Formal Languages and Automata

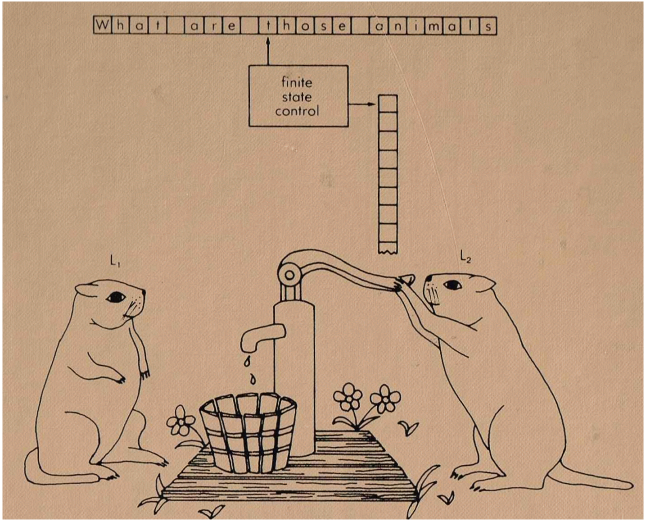
Pumping Lemma

Ryan Stansifer

Computer Sciences
Florida Institute of Technology
Melbourne, Florida USA 32901

http://www.cs.fit.edu/~ryan/

2 March 2024

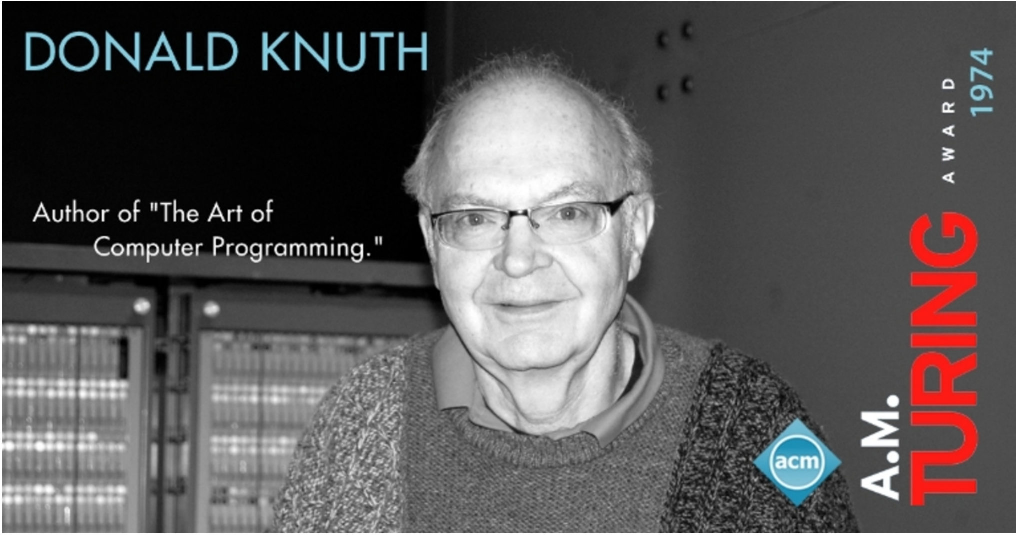


Silly pun on the cover of Harrison's book

A detour to some of the wisdom of Donald Knuth.

The point in my words: “Writing a computer program *or a proof* requires understanding the solution to a problem so well you can explain it to a mindless automaton, and yet express it so eloquently a fellow human can rapidly apprehend the method.”

So just this once, with just this one theorem, we strive to make excellent proofs using the pumping lemma. [Skip to his quotes.]

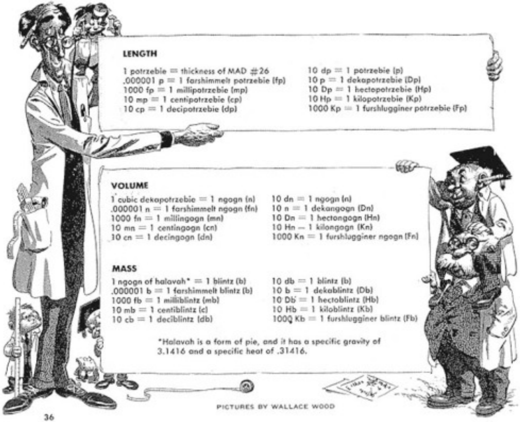

DONALD KNUTH
Author of "The Art of
Computer Programming."
acm
A.M.
TURING
AWARD
1974

THE POTRZEBIE SYSTEM
OF WEIGHTS AND MEASURES

THE POTRZEBIE SYSTEM

This new system of measuring, which is destined to become the measuring system of the future, has decided improvements over the other systems now in use. It is based upon measurements taken 6-9-12 at the Physics Lab. of Milwaukee Lutheran High School, in Milwaukee, Wis., when the thickness of MAD Magazine #26 was determined to be 2.26334851-7438173216473 mm. This length is the basis for the entire system, and is called one potrzebie of length.

The Potrzebie has also been standardized at 3515.-3502 wave lengths of the red line in the spectrum of cadmium. A partial table of the Potrzebie System, the measuring system of the future, is given below .



