Mathematical Preliminaries

Sipser pages 1-28

Mathematical Preliminaries

- This course is about the fundamental capabilities and limitations of computers. It has 3 parts

1. Automata
   - Models of computation
   - These are data as well as programs
2. Computability
   - Some things cannot be solved
3. Complexity
   - what is the root of the hardness
   - can a less than perfect solution suffice
   - some are only hard i the worst case
   - could randomized computation help?
   - Cryptography, hard on purpose

What you should learn

- Understand the limits of computability
- Understand different models of computation, including deterministic and nondeterministic models
- Understand that particular models not only perform computation, but are data and can be analyzed and computed
- Have significant mastery of the techniques of reduction, diagonalization, and induction
- Demonstrate significant mastery of rigorous mathematical arguments

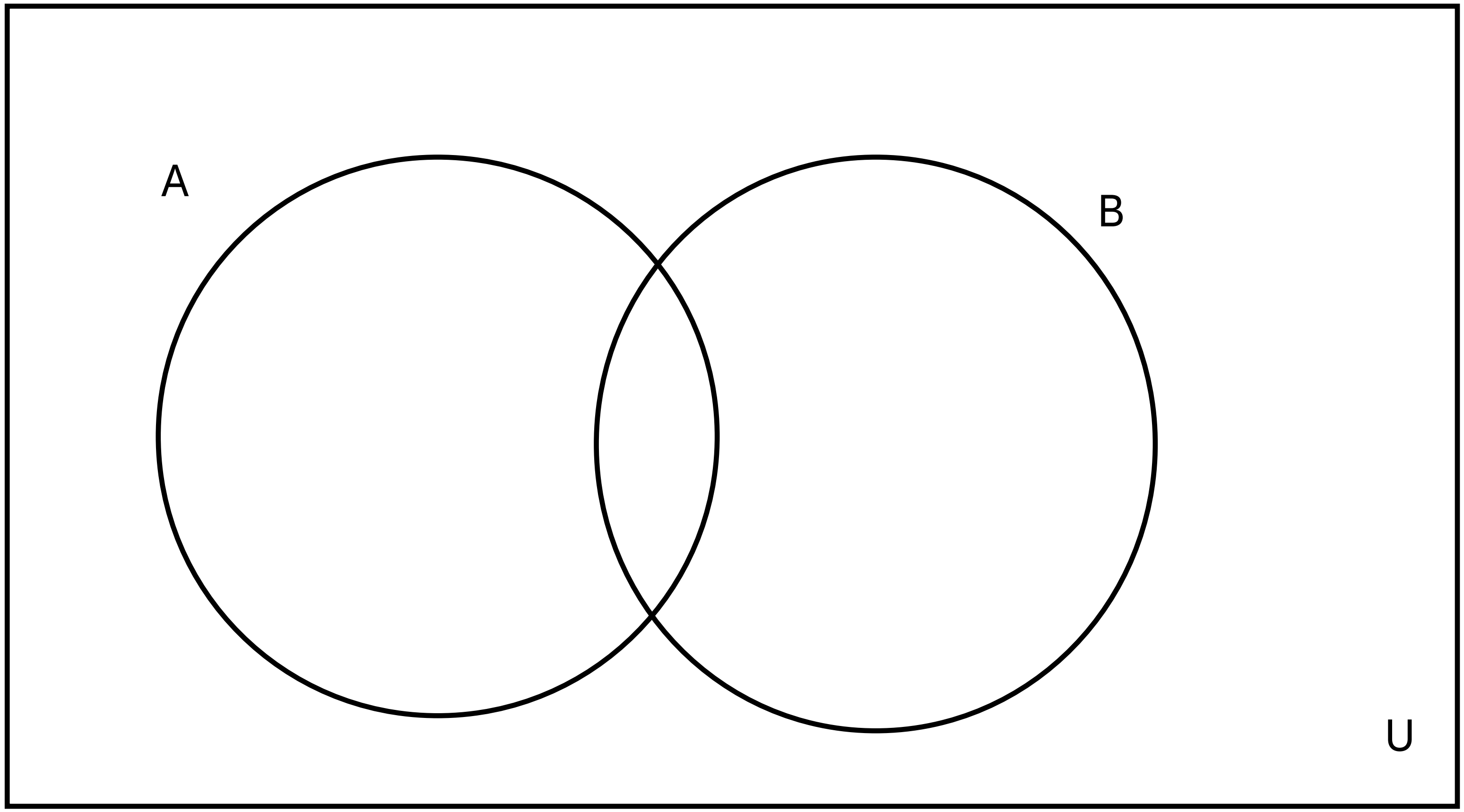
Sets

- Sets are collections in which order of elements and duplication of elements do not matter.
  - {1,a,1,1} = {a,a,a,1} = {a,1}
  - Notation for *membership*: $1 \in \{3,4,5\}$
  - *Set-former* notation: $\{x \mid P(x)\}$ is the set of all *x* which
  - satisfy the property *P*.
  - $\{x \mid x \in N \text{ and } 2 \geq x \geq 5\}$
  - $\{x \in N \mid 2 \geq x \geq 5\}$
  - Often a *universe* is specified. Then all sets are assumed to be subsets of the universe (*U*), and the notation
  - $\{x \mid P(x)\}$ stands for $\{x \in U \mid P(x)\}$

Operations on Sets

- *empty set* : $\emptyset$
- Union: $A \cup B = \{x \mid x \in A$ or $x \in B\}$
- Intersection: $A \cap B = \{x \mid x \in A$ and $x \in B\}$
- Difference: $A - B = \{x \mid x \in A$ and $x \notin B\}$
- Complement: $\underline{A} = U - A$

Venn Diagrams



Laws

- $A \cup A = A$
- $A \cup B = B \cup A$
- $A \cup (B \cup C) = (A \cup B) \cup C$
- $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
- $\underline{A \cup B} = \underline{A} \cap \underline{B}$
- $A \cup \varnothing = A$

- $A \cap A = A$
- $A \cap B = B \cap A$
- $A \cap (B \cap C) = (A \cap B) \cap C$
- $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
- $\underline{A \cap B} = \underline{A} \cup \underline{B}$
- $A \cap \varnothing = \varnothing$

Subsets and Powerset

