Automaten mit dot erstellen

1 Was ist dot?

dot ist ein Programm zum Kompilieren von dot-Dateien in verschiedene Grafikformate, sowie der Name einer Sprache, mit der man Graphen spezifizieren kann. Unter Anderem können damit sehr einfach ansprechende Grafiken von Automaten erstellt werden.
Das Programm dot ist Teil des Pakets graphviz und ist auf den Coli-Servern bereits vorinstalliert. Unter den meisten Linux-Distributionen sollte das Paket auch in den offiziellen Paketquellen zu finden sein. Wenn Du also z.B. Ubuntu hast, kannst Du es mit `sudo apt-get install graphviz` installieren.
dot ist nicht nur praktisch um selbst Automaten zu spezifizieren, sondern auch, weil man seine Programme einfach eine dot-Datei erstellen lassen kann (z.B. von einem hierarchischen Clustering) und ganz schnell eine anschauliche Grafik bekommt, weil dot die Anordnung von (fast) allem selbst feststellt.

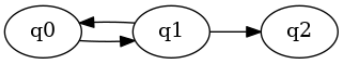
2 Der dot-Workflow

Ein Graph wird zunächst in einer dot-Datei spezifiziert. Dabei werden die einzelnen Knoten und deren Aussehen, sowie die Verbindungen zwischen ihnen (Kanten) definiert. Diese dot-Datei wird dann mit dem dot-Programm in eine Grafik (z.B. PNG) konvertiert, die dann zum Beispiel in LaTeX eingebunden werden kann.

3 Ein ganz einfacher Graph

Im Folgenden ist der dot-Code für einen einfachen Graphen zu sehen:

```
1 digraph {
2   q0
3   q1
4   q2
5
6   q0 -> q1
7   q1 -> q2
8   q1 -> q0
9 }
```



In Zeile 1 wird ein digraph, also ein gerichteter Graph definiert. Alles zwischen den folgenden geschweiften Klammern gehört zu diesem Graphen.
In Zeilen 2 bis 4 werden die benutzten Zustände definiert. Jeder Zustand muss einen

einzigartigen Namen haben. Die Standard-Zustände werden als Ellipsen dargestellt und mit ihren Namen bezeichnet.
In Zeilen 6 bis 8 werden die Kanten definiert. `q0 -> q1` heißt zum Beispiel, dass es eine Kante von `q0` zu `q1` gibt. Die Standard-Kanten werden als Pfeile dargestellt und sind nicht beschriftet.

4 Von der `dot`-Datei zum Bild

Um aus der soeben erstellten `dot`-Datei eine PNG-Grafik zu erzeugen, muss nun das Programm `dot` über die Kommandozeile aufgerufen werden. Im Folgenden Beispiel gehe ich davon aus, dass unser `dot`-Code in einer Datei `graph.dot` gespeichert ist und das Ergebnis in `graph.png` gespeichert werden soll. Der auszuführende Befehl lautet dann:

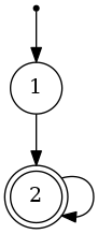
```
1 dot -Tpng graph.dot > graph.png
```

Mit der Option `-Tpng` wird angegeben, dass das Ausgabeformat ein PNG-Bild ist. Mit `>` wird die Ausgabe in die dahinter stehende Datei umgeleitet, d.h. dort gespeichert.

5 Benutzerdefinierte Knoten

Knoten können natürlich nicht nur als Ellipsen dargestellt werden, sondern es gibt viele andere mögliche Formen. Für Automaten interessant sind noch die Formen Punkt, Kreis, und Doppelkreis.
Bei der Definition der Knoten kann die Form durch `[shape=point]` für einen Punkt und `[shape=doublecircle]` für einen Doppelkreis hinter dem entsprechenden Knoten angegeben werden.
Das Label eines Knoten (also der Text, der im Knoten angezeigt wird) kann durch `[label="Neues Label"]` angegeben werden.
Diese Optionen lassen sich auch zum Beispiel mit `[shape=doublecircle, label="B"]` kombinieren.
Zum Abschluss noch ein Beispielgraph:

```
1 digraph {
2   q0 [shape=point]
3   q1 [shape=circle, label="1"]
4   q2 [shape=doublecircle, label="2"]
5
6   q0 -> q1
7   q1 -> q2
8   q2 -> q2
9 }
```