LENGTH

1 potrzebie = thickness of MAD #26
.000001 p = 1 farshimmelt potrzebie (fp)
1000 fp = 1 millipotrzebie (mp)
10 mp = 1 centipotrzebie (cp)
10 cp = 1 decipotrzebie (dp)
10 dp = 1 potrzebie (p)
10 p = 1 dekapotrzebie (Dp)
10 Dp = 1 hectepotrzebie (Hp)
10 Hp = 1 kilopotrzebie (Kp)
1000 Kp = 1 furshlugginer potrzebie (Fp)

VOLUME

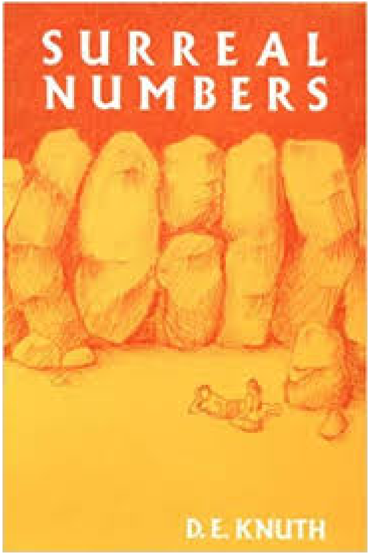
1 cubic dekapotrzebie = 1 ngogn (n)
.000001 n = 1 farshimmelt ngogn (fn)
1000 fn = 1 millingogn (mn)
10 mn = 1 centingogn (cn)
10 cn = 1 decingogn (dn)
10 dn = 1 ngogn (n)
10 n = 1 dekangogn (Dn)
10 Dn = 1 hectongogn (Hn)
10 Hn = 1 kilongogn (Kn)
1000 Kn = 1 furshlugginer ngogn (Fn)

MASS

1 ngogn of halavah* = 1 blintz (b)
.000001 b = 1 farshimmelt blintz (b)
1000 fb = 1 milliblintz (mb)
10 mb = 1 centiblintz (c)
10 cb = 1 deciblintz (db)
10 db = 1 blintz (b)
10 b = 1 dekablintz (Db)
10 Db = 1 hectoblintz (Hb)
10 Hb = 1 kiloblintz (Kb)
1000 Kb = 1 furshlugginer blintz (Fb)

*Halavah is a form of pie, and it has a specific gravity of 3.1416 and a specific heat of .31416.

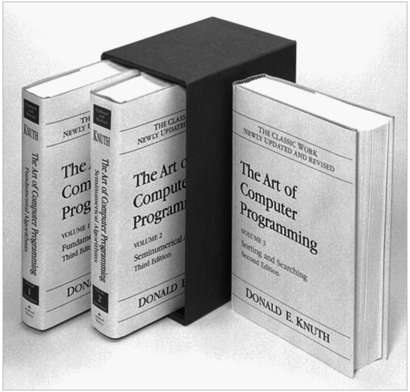
PICTURES BY WALLACE WOOD


SURREAL
NUMBERS
D. E. KNUTH

Donald E. Knuth (1938–)

Introduction to Knuth's organ composition YouTube [17 minutes]


TEX Friendly Zone
NEED A TUG?
TEX
http://www.tug.org/


THE CLASSIC WORK
NEWLY UPDATED AND REVISED
The Art of
Computer
Programming
VOLUME 3
Sorting and Searching
Second Edition
DONALD E. KNUTH
VOLUME 2
Third Edition
KNUTH

Science is knowledge which we understand so well that we can teach it to a computer; and if we don't fully understand something, it is an art to deal with it.

Knuth, Turing Award Lecture, 1974.

Science is what we understand well enough to explain to a computer. Art is everything else we do.

Knuth, 1995, foreword to the book $A = B$, page xi.

[Software] is harder than anything else I've ever had to do.

Knuth, *Notices of the AMS*, 49 (3), 2002, page 320, 2002

Let us change our traditional attitude to the construction of programs: Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.

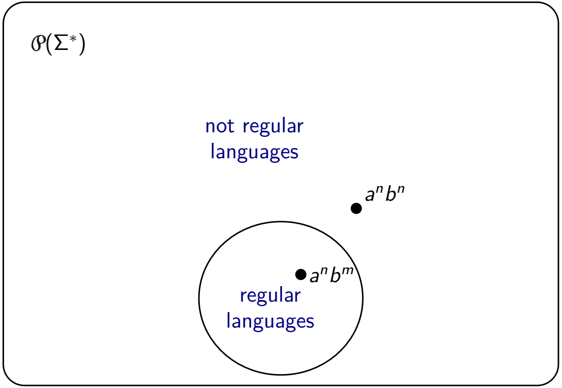
Knuth, "Literate Programming," *The Computer Journal*, volume 27, 1984.

The point in my words: "Writing a computer program *or a proof* requires understanding the solution to a problem so well you can explain it to a mindless automaton, and yet express it so eloquently a fellow human can rapidly apprehend the method."

Biographies appear in:

- O'Regan, *Giants of Computing: A Compendium of Select, Pivotal Pioneers*, 2013
- Shasha and Lazere, *Out of Their Minds: The Lives and Discoveries of 15 Great Computer Scientists*, 1995
- Slater, *Portraits in Silicon*, 1987

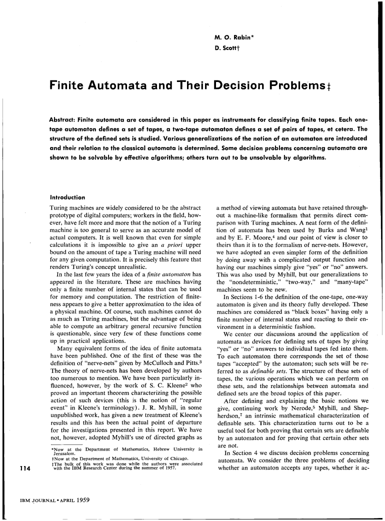
There are languages that are *not* regular.
Negative results are difficult to prove.



The Pumping Lemma

It was first proved by Rabin and Scott in 1959.

Michael Oser Rabin and Dana Stewart Scott (Apr. 1959). "Finite Automata and Their Decision Problems". In: *IBM Journal of Research and Development* 3, pages 114–125. DOI: `10.1147/rd.32.0114`. Reprinted in E. F. Moore, editor, *Sequential Machines: Selected Papers*, Addison-Wesley, 1964



M. O. Rabin*
D. Scott†

Finite Automata and Their Decision Problems‡

Abstract: Finite automata are considered in this paper as instruments for classifying finite tapes. Each one-tape automaton defines a set of tapes, a two-tape automaton defines a set of pairs of tapes, et cetera. The structure of the defined sets is studied. Various generalizations of the notion of an automaton are introduced and their relation to the classical automata is determined. Some decision problems concerning automata are shown to be solvable by effective algorithms; others turn out to be unsolvable by algorithms.

