Regular expressions: what for?

- They are used in any task that require *pattern matching*:
  - search engines,
  - protein analysis in bioinformatics,
  - search and replace functionalities in word processors and text editors,
  - text processing utilities like `sed` and `awk` in Unix and Linux.
- Many programming languages provide various regular expression capabilities.
- Lexical analysers in compilers also use regular expressions.
- ...

Finite representation of languages

Consider the following language:

L = all strings of $\square$s and $\diamond$s that have two or three occurrences of $\diamond$, the first and second of which are not consecutive

Can we describe this **infinite language** with some kind of **finite 'pattern'?**

- Strings in L can start with any number (possibly none) of $\square$s: $\square^*$
- Then comes the first $\diamond$.

  It should be alone, so must be followed by at least one $\square$: $\square^*\diamond\square$
- Then any number (possibly none) of $\square$s: $\square^*\diamond\square\square^*$
- Then comes the second $\diamond$: $\square^*\diamond\square\square^*\diamond$
- Then
  - **either** no more $\diamond$s but there can be any number (possibly none) $\square$s,
  - **or**, after some (possibly none) $\square$s, there is a third $\diamond$ somewhere, followed by any number (possibly none) $\square$s:

$\square^*\diamond\square\square^*\diamond(\square^* \cup \square^*\diamond\square^*)$ $\rightsquigarrow$ **regular expression** representing L

From language to regular expression: Example 2

L = all the strings consisting of some number (possibly none) of as, followed by some number (possibly none) of bs

$= \{\varepsilon, a, b, aa, bb, ab, aaa, aab, bbb, abb, \ldots, aaaabbbbbbb, \ldots\}$

Let's try: $\underline{a^*b^*}$

$\rightsquigarrow$ L is represented by the regular expression a^*b^*

Notation: $\boxed{L = \text{Language\_of}(a^*b^*)}$

read: "the language of a^*b^* is L"

or "the language represented by a^*b^* is L"

From regular expression to language

Example 3: Language_of$\big(a(a^* \cup b^*)\big)$ $= ?$

$=$ all words starting with a, followed by either a (possibly empty) word of as, or a (possibly empty) word of bs

$= \{a,\ aa,\ ab,\ aaa,\ abb,\ aaaa,\ abbb,\ aaaaaa,\ abbbbbbb,\ \ldots\}$

Example 4: Language_of$\big(a(a \cup b)^*\big)$ $= ?$

$=$ all words starting with a, followed by any word over $\{a, b\}$

$=$ all words over the alphabet $\{a, b\}$ starting with a

$= \{a,\ aa,\ ab,\ aaa,\ aab,\ aba,\ abb,\ aaaaaa,\ abbaabab,\ aabbababba,\ \ldots\}$

The two examples are NOT the same!

From regular expression to language: Example 5

Language_of$\big((b \cup aaa^*)^*\big) \;=\; ?$

- Let's start with Language_of(aaa^*): all words of as of length ≥ 2

  $= \{aa,\ aaa,\ aaaa,\ aaaaa,\ \ldots\}$

- Language_of$(b \cup aaa^*)$: as above, plus the word b

  $= \{b,\ aa,\ aaa,\ aaaa,\ aaaaa,\ \ldots\}$

- Language_of$\big((b \cup aaa^*)^*\big)$: we can take any number (possibly none) of words from Language_of$(b \cup aaa^*)$ and concatenate them

Some questions checking our understanding of this language:

- Is there a word of as and bs that is not in Language_of$\big((b \cup aaa^*)^*\big)$?
- Do the words ε, $aabbaaabab$, and $bbbabbaaabaa$ belong to the language Language_of$\big((b \cup aaa^*)^*\big)$?

Summary: describing regular expressions by induction

Each regular expression (over an alphabet S) is a string consisting of

- symbols from S,
- plus symbols from $\cup$, $*$, $($, $)$, ε, $\emptyset$.

The set of all **regular expressions over** S is defined inductively:

- *Basis:* $\emptyset$, ε and each symbol in S are regular expressions.
- *Inductive step:* If R and Z are regular expressions, then so are $(R \cup Z)$, (RZ), and (R^*).
- *Closure:* No other string is a regular expression.

