Notations for Automata Theory

CS154

Chris Pollett

Jan 29, 2007.

Outline

- Mathematical Notation and terminology
- Sets

Mathematical Notions and Terminology

- In order to begin learning about automata theory and computability we need to first fix some common notations based on sets as well as learn about various methods of proof.

Sets

- A **set** is a group of objects represented together as a unit.

  {7, 21, 57} -- the set containing the number 7, 21, 57

  { {}, {a}, apple} -- the set containing the empty set, the set {a}, and an apple.
- We use $\in$ to mean element of and $\notin$ to mean not element of. For example,

  $7 \in \{7,21, 57\}$, $5 \notin \{7,21, 57\}$

Subsets

- To make statements about sets we will often use abbreviations:
  - We write ∀ to mean "for all", write ∃ to mean "exists", and we will write ∧, ∨, ¬, ⇒ for "and", "or", "not", and "implies"
- As an example, consider the symbol ⊆ which means subset of. This can be expressed by saying A ⊆ B means

  $\forall x\ (x \in A \Rightarrow x \in B)$. "for all objects x, x is in A implies x is in B"

  For example

  $\{7, 21\} \subseteq \{3, 7, 5, 21, 82\}$ since $7 \in \{3, 7, 5, 21, 82\}$ and $21 \in \{3, 7, 5, 21, 82\}$
- We write A=B to mean $A \subseteq B \wedge B \subseteq A$.
- We write $A \subset B$ (A is **proper subset** of B) to mean $A \subseteq B \wedge \neg B \subseteq A$

More on sets

- Order of elements doesn't matter for sets:
  $\{1,5\} = \{5,1\}$
- Repetitions also don't matter:
  $\{1,1,1,4,4,5\} = \{1, 4, 5\}$
- If we want repetitions to matter but still don't care about order then we have a **multiset**.
- To get a better grasp of how sets work, you might think about how you would implement them as a class in a computer language like Java.
- In the next few slides we will define operations on sets which would need to be implemented as methods.

Set comprehension and basic ways to make new sets

- The set which doesn't have any elements in it is called the empty set and is denoted by either {} or $\emptyset$.
- To create new sets we will sometimes write

  $\{x \mid P(x)\}$ which should be read as "the set of objects x such that property P(x) holds." (Sometimes this is called **set comprehension**)
- For example, given two sets A and B, we can:
  - Take their **union**

    $A \cup B = \{x \mid x \in A \vee x \in B\}$
  - Take their **intersection**

    $A \cap B = \{x \mid x \in A \wedge x \in B\}$
  - Take their **difference**

    $A - B = \{x \mid x \in A \wedge x \notin B\}$
- If we have a universe, U, under consideration, then taking the difference with respect to this set is called taking a **complement**. $\overline{A} = U \backslash A$.

Examples, Partitions, and DeMorgan's Laws

- Let A={1,2} and B={2,3}, then $A \cup B$= {1, 2, 3}, $A \cap B$={2}. A-B= {1}.
- If U={1,2,3}, then $\overline{A}$={3}.
- Given two sets C and D if $C \cap D = \emptyset$. We say C and D are **disjoint.**
- Notice for any set C, $C = \overline{\overline{C}}$.
- If $S_1, \ldots, S_n$ are each subsets of S such that: (1) $S_1 \cup S_2 \ldots \cup S_n = S$, (2) for each i and j, $S_i \cap S_j = \emptyset$, and (3) for each i, $S_i \neq \emptyset$; then we call $S_1, \ldots, S_n$ a **partition** of S.
- DeMorgan gave some useful relationships between the set creation operations we have defined so far. Namely, for any sets C and D, $C \cap D = \overline{C} \cup \overline{D}$ and $C \cup D = \overline{C} \cap \overline{D}$

Cartesian Product and Relations

- Another useful way to create sets is to be able to create the set of ordered pairs from two sets A,B denoted: $A \times B=\{(a,b) \mid a \in A \wedge b \in B\}$.
- Here (a,b) abbreviates {a, {a,b}}.
- This operation is called **Cartesian product**.
- We can iterate it to make 3-tuples, 4-tuples, etc: AxBxC = Ax(BxC), AxBxCxD, etc.
- If rather than use different sets we always use the same set, then we are taking the **Cartesian power**.
- We define $A^1 := A$, $A^{n+1} := A^n x A$.
- Given sets $A_1, A_2 \ldots, A_n$ a subset $R \subseteq A_1 x..x A_n$ is called an **n-ary relation**. (For n=1, **unary relation**; for n=2, **binary relation**).
- For example, a **graph** is an ordered pair (V,E) where V is a set of vertices and E is a set of edges on V. An **edge** is a pair $(v,w) \in VxV$. So E is a binary relation.

