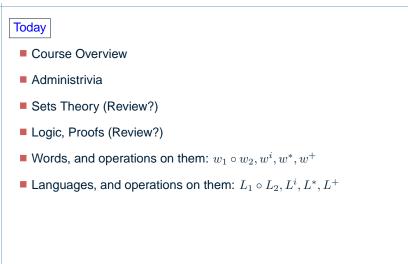


What This Course Is About

Mathematical theory of computation!

- We'll study different "machine models" (finite automata, pushdown automata)...
- ... with a view toward characterizing what they can compute.

Sets, Languages, Logic



Portions © 2000 Rance Cleaveland © 2004 James Riely

Automata Theory and Formal Grammars: Lecture 1 - p.2/74

Why Study This Topic?

To understand the limits of computation.

Some things require more resources to compute, and others cannot be computed at all. To study these issues we need mathematical notions of "resource" and "compute".

To learn some programming tools.

Automata show up in many different settings: compilers, text editors, communications protocols, hardware design, ...

First compilers took several person-years; now written by a single student in one semester, thanks to theory of parsing.

To learn about program analysis.

Microsoft is shipping two model-checking tools. PREfix discovered \geq 2000 bugs in XP (fixed in SP2).

To learn to think analytically about computing.

Why Study This Topic?

- This course focuses on machines and logics. Analysis technique: model checking (SE431).
- CSC535 focuses on languages and types.
 Analysis technique: type checking (CSC535).
- Both approaches are very useful.
 For example, in Computer Security (SE547).

Administrivia

 Course Homepage: http://www.depaul.edu/~jriely/csc444fall2003/
 Syllabus: http://www.depaul.edu/~jriely/csc444fall2003/syllabus.html

Portions © 2000 Rance Cleaveland © 2004 James Riely

Automata Theory and Formal Grammars: Lecture 1 – p.5/74

Set Theory: Sets, Functions, Relations

Portions © 2000 Rance Cleaveland © 2004 James Riely

Automata Theory and Formal Grammars: Lecture 1 - p.6/74

Sets

Sets are collections of objects.

- $\blacksquare \ \{ \ \}, \ \{42\}, \ \{\text{alice}, \ \text{bob} \}$
- $\blacksquare \mathbb{N} = \{0, 1, 2, \ldots\}$
- $\blacksquare \mathbb{Z} = \{..., -2, -1, 0, 1, 2, ...\}$
- **E** R = the set of real numbers includings \mathbb{Z} , $\sqrt{2}$, π , etc
- $\blacksquare \{ x \in \mathbb{N} \mid x \ge 5 \}$

Sets are unordered and insensitive to repetition.

- $\blacksquare \{42,27\} = \{27,42\}$
- $\blacksquare \{42, 42\} = \{42\}$

What Do the Following Mean?

$\emptyset, \{\}$	empty set
$a \in A$	membership
$A\subseteq B$	subset
$A\cup B$	union
$A\cap B$	intersection
$\circ A$	complement
A - B	set difference $= A \cap \circ B$
$\bigcup_{i\in I} A_i$	indexed union
$\bigcap_{i\in I} A_i$	indexed intersection
2^A	power set (set of all subsets)
$A \times B$	Cartesian product = { $\langle a, b \rangle \mid a \in A, b \in B$ }
A	size (cardinality, or number of elements)

Examples

Let $A = \{m, n\}$ and $B = \{x, y, z\}$
What is <i>A</i> ? <i>B</i> ?
2, 3
• What is $A \times B$? $ A \times B $?
$\{\langle {\sf m}, {\sf x} angle, \langle {\sf m}, {\sf y} angle, \langle {\sf m}, {\sf z} angle, \langle {\sf n}, {\sf x} angle, \langle {\sf n}, {\sf y} angle, \langle {\sf n}, {\sf z} angle \}$, $2 imes 3 = 6$
• What is 2^{A} ? $ 2^{A} $?
$\{\emptyset,\{m\},\{n\},\{m,n\}\}$, $2^2=4$
• What is 2^{B} ? $ 2^{B} $?
$\{\emptyset,\{{\sf x}\},\{{\sf y}\},\{{\sf z}\},\{{\sf x},{\sf y}\},\{{\sf x},{\sf z}\},\{{\sf y},{\sf z}\},\{{\sf x},{\sf y},{\sf z}\}\}$, $2^3=8$

Portions © 2000 Rance Cleaveland © 2004 James Riely

Automata Theory and Formal Grammars: Lecture 1 - p.9/74

Equality on Sets

Let A and B be sets. When does A = B? When they contain the same elements. When $A \subseteq B$ and $B \subseteq A$. Some Set Equalities $A \cup \emptyset = A$

 $\begin{array}{rcl} A \cap \emptyset & = & \emptyset \\ \circ A \cup B & = & \circ A \cap \circ B & (\mbox{De Morgan}) \\ A \cup (B \cap C) & = & (A \cup B) \cap (A \cup C) & (\mbox{Distributivity}) \end{array}$

Cardinality

Portions © 2000 Rance Cleaveland © 2004 James Riely

Cardinality is easy with finite sets.

