# CPSC 365 - Proofs

1-19-15

## Overview

- Types of Proofs
  - Trivial: something that does not need to be proved
    B is true, so $A \rightarrow B$ regardless of A
  - Direct: direct line of reasoning from $A \rightarrow B$
  - Indirect: assuming the premise A and the negation of the result B, and proving that a contradiction can be found, and the result valid.
    $A \cap \neg B \rightarrow C \cap \neg C$, and so $A \rightarrow B$

    alternately, showing a direct line of reasoning from $\neg A \rightarrow \neg B$ (contrapositive)
  - Induction: prove $B(1)$, $B(k) \rightarrow B(k+1)$ for all $k \in \mathbb{N}$, which implies that $B(n)$ for all $n \in \mathbb{N}$

  + other types of proofs exist... (36 Methods of Mathematical Proof)

- Styles of Proof
  - "Two Column Proof": think high school geometry, one column of steps, one column of more detailed reasoning ... basically, the proof of the illiterate
  - "Flow Chart / Crayon Proof": when your proof covers 10 pages and it is hard to tell if the pages are in order... you have the actual solution, but it requires a legend to decipher
  - "Paragraph Proof": well constructed, concise, everything you need!

- don't conform exclusively to one "type" of proof, but use what you need, when you need... aim for the "paragraph proof" but still use characters, notation / terminology, "display math" with proper whitespace and formatting, diagrams, etc... develop your own standard methodology

- Tools for Proofs
  - Mathematical Notation /
    - introduce and use characters / symbols for objects, data structures, and anything mathematical (discrete mathematics, set theory, graph theory)
    - use appropriate shorthand:

      $\exists, \exists!, \nexists, \in, \notin, \forall, \cap, \cup, \wedge, \vee, \times, \{\}, ()$

      $\Rightarrow, \Leftarrow, \Leftrightarrow, \rightarrow, \leftarrow$

      s.t., w.r.t., w.l.o.g.
    - set notation is convenient, think of it as a "language"
    - don't trade clarity and readability for compactness and brevity: there should be a balance between these qualities, so that you can still understand your proof next week
  - Diagrams
    - appropriate diagrams can save a paragraph of clumsy detail, or make your arguments more intuitive and powerful
    - examples:
      - set membership during some process
      - topology and evolution of graph algorithms
      - flow diagrams for complicated operations
      - structure of data structures
    - include diagrams from the problem in your solution, and use it to add your notation, characters, terminology, and even steps of the proof to give a roadmap

- Components / Bookends
  - sometimes a problem naturally breaks down into fairly distinct components, and the solution should be as well ... avoid overkill, but use these various components to "scaffold" solutions
  - components
    - claims / observations: sometimes these can be obvious or "self-proving" and sometimes they will need to be proved as well
    - lemmas: effectively minor "theorems", these are used only as stepping stones to help prove a theorem or a bigger result
    - propositions: typically statements that are used / assumed, but not proved until later for clarity and coherence of argument.
    - theorems: conditional statement where a premise or set of hypotheses are used to prove a result ... often a series of problems can be thought of as a series of theorems to be proved, when one problem builds on the result of another (and the converse
    - corollaries: statements which typically follow from another statement with little or no further proof (prove the theorem, and get this for free)

## Method

- Method
  - solving a problem / proving a result

    UNDERSTAND

    PLAN

    SOLVE ⇔ FIX

    PLAN

    WRITE

## UNDERSTAND

- rewrite the problem, or comment on the printed problem, elaborating, explaining, rephrasing, connecting, dissecting, detecting EVERY piece of information provided
- identify key components of the problem / parts of the whole
- look for the "intuition block" needed to "unlock" the proof... what is needed
- find some basic examples / counterexamples that illustrate the result

## PLAN

- plan out the steps of the solution that correspond to the components
- establish the order of working / importance of steps (don't prove the fine detail until you are confident that the overall method is the right / reasonable one)
- draw diagrams, introduce notation, characters, terminology

## SOLVE ↔ FIX

- attempt to solve each step of the PLAN, and adapt the PLAN when required
- effectively, take the intuition and the concept, and formalise the working step by step until the proof is complete or a fundamental error is discovered
- sometimes, you need to go back to PLAN and UNDERSTAND as well

## PLAN

- when the solution is coherent and all the steps are consistent, take the actual content and layout the entire proof / argument into a readable and organized narrative
- "think backwards, write forwards"

## WRITING

- write the solution as if it were another student in the class: with the same tools and background knowledge, but not necessarily with the ability to solve or understand the problem (don't assume or "gloss over" your intuition)
- aim equally for coherence and conciseness
- use characters, diagrams, terminology, etc... but use WORDS as well