Power Set

- Given a set A, we define its power set, 2^A, to be the set of all subsets of A. i.e., $2^A :=$

  $\{X \mid X \subseteq A\}$
- For example, if $A=\{a,b,c\}$, then 2^A is

  $\{\emptyset, \{a\}, \{b\}, \{c\}, \{a,b\}, \{a,c\}, \{b,c\}, \{a,b,c\}\}$

Successor sets

- Define $S(x) := x \cup \{x\}$.
- Notice S(x) is a set with one more element than the set x. For instance:
  $S(\emptyset) = \emptyset \cup \{\emptyset\} = \{\emptyset\}$,
  $S(S(\emptyset)) = S(\{\emptyset\}) = \{\emptyset\} \cup \{\{\emptyset\}\} = \{\emptyset, \{\emptyset\}\}$,
  $S(S(S(\emptyset))) = \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\}$, etc.
- As we will see we can use this successor operation to define the natural numbers.

Axiom of Infinity

- In a similar fashion to set comprehension, we will sometimes create new sets by believing in the existence of a set which satisfies some list of properties.
- For instance, the **axiom of infinity** says there is a set satisfying:

1. $\emptyset \in \mathbf{N}$
2. if $x \in \mathbf{N}$ then $S(x) \in \mathbf{N}$.

- You might ask yourself how exactly one could implement this axiom on a computer?
  - The most important property of a set is what elements it contains.
  - We could define a class InfiniteSet.
  - Its constructor could take a Set of starting elements, *start*.
  - InfiniteSet's isIn(x) method takes a Set x and returns whether it is in the instance of InfiniteSet or not.
  - To do this, it first checks if if x is $\emptyset$ or if $x \in$ *start*.
  - If not, it check is x=S(y), for some y (this check is implementable if y is not too complicated). If not, then isIn(x) returns false.
  - If yes, then isIn(x) returns the values of isIn(y).

The Set of Natural Numbers

- We can view the smallest set satisfying the axiom of infinity as the set of **natural numbers, N**, if we interpret $\emptyset$ to mean 0 and $S(x)$ to mean $x+1$.
- To make this smallest set we can take *start*= $\emptyset$, in our constructor of InfiniteSet from the last slide
- For two natural numbers x,y we say $x<y$ if $x \in y$.
- Notice we can define addition and multiplication on the natural numbers with rules:

1. $x + 0 = x$
2. $x + S(y) = S(x+y)$

1. $x \cdot 0 = 0$
2. $x \cdot S(y) = x \cdot y + x$

- These rules give us algorithms for computing + and · based on an algorithm for $S(x)$.

Functions

- A function is a binary relation f $\subseteq$ DxR such that each x $\in$ D occurs in exactly one pair (x,y) $\in$ f.
- Rather than write f $\subseteq$ DxR, we write f: D->R to say f is a function or mapping from D (called the **domain**) to R (called the **range**).
- We write f(a)=b to say that f maps the element a of D to b of R.
- For example, f: **N** -> **N,** where $f(x) = x^2$ is a function.

Types of Functions

- A function is **one-to-one** (aka **injective),** if for every x,y in its domain $f(x) \neq f(y)$.
- For example, f: **N -> N,** where f(x)=2x is injective.
- A function is **onto** (aka **surjective),** if for every y in its range there is some x in such that f(x)=y.
- For example, f: **N -> N,** $f(x)= \lfloor x/2 \rfloor$ is surjective.
- A function is a **bijection** if it is one-to-one and onto.

Size of sets

- A set is called **finite** if there is a bijection between it and some natural number.
- The **cardinality** (aka size) of a finite set A, |A|, is the natural number it is bijective with.
- For example, consider f:{a,b}-->{∅,{∅}}=S(S(∅)) =2 defined by f(a) = ∅ and f(b) ={∅}. This is a bijection and shows |A| = 2.
- We call a set **countably infinite** if there is a bijection between it and the set of natural numbers.
- We call a set **uncountably infinite**, if it is not finite and it is not countable infinite.