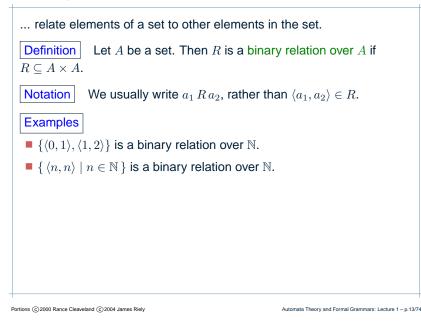
 $|\{1,2,3\}| = |\{a,b,c\}|$

What about infinite ones?

To answer this we need to understand functions.

Automata Theory and Formal Grammars: Lecture 1 - p.10/74

Binary Relations



Equivalence Classes

Let *R* be an equivalence relation $R \subseteq A \times A$. Let $a \in A$. Then we write $[a]_R$ for the set of elements equivalent to *a* under *R*.

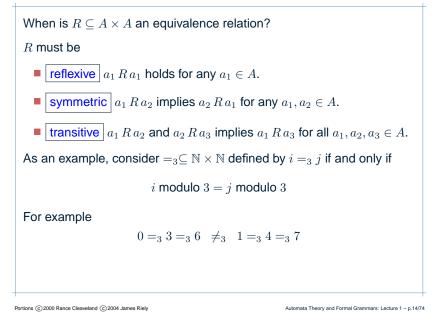
 $[a]_R = \{ a' \mid a R a' \}$

Note that $[a]_R \subseteq A$.

What is $[1]_{=_3}$?

 $\{1,4,7,10,\ldots\}$

Equivalence Relations



Functions

- When is $R \subseteq A \times B$ a function (ie, a total function)?
- R must be
 - deterministic If $a R b_1$ and $a R b_2$ then $b_1 = b_2$.
 - **total** For every $a \in A$, there exists $b \in B$ such that a R b holds.
- Equivalently... For every $a \in A$, require $|\{b \mid a R b\}| = 1$. If we require only determinism, we define partial functions.
- Functions map elements from one set to elements from another.

 $f:A \to B$

- $\blacksquare A: \text{domain of } f$
- $\blacksquare B: \text{codomain of } f$
- f(a): result of applying f to $a \in A f(a) \in B$.

Relational Inverse

Definition $b R^{-1} a$ if and only if a R b.

Is the inverse of a function always a function?

Portions © 2000 Rance Cleaveland © 2004 James Riely

Automata Theory and Formal Grammars: Lecture 1 - p.17/74

Which $f : \mathbb{N} \to \mathbb{N}$ is injective/Surjective?

$f(x) = x + 1$ $f(x) = \lfloor \frac{x}{2} \rfloor$ $f(x) = x $	injective, not surjective surjective, not injective bijective
	What if instead $f : \mathbb{Z} \to \mathbb{Z}$?
$f(x) = x + 1$ $f(x) = \lfloor \frac{x}{2} \rfloor$ $f(x) = x $	bijective surjective, not injective neither injective nor surjective

Bijections

When Is $f : A \rightarrow B \dots$ injective (or one-to-one)? When $f(a_1) = f(a_2)$ implies $a_1 = a_2$ for any $a_1, a_2 \in A$. When f^{-1} is deterministic ... surjective (onto)? When for any $b \in B$ there is an $a \in A$ with f(a) = b. When f^{-1} is total ... bijective? When it is injective and surjective. When f^{-1} is a function

Portions © 2000 Rance Cleaveland © 2004 James Riely

b

Automata Theory and Formal Grammars: Lecture 1 - p.18/74

More On Functions

Let $f: A \to B$
If $S \subseteq A$ then how is $f(S)$ defined?
$f(S) = \{ f(a) \mid a \in S \}.$
We have lifted f from $A \to B$ to $2^A \to 2^B$.
• What is $f(A)$ called?
The range of f.
If $g: B \to C$ then how is $g \circ f$ defined?
$g \circ f : A \to C$ is defined as $g \circ f(a) = g(f(a))$.
If f is a bijection, what is $(f^{-1})^{-1}$?
f
If f is a bijection, what is $f \circ f^{-1}(b)$?

Cardinality Revisited

Definition Two infinite sets have the same cardinality if there exists a bijection between them.

Recall the naturals (\mathbb{N}) , integers (\mathbb{Z}) and reals (\mathbb{R}) .

Theorem

- $\blacksquare |\mathbb{N}| = |\mathbb{Z}|$
- $\blacksquare |\mathbb{N}| = |\mathbb{N} \times \mathbb{N}|$
- $\blacksquare |\mathbb{N}| \neq |2^{\mathbb{N}}|$
- $|2^{\mathbb{N}}| = |\mathbb{R}|$

How would you prove these statements?

Portions © 2000 Rance Cleaveland © 2004 James Riely

Automata Theory and Formal Grammars: Lecture 1 - p.21/74

Languages and Computation

What are computers? Symbol pushers

- They take in sequences of symbols ...
- ... and produce sequences of symbols.

