Algorithmics I - Tutorial Sheet 5

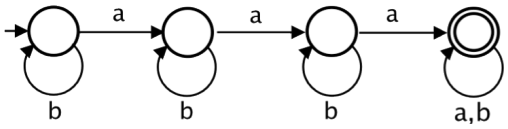
Computability

---

1. Give a DFAs for each of the following languages:
   (a) $L = \{s \mid s \text{ contains at least three } a\text{'s}\}$ where $S = \{a, b\}$;

Solution: The states are $S = \{s_0, s_1, s_2, s_3\}$, with s_0 the initial state and s_3 the only accepting state. The transition relation is given by:

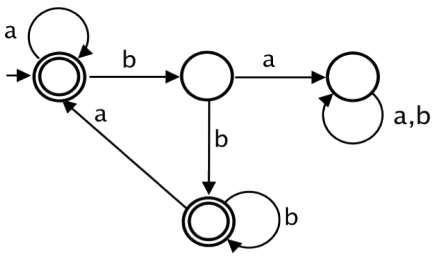
- $T((s_0, a), s_1)$
- $T((s_0, b), s_0)$
- $T((s_1, a), s_2)$
- $T((s_1, b), s_1)$
- $T((s_2, a), s_3)$
- $T((s_2, b), s_2)$
- $T((s_3, a), s_3)$
- $T((s_3, b), s_3)$



   (b) $L = \{s \mid s \text{ contains no singleton } b\text{'s}\}$ where $S = \{a, b\}$ and a singleton b is a b that neither has a b as its immediate predecessor or successor;

Solution: The states are $S = \{s_0, s_1, s_2, s_3\}$, with s_0 the initial state and s_0 and s_3 the accepting states. The transition relation is given by:

- $T((s_0, a), s_0)$
- $T((s_0, b), s_1)$
- $T((s_1, a), s_2)$
- $T((s_1, b), s_3)$
- $T((s_2, a), s_2)$
- $T((s_2, b), s_2)$
- $T((s_3, a), s_0)$
- $T((s_3, b), s_3)$

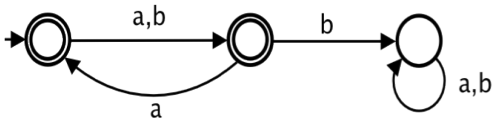


(c) $L = \{s \mid$ every odd position of s contains an $a\}$ where $S = \{a, b\}$;

Solution: The following solution assumes that the initial character is in position zero, an even position.

The states are $S = \{s_0, s_1, s_2\}$, with s_0 the initial state and s_0 and s_1 the accepting states. The transition relation is given by:

- $T((s_0, a), s_1)$
- $T((s_0, b), s_1)$
- $T((s_1, a), s_0)$
- $T((s_1, b), s_2)$
- $T((s_2, a), s_2)$
- $T((s_2, b), s_2)$

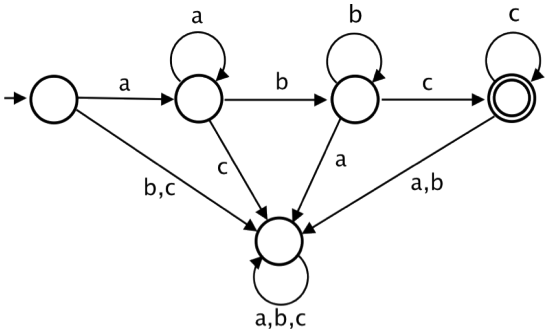


(d) $L = \{a^n b^m c^p \mid n{>}0,\ m{>}0$ and $p{>}0\}$ where $S = \{a, b, c\}$.

