About These Notes

- Designed to be used with Sipser’s *Introduction to the Theory of Computation*
- Available through “Lecture notes” link on course web page
- Note that examples & various other things are not included here
Basic Objects and Notation

- $\mathbb{N} = \{ 1, 2, 3, 4, ... \}$
  - Some texts include 0; Sipser doesn't
- $\mathbb{Z} = \{ ..., -2, -1, 0, 1, 2, ... \}$
  - $\mathbb{Z}^+ = \mathbb{N}$
- $\mathbb{R}$ = set of all real numbers
- $\mathbb{Q}$ = set of all rational (quotient) numbers
- PosEven = $\{ 2n | n \in \mathbb{N} \}$
  - Tiny constraints are sometimes added to the left of $|$, as in
    - $\text{PosEven} = \{ n \in \mathbb{N} : n \% 2 = 0 \}$
  - Sometimes : is used instead of |
Scope of Intentional Notation

- Variables inside \( \{ x \mid \text{pred}(x) \} \) are local
- Think of the specification as a *mathematical program*
  - We will see many programming languages this term: DFAs, NFAs, Regex, PDAs, TMs, C++, ...
  - Mathematical notation is a type of precise specifier – i.e., a programming language
  - Turns out it is far more powerful than our ordinary programming languages – we'll prove this later
Scope of Intentional Notation

\[ \{ 2n \mid n \in \mathbb{N} \} = \{ n \in \mathbb{N} : n \% 2 = 0 \} \]

These are two different programs that produce the same set.
Set Operations

- **∪** = Union = Disjunction = Or
  - \{ 0, 3, 6, 9 \} ∪ \{ 0, 2, 4, 6, 8 \} = ?

- **∩** = Intersection = Conjunction = And
  - \{ 0, 3, 6, 9 \} ∩ \{ 0, 2, 4, 6, 8 \} = ?
Set Operations

- Complement
  PosOdd = PosEven\(^c\) ... depending
  - For a set A,
    \[ A^c = \overline{A} = \text{Universe} - A \]
    Universe is implicit. Be careful!!!

- Set difference
  PosOdd = \(\mathcal{N}\) - PosEven = \{ 1, 3, 5, ...\}
  - For sets A & B,
    \[ A - B = \{ x \mid x \in A \text{ and } x \notin B \} \]
    \[ = A \cap B^c \]
More Set Operations

- Cardinality
  If $A$ is a set, then

$$|A| = \begin{cases} n & \text{if } n \in \mathbb{Z} \text{ and } A \text{ contains } n \text{ elements} \\ \infty & \text{otherwise} \end{cases}$$

$\infty$ is not very precise, but oh well. We’ll improve upon this later when we start counting infinities.
More on Sets

- The empty set: $\emptyset = \{\}$
- The number of $\{\}$ matters:
  - $\emptyset \neq \{ \emptyset \} \neq \{ \{ \emptyset \}, \emptyset \}$
- Elements of a set
  - $\{ 5 \} \in \{ 1, \{2,3\}, \emptyset, \{ 5 \}, 2 \}$
  - $\{ 3 \} \not\in \{ 1, \{2,3\}, \emptyset, \{ 5 \}, 2 \}$
  - $\emptyset \in \{ 1, \{2,3\}, \emptyset, \{ 5 \}, 2 \}$
  - Is $\emptyset$ always a member of a set even if it is not explicitly listed?
- Subset
  - $\{ 1, \{5\} \} \subseteq \{ 1, \{2,3\}, \emptyset, \{ 5 \}, 2 \}$
  - $\emptyset \subseteq \{ 1, \{2,3\}, \emptyset, \{ 5 \}, 2 \}$
  - $\{2,3\} \not\subset \{ 1, \{2,3\}, \emptyset, \{ 5 \}, 2 \}$
More Set Operations

- Cartesian Product (aka Cross Product)
  If A and B are sets, then
  \[ A \times B = \{ (a,b) \mid a \in A \text{ and } b \in B \} \]

  - Note that the \( \times \) operator "preserves structure" by wrapping parentheses and commas around its arguments
  - \( \{ 1,3 \} \times \{ c, d, f \} = ? \)
  - If \(|A| = n\) and \(|B| = m\), then \(|A \times B| = ?\)
More Set Operations

