Automata Theory and Formal Grammars: Lecture 1

Sets, Languages, Logic

Sets, Languages, Logic

Today

- Course Overview
- Administrivia
- Sets Theory (Review?)
- Logic, Proofs (Review?)
- Words, and operations on them: $w_1 \circ w_2, w^i, w^*, w^+$
- Languages, and operations on them: $L_1 \circ L_2, L^i, L^*, L^+$

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.

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

Set Theory: Sets, Functions, Relations

Sets

Sets are collections of objects.

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

Sets are unordered and insensitive to repetition.

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

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 $= \{ \langle a, b \rangle \mid a \in A, b \in B \}$
$\lvert A \rvert$	size (cardinality, or number of elements)

Examples

Let $A = \{\mathsf{m}, \mathsf{n}\}$ and $B = \{\mathsf{x}, \mathsf{y}, \mathsf{z}\}$

- What is $|A|$? $|B|$?

  $2, 3$
- What is $A \times B$? $|A \times B|$?

  $\{\langle \mathsf{m}, \mathsf{x}\rangle, \langle \mathsf{m}, \mathsf{y}\rangle, \langle \mathsf{m}, \mathsf{z}\rangle, \langle \mathsf{n}, \mathsf{x}\rangle, \langle \mathsf{n}, \mathsf{y}\rangle, \langle \mathsf{n}, \mathsf{z}\rangle\}$, $2 \times 3 = 6$
- What is 2^A? $|2^A|$?

  $\{\emptyset, \{\mathsf{m}\}, \{\mathsf{n}\}, \{\mathsf{m}, \mathsf{n}\}\}$, $2^2 = 4$
- What is 2^B? $|2^B|$?

  $\{\emptyset, \{\mathsf{x}\}, \{\mathsf{y}\}, \{\mathsf{z}\}, \{\mathsf{x}, \mathsf{y}\}, \{\mathsf{x}, \mathsf{z}\}, \{\mathsf{y}, \mathsf{z}\}, \{\mathsf{x}, \mathsf{y}, \mathsf{z}\}\}$, $2^3 = 8$

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

$$A \cup \emptyset = A$$
$$A \cap \emptyset = \emptyset$$
$$\circ A \cup B = \circ A \cap \circ B \quad \text{(De Morgan)}$$
$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C) \quad \text{(Distributivity)}$$

Cardinality

Cardinality is easy with finite sets.

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

What about infinite ones?

To answer this we need to understand functions.

Binary Relations

... relate elements of a set to other elements in the set.

Definition Let A be a set. Then R is a binary relation over A if $R \subseteq A \times A$.

Notation We usually write $a_1 \, R \, a_2$, rather than $\langle a_1, a_2 \rangle \in R$.

Examples

- $\{\langle 0, 1 \rangle, \langle 1, 2 \rangle\}$ is a binary relation over $\mathbb{N}$.
- $\{ \langle n, n \rangle \mid n \in \mathbb{N} \}$ is a binary relation over $\mathbb{N}$.

Equivalence Relations

When is $R \subseteq A \times A$ an equivalence relation?

R must be

- **reflexive** $a_1 \, R \, a_1$ holds for any $a_1 \in A$.
- **symmetric** $a_1 \, R \, a_2$ implies $a_2 \, R \, a_1$ for any $a_1, a_2 \in A$.
- **transitive** $a_1 \, R \, a_2$ and $a_2 \, R \, a_3$ implies $a_1 \, R \, a_3$ for all $a_1, a_2, a_3 \in A$.

As an example, consider $=_3 \subseteq \mathbb{N} \times \mathbb{N}$ defined by $i =_3 j$ if and only if

$$i \text{ modulo } 3 = j \text{ modulo } 3$$

For example

$$0 =_3 3 =_3 6 \quad \neq_3 \quad 1 =_3 4 =_3 7$$

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 $[1]_{=_3}$?

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

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 $\left|\{ b \mid a \, R \, b \}\right| = 1$.
  If we require only determinism, we define partial functions.
- Functions map elements from one set to elements from another.
  $f : A \to B$
  - A : domain of f
  - B : codomain of f
  - $f(a)$: result of applying f to $a \in A$ — $f(a) \in B$.

Relational Inverse

$R^{-1} \subseteq B \times A$ is the inverse of $R \subseteq A \times B$.

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

Is the inverse of a function always a function?

Bijections

When Is $f : A \to B$ ...

- ... 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

Which $f : \mathbb{N} \to \mathbb{N}$ Is Injective/Surjective?

