Bei unserer Wiederholung gegen Ende der Vorlesung stellten wir zunächst die Fragen nach dem **Pumping-Lemma** zurück – und vergaßen Sie am Ende wieder aufzugreifen – sorry! Hier nochmal in aller Ruhe:

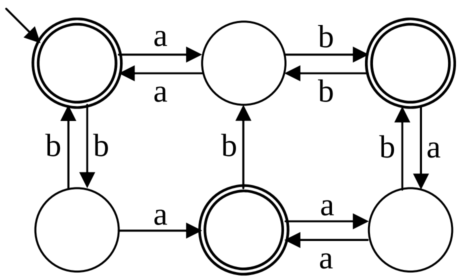
- **Woher kommt das *n* beim Pumping-Lemma?**
- **Und wo geht es hin ☺ – d.h. wie wird es verwendet?**

1. Woher kommt das *n* beim Pumping-Lemma?

Einen endlichen Automaten können wir uns vorstellen als eine Stadt mit ausschließlich Einbahnstraßen (gerichteten Kanten), mit x Kreuzungen (Zuständen) $z_1, ..., z_x$ und mit einem eingeschränkten Vorrat von y Straßennamen (Symbolen)) $a_1, ..., a_y$ (Alphabet). Von jeder Kreuzung aus gehen y Einbahnstraßen (Kanten) ab, mit jedem Namen a_i eine (von z_j nach $\delta(z_j, a_i)$). Zu jedem Namen a_i existieren in der Stadt also x Straßen dieses Namens, denn an jeder Kreuzung beginnt eine. In die Kreuzungen können unterschiedlich viele Einbahnstraßen einmünden, 0, eine, oder mehrere, auch mehr als y.

An manchen Kreuzungen steht ferner ein Schild, auf dem „akzeptiert“ steht (akzeptierende Zustände). Auf einer ganz speziellen Kreuzung (Anfangszustand) werden wir abgesetzt mit einem Zettel, auf dem eine Liste mit Straßennamen steht (Wort). Sagen wir, auf jeder Listenzeile stehe ein Name (Symbol). Wir sollen mit dieser Liste als Wegbeschreibung durch die Stadt laufen. Am Ende sagt uns das Schild (oder dessen Fehlen) an der erreichten Kreuzung, ob die Stadt die Wegbeschreibung (der Automat das Wort) akzeptiert oder nicht.

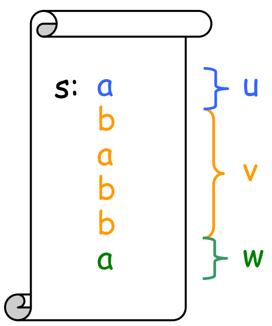
unsere kleine Stadt
x=6, y=2

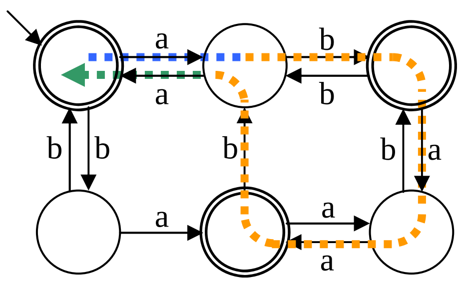


Die Kreise mit Doppelrand entsprechen akzeptierenden „Kreuzungen“

„Kreuzungsnamen/nummern“ sind nicht angezeigt

Wenn wir eine letztlich akzeptierte Wegbeschreibung s mit mindestens so vielen Listenzeilen haben, wie die Stadt Kreuzungen hat (mindestens x), wissen wir, dass wir mindestens eine Kreuzung zweimal passieren müssen: Wir sind nämlich anfangs auf einer, zwischen je zwei Listenzeilen auf einer und am Ende auf einer, passieren also mindestens x+1 Kreuzungen – mehr als es unterschiedliche in der Stadt gibt.





Also finden wie eine Zahl *n* (hier das *x* unserer Stadt) mit der Eigenschaft, dass jede akzeptierte Wegbeschreibung *s* der Länge *n* oder länger an mindestens einer Kreuzung mindestens zweimal vorbeiführt. Die (eventuell leere) Wegbeschreibung bis zur ersten wiederholten Kreuzung nennen wir ***u***, von da bis zum zweiten Besuch dieser Kreuzung ***v*** und den (eventuell leeren) Rest ***w***, so dass *s* = ***u v w***. Wenn wir nach Ablaufen von ***u*** den Rundweg gemäß ***v*** von einer solchen Kreuzung zu sich selbst rasch nochmal wiederholen, bevor wir gemäß der restlichen Wegbeschreibung ***w*** weitergehen, gehen wir natürlich anschließend wieder über dieselben letzten Straßen wie ursprünglich, und die zuletzt erreichte Kreuzung ist die gleiche, akzeptiert also wieder. Genauso geht es uns mit zusätzlichen Wiederholungen. Da wir vor und nach *v* an derselben Kreuzung stehen, akzeptiert die Stadt also nicht nur *uvw*, sondern auch *uvvw*, *uvvvw*, usw.

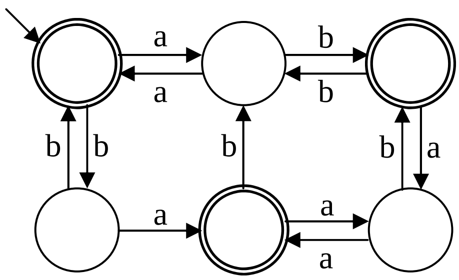
Wir beachten, dass wir nach höchstens *n* Straßenzügen eine Kreuzung zum zweiten Mal erreicht haben müssen, und zwar nach *uv*, d.h. für die Länge der Teilliste: $|uv| \leq n$.