Introduction

Turing machines are widely considered to be the abstract prototype of digital computers; workers in the field, however, have felt more and more that the notion of a Turing machine is too general to serve as an accurate model of actual computers. It is well known that even for simple calculations it is impossible to give an *a priori* upper bound on the amount of tape a Turing machine will need for any given computation. It is precisely this feature that renders Turing's concept unrealistic.

In the last few years the idea of a *finite automaton* has appeared in the literature. These are machines having only a finite number of internal states that can be used for memory and computation. The restriction of finiteness appears to give a better approximation to the idea of a physical machine. Of course, such machines cannot do as much as Turing machines, but the advantage of being able to compute an arbitrary general recursive function is questionable, since very few of these functions come up in practical applications.

Many equivalent forms of the idea of finite automata have been published. One of the first of these was the definition of "nerve-nets" given by McCulloch and Pitts.[3] The theory of nerve-nets has been developed by authors too numerous to mention. We have been particularly influenced, however, by the work of S. C. Kleene[2] who proved an important theorem characterizing the possible action of such devices (this is the notion of "regular event" in Kleene's terminology). J. R. Myhill, in some unpublished work, has given a new treatment of Kleene's results and this has been the actual point of departure for the investigations presented in this report. We have not, however, adopted Myhill's use of directed graphs as a method of viewing automata but have retained throughout a machine-like formalism that permits direct comparison with Turing machines. A neat form of the definition of automata has been used by Burks and Wang[1] and by E. F. Moore,[4] and our point of view is closer to theirs than it is to the formalism of nerve-nets. However, we have adopted an even simpler form of the definition by doing away with a complicated output function and having our machines simply give "yes" or "no" answers. This was also used by Myhill, but our generalizations to the "nondeterministic," "two-way," and "many-tape" machines seem to be new.

In Sections 1-6 the definition of the one-tape, one-way automaton is given and its theory fully developed. These machines are considered as "black boxes" having only a finite number of internal states and reacting to their environment in a deterministic fashion.

We center our discussions around the application of automata as devices for defining sets of tapes by giving "yes" or "no" answers to individual tapes fed into them. To each automaton there corresponds the set of those tapes "accepted" by the automaton; such sets will be referred to as *definable sets*. The structure of these sets of tapes, the various operations which we can perform on these sets, and the relationships between automata and defined sets are the broad topics of this paper.

After defining and explaining the basic notions we give, continuing work by Nerode,[5] Myhill, and Shepherdson,[7] an intrinsic mathematical characterization of definable sets. This characterization turns out to be a useful tool for both proving that certain sets are definable by an automaton and for proving that certain other sets are not.

In Section 4 we discuss decision problems concerning automata. We consider the three problems of deciding whether an automaton accepts any tapes, whether it ac-

*Now at the Department of Mathematics, Hebrew University in Jerusalem.
†Now at the Department of Mathematics, University of Chicago.
‡The bulk of this work was done while the authors were associated with the IBM Research Center during the summer of 1957.

114

IBM JOURNAL • APRIL 1959


MICHAEL RABIN & DANA SCOTT
Introduced nondeterministic machines by publishing "Finite Automata and Their Decision Problem."
acm
A.M. TURING AWARD
1976

The Pumping Lemma (Regular Languages)

Linz 6th, section 4.3, theorem 4.8, page 118.
HMU 3rd, section 4.1.1, theorem 4.1, page 128.
Aho & Ullman, section 2.3.3, theorem 2.7, page 128.
Kozen, theorem 11.1, page 70.
Sipser 3rd, theorem 1.7, page 77.
Floyd & Beigel, section 4.9, theorem 4.47, page 293.
Hein 4th, theorem 11.4.3, page 793.
Sudkamp 3rd, section 5.6, theorem 6.6.3, page 207.
McCormick, section 9.6, claim 9.5, page 185.
Drobot, theorem 3.1, page 75.
Salomaa, theorem 3.12, page 62.
Pumping lemma for regular languages

The pumping lemma was rediscovered by Bar-Hillel, Perles, and Shamir as as simplification of their pumping lemma for context-free languages (HMU).

Yehoshua Bar-Hillel, Micha A. Perles, and Eliyahu Shamir (1961). "On Formal Properties of Simple Phrase Structure Grammars". In: *Zeitschrift für Phonologie, Sprachwissenschaft und Kommunikationsforschung* 14, pages 143–172. Reprinted in **BarHillel:1964:LI** and **Luce:1963:RMP**

Rules of Logic

The pumping lemma as a piece of mathematics in relatively complicated.

Mistakes can easily be made especially if one is not facile with logic.

It is especially awkward to state clearly the nature of proofs involving lternating quantifiers and negation.

We could get my being sloppy, but

1. math and proofs are formal languages,
2. importnat to related areas of CS: model checking, theorem proving, etc.,
3. ignorance of the rules results in mistakes and confusion.

We could get my being sloppy, but

1. math and proofs are formal languages,
2. importnat to related areas of CS: model checking, theorem proving, etc.,
3. ignorance of the rules results in mistakes and confusion.

So, we next look at some logic.

We could get my being sloppy, but

1. math and proofs are formal languages,
2. importnat to related areas of CS: model checking, theorem proving, etc.,
3. ignorance of the rules results in mistakes and confusion.

So, we next look at some logic. But quickly and superficially because that is another course of study.

Implication

Statement	If p then q	If you believe you will fail, you will fail.	
Converse	If q then p	If you fail, you must have stopped believing in yourself.	
Contrapositive	If not-q then not-p	If you don't fail, you must have believed in yourself.	
Inverse	If not-p then not-q	If you believe in yourself, you will succeed.	