Mathematically, languages are sets of sequences of symbols ("words") taken from some alphabet.

Computers are language processors.

We'll study different classes of languages with a view toward characterizing how much computing power is needed to "process" them.

But first, we need precise definitions of alphabet, word and language.



Portions © 2000 Rance Cleaveland © 2004 James Riely

Automata Theory and Formal Grammars: Lecture 1 - p.22/74

Alphabets

An alphabet is a finite, nonempty set of symbols.
Examples
<pre>4 a, b,, z }</pre>
■ { a, b,, z, ä, ö, ü, ß }
• { 0,1 }
ASCII
Alphabets are usually denoted by Σ .

Words

Words

A word (or string) or over an alphabet is a finite sequence of symbols from the alphabet.

Examples

sour

- süß
- **010101110**

We write the empty string as ε .

Let Σ^* be the set of all words over alphabet Σ .

Portions © 2000 Rance Cleaveland © 2004 James Riely

Automata Theory and Formal Grammars: Lecture 1 - p.25/74

Operations on Words: Length

Definition Let Σ be an alphabet. The length function $|-|: \Sigma^* \to \mathbb{N}$ is defined inductively as follows.

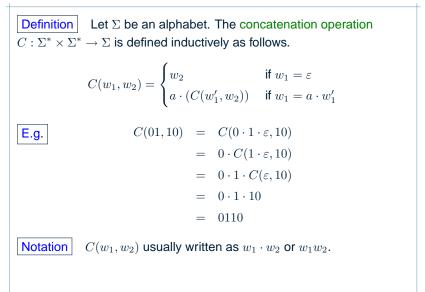
$$|w| = \begin{cases} 0 & \text{if } w = \varepsilon \\ 1 + |w'| & \text{if } w = a \cdot w' \end{cases}$$
$$|abb| = |a \cdot b \cdot b \cdot \varepsilon|$$
$$= 1 + |b \cdot b \cdot \varepsilon|$$
$$= 1 + 1 + |b \cdot \varepsilon|$$
$$= 1 + 1 + 1 + |\varepsilon|$$
$$= 1 + 1 + 1 + 0$$
$$= 3$$

1

A 16

	Words as Lists	
	One can think about strings as a ε -terminated list of	symbols.
	Examples	
	sour = $\mathbf{s} \cdot \mathbf{o} \cdot \mathbf{u} \cdot \mathbf{r} \cdot \boldsymbol{\varepsilon}$	
	s $\ddot{\mathbf{s}}$ s $\ddot{\mathbf{s}}$ s $\ddot{\mathbf{s}} \cdot \ddot{\mathbf{s}} \cdot \varepsilon$	
	$\bullet 010101110 = 0 \cdot 1 \cdot 0 \cdot 1 \cdot 0 \cdot 1 \cdot 1 \cdot 1 \cdot 0 \cdot \varepsilon$	
_		
I	Portions © 2000 Rance Cleaveland © 2004 James Riely Automata	Theory and Formal Grammars: Lecture 1 – p.26/7

Operations on Words: Concatenation



E.g.

Substrings

Using concatenation, we can define substrings.

- v is a substring of a string w if there are strings x and y s.t.
 w = xvy
- if w = uv for some string u then v is a suffix of w
- if w = uv for some string v then u is a prefix of w

Degenerate cases:

- \bullet is a substring of any string
- Any string is a substring of itself
- \mathbf{I} ε is a prefix and suffix of any string
- Any string is a prefix and suffix of itself

Portions © 2000 Rance Cleaveland © 2004 James Riely

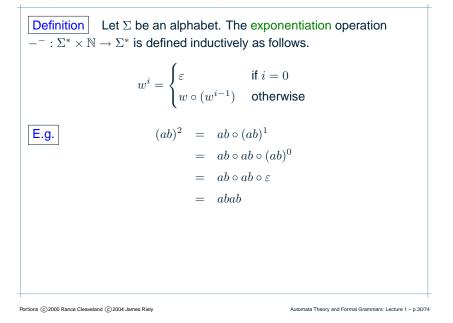
Automata Theory and Formal Grammars: Lecture 1 - p.29/74

Operations on Words: Reverse

	be an alphabet. The reverse operation fined inductively as follows.
	$w^{\mathcal{R}} = \begin{cases} \varepsilon & \text{if } w = \varepsilon \\ C(w^{\mathcal{R}}, a) & \text{if } w = a \cdot u \end{cases}$
E.g.	$abc^{\mathcal{R}} = (a \cdot b \cdot c \cdot \varepsilon)^{\mathcal{R}}$ $= C((b \cdot c \cdot \varepsilon)^{\mathcal{R}}, a)$
	$= C(C((c \cdot \varepsilon)^{\mathcal{R}}, b), a)$
	$= C(C(C((\varepsilon)^{\mathcal{R}}, c), b), a)$
	$= C(C(C(\varepsilon,c),b),a)$
	= C(C(c,b),a)
	= C(cb, a)
	= cba