- A is a *subset* of B if all elements of A are elements of B as well. Notation: $A \subseteq B$.
- The *powerset* $\mathbb{P}(A)$ is the set whose elements are all subsets of A: $\mathbb{P}(A) = \{X \mid X \subseteq A\}$.
- **Fact**. If A has n elements, then $\mathbb{P}(A)$ has 2^n elements.
- In other words, $|\mathbb{P}(A)| = 2^{|A|}$, where $|X|$ denotes the number of elements (*cardinality*) of X.

Proving Equality and non-equality

- To show that two sets A and B are **equal**, you need to do two proofs:
  - Assume $x \in A$ and then prove $x \in B$
  - Assume $x \in B$ and then prove $x \in A$
- **Example**. Prove that $P(A \cap B) = P(A) \cap P(B)$.

- To prove that two sets A and B are **not equal**, you need to produce a *counterexample* : an element x that belongs to one of the two sets, but does not belong to the other.
- **Example**. Prove that $P(A \cup B) \neq P(A) \cup P(B)$.
- Counterexample: A={1}, B={2}, X={1,2}. The set X belongs to $P(A \cup B)$, but it does not belong to $P(A) \cup P(B)$.

Functions and Relations

- Functions establish input-output relationships
- We write $f(x) = y$
  - For every input there is exactly one output
    - if $f(x) = y$ and $f(x)=z$ then $y=z$
  - Well call the set of input for which f is valid the domain
  - We call the set of possible output the range
  - We write $f\colon \text{Domain} \rightarrow \text{Range}$

- Some functions take more than 1 argument
  - $F(x_1, \ldots x_n) = y$
  - We call n the arity of f
  - The domain of a function with n inputs is an n-tuple

Into and Onto

- A function that maps some input to every one of the elements of the range is said to be onto. Forall y Exists x . F(x) = y

- A function is into if every element in the domain maps to some element of the range.
  - This means f(x) is defined for every x in the domain
  - The squareRoot: Real -> Real   is not into since squareRoot(-3) is not defined

Relation

- An input output relationship where a single input can have more than 1 output is called a relation.
- Less(4) = {3,2,1,0} i.e. a set of results

Because the output is not unique, we write this as Less(4,3), Less(4,2), Less(4,1), Less(4,0) we can think of this a set of tuples.

{(4,3),(4,2),(4,1),(4,0)}

Relations as sets

- An n-ary relation is a set of n-tuples.
- Some relations are infinite
  - What are some examples?