- Generalizing to more sets:

\[ \prod_{i=1}^{3} A_i = A_1 \times A_2 \times A_3 \]
More Set Operations

- **Power set**  
  \[ \mathcal{P}(A) = 2^A = \{ x \mid x \subseteq A \} \]

- **Important equivalence**  
  \[ x \subseteq A \text{ means the same as } x \in \mathcal{P}(A) \]

- **Examples:**  
  - \[ \mathcal{P}\{1,2\} = \{ \emptyset, \{1\}, \{2\}, \{1,2\} \} \]
  - \[ \mathcal{P}(\emptyset) = ? \]
  - \[ \mathcal{P}(\mathbb{N}) = ? \]
  - If \(|A|=n\) then \(|\mathcal{P}(A)| = ?\)  
  (Implicit here that \(n \neq \infty\))
Propositional logic

- Variables stand for truth values
- Simple procedure for evaluating truth value of statement

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>x ∨ y</th>
<th>x ∧ y</th>
<th>x ∨ ¬ x</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>t</td>
</tr>
<tr>
<td>f</td>
<td>t</td>
<td>t</td>
<td>f</td>
<td>t</td>
</tr>
<tr>
<td>t</td>
<td>f</td>
<td>t</td>
<td>f</td>
<td>t</td>
</tr>
<tr>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
</tr>
</tbody>
</table>
Propositional Logic

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>\neg y \lor x</th>
<th>\neg x \lor y</th>
<th>(x \lor \neg y) \land (y \lor \neg x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>f</td>
<td>t</td>
<td>t</td>
<td>t</td>
</tr>
<tr>
<td>f</td>
<td>t</td>
<td>f</td>
<td>t</td>
<td>f</td>
</tr>
<tr>
<td>t</td>
<td>f</td>
<td>t</td>
<td>f</td>
<td>f</td>
</tr>
<tr>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
</tr>
</tbody>
</table>

\neg x \lor y \text{ is equivalent to } x \rightarrow y

(\neg x \lor y) \land (\neg y \lor x) \text{ is equivalent to } x \leftrightarrow y
Propositional Logic

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>y→x</th>
<th>x→y</th>
<th>x↔y</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>f</td>
<td>t</td>
<td>t</td>
<td>t</td>
</tr>
<tr>
<td>f</td>
<td>t</td>
<td>f</td>
<td>t</td>
<td>f</td>
</tr>
<tr>
<td>t</td>
<td>f</td>
<td>t</td>
<td>f</td>
<td>f</td>
</tr>
<tr>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
</tr>
</tbody>
</table>

The statement $x \rightarrow y$ is true unless $x$ is true and $y$ is false. In particular, it's true even when $x$ is false and $y$ is false.

The statement $x \rightarrow y$ is a claim that "$x$ being true forces $y$ to be true". That claim can itself be either true or false. The claim does not say what happens when $x$ is false.
Propositional Logic

- \( x \rightarrow y \)
  - read as "x implies y" or "if x, then y"

- \( x \leftrightarrow y \)
  - means "x and y have the same truth values"; they are always in agreement.
  - read as "x if and only if y" or "x iff y" or "x is equivalent to y"

Examples; all statements below are true

- \( 5 + 7 = 12 \rightarrow 5^2 = 25 \)
- username unknown \( \rightarrow \) login denied
- password incorrect \( \rightarrow \) login denied
- login denied \( \not\rightarrow \) password incorrect
- \( 2 + 2 = 5 \rightarrow 5^2 = 25 \)
- \( 2 + 2 = 5 \leftrightarrow 5^2 = 10 \)
These implications capture *coincidence*, not necessarily *causality*, but not necessarily *mere coincidence* either.

We'll use double lined arrows $\Rightarrow$ to emphasize the causality part of the relationship.

