Lecture L&M 7

Pumping Lemma for Context-Free Languages, Turing Machines

7.1 Pumping lemma for context-free languages

For each of the following languages L, use the pumping lemma to prove that it is not context-free:

A. $L = \{a^k \mid \exists\, n \in \mathbb{N}\colon k = n^2\}$

Let k be arbitrary. Choose $z = a^{k^2}$, and let $z = uvwxy$ with $|vwx| \leq k$ and $|vx| > 0$. It is obvious that v, w and x consist of only as. So $uv^2wx^2y = a^n$ with $n = k^2 + |vx|$. It follows from this that $k^2 < n \leq k^2 + k$. Since $k > 0$, $(k+1)^2 = k^2 + 2k + 1 > k^2 + k \geq n$, so n cannot be a square (the root of n is strictly between k and $k+1$) so $a^n \notin L$.

B. $L = \{a^i b^j c^i d^j \mid i, j \geq 0\}$

Let k be arbitrary. Choose $z = a^k b^k c^k d^k$, and let $z = uvwxy$ with $|vwx| \leq k$ and $|vx| > 0$. It follows that $vwx = e^i$ for $e \in \{a, b, c, d\}$ and $1 \leq i \leq k$, or $vwx = e^i f^j$ for $ef \in \{ab, bc, cd\}$ and $1 \leq i + j \leq k$. In both cases $uv^0wx^0y = uwy$ contains not as many as as cs or not as many bs as ds (or both).

More precisely: v or x is non-empty. Call a and c complementary, and b and d as well. Let e be a symbol that occurs in vx. The complementary symbol $\bar{e}$ does *not* occur in vx. Since the number of e in $z = uvwxy$ is equal to the number of $\bar{e}$, the number of e in uwy is not equal to the number of $\bar{e}$.

C. *(extra)* $L = \{a^i b^{2i} a^i \mid i \geq 0\}$

Let k be arbitrary. Choose $z = a^k b^{2k} a^k$, and let $z = uvwxy$ with $|vwx| \leq k$ and $|vx| > 0$. It follows that $vwx = a^i b^j$ or $vwx = b^j a^i$ with $0 < i + j \leq k$. We only consider the first case; the second is symmetrical.

There are four options:

- $v = a^m$, $w = a^n$, $x = a^{i-m-n}$ with $n > 0$ (since $|vx| > 0$).
  Then $uv^0wx^0y = a^{k-i-n}b^{2k}a^k \notin L$.
- $v = a^m$, $w = a^n$ and $x = a^{i-m-n}b^j$ with $j > 0$.
  Then $uv^0wx^0y = a^{k-i+n}b^{2k-j}a^k \notin L$.
- $v = a^m$, $w = a^{i-m}b^{j-n}$ and $x = b^n$ with $m + n > 0$.
  Then $uv^0wx^0y = a^{k-m}b^{2k-n}a^k \notin L$.
- $v = a^i b^m$, $w = b^{j-m-n}$ and $x = b^n$ with $m > 0$.
  Then $uv^0wx^0y = a^{k-i}b^{2k-m-n}a^k \notin L$.

D. *(extra extra)* L is the set of finite-length prefixes of the infinite word

$$abaabaaabaaaab \ldots ba^n ba^{n+1} b \ldots.$$

Let k be arbitrary. Consider the string $z = abaab \dots ba^k b$, which is in the language and has length greater than k, and assume $z = uvwxy$ with $|vwx| \leq k$ and $|vx| \geq 1$.

First assume $|v| \geq 1$.

Case 1: v has no bs. In this case, v consists solely of as and lies between two consecutive b, or v is the initial a of z. In the first subcase, v occurs in z in a position of the form

$$\dots ba^n ba^i v a^j ba^{n+2} b \dots$$

where $i + |v| + j = n + 1$. Pumping v produces an incorrect number of a following $ba^n b$ and, consequently, the resulting string is not in the language. In the second subcase, pumping v produces either 0 or more than 1 initial a, again meaning that the resulting string is not in the language.

Case 2: v has two or more b. In this case, v contains a substring $ba^n b$. Pumping v produces a string with two substrings of the form $ba^n b$. No string with this property is in the language.