FOR EXAMPLE:

- Over $S = \{a, b\}$: $\quad \emptyset, \quad \varepsilon, \quad ((a^*)(b^*)), \quad (((a((a \cup b)^*))b)(b^*))$
- Over $S = \{x, y\}$: $\quad \emptyset, \quad \varepsilon, \quad (((x(y^*))y)(y \cup z)), \quad (((x(y^*))y)(y \cup z))$
- Over $S = \{\diamond, \square\}$: $\quad \emptyset, \quad \varepsilon, \quad ((\square \cup (\diamond^*))((\diamond\square)^*))$

Regular expressions: notational conventions

These brackets are pretty incomprehensible. So we introduce some conventions:

- We omit the outermost brackets.
- We omit the brackets when concatenate expressions,

  so e.g. we write $abab$ instead of $(((ab)(ab))b)$.
- * 'binds tighter' than concatenation,

  so e.g. we write aab^* instead of $(aa)(b^*)$.

For example: (cf. the previous slide)

$$a^*b^*, \qquad a(a \cup b)^*bb^*, \qquad xy^*y(y \cup z), \qquad (\Box \cup \Diamond^*)(\Diamond\Box)^*$$

But take care: say, $(aa)^*b$ and aa^*b are NOT the same!

Always use brackets if in any doubt!

From regular expression to language: Example 6

$$\text{Language\_of}\big((b \cup a)aa^*\big) \;=\; ?$$

Let's do it, 'from inside out':

- $\text{Language\_of}(b \cup a) \;=\; \{a, b\}$
- $\text{Language\_of}\big((b \cup a)a\big) \;=\; \{aa, ba\}$
- $\text{Language\_of}\big(b \cup a)aa^*\big)$ = all words starting with aa followed by a (possibly empty) word of as, and all words starting with ba followed by a (possibly empty) word of as.

Compare this language with $\text{Language\_of}(b \cup aaa^*)$ (see Example 5).

Brackets DO MATTER: the two languages are NOT the same!

Summary: regular expressions represent languages

Every regular expression over some alphabet S **represents** a language over S.

These are two different things:

a regular expression R	$\longleftrightarrow$	the language R represents: Language_of(R)
a string consisting of symbols from S and $\cup,\ *,\ (,\ ),\ \varepsilon,\ \emptyset$		a set of words over S

FOR EXAMPLE:

$1(1 \cup 2)^*$	Language_of$\big(1(1 \cup 2)^*\big)$ $=$	all words of 1s and 2s starting with 1
$(a \cup b)bba$	Language_of$\big((a \cup b)bba\big)$ $=$	$\{abba,\ bbba\}$
finite 'pattern'		words following the pattern (can be infinitely many)

Summary: how regular expressions represent languages

Inductive definition of the language $\boxed{\text{Language\_of}(R)}$ represented by the regular expression R:

We follow the inductive definition of regular expressions on slide 224.

Basis: $\text{Language\_of}(\emptyset) = \emptyset$

$\text{Language\_of}(\varepsilon) = \{\varepsilon\}$

$\text{Language\_of}(a) = \{a\}$ for any symbol a in the alphabet S.

Inductive step: If R and Z are regular expressions, then

- $\text{Language\_of}(R \cup Z)$ = all words that belong either to $\text{Language\_of}(R)$ or to $\text{Language\_of}(Z)$
- $\text{Language\_of}(RZ)$ = any word from $\text{Language\_of}(R)$ followed by any word from $\text{Language\_of}(Z)$
- $\text{Language\_of}(R^*)$ = any word from $\text{Language\_of}(R)$ followed by any word from $\text{Language\_of}(R)$ followed by any word from $\text{Language\_of}(R)$ ...

the number of iterations is arbitrary (possibly none)

From regular expression to language: Example 7

$$\text{Language\_of}\big(\varepsilon \cup c^*(a \cup bc^*)\big) \;=\; ?$$

all the strings that are either empty or start with some (possibly none) cs, followed by either an a, or a b followed by some (possibly none) cs

$$= \{\varepsilon,\ a,\ b,\ bc,\ bcc,\ ca,\ cb,\ cbcc,\ \ldots\}$$

NOT in $\text{Language\_of}\big(\varepsilon \cup c^*(a \cup bc^*)\big)$: $\quad ab,\ \ aa,\ \ caac,\ \ \ldots$

Regular languages

A **regular language**
is *any* language that can be represented by a regular expression.

