

Harvard CS121 and CSCI E-207

Lecture 2: Mathematical Preliminaries

Harry Lewis

September 8, 2009

Reading: Sipser, Chapter 0

Sets

- **Sets** are defined by their members

$A = B$ means that for every x , $x \in A$ iff $x \in B$

Example: $\mathcal{N} = \{0, 1, 2, \dots\}$

- **Cardinality**

Sets can be **finite** (e.g. $\{1, 3, 5\}$) or **infinite** (e.g. \mathcal{N}).

Q: Is $\{\mathcal{N}\}$ finite?

If A is finite ($A = \{a_1, \dots, a_n\}$ for some $n \in \mathcal{N}$), then its **cardinality** (or **size**) $|A|$ is the number of elements in A .

The **empty set** \emptyset has cardinality 0.

Cardinality of infinite sets to be discussed later!

Set Operations

\cup union $\{a, b\} \cup \{b, c\} = \{a, b, c\}$

\cap intersection $\{a, b\} \cap \{b, c\} = \{b\}$

$-$ difference $\{a, b\} - \{b, c\} = \{a\}$

- A and B are **disjoint** iff $A \cap B = \emptyset$

Set Operations

\cup union $\{a, b\} \cup \{b, c\} = \{a, b, c\}$

\cap intersection $\{a, b\} \cap \{b, c\} = \{b\}$

$-$ difference $\{a, b\} - \{b, c\} = \{a\}$

- A and B are **disjoint** iff $A \cap B = \emptyset$
- The **power set** of $S = P(S) = \{X : X \subseteq S\}$
 - e.g. $P(\{a, b\}) = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$
 - Q: What is $|P(S)|$?

Set Operations

\cup union $\{a, b\} \cup \{b, c\} = \{a, b, c\}$

\cap intersection $\{a, b\} \cap \{b, c\} = \{b\}$

$-$ difference $\{a, b\} - \{b, c\} = \{a\}$

- A and B are **disjoint** iff $A \cap B = \emptyset$
- The **power set** of $S = P(S) = \{X : X \subseteq S\}$

e.g. $P(\{a, b\}) = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$

Q: What is $|P(S)|$?

- The **Cartesian product** of sets A, B

$A \times B = \{(a, b) : a \in A, b \in B\}$

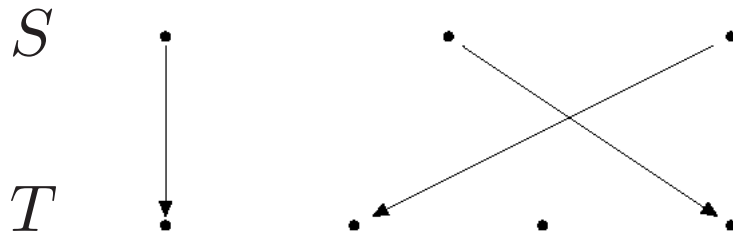
triples, ...

Functions

A **function** $f : S \rightarrow T$ maps each element $s \in S$ to (exactly one) element of T , denoted $f(s)$.

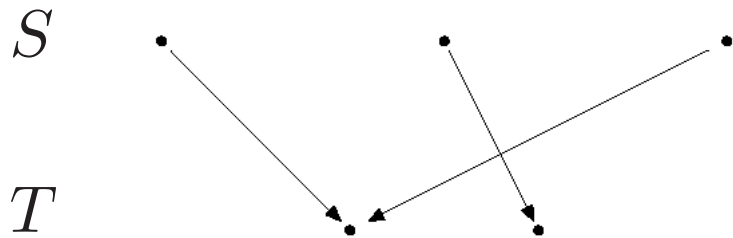
For example, $f(n) = n^2$ is a function from $\mathcal{Z} \rightarrow \mathcal{N}$
($\mathcal{Z} =$ all integers)