- We often use infix notation to denote binary relations $5 < 4$, $x \in S$, $(2+3) \downarrow 5$

- An n-ary function is a (n+1)-ary relation

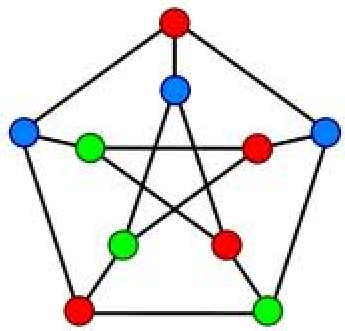
Equivalence Relations

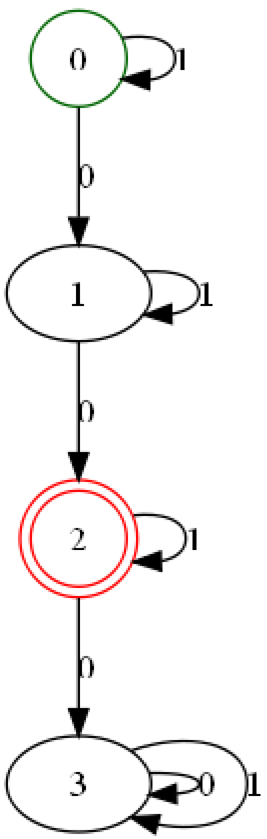
- A binary relation, •, with these three properties
  1. Reflexive $x \bullet x$
  2. Symmetric $x \bullet y$ implies $y \bullet x$
  3. Transitive $x \bullet y$ and $y \bullet z$ implies $x \bullet z$

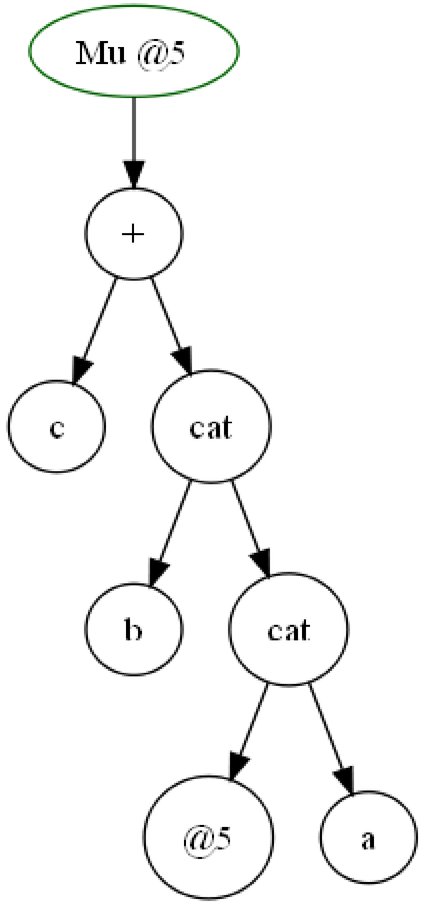
1. Is called an Equivalence Relation

Graphs

- Graphs have nodes (vertices) and edges

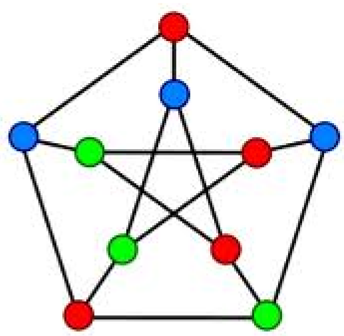






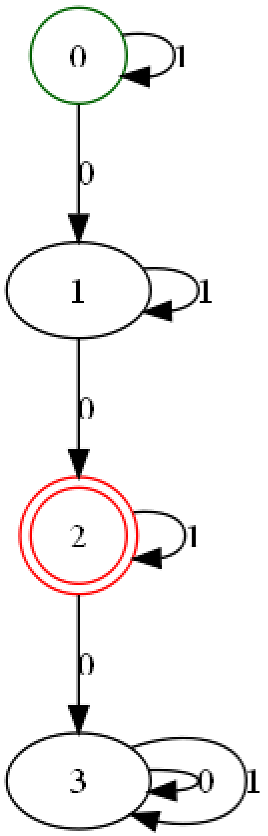
Directed Graphs

- When the edges have a direction (usually drawn with an arrow) the graph is called a directed graph



Undirected graph

Directed graph

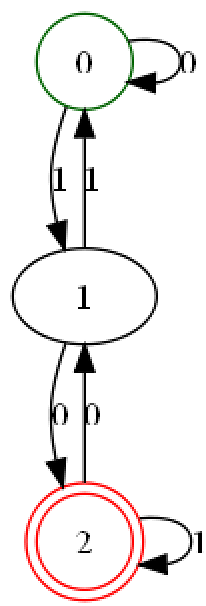


Degree

- The number of edges attached to a node is called its degree.