In other words, a language L is **regular** if $L = \text{Language\_of}(R)$
for some regular expression R.

For instance, all the languages in Examples 1–7 above are regular.

NOT every language is regular !

Regular languages and finite automata

(1) **There is a general 'mechanical' procedure $\mathcal{P}_1$ that converts any regular expression R to an NFA A_R such that $L(A_R) = \text{Language\_of}(R)$.**

So every regular language is accepted by some automaton.

Moreover, there is a general way back:

(2) **There is a general 'mechanical' procedure $\mathcal{P}_2$ that converts any NFA A to a regular expression R_A such that $\text{Language\_of}(R_A) = L(A)$.**

So every language that is accepted by some automaton is regular.

In other words:

Regular languages are precisely those languages that are accepted by finite automata.

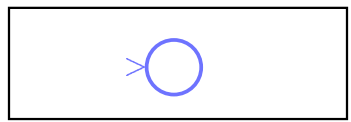
From regular expressions to NFA

The procedure which
converts any regular expression R to an NFA A_R such that $L(A_R) = \text{Language\_of}(R)$
operates along the inductive definition of the regular expression R

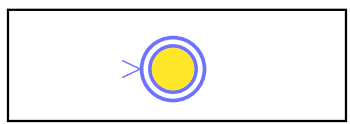
(again, see slide 224):

Basis cases:

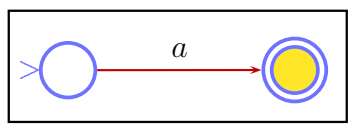
- If $R = \emptyset$. Then $\text{Language\_of}(R) = \emptyset$. A_R :



- If $R = \varepsilon$. Then $\text{Language\_of}(R) = \{\varepsilon\}$. A_R :

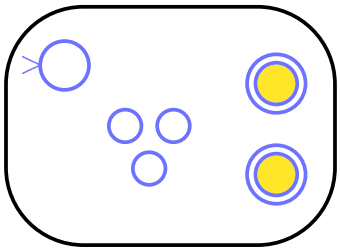


- If $R = a$. Then $\text{Language\_of}(R) = \{a\}$. A_R :

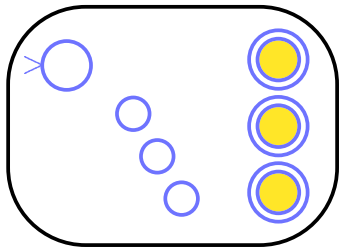


Inductive cases: Automaton accepting Language_of$(R \cup Z)$

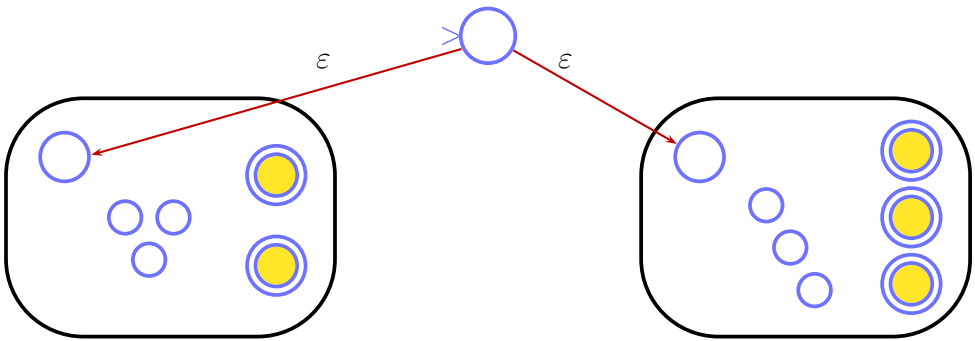
$\boxed{A_R:}$ accepts Language_of(R)



$\boxed{A_Z:}$ accepts Language_of(Z)

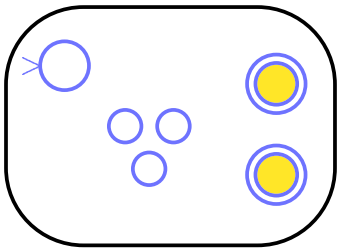


$\boxed{A_{R \cup Z}:}$ accepts Language_of$(R \cup Z)$

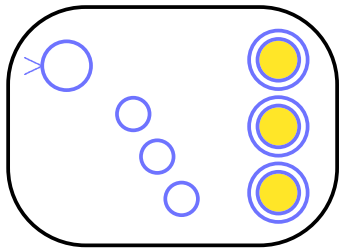


Inductive cases: Automaton accepting Language_of(RZ)

