Turing Machines and Language Classification

Nicolò Felicioni[1]

Dipartimento di Elettronica e Informazione
Politecnico di Milano

nicolo . felicioni @ polimi . it

March 16, 2021

[1] Mostly based on Nicholas Mainardi's material.

The ultimate computing model

Definition

- Willing to further extend the capabilities of the PDA we substitute the stack with one (or more) tapes
- The resulting automaton is the Turing Machine, which is able to do anything we define as computable[a]
- The TM is able to choose whether to advance, go back or stand still with the cursor on the tape
- Multiple tapes can be added, but they do not increase the computing power

[a] or, at least, Alonso Church believed so, and we pretty much agree

The ultimate computing model

Definition

- For greater comfort, we use a marker symbol Z_0 to indicate the beginning of the tape (in case it's not there, we can add it)
- We assume that the input is positioned on the input tape
- All the other tapes, if any, are filled with the blank symbol ƀ
- All the actions on the tapes contents are driven by a state machine
- Notation:$<\mathit{input\_tape}><\mathit{tape}_1>, \ldots, <\mathit{tape}_n> \mid <\mathit{tape}_1>, \ldots, <\mathit{tape}_n>, (R \mid S \mid L)^{n+1}$

Formalization

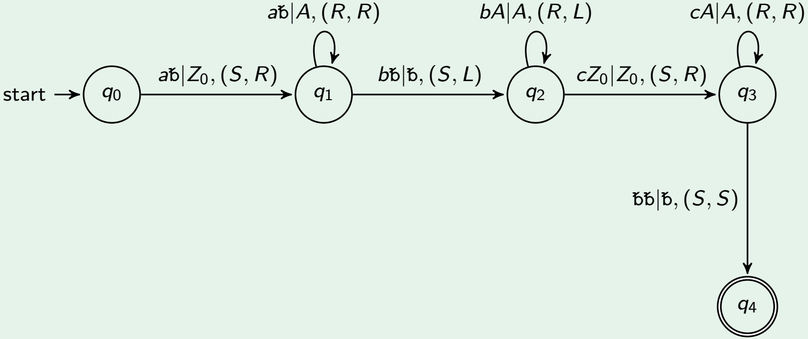
Definition

- A k-tapes recognizer TM is formally defined as a 6-tuple $(\mathbf{Q}, \mathbf{I}, \Gamma, \delta, q_0, \mathbf{F})$, where:
    - $\mathbf{Q}$, is the set of states of the automata
    - $\mathbf{I} \cup \{ƀ\}$ is the alphabet of the input string which will be checked
    - $\Gamma \cup \{ƀ\}$ is the alphabet of the symbols on the tapes
    - $\delta : \mathbf{Q} \times \mathbf{I} \times \Gamma^k \mapsto \mathbf{Q} \times \Gamma^k \times \{R, S, L\}^{k+1}$ the transition function
    - $q_0 \in \mathbf{Q}$ the (unique) initial state from where the automaton starts
    - $\mathbf{F} \subseteq \mathbf{Q}$ the set of final accepting states of the automaton

A first example

$L = a^n b^n c^n, n \geq 1$

- The trick is that we can sweep over the memory tape as many times as we want



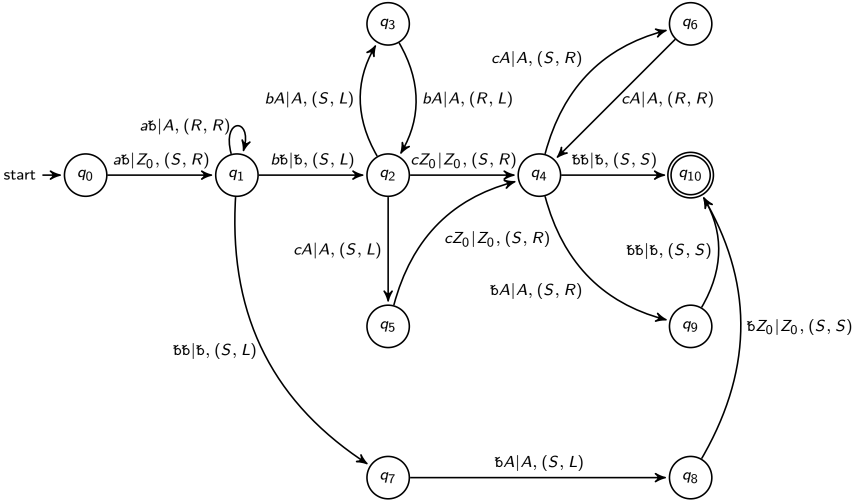
start
q_0
q_1
q_2
q_3
q_4
$a\mathfrak{b}|Z_0,(S,R)$
$a\mathfrak{b}|A,(R,R)$
$b\mathfrak{b}|\mathfrak{b},(S,L)$
$bA|A,(R,L)$
$cZ_0|Z_0,(S,R)$
$cA|A,(R,R)$
$\mathfrak{b}\mathfrak{b}|\mathfrak{b},(S,S)$