Special varieties of functions



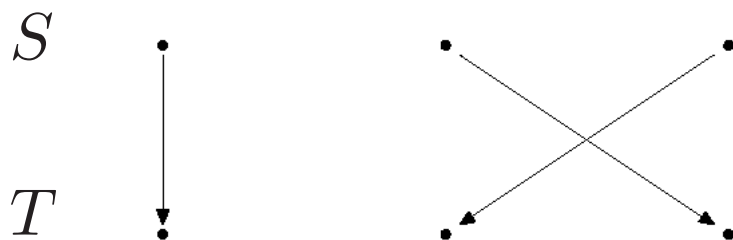
1-1:

$$s_1 \neq s_2 \Rightarrow f(s_1) \neq f(s_2)$$



Onto:

For every $t \in T$
there is an $s \in S$
such that $f(s) = t$

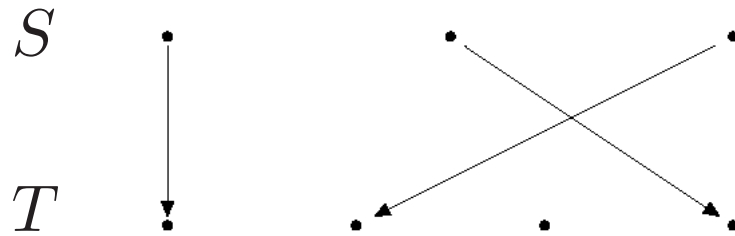


Bijection:

1-1 and onto

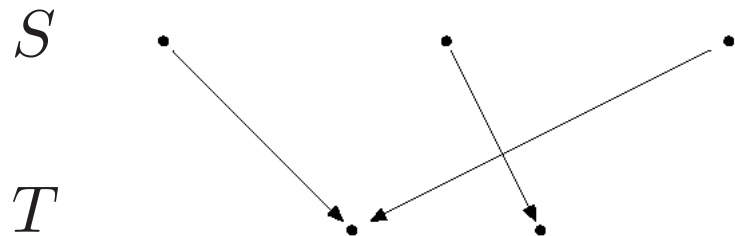
“1-1 Correspondence”

Special varieties of functions



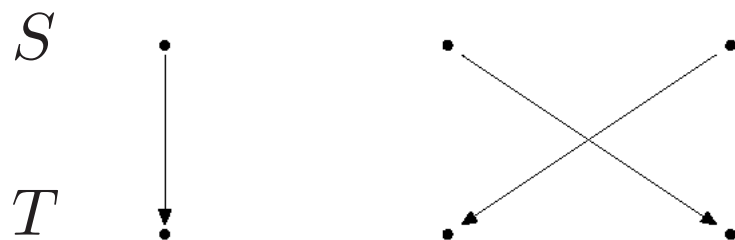
1-1:

$$s_1 \neq s_2 \Rightarrow f(s_1) \neq f(s_2)$$



Onto:

For every $t \in T$
there is an $s \in S$
such that $f(s) = t$



Bijection:

1-1 and onto

“1-1 Correspondence”

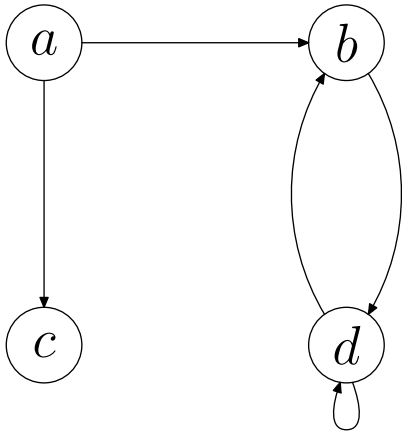
- Formal definition of cardinality: S has (finite) cardinality $n \in \mathcal{N}$ iff there is a bijection $f: \{1, \dots, n\} \rightarrow S$.

Relations

- A **k -ary relation** on S_1, \dots, S_k is a subset of $S_1 \times \dots \times S_k$
[A function $f : S \rightarrow T$ corresponds to the relation $\{(s, f(s)) : s \in S\} \subseteq S \times T$.]
- A **binary relation** on S is a subset of $S \times S$
- For example, $(\text{GWB}, \text{GHWB}) \in \text{Son}$, where $\text{Son} = \{(x, y) : x \text{ is a son of } y\}$