- In a labeled graph, nodes have in-degree and out-degree

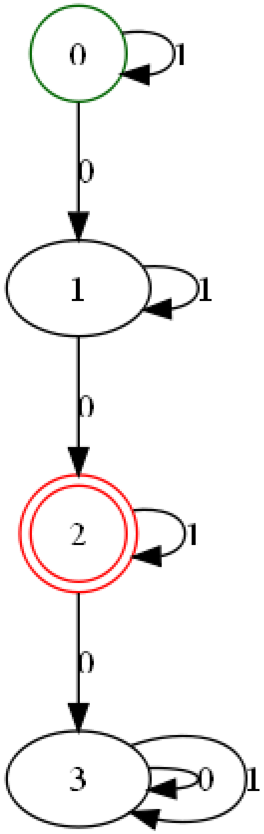
- What are the in-degree and
out-degree of node 0?

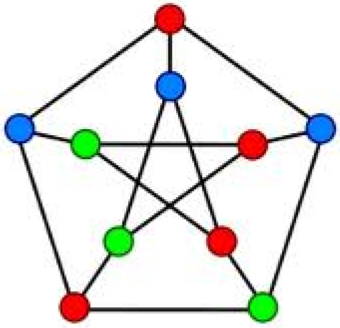


Labeled Graphs

- When the edges are labeled the graph is called a labled-graph

Labeled graph





unlabled graph

Paths

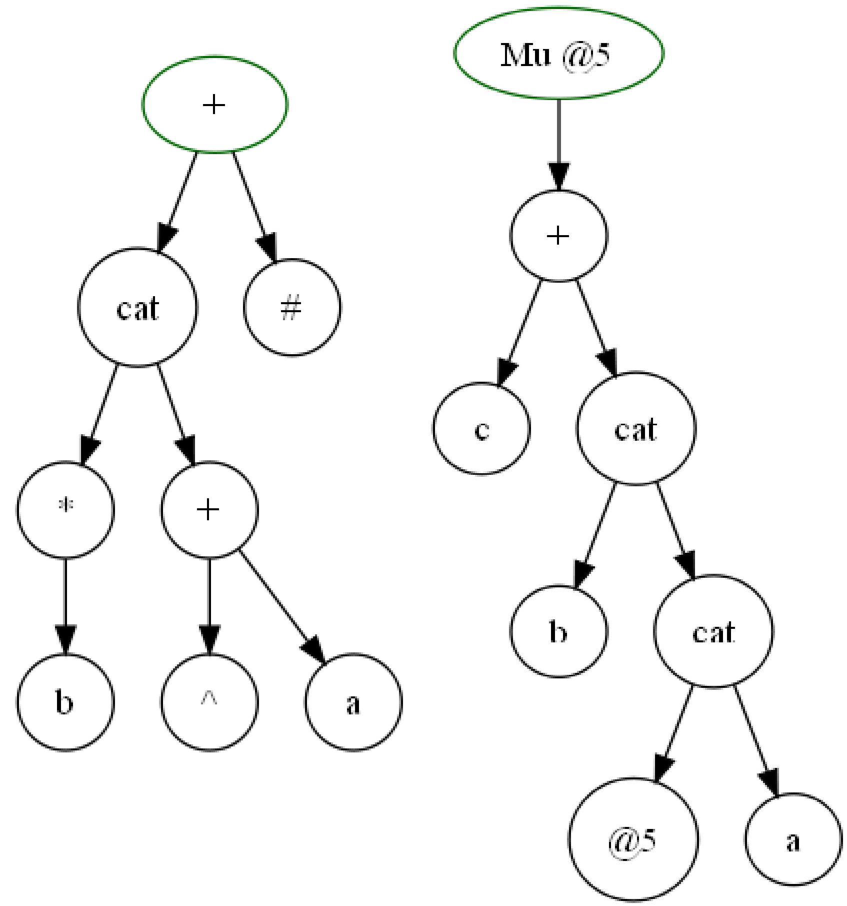
- A path is a sequence of nodes connectd by edges
- A graph is Connected if every two nodes are connected by a graph.
- A path is a cycle if the first and last node in the path are the same.
- A cycle is simple if only the first and last node are the same

Trees

- A graph is a tree if it is connected and has no simple cycles.

The unique node with in-degree 0 is called the root.