A bit more complex

$L = a^n b^{n \div 2} c^{n \div 2}, n \geq 1$

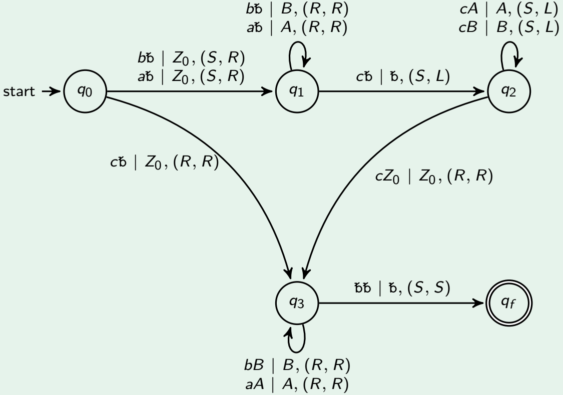
- A slight deviation from the common idea of counting
- More than a single solution possible (it's the same as saying : "devise a program that writes such strings")
- One method to perform division: stop on the input tape, continue in memory
- Other methods involve writing twice the symbols on the tape

A bit more complex



The Parrot Language

$L = \{wcw \mid w \in \{a, b\}^*\}$


$b\mathfrak{b} \mid B, (R, R)$
$a\mathfrak{b} \mid A, (R, R)$
$cA \mid A, (S, L)$
$cB \mid B, (S, L)$
$b\mathfrak{b} \mid Z_0, (S, R)$
$a\mathfrak{b} \mid Z_0, (S, R)$
$c\mathfrak{b} \mid \mathfrak{b}, (S, L)$
start
q_0
q_1
q_2
$c\mathfrak{b} \mid Z_0, (R, R)$
$cZ_0 \mid Z_0, (R, R)$
$\mathfrak{b}\mathfrak{b} \mid \mathfrak{b}, (S, S)$
q_3
q_f
$bB \mid B, (R, R)$
$aA \mid A, (R, R)$

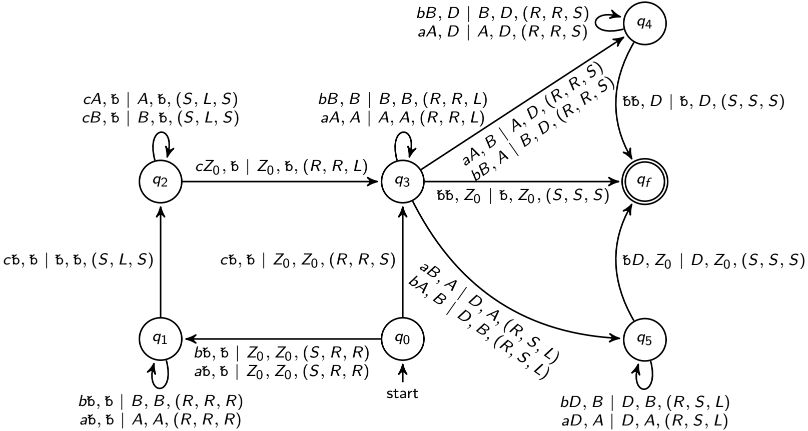
An Enriched Parrot Language

What if we add the language $L_1 = \{wcw^R \mid w \in \{a, b\}^*\}$ to the parrot language? We need to recognize $L = \{wcw \mid w \in \{a, b\}^*\} \cup \{wcw^R \mid w \in \{a, b\}^*\}$.

