

# Wir planen die Produktion - mit Expertenwissen!

## Storyboard

In der Technikum Digital Factory stellen wir zurzeit Komponenten für Forschungs- und Ausbildungszwecke her. Da dabei mehrere Arten von Komponenten in der gleichen Produktionsumgebung gefertigt werden, ändert sich der Ablauf in der Fabrik regelmäßig. Bei jeder Änderung des Produktionsablaufs muss aber erneut geplant werden, wie die einzelnen Werkstücke die Fabrik durchlaufen müssen, damit unsere Produkte gefertigt werden können. Dabei muss die Dauer jedes Bearbeitungsschritts, die Verfügbarkeit mehrerer gleicher Maschinen und die Reihenfolge der Bearbeitungsschritte berücksichtigt werden. Da diese Planungsaufgabe für jede Änderung der Produktion erneut durchgeführt werden muss, wollen wir sie automatisch lösen lassen.

Abbildung 1 zeigt eine der Produktionsstraßen, in der die ankommenden Teile an vier Stationen verarbeitet werden. Die zu verplanenden Stationen sind eine Fräse, eine Bohrstation, eine Station zum Zusammenbau und eine Polierstation. Dabei dauern die Bearbeitungsschritte an den Stationen unterschiedlich lang und die Fräse ist mehrmals vorhanden. Da das vorliegende Problem eine Planungsaufgabe ist, wollen wir Logikprogrammierung anwenden. Im Zuge von Logikprogrammierung formulieren wir ein Problem in Form von Einschränkungen, damit es automatisch gelöst werden kann. Diese Art der Problemformulierung ist speziell geeignet, um solche Planungsaufgaben zu lösen (siehe AIAV Video [Heutige Anwendungen - Fokus Logikprogrammierung](#)).

Dabei verbinden wir Logikprogrammierung mit einer Ontologie, um aus Wissen über die einzelnen Produktionsschritte automatisch den Produktionsablauf zu planen. In der Ontologie (siehe AIAV Video [Ontologien](#)) legen wir fest, wie sich der Zustand des Werkstücks durch die Bearbeitung an einer Station ändert. Basierend auf Informationen über die Dauer jedes Produktionsschrittes können wir dann mittels Logikprogrammierung (siehe AIAV Video [Wissensrepräsentation und Logikprogrammierung](#)) den Zeitlichen Ablauf in der Fabrik planen. Dieses Vorgehen erlaubt es uns, bei einem sich ändernden Produktionsumfeld die Ontologie anzupassen und den Ablauf automatisch neu planen zu lassen.

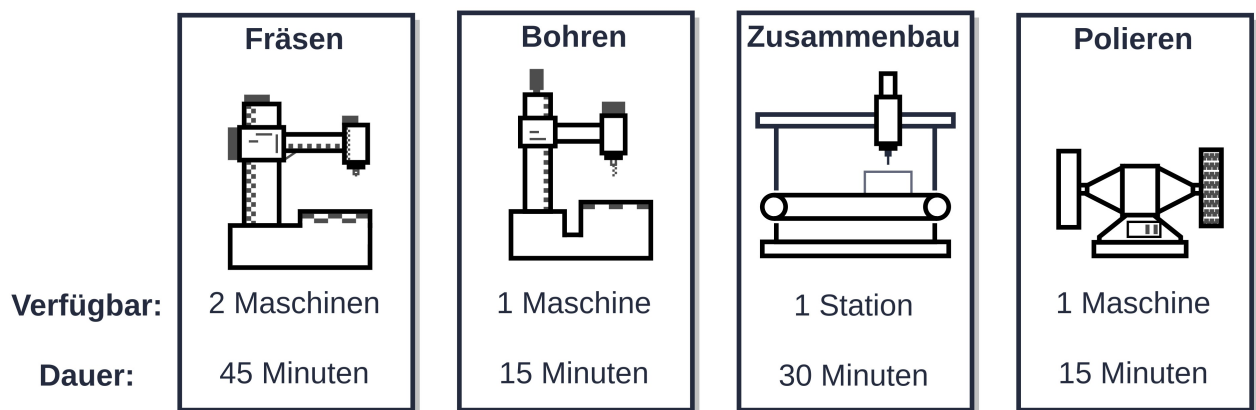


Abbildung 1: Der zu planende Produktionsablauf umfasst eine Fräse, eine Bohrstation, eine Station zum Zusammenbau und eine Poliermaschine. Dabei dauert jeder Bearbeitungsschritt unterschiedlich lang und manche Maschinen, wie z.B. die Fräse, sind mehrmals vorhanden.

