

Harvard CS121 and CSCI E-207

Lecture 3: Doing Proofs

Harry Lewis

September 10, 2009

Reading: Sipser, Chapter 0

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 $n \in \mathcal{N}$)

What is a proof?

A proof is a formal argument of the truth of some mathematical statement.

- “Formal” means that the successive statements are unambiguous, and the steps interlock in a logical vise-grip.
- “Formal” also means that the argument could, in principle, be put in syntax that a machine could check.
- But proofs are meant to be read by human beings, and ordinary conventions and courtesies of human communication should be observed!

An example (Sipser, Theorem 0.20)

Prove that $\overline{A \cup B} = \overline{A} \cap \overline{B}$.

An example (Sipser, Theorem 0.20)

Prove that $\overline{A \cup B} = \overline{A} \cap \overline{B}$.

1. Do we know what the statement to be proved means?
 - What kinds of things does it talk about? (What are A and B ?)
 - What does the notation mean? (What are \cup and $\overline{\quad}$?)

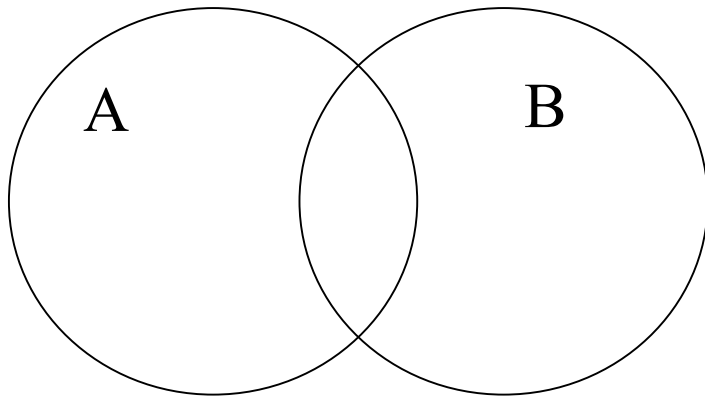
An example (Sipser, Theorem 0.20)

Prove that $\overline{A \cup B} = \overline{A} \cap \overline{B}$.

1. Do we know what the statement to be proved means?
 - What kinds of things does it talk about? (What are A and B ?)
 - What does the notation mean? (What are \cup and $\overline{}$?)
2. Try a simple example first.
 - Say, $A = \{1, 2\}$ and $B = \{2, 3\}$.
 - Back up to Step 1 if necessary! Ask `cs121@seas.harvard.edu` or `cscie207@fas.harvard.edu` if necessary but most of the time the information is in the notes and problem sets.

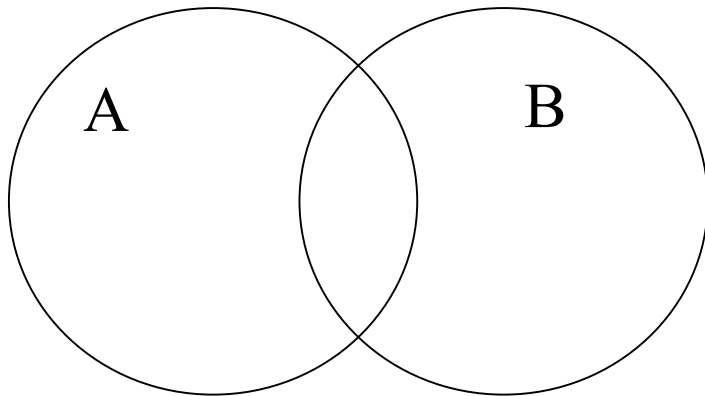
Proof example, continued

3. Try drawing a picture and play with it.



Proof example, continued

3. Try drawing a picture and play with it.



4. Decide on a proof strategy

- If we are trying to prove two sets equal, a good strategy is *mutual inclusion*: Prove everything in the first is in the second, and then that everything in the second is in the first.
- This illustrates a more general strategy: Try breaking the problem into simpler chunks and solving them separately.

Now really do the proof

1. Prove that for any sets A and B , $\overline{A \cup B} \subseteq \overline{A} \cap \overline{B}$.
2. Prove that for any sets A and B , $\overline{A} \cap \overline{B} \subseteq \overline{A \cup B}$.

(Done in class.)

Hints for writing up good proofs (thanks largely to Tom Leighton)

1. State the game plan, including the general proof technique you are using.

Hints for writing up good proofs (thanks largely to Tom Leighton)

1. State the game plan, including the general proof technique you are using.
2. Keep the flow linear and use English to explain when you are moving from step to step.

Hints for writing up good proofs (thanks largely to Tom Leighton)

1. State the game plan, including the general proof technique you are using.
2. Keep the flow linear and use English to explain when you are moving from step to step.
3. Proofs are read by human beings, not machines.