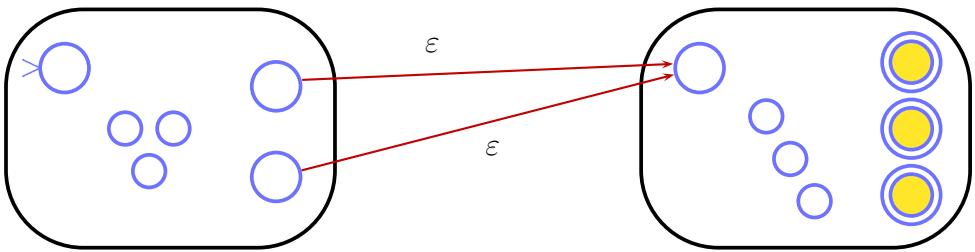
$\boxed{A_R:}$ accepts Language_of(R)



$\boxed{A_Z:}$ accepts Language_of(Z)

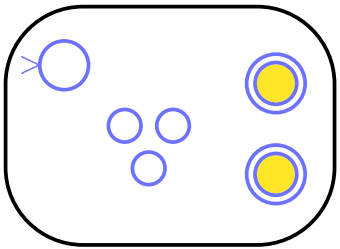


$\boxed{A_{RZ}:}$ accepts Language_of(RZ)

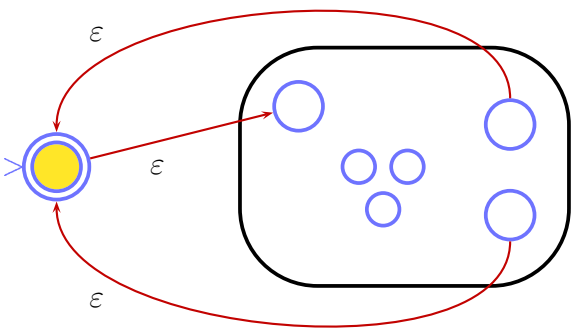


Inductive cases: Automaton accepting Language_of(R^*)

$\boxed{A_R:}$ accepts Language_of(R)



$\boxed{A_{R^*}:}$ accepts Language_of(R^*)



Example: how the procedure works

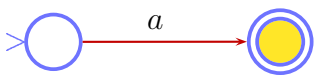
We apply the procedure to the regular expression

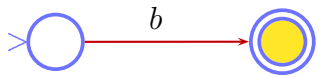
$$\big((a \cup ab)^* ba\big)^*$$

We construct step by step (going 'inside out') an NFA A such that

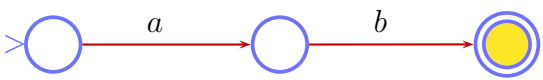
$$L(A) \;=\; \text{Language\_of}\big(((a \cup ab)^* ba)^*\big)$$

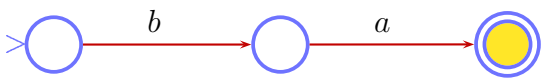
Step 0: automata accepting Language_of(a) and Language_of(b)



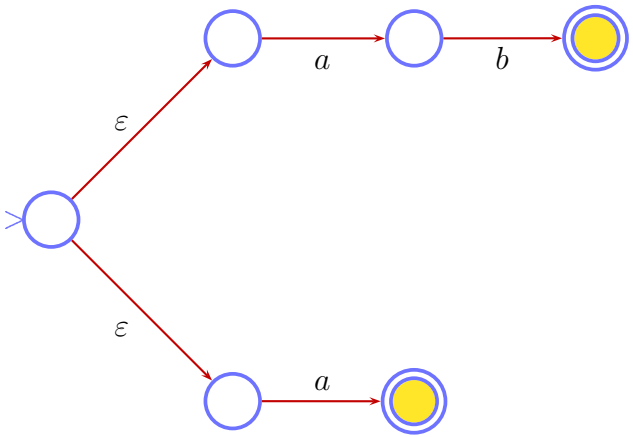


Step 1: automata accepting Language_of(ab) and Language_of(ba)