$f(x) = x + 1$	injective, not surjective
$f(x) = \lfloor \frac{x}{2} \rfloor$	surjective, not injective
$f(x) = \lvert x \rvert$	bijective

What if instead $f : \mathbb{Z} \to \mathbb{Z}$?

$f(x) = x + 1$	bijective
$f(x) = \lfloor \frac{x}{2} \rfloor$	surjective, not injective
$f(x) = \lvert x \rvert$	neither injective nor surjective

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 $f \circ f^{-1}(b)$?

  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

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

How would you prove these statements?

Words

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.

Alphabets

An alphabet is a finite, nonempty set of symbols.

Examples

- { a, b, . . . , z }
- { a, b, . . . , z, ä, ö, ü, ß }
- { 0,1 }
- ASCII

Alphabets are usually denoted by Σ.

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.

Words as Lists

One can think about strings as a ε-terminated list of symbols.

Examples

- sour $=$ s $\cdot$ o $\cdot$ u $\cdot$ r $\cdot\, \varepsilon$
- süß $=$ s $\cdot$ ü $\cdot$ ß $\cdot\, \varepsilon$
- 010101110 $= 0 \cdot 1 \cdot 0 \cdot 1 \cdot 0 \cdot 1 \cdot 1 \cdot 1 \cdot 0 \cdot \varepsilon$

Operations on Words: Length

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

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

E.g.

$$\begin{aligned} |abb| &= |a \cdot b \cdot b \cdot \varepsilon| \\ &= 1 + |b \cdot b \cdot \varepsilon| \\ &= 1 + 1 + |b \cdot \varepsilon| \\ &= 1 + 1 + 1 + |\varepsilon| \\ &= 1 + 1 + 1 + 0 \\ &= 3 \end{aligned}$$

Operations on Words: Concatenation

Definition Let Σ be an alphabet. The concatenation operation $C : \Sigma^* \times \Sigma^* \rightarrow \Sigma$ is defined inductively as follows.

$$C(w_1, w_2) = \begin{cases} w_2 & \text{if } w_1 = \varepsilon \\ a \cdot (C(w_1', w_2)) & \text{if } w_1 = a \cdot w_1' \end{cases}$$

E.g.

$$\begin{aligned} C(01, 10) &= C(0 \cdot 1 \cdot \varepsilon, 10) \\ &= 0 \cdot C(1 \cdot \varepsilon, 10) \\ &= 0 \cdot 1 \cdot C(\varepsilon, 10) \\ &= 0 \cdot 1 \cdot 10 \\ &= 0110 \end{aligned}$$

Notation $C(w_1, w_2)$ usually written as $w_1 \cdot w_2$ or $w_1 w_2$.

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:

- ε is a substring of any string
- Any string is a substring of itself
- ε is a prefix and suffix of any string
- Any string is a prefix and suffix of itself

Operations on Words: Exponentiation

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

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

E.g.

$$\begin{aligned} (ab)^2 &= ab \circ (ab)^1 \\ &= ab \circ ab \circ (ab)^0 \\ &= ab \circ ab \circ \varepsilon \\ &= abab \end{aligned}$$

Operations on Words: Reverse

Definition Let Σ be an alphabet. The reverse operation $-^{\mathcal{R}} : \Sigma^* \to \Sigma^*$ is defined 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.

$$\begin{aligned} abc^{\mathcal{R}} &= (a \cdot b \cdot c \cdot \varepsilon)^{\mathcal{R}} \\ &= C((b \cdot c \cdot \varepsilon)^{\mathcal{R}}, a) \\ &= C(C((c \cdot \varepsilon)^{\mathcal{R}}, b), a) \\ &= C(C(C((\varepsilon)^{\mathcal{R}}, c), b), a) \\ &= C(C(C(\varepsilon, c), b), a) \\ &= C(C(c, b), a) \\ &= C(cb, a) \\ &= cba \end{aligned}$$

Properties of Operators on Words

$$\varepsilon \circ w = w$$

$$w \circ \varepsilon = w$$

$$w_1 \circ (w_2 \circ w_3) = (w_1 \circ w_2) \circ w_3$$

$$|w_1 \circ w_2| = |w_1| + |w_2|$$

$$w^1 = w$$

$$w^{i+j} = w^i \circ w^j$$

$$(w^{\mathcal{R}})^{\mathcal{R}} = w$$

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^*.

Formal Definitions Using Recursive Sets

More Formally: Alphabets and Words

Definition (Alphabet) An alphabet is a finite, non-empty set of symbols.