## Ontologien

Mittels [Ontologien](#) können wir Expertenwissen in eine maschinenlesbare Form bringen (siehe AIAV Video [Ontologien](#)), indem wir Vorgänge in der Umwelt mittels Konzepten und Instanzen beschreiben. Konzepte stellen dabei allgemeine Zusammenhänge dar, während Instanzen konkrete Beispiele für diese Zusammenhänge sind. Ein Beispiel für so ein Konzept ist "*Künstler malen Kunstwerke*.", welches im allgemeinen Künstler mit Kunstwerken in Beziehung setzt (siehe Abbildung 2). Basierend auf so einem Konzept werden dann Instanzen, also konkrete Beispiele für die Konzepte, festgehalten. "*Da Vinci ist ein Künstler, Mona Lisa ein Kunstwerk*.", sind jeweils Instanzen der Konzepte von *Künstler* und *Kunstwerk*. Vom Wissen aus unserem Konzept können wir nun ableiten, dass Da Vinci die Mona Lisa gemalt hat.

Dieses Vorgehen erlaubt es uns mittels Konzepten und Instanzen eine zusammenhängende Wissensbasis zu erstellen, die einen bestimmten Wirkungsraum beschreibt und währenddessen auch einfach maschinell verarbeitet werden kann. Die Verarbeitung erfolgt dabei durch den sogenannten Reasoner, welcher Anfragen verarbeiten und Zusammenhänge zwischen Instanzen von der Wissensbasis ableiten kann.

Bekannte Konzepte und Instanzen werden in Form eines sogenannten [Knowledge Graphs dargestellt](#), um die Wissensbasis einfach lesbar zu machen. Abbildung 2 stellt unser Beispiel als so einen Knowledge Graph dar.

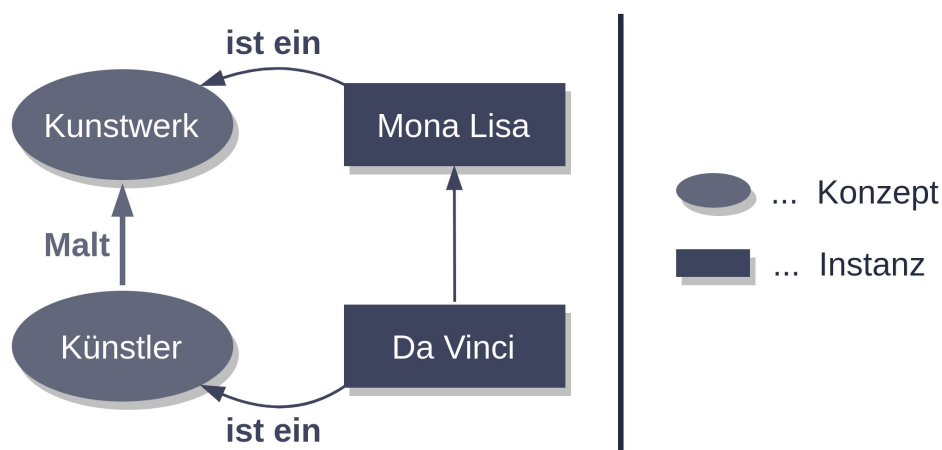


Abbildung 2: Wir stellen unsere Wissensbasis in form eines Knowledge Graphs dar, um sie einfach verständlich zu machen. Das Konzept **Künstler malen Kunstwerke** ist der Wissensbasis dabei bekannt. Da Da Vinci ein Künstler und Mona Lisa ein Kunstwerk ist, können wir aus der Wissensbasis ableiten, dass Da Vinci die Mona Lisa gemalt hat.

## Logikprogrammierung

[Python](#) oder [C](#) sind gängige Beispiele für Programmiersprachen in denen auf künstlicher Intelligenz basierende Lösungen implementiert werden. Diese beiden Sprachen sind Teil der sogenannten [imperativen Programmiersprachen](#), da man bei der Programmierung klare Anweisungen gibt, welche Aktionen das Programm durchführen soll. Dabei entscheidet der Programmierer wie das Programm vorgeht um ein Problem zu lösen.