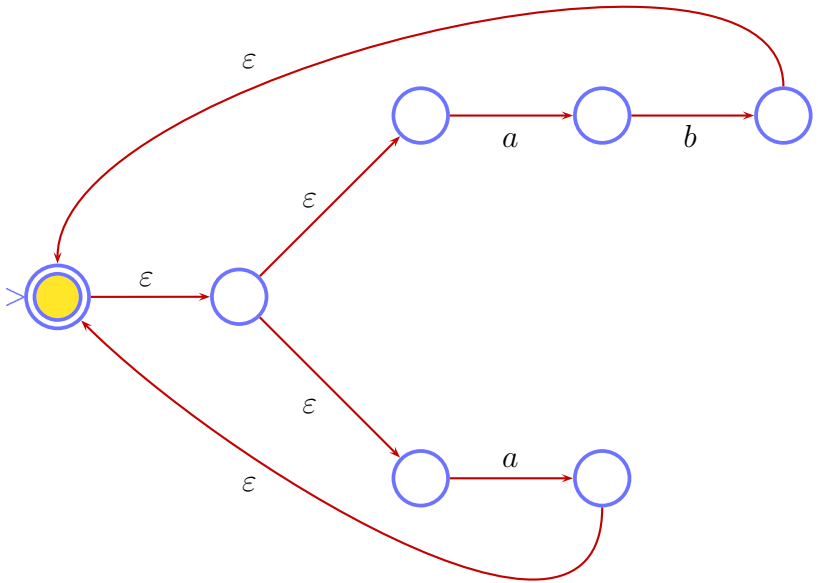




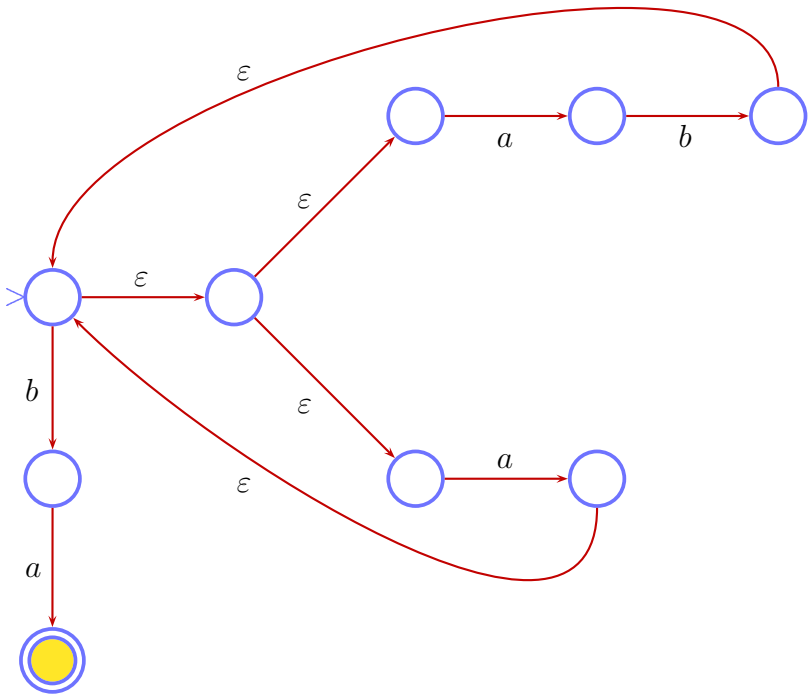
Step 2: automaton accepting Language_of$(a \cup ab)$