Definition (Σ^*) Let Σ be an alphabet. The set Σ^* of words (or strings) over Σ is defined recursively as follows.

- $\varepsilon \in \Sigma^*$
- If $a \in \Sigma$ and $w \in \Sigma^*$ then $a \cdot w \in \Sigma^*$

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

- $\varepsilon \in \{0,1\}^*$
- $0 \cdot \varepsilon \in \{0,1\}^*$
- $0 \cdot 1 \cdot 1 \cdot 0 \cdot \varepsilon \in \{0,1\}^*$

Notation Instances of $\cdot$, trailing ε’s are usually omitted: $0,\ \ 0110$ written rather than $0 \cdot \varepsilon,\ \ 0 \cdot 1 \cdot 1 \cdot 0 \cdot \varepsilon$.

Recall Fibonacci

The n^{th} Fibonacci number $f(n)$:

$$\begin{aligned} f(0) &= 0 \\ f(1) &= 1 \\ f(n) &= f(n-1) + f(n-2), \text{ for } n \geq 2 \end{aligned}$$

$0, 1, 1, 2, 3, 5, 8, 13, 21, ...$

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.

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.

Example A set A can be defined as follows.

- $1 \in A$
- If $a \in A$ then $a + 3 \in A$

What are elements in A?

$A = \{1, 4, 7, ...\} = [1]_{=_3}$

Elements of Recursively Defined Sets

The previous definition specifies that $A = \bigcup_{i=0}^{\infty} A_i$, where

$$\begin{aligned} A_0 &= \emptyset \\ A_{i+1} &= \{1\} \cup \{\, a+3 \mid a \in A_i \,\} \end{aligned}$$

E.g.

$$\begin{aligned} A_0 &= \emptyset \\ A_1 &= \{1\} \cup \emptyset = \{1\} \\ A_2 &= \{1\} \cup \{4\} = \{1, 4\} \\ A_3 &= \{1\} \cup \{4, 7\} = \{1, 4, 7\} \\ A_4 &= \end{aligned}$$

More Generally

Recursive set definitions consist of rules of following forms:

$c \in A$ for some constant c

If $a \in A$ and $p(a)$ then $f(a) \in A$ for some predicate p and function f

Then $A = \bigcup_{i=0}^{\infty} A_i$, where

$$\begin{aligned} A_0 &= \emptyset \\ A_{i+1} &= \{\, c \mid \ c \in A \text{ is a rule } \} \cup \\ &\quad \{\, f(a) \mid \text{If } a \in A \text{ and } p(a) \text{ then } f(a) \in A \text{ is a rule} \\ &\qquad \wedge\, a \in A_i \wedge p(a) \} \end{aligned}$$

E.g. In previous example:

$$\begin{aligned} p(a) &\quad \text{is} \quad \text{“true”} \\ f(a) &= a + 3 \end{aligned}$$

More Formally: Alphabets and Words

Definition (Σ^*) Let Σ be an alphabet. The set Σ^* of words (or strings) over Σ is defined recursively as follows.

- $\varepsilon \in \Sigma^*$
- If $a \in \Sigma$ and $w \in \Sigma^*$ then $a \cdot w \in \Sigma^*$

$(\Sigma^*) = \bigcup_{i=0}^{\infty} (\Sigma^*)_i$, where

$$\begin{aligned} (\Sigma^*)_0 &= \emptyset \\ (\Sigma^*)_{i+1} &= \varepsilon \cup \{\, a \cdot w \mid a \in \Sigma \text{ and } w \in (\Sigma^*)_i \,\} \end{aligned}$$

Convention: we write $0, 10$ rather than $0 \cdot \varepsilon, 1 \cdot 0 \cdot \varepsilon$.

An example

$(\Sigma^*) = \bigcup_{i=0}^{\infty} (\Sigma^*)_i$, where

$$\begin{aligned} (\Sigma^*)_0 &= \emptyset \\ (\Sigma^*)_{i+1} &= \varepsilon \cup \{\, a \cdot w \mid a \in \Sigma \text{ and } w \in (\Sigma^*)_i \,\} \end{aligned}$$

For example, let $\Sigma = \{0, 1\}$

$$\begin{aligned} (\Sigma^*)_0 &= \emptyset \\ (\Sigma^*)_1 &= \{\varepsilon\} \cup \emptyset = \{\varepsilon\} \\ (\Sigma^*)_2 &= \{\varepsilon\} \cup \{0, 1\} = \{\varepsilon, 0, 1\} \\ (\Sigma^*)_3 &= \{\varepsilon\} \cup \{0, 1, 00, 01, 10, 11\} = \{\varepsilon, 0, 1, 00, 01, 10, 11\} \\ (\Sigma^*)_4 &= \end{aligned}$$

$\{0, 1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, ...\}$

Generally...