- There is an issue: We cannot infer from the input string if we have to match L_1 or the parrot language
- This is no longer a problem with a TM!
- Indeed, we can deal with both cases without guessing a-priori which sublanguage should be recognized
- How can we do this? ....

An Enriched Parrot Language

We can use 2 tapes instead of only one! The first one acts as a queue, the second one as a stack:


$bB, D \mid B, D, (R, R, S)$
$aA, D \mid A, D, (R, R, S)$
q_4
$cA, \mathfrak{b} \mid A, \mathfrak{b}, (S, L, S)$
$cB, \mathfrak{b} \mid B, \mathfrak{b}, (S, L, S)$
$bB, B \mid B, B, (R, R, L)$
$aA, A \mid A, A, (R, R, L)$
$aA, B \mid A, D, (R, R, S)$
$bB, A \mid B, D, (R, R, S)$
$\mathfrak{bb}, D \mid \mathfrak{b}, D, (S, S, S)$
q_2
$cZ_0, \mathfrak{b} \mid Z_0, \mathfrak{b}, (R, R, L)$
q_3
$\mathfrak{bb}, Z_0 \mid \mathfrak{b}, Z_0, (S, S, S)$
q_f
$c\mathfrak{b}, \mathfrak{b} \mid \mathfrak{b}, \mathfrak{b}, (S, L, S)$
$c\mathfrak{b}, \mathfrak{b} \mid Z_0, Z_0, (R, R, S)$
$aB, A \mid D, A, (R, S, L)$
$bA, B \mid D, B, (R, S, L)$
$\mathfrak{b}D, Z_0 \mid D, Z_0, (S, S, S)$
q_1
q_0
q_5
$b\mathfrak{b}, \mathfrak{b} \mid Z_0, Z_0, (S, R, R)$
$a\mathfrak{b}, \mathfrak{b} \mid Z_0, Z_0, (S, R, R)$
start
$b\mathfrak{b}, \mathfrak{b} \mid B, B, (R, R, R)$
$a\mathfrak{b}, \mathfrak{b} \mid A, A, (R, R, R)$
$bD, B \mid D, B, (R, S, L)$
$aD, A \mid D, A, (R, S, L)$

An Enriched Parrot Language

However, we may wonder: can L be recognized with a 2 tapes TM, but not with a 1 tape TM?

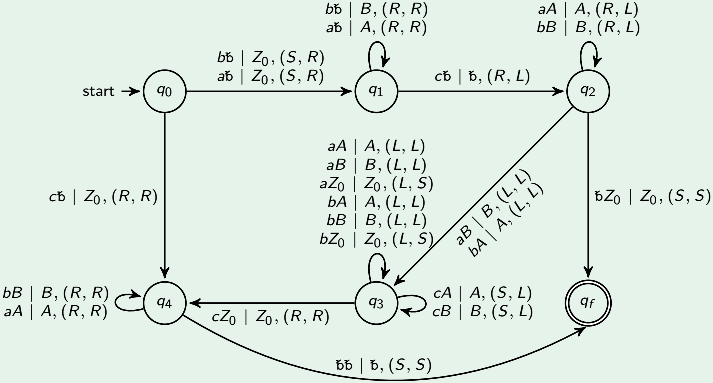
$$\Downarrow$$

No! Recall that all types of TMs are equivalent. A recognizer for L can be built with only 1 tape, it is just more complex than using an additional tape. How?

- The TM starts recognizing the sublanguage L_1, using the tape as a stack
- In case of a failure, the head of the tape is moved back to the first cell, while the head of the input tape is moved back to the first cell after the one containing the c character.
- The tape can now be used as a queue to recognize the parrot language.

An Enriched Parrot Language

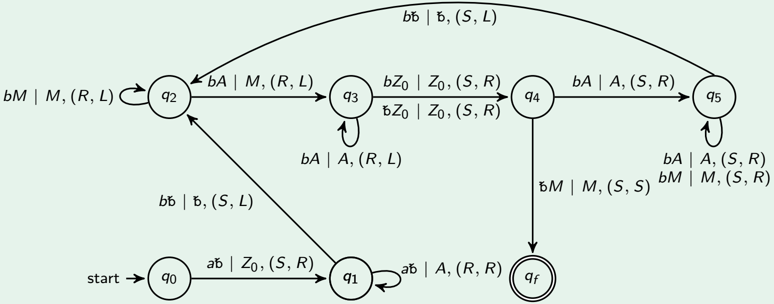
$L = \{wcw \mid w \in \{a,b\}^*\} \cup \{wcw^R \mid w \in \{a,b\}^*\}$ with a 1 tape TM



A Powerful Computational Model

$L = \{a^n b^{n^2}, n \geq 1\}$ with a 1 tape TM

Basic Idea: At each iteration, we read n times b by employing the tape as a stack. We keep track of how many iterations have been performed through a marker symbol M, which replaces the A symbol used to count.