Implication

$$A \Rightarrow B$$

- The statement $A \Rightarrow B$ is logically equivalent to $(\neg A)$ or B
- The contrapositive is $(\neg B) \Rightarrow (\neg A)$ or equivalently $(\neg A)$ or B
- The converse is $B \Rightarrow A$
- The inverse is $\neg A \Rightarrow \neg B$ or equivalently $(\neg A)$ and B

The contrapositive of a statement is logically equivalent to it. The converse and the inverse are not necessarily equivalent to the statement.

Truth Tables

P	Q	$P \wedge Q$
$\bot$	$\bot$	$\bot$
$\bot$	$\top$	$\bot$
$\top$	$\bot$	$\bot$
$\top$	$\top$	$\top$

P	Q	$P \Rightarrow Q$
$\bot$	$\bot$	$\top$
$\bot$	$\top$	$\top$
$\top$	$\bot$	$\bot$
$\top$	$\top$	$\top$

The first row represents the case when both propositions P and Q are considered false. The last row represents the case when both are true.

For two propositions there are $2^2 = 4$ rows in the truth table. There are $2^{2^2} = 16$ possible distinct outcomes in the last column of the truth table.

Quantifiers

Some politician is crooked	$\exists x\,(p(x) \wedge q(x))$
No politician is crooked	$\forall x\,(p(x) \Rightarrow \neg q(x))$
All politicians are crooked	$\forall x\,(p(x) \Rightarrow q(x))$
Not all politicians are crooked	$\exists x\,(p(x) \vee \neg q(x))$
Every politician is crooked	$\forall x\,(p(x) \Rightarrow q(x))$
There is an honest politician	$\exists x\,(p(x) \wedge \neg q(x))$
No politician is honest	$\forall x\,(p(x) \Rightarrow q(x))$
All politicians are honest	$\forall x\,(p(x) \Rightarrow \neg q(x))$

For-all, Implication

$$\forall x (A(x) \Rightarrow B(x))$$

- Logically equivalent to $\forall x ((\neg A(x))$ or $B(x))$
- The contrapositive is $\forall x (\neg B(x)) \Rightarrow (\neg A(x))$
- The converse is $\forall x (B(x) \Rightarrow A(x))$ or $\forall x (A(x)$ or $(\neg B(x)))$
- The negation is $\neg \forall x (A(x) \Rightarrow B(x))$ or $\exists x (A(x)$ and $(\neg (B(x))))$
- The negation of the converse $\neg \forall x (B(x) \Rightarrow A(x))$ or $\exists x ((\neg A(x))$ and $B(x))$

For-all, There-exists, Implication

$$\forall x(A(x) \Rightarrow \exists y(B(x,y) \text{ and } C(x,y)))$$

- Logically equivalent to $\forall x(\neg A(x) \text{ or } \exists y(B(x,y) \text{ and } C(x,y)))$
- Negation is

$$\neg\forall x(\neg A(x) \text{ or } \exists y(B(x,y) \text{ and } C(x,y)))$$

$$\exists x\ \neg(\neg A(x) \text{ or } \exists y(B(x,y) \text{ and } C(x,y)))$$

$$\exists x(A(x) \text{ and } \neg\exists y(B(x,y) \text{ and } C(x,y)))$$

$$\exists x(A(x) \text{ and } \forall y\ \neg(B(x,y) \text{ and } C(x,y)))$$

$$\exists x(A(x) \text{ and } \forall y(\neg B(x,y) \text{ or } \neg C(x,y)))$$

$$\exists x(A(x) \text{ and } \forall y(B(x,y) \Rightarrow \neg C(x,y)))$$

Rules “dilemma”. Logic Deamon. Fitch.

Derived Rules **(6.3.2)**

Modus Tollens (MT) (Latin for “mode that denies”) $$\frac{A \to B, \quad \neg B}{\neg A}$$	*Proof by Cases (Cases)* $$\frac{A \vee B, \quad A \to C, \quad B \to C}{C}$$
Hypothetical Syllogism (HS) $$\frac{A \to B, \quad B \to C}{A \to C}$$	*Constructive Dilemma (CD)* $$\frac{A \vee B, \quad A \to C, \quad B \to D}{C \vee D}$$

Proof Writing Guidelines

- Keep the reader informed.

Donald Ervin Knuth, Tracy L. Larrabee, and Paul Morris Adrian Roberts (1989). *Mathematical Writing*. Washington, D.C.: Mathematical Association of America

Daniel J. Velleman (2019). *How to Prove It*. third. Cambridge, England: Cambridge University Press

Hamilton Guide

Proof Writing Guidelines

- Use the first person plural or "we" when writing proofs. It avoids awkward passive voice, the pretension of the first person "I", or awkward construction with the third person singular "one."
- Be polite: always introduce your variables the first time they appear.
- In mathematics, we commonly write statements like "given x or "consider an arbitrary x" in the context of proving universal statements. Don't use the word "arbitrary" in other contexts.
- In mathematics, you are allowed to *assume* anything you like. Make it clear why: implication elimination, modus tollens (proof by contradiction), the induction hypothesis.
- Avoid using abbreviations in proofs, e.g., WLOG. But I like to use *iff* (if, and only if) and *QED*.

The Pumping Lemma in Words

The pumping lemma states that a “long” string can be accepted by a “small” DFA only if an infinite number of strings of a similar form are accepted as well.

An infinite regular language must accept strings of the form xy^iz for all $i \geq 0$ for some x, y, z in Σ^*. The “pumping” part is that one can “pump” the string y over and over again.

The pumping lemma is significant in that it provides a way to prove that a language is *not* regular.