Automata Theory and Formal Grammars: Lecture 1 - p.31/74

Operations on Words: Exponentiation



Properties of Operators on Words

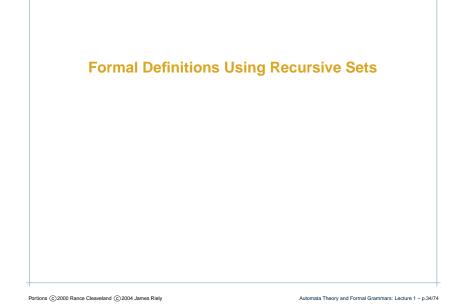
$$\varepsilon \circ w = w$$
$$w \circ \varepsilon = w$$
$$w_1 \circ (w_2 \circ w_3) = (w_1 \circ w_2) \circ w_3$$
$$|w_1 \circ w_2| = |w_1| + |w_2|$$
$$w^1 = w$$
$$w^{i+j} = w^i \circ w^j$$
$$(w^{\mathcal{R}})^{\mathcal{R}} = w$$

Portions © 2000 Rance Cleaveland © 2004 James Riely

Automata Theory and Formal Grammars: Lecture 1 - p.32/74

Conventions

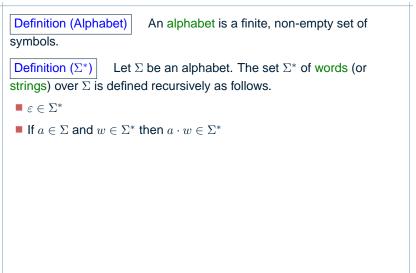
- Σ is an arbitrary alphabet. (In examples, Σ should be clear from context.)
- The variables a-e range over letters in Σ .
- The variables u-z range over words over Σ^* .



Portions © 2000 Rance Cleaveland © 2004 James Riely

Automata Theory and Formal Grammars: Lecture 1 - p.33/74

More Formally: Alphabets and Words

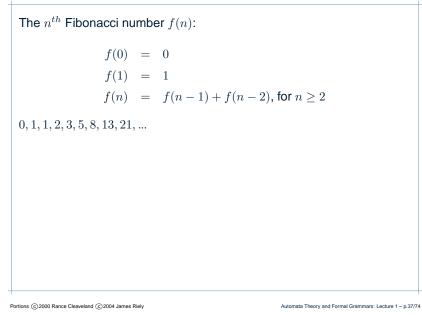


What?

- ε is a special symbol representing the empty string (i.e. a string with no symbols). You can also think of it as the "end-of-word" marker.
- $a \cdot w$ represents a word consisting of the letter a followed by the word w.

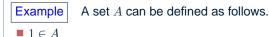
Examples• $\varepsilon \in \{0,1\}^*$ • $0 \cdot \varepsilon \in \{0,1\}^*$ • $0 \cdot 1 \cdot 1 \cdot 0 \cdot \varepsilon \in \{0,1\}^*$ NotationInstances of \cdot , trailing ε 's are usually omitted:0, 0110written rather than $0 \cdot \varepsilon, 0 \cdot 1 \cdot 1 \cdot 0 \cdot \varepsilon.$

Recall Fibonacci



Sets Can Also Be Defined Recursively

Recursive set definitions consist of rules explaining how to build up elements in the set from elements already in the set.



If $a \in A$ then $a + 3 \in A$

What are elements in A?

```
A=\{1,4,7,\ldots\}=[1]_{=_3}
```

Recursive Definitions for Functions

Recursion A method of defining something "in terms of itself".

Fibonacci is defined in terms of itself.

Why is this OK?

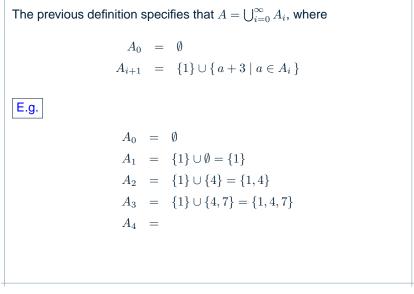
Because:

- There are "base cases" (n = 0, 1).
- Applications of *f* in body are to "smaller" arguments.

Portions © 2000 Rance Cleaveland © 2004 James Riely

Automata Theory and Formal Grammars: Lecture 1 – p.38/74

Elements of Recursively Defined Sets



More Generally

Recursive set definitions consist of rules of following forms: $c \in A$ for some constant cIf $a \in A$ and p(a) then $f(a) \in A$ for some predicate p and function fThen $A = \bigcup_{i=0}^{\infty} A_i$, where $A_0 = \emptyset$ $A_{i+1} = \{c \mid c \in A \text{ is a rule} \} \cup$ $\{f(a) \mid \text{If } a \in A \text{ and } p(a) \text{ then } f(a) \in A \text{ is a rule}$ $\land a \in A_i \land p(a) \}$ E.g. In previous example: p(a) is "true"f(a) = a+3

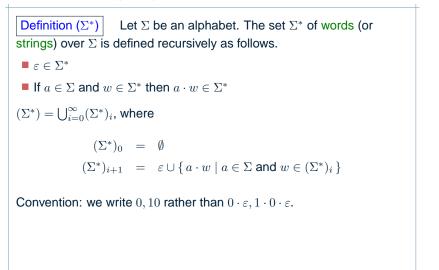
Portions © 2000 Rance Cleaveland © 2004 James Riely

