Review Session 3

Equivalence, consistency, entailment

A sentence **φ** is **logically equivalent** to a sentence **ψ** if and only they have the same value for **every** propositional interpretation.

A sentence **φ** is **consistent** with a sentence **ψ** if and only if there is **a truth assignment** that satisfies **both φ** and **ψ**.

A premise **φ** **logically entails** a conclusion **ψ** (written as $\varphi \vDash \psi$) if and only if **every interpretation that satisfies φ also satisfies ψ**.

Which table shows **logical equivalence**?

<table>
<tr><th>p => q</th><th>~p | p & q</th></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td>1</td><td>1</td></tr>
</table>

Table A

p => q	~q => p
1	1
0	1
1	1
1	0

Table B

Which table shows logical equivalence?

p => q	~p	p & q
1	1
0	0
1	1
1	1

Table A

p => q	~q => p
1	1
0	1
1	1
1	0

Table B

Based on the truth tables, what does ~p logically entail?

Premises	
~p	p => q
0	1
0	0
1	1
1	1

p => q

Premises	
~p	~p & ~q
0	0
0	0
1	0
1	1

~p & ~q

Based on the truth tables, what does ~p logically entail?

Premises	
~p	p => q
0	1
0	0
1	1
1	1

p => q

Premises	
~p	~p & ~q
0	0
0	0
1	0
1	1

~p & ~q

Which set of sentences is logically consistent?

Constants		Premises	
p	q	~p	~p & ~q
1	1	0	0
1	0	0	0
0	1	1	0
0	0	1	1

Table A

Constants		Premises	
p	q	~p	~p | ~q
1	1	0	0
1	0	0	1
0	1	1	1
0	0	1	1

Table B

Which set of sentences is **logically consistent**?

Constants		Premises	
p	q	~p	~p & ~q
1	1	0	0
1	0	0	0
0	1	1	0
0	0	1	1

Table A

Constants		Premises	
p	q	~p	~p | ~q
1	1	0	0
1	0	0	1
0	1	1	1
0	0	1	1

Table B

Soundness & completeness

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

$$\text{If } \Delta \vdash \phi \text{, then } \Delta \vDash \phi.$$

A proof system is **complete** if and only if every logical conclusion is provable.

$$\text{If } \Delta \vDash \phi \text{, then } \Delta \vdash \phi.$$

Functional logic

Functional logic is **sound** but **not complete**.

This implies...

- For all the proofs we wrote, if we prove sentence ϕ from a set of premises Δ, we can say $\Delta \vDash \phi$.
- There are some sentence ψ that can be logically concluded from Δ ($\Delta \vDash \psi$), but we **do not** have a finite proof for it.

Terms VS Sentences

A **term** is either a variable, an object constant, or a functional term

Terms represent objects/nouns. A term **cannot** have a **true/false value.**

Relational **sentences** are not **terms** and **cannot** be **nested** in terms or relational sentences.

Term or sentence?

x is a variable; p, q are unary relation constants; f is a function constant

1. x
2. p(x)
3. f(x)
4. p(f(x))
5. f(f(x))
6. p(x) & q(f(x))

Term or sentence?

x is a variable; p, q are unary relation constants; f is a function constant

1. x - **term.**
2. p(x) - **sentence.**
3. f(x) - **term.**
4. p(f(x)) - **sentence.**
5. f(f(x)) - **term.**
6. p(x) & q(f(x)) - **sentence.**

Term or sentence?

x is a variable; p, q are unary relation constants; f is a function constant

1. x - **term.** Variables are terms.
2. p(x) - **sentence.** p(x) is a relational sentence, since p is a relation constant.
3. f(x) - **term.** Functional expressions are terms.
4. p(f(x)) - **sentence.** A relation constant applied to a functional term is a sentence.
5. f(f(x)) - **term.** A function constant applied to a functional term is a term.
6. p(x) & q(f(x)) - **sentence.** A logical expression combining relational sentences is a sentence.

Which type of induction should I use?

Look at the premises!

1 object constant, 1 **unary** function constant => **Linear induction**

1 object constant, 2 **unary** function constants => **Tree induction**

1 object constant, 1 **binary** function constant => **Structural induction**

Which type of induction should I use?

Look at the premises! (generalized form)

1 unary function constant => **Linear induction**

n unary function constants => **Tree induction**

n n-ary function constant => **Structural induction**

Inductive Hypothesis

Base Case

Inductive Conclusion

$$\frac{\phi[a] \qquad \forall x.(\phi[x] \Rightarrow \phi[s(x)])}{\forall x.\phi[x]}$$

Inductive Case

Overall Conclusion

Induction proofs

For linear, tree, and structural induction proofs, you need the **base case** (phi(a)) and then **one or more inductive cases** (inductive hypothesis => inductive conclusion)

We know that **to prove an implication,** we need to **begin with an assumption**. But what should we assume?