- Usually when our statements concern variables
- Example: $x > 5 \Rightarrow x^2 > 25$ ($\Rightarrow$ instead of $\rightarrow$)

Speaking of which...
Predicate Logic (With Quantifiers)

- A predicate takes some inputs and is either true or false once the inputs are specified
  - $P(x,y) = x \land \neg y$
  - $Q(x) = x^2 < 27$
    (the types of the inputs should be explicitly or implicitly specified)
"For all" – universal quantifier – \( \forall \)

- \( \forall x \, P(x) \) means that for every possible \( x \), \( P(x) \) is true.
- Once \( P \)'s behavior is known and the universe of possible values of \( x \) is known, the statement \( \forall x \, P(x) \) is either true or false.
- Example: \( \forall x \, x^2 < 27 \) is false when \( x \) ranges over the elements of \( \mathbb{N} \).
- Counterexample: \( 6^2 \not< 27 \)
- It is true when \( x \) ranges over \( \{1, 2, 3, 4, 5\} \).
- Can prove by checking each \( x \).
"There exists" – existential quantifier – $\exists$

- $\exists x \ P(x)$ means that $P(x)$ is true for one or more possible values of $x$
- Once $P$'s behavior is known and the universe of possible values of $x$ is known, the statement $\exists x \ P(x)$ is either true or false
- Example: $\exists x \ x^2 - (x-1)^2 > 27$ is true when $x$ ranges over the elements of $\mathbb{N}$
Combining Quantifiers

- $(\forall x \in \mathbb{N}) (\exists y \in \mathbb{N}) \ y = 3x$ ?
- $(\forall x \in \mathbb{N}) (\exists y \in \mathbb{N}) \ 3y = x$ ?
- $(\exists e \in \mathbb{R}) (\forall x \in \mathbb{R}) x \cdot e = x$ ?
- $(\forall x \in \mathbb{R}) (\exists i \in \mathbb{R}) x \cdot i = 1$ ?
- $\neg [(\exists x \in \mathbb{Q}) \ x \cdot x = 2]$ ?
- $(\forall x \in \mathbb{Q}) \ x \cdot x \neq 2$ ?

- You can't prove a $\forall$ statement over an infinite set by enumerating cases; you have to use a different argument
Relations

- A relation is a predicate that takes two (or more) inputs

Examples

- "<" between two elements of \( \mathbb{N} \)
- \( r^2 = x^2 + y^2 \). The relation is "the points \( x, y \) lie on a circle of radius \( r \) centered at the origin", on three elements of \( \mathbb{R} \)
- Relations need not be specified by a formula, and they need not be infinite
  - \( \otimes = \{ (1,2), (2,1), (5,4) \} \)
  - \( x \otimes y \Leftrightarrow \) the program \( x \) always takes longer to run than the program \( y \)
- The numbers \( p, q \in \mathbb{R} \) are related if \( p/q \) is a power of 2

- If some relation doesn’t have a standard syntax (like the last example), we invent a benign name for it like \( \sim \) and use infix notation:
  - \( 3 \sim 6 \) but \( \neg (6 \sim 9) \) under that definition of \( \sim \)
Statements about Binary Relations

☐ A relation $\sim$ is **reflexive** if this statement is true:
  $\forall x \ x \sim x$

☐ A relation $\sim$ is **symmetric** if:
  $\forall x,y \ x\sim y \Rightarrow y\sim x$

☐ A relation $\sim$ is **antisymmetric** if:
  $\forall x,y \ [ (x\sim y \land y\sim x) \Rightarrow x=y ]$

☐ A relation $\sim$ is **transitive** if:
  $\forall x,y,z \ [ (x\sim y \land y\sim z) \Rightarrow x\sim z ]$

☐ A relation $\sim$ is an **equivalence relation** if $\sim$ is reflexive, symmetric, and transitive

(see example in 2009 quiz solutions)
Examples of Binary Relations

- \leq \text{ over } \mathbb{N}
- < \text{ over } \mathbb{N}
- a \sim b \text{ meaning } a^2 = b^2 \text{ over } \mathbb{Z}
- a \triangleleft b \text{ meaning } |a - b| < 3 \text{ over } \mathbb{N}
- \otimes \text{ over } \mathcal{R}
- \circledast \text{ over the set of all C++ programs}
Relevance

- We will work with some relations having to do with how computation happens
- We will often work to discover the truth or falsity of statements that use $\forall$ and/or $\exists$ quantifiers

Example: if A and B are sets, then these three statements each say exactly the same thing:

1. $A=B$
2. $(A \subseteq B) \land (B \subseteq A)$
3. $[\forall x \in A \rightarrow x \in B] \land [\forall x \in B \rightarrow x \in A]$