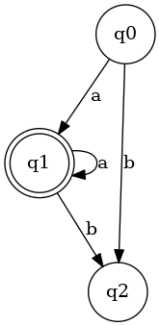


6 Benutzerdefinierte Kanten

Kanten lassen sich ebenfalls anpassen. Für das Automatenzeichnen ist hier vorallem das Hinzufügen eines Labels spannend. Da dies analog zu den Labels von Knoten ist, gebe ich hier nur einen Beispielgraphen an:

```
digraph {
  q0 [shape=circle]
  q1 [shape=doublecircle]
  q2 [shape=circle]

  q0 -> q1 [label="a"]
  q0 -> q2 [label="b"]
  q1 -> q1 [label="a"]
  q1 -> q2 [label="b"]
}
```



7 Ein Beispiel-Automat

Der letzte Graph sieht ja schon fast wie ein Automat aus. Allerdings fehlen noch ein paar kleine Details:

- Automaten werden für gewöhnlich von links nach rechts gezeichnet
- Der Startzustand ist noch nicht markiert

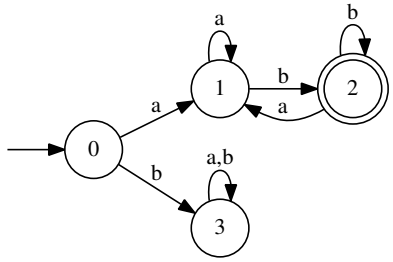
Um den Graphen nicht von oben nach unten, sondern von links nach rechts ausrichten zu lassen, kann man am Anfang der Definition die Zeile `rankdir=LR` hinzufügen.

Um den Startzustand zu markieren, können wir einen Pseudozustand definieren und ihn als `point` zeichnen lassen und unsichtbar machen, indem wir `style` auf `"invis"` und `width` wir auf 0 setzen, damit er auch keinen Platz verbraucht. Von diesem aus können wir eine Kante zum Startzustand ziehen.
Manchmal möchte man ein Bild von dot irgendwo einbinden, das keinen schneeweißen Hintergrund hat. In dem Fall sieht es oft am besten aus, wenn man den Hintergrund transparent macht, indem man `bgcolor` auf `"transparent"` setzt. Außerdem muss man sicherstellen, dass das Grafikformat Transparenz darstellen kann, insofern sind insbesondere PNG, PDF und SVG geeignet.
Im Folgenden der Code für einen kompletten Automaten:

```
digraph {
  rankdir=LR
  bgcolor="transparent"

  p  [shape=point, width=0, style="invis"]
  q0 [shape=circle, label="0"]
  q1 [shape=circle, label="1"]
  q2 [shape=doublecircle, label="2"]
  q3 [shape=circle, label="3"]

  p  -> q0
  q0 -> q1 [label="a"]
  q0 -> q3 [label="b"]
  q1 -> q1 [label="a"]
  q1 -> q2 [label="b"]
  q2 -> q2 [label="b"]
  q2 -> q1 [label="a"]
  q3 -> q3 [label="a,b"]
}
```



8 Welches Ausgabeformat sollte ich nehmen?

`dot` unterstützt einige Ausgabeformate wie PNG, GIF, SVG, EPS und PDF. Man unterscheidet zwei verschiedene Arten von Grafikformaten: Pixelgrafiken und Vektorgrafiken. Bei Pixelgrafiken (PNG und GIF) wird jeder einzelne Punkt abgespeichert, bei Vektorgrafien (SVG ,EPS und PDF) besteht das Bild aus geometrischen Objekten.

Der Vorteil von Vektorgrafiken ist, dass man sie beliebig vergrößern kann ohne dass das Bild verpixelt wird; das macht sich zum Beispiel besser auf einer großen Leinwand. Pixelgrafiken haben dafür den Vorteil, dass das Bild nicht immer beim Öffnen berechnet werden muss, was bei sehr großen Dateien ein Vorteil sein kann.[1]
Zum Einbinden in LaTeX ist aber PDF wohl am besten geeignet: es unterstützt Transparenz, ist eine Vektorgrafik und kann mit problemlos z.B. mit `\includegraphics` eingebunden werden.

9 Weiterführende Links

Alles Weitere, was man über `dot` wissen will (sogar eine formale Grammatik der Sprache) findet man unter `http://graphviz.org/Documentation/dotguide.pdf`.

[1] Du kannst ja mal schauen, ob Du herausfindest, welche Bilder in diesem Dokument als Pixelgrafiken und welche als Vektorgrafiken eingebunden wurden ;)