- Look at the **desired conclusion**: Av: phi(v), where phi(v) is a sentence
- For the **inductive case**, we need **two things**:
  - An **implication** e.g. (phi(mu) => phi(s(mu))
  - A **universal quantifier**
- To get the implication, we need to **make an assumption** (assume inductive hypothesis)
- To be able to add the universal quantifier, we need to **use placeholders**
- Therefore, you should **assume phi([placeholder]),** where AX:phi(X) is the goal

How to do a: linear induction proof

1. You know what you want to prove: phi(v)
2. The inductive case you need is phi(mu) => phi(s(mu))
3. To do this, assume phi(mu) and then prove phi(s(mu)) from the premises & rules of inference
4. Remember to use placeholders in your assumptions, so that you can add the universal at the end!

Linear Induction

$$\frac{\phi[a] \qquad \forall \mu.(\phi[\mu] \Rightarrow \phi[s(\mu)])}{\forall \nu.\phi[\nu]}$$

How to do a: tree induction proof

1. You know what you want to prove: phi(v)
2. This means you need two sub-proofs:
    a. phi(mu) => phi(f(mu))
    b. phi(mu) => phi(g(mu))
3. For each sub-proof, assume phi(mu) and then prove the desired consequent
4. Remember to use placeholders in your assumptions, so that you can add the universal at the end!

Tree Induction

$$\frac{\begin{array}{l}\phi[a] \\ \forall \mu.(\phi[\mu] \Rightarrow \phi[f(\mu)]) \\ \forall \mu.(\phi[\mu] \Rightarrow \phi[g(\mu)])\end{array}}{\forall \nu.\phi[\nu]}$$

How to do a: structural induction proof

1. Once again, we need a sub proof that will prove our implication. This time, it's more complex since we're dealing with multiple variables
2. To prove the implication, we need to assume phi(lambda) & phi(mu)
3. This time, you'll need to use two different placeholders, since you're trying to add two different universal quantifiers

Structural Induction

$$\frac{\phi[a] \quad \phi[b] \quad \forall\lambda.\forall\mu.((\phi[\lambda] \wedge \phi[\mu]) \Rightarrow \phi[h(\lambda,\mu)])}{\forall\nu.\phi[\nu]}$$

Proofs

(2019 final Q4)

1. $\forall x.(p(x) \rightarrow q(f(x)))$
2. $\forall y.(q(y) \rightarrow p(y)))$
3. $\forall z.p(a)$

Goal: $\neg \exists x.(\neg p(x) \ \land \ \neg q(x))$

Objects	a
Functions	f

Select All

1.	AX:(p(X) => q(f(X)))	Premise
2.	AY:(q(Y) => p(Y))	Premise
3.	AZ:p(a)	Premise
4.	p(a)	Universal Elimination: 3
5.	p([c]) => q(f([c]))	Universal Elimination: 1
6.	q([c]) => p([c])	Universal Elimination: 2
7.	p([c])	Assumption
8.	q(f([c]))	Implication Elimination: 5, 7
9.	q(f([c])) => p(f([c]))	Universal Elimination: 2
10.	p(f([c]))	Implication Elimination: 9, 8
11.	p([c]) => p(f([c]))	Implication Introduction: 7, 10
12.	AX:(p(X) => p(f(X)))	Universal Introduction: 11
13.	AX:p(X)	Induction: 4, 12
14.	EX:(~p(X) & ~q(X))	Assumption
15.	AX:p(X)	Reiteration: 13
16.	EX:(~p(X) & ~q(X)) => AX:p(X)	Implication Introduction: 14, 15
17.	EX:(~p(X) & ~q(X))	Assumption
18.	~p([c]) & ~q([c])	Assumption
19.	AX:p(X)	Assumption
20.	~p([c])	And Elimination: 18
21.	AX:p(X) => ~p([c])	Implication Introduction: 19, 20
22.	AX:p(X)	Assumption
23.	p([c])	Universal Elimination: 22
24.	AX:p(X) => p([c])	Implication Introduction: 22, 23
25.	~AX:p(X)	Negation Introduction: 24, 21
26.	~p([c]) & ~q([c]) => ~AX:p(X)	Implication Introduction: 18, 25
27.	AX:(~p(X) & ~q(X) => ~AX:p(X))	Universal Introduction: 26
28.	~AX:p(X)	Existential Elimination: 17, 27
29.	EX:(~p(X) & ~q(X)) => ~AX:p(X)	Implication Introduction: 17, 28
30.	~EX:(~p(X) & ~q(X))	Negation Introduction: 16, 29