Functions

- $f : A \rightarrow B$ is a statement saying
  “$f$ is a function that maps $A$ to $B$”
  - inputs are in $A$ (domain), outputs are in $B$ (range)
    If $x \in A$, then $f(x)$ is the associated element of $B$
  - $\forall x \in A \ \exists y \in B \ f(x) = y$
    “every input produces some output”

- The function consists of both the type statement $f : A \rightarrow B$ and the actual associations
  - $\rightarrow$ does not mean “implies” in this notation
Functions

- \( g : \mathbb{N} \rightarrow \mathbb{R} \) via \( g(x) = x^{1/3} \)
- \( h : \mathbb{N} \rightarrow \{\text{true, false}\} \) via

\[
h(x) = \begin{cases} 
\text{true} & \text{if } x \text{ and } x + 2 \text{ are both prime} \\
\text{false} & \text{otherwise}
\end{cases}
\]

- \( f : \{1, 2\} \rightarrow \mathbb{R} \) via

\[
f(1) = \pi
f(2) = -37
\]

- Note that functions don’t have to be specified or even specifiable “by formula”
One-to-one and Onto Functions

- $f : A \rightarrow B$ is one-to-one if
  \[ \forall x, y \in A \ [ x \neq y \rightarrow f(x) \neq f(y) ] \]

- $f : A \rightarrow B$ is onto if
  \[ \forall y \in B \ \exists x \in A \ f(x) = y \]

- $f, g, h$ on previous page?
An **alphabet** is a *finite* set, usually called $\Sigma$
- Example: $\Sigma = \{ \text{a, b, c} \}$
- Example: $\Sigma = \{ 0, 1 \}$

A **string** is a *finite* ordered sequence of zero or more characters from an alphabet
- Example: abcabab

**Empty string**: $\varepsilon$ (epsilon)
- The *unique* string with length 0
- Think of this as ""
- Some books use the symbol $\lambda$ instead of $\varepsilon$
- Note that $\emptyset$ is not a string at all
Operations on Strings

- Concatenation
  - $0101 \cdot 11 = 010111$
  - Sometimes written without '\\cdot '\n    - Particularly with variables
    - Example: if $x$ and $y$ are strings then $xy = x \cdot y$
  - $\varepsilon$ is the identity for this operation
    - For every string $x$, $x \cdot \varepsilon = \varepsilon \cdot x = x$
    - Thus $11 = \varepsilon \varepsilon 1 \varepsilon 1$
  - Note that concatenation does not mark the place where the two strings are joined
    - $0 \cdot 11 = 01 \cdot 1 = 011 \cdot \varepsilon$
Operations on Strings

- **Exponentiation. Inductive definition**
  - **Basis:** For every string $x$,
    \[ x^0 = \varepsilon \quad \text{(not} = 1) \]
  - **Induction step:** if $x$ is a string and $n \geq 0$ is a whole number, then
    \[ x^{n+1} = x \cdot x^n \]
  - Exponents may *only* be whole numbers.
    \[ x^{1.5} \text{ is undefined} \]

- $(001)^3 = 001001001$
  - Parentheses for grouping only

- $\varepsilon^5 = ?$
Operations on Strings

- **Length.** The *length* of a string $x$ is the number of characters in $x$ and is written $|x|$
  - $|0101| = |ε0101εε| = 4$ (ε is not a character)
  - $|ε| = 0$
  - Remember that $|A|$ has a different meaning when $A$ is a set

- **Reversal.** Inductive definition
  - **Basis:** $ε^R = ε$
  - **Induction step:** if $x$ is a string and $c$ is a character, then $(xc)^R = ?$

- $(011011)^R = 110110$
Languages

A **language** is a set of strings. Suppose $\Sigma=\{a,b\}$. Examples:
- $A = \emptyset$ (the empty language)
- $B = \{abba, babb, \varepsilon, aaaaaaaaaaaaaaaaaaaaa\}$
- $C = \{ x \mid x \text{ contains an even number of \textquote{a} } \}$
  
  = ?
- Note $\emptyset \neq \{ \varepsilon \} !!$

Convention: we typically use lower-case letters ($x, y, z$) for string variables and upper-case letters ($A, B, C$) for language variables
Operations on Languages