Analog zu den imperativen Programmiersprachen steht die [Logikprogrammierung](#) (siehe AIAV Video [Wissensrepräsentation und Logikprogrammierung](#)). Dabei formulieren wir das Problem in Form von Vorgaben, bestehend aus logischen Operatoren. Beispiele für solche Operatoren sind UND und ODER, welche zwei oder mehrere Eingänge miteinander verknüpfen. Sie können sich diese Vorgaben wie eine

Art Spezifikation vorstellen; diese gibt an was erfüllt sein muss, sodass eine Lösung für das Problem vorhanden ist. Dabei schränkt jede Vorgabe die Menge an möglichen Lösungen weiter ein, bis eine Lösung vorhanden ist, die unseren gewünschten Zweck erfüllt. Aufgrund dieses Vorgehens nennt man die Vorgaben auch Einschränkungen oder Constraints.

Nach Festlegen der Einschränkungen werden diese in ein Lösungsprogramm, einem sogenannten Boolean Satisfiability (SAT) Solver, gegeben. Dieser vereinfacht zunächst die Gesamtheit der vorgegebenen Einschränkungen. Diese vereinfachte Form kann dann ausgegeben oder zur Suche nach spezifischen Lösungen verwendet werden. Ein gängiger SAT Solver ist Teil der Programmiersprache [Prolog](#). Prolog erlaubt es, Einschränkungen in einer eigenen Sprache zu formulieren, zu vereinfachen und dynamisch Anfragen basierend auf den Einschränkungen zu verarbeiten. Neben dedizierten Logik Programmiersprachen, wie Prolog, gibt es aber auch Solver, welche die Formulierung der Einschränkungen in bestehenden Programmiersprachen erlauben. Ein Beispiel für so eine Bibliothek ist [Z3](#), ein offener SAT Solver von Microsoft. Dieser Unterstützt unter anderem C, C++, Java und Python.

## Praktische Implementierung

Nachdem wir nun Ontologien und Logikprogrammierung verstehen, kommen wir auf unser eigentliches Problem zurück. Für die praktische Implementierung wollen wir, basierend auf Wissen über die verfügbaren Maschinen und den zu produzierenden Komponenten, den zeitlichen Produktionsablauf in der Fabrik planen. Also wollen wir wissen, wann welche Werkstücke in jeder Maschine bearbeitet werden.

Die Ontologie soll die Grundlage für die Planung sein, da diese einfach verändert werden kann, um den Produktionsablauf anzupassen. In dieser halten wir unser Wissen über die Maschinen und zu produzierenden Komponenten fest. Abbildung 3 zeigt dabei die für den Use Case definierte Ontologie. Diese beinhaltet zwei Konzepte die beschreiben, dass Maschinen Werkstücke erzeugen und Werkstücke von Maschinen verarbeitet werden. Dabei definieren wir die einzelnen Maschinen als Instanzen des Konzepts *Maschine* und den Fertigstellungsgrad des Werkstücks als Instanzen des Konzepts *Werkstück*. Der Ablauf, in dem die Werkstücke bearbeitet werden, ergibt sich dann aus den Beziehungen der einzelnen Instanzen von Werkstücken und Maschinen.

Anhand der Ontologie in Abbildung 3 bearbeiten wir die Anfrage, in welcher Beziehung das unbearbeitete und fertige Werkstück zueinander stehen. Die Antwort auf diese Anfrage beinhaltet sowohl Werkstücke als auch Maschinen und repräsentiert den Produktionsablauf.