Solution: The states are $S = \{s_0, s_2, s_2, s_3, s_4\}$, with s_0 the initial state and s_3 the only accepting states. The transition relation is given by:

- $T((s_0, a), s_1)$
- $T((s_0, b), s_4)$
- $T((s_0, c), s_4)$
- $T((s_1, a), s_1)$
- $T((s_1, b), s_2)$
- $T((s_1, c), s_4)$
- $T((s_2, a), s_4)$
- $T((s_2, b), s_2)$

- $T((s_2, c), s_3)$
- $T((s_3, a), s_4)$
- $T((s_3, b), s_4)$
- $T((s_3, c), s_3)$
- $T((s_4, a), s_4)$
- $T((s_4, b), s_4)$
- $T((s_4, c), s_4)$



2. Assuming a binary input and that the alphabet is $\{0, 1, \#\}$, design Turing machines to compute the following functions:
   (a) $f(k) = k-1$ (you can assume that if the output would be negative the machine enters an error state);

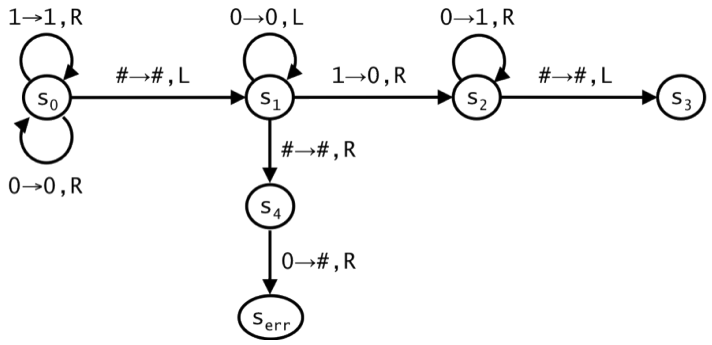
Solution: The Turing Machine here is similar to the one given in lectures for the function $f(k) = k+1$. It is in fact the dual: we find the rightmost 1 change it to 0 and, then `move right` changing all 0's to 1's.

The special case is when there are no 1's in the input, i.e. when the input is a single 0. In the special case, since we can only represent non-negative integers (assuming the alphabet is $\{0, 1, \#\}$) we replace the single 0 on the input with a blank (#) and enter an 'error' state (an alternative would be to add 'minus' to the alphabet).

The pseudo-code is as follows:

```
move right seeking first blank square;
move left looking for first 1 or blank;
if (1 found)
    change the 1 to 0;
    move right changing each 0 to 1;
    halt when blank square reached;
else if (blank found)
    move right change 0 to #;
    move right;
    enter error state and halt;
```

The state transition diagram is shown below:

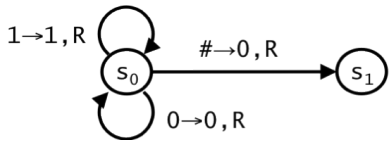


(b) $f(k) = 2{\cdot}k$.

Solution: Multiplication by 2 involves simply appending a 0 to the end of the input. The pseudocode is as follows:

> `move right` seeking first blank square;
> when found `replace` the blank by 0 and `halt`;

The state transition diagram is shown below:



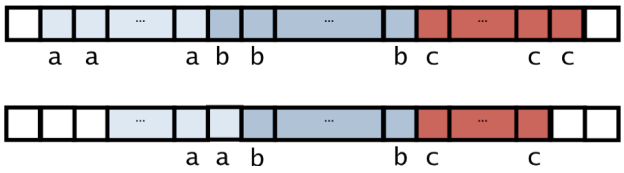
Hint: As for the similar example in the lectures (designing a machine for the function $f(k) = k + 1$), look for a pattern between the binary representations of the input and output.

3. Design a Turing Machine which recognises the following language $L = \{a^n b^n c^n \mid n{\geq}0\}$ where $S = \{a, b, c\}$. More precisely, the machine that will accept precisely those strings that consist of a number of consecutive a's followed by the same number of consecutive b's followed by the same number of consecutive c's.

Hint: one complication is to ensure that one does not introduce blanks (i.e. # symbols) into the string when reading and deleting characters. Therefore, first consider how to read in the sequence $a^n b^m c^k$ and return the sequence $a^{n-1} b^{m-1} c^{k-1}$ without introducing blanks into the middle (the case when $n = 1$ might need to be dealt with as a special case).