- encouragements / good practice
  - layout
    - use margins, and stick to them!
    - write horizontally and NEATLY
    - don't cramp in the margins, use*
    - USE WHITESPACE
    - center or indent math, diagrams (block calculations, not EVERYTHING)
  - notation
    - respect common practice for naming entities / data structures (such as lowercase for vectors $x, y, z$, uppercase for matrices $A, W$, nodes are $u, v \in V$, edges are $e = (u,v) \in E$, indices are $i, j, k$, $n = |V|$, $m = |E|$, graphs $G$, trees $T$, etc...)
    - use sequential numbering for equations if needed
  - general
    - aim for a "paragraph proof"
    - plan really well
    - develop a concept of "mathematical elegance" (Hardy's Apology (?))
    - learn from examples (particularly Dan Spielman's solutions, no doubt)
    - "Art of Problem Solving" is pretty good
    - when you have proved something, TRY TO BREAK IT
    - "Everything should be made as simple as possible, but no simpler" (Einstein, probably)

+ "D-day for writing a proof" (Slide)

*a redirect if needed

# Problem Set 0

## Direct Proof (BAD)

(2), (3) => (1)

T has $n-1$ edges, and is connected. Consider some $u \in V$. As T is connected, there exists some $v \in V$ such that $(u,v) \in E$. Let $C = u$, where $C \subseteq V$. Then we can add $v$ to $C$ such that $C = C \cup \{v\}$, where $C \subseteq V$, E restricted to C or $E_c$ contains only one edge $(u,v) \in E_c$ such that $|E_c| = 1$, and no cycles are introduced as the edge added corresponds to the node added, and is a bridge (where an edge cannot be a bridge and part of a cycle at the same time). Now, recursively, there exists some $w \in V$ such that $w \notin C$ (unless $|V \setminus C| = 0$ and $C = V$) and for some $x \in C$ there exists $(x,w) \in E$. Hence, we add $w$ to $C$ such that $C = C \cup \{w\}$, where $C \subseteq V$, $E_c$ now contains only one more edge $(x,w)$ and $|E_c| \mapsto |E_c| + 1$, and no cycles were introduced. As $n = |V|$, we can only add $n-1$ nodes to C until $C = V$, and so $|E_c| = n-1$. Hence $(C, E_c) = (V, E) = T$ and T has no cycles (as no cycles were introduced).

## Indirect Proof (GOOD)

(2), (3) => (1)

From (2) and (3), $T = (V,E)$ has $|E| = n-1$ edges, and is connected.

Suppose that T has a cycle as well. Then $\exists\, e \in E$ that is not a bridge, such that $G = (V, E \setminus e)$ remains connected. $|E \setminus e| = n-2$ is trivial. However, as G is still connected, $|E \setminus e| \geq |V| - 1 = n-1$, which is a contradiction.

Thus, T has no cycles, and (1) is proved.

## Induction Proof (Method)

(1), (3) => (2)

From (1) and (3), $T$ has no cycles and is connected.

First, $|E| \geq n-1$ as $T$ is connected. We will use induction to prove $|E| \leq n-1$ as $T$ has no cycles (and is connected). Let $S(n)$ be the logical statement that "a directed, acyclic graph $T$ with $|V| = n$ has at most $|E| \leq n-1$."

$S(0)$ : Trivial.

$S(1)$ : $n = 1$, and $V = \{v\}$. Then $E = \{ \}$ and $|E| = 0 = 1-1$.

$S(k)$ : Assume $S(k)$.

Consider a directed, acyclic graph $G = (V, E)$ with $|V| = k+1$. For some $v \in V$ let $H$ be the induced subgraph on $V \setminus \{v\}$, which has $t \geq 1$ connected components $H_i = (V_i, E_i)$ for $i = 1 \ldots t$. Each component is connected and acyclic (inherited from $G$) and $|V_i| \leq k$ for $i = 1 \ldots t$, and $S(k)$ implies that $|E_i| \leq |V_i| - 1$. Noting that

$$\sum_{i=1}^{t} |V_i| = k = |V| - 1$$

it follows that

$$\sum_{i=1}^{t} |E_i| \leq k - t$$

and as at most $t$ edges were removed with $v$ as $G$ is acyclic and connected $|E| \leq k - t + t = k$ as well.

Thus, $S(k)$ implies $S(k+1)$.

Hence, it follows by induction that $S(n)$ is true for all $n \in \mathbb{N}$, and (2) is proved.