- Concatenation
  - $A \cdot B = \{ x \cdot y \mid x \in A \land y \in B \}$
  - Similar to Cartesian product, but not same
  - $A = \{ 0, 001 \}$ and $B = \{ 01, \varepsilon \}$
  - $A \times B = \{ (0,01), (0,\varepsilon), (001,01), (001,\varepsilon) \}$
  - $A \cdot B = ?$
  - ( $|A \cdot B|$ is not necessarily $|A| \times |B|$ )
  - $A \cdot \emptyset = ?$ ( $\cdot \emptyset ???$ )
Operations on Languages

- **Reversal**
  \[ A^R = \{ \ x^R \ | \ x \in A \} \]

- **Exponentiation. Inductive definition**
  - **Basis**: if A is a language, then \(A^0 = \{ \varepsilon \}\)
  - **Induction step**: If A is a language and \(n \geq 0\) is a whole number, then
    \[ A^{n+1} = A \cdot A^n \]

- **A = ∅; \ A^3 = ?**
- **B = \{ aab, bb \}; B^2 = ?**
- **C = \{ x \ | \ x \ contains \ an \ even \ number \ of \ ‘a’s\} \ C^3 = ?**
Operations on Languages

* (Kleene Star) [defined on p. 44 as “star”)

- If A is a language, then
  \[ A^* = \bigcup_{i=0}^{\infty} A^i \]

- **Very important operation**
- Think of \( A^* \) as “set of all concatenations of zero or more things from A”

- B = \{ aab, bb \}; \( B^* = ? \)

- Note that * is an operator on languages, not strings (for now)
- But that exponentiation applies to both

- Subtlety: \( \emptyset^* = \{ \varepsilon \} \) because we defined \( A^0 = \{ \varepsilon \} \)
Important Idiom: $\Sigma^*$

- Every alphabet $\Sigma$ is a finite set of 1-character strings, so $\Sigma$ is automatically a small language.
- So $\Sigma^*$ means “set of all concatenations of zero or more things from $\Sigma$”.
  - In other words: set of all strings formed from the alphabet $\Sigma$.

- $\Sigma = \{a\}; \quad \Sigma^* = \{\varepsilon, a, aa, aaa, aaaa, \ldots\}$
- $\Sigma = \{0,1\}; \quad \Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$
  - **Lexicographical order** is often convenient: shortest strings first, then sorted by dictionary order.
Important Idioms

☐ Equivalent statements
  ■ Let x be a string
  ■ Let x ∈ Σ*

☐ Equivalent statements
  ■ Let B be a language
  ■ Let B ⊆ Σ*

■ Alternative: Let B … ?
  ■ \( \mathcal{P}(\Sigma^*) \) is the set of all languages over \( \Sigma \)
  ■ Recall \( \mathcal{P}(\Sigma^*) \) denotes power set of \( \Sigma^* \)
Orientation

- Strings will be the *inputs*, *outputs*, and *source codes* of our programs.
- Languages will be the “birds-eye views” of the overall behavior of particular programs.
  - Each language will include particular strings of interest and exclude others.
  - The language might be a *specification* of what some program is desired to do or it might be a *description* of what a program actually does.
- For human communications:
  - Strings = sentences or utterances.
  - Language = set of those strings that make sense.
  - Program = a person who speaks the language.
More Language Examples

Let $L_1 = \{ x \in \{0,1\}^* : |x| \text{ is a multiple of 3} \}$

Let $\Sigma$ be the ASCII alphabet and:

- Let $L_2 = \{ p \in \Sigma^* : \text{gcc does not report syntax errors when compiling } p \}$
- Let $L_3 = \{ p \in \Sigma^* : p \text{ is a syntactically correct C program} \}$

We might hope that $L_2 = L_3$

Let $L_4 = \{ p \in L_2 : p \text{ eventually prints “Hello” when you run it} \}$

$L_4$ is uncomputable (we’ll see why later in the semester!)
A **language class** $\mathcal{C}$ over an alphabet $\Sigma$ is a set of languages over $\Sigma$

- The class of *all* languages over $\Sigma$ is $\mathcal{P}(\Sigma^*)$
- So $\mathcal{C} \subseteq \mathcal{P}(\Sigma^*)$