Automata Theory and Formal Grammars: Lecture 1 – p.41/74

An example

$$\begin{split} (\Sigma^*) &= \bigcup_{i=0}^{\infty} (\Sigma^*)_i, \, \text{where} \\ &(\Sigma^*)_0 &= \emptyset \\ &(\Sigma^*)_{i+1} &= \varepsilon \cup \{ a \cdot w \mid a \in \Sigma \text{ and } w \in (\Sigma^*)_i \, \} \end{split}$$
For example, let $\Sigma = \{0, 1\}$ $&(\Sigma^*)_0 &= \emptyset \\ &(\Sigma^*)_1 &= \{\varepsilon\} \cup \emptyset = \{\varepsilon\} \\ &(\Sigma^*)_2 &= \{\varepsilon\} \cup \{0, 1\} = \{\varepsilon, 0, 1\} \\ &(\Sigma^*)_3 &= \{\varepsilon\} \cup \{0, 1, 00, 01, 10, 11\} = \{\varepsilon, 0, 1, 00, 01, 10, 11\} \\ &(\Sigma^*)_4 &= \\ &\{0, 1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, \ldots\} \end{split}$

More Formally: Alphabets and Words



Portions © 2000 Rance Cleaveland © 2004 James Riely

Automata Theory and Formal Grammars: Lecture 1 – p.42/74

Generally...

$\Sigma^* = \bigcup_{i=0}^{\infty} (\Sigma$	*) _i ,	where
$(\Sigma^*)_0$	=	Ø
$(\Sigma^*)_1$	=	$\{\varepsilon\}$
$(\Sigma^*)_2$	=	$\{\varepsilon\}\cup\{a\cdot\varepsilon\mid a\in\Sigma\}$
	=	$\{\varepsilon\} \cup \{ a \mid a \in \Sigma \}$
$(\Sigma^*)_3$	=	$\{\varepsilon\} \cup \{a \cdot w' \mid a \in \Sigma \land w' \in (\Sigma^*)_2\}$
	=	$\{\varepsilon\} \cup \{ a \cdot \varepsilon \mid a \in \Sigma \} \cup \{ a_1 \cdot a_2 \cdot \varepsilon \mid a_1, a_2 \in \Sigma \}$
	=	$\{\varepsilon\} \cup \{a \mid a \in \Sigma\} \cup \{a_1a_2 \mid a_1, a_2 \in \Sigma\}$
	:	

Note. $(\Sigma^*)_i$ consists of all words containing up to i-1 symbols from Σ .



Languages

... are just sets of words, i.e. subsets of Σ^* !

Definition Let Σ be an alphabet. Then a language over Σ is a subset of Σ^* .

What is 2^{Σ^*} ? Question

Portions © 2000 Rance Cleaveland © 2004 James Riely

The set of all languages over Σ !

Portions © 2000 Rance Cleaveland © 2004 James Riely

Automata Theory and Formal Grammars: Lecture 1 - p.45/74

Examples of Language Operations

$\{ab, aa\} \circ \{bb, a\} = \{ab \circ bb, ab \circ a, aa \circ bb, aa \circ a\}$ $= \{abbb, aba, aabb, aaa\}$ $\{01,1\}^2 = \{01,1\} \circ \{01,1\}^1$ $= \{01,1\} \circ \{01,1\} \circ \{01,1\}^0$ $= \{01, 1\} \circ \{01, 1\} \circ \{\varepsilon\}$ $= \{0101, 011, 101, 11\}$

Operations on Languages

The usual set operations may be applied to languages: \cup , \cap , etc. One can also "lift" operations on words to languages.

Definition Let Σ be an alphabet, and let $L, L_1, L_2 \subseteq \Sigma^*$ be languages.

Concatenation: $L_1 \circ L_2 = \{ w_1 \circ w_2 \mid w_1 \in L_1 \land w_2 \in L_2 \}.$

Exponentiation: Let $i \in \mathbb{N}$. Then L^i is defined recursively as follows.

$$L^{i} = \begin{cases} \{\varepsilon\} & \text{if } i = 0 \\ L \circ L^{i-1} & \text{otherwise} \end{cases}$$

Automata Theory and Formal Grammars: Lecture 1 - p.46/74

Operations on Languages: Kleene Closure

Kleene closure (pronounced "clean-y") is another important operation on languages.

Definition Let Σ be an alphabet, and let $L \subseteq \Sigma^*$ be a language. Then the Kleene closure, L^* , of L is defined recursively as follows.

1.
$$\varepsilon \in L^*$$
.

2. If $w \in L$ and $w' \in L^*$ then $w \circ w' \in L^*$

E.g.
$$\{01\}^* = \{\varepsilon, 01, 0101, 010101, ...\}$$

What is \emptyset^* ?