Case 3: v has one b. Then v can be written $a^i b a^j$ and occurs in z as one of the following three cases:

- $va^{2-j}b \dots$ with $i = 1$ and $j \leq 2$; i.e., z starts with v. Pumping v produces a string starting with $aba^{i+j}baab$, which cannot occur in the language.
- $\dots ba^{n-1}ba^{n-i}va^{n+1-j}b \dots$ Pumping v produces the substring

  $$\dots ba^{n-1}ba^{n-i}a^i ba^j a^i ba^j a^{n+1-j} b \dots = ba^{n-1}ba^n ba^{j+i}ba^{n+1}b \dots$$

  which cannot occur in a string in the language.
- $ba^{k-1}ba^{k-i}v$ with $j = 0$; i.e., z ends with v. Pumping v produces a string ending with $ba^k ba^i b$, which cannot occur in the language.

Precisely the same argument holds if x is the nonempty substring of vx.

7.2 Standard Turing machines

For each of the following languages, give a standard Turing machine that recognises it.

The solutions below have been done in JFLAP. Turing machines in JFLAP are different from those in the book in a number of ways:

- JFLAP uses □ for "blank", instead of B as in the book.
- The JFLAP-tape is infinite to the left as well (in the terminology of the book: it's a 2-way tape). The read head is not on a B or □ at the start, but immediately on the first symbol of the input word. This means that you don't have to start with a $B/B\ R$ transition by default.

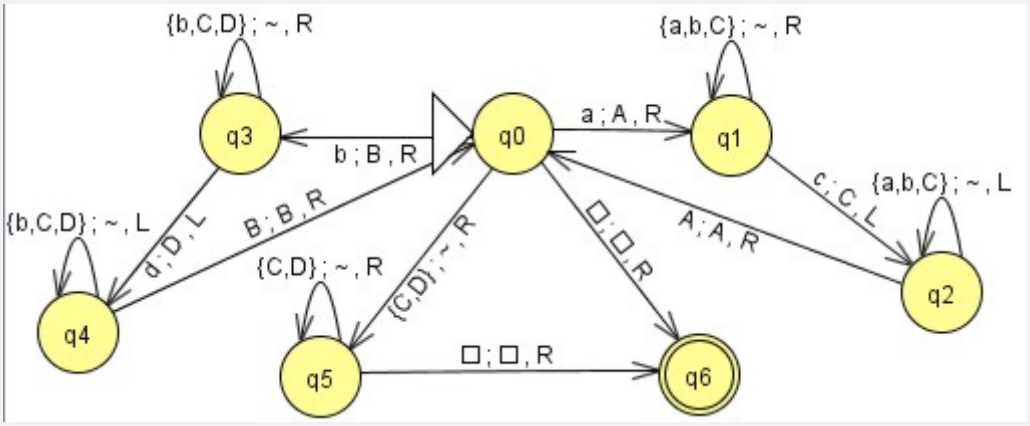
The solutions below make use of a convenient feature of JFLAP: abbreviations can be used on the read- and write position; this yields a template for multiple transitions.

- ˜ on the read position means "an arbitrary symbol"

- $!x$ on the read position means "any symbol but x"
- $\{a, b, c\}$ on the read position means "one of the symbols a, b or c"
- ˜ on the write position means "leave the current symbol"

E. Language L from Exercise B.

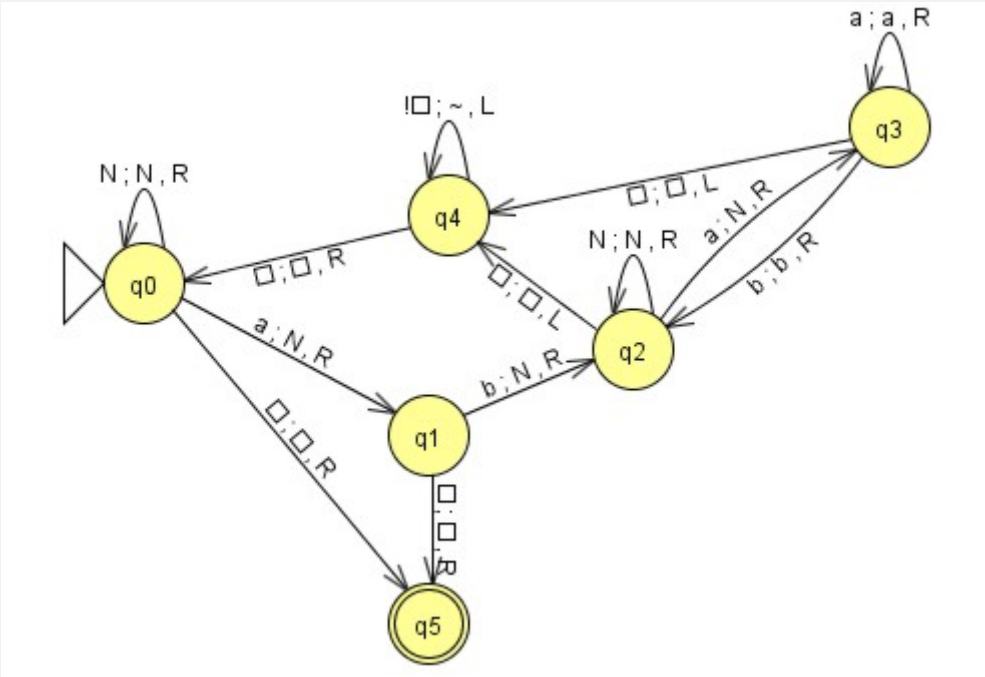
The machine repeatedly replaces an a by an A, then looks for the first c and replaces it by a C, and goes back to the latest A. Then the same thing happens for b/B and d/D. Note that this can only occur after the replacement of the a: the automaton will halt in the left half if there is another a on the tape, and in the right half if there is already a B on the tape. Finally, it is tested whether the rest of the tape now consists of only Cs and Ds.