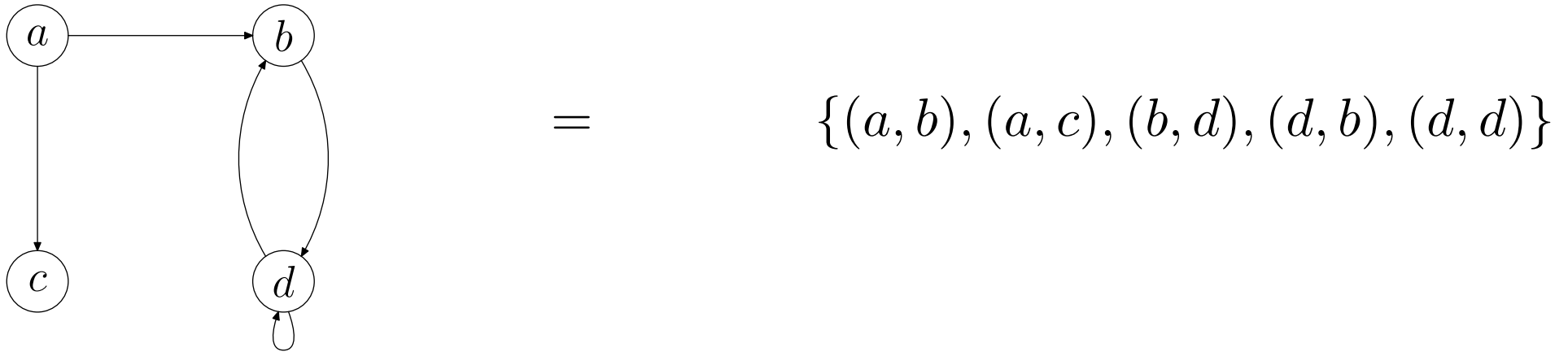
What is a (directed) graph?

- For finite S , a binary relation can be pictured as a “directed graph”:



What is a (directed) graph?

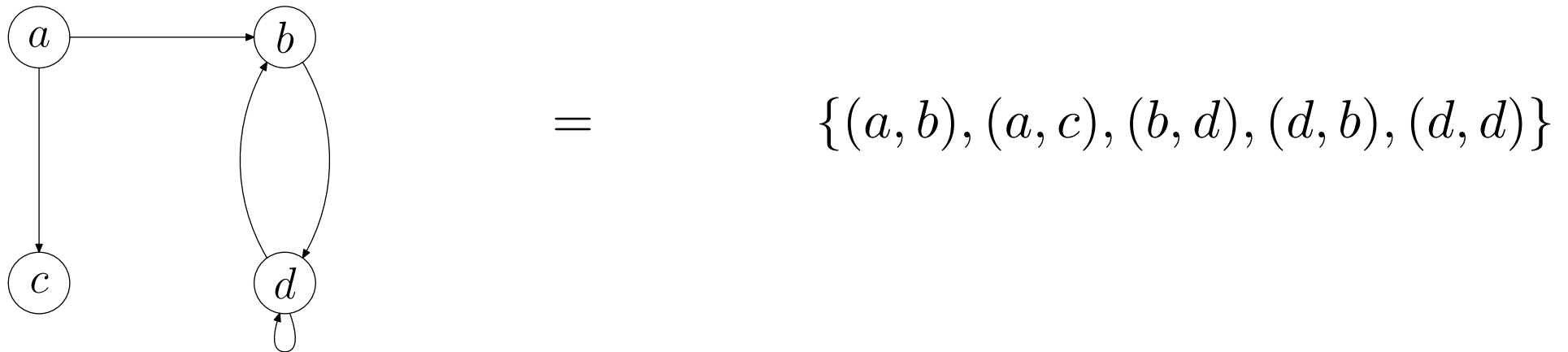
- For finite S , this can be pictured as a “directed graph”:



- Formally, a **directed graph** G consists of a finite set V of **vertices** (or **nodes**), and a set of edges $E \subseteq V \times V$.

What is a (directed) graph?