Step 3: automaton accepting Language_of$\big((a \cup ab)^*\big)$

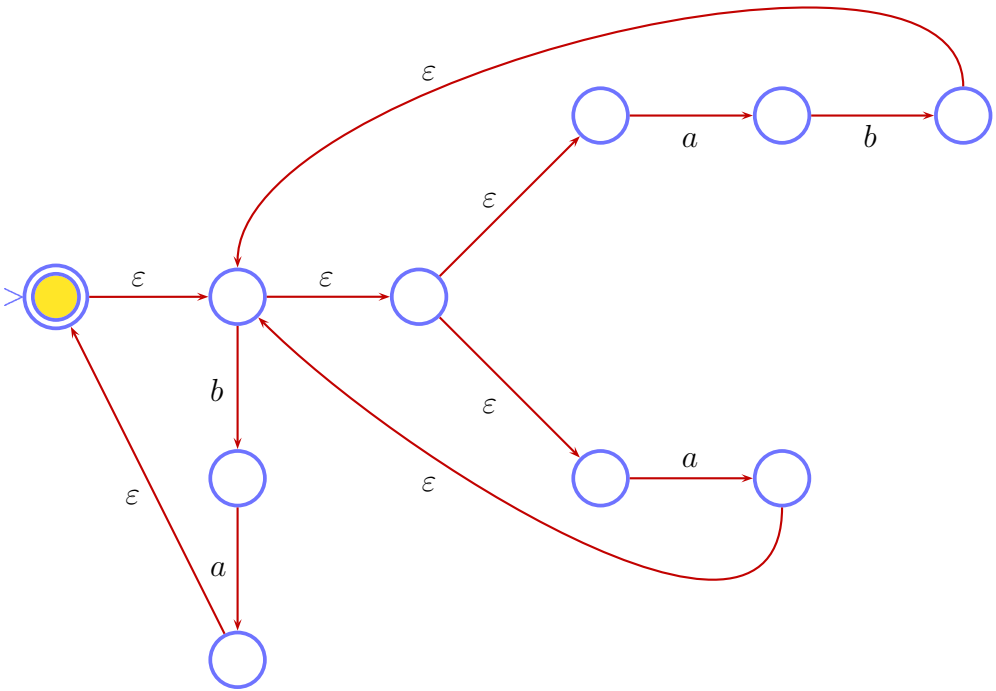


Step 4: automaton accepting Language_of$\big((a \cup ab)^*ba\big)$



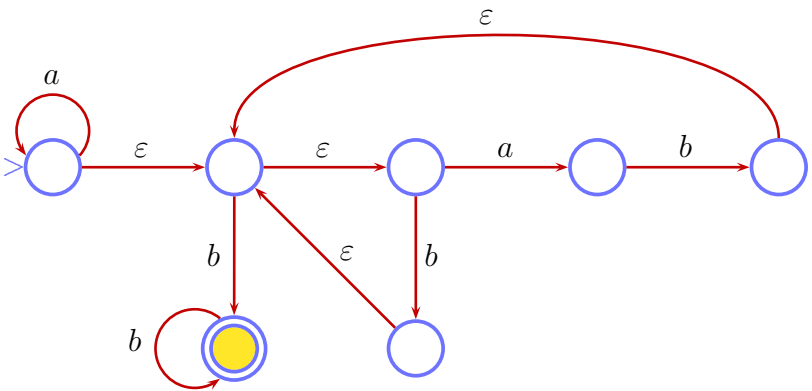
Final step of the procedure

Automaton accepting Language_of$\Big(\big((a \cup ab)^* ba\big)^*\Big)$:



Another example

Automaton accepting Language_of$\left(a^*(b \cup ab)^*bb^*\right)$:



Finite automata and regular languages: summary

- Finite automata may be regarded as programs that use **fixed amounts of memory** (represented by states) regardless of the input.
- Finite automata can be used as **recognition devices:** they accept certain inputs and reject others.
- **Nondeterminism** does not increase the computational power of finite automata, but nondeterministic automata are easier to design than **deterministic** ones.
- The languages accepted by finite automata are precisely the **regular** ones.

Nonregular languages

> Since finite automata are theoretical models for programs using a constant amount of memory regardless of the input, one ought to expect that the power of such programs should be quite limited.

Since regular languages are all accepted by some finite automaton, there must be languages which are **not regular**, they cannot be given with the help of *any* regular expression.

Nonregular languages: an example

Consider the following language over the alphabet $\{a, b\}$:

$$\begin{aligned} L \;=\; & \text{all words starting with a word of } a\text{'s} \\ & \text{followed by an equal-length word of } b\text{'s} \\ =\; & \{a^n b^n \mid n = 0, 1, 2, \dots\} \end{aligned}$$

Suppose that a finite automaton A tries to recognise words in this language. Then A must store the entire prefix a^n before the first b shows up. Otherwise A will not be able to compare the length of the coming word of bs with the length of the prefix of as.

As each automaton is capable for a fixed finite amount of storage with the help of its states, no automaton A exists such that $L(A) = L$.

There is a precise mathematical proof justifying this *informal* argument:

this language L is indeed **not regular.**