$\Sigma^* = \bigcup_{i=0}^{\infty} (\Sigma^*)_i$, where

$$\begin{aligned} (\Sigma^*)_0 &= \emptyset \\ (\Sigma^*)_1 &= \{\varepsilon\} \\ (\Sigma^*)_2 &= \{\varepsilon\} \cup \{\, a \cdot \varepsilon \mid a \in \Sigma \,\} \\ &= \{\varepsilon\} \cup \{\, a \mid a \in \Sigma \,\} \\ (\Sigma^*)_3 &= \{\varepsilon\} \cup \{\, a \cdot w' \mid a \in \Sigma \wedge w' \in (\Sigma^*)_2 \,\} \\ &= \{\varepsilon\} \cup \{\, a \cdot \varepsilon \mid a \in \Sigma \,\} \cup \{\, a_1 \cdot a_2 \cdot \varepsilon \mid a_1, a_2 \in \Sigma \,\} \\ &= \{\varepsilon\} \cup \{\, a \mid a \in \Sigma \,\} \cup \{\, a_1 a_2 \mid a_1, a_2 \in \Sigma \,\} \\ &\vdots \end{aligned}$$

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

Languages

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^*.

Question What is 2^{Σ^*}?

The set of all languages over Σ!

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: $L_1 \circ L_2 = \{\, w_1 \circ w_2 \mid w_1 \in L_1 \wedge w_2 \in L_2 \,\}$.

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

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

Examples of Language Operations

$$\begin{aligned} \{ab, aa\} \circ \{bb, a\} &= \{ab \circ bb, ab \circ a, aa \circ bb, aa \circ a\} \\ &= \{abbb, aba, aabb, aaa\} \end{aligned}$$

$$\begin{aligned} \{01, 1\}^2 &= \{01, 1\} \circ \{01, 1\}^1 \\ &= \{01, 1\} \circ \{01, 1\} \circ \{01, 1\}^0 \\ &= \{01, 1\} \circ \{01, 1\} \circ \{\varepsilon\} \\ &= \{0101, 011, 101, 11\} \end{aligned}$$

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 $\emptyset^*$?

What is L^* Mathematically?

Since L^* is defined recursively, we know that $L^* = \bigcup_{i=0}^{\infty} (L^*)_i$, where:

$$(L^*)_0 = \emptyset$$
$$(L^*)_{i+1} = \{\varepsilon\} \cup \{\, u \circ v \mid u \in L \text{ and } v \in (L^*)_i \,\}$$

$$\begin{aligned}
(L^*)_1 &= \{\varepsilon\} \\
(L^*)_2 &= \{\varepsilon\} \cup \{\, w \circ \varepsilon \mid w \in L \,\} \\
&= \{\varepsilon\} \cup L \\
(L^*)_3 &= \{\varepsilon\} \cup \{\, w \circ w' \mid w \in L \wedge w' \in (L^*)_2 \,\} \\
&= \{\varepsilon\} \cup L \cup (L \circ L)
\end{aligned}$$

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

A Variation on L^*

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

- $L \subseteq L^+$.
- 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^+!

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

$$L \circ \emptyset = \emptyset \quad (1)$$
$$L \circ \{\varepsilon\} = L \quad (2)$$
$$L_1 \circ (L_2 \circ L_3) = (L_1 \circ L_2) \circ L_3 \quad (3)$$
$$L_1 \circ (L_2 \cup L_3) = (L_1 \circ L_2) \cup (L_1 \circ L_3) \quad (4)$$
$$L^1 = L \quad (5)$$
$$L^{i+j} = L^i \circ L^j \quad (6)$$
$$L^* = \bigcup_{i=0}^{\infty} L^i \quad (7)$$
$$L^+ = \bigcup_{i=1}^{\infty} L^i \quad (8)$$
$$L^+ = L \circ L^* \quad (9)$$

Logic

Logic ...

... is the study of propositions and proofs.

Propositions: Statements that are either true or false.

Proof: A rigorous argument that a proposition is true.

Propositions are built up ....

- ... from (nonlogical/domain-specific) predicates and atomic propositions...