$b\mathfrak{b} \mid \mathfrak{b}, (S, L)$
$bM \mid M, (R, L)$
q_2
$bA \mid M, (R, L)$
q_3
$bZ_0 \mid Z_0, (S, R)$
$\mathfrak{b}Z_0 \mid Z_0, (S, R)$
q_4
$bA \mid A, (S, R)$
q_5
$bA \mid A, (R, L)$
$bA \mid A, (S, R)$
$bM \mid M, (S, R)$
$\mathfrak{b}M \mid M, (S, S)$
$b\mathfrak{b} \mid \mathfrak{b}, (S, L)$
start
q_0
$a\mathfrak{b} \mid Z_0, (S, R)$
q_1
$a\mathfrak{b} \mid A, (R, R)$
q_f

Complement

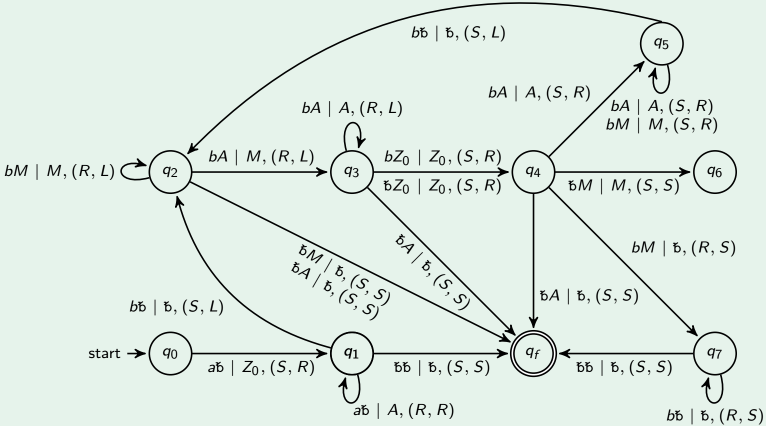
- To design the automaton for the complement of a language, we cannot follow the same procedure seen for FSA and PDA
- unless we are sure the TM always halts, but we will see later in the course this is not always possible
- Therefore, it is better to design the automaton without an algorithm, but using the original one as a building block

$L = \{a^n b^{n^2}, n \geq 1\}^C$

We can see L as $L_1 \cup L_2, L_1 = \{a^n b^m \mid m \neq n^2, m \geq 0, n \geq 1\}, L_2 = (a^+ b^*)^C$ Then, we can build an automaton for L_1 starting from the original one, and merge it with the FSA recognizing L_2

