CS 157 – FITCH TRICKS

Stefan Swaans

Logical Entailment and Provability

A set Δ of premises *logically entails* a conclusion φ (written $\Delta \models \varphi$) if and only if every truth assignment that satisfies Δ also satisfies φ.

A conclusion φ is said to be *provable* from a set Δ of premises (written $\Delta \vdash \varphi$) if and only if there is a finite proof of φ from Δ.

Soundness and Completeness

Soundness: A proof system is *sound* if and only if every provable conclusion is logically entailed.

$$(\Delta \vdash \varphi) \text{ implies } (\Delta \models \varphi)$$

Completeness: Our proof system is *complete* if and only if every logically entailed conclusion is provable.

$$(\Delta \models \varphi) \text{ implies } (\Delta \vdash \varphi)$$

General Tricks

- The premises are usually given for a reason — use them!

- Every time you create an assumption, you need to **know what implication you want to prove.**
  - An implication is the *only* thing we can get from assuming something! Assumptions are *not useful for anything else.*

General Tricks (cont.)

- Sometimes it helps to slow down and actually process what you are trying to prove.
  - This can help you to gain an intuition for the correct approach to the problem.

- If all else seems to fail, try revisiting your initial assumption. Is there a different path you could take to prove your goal?

Recursion Mindset

- It may help to view proofs **recursively**.
  - Say we want to prove A
  - In order to prove A, we must prove B and C
  - In order to prove B, we must prove X
  - In order to prove C, we must prove Y
  - Etc…
- View each step of the recursion as a subproblem.
  - "Given **all sentences in all of your superproofs**, prove X."
  - Sometimes you may have to pull from a distant superproof!

Implication Tricks

- **If your goal is X=>Y, always start by assuming X!**
  - Example: Given q, show p=>q
- If you're given X=>Y and your goal is Y, show X.
  - Example: Given (p=>q)=>r and q, show r
- You can always show X=>X for any X.
  - Example: Show p=>p

Negation Tricks

- If you want to prove X and X doesn't seem to be "readily available" anywhere in your premises or superproofs, very likely you should **assume ~X**.
  - Example: Given p and ~p, show q
- When you assume ~X, **remember the goal is to show ~X=>Y and ~X=>~Y for some Y. (Negation Intro.)**
- Picking Y can sometimes be one of the hardest parts of the proof. If you can, try to keep Y as simple a sentence as possible. If Y or ~Y is already proven/given, that's a good candidate.
  - Disclaimer: There are some (difficult) proofs in which the correct Y is not obvious!

Negation Tricks (cont.)

- If you ever have X and ~X, you can prove **anything!**
  - Useful for creating Negation Introductions!
  - Example: Given p and ~p, show (q=>r=>z)&((r|s)=>q))
- If you have X=>Y and ~Y, you can easily prove ~X.
  - Useful for creating Negation Introductions!
  - Example: Given p=>q and ~q, show ~p

Or Tricks

- When dealing with some sentence containing X|Y, **you'll probably need to use Or Introduction and/or Or Elimination!**
  - Example: Given p=>q, show p=>q|s
- Or Introduction is very powerful when combined with Negation Introduction.
  - When you have ~(X|Y), you can show ~X and ~Y.
  - Example: Given ~(p|q), show ~p

Or Tricks (cont.)

- How do we know whether to consider Or Elimination or Or Introduction?
  - Generally, if we're trying to *prove* some X|Y, look to use Or Introduction.
  - Generally, if we have some X|Y and want to prove *something else*, look to use Or Elimination.

And Tricks

- If you ever have some X&Y, there is no harm in using And Elimination to get X and Y. They are probably useful on their own.