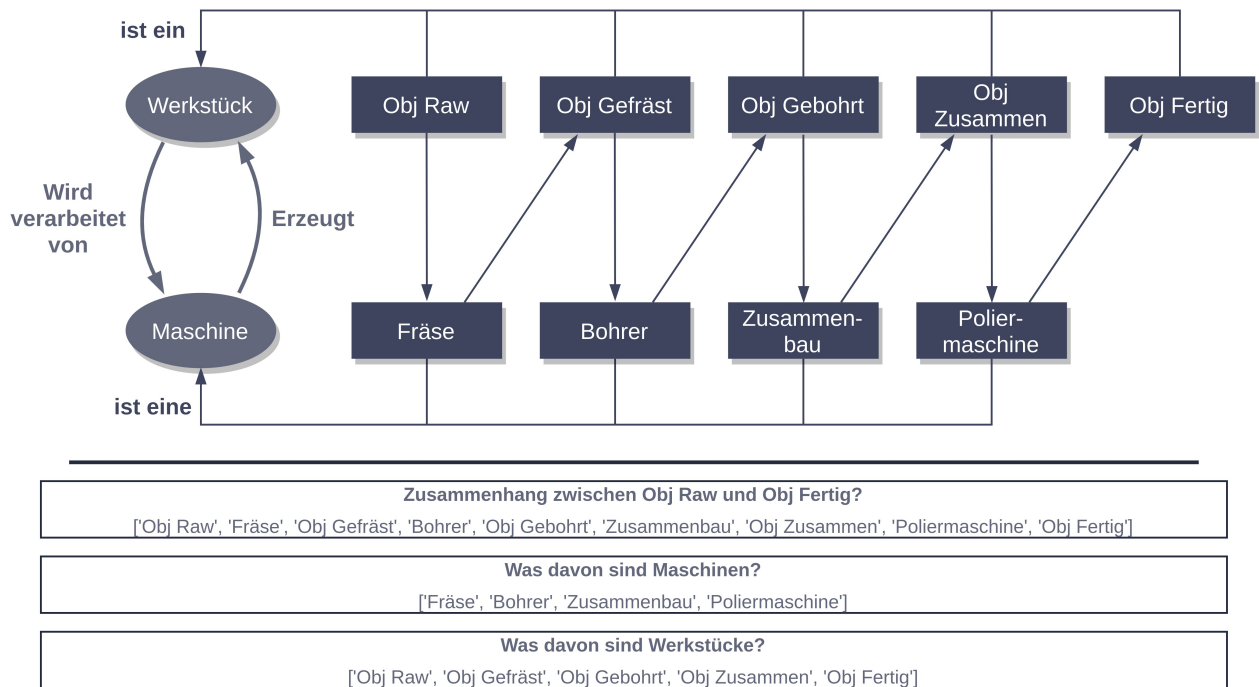


Abbildung 3: Wir erstellen eine Ontologie, anhand derer die Produktion geplant wird. Dabei wissen wir aus zwei Konzepten, dass Maschinen Werkstücke erzeugen und Werkstücke von Maschinen verarbeitet werden. Durch Verknüpfung der verschiedenen Instanzen können wir so den Produktionsablauf beschreiben. Die benötigten Verarbeitungsschritte eines Werkstücks können mittels Anfragen ermittelt werden.

Nach Festlegen der Ontologie wenden wir nun Logikprogrammierung an, um den zeitlichen Verlauf der Produktion zu planen. Dafür benötigen wir die aus der Ontologie ermittelte Produktionsabfolge sowie die Dauer der Bearbeitungsschritte jeder Maschine und die Anzahl der verfügbaren Maschinen (siehe Abbildung 1). Wir teilen zunächst die verfügbare Zeit in Zeitslots mit einer fixen Länge (z.B. 15 Minuten) ein und definieren die benötigte Zeit für jeden Produktionsschritt in Form dieser Slots. Wir betrachten die verfügbare Zeit in Form dieser Slots, um die Anzahl der resultierenden Einschränkungen zu verringern und damit die benötigte Rechenleistung beim Ermitteln des Plans so gering wie möglich zu halten. Anschließend definieren wir die Anzahl der Teile, die wir in den verfügbaren Zeitslots verplanen wollen.