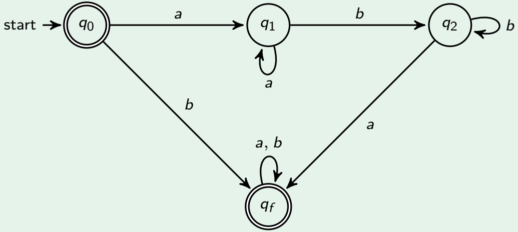
Complement

Automaton for $L_1 = \{a^n b^m \mid m \neq n^2, m \geq 0, n \geq 1\}$


$b\mathfrak{b} \mid \mathfrak{b}, (S, L)$
q_5
$bA \mid A, (S, R)$
$bA \mid A, (S, R)$
$bM \mid M, (S, R)$
$bA \mid A, (R, L)$
$bM \mid M, (R, L)$
q_2
$bA \mid M, (R, L)$
q_3
$bZ_0 \mid Z_0, (S, R)$
$\mathfrak{b}Z_0 \mid Z_0, (S, R)$
q_4
$\mathfrak{b}M \mid M, (S, S)$
q_6
$\mathfrak{b}M \mid \mathfrak{b}, (S, S)$
$\mathfrak{b}A \mid \mathfrak{b}, (S, S)$
$\mathfrak{b}A \mid \mathfrak{b}, (S, S)$
$bM \mid \mathfrak{b}, (R, S)$
$\mathfrak{b}A \mid \mathfrak{b}, (S, S)$
$b\mathfrak{b} \mid \mathfrak{b}, (S, L)$
start
q_0
$a\mathfrak{b} \mid Z_0, (S, R)$
q_1
$\mathfrak{b}\mathfrak{b} \mid \mathfrak{b}, (S, S)$
q_f
$\mathfrak{b}\mathfrak{b} \mid \mathfrak{b}, (S, S)$
q_7
$a\mathfrak{b} \mid A, (R, R)$
$b\mathfrak{b} \mid \mathfrak{b}, (R, S)$

Complement

Automaton to Recognize $L_2 = (a^+ b^*)^C$

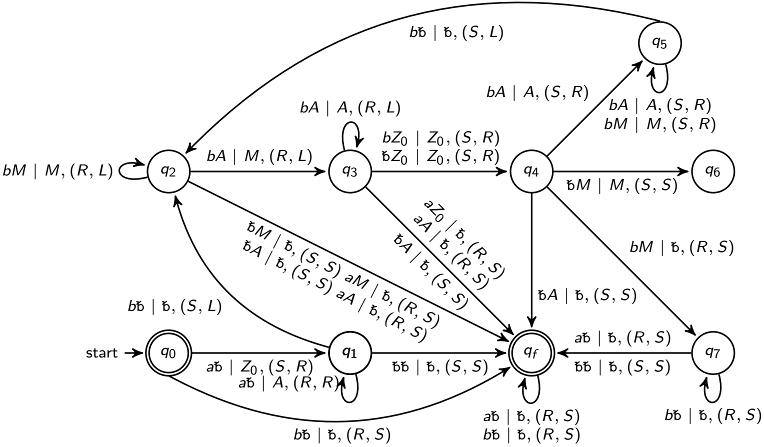


Strategy to combine TM for L_1 and FSA for L_2:

- We can now replace the states q_1 and q_2 with the automaton recognizing L_1, with a transition reading an a allowed from each possible state of the automaton for L_1 where a b has already been read.
- Beware of final states while combining the two automata ...

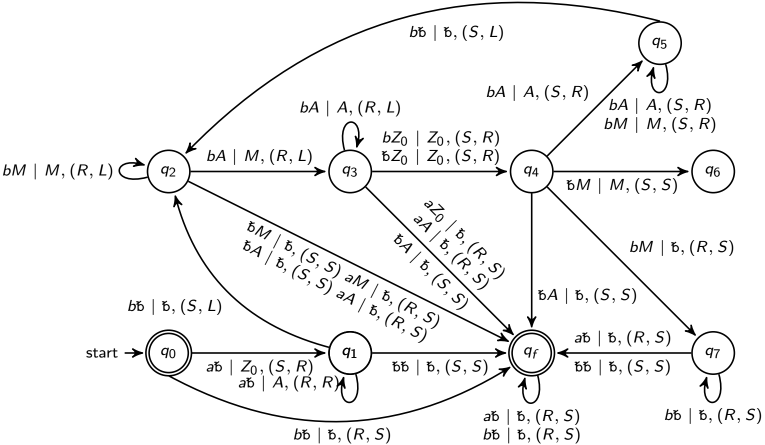
Complement

TM for $L = \{a^n b^{n^2}, n \geq 1\}^C$?



Complement

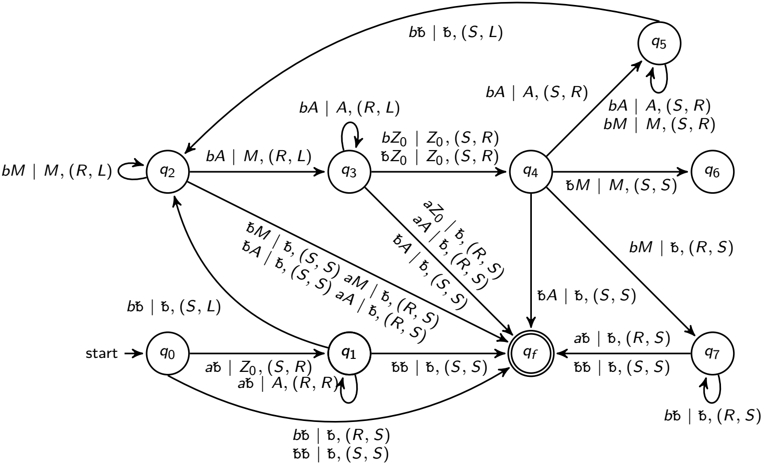
TM for $L = \{a^n b^{n^2}, n \geq 1\}^C$?