F. Language L from Exercise D.

The solution below uses an auxiliary symbol N that is used to mark the parts of the input that have already been "explained". The machine reads an initial string of Ns and then replaces one a and one b by N. Subsequently, after every b in the rest of the string, another a is replaced by N. After

this, the read head returns to the start.



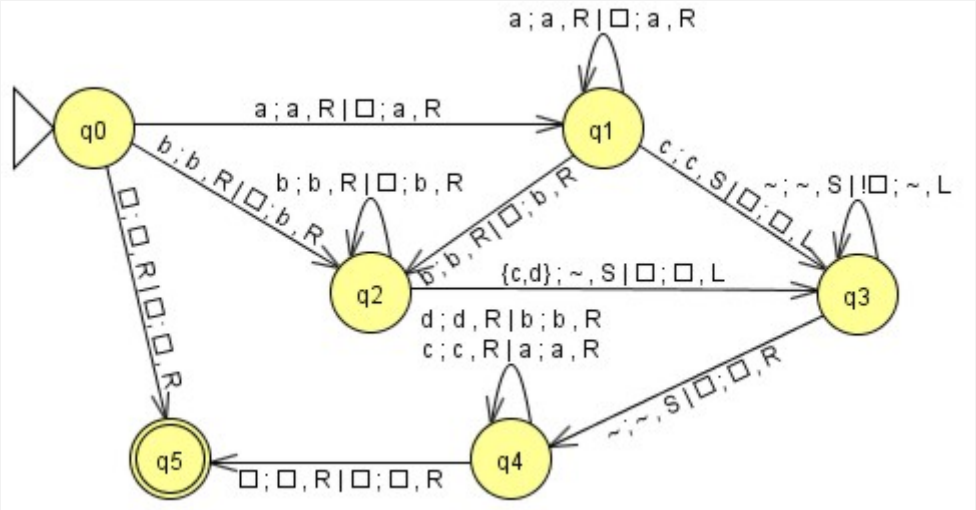
7.3 Multi-tape Turing machines

For each of the following languages, give a multi-tape Turing machine that recognises it. The input tape (tape 1) must only be processed from left to right (i.e. only the directions S and R, but not L, are allowed for tape 1).

G. *(extra)* Language L from Exercise B.

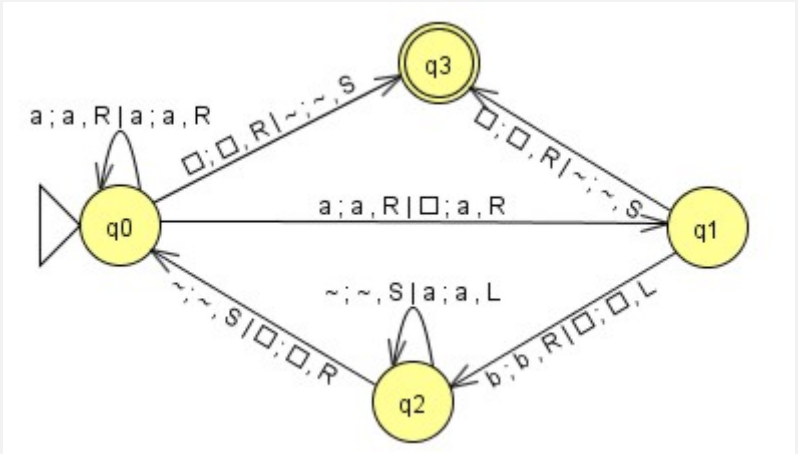
The following solution copies the a and b to tape 2, and then returns the read head from tape 2 to the start. Subsequently, c on tape 1 are read simultaneously with a on tape 2, and d on tape 1 are read simultaneously with b on tape 2. The machine accepts a word if eventually a blank is found on

tape 1 and 2 at the same time.