Portions © 2000 Rance Cleaveland © 2004 James Riely

Automata Theory and Formal Grammars: Lecture 1 – p.49/74

A Variation on L^*

Definition	Let $L \subseteq \Sigma^*$. Then L^+ is defined inductively as follows.
$\blacksquare L \subseteq L^+.$	

If $v \in L$ and $w \in L^+$ then $v \circ w \in L^+$.

Difference between L^*, L^+ : ε is not guaranteed to be an element of $L^+!$

What is *L*^{*} Mathematically?

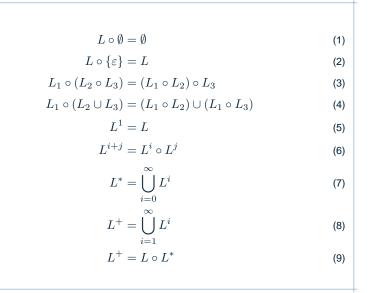
Since L^* is defined recursively, we know that $L^* = \bigcup_{i=0}^{\infty} (L^*)_i$, where:
$(L^*)_0 = \emptyset$
$(L^*)_{i+1} = \{\varepsilon\} \cup \{u \circ v \mid u \in L \text{ and } v \in (L^*)_i\}$
$(L^*)_1 = \{\varepsilon\}$
$(L^*)_2 = \{\varepsilon\} \cup \{w \circ \varepsilon \mid w \in L\}$
$= \{\varepsilon\} \cup L$
$(L^*)_3 = \{\varepsilon\} \cup \{w \circ w' \mid w \in L \land w' \in (L^*)_2\}$
$= \{\varepsilon\} \cup L \cup (L \circ L)$
(I^*) consists of words obtained by gluing together up to $i = 1$ conject

 $(L^*)_i$ consists of words obtained by gluing together up to i-1 copies of words from L.

Portions © 2000 Rance Cleaveland © 2004 James Riely

Automata Theory and Formal Grammars: Lecture 1 - p.50/74

Properties of $L_1 \circ L_2$, L^i , L^* , L^+



Logic	

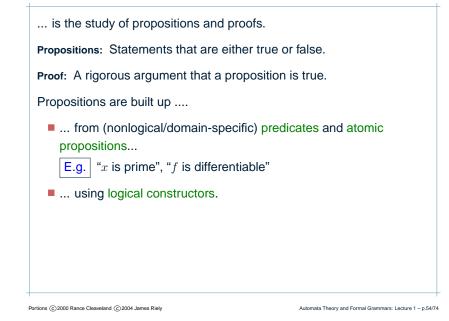
What Do the Following Logical Constructors Mean?

- ∧ conjunction ("and")
- ∨ disjunction ("or")
- negation ("not")
- \longrightarrow implication ("if ... then", "implies")
- \longleftrightarrow bi-implication ("if and only if")
- ∀ universal quantifier ("for all")
- ∃ existential quantifier ("there exists")

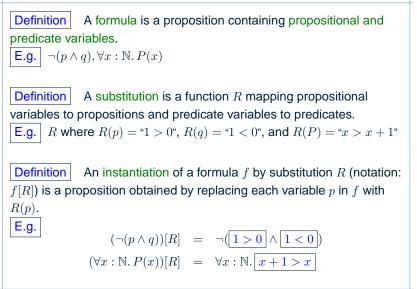
Examples (Propositions)

- 1. $\forall f : \mathbb{R} \to \mathbb{R}$. "*f* is differentiable" \longrightarrow "*f* is continuous"
- $\textbf{2. } \neg \exists x \in \mathbb{N}. \text{``}x \text{ is prime''} \land (\forall y \in \mathbb{N}. \, y \geq x \longrightarrow \neg \text{``}y \text{ is prime''}).$

Logic ...



Formulas and Instantiations



Logical Implications, Logical Equivalences, and Tautologies

Definition Let f, g be formulas.

- I logically implies g (notation: f ⇒ g) if for every substitution R such that f[R] is true, g[R] is also true.
- f and g are logically equivalent (notation: $f \iff g$) if for every substitution R, f[R] and g[R] are either both true or both false.
- *f* is a tautology if for every substitution *R*, *f*[*R*] is true (equivalently, $f \equiv \text{true}$).

Intuitively, $f \implies g$ and $f \equiv g$ reflect truths that hold independently of any domain-specific information.

Portions © 2000 Rance Cleaveland © 2004 James Riely

Propositions, Natural Language, and Mathematics

In this course we will devote a lot of attention to proofs of assertions about different models of computation.

These statements are usually given in English, e.g.

A language L is regular if and only if it can be recognized by some FA M.

In order to prove statements like these it is extremely useful to know the "logical structure" of the statement: that is, to "convert" it into a proposition!