$q_0 \in F \Rightarrow$ This TM always halts in $q_0 \Rightarrow$ TM accepts $L = \{a, b\}^*$

Complement

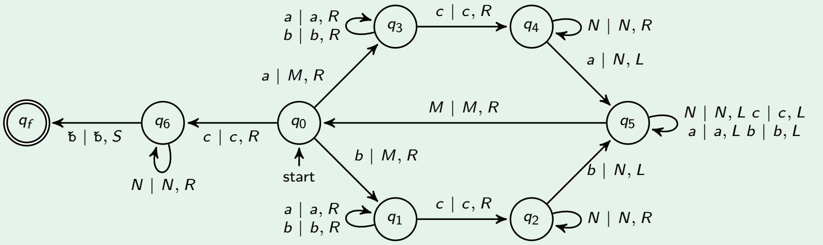
Correct TM for $L = \{a^n b^{n^2}, n \geq 1\}^C$:



Single Tape TM

- We have only the input tape available, which is obviously writable.
- They are equivalent to k-tapes TM
- The convention is $\langle$input tape read$\rangle \mid \langle$input tape write$\rangle$, $(R \mid S \mid L)$

Example: $L = \{wcw \mid w \in \{a, b\}^*\}$



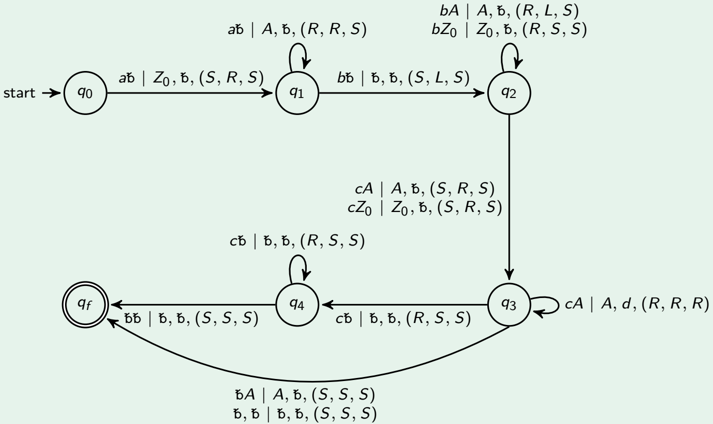
Transducer TM

Definition

- A k-tapes transducer TM is formally defined as a 8-tuple $(\mathbf{Q}, \mathbf{I}, \Gamma, \delta, q_0, \mathbf{F}, \eta, \mathbf{O})$, where:
  - $\mathbf{Q}$, is the set of states of the automata
  - $\mathbf{I} \cup \{\mathfrak{b}\}$ is the alphabet of the input string which will be read
  - $\Gamma \cup \{\mathfrak{b}\}$ is the alphabet of the symbols on the tapes
  - $\delta : \mathbf{Q} \times \mathbf{I} \times \Gamma^k \mapsto \mathbf{Q} \times \Gamma^k \times \{R, S, L\}^{k+1}$ the transition function
  - $q_0 \in \mathbf{Q}$ the (unique) initial state from where the automaton starts
  - $\mathbf{F} \subseteq \mathbf{Q}$ the set of final accepting states
  - $\mathbf{O}$ the output alphabet (may coincide with $\mathbf{I}$)
  - $\eta : \mathbf{Q} \times \mathbf{I} \times \Gamma^k \mapsto \mathbf{O}^* \times \{R, S\}$, the translation function, which emits characters on the output tape of the automaton
- Notation: $\langle input\_tape\rangle\langle tape_1\rangle \ldots \langle tape_k\rangle \mid \langle tape_1\rangle, \ldots, \langle tape_k\rangle, \langle output\_tape\rangle, (R \mid S \mid L)^k + 1(R \mid S)$