Proofs

(2017 final Q4 / Exercise 13.5)

1. $p(a)$
2. $\forall x.(p(x) \Rightarrow q(f(x)))$
3. $\forall x.(q(x) \Rightarrow p(f(x)))$

Goal: $\forall x.(p(x) \vee q(x))$

Objects	a
Functions	f

☐	Select All	
☐ 1.	p(a)	Premise
☐ 2.	AX:(p(X) => q(f(X)))	Premise
☐ 3.	AX:(q(X) => p(f(X)))	Premise
☐ 4.	p(a) | q(a)	Or Introduction: 1
☐ 5.	p([c]) | q([c])	Assumption
☐ 6.	p([c])	Assumption
☐ 7.	p([c]) => q(f([c]))	Universal Elimination: 2
☐ 8.	q(f([c]))	Implication Elimination: 7, 6
☐ 9.	p(f([c])) | q(f([c]))	Or Introduction: 8
☐ 10.	p([c]) => p(f([c])) | q(f([c]))	Implication Introduction: 6, 9
☐ 11.	q([c])	Assumption
☐ 12.	q([c]) => p(f([c]))	Universal Elimination: 3
☐ 13.	p(f([c]))	Implication Elimination: 12, 11
☐ 14.	p(f([c])) | q(f([c]))	Or Introduction: 13
☐ 15.	q([c]) => p(f([c])) | q(f([c]))	Implication Introduction: 11, 14
☐ 16.	p(f([c])) | q(f([c]))	Or Elimination: 5, 10, 15
☐ 17.	p([c]) | q([c]) => p(f([c])) | q(f([c]))	Implication Introduction: 5, 16
☐ 18.	AX:(p(X) | q(X) => p(f(X)) | q(f(X)))	Universal Introduction: 17
☐ 19.	AX:(p(X) | q(X))	Induction: 4, 18

Proofs

(Exercise 13.4)

1. $p(a)$
2. $\forall x.\forall y.(p(x) \mid p(y) \Rightarrow q(h(x,y)))$

Goal: $\forall x.(p(x) \vee q(x))$

Objects a

Functions h

	Select All	
1.	p(a)	Premise
2.	AX:AY:(p(X) | p(Y) => p(h(X,Y)))	Premise
3.	p([c]) & p([d])	Assumption
4.	p([c])	And Elimination: 3
5.	p([c]) | p([d])	Or Introduction: 4
6.	AY:(p([c]) | p(Y) => p(h([c],Y)))	Universal Elimination: 2
7.	p([c]) | p([d]) => p(h([c],[d]))	Universal Elimination: 6
8.	p(h([c],[d]))	Implication Elimination: 7, 5
9.	p([c]) & p([d]) => p(h([c],[d]))	Implication Introduction: 3, 8
10.	AY:(p([c]) & p(Y) => p(h([c],Y)))	Universal Introduction: 9
11.	AX:AY:(p(X) & p(Y) => p(h(X,Y)))	Universal Introduction: 10
12.	AX:p(X)	Induction: 1, 11
Goal	AX:p(X)	Complete

Proofs

(Exercise 13.6)

1. $p(a)$
2. $p(s(a))$
3. $\forall x.(p(x) \Rightarrow p(s(s(x))))$

Goal: $\forall x.p(x)$

Objects	a	
Functions	s	

☐	Select All	
☐ 1.	p(a)	Premise
☐ 2.	p(s(a))	Premise
☐ 3.	AX:(p(X) => p(s(s(X))))	Premise
☐ 4.	| p([c]) & p(s([c]))	Assumption
☐ 5.	| p([c])	And Elimination: 4
☐ 6.	| p(s([c]))	And Elimination: 4
☐ 7.	| p([c]) => p(s(s([c])))	Universal Elimination: 3
☐ 8.	| p(s(s([c])))	Implication Elimination: 7, 5
☐ 9.	| p(s([c])) & p(s(s([c])))	And Introduction: 6, 8
☐ 10.	p([c]) & p(s([c])) => p(s([c])) & p(s(s([c])))	Implication Introduction: 4, 9
☐ 11.	p(a) & p(s(a))	And Introduction: 1, 2
☐ 12.	AX:(p(X) & p(s(X)) => p(s(X)) & p(s(s(X))))	Universal Introduction: 10
☐ 13.	AX:(p(X) & p(s(X)))	Induction: 11, 12
☐ 14.	p([c]) & p(s([c]))	Universal Elimination: 13
☐ 15.	p([c])	And Elimination: 14
☐ 16.	p(s([c]))	And Elimination: 14
☐ 17.	AX:p(X)	Universal Introduction: 15

Goal	AX:p(X)	Complete

Proof Strategies

Proof strategies

Where to get started? **By making a plan**

Don't: panic if your plan is wrong! If you try a certain plan and things don't seem to be working, this is teaching you something about how the proof works. You're getting there!

