implikation.

Ja, wir brauchen Implikation, cool. Was wird impliziert?

Klaus mit dem Fahrrad fährt, dann muss auch die Sonne scheinen. Cool, gut. Klaus fährt mit dem Fahrrad.

Eine Aussage.

Und die Sonne scheint, das ist die andere Aussage. Das ist das gleiche Prinzip wie oben, halt einfach nur, dass man beim Nicht anstatt Nicht Implikation hinschreibt. Davon? Genau, die Implikation und dann das Ist-Gleich-Zeichen. Nee? Nee.

Ah, nein, das ist eine Mischung. Das ist dazwischen. Implikation verbindet zwei Aussagen. Das heißt, wenn Klaus Fahrrad fährt, dann scheint auch die Sonne. Wenn wir sagen, der fährt nur Fahrrad, wenn die Sonne scheint, also kann es nicht sein, dass er Fahrrad fährt und es regnet.

Gut, da meine Zeit abgelaufen ist, aber nur für heute, überlasse ich euch die beiden anderen mal. Könnt ihr euch das überlegen bis zum nächsten Mal, wie ihr das als logische Aussagen darstellen würdet. Es stimmt nicht, dass die Erde eine Scheibe ist.

Und wenn die Sonne scheint, fährt Klaus mit dem Fahrrad. Nachher gucken, denkt mal, gucken wir uns an, wie man damit rechnen kann und neues Wissen herleiten kann. Gut, noch Fragen? Dann wünsche ich euch einen guten Appetit. Ciao.

Ja, ich würde es nicht als Code bezeichnen. Wir können das natürlich auch verwenden in Soldokode. Statt FNLs macht man das. Ähm, dann das schon. Man könnte es dafür verwenden, aber die Idee hier ist noch eine andere, dass wir gar keinen, nicht von Code reden.

Ich denke, das ist so für Prompts gedacht.

Ja, damit die Prompts eindeutig sind, könnten wir uns machen. Aber da sind auch die Large Language Models ganz gut,ähnliche Sätze zu erkennen, weil sie das Embedding machen. Also dann, aber es wäre eindeutiger. Also die verstehen uns ja auch falsch. Also mit sowas wäre es eindeutiger, das stimmt.

Ja, wir wissen den Namen. Check, Endur. Ja, genau, das habe ich gesehen, aber noch nicht verantwortet, glaube ich. Weil das Ding ist, es kommt eine gewisse Bewährung. Endur. OMD.

Sorry, jetzt ist gerade das Bild weggegangen, wieso das jetzt?