An Example

$\tau(a^n b^k c^m) = d^{min(n,m,k)} \mid n, m, k \geq 1$


start
q_0
$a\text{ƀ} \mid Z_0, \text{ƀ}, (S, R, S)$
q_1
$a\text{ƀ} \mid A, \text{ƀ}, (R, R, S)$
$b\text{ƀ} \mid \text{ƀ}, \text{ƀ}, (S, L, S)$
q_2
$bA \mid A, \text{ƀ}, (R, L, S)$
$bZ_0 \mid Z_0, \text{ƀ}, (R, S, S)$
$cA \mid A, \text{ƀ}, (S, R, S)$
$cZ_0 \mid Z_0, \text{ƀ}, (S, R, S)$
q_3
$cA \mid A, d, (R, R, R)$
$c\text{ƀ} \mid \text{ƀ}, \text{ƀ}, (R, S, S)$
q_4
$c\text{ƀ} \mid \text{ƀ}, \text{ƀ}, (R, S, S)$
$\text{ƀƀ} \mid \text{ƀ}, \text{ƀ}, (S, S, S)$
q_f
$\text{ƀ}A \mid A, \text{ƀ}, (S, S, S)$
$\text{ƀ}, \text{ƀ} \mid \text{ƀ}, \text{ƀ}, (S, S, S)$

A Surprising Development

Consider the language $L = N_{\texttt{bin}} a^N$, that is an integer N in its binary representation which specifies the length of the subsequent sequence of a. Do we really need a TM for it?

- We simply needs to read N and count a number of a equal to N
- if N is bounded, we can do it with an FSA
- If N is unbounded, we need at least a stack to store it in its binary form and decrementing it by 1 each time an a is read. The string is accepted if and only if the binary number 0 is found on the stack when the string has been entirely read.
- Can we decrement using only a stack? ...

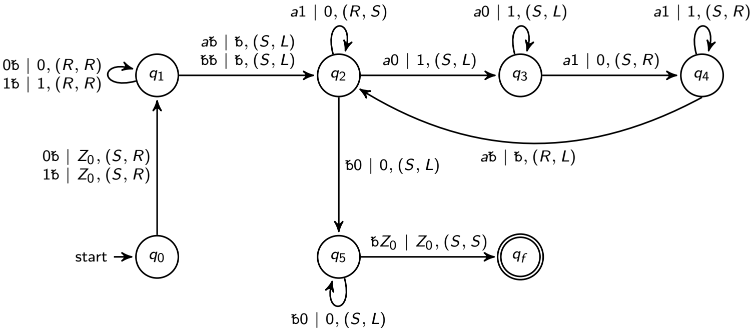
A Surprising Development

... Turns out we cannot decrement with a PDA!

- The issue stands generally in borrows and carries, respectively for decrementing and incrementing
- Indeed, subtracting by 1 can be done to the least significant digit unless it is 0. In this case, subtracting by 1 requires a borrow from the closest more significant digit.
- If this digit is 0 too, the borrow is required to the next more significant digit until a digit different from 0 is found.
- If we have to do it with a stack, we erase the memory in order to go back to the first non-zero digit
- For binary numbers, this is not an issue since all the erased cells are 1. However, how many cells have we just erased?
- This information is lost with just a stack, thus a PDA is not sufficient!

A Surprising Development

Let's design the TM to recognize this language $L = N_{\texttt{bin}} a^N$:


$a1 \mid 0, (R, S)$
$a0 \mid 1, (S, L)$
$a1 \mid 1, (S, R)$
$a\mathfrak{b} \mid \mathfrak{b}, (S, L)$
$\mathfrak{b}\mathfrak{b} \mid \mathfrak{b}, (S, L)$
$0\mathfrak{b} \mid 0, (R, R)$
$1\mathfrak{b} \mid 1, (R, R)$
q_1
q_2
$a0 \mid 1, (S, L)$
q_3
$a1 \mid 0, (S, R)$
q_4
$0\mathfrak{b} \mid Z_0, (S, R)$
$1\mathfrak{b} \mid Z_0, (S, R)$
$\mathfrak{b}0 \mid 0, (S, L)$
$a\mathfrak{b} \mid \mathfrak{b}, (R, L)$
start
q_0
q_5
$\mathfrak{b}Z_0 \mid Z_0, (S, S)$
q_f
$\mathfrak{b}0 \mid 0, (S, L)$

A Surprising Development

The language L is really common to be found in protocol formats!