 $\forall L$. "*L* is a language" \longrightarrow ("*L* is regular" $\longleftrightarrow \exists M$. "*M* is a FA" \land "*M* recognizes *L*")

Examples of Implications, Equivalences and Tautologies

$p \wedge q$	\implies	$p \vee q$	Disjunctive weakening (I)
p	\implies	$p \vee q$	Disjunctive weakening (II)
$\neg \neg p$	≡	p	Double negation
$p \longrightarrow q$	≡	$(\neg p) \lor q$	Material implication
$\neg (p \lor q)$	≡	$(\neg p) \land (\neg q)$	DeMorgan's Laws
$p \longrightarrow q$	≡	$(\neg q) \longrightarrow (\neg p)$	Contrapositive
$\neg \forall x. P(x)$	\equiv	$\exists x. \neg P(x)$	DeMorgan's Laws
$p \vee (\neg p)$	≡	true	Law of Excluded Middle

Portions © 2000 Rance Cleaveland © 2004 James Riely

Automata Theory and Formal Grammars: Lecture 1 - p.58/74

Translating Natural Langauge into Logic

Phrase	Logical construct
" not"	-
" and"	^
" or"	\vee
"if then, " implies",	\longrightarrow
" only if"	
" if and only if",	\longleftrightarrow
" is logically equivalent to"	
" all", " any"	\forall
" exists", " some"	Э

E.g.

Automata Theory and Formal Grammars: Lecture 1 - p.57/74

Proofs

Proofs are rigorous arguments for the truth of propositions. They come in one of two forms.

Direct proofs: Use "templates" or "recipes" based on the logical form of the proposition.

Indirect proofs: Involve the direct proof of a proposition that logically implies the one we are interested in.

Direct Proofs

Logical Form	Proof recipe
$\neg p$	Assume p and then derive a contradiction.
$p \wedge q$	Prove p ; then prove q .
$p \lor q$	Prove either p or q .
$p \longrightarrow q$	Assume p and then prove q .
$p \longleftrightarrow q$	Prove $p \longrightarrow q$; then prove $q \longrightarrow p$.
$\forall x. P(x)$	Fix a generic x and then prove $P(x)$.
$\exists x. P(x)$	Present a specific value a and prove $P(a)$.

Portions © 2000 Rance Cleaveland © 2004 James Riely

Automata Theory and Formal Grammars: Lecture 1 - p.62/74

Portions © 2000 Rance Cleaveland © 2004 James Riely

Automata Theory and Formal Grammars: Lecture 1 - p.61/74

Sample Direct Proof

A theorem is a statement to be proved.

Theorem A language L is regular if and only if it is recognized by some FA M.

Logical Form

 $\forall L$. "*L* is a language" \longrightarrow ("*L* is regular" $\longleftrightarrow \exists M$. "*M* is a FA" \land "*M* recognizes *L*")

Proof Fix a generic L (\forall) and assume that L is a language (\longrightarrow); we must prove that L is regular if and only if it is recognized by some FA M. So assume that L is regular; we must now show that some FA M exists such that M recognizes L (first part of \longleftrightarrow).... Now assume that some FA M exists such that M recognizes L; we must show that L is regular (second part of \longleftrightarrow)....

This is not a complete proof; we need to know the definitions to continue. But notice that the logical form gets us started!

Indirect Proofs

... rely on proof of a statement that logically implies the one we are interested in.

Examples

To prove	It suffices to prove	Terminology
p	$\neg \neg p$	"Proof by contradiction"
$p \longrightarrow q$	$(\neg q) \longrightarrow (\neg p)$	"Proof by contrapositive"

Mathematical Induction...

... is an indirect proof technique for statements having logical form

 $\forall k \in \mathbb{N}. P(k).$

Induction proofs have two parts.

Base case: Prove P(0).

Induction step: Prove $\forall k \in \mathbb{N}$. $(P(k) \longrightarrow P(k+1))$. The P(k) is often called the induction hypothesis.

Note that an induction proof is actually a proof of the following:

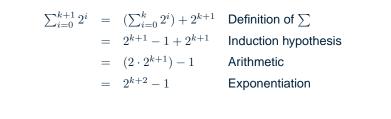
$$P(0) \land (\forall k \in \mathbb{N}. P(k) \longrightarrow P(k+1))$$

Why does this logically imply $\forall k \in \mathbb{N}$. P(k)?

Portions © 2000 Rance Cleaveland © 2004 James Riely

Automata Theory and Formal Grammars: Lecture 1 – p.65/74

Proof (cont.)



Sample Induction Proof

Theorem For any natural number k , $\sum_{i=0}^{k} 2^{i} = 2^{k+1} - 1$
Logical Form $\forall k \in \mathbb{N}$. $P(k)$, where $P(k)$ is $\sum_{i=0}^{k} 2^{i} = 2^{k+1} - 1$
Proof The proof proceeds by induction.
Base case: We must prove $P(0)$, i.e. $\sum_{i=0}^{0} 2^{i} = 2^{1} - 1$. But $\sum_{i=0}^{0} 2^{i} = 2^{0} = 1 = 2 - 1 = 2^{1} - 1$.
Induction step: We must prove $\forall k \in \mathbb{N}$. $P(k) \longrightarrow P(k+1)$. So fix a generic $k \in \mathbb{N}$ and assume (induction hypothesis) that $P(k)$ holds, i.e. that $\sum_{i=0}^{k} 2^i = 2^{k+1} - 1$ is true. We must prove $P(k+1)$, i.e. that $\sum_{i=0}^{k+1} 2^i = 2^{k+2} - 1$. The proof proceeds as follows.

