Projektlabor Robotik MINTgrün

Abläufe strukturieren mit Zustandsautomaten

Disclaimer

Zustandsautomaten sind eine wichtige Grundlage der theoretischen Informatik. In diesem Artikel geht es in erster Linie darum, wie das Konzept praktisch dazu verwendet werden kann, um Handlungsabläufe von Robotern zu programmieren. Deshalb verwenden wir in den ersten Abschnitten Begriffe, die uns intuitiver erscheinen als die in einem anderen Kontext entstandenen Fachbegriffe. Um den Einstieg in die Theorie zu vereinfachen, haben wir die Fachbegriffe in Klammern hinzugefügt.

Motivation

Hier geht es um eine Problemstellung, das so alt wie die Informatik (und älter als die ersten Computer) ist:

Wie bringe ich einer Maschine bei, in unterschiedliche Situationen auf unterschiedliche Reize angemessen zu reagieren?

Es lohnt sich, diese Frage einmal genauer unter die Lupe zu nehmen. Offenbar gibt es:

- **unterschiedliche Reaktionen (Ausgaben)** - z.B. kann ein Roboter vorwärts, rückwärts oder auf der Stelle im Kreis fahren
- **unterschiedliche Reize (Eingaben)** - z.B. kann vor dem Roboter eine Wand, ein Mensch oder ein leerer Raum erkannt werden.
- **unterschiedliche Situationen (Zustände)** - z.B. Auf der Suche nach der Steckdose, auf der Flucht vor einem Gegner oder im Ruhezustand.

Es reicht also nicht aus, auf einen bestimmten Reiz/Eingabe stets mit der gleichen Aktion/Ausgabe zu reagieren. Um eine Steckdose zu finden, ist es eine vielleicht gute Idee direkt auf die nächste Wand zuzufahren - beim Ausweichen vor einem Verfolger eher nicht…

Man kann sich vorstellen, dass die Sache schnell unübersichtlich wird. Und wenn wichtige Kombinationen von Eingaben und Situationen unberücksichtigt bleiben, gibt es Probleme.

Struktur

Die ganze Sache wird klarer, wenn man sie strukturiert aufzeichnet. Dabei kommt uns eine interessante Erkenntnis der theoretischen Informatik zu pass:

Es reicht aus, beim Eintreten einer Situation (Zustand) genau *eine* festgelegte Handlung (Ausgabe) auszuführen und erst danach die Eingabedaten anzuschauen und anhand ihrer zu entscheiden, welche Situation als nächstes vorliegt.

Handlungsmuster, die eine komplexere Abfolge von Eingaben überprüfen, Handeln und Zustände wechseln benötigen (Mealy Automat), lassen sich nach einem einfachen Schema in solche einfacher strukturierten Abfolgen (Moore Automat) zerlegen.

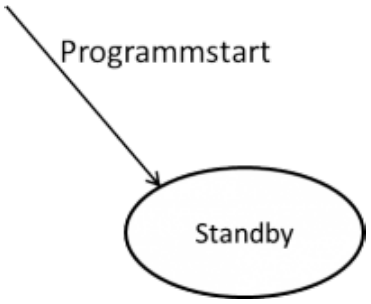
Wir brauchen also nur die Zustände und Bedingungen für einen Zustandswechsel aufzuschreiben, um die Struktur einer so aufgebauten Strategie vollständig zu beschreiben. Zusätzlich brauchen wir eine Liste, die für jeden Situation (Zustand) eine Aktion (Ausgabe) enthält, die ausgeführt wird, wenn die Situation eintritt (der Zustand betreten wird).