$\emptyset$

$\text{FIN} = \{ A \subseteq \Sigma^* : |A| = 0 \lor |A| \in \mathbb{N} \}$

- The class of all finite languages

$\text{ALL} = \mathcal{P}(\Sigma^*) = \{ A \mid A \subseteq \Sigma^* \}$

- The class of all languages

- Human communication version: the class of Indo-European languages, or the class of Romance languages
Picture so far

Each point is a language in this Venn diagram

\[ \text{ALL} = \mathcal{P}(\Sigma^*) \]
WARNING

- Students often confuse strings, languages, and classes of languages
- Every time you encounter an object you need to (correctly) know which type it is supposed to be
  - If you are working on the wrong plane, nothing will make sense at all
- Remember:
  - string: what a program is computing with at one moment; strings are always finite
  - language: a characterization of the program’s overall behavior; languages are often infinite
  - class: a characterization of computational power; what “these type of programs” are able to do; classes are usually infinite
91.304 Foundations of (Theoretical) Computer Science

G. Pecelli

giam@cs.uml.edu
Proofs

- **Proof Techniques**
  - **Construction** - best: gives an algorithm for producing the desired result.
  - **Contradiction** - least informative: no algorithm (usually).
  - **Induction** - intermediate: usually results in a recursive procedure; usually refers to countably infinite sets.
Proofs: Construction

A theorem will be a statement of the form:

- Premise $\Rightarrow$ Conclusion

We start from the Premise, and using its statements and what else we know to be true about the kind of object we are studying, we construct, in a finite number of steps, an object satisfying the Conclusion.
Proofs: Construction

**Def.:** Let $G = (V, E)$ be a graph; we say $G$ is $k$-regular if every node of the graph has degree $k$.

**Theorem 0.22.** For each even number $n > 2$, $\exists$ a 3-regular graph with $n$ nodes.

**Proof:** for each even $n > 2$, we will give an effective procedure that produces such a graph.
Proofs: Construction

Proof: details. Let \( n \) be even, \( n > 2 \). \( G = (V, E) \) is constructed as follows: \( V = \{0, 1, \ldots, n-1\} \) (name \( n \) vertices - this is the easy part). We now construct the edges:

\[
E = \{\{i, i+1\} \mid 0 \leq i \leq n-2\} \cup \{\{n-1, 0\}\} \\
\cup \{\{i, i+n/2\} \mid 0 \leq i \leq n/2-1\}.
\]

Observe that the first two unions give us a cycle connecting all the vertices: they thus have degree 2 at this point. The third union adds edges connecting "antipodal" vertices - adding one degree to each for a total of 3.

QED
Proofs: Construction

- **Note**: the construction need not be easy or obvious (most of the time it isn't).

- Stare at the problem, think about it, draw pictures (if possible), doodle... eventually something may happen (no guarantee).

- The only way you get reasonably proficient at concocting proofs of (simple) theorems is to keep trying (there is no effective procedure for coming up with effective procedures...)


Proofs: Contradiction

- Sometimes, no matter how hard we try, we cannot come up with a construction, and we can't come up with a counterexample (which would prove that our hoped-for theorem is not true).

- Simple logic gives us the equivalence of the two implications: $P \Rightarrow Q$ and $\neg Q \Rightarrow \neg P$.

- **Possible solution**: try to prove $\neg P$ by assuming $\neg Q$. Since the two implications are equivalent, success will give you what you want.
Proofs: Contradiction

**Def.** a number is **rational** if it is a fraction of two integers: \( m/n \). It is **irrational** if it is **not** rational

**Theorem 0.24:** \( \sqrt{2} \) is irrational.

**Proof:** generally, you should not be able to "construct a negative". The implication you want to prove is:

if the properties of rational numbers are true, then \( r = \sqrt{2} \) is irrational.