Die Einschränkung, dass allen Zeitslots keines oder ein einziges Teil zugeordnet werden muss, stellt die Grundlage der Einteilung dar und wird gespeichert. Sie legt die Zeitslots für jede Maschine fest. Anschließend werden die Einschränkungen pro zu verarbeitendem Teil aufgestellt. Dabei müssen wir beachten, dass jedes Teil nicht gleichzeitig an zwei Orten, also in mehreren Maschinen, sein kann und dass die Anzahl und Reihenfolge der Verarbeitungsschritte im Vorhinein nicht bekannt sind. Deswegen gehen wir wie folgt vor:

- Iteration über alle möglichen Zeitpunkte an denen die Produktion eines Teils starten kann:
  - Iteration über alle Verarbeitungsschritte:
  - Iteration über alle Maschinen im aktuellen Verarbeitungsschritt:
    - Aufstellen der Einschränkung, dass das aktuelle Teil zur aktuellen Zeit in der aktuellen Maschine ist
    - Aufstellen der Einschränkungen, dass das aktuelle Teil zur aktuellen Zeit in keiner anderen Maschine ist

- Aufstellen der Einschränkungen, dass das aktuelle Teil nie in einer Maschine gleichen Typs ist (da jeder Verarbeitungsschritt nur einmal ausgeführt wird)

Einschränkungen werden anhand dieses Ablaufs so lange erstellt und gespeichert, bis alle zu verplanenden Teile zu allen möglichen Startzeitpunkten berücksichtigt wurden. Anschließend erstellen wir ein [Z3](#) Modell mit allen gespeicherten Einschränkungen, welches durch den Solver gelöst wird. Die Lösung kommt dabei in Form einer Tabelle, die beschreibt welches Werkstück in welchen Zeitslots in welcher Maschine ist (siehe Abbildung 4). Dabei stellt jede Zahl (1 - 6) den Index eines Werkstücks dar, während ein Leeres Feld heiß, dass die jeweilige Maschine zu diesem Zeitpunkt leer ist.

In Abbildung 4 sehen wir, dass beim oben vorgestellten Problem ein Engpass in der Produktion beim Zusammenbau der Werkstücke entsteht. Hier kann der Durchsatz in der Produktion durch eine zweite Station zum Zusammenbau erhöht werden.

		Fräsen 1	Fräsen 2	Bohren	Zusammenbau	Polieren
<b>T0:</b>	08:00		6			
<b>T1:</b>	08:15		6			
<b>T2:</b>	08:30	4	6			
<b>T3:</b>	08:45	4		6		
<b>T4:</b>	09:00	4	5		6	
<b>T5:</b>	09:15		5	4	6	
<b>T6:</b>	09:30	2	5		4	6
<b>T7:</b>	09:45	2		5	4	
<b>T8:</b>	10:00	2	1		5	4
<b>T9:</b>	10:15		1	2	5	
<b>T10:</b>	10:30	3	1		2	5
<b>T11:</b>	10:45	3		1	2	
<b>T12:</b>	11:00	3			1	2
<b>T13:</b>	11:15			3	1	
<b>T14:</b>	11:30				3	1
<b>T15:</b>	11:45				3	
<b>T16:</b>	12:00					3
<b>T17:</b>	12:15					
<b>T18:</b>	12:30					
<b>T19:</b>	12:45					

Abbildung 4: Die anhand der Ontologie erstellten Einschränkungen werden in einen SAT Solver gegeben, welcher eine Lösung ermittelt, die alle Einschränkungen erfüllt. Für unser Problem kommt diese Lösung in Form einer Tabelle, welche die Werkstücke bestimmten Zeitslots in den Maschinen zuordnet. Hier wurden sechs Werkstücke über 16 Zeitslots für den in Abbildung 1 beschriebenen Produktionsablauf verplant.

## Fazit

Ontologien erlauben es uns, Wissen über ein bestimmtes Umfeld effizient in eine für Maschinen verständliche Form zu bringen und dynamisch Anfragen zu verarbeiten. Umfangreiche Software Pakete zur Formulierung und Abfrage von Ontologien sind gratis verfügbar und ermöglichen die Verarbeitung von Ontologien mit nur wenig Rechenleistung.

Mittels Logikprogrammierung können wir Planungsprobleme in Form von Einschränkungen formulieren und automatisch lösen lassen. In diesem Use Case wurden Ontologien mit Logikprogrammierung gekoppelt, um ein Planungsproblem in der Produktion autonom, auf Basis von Expertenwissen über den Produktionsablauf, zu planen. Die Kombination von Logikprogrammierung und Ontologien erlaubt es uns, den Produktionsablauf direkt aus den Informationen über die Produktionsschritte zu planen.

Ändern sich also Aspekte der Produktion, können diese in die Ontologie eingespielt werden. Anschließend wird der Produktionsplan autonom überarbeitet.