Beispiel: Entwicklung einer Staubsaugerroboter-Steuerung

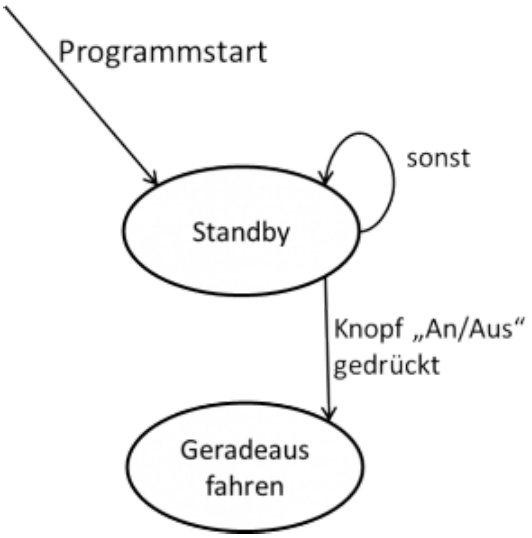
Genug Theorie!

Alles wird viel anschaulicher, wenn wir nach und nach alles, was der Roboter tun soll, nach der oben beschriebenen Struktur in einem Grafen aufzeichnen:

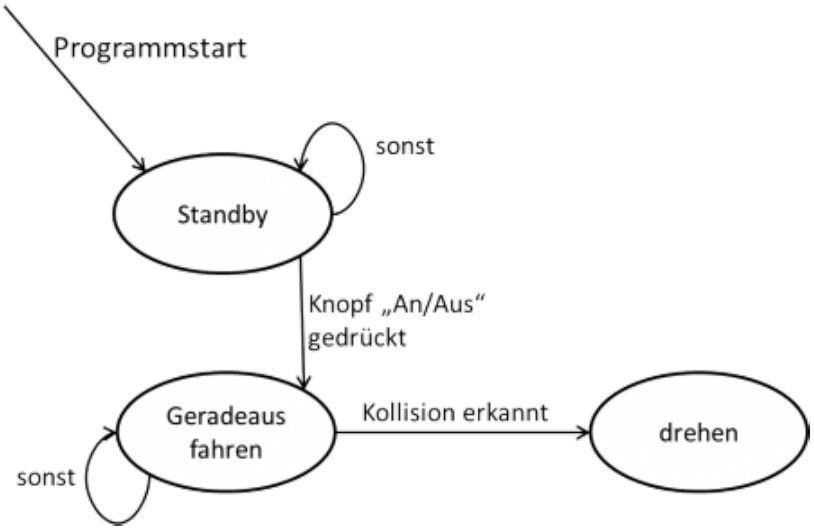
Nachdem er mit Strom versorgt wird, erst einmal nichts tun und auf Befehle warten:



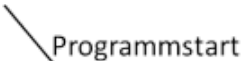
Auf Knopfdruck soll er mit dem Saubermachen anfangen. Dazu soll er erst einmal einfach geradeausfahen und saugen.

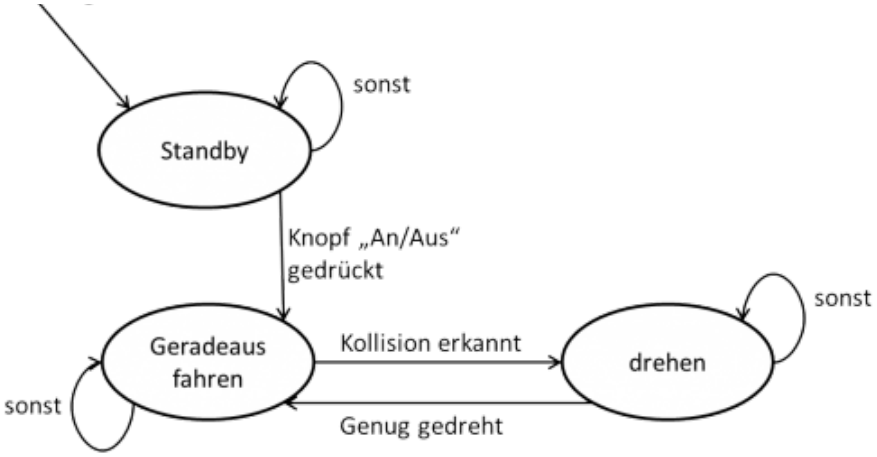


Mit diesem Verhalten würde sich der Roboter an der ersten Wand den Kopf einrennen. Er braucht also eine Funktion, um Kollisionen zu erkennen und eine Wende einzuleiten:

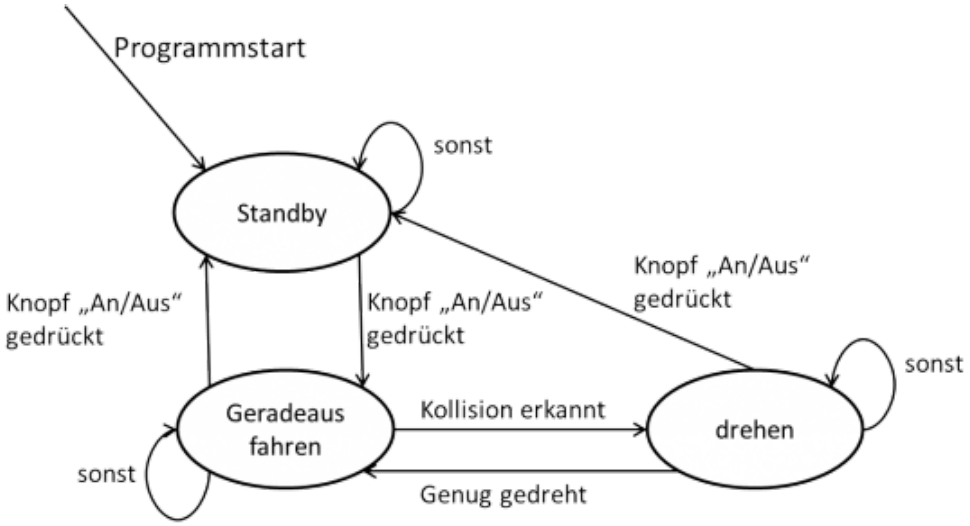


Einfach nur mit dem Wende anfangen reicht nicht. Wir müssen auch irgendein Kriterium defineren, nach dem entschieden wird, dass der Roboter sich weit genug gedreht hat. Das könnte z.B. eine verstrichene Zeit, eine Anzahl von Radumdrehungen oder die Ausgabe eines Sensors sein.





Was fehlt noch? Es wäre auch schön, den Roboter wieder abschalten zu können...



Umsetzung in einem Programm

Ein Grund, der die Modellierung eines Verhaltens als Zustandsautomat so attraktiv macht, ist dass die Umsetzung in ein funktionierendes Programm sehr einfach ist.

1. Schritt: Eine Variable, die den aktuellen Zustand speichert

Als erstes brauchen wir etwas, in dem wir den aktuellen Zustand speichern. Diese Variable bildet sozusagen das Gedächtnis des Zustandsautomaten. Besonders eignet sich hierfür ein sogenannter enum [http://de.wikibooks.org/wiki/C%2B%2B-Programmierung/_Weitere_Grundelemente/_Aufz%C3%A4hlungen] wert, der eine komfortable Möglichkeit bietet, verschiedenen Werten Namen zu geben:

```
//wir definieren einen neuen Variablentypen "ZustandsTyp", der die Werte standby, geradeau und drehen annehmen kann.
enum ZustandsTyp{
 standby,
 geradeaus,
 drehen
}

// Die eigentliche Variable definieren wir hier - und geben gleich einen Anfangswert an:
ZustandsTyp aktuellerZustand= standby;
```

Hinweis: In **Processing** muss die Definition des enums in einer eigenen Datei („Tab“) mit der Endung .java stehen. (vgl. auch http://stackoverflow.com/questions/13370090/enums-in-processing-2-0 [http://stackoverflow.com/questions/13370090/enums-in-processing-2-0])