Pumping Lemma in English

Pumping Lemma in English

For all languages, if the language is regular, then

Pumping Lemma in English

For all languages, if the language is regular, then there is a positive number such that

Pumping Lemma in English

For all languages, if the language is regular, then there is a positive number such that for all sufficiently long strings w in the language

Pumping Lemma in English

For all languages, if the language is regular, then there is a positive number such that for all sufficiently long strings w in the language there is a partition xyz of w with xy short and $y \neq \epsilon$ such that

Pumping Lemma in English

For all languages, if the language is regular, then there is a positive number such that for all sufficiently long strings w in the language there is a partition xyz of w with xy short and $y \neq \epsilon$ such that xy^iz is in the language for all i.

The Pumping Lemma for Regular Languages

The Pumping Lemma for Regular Languages

$$\forall L \subseteq \Sigma^* \ \Big(\mathrm{Regular}(L) \Rightarrow$$

The Pumping Lemma for Regular Languages

$$\forall\, L \subseteq \Sigma^* \;\; \Big(\mathrm{Regular}(L) \Rightarrow$$

$$\exists\, m \in \mathbb{N} \;\; \Big[m > 0 \text{ and}$$

The Pumping Lemma for Regular Languages

$$\forall\, L \subseteq \Sigma^* \; \Big(\mathrm{Regular}(L) \Rightarrow$$

$$\exists\, m \in \mathbb{N} \; \Big[m > 0 \text{ and}$$

$$\forall\, w \in \Sigma^* \; \Big([w \in L \text{ and } |w| > m] \Rightarrow$$

The Pumping Lemma for Regular Languages

$$\forall L \subseteq \Sigma^* \ \Big(\text{Regular}(L) \Rightarrow$$

$$\exists m \in \mathbb{N} \ \Big[m > 0 \text{ and}$$

$$\forall w \in \Sigma^* \ \Big([w \in L \text{ and } |w| > m] \Rightarrow$$

$$\exists x, y, z \in \Sigma^* \ \big[(w = xyz \text{ and } |xy| \leq m \text{ and } |y| \geq 1) \text{ and}$$

The Pumping Lemma for Regular Languages

$$\forall L \subseteq \Sigma^* \ \Big(\text{Regular}(L) \Rightarrow$$

$$\exists\, m \in \mathbb{N} \ \Big[m > 0 \text{ and}$$

$$\forall\, w \in \Sigma^* \ \Big([w \in L \text{ and } |w| > m] \Rightarrow$$

$$\exists\, x, y, z \in \Sigma^* \ \big[(w = xyz \text{ and } |xy| \leq m \text{ and } |y| \geq 1) \text{ and}$$

$$\forall\, i \in \mathbb{N} \ (xy^i z \in L) \big] \Big) \Big] \Big)$$

Quiz

1. $w \in L$ and $|w| > m$
2. $\exists\, x, y, z \in \Sigma^*$
3. $xy^iz \in L$
4. $\exists\, m \in \mathbb{N}$
5. $\forall\, L \subseteq \Sigma^*$
6. $w \in L \Rightarrow |w| > m$
7. $\text{Regular}(L)$
8. $m > 0$
9. $\forall\, w \in \Sigma^*$
10. $w = xyz$ and $|xy| \leq m$ and $|y| \geq 1$
11. $\forall\, i \in \mathbb{N}$
12. $xy^iz \notin L$

$\boxed{\;A\;} \ \Big(\boxed{\;B\;} \Rightarrow$

$\boxed{\;C\;} \ [\boxed{\;D\;}$ and

$\boxed{\;E\;} \ (\boxed{\;F\;} \Rightarrow$

$\boxed{\;G\;} \ [\boxed{\;H\;}$ and

$\boxed{\;I\;} \ (\boxed{\;J\;})$

$])] \Big)$

The Extended Pumping Lemma for Regular Languages

The Extended Pumping Lemma for Regular Languages

$$\forall L \subseteq \Sigma^* \ \Big(\text{Regular}(L) \Rightarrow$$

The Extended Pumping Lemma for Regular Languages

$$\forall\, L \subseteq \Sigma^* \; \Big(\mathrm{Regular}(L) \Rightarrow$$

$$\exists\, m \in \mathbb{N} \; \Big[m > 0 \text{ and}$$

The Extended Pumping Lemma for Regular Languages

$$\forall\, L \subseteq \Sigma^* \; \Big(\text{Regular}(L) \Rightarrow$$

$$\exists\, m \in \mathbb{N} \; \Big[m > 0 \text{ and}$$

$$\forall\, u, w, v \in \Sigma^* \; \Big([uwv \in L \text{ and } |w| > m] \Rightarrow$$

The Extended Pumping Lemma for Regular Languages

$$\forall\, L \subseteq \Sigma^* \;\Big(\mathrm{Regular}(L) \Rightarrow$$

$$\exists\, m \in \mathbb{N} \;\Big[m > 0 \text{ and}$$

$$\forall\, u, w, v \in \Sigma^* \;\Big([uwv \in L \text{ and } |w| > m] \Rightarrow$$

$$\exists\, x, y, z \in \Sigma^* \;\big[(w = xyz \;\text{ and }\; |xy| \leq m \;\text{ and }\; |y| \geq 1) \text{ and}$$

The Extended Pumping Lemma for Regular Languages

$$\forall\, L \subseteq \Sigma^* \;\Big(\text{Regular}(L) \Rightarrow$$

$$\exists\, m \in \mathbb{N} \;\Big[m > 0 \text{ and}$$

$$\forall\, u, w, v \in \Sigma^* \;\Big([uwv \in L \text{ and } |w| > m] \Rightarrow$$

$$\exists\, x, y, z \in \Sigma^* \;\big[(w = xyz \;\text{ and }\; |xy| \leq m \;\text{ and }\; |y| \geq 1) \text{ and}$$

$$\forall\, i \in \mathbb{N} \;(\, uxy^i zv \in L \,) \big] \Big) \Big] \Big)$$

While the pumping lemma states that all regular languages satisfy the conditions described above, the converse of this statement is not true: a language that satisfies these conditions may still be non-regular. In other words, both the original and the extended/general version of the pumping lemma give a necessary *but not sufficient* condition for a language to be regular.