  E.g. "x is prime", "f is differentiable"

- ... using logical constructors.

What Do the Following Logical Constructors Mean?

$\wedge$	conjunction ("and")
$\vee$	disjunction ("or")
$\neg$	negation ("not")
$\longrightarrow$	implication ("if ... then", "implies")
$\longleftrightarrow$	bi-implication ("if and only if")
$\forall$	universal quantifier ("for all")
$\exists$	existential quantifier ("there exists")

Examples (Propositions)

1. $\forall f : \mathbb{R} \to \mathbb{R}.$ "f is differentiable" $\longrightarrow$ "f is continuous"
2. $\neg\exists x \in \mathbb{N}.$ "x is prime" $\wedge$ ($\forall y \in \mathbb{N}. y \geq x \longrightarrow \neg$"$y$ is prime").

Formulas and Instantiations

Definition A formula is a proposition containing propositional and predicate variables.

E.g. $\neg(p \wedge q), \forall x : \mathbb{N}. P(x)$

Definition A substitution is a function R mapping propositional variables to propositions and predicate variables to predicates.

E.g. R where $R(p) =$ "$1 > 0$", $R(q) =$ "$1 < 0$", and $R(P) =$ "$x > x + 1$"

Definition An instantiation of a formula f by substitution R (notation: $f[R]$) is a proposition obtained by replacing each variable p in f with $R(p)$.

E.g.

$$(\neg(p \wedge q))[R] = \neg(\boxed{1 > 0} \wedge \boxed{1 < 0})$$
$$(\forall x : \mathbb{N}. P(x))[R] = \forall x : \mathbb{N}. \boxed{x + 1 > x}$$

Logical Implications, Logical Equivalences, and Tautologies

Definition Let f, g be formulas.

- f logically implies g (notation: $f \implies g$) 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, $f \equiv$ true).

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

Examples of Implications, Equivalences and Tautologies

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

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!

E.g.

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

Translating Natural Langauge into Logic

Phrase	Logical construct
"... not ..."	$\neg$
"... and ..."	$\wedge$
"... or ..."	$\vee$
"if ... then ..., "... implies ...", "... only if ..."	$\longrightarrow$
"... if and only if ...", "... is logically equivalent to ..."	$\longleftrightarrow$
"... all ...", "... any ..."	$\forall$
"... exists ...", "... some ..."	$\exists$

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.
$p \vee q$	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 $P(a)$.

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" $\wedge$ "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:

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

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

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.

Proof (cont.)

$\sum_{i=0}^{k+1} 2^i$	$=$	$(\sum_{i=0}^{k} 2^i) + 2^{k+1}$	Definition of $\sum$
	$=$	$2^{k+1} - 1 + 2^{k+1}$	Induction hypothesis
	$=$	$(2 \cdot 2^{k+1}) - 1$	Arithmetic
	$=$	$2^{k+2} - 1$	Exponentiation

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)

Theorem Consider $f : \mathbb{N} \to \mathbb{N}$ given as follows.

$$f(n) = \begin{cases} 1 & \text{if } n = 0, 1 \\ f(n-1) + f(n-2) & \text{otherwise} \end{cases}$$

Prove that $f(n) \leq (\frac{5}{3})^n$ all $n \in \mathbb{N}$.

Logical Form $\forall n \in \mathbb{N}.\ P(n)$, where $P(n)$ is "$f(n) \leq (\frac{5}{3})^n$".

Proof Proceeds by strong induction. So fix $n \in \mathbb{N}$ and assume (induction hypothesis) $\forall k \in [0..n).P(k)$; we must prove $P(n)$. We now do a case analysis on n.

Example Strong Induction Proof (cont.) (Skip)

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

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

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

$$\begin{array}{lll} f(n) & = & f(n-1) + f(n-2) & \text{Definition of } f \\ & \leq & (\frac{5}{3})^{n-1} + (\frac{5}{3})^{n-2} & \text{Induction hypothesis (twice)} \\ & = & (\frac{5}{3})^{n-2} \cdot (\frac{5}{3} + 1) & \text{Factoring} \\ & = & (\frac{5}{3})^{n-2} \cdot (\frac{8}{3}) & \text{Algebra} \\ & < & (\frac{5}{3})^{n-2} \cdot (\frac{5}{3})^2 & \frac{8}{3} = \frac{24}{9} < \frac{25}{9} = (\frac{5}{3})^2 \\ & = & (\frac{5}{3})^n & \text{Exponents} \end{array}$$

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 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 $A_0 = \emptyset$.

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.