Portions © 2000 Rance Cleaveland © 2004 James Riely

Automata Theory and Formal Grammars: Lecture 1 - p.66/74

Strong Induction (Skip)

- Also used to prove statements of form $\forall n \in \mathbb{N}. P(n)$
- Like regular induction but with "stronger" induction hypothesis and no explicit base case.

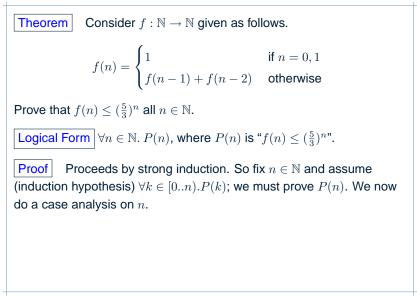
Notation $[i..j) = \{i, i+1, ..., j-1\}.$

Strong induction argument consists of proof of following

$$\forall n \in \mathbb{N}. (\forall k \in [0..n). P(k)) \longrightarrow P(n)$$

- $\forall k \in [0..n)$. P(k) is usually called the induction hypothesis.
- Proof usually requires a case analysis on values *n* can take.

Example Strong Induction Proof (Skip)



Portions © 2000 Rance Cleaveland © 2004 James Riely

Proving Properties of Recursively Defined Sets

Suppose A is a recursively defined set; how do we prove a statement of form:

 $\forall a \in A. P(a)$

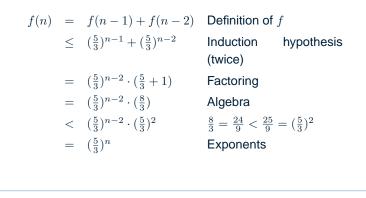
Use induction!

- Recall that $A = \bigcup_{i=0}^{\infty} A_i$.
- $\forall a \in A. P(a)$ is logically equivalent to $\forall k \in \mathbb{N}. (\forall a \in A_k. P(a)).$
- The latter statement has the correct form for an induction proof!

Example Strong Induction Proof (cont.) (Skip)

n = 0: We must show P(0), i.e. $f(0) \le (\frac{5}{3})^0$. But $f(0) = 1 = (\frac{5}{3})^0$. n = 1: We must show P(1), i.e. $f(1) \le (\frac{5}{3})^1$. This follows because $f(1) = 1 < \frac{5}{3} = (\frac{5}{3})^1$.

 $n \ge 2$: In this case we argue as follows.



Portions © 2000 Rance Cleaveland © 2004 James Riely

Automata Theory and Formal Grammars: Lecture 1 - p.70/74

Example Proof about Recursively Defined Set

Theorem Let $A \subseteq \mathbb{N}$ be the set defined as follows.		
1. $0 \in A$		
2. If $a \in A$ then $a + 3 \in A$.		
Prove that any $a \in A$ is divisible by 3.		
Logical form $\forall a \in A. P(a)$, where $P(a)$ is "a is divisible by 3".		
Proof Proceeds by induction. The statement to be proved has form $\forall k \in \mathbb{N}. Q(k)$, where $Q(k)$ is $\forall a \in A_k. P(a)$.		
Base case: $k = 0$. We must prove $Q(0)$, i.e. $\forall a \in A_0$. $P(a)$, i.e. for every $a \in A_0$, a is divisible by 3. This follows immediately since $A_0 = \emptyset$.		

Automata Theory and Formal Grammars: Lecture 1 - p.69/74

Sample Proof (cont.)

Induction step: We must prove that $\forall k \in \mathbb{N}$. $(Q(k) \longrightarrow Q(k+1))$. So fix $k \in \mathbb{N}$ and assume Q(k), i.e. $\forall a \in A_k$. P(a) (induction hypothesis). We must show $Q(k+1) = \forall a \in A_{k+1}$. P(a), i.e. we must show that every $a \in A_{k+1}$ is divisible by 3 under the assumption that every $a \in A_k$ is divisible by 3. So fix a generic $a \in A_{k+1}$. Based on the definition of A a is added into A_{k+1} in one of two ways.

- a = 0. In this case *a* is certainly divisible by 0, since $0 = 0 \cdot 3$.
- a = b + 3 for some $b \in A_k$. By the induction hypothesis b is divisible by 0, i.e. $b = 3 \cdot c$ some $c \in \mathbb{N}$. But then $a = b + 3 = 3 \cdot (c + 1)$, and thus a is divisible by 3.

Notes on Proof

- In the induction proof the base case was trivial; this will always be the case when using induction to prove properties of recursive sets! So it can be omitted.
- The induction step amounts to showing that the constants have the right property and that each application of a rule "preserves" the property.

Portions © 2000 Rance Cleaveland © 2004 James Riely

Automata Theory and Formal Grammars: Lecture 1 - p.73/74

Portions © 2000 Rance Cleaveland © 2004 James Riely

Automata Theory and Formal Grammars: Lecture 1 - p.74/74