Myhill-Nerode theorem provides a necessary *and sufficient* condition for a formal language to be regular. Compared to the pumping lemma, it is not as easy to prove nor to apply. We omit it here.

The pumping lemma can only be use to show that a language is *not* regular. It can never be used to show that a language *is* regular.

The pumping lemma says you can "pump" all sufficiently long strings in a regular language. What it mean mean to "pump" all sufficiently long strings?

$$\boxed{\forall\, w \in \Sigma^*} \;\Big(\; \boxed{[w \in L \text{ and } |w| > m]} \Rightarrow$$

$$\boxed{\exists\, x, y, z \in \Sigma^*} \;\Big[\; \boxed{(w = xyz \text{ and } |xy| \le m \text{ and } |y| \ge 1)} \text{ and}$$

$$\boxed{\forall\, i \in \mathbb{N}} \;\Big(\; \boxed{xy^i z \in L} \;\Big)\Big]\Big)\Big]$$

Negation

What does it mean to contradict the the previous statement? What does it mean that not all sufficiently long strings can be pumped? It means: some sufficiently long string cannot be pumped.

$$\boxed{\exists\, w \in \Sigma^*} \;\Big(\boxed{[w \in L \text{ and } |w| > m]} \text{ and }$$
$$\boxed{\forall\, x, y, z \in \Sigma^*} \;\Big[\boxed{(w = xyz \text{ and } |xy| \leq m \text{ and } |y| \geq 1)} \Rightarrow$$
$$\boxed{\exists\, i \in \mathbb{N}} \;\big(\boxed{xy^i z \notin L} \big)\Big]\Big)\Big]$$

Applying or Using the Pumping Lemma

For any language L

L regular	implies	this property holds for L
this property does *not* hold	implies	L is not regular
L not regular	implies	??
this property does hold	implies	??

Characterizing Property

We do *not* have here a property that characterizes regular languages.

Mathematically, logically, a property that is said to *characterizes* something is one that holds if, and only if. Suppose some hypothetical property characterizes red languages. For any language L

L is red	implies	the property holds for L
the property does not hold	implies	L is not red
L not red	implies	the property does not hold for L
the property does hold	implies	L is red

Proof of the Pumping Lemma

MISSING [see books]

The Pumping Lemma: $\forall L \subseteq \Sigma^* \Big(\mathrm{Regular}(L) \Rightarrow P(L) \Big)$

A Template of a Proof Using the Pumping Lemma

- We apply the Pumping Lemma to the language $L_0 \subseteq \Sigma^*$.
- We assume that the language is regular in order to obtain a contradiction.
- It follows from the assumption that all sufficiently long strings can be "pumped."
- We prove that, in fact, not all long sufficiently long strings can be "pumped."
- ...
- Hence, some sufficiently long strings cannot be "pumped."
- The assumption that the language is regular has led to a contradiction.
- Therefore, the language L_0 is not a regular language. QED

royal "we"
to avoid passive voice

$$\forall L \subseteq \Sigma^* \Big(\text{Regular}(L) \Rightarrow P(L) \Big)$$

A Template of a Proof Using the Pumping Lemma

- We apply the Pumping Lemma to the language $L_0 \subseteq \Sigma^*$.
- We assume that the language is regular in order to obtain a contradiction.
- It follows from the assumption that all sufficiently long strings can be "pumped."
- We prove that, in fact, not all long sufficiently long strings can be "pumped."
- ...
- Hence, some sufficiently long strings cannot be "pumped."
- The assumption that the language is regular has led to a contradiction.
- Therefore, the language L_0 is not a regular language. QED

“apply” means
$\forall$ elimination

ma: $\forall L \subseteq \Sigma^* \Big(\text{Regular}(L) \Rightarrow P(L) \Big)$

A Template of a Proof Using the Pumping Lemma

- We apply the Pumping Lemma to the language $L_0 \subseteq \Sigma^*$.
- We assume that the language is regular in order to obtain a contradiction.
- It follows from the assumption that all sufficiently long strings can be “pumped.”
- We prove that, in fact, not all long sufficiently long strings can be “pumped.”
- ...
- Hence, some sufficiently long strings cannot be “pumped.”
- The assumption that the language is regular has led to a contradiction.
- Therefore, the language L_0 is not a regular language. QED

The Pumping Lemma: $\forall L \subseteq \Sigma^* \Big(\text{Regular}(L) \Rightarrow$

what specific language

A Template of a Proof Using the Pumping Lemma

- We apply the Pumping Lemma to the language $L_0 \subseteq \Sigma^*$.
- We assume that the language is regular in order to obtain a contradiction.
- It follows from the assumption that all sufficiently long strings can be "pumped."
- We prove that, in fact, not all long sufficiently long strings can be "pumped."
- ...
- Hence, some sufficiently long strings cannot be "pumped."
- The assumption that the language is regular has led to a contradiction.
- Therefore, the language L_0 is not a regular language. QED

The Pumping Lemma: $\forall L \subseteq \Sigma^* \Big(\text{Regular}(L) \Rightarrow P(L) \Big)$

$\Rightarrow$ elimination

modus ponens

A Template of a Proof Using the Pumping Lemma

- We apply the Pumping Lemma to the language $L_0 \subseteq \Sigma^*$.
- We assume that the language is regular in order to obtain a contradiction.
- It follows from the assumption that all sufficiently long strings can be "pumped."
- We prove that, in fact, not all long sufficiently long strings can be "pumped."
- ...
- Hence, some sufficiently long strings cannot be "pumped."
- The assumption that the language is regular has led to a contradiction.
- Therefore, the language L_0 is not a regular language. QED

