Duality of Satisfiability and Validity

- Recall: In propositional logic, satisfiability and validity are dual concepts:
  \[ F \text{ is valid iff } \neg F \text{ is unsatisfiable} \]
- This duality also holds in first-order logic.
- Thus, if we have a technique for deciding validity in FOL, this immediately yields a way to decide satisfiability.
- Hence, we’ll only focus on proving validity in this lecture.

Semantic Argument Method to Prove Validity

- We will use the semantic argument technique from earlier to prove validity of first-order formulas.
- This technique is not particularly amenable to automation, but is useful for paper-and-pencil proofs of validity.
- Recall: Semantic argument method is a proof by contradiction.
- Basic idea: Assume that \( F \) is not valid, i.e., there exists some \( S, \sigma \) such that \( S, \sigma \not\models F \)
- Then, apply proof rules.
- If can derive contradiction on every branch of proof, \( F \) is valid.

Proof Rules I (Review)

- All proof rules from prop. logic carry over to first-order logic.
- As before, proof rules come in pairs, for each connective, we have one case for \( \models \), one case for \( \not\models \).
- Negation elimination:
  \[
  S, \sigma \models \neg F \\
  S, \sigma \not\models \neg F \\
  S, \sigma \models F
  \]
- And elimination rule:
  \[
  S, \sigma \models F \land G \\
  S, \sigma \models F \\
  S, \sigma \models G
  \]
  \[
  S, \sigma \not\models F \land G \\
  S, \sigma \not\models F \\
  S, \sigma \not\models G
  \]

Proof Rules II (Review)

- Or elimination:
  \[
  S, \sigma \models F \lor G \\
  S, \sigma \models F \\
  S, \sigma \models G
  \]
  \[
  S, \sigma \not\models F \lor G \\
  S, \sigma \not\models F \\
  S, \sigma \not\models G
  \]
- Implication elimination:
  \[
  S, \sigma \models F \rightarrow G \\
  S, \sigma \not\models F \\
  S, \sigma \models G
  \]
  \[
  S, \sigma \not\models F \rightarrow G \\
  S, \sigma \not\models F \\
  S, \sigma \not\models G
  \]
- If and only if elimination:
  \[
  S, \sigma \models F \leftrightarrow G \\
  S, \sigma \models F \\
  S, \sigma \models G \\
  S, \sigma \not\models F \\
  S, \sigma \not\models G
  \]
  \[
  S, \sigma \not\models F \leftrightarrow G \\
  S, \sigma \models F \land \neg G \\
  S, \sigma \models F \lor \neg G \\
  S, \sigma \models \neg F \land G
  \]
### Proof Rules III (New)
- We need new rules to eliminate universal and existential quantifiers.
  - Universal elimination I:
    
    \[ U, I, \sigma \models \forall x. F \quad \text{for any} \quad o \in U \]
    
    \[ U, I, \sigma[x \mapsto o] \models F \]
  - Example: Suppose \( U, I, \sigma \models \forall x. \text{hates}(\text{jack}, x) \)
  - Using the above proof rule, we can conclude:
    
    \[ U, I, \sigma[x \mapsto \text{I(jack)}] \models \text{hates}(\text{jack}, x) \]

### Existential Elimination Rule I
- Existential elimination I:
  
  \[ U, I, \sigma \models \exists x. F \quad \text{(for a fresh} \quad o \in U) \]
  
  \[ U, I, \sigma[x \mapsto o] \models F \]
- Again, fresh means an object that has not been used before
- If \( U, I, \sigma \) entail \( \exists x. F \), we know there is some object for which \( F \) holds, but we don’t know which object
- If we introduce an object \( o \) we have previously used, we might know something else about \( o \)

### Proof Rules V (New)
- Finally, we need a rule for deriving for contradictions
  - Contradiction rule:
    
    \[ U, I, \sigma \models p(s_1, \ldots, s_n) \]
    
    \[ U, I, \sigma \not\models p(t_1, \ldots, t_n) \]
    
    \[ (I, \sigma)(s_i) = (I, \sigma)(t_i) \quad \text{for all} \quad i \in [1, n] \]
    
    \[ U, I, \sigma \not\models \bot \]
  - Example: Suppose we have \( S, \{x \mapsto a\} \models p(x) \) and \( S, \{y \mapsto a\} \not\models p(y) \)
  - The proof rule for contradiction allows us to derive \( \bot \)