Hints for writing up good proofs (thanks largely to Tom Leighton)

1. State the game plan, including the general proof technique you are using.
2. Keep the flow linear and use English to explain when you are moving from step to step.
3. Proofs are read by human beings, not machines.
4. Use as little new symbolism as possible, and use old symbolism correctly.

Hints for writing up good proofs (thanks largely to Tom Leighton)

1. State the game plan, including the general proof technique you are using.
2. Keep the flow linear and use English to explain when you are moving from step to step.
3. Proofs are read by human beings, not machines.
4. Use as little new symbolism as possible, and use old symbolism correctly.
5. Avoid “clearly,” which bullies the reader and often hides errors.

Hints for writing up good proofs (thanks largely to Tom Leighton)

1. State the game plan, including the general proof technique you are using.
2. Keep the flow linear and use English to explain when you are moving from step to step.
3. Proofs are read by human beings, not machines.
4. Use as little new symbolism as possible, and use old symbolism correctly.
5. Avoid “clearly,” which bullies the reader and often hides errors.
6. When you are done, explain why you are done.

Pigeonhole Principle

- If there are more pigeons than pigeonholes and every pigeon is in a pigeonhole, then some pigeonhole must contain at least two pigeons
- or more formally
- For any finite sets S and T and any function $f : S \rightarrow T$, if $|S| > |T|$ then there exist $s_1, s_2 \in S$ such that $s_1 \neq s_2$ but $f(s_1) = f(s_2)$
- A proof by pigeonhole: In any group of people, two have the same number of friends
- A little LaTeX ...

Nonconstructive proofs

- We have already seen a constructive proof—a proof that actually delivers the goods
 - Not every symmetric, transitive relation is reflexive
- The proof about people and their friends is nonconstructive

Nonconstructive Proof Example #2: Numbers with a certain property

- Proof that there exist irrational a, b such that a^b is rational
 - Is $\sqrt{2}^{\sqrt{2}}$ rational? Don't know. But consider both possibilities:
 1. $\sqrt{2}^{\sqrt{2}}$ is rational. In that case we are done ($a = b = \sqrt{2}$)
 2. $\sqrt{2}^{\sqrt{2}}$ is irrational. But then

$$\left(\sqrt{2}^{\sqrt{2}}\right)^{\sqrt{2}} = (\sqrt{2})^{\sqrt{2} \cdot \sqrt{2}} = (\sqrt{2})^2 = 2$$

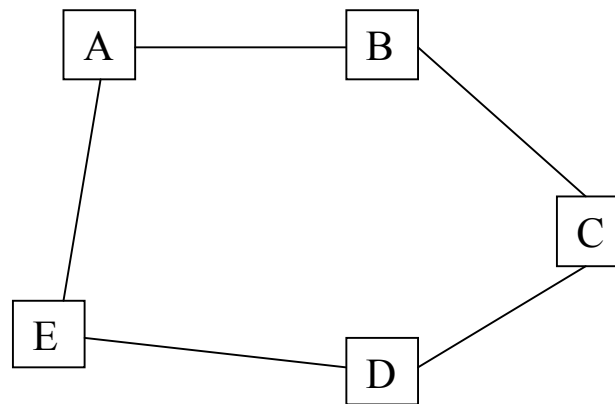
which is rational and we are done ($a = \sqrt{2}^{\sqrt{2}}, b = \sqrt{2}$)

- Proof does not tell us whether case 1 or case 2 holds, but one or the other must be true
- Law of the Excluded Middle or *tertium non datur*

A combined constructive and nonconstructive proof

(A) In any group of six people there are either three that know each other or three that don't, but (B) in some groups of five people there are no three who mutually know each other and also no three who mutually don't know each other.

- Constructive proof of (B):



Proof, continued

- Nonconstructive proof of (A) by contradiction
 - Suppose not. Then in some particular group of 6 people, there are no 3 who mutually know each other and no 3 who mutually don't.

Proof, continued

- Pick some individual X . Either X knows 3 of the other 5, or there are some 3 of the other 5 whom X does not know. (Pigeonhole)
- If X knows 3, say A, B, C , then no two of them can know each other. For example if A knew B , then X, A, B would all know each other. But then no two of A, B, C know each other, contradiction.
- If there are 3 whom X does not know, say A, B, C , then each two of those must know each other. For example, if A and B did not know each other, then no two of X, A, B would know each other. But then A, B, C all know each other, contradiction.

Proof, finis

- NB: The opposite of a “for all” statement is a “for some . . . not” or “there exists . . . not” statement.
- In theory at least, (A) could also be solved by **exhaustive search**.
- It would be enough to say that the second case is “symmetrical”
- This is also an example of case analysis

A nonconstructive proof that tells us almost nothing

- There exists an unbiased 7-sided die
- ???

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. $\emptyset^* = ?$