Nodes of degree 1 (other than the root) are called leaves



Strings and Languages

- Strings are defined with respect to an *alphabet*, which is an arbitrary *finite* set of symbols. Common alphabets are {0,1} (*binary*) and ASCII. But *any finite set can be an alphabet!*

- A *string* over an alphabet Σ is any finite sequence of elements of Σ.

- `Hello` is an ASCII string; 0101011 is a binary string.

- The *length* of a string w is denoted $|w|$. The set of all strings of length n over Σ is denoted Σ^n.

More strings

- $\Sigma^0=\{\boldsymbol{\varepsilon}\}$, where $\boldsymbol{\varepsilon}$ is the *empty string* (common to all alphabets). Another notation is to use Λ rather than ε
- Σ^* is the set of *all* strings over Σ:
- $\Sigma^* = \{\Sigma\} \cup \Sigma \cup \Sigma^2 \cup \Sigma^3 \cup \; \ldots$
-
- Σ^+ is Σ^* with the empty string excluded:
- $\Sigma^* = \Sigma \cup \Sigma^2 \cup \Sigma^3 \cup \; \ldots$

String concatenation

- If u=`one` and v=`two` then $u \bullet v$=`onetwo` and
- $v \bullet u$=`twoone`. Dot is usually omitted; just write uv for $u \bullet v$.
- Laws:
- $u \bullet (v \bullet w) = (u \bullet v) \bullet w$
- $u \bullet \varepsilon = u$
- $\varepsilon \bullet u = u$
- $|u \bullet v| = |u| + |v|$

- The n^{th} power of the string u is $u^n = u \bullet u \ \ldots \ u$, the concatenation of n copies of u.
- E.g., `One`^{3} = `oneoneone` .
- Note $u^0 = \varepsilon$.

Can you tell the difference?

- There are three things that are sometimes confused.

ε – the empty string (“”)

$\varnothing$ – the empty set ({ })

$\{\varepsilon\}$ – the set with just the empty string as an element

Languages

- A *language* over an alphabet Σ is any subset of Σ^*. That is, any set of strings over Σ.

- Some languages over {0,1}:
  - {ε,01,0011,000111, … }
  - The set of all binary representations of prime numbers: {10,11,101,111,1011, … }

- Some languages over ASCII:
  - The set of all English words
  - The set of all C programs

Language concatenation

- If L and L' are languages, their concatenation L • L' (often denoted LL') is the set
- $\{u \bullet v \mid u \in L \text{ and } v \in L'\}$.
-
- **Example**. {0,00} • {1,11} = {01,011,001,0011}.
-
- The n^{th} power L^n of a language L is L • L ... L, n
- times. The zero power L^0 is the language $\{\varepsilon\}$, by definition.
-
- **Example**. $\{0,00\}^4=\{0^4,0^5,0^6,0^7,0^8\}$

Kleene Star

- Elements of L^* are ε and all strings obtained by concatenating a finite number of strings in L.
  - $L^* = L^0 \cup L^1 \cup L^2 \cup L^3 \cup \ldots$
  - $L^+ = L^1 \cup L^2 \cup L^3 \cup \ldots$
  - Note: $L^* = L^+ \cup \{\varepsilon\}$
- **Example**. $\{00,01,10,11\}^*$ is the language of all even length binary strings.

Class Exercise

- Fill in the blanks to define some laws:

$L^* \cup \{\varepsilon\}$ = ____________

$L^+ \bullet \{\varepsilon\}$ = ____________

$\{\varepsilon\} \bullet \{\varepsilon\}$ = ____________

$\varnothing \bullet L$ = ____________

$L^* \bullet L^*$ = ____________

$(L^*)^*$ = ____________

$L \bullet L^*$ = ____________

$\varnothing^*$ = ____________

$\{\varepsilon\}^*$ = ____________

$L \bullet L^*$ = ____________

Mathematical Statements

- *Statements* are sentences that are true or false:
  - [1.] 0=3
  - [2.] ab is a substring of cba
  - [3.] Every square is a rectangle

-

- *Predicates* are parameterized statements; they are true or false depending on the values of their parameters.
  - [1.] $x>7$ and $x<9$
  - [2.] $x+y=5$ or $x-y=5$
  - [3.] If $x=y$ then $x^2=y^2$

Logical Connectives

- *Logical connectives* produce new statements from simple ones:
  - Conjunction; $A \wedge B$; A and B
  - Disjunction; $A \vee B$; A or B
  - Implication; $A \Rightarrow B$; if A then B
  - Negation; $\neg A$ not A
  - Logical equivalence; $A \Leftrightarrow B$
  - A if and only if B
  - A iff B

Quantifiers

- The *universal quantifier* ($\forall$ “for every”) and the *existential quantifier* ($\exists$ “there exists”) turn predicates into other predicates or statements.
  - There exists x such that $x+7=8$.
  - For every x, $x+y > y$.
  - Every square is a rectangle.