Eventuell ist das n kleiner als man denkt!

Übrigens gibt es im Lande unendlich viele solche Städte, die genau dieselben Wegbeschreibungen akzeptieren wie unsere, darunter eine mit der geringstmöglichen Anzahl von Kreuzungen, namens Minimalstadt (Minimalautomat). Es kann also gut sein, dass diese Wiederholungsmöglichkeit von *v* in *s* = *uvw* für alle akzeptierten Wegbeschreibungen bereits mit einem geringeren *n* auftritt als mit der Kreuzungsanzahl *x* <u>unserer</u> Stadt. Im folgenden Beispiel können wir (beispielsweise durch die Verwendung der Konstruktion des Minimalautomaten) beweisen, dass Minimalstadt dieselben Wegbeschreibungen akzeptiert wie unsere kleine Stadt.

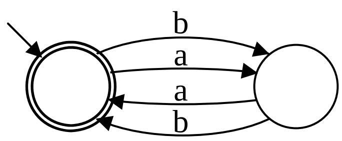
unsere kleine Stadt

x=6, y=2



Minimalstadt

x=2, y=2, **n=2**



Sobald wir nachgeprüft haben, dass Minimalstadt dieselben Wegbeschreibungen akzeptiert wie unsere kleine Stadt, können wir die Anzahl seiner Kreuzungen (Zustände), hier also 2, als *n* benutzen. In der Tat tritt im Beispiel die Wiederholungsmöglichkeit in der Wegbeschreibung (wenn auch noch nicht der erneute Besuch einer Kreuzung in unserer Stadt!) bereits nach 2 Symbolen auf: Mit $u = \varepsilon$, $v = ab$ und $w = abba$ wird nicht nur $(ab)(abba)$ akzeptiert, sondern auch $(ab)(ab)(abba)$, $(ab)(ab)(ab)(abba)$ usw.

Die Wiederholungsmöglichkeit kann je nach Sprache und Wort durchaus früher auftreten als die Größe des Minimalautomaten der Sprache hergibt: Der hat z.B. für die Sprache $L = \{a^m b^n \mid m, n \geq 3\}$ über $\{a,b\}$ vier Zustände, jedoch kann bereits das erste *a* in *ab* beliebig oft wiederholt werden, ohne dass ein Wort außerhalb der Sprache entsteht.

2. Wie verwenden wir das n im Pumping-Lemma?

Wir verwenden das Pumping-Lemma und einen indirekten Beweis, um für gegebene Sprachen *L* über gegebenen Alphabeten (gegebenenfalls) zu zeigen, dass sie nicht regulär sind, d.h. dass es keinen Automaten geben kann, der genau jedes Wort in *L* akzeptiert (und alle anderen Wörter nicht) – weil nämlich für *L* das Pumping-Lemma nicht gilt.
Wir verwenden insbesondere das *n* im Pumping-Lemma, um mit möglichst wenig Mühe ein Gegenbeispiel s = *uvw* aus *L* mit $|uv| \leq n$ und $|v| > 0$ zu finden, für das nicht alle Erweiterungen $uv^n w$ Wörter aus *L* sind, so dass *L* letztlich keine von einem Automaten akzeptierte Sprache sein kann.

Beispiel

$L = \{a^m b^m \mid m > 0\}$.

Annahme: Ein Automat akzeptiert *L*. Sei *n* für *L* wie im Pumping-Lemma.

a) Ein **möglicher – aber nicht optimaler** – Einsatz von *n*:

Sobald $2m \geq n$ ist, gibt es für $s = a^m b^m$ ein „folgenlos wiederholbares“ *v* in

$s = a^m b^m = uvw$ mit $|uv| \leq n$, $|v| > 0$, das qualitativ folgende Lagen einnehmen kann:

```
     s =  a a ... a a  b b ... b b
            |     |    |     |
i.   v =    a ... a    |     |        nur a's
ii.  v =          a ...b     |        a's und dann b's
iii. v =               b ... b        nur b's
```

Bilden wir jetzt *uvvw*, ist dies keinesfalls mehr von der Form $a^k b^k$, denn im Falle …

(i) hat *uvvw* mehr *a*'s als *b*'s,
(ii) kommen in *uvvw* nach *b*'s wieder *a*'s vor,
(iii) hat *uvvw* mehr *b*'s als *a*'s.

b) Ein **geschickterer** Einsatz von *n* macht es uns aber leichter:

Mit *n* aus dem Pumping-Lemma und dem Wort $a^n b^n$ brauchen wir keine Fallunterscheidung mehr, denn dann haben wir zwangsläufig den Fall (i) …

```
s =  a a ... a a b b ... b b
v =    a ... a                  nur a's!
```

Übrigens gibt es …

- neben dem besprochenen „*uvw*-Pumping-Lemma“ für reguläre Sprachen
- noch ein „*uvwxy*-Pumping-Lemma“ für kontextfreie Sprachen, bei dem die hinreichend langen Wörter an zwei Stellen aufgepumpt werden können, ohne die Sprache zu verlassen.

Wie nicht anders zu erwarten, verwendet man es u.a. dazu, für spezielle Sprachen nachzuweisen, dass sie nicht kontextfrei sind.

Vgl. z.B.: http://de.wikipedia.org/wiki/Pumping-Lemma

Bernd Baumgarten, 2009