- For instance, IP messages usually contain a length field in their header
- However, this length field employs a fixed number of bytes (e.g. one octet), thus the number N is bounded

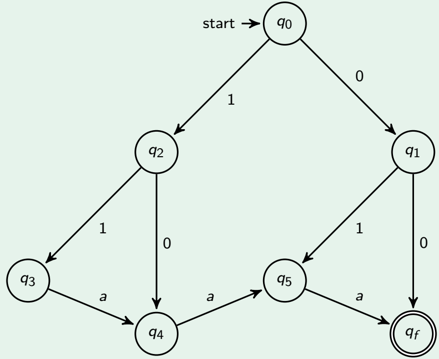
$$\Downarrow$$

This language can be recognized by a FSA!

Ok, but how many states?

A Surprising Development

$L = N_{\texttt{bin}} a^N, |N_{\texttt{bin}}| = 2$

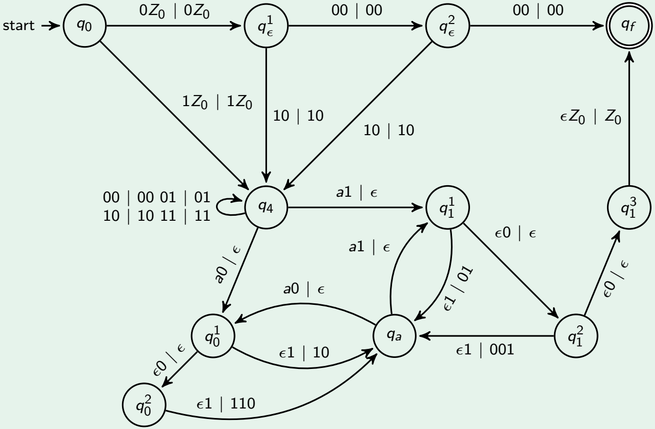


If the upper bound is N, then we need approximately $2N$ states → not feasible for common protocols which require 4-octets length fields

A Surprising Development

Can we do it with a PDA employing about $|N_{\texttt{bin}}|$ states? Yes

$L = N_{\texttt{bin}} a^N, |N_{\texttt{bin}}| = 3$



Identifying Minimum Computational Power Required For a Language

Associate each language with the minimum power automaton needed to recognize it:

1. $L_1 = \{a^n b^{\frac{n}{2}} c^{\frac{n}{2}} \mid n \geq 1\}$
2. $L_2 = \{a^n b^n \mid n \geq 1\} \cup \{a^n c^n \mid n \geq 1\}$
3. $L_3 = \{a^n b^n \mid 1 \leq n \leq 20\}$
4. $L_4 = \{a^n b^n b^n \mid n \geq 1\}$
5. $L_5 = \{a^n b^{2n} a^n\}$
6. $L_6 = \{a^i b^j c^k \mid i + k = j\}$

Identifying Minimum Computational Power Required For a Language

Associate each language with the minimum power automaton needed to recognize it:

1. $L_1 = \{a^n b^{\frac{n}{2}} c^{\frac{n}{2}} \mid n \geq 1\} \implies$ TM : We have already done it
2. $L_2 = \{a^n b^n \mid n \geq 1\} \cup \{a^n c^n \mid n \geq 1\} \implies$ PDA : The two sub-languages can both be recognized by a PDA. We can decide among them by reading a b or c
3. $L_3 = \{a^n b^n \mid 1 \leq n \leq 20\} \implies$ FSA : There is an upper bound on the number of a and b
4. $L_4 = \{a^n b^n b^n \mid n \geq 1\} \implies$ PDA: It is equivalent to $a^n b^{2n}$
5. $L_5 = \{a^n b^{2n} a^n\} \implies$ TM: Same reason as $a^n b^n c^n$, i.e. we cannot count the a after we count the b
6. $L_6 = \{a^i b^j c^k \mid i + k = j\} \implies$ PDA: $L_6 = \{a^i b^i b^k c^k\}$, that is equivalent to recognizing two times $a^n b^n$ with unrelated n