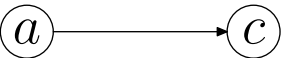
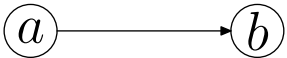
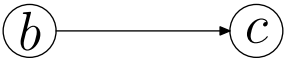
- For finite S , this can be pictured as a “directed graph”:

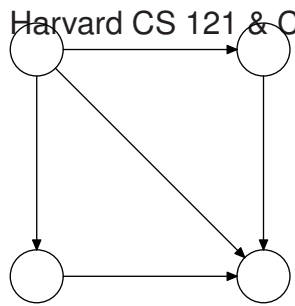


- Formally, a **directed graph** G consists of a finite set V of **vertices** (or **nodes**), and a set of edges $E \subseteq V \times V$.
- **NB:** Because a relation (or edge set) is a *set* (of ordered pairs), there can be only one arrow from one node to another.

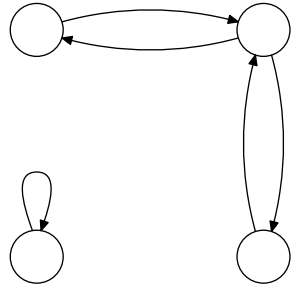
Properties of Binary Relations

A relation $R \subseteq S \times S$ is:

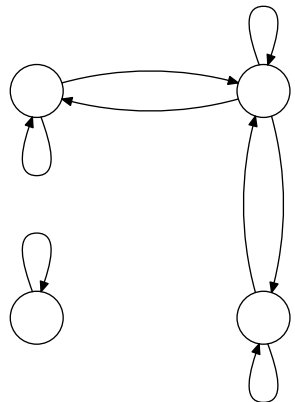
- **reflexive** if  for each $a \in S$
i.e., $(a, a) \in R$ for each $a \in S$
- **symmetric** if  whenever 
i.e., for any $a, b \in S$, if $(a, b) \in R$ then $(b, a) \in R$
- **transitive** if  whenever
 and 
i.e., if $(a, b) \in R$ and $(b, c) \in R$, then $(a, c) \in R$



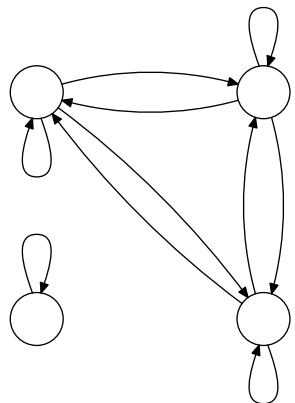
Transitive, not symmetric nor reflexive



Symmetric, not reflexive, nor transitive



Reflexive & symmetric, not transitive



Reflexive, transitive, and symmetric

Equivalence Relations

A relation that satisfies all three properties is called an **equivalence relation**

An equivalence relation decomposes S into **equivalence classes**—any two members of the same equivalence class bear the relation to each other.

Which Properties Do These Relations Have?

Domain

Relation

People

Lives-In-The-Same-City-As

People

Is-An-Ancestor-Of

People

Is-A-Brother-Of

Strings and Languages

- **Symbol** a, b, \dots
- **Alphabet** A finite, nonempty set of symbols
usually denoted by Σ
- **String** (informal) Finite number of symbols “put together”
e.g. $abba, b, bb$
Empty string denoted by ε
- Σ^* = set of all strings over alphabet Σ
e.g. $\{a, b\}^* = \{\varepsilon, a, b, aa, ab, \dots\}$

More on Strings

- **Length** of a string x is written $|x|$

$$|abba| = 4$$

$$|a| = 1$$

$$|\varepsilon| = 0$$

The set of strings of length n is denoted Σ^n .

Concatenation

- **Concatenation** of strings

Written as $x \cdot y$, or just xy

Just follow the symbols of x by the symbols of y

$$x = abba, y = b \Rightarrow xy = abbab$$

$$x\varepsilon = \varepsilon x = x \text{ for any } x$$

- The **reversal** x^R of a string x is x written backwards.

$$\text{If } x = x_1x_2 \cdots x_n, \text{ then } x^R = x_nx_{n-1} \cdots x_1.$$

Formal Inductive Definitions

- Like recursive data structures and recursive procedures when programming.

- **Strings** and their **length**:

ε is a string of length 0.

If x is a string of length n and $\sigma \in \Sigma$, then $x\sigma$ is a string of length $n + 1$.