### Universal Elimation Rule II
- Universal elimination II:
  
  \[ U, I, \sigma \not\models \forall x. F \quad \text{(for a fresh} \quad o \in U) \]
  
  \[ U, I, \sigma[x \mapsto o] \not\models F \]
  - By a fresh object constant, we mean an object that has not been previously used in the proof
  - Why do we have this restriction?
  - If \( U, I, \sigma \) do not entail \( \forall x. F \), we know there is some object for which \( F \) does not hold, but we don’t know which one
  - If we have have used an object \( o \) before in the proof, we might know something else about \( o \)

### Example 1: Proving Validity
- Prove the validity of formula:
  
  \[ F : (\forall x. p(x)) \rightarrow (\forall y. p(y)) \]
  - We start by assuming it is not valid, i.e., there exists some \( S, \sigma \) such that \( S, \sigma \not\models F \).
  1. \( S, \sigma \not\models (\forall x. p(x)) \rightarrow (\forall y. p(y)) \) \quad \text{assumption}
  2. \( S, \sigma \models \forall x. p(x) \) \quad 1 and \( \rightarrow \)
  3. \( S, \sigma \not\models \forall y. p(y) \)
  4. \( S, \sigma[y \mapsto o] \not\models p(y) \) \quad 3 and \( \not\models \forall x \)
  5. \( S, \sigma[x \mapsto o] \models p(x) \) \quad 2 and \( \models \forall x \)
  6. \( S, \sigma \models \bot \)

### Universal Elimation Rule II
- Existential elimination II:
  
  \[ U, I, \sigma \not\models \exists x. F \quad \text{(for any} \quad o \in U) \]
  
  \[ U, I, \sigma[x \mapsto o] \not\models F \]
  - Why can we instantiate \( x \) with any object?
  - Because if \( U, I, \sigma \) do not entail \( \exists x. F \), this means there does not exist any object for which \( F \) holds
  - Thus, no matter what object \( x \) maps to, it still won’t entail \( F \)
  - Therefore, ok to instantiate \( x \) with any object, regardless of whether it has been used before
Example 2

- Is this formula valid? Yes!
  
  \[ F : (\forall x. (p(x) \lor q(x))) \rightarrow (\exists x. p(x) \lor \forall x. q(x)) \]

- Informal argument: Suppose \( \forall x. (p(x) \lor q(x)) \) holds

- This means either \( q(x) \) for all objects (i.e., \( \forall x. q(x) \))

- Or if \( q(x) \) does not hold for some object \( o \), then \( p(x) \) must hold for that object \( o \) (i.e., \( \exists x. p(x) \))

- Thus, antecedent implies \( \exists p(x) \lor \forall x. q(x) \)

Example 4

- Let’s now prove validity using semantic argument method
  
  \[ F : (\forall x. (p(x) \lor q(x))) \rightarrow (\exists x. p(x) \lor \forall x. q(x)) \]

- Let’s assume there is some \( S, \sigma \) that does not entail \( \phi \), and derive contradiction on all branches

<table>
<thead>
<tr>
<th>Step</th>
<th>Formula</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>( S, \sigma \not\models F )</td>
<td>assumption</td>
</tr>
<tr>
<td>2.</td>
<td>( S, \sigma \models \forall x. (p(x) \lor q(x)) )</td>
<td>1 and ( \rightarrow )</td>
</tr>
<tr>
<td>3.</td>
<td>( S, \sigma \not\models \exists x. (p(x) \lor \forall x. q(x)) )</td>
<td>1 and ( \rightarrow )</td>
</tr>
<tr>
<td>4a.</td>
<td>( S, \sigma \not\models (\forall x. p(x)) )</td>
<td>3 and ( \lor )</td>
</tr>
<tr>
<td>5.</td>
<td>( S, \sigma \not\models \forall x. q(x) )</td