The Pumping Lemma: $\forall L \subseteq \Sigma^* \Big(\text{Regular}(L) \Rightarrow P(L) \Big)$

state what
next to be d

A Template of a Proof Using the Pumping Lemma

- We apply the Pumping Lemma to the language $L_0 \subseteq \Sigma^*$.
- We assume that the language is regular in order to obtain a contradiction.
- It follows from the assumption that all sufficiently long strings can be "pumped."
- We prove that, in fact, not all long sufficiently long strings can be "pumped."
- ...
- Hence, some sufficiently long strings cannot be "pumped."
- The assumption that the language is regular has led to a contradiction.
- Therefore, the language L_0 is not a regular language. QED

The Pumping Lemma: $\forall L \subseteq \Sigma^* \Big(\text{Regular}(L) \Rightarrow P(L) \Big)$

A Template of a Proof Using the Pumping Lemma

- We apply the Pumping Lemma to the language $L_0 \subseteq \Sigma^*$.
- We assume that the language is … contradiction.
- It follows from the assumption th… all sufficiently long strings can be "pumped."
- We prove that, in fact, not all long sufficiently long strings can be "pumped."
- …
- Hence, some sufficiently long strings cannot be "pumped."
- The assumption that the language is regular has led to a contradiction.
- Therefore, the language L_0 is not a regular language. QED

complete the proof

The Pumping Lemma: $\forall L \subseteq \Sigma^* \Big(\text{Regular}(L) \Rightarrow P(L) \Big)$

recap what was proved

A Template of a Proof Using the Pumping Lemma

- We apply the Pumping Lemma to the language $L_0 \subseteq \Sigma^*$.
- We assume that the language is regular in order to obtain a contradiction.
- It follows from the assumption that all sufficiently long strings can be "pumped."
- We prove that, in fact, not all long sufficiently long strings can be "pumped."
- ...
- Hence, some sufficiently long strings cannot be "pumped."
- The assumption that the language is regular has led to a contradiction.
- Therefore, the language L_0 is not a regular language. QED

A Template of a Proof Using the Pumping Lemma

proof by contradiction

modus tollens

- We apply the Pumping Lemma to the language $L_0 \subseteq$
- We assume that the language is regular in order to obtain a co
- It follows from the assumption that all sufficiently long strings … be "pumped."
- We prove that, in fact, not all long sufficiently long string… can be "pumped."
- . . .
- Hence, some sufficiently long strings cannot be "pumped."
- The assumption that the language is regular has led to a contradiction.
- Therefore, the language L_0 is not a regular language. QED

A Template of a Proof Using the Pumping Lemma

- We apply the Pumping Lemma to the language L_0
- We assume that the language is regular in order to obtain
- It follows from the assumption that all sufficiently long strings can be "pumped."
- We prove that, in fact, not all long sufficiently long strings can be "pumped."
- . . .
- Hence, some sufficiently long strings cannot be "pumped."
- The assumption that the language is regular has led to a contradiction.
- Therefore, the language L_0 is not a regular language. QED

the proof is complete
Quod Erat Demonstrandum

The Pumping Lemma: $\forall L \subseteq \Sigma^* \Big(\text{Regular}(L) \Rightarrow P(L) \Big)$

A Template of a Proof Using the Pumping Lemma

1. We apply the Pumping Lemma to the language $L_0 \subseteq \Sigma^*$.
2. We assume that L_0 is a regular language for purposes of obtaining a contradiction.
3. From the assumption it follows that all long strings in L_0 can be "pumped."
4. We prove that, in fact, not all long strings in L_0 can be "pumped."
5. ...
6. Hence, some long strings in L_0 cannot be "pumped."
7. The assumption that the language is regular has led to a contradiction.
8. Therefore, the language L_0 is not a regular language. QED

Proving a Language is Not Regular

- ($\forall$ intro.) Let m be an arbitrary integer such that $m > 0$.
- ($\exists$ intro.) We pick a string w_m. We show $w_m \in L$ and $\text{len}(w_m) \geq m$.
- ($\forall$ intro.) Let x, y, z be arbitrary strings such that $xyz = w_m$, $\text{len}(xy) \leq m$, and $0 < \text{len}(y)$.
- ($\exists$ intro.) We pick a number i_0. We show that $xy^{i_0}z$ is not in L_0.

Proving a Language is Not Regular

- ($\forall$ intro.) Let m be an arbitrary integer such that $m > 0$.
- ($\exists$ intro.) We pick a string w_m. We show $w_m \in L$ and $\text{len}(w_m) \geq m$.
- ($\forall$ intro.) Let x, y, z be arbitrary strings such that $xyz = w_m$, $\text{len}(xy) \leq m$, and $0 < \text{len}(y)$.
- ($\exists$ intro.) We pick a number i_0. We show that $xy^{i_0}z$ is not in L_0.

A key to understanding how one meets one's proof obligations is to think of arbitrary values (for-all introduction) as having been designed by a malevolent opponent to make it as difficult as possible to complete the proof.

Proving a Language is Not Regular

- ($\forall$ intro.) Let m be an arbitrary integer such that $m > 0$.
- ($\exists$ intro.) We pick a string w_m. We show $w_m \in L$ and $\text{len}(w_m) \geq m$.
- ($\forall$ intro.) Let x, y, z be arbitrary strings such that $xyz = w_m$, $\text{len}(xy) \leq m$, and $0 < \text{len}(y)$.
- ($\exists$ intro.) We pick a number i_0. We show that $xy^{i_0}z$ is not in L_0.

A key to understanding how one meets one's proof obligations is to think of arbitrary values (for-all introduction) as having been designed by a malevolent opponent to make it as difficult as possible to complete the proof.

It may be necessary to break into cases to cover all the arbitrary choices.

In any application of the pumping lemma there is a complicated shell or boilerplate which is always the same.