Solution: The idea here is to repeatedly delete one a, one b and one c until there is nothing left, but if anything 'goes wrong' along the way, then immediately move into a reject ('no') state.

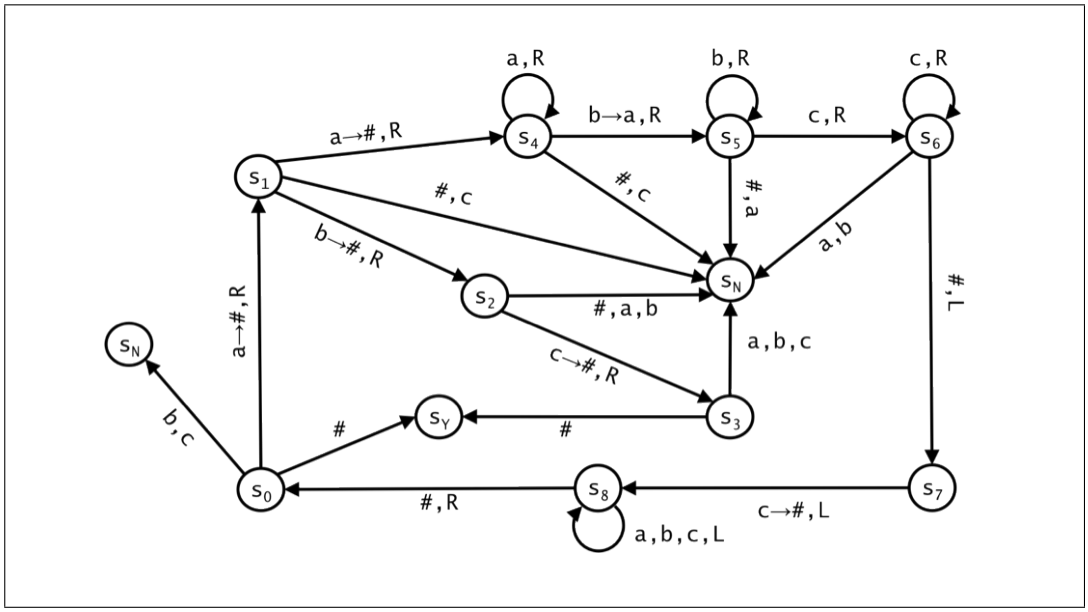
To avoid introducing blanks between non-blanks, in each loop when there are two or more a's at the start, we delete the first two a's, overwrite the first b by an a, and delete the last c. See the example below:



The pseudocode is as follows:

```
if (blank) accept;
else if (b or c) reject;
else // i.e. an 'a' is read
    delete a and move right;
    if (c or blank) reject;
    else if (b) // special case: single 'a' so look for 'bc'
        delete b and move right;
        if (a or b or blank) reject;
        else // i.e. a 'c' is read
            delete c and move right;
            if (a or b or c) reject;
            else accept; // i.e. a 'blank' is read
    else // more than one a at the start
        delete (second) a and move right over a's;
        if (c or blank) reject;
        else // i.e. a 'b' is read after the 'a's
            replace b by a and move right; // replace first b by a
            move right over b's;
            if (a or blank) reject;
            else // i.e. a 'c' is read after the 'b's
                move right over c's;
                if (a or b) reject;
                else // i.e. a 'blank' is read after the 'c's
                    move left;
                    delete c; // remove last c
                    move left over non blanks;
                    move right and restart;
```

The state transition diagram is shown below. Some of the labels are abbreviated: if the direction of movement is immaterial it is omitted, and if the tape symbol is unchanged, the symbol is written without an arrow.


a,R
b,R
c,R
a→#,R
b→a,R
c,R
s1
s4
s5
s6
#,c
#,c
#,a
a,b
b→#,R
sN
s2
#,a,b
#,L
a→#,R
sN
a,b,c
c→#,R
b,c
#
sY
#
s3
s0
s8
s7
#,R
c→#,L
a,b,c,L