- **Example**. True or false?
  - $(\forall x)(\forall y)\ x+y=y$
  - $(\forall x)(\exists y)\ x+y=y$
  - $(\exists x)(\forall y)\ x+y=y$
  - $(\forall y)(\exists x)\ x+y=y$
  - $(\exists y)(\forall x)\ x+y=y$
  - $(\exists x)(\exists y)\ x+y=y$

Proving Implications

- Most theorems are stated in the form of (universally quantified) implication: if A, then B
- To prove it, we *assume* that A is true and proceed to derive the truth of B by using logical reasoning and known facts.
- **Silly Theorem**. If 0=3 then 5=11.
- *Proof*. Assume 0=3. Then 0=6 (why?). Then 5=11 (why?).

- Note the implicit universal quantification in theorems:
- **Theorem A**. If x+7=13, then x^2=x+20.
- **Theorem B**. If all strings in a language L have even length, then all strings in L* have even length.

Converse

- The *converse* of the implication $A \Rightarrow B$ is the implication $B \Rightarrow A$. It is quite possible that one of these implications is true, while the other is false.
- E.g., $0=1 \Rightarrow 1=1$ is true,
- but $1=1 \Rightarrow 0=1$ is false.
  - Note that the implication $A \Rightarrow B$ is true in all cases except when A is true and B is false.
-
- To prove an equivalence $A \Leftrightarrow B$, we need to prove a pair of converse implications:
  - (1) $A \Rightarrow B$,
  - (2) $B \Rightarrow A$.

Contrapositive

- The *contrapositive* of the implication $A \Rightarrow B$ is the implication $\neg B \Rightarrow \neg A$. If one of these implications is true, then so is the other. It is often more convenient to prove the contrapositive!

- **Example**. If L_1 and L_2 are non-empty languages such that $L_1^* = L_2^*$ then $L_1 = L_2$.

- *Proof*. Prove the contrapositive instead. Assume $L_1 \neq L_2$. Let w be the shortest possible non-empty string that belongs to one of these languages and does not belong to the other (e.g. $w \in L_1$ and $w \notin L_2$). Then $w \in L_1^*$ and it remains to prove $w \notin L_2^*$. [Finish the proof. Why is the assumption that $L_1, L_2 \neq \varnothing$ necessary?]

Reductio ad absurdum- Proof by Contradiction

- Often, to prove $A \Rightarrow B$, we assume both A and $\neg$ B, and then proceed to derive something absurd (obviously non-true).
-
- **Example**. If L is a finite language and $L \bullet L = L$, then $L=\varnothing$ or $L=\{\varepsilon\}$.
- *Proof*. Assume L is finite, $L \bullet L = L$, $L \neq \varnothing$, and $L \neq \{\varepsilon\}$. Let w be a string in L of maximum length. The assumptions imply that $|w|>0$. Since $w^2 \in L^2$, we must have $w^2 \in L$. But $|w^2|=2|w|>|w|$, so L contains strings longer than w. Contradiction.
- qed
-

Formatting Proofs

- A proof is a convincing argument.

- The way you format a proof (especially for a homework) can alter its degree of convincingness[1]

- There are rules for formatting a proof.

1. is that a word?

3 parts to a proof

1. Facts
   A. $(F \bullet G)\ x = F(G(x))$

Facts. Things you know, include function definitions and facts about arithmetic, like x+y = y+x. There will often be many facts

2. Assertion
   1. Prove $(F \bullet G)$ is associative
   2. $((F \bullet G) \bullet H)\ x \;=\; (F \bullet (G \bullet H))\ x$

State clearly what you are proving. Write it down in all its gory detail

3. Steps

   Transforming lhs into rhs

   $((F \bullet G) \bullet H)\ x$ by A

   $(F \bullet G)\ (H\ x)$ by A

   $(F\ (G(H\ x))$ by A

   $(F\ ((G \bullet H)\ x))$ by A backwards

   $(F \bullet (G \bullet H))\ x$ by A backwards

Steps. List each step. Every step should be justified by a fact. State how you are performing the steps. Here I transform lhs into rhs.