Though we often tire of the boilerplate, students are not allowed to omit or modify the boilerplate in homework and exams.

In any application of the pumping lemma there is a complicated shell or boilerplate which is always the same.

Though we often tire of the boilerplate, students are not allowed to omit or modify the boilerplate in homework and exams.

An example proof using the pumping lemma.

Theorem. The language $L_0 = \{ a^n b^n \mid 0 \leq n \}$ is not regular.

Proof. **We apply the pumping lemma to the language $L_0 \subseteq \Sigma^*$. We assume that L_0 is a regular language for purposes of obtaining a contradiction. From this assumption it follows that all sufficiently long strings in L_0 can be "pumped." We will prove that, in fact, some sufficiently long strings in L_0 cannot be "pumped."**

Let m be an arbitrary integer such that $m > 0$. We pick the string $w_m = a^m b^m$. We have $w_m = a^m b^m \in L_0$ because $0 \leq m$, and $\text{len}(a^m b^m) = 2m > m$.

Let x, y, z **be arbitrary strings such that** $w_m = xyz$**,** $\text{len}(xy) \leq m$**, and** $0 < \text{len}(y)$**. We pick the integer** $i_0 = 0$ **and we will prove** $xy^{i_0}z \notin L_0$.

Let the length of x be called h and the length of y be called j. We can write the string w_m this way: $w_m = a^h a^j a^{m-(h+j)} b^m$.

$$w_m = \overbrace{\underbrace{a \cdots a}_{h}\ \underbrace{a \cdots a}_{j}\ a \cdots a}^{m}\ \overbrace{b \cdots b}^{m} \qquad \in L_0$$

We know $h + j \leq m$ and $0 < j$. So xy^0z is equal to $a^h a^{m-(h+j)} b^m$.

$$xy^{i_0}z = \overbrace{\underbrace{a \cdots a}_{h}\ a \cdots a}^{m-j}\ \overbrace{b \cdots b}^{m} \qquad \notin L_0$$

The number of initial a's is $h + m - (h + j) = m - j < m$. But $m - j$ is not equal to m. So, $xy^{i_0}z \notin L_0$. **Hence, some long strings cannot be "pumped" in** L_0**. The assumption has led to a contradiction.**

Therefore, the language L_0 **is not a regular language. QED**

Theorem [Linz 6th, Section 4.3, Example 4.8]. The language $L_0 = \{ ww^R \mid w \in \Sigma^* \}$ is not regular.

Proof. **We apply the pumping lemma to the language $L_0 \subseteq \Sigma^*$. We assume that L_0 is a regular language for purposes of obtaining a contradiction. From this assumption it follows that all sufficiently long strings in L_0 can be "pumped." We will prove that, in fact, some sufficiently long strings in L_0 cannot be "pumped."**

Let m be an arbitrary integer such that $m > 0$. We pick the string $w_m = a^m b^m b^m a^m$. We have $w_m = a^m b^m b^m a^m \in L_0$ because it is the same forwards and backwards, and $\text{len}(w_m) = 4m > m$.

Let x, y, z **be arbitrary strings such that** $w_m = xyz$**,** $\text{len}(xy) \leq m$**, and** $0 < \text{len}(y)$**. We pick the integer** $i_0 = 0$ **and we will prove** $xy^{i_0}z \notin L_0$.
But it is obvious that one or more fewer a's in the first m characters will result in a string which in not a palindrome. (The first b from the left does not match the corresponding character from the right as the last m characters are a's.)

Hence, some sufficiently long strings cannot all be "pumped" in L_0**. The assumption has led to a contradiction.**

Therefore, the language L_0 **is not a regular language. QED**

Theorem [Linz 6th, Section 4.3, Example 4.12]. The language $L_0 = \{ a^n b^l c^{n+k} \mid 0 \leq n, k \}$ is not regular.

Theorem [Linz 6th, Section 4.3, Exercise 5b]. The language $L_0 = \{\, a^n b^l a^k \mid n + l \geq k \,\}$ is not regular.

Proof. **We apply the pumping lemma to the language $L_0 \subseteq \Sigma^*$. We assume that L_0 is a regular language for purposes of obtaining a contradiction. From this assumption it follows that all long strings can be "pumped" in L_0. We will prove that, in fact, some long strings cannot be "pumped" in L_0.**

Let m be an arbitrary integer such that $m > 0$. We pick the string $w_m = a^m b a^{m+1}$. We have $w_m \in L_0$ because $m + 1 \geq m + 1$, and $\text{len}(a^m b a^{m+1}) = 2m + 2 > m$.

Let x, y, z **be arbitrary strings such that** $w_m = xyz$**,** $\mathrm{len}(xy) \leq m$**, and** $0 < \mathrm{len}(y)$**. We pick the integer** $i_0 = 0$ **and we will prove** $xy^{i_0}z \notin L_0$.

Let the length of x be called h and the length of y be called j. We can write the string w_m this way: $w_m = a^h a^j a^{m-(h+j)} b a^{m+1}$.

$$w_m = \overbrace{\underbrace{a \cdots a}_{h}\ \underbrace{a \cdots a}_{j}\ a \cdots a}^{m}\ b\ \overbrace{a \cdots a}^{m+1} \qquad \in L_0$$

We know $h + j \leq m$ and $0 < j$. So xy^0z is equal to $a^h a^{m-(h+j)} b a^{m+1}$.

$$xy^{i_0}z = \overbrace{\underbrace{a \cdots a}_{h}\ a \cdots a}^{m-j}\ b\ \overbrace{a \cdots a}^{m+1} \qquad \notin L_0$$

The number of initial a's is $h + m - (h + j) = m - j < m$. But $m - j + 1$ is not greater than or equal to $m + 1$. So, $xy^{i_0}z \notin L_0$. **Hence, long strings cannot be "pumped" in** L_0**. The assumption has led to a contradiction.**

Therefore, the language L_0 **is not a regular language. QED**