H. *(extra)* Language L from Exercise D.

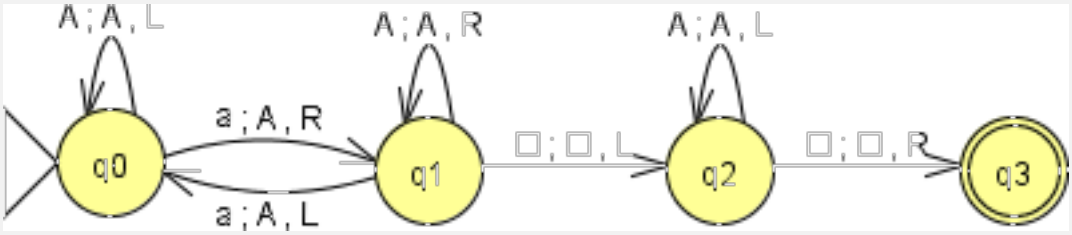
The solution below reads as on tape 1 and 2 synchronously. (At the first iteration, there are no a on tape 2 yet, but later there are.) After this, another a is read on tape 1 and copied to tape 2; then a b on tape 1. Now, tape 2 returns to the start, and the previous is repeated. The machine accepts a word if eventually a blank is found on tape 1.



7.4 Two-way Turing machines

I. Construct a two-way Turing machine with input alphabet $\{a\}$ that accepts an input when it is of the form a^{2n+1} for some $n \geq 0$, but only if the read head is initially on the middle symbol.

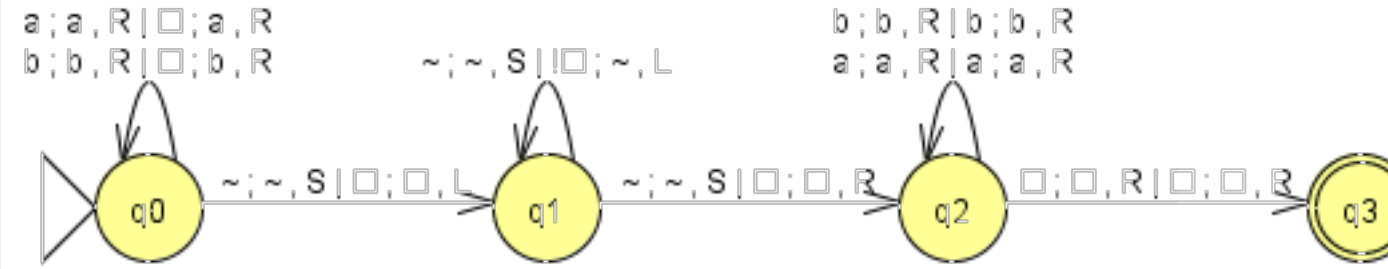
The machine will oscillate, with a growing amplitude: first one step to the right, then two to the left, then three to the right, etc. The count is kept by converting every new a into an A. A word is accepted when a blank is found when moving to the right, and then another blank is found when moving to the left.



7.5 Non-deterministic Turing machines

J. Construct a (possibly nondeterministic) two-tape Turing machine that decides the language $L = \{uu \mid u \in \{a, b\}^*\}$ such that every computation on input w with $n = |w| \geq 3$ terminates after at most $2n + 3$ transitions.

Tape 1 is copied to tape 2. The non-determinism can be used to go back to the start of tape 2 at an arbitrary time and match the rest of tape 1 to the copy on tape 2. This results in the following Turing machine:



(Remember that an additional $B/B\ R|B/B\ R$-step is required at the start according to the method from the book.) Every accepting run (of a word w with$|w| = n$) is $3n/2 + 4$ steps long. (For $n < 3$ this is greater than $2n + 2$.) If the word is not accepted, the maximum run is $n + 1$ steps to copy the complete word, followed by $n + 2$ steps in which the read head of tape 2 returns to the start, so $2n + 3$ steps.