Do: know what you're trying to accomplish at each step in the proof. For every assumption, you should be able to write down what conclusion you're trying to prove.

- If you're not sure, you can test one conclusion, see how it goes, and try another if it doesn't work

Proof strategies

Two strategies for planning

Strategy 1: Use the proof/premise type

- Look at the Fitch rules for the premises you have and the conclusion you want. What steps could you take to go from one to the other?

Strategy 2: Use the premises to build intuition

- Based on the premises, can you informally say (not mathematically) why the conclusion must be true? How can you build a proof based on this?

Proof strategies

Two strategies for planning:

Strategy 1: Use proof type

1. Figure out the type of proof you're doing. Based on the premises, is this structural induction? If the goal is a negation, is this a proof by contradiction? Etc.
2. Identify the sub-conclusions you need to prove, based on proof type. E.g. if this is a negation proof, then you'll need to prove two implications.
3. Prove sub-conclusions, one-at-a-time

Proof strategies

Strategy 2: Use premises to derive intuition

1. Based on the premises, can you explain (in words) why the conclusion must be true?
2. How can you translate this into steps/sub-proofs?

For example: If we have **AX:p(X)** and our conclusion is **~EX:~p(X)**, we might say "there can't be an X where ~p(X) because we know p(X) is true for all X." We can see that the premise contradicts EX:~p(X), so we could try a proof by contradiction, by assuming EX:~p(X)

Another example: Suppose we have AX: (p(X) => q(X)), AX:(~q(X) ⇔ ~r(X)), and p(a). Our goal is to prove r(a). When we verbalize why this is true, we'd say something like "since p(a) is true, we know that q(a) must be true. Since q(a) is true, we can see that ~r(a) cannot be true." This divides into sub-proofs: first, we prove that p(a) => q(a), and then we do a proof by contradiction to show that q(a) => r(a)

What if I get stuck?

Stay calm! If you're stressed out trying to solve the current proof, try

- Focusing on getting partial credit. Any progress is good progress!
- Working on another problem that you feel more confident about. Give yourself a break and collect some more points!

If you're **stuck in the middle**

- Check how many steps you're at. If it's almost as many as our solution and you're not finished or sure what you're trying to accomplish, this is probably a sign that your strategy is not effective. Refer to the Fitch rules to see if there are other tools you can use

What if I get stuck?

If you're **stuck at the beginning**

- Try to **verbalize** why the conclusion **must be true,** based on the premises. Doing this will help you understand how the proof works.
- **Look at the fitch rules** related to your premises and conclusion. E.g. if the premise has an existential quantifier and the conclusion has a universal quantifier, look at the rules for existentialism and universals. What does it take to remove an existential? To add a universal?
- Check for **negations.** If there's a negation on your conclusion, you'll probably need to do a negation introduction, by proving a contradiction
- Write out intermediary steps. If you know that you want to prove AX:p(X)=>q(X)&r(X) and you have q(X), what would be the second to last step? Can you figure out how to prove this (or at least attempt it?)

General tips for proofs

- Invest your time in strategizing, before you start writing down steps.
- Every time you make an assumption, know a) why, and b) what you're trying to prove (i.e. which implication is your goal?)
- If you're trying to create a negation, there will be two cases
  - **Case 1:** what you can prove from the premises/earlier assumptions
  - **Case 2:** what you can prove from the most recent assumption (should be the negation of what you can prove from the premises/earlier assumptions)
  - To decide what your goal should be, think about what in the premises & the statement you're negating is contradictory
- **Remember to submit!**

Using **Or Elimination**

When should I use **or elimination**?

- You have an or expression in a premise or assumption and you can use all elements of the or expression to prove some desired conclusion

How should I use or elimination?

1. Determine what you're trying to prove
2. Prove that each term in the or expression implies this goal

Or Elimination

$$\phi_1 \vee \ldots \vee \phi_n$$

$$\phi_1 \Rightarrow \psi$$

$$\ldots$$

$$\phi_n \Rightarrow \psi$$

$$\psi$$

Using Existential Elimination

When to use existential elimination?

- You have an existentially quantified expression that implies something you want to prove

How to use existential elimination?

- Assume phi of a placeholder term
- Prove psi (be careful using the placeholder term, since this term cannot occur free in psi)
- Do universal introduction

Existential Elimination

$$\frac{\exists \nu.\phi \qquad \forall \nu.(\phi \Rightarrow \psi)}{\psi}$$

where ν does not occur free in ψ