(i.e. start with ε and add one symbol at a time, like $\varepsilon a a b a$, but we don't write the initial ε unless the string is empty)

Inductive definitions of string operations

- The **concatenation** of x and y , defined by induction on $|y|$.

$$[|y| = 0] \quad x \cdot \varepsilon = x$$

$$[|y| = n + 1] \text{ write } y = z\sigma \text{ for some } |z| = n, \sigma \in \Sigma$$
$$\text{define } x \cdot (z\sigma) = (x \cdot z)\sigma$$

Inductive definitions of string operations

- The **concatenation** of x and y , defined by induction on $|y|$.

$$[|y| = 0] \quad x \cdot \varepsilon = x$$

$$[|y| = n + 1] \text{ write } y = z\sigma \text{ for some } |z| = n, \sigma \in \Sigma$$

$$\text{define } x \cdot (z\sigma) = (x \cdot z)\sigma,$$

- The **reversal** of x , defined by induction on $|x|$:

$$[|x| = 0] \quad \varepsilon^R = \varepsilon$$

$$[|x| = n + 1] \quad (y\sigma)^R = \sigma \cdot y^R,$$

$$\text{for any } |y| = n, \sigma \in \Sigma$$

A Proof by Induction

Proposition: $(x \cdot y) \cdot z = x \cdot (y \cdot z)$ for all $x, y, z \in \Sigma^*$.

(So it doesn't matter what order we concatenate, so we can just write xyz in the future)

- Proof by “induction” on $|z|$...

Proof that $(x \cdot y) \cdot z = x \cdot (y \cdot z)$, cont.

Proofs by Induction

To prove $P(n)$ for all $n \in \mathcal{N}$:

1. “Base Case”: Prove $P(0)$.
2. “Induction Hypothesis”: Assume that $P(k)$ holds for all $k \leq n$ (where n is fixed but arbitrary)
3. “Induction Step”: Given induction hypothesis, prove that $P(n + 1)$ holds.

If we prove the Base Case and the Induction Step, then we have proved that $P(n)$ holds for $n = 0, 1, 2, \dots$ (i.e., for all $n \in \mathcal{N}$)

Languages

- A **language** L over alphabet Σ is a set of strings over Σ (i.e. $L \subseteq \Sigma^*$)

Computational problem: given $x \in \Sigma^*$, is $x \in L$?

Any YES/NO problems can be cast as a language.

- Examples of simple languages:
 - All words in the *American Heritage Dictionary*
 $\{a, aah, aardvark, \dots, zyzzyva\}$.
 - \emptyset
 - Σ^*
 - Σ
 - $\{x \in \Sigma^* : |x| = 3\} = \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$

More complicated languages

- The set of strings $x \in \{a, b\}^*$ such that x has more a 's than b 's.
- The set of strings $x \in \{0, 1\}^*$ such that x is the binary representation of a prime number.
- All 'C' programs that do not go into an infinite loop.
- $L_1 \cup L_2$, $L_1 \cap L_2$, $L_1 - L_2$ if L_1 and L_2 are languages.
- \vdots

The highly abstract and metaphorical term “language”

- A language can be either finite or infinite
- A language need not have any “internal structure”

Be careful to distinguish

ε The empty string (a string)

\emptyset The empty set (a set, possibly a language)

$\{\varepsilon\}$ The set containing one element, which is the empty string (a language)

$\{\emptyset\}$ The set containing one element, which is the empty set (a set of sets, maybe a set of languages)

Operations on Languages

- Set operations \cup \cap $-$

- Concatenation of Languages

$$L_1L_2 = \{xy : x \in L_1, y \in L_2\}$$

e.g. $\{a, b\}\{a, bb\} = \{aa, ba, abb, bbb\}$

e.g. $\{\varepsilon\}L = L$

e.g. $\emptyset L = ?$

Kleene star

- Kleene Star

$$L^* = \{w_1 \cdots w_n : n \geq 0, w_1, \dots, w_n \in L\}$$

e.g. $\{aa\}^* = \{\varepsilon, aa, aaaa, \dots\}$

e.g. $\{ab, ba, aa, bb\}^* = \text{all even length strings}$

e.g. $\Sigma^* = \text{Kleene Star of } \Sigma$

e.g. $\emptyset^* = ?$