Its contrapositive (equivalent) is

if \( r = \sqrt{2} \) is **not irrational** (i.e., rational) then \( r \) must have some impossible properties.
Proofs: Contradiction

**Proof**: details. If $\sqrt{2}$ is rational, then $\sqrt{2} = m/n$ for some pos. integers $m, n$. We can also assume that $m$ and $n$ are relatively prime (= they have no common divisors - if they do, just divide them out of both). This implies that (at least) one of the two is odd. Multiply both sides by $n$ and square: $2n^2 = m^2$. This implies that $m^2$ is even, which, in turn, implies that $m$ is even (the product of two odd numbers is always odd). This implies that $m = 2k$, for some pos. int. $k$. 
Proofs: Contradiction

We can re-write the equation as

\[ 2n^2 = m^2 = (2k)^2 = 4k^2; \]

dividing both sides by 2 gives \( n^2 = 2k^2 \), and this can be used to show that \( n \) is also even. So \( m \) and \( n \) have a common divisor, 2, contradicting the assumption that they are relatively prime.
Proofs: Induction

You are in front of a staircase to Heaven (it's that long...). How can you prove that you can climb it as far up as you need?

An induction proof goes roughly like this:

1. Prove that you can get one foot on the first step.
2. Prove that, if you have a foot on any step, you can get a foot on the next step.
Proofs: Induction

Finding something to prove...

You are looking for a formula (a predicate, an equation) that depends on natural numbers. It is often easy to find out what the formula "looks like" for small values of the natural number, and it may be a little harder to "guess" at the formula that should be valid for all values of the number. Once you have gone through these stages, you have the job of proving that the formula (predicate) is valid for all values of the natural number.
Proofs: Induction

Let us consider a "sum of squares":

\[ S_2(n) = \sum_{i=0}^{n} i^2 = ?? \]

- For \( i = 0 \), \( S_2(0) = 0^2 = 0 \).
- For \( i = 1 \), \( S_2(1) = 1^2 = 1 \).
- For \( i = 2 \), \( S_2(2) = 1^2 + 2^2 = 5 \).
- For \( i = 3 \), \( S_2(3) = 1^2 + 2^2 + 3^2 = 14 \).

This does not seem to lead us to anything really obvious, but we can peek at the simpler formula:

\[ S(n) = \sum_{i=0}^{n} i = \frac{n(n+1)}{2}. \]
Proofs: Induction

This suggests that a formula for our case might look like:

\[ an(bn + c)(dn + e) \]

where the degree of \( n \) is one higher than the power in the sum, and where \( a, b, c, d, e \) are rational (mostly integer) constants.

- \( S_2(0) = 0 \) puts no constraints on the coefficients.
- \( S_2(1) = 1 \) implies \( a(b + c)(d + e) = 1 \). We can start picking some choices for \( b, c, d, e \) : if we choose them all 1, then \( a = 1/4 \). If we choose \( b = 2, c = d = e = 1 \), then \( a = 1/6 \). Many other choices are possible, but we have to start somewhere...
Proofs: Induction

- $S_2(2) = 5$.
  - $a = 1/4$ gives $(1/4)2(2 + 1) (2 + 1) = 12/4 \neq 5$.
  - $a = 1/6$ gives $(1/6)2(2*2 + 1) (2 + 1) = 30/6 = 5$.

- $S_2(3) = 14$.
  - $a = 1/6$ gives $(1/6)3(2*3+ 1) (3 + 1) = 84/6 = 14$.

We can try a few more, just for luck... and we find that the formula

$$S_2(n) = \sum_{i=0}^{n} i^2 = \frac{n(2n+1)(n+1)}{6}$$

holds for everything we try...

Time to prove it...
Proofs: Induction

- Step 1: prove that I can get a foot on the first step = prove the formula holds for \( n = 0 \). This is actually trivial, since I built the formula that way. In some cases, it may be a bit harder.

- Step 2: prove that, if I can get a foot on any step, I can get a foot on the next step = prove that if the formula holds for some value \( n \), then it must hold for \( n + 1 \).
Proofs: Induction

- Assume $S_2(n) = n(2n + 1)(n + 1)/6$ for some $n$.

  - $S_2(n + 1) = \sum_{i=0}^{n+1} i^2 = \sum_{i=0}^{n} i^2 + (n + 1)^2 = [S_2(n)] + (n + 1)^2$

  - $= \frac{n(2n + 1)(n + 1)}{6} + (n + 1)^2$ induction assumption

  - $= \frac{(n + 1)(2(n + 1) + 1)(n + 1 + 1)}{6}$ simple algebra.

The final expression is the formula for $S_2(n + 1)$.

The induction proof is complete... $S_2(0)$ is given by the formula (proven separately), which implies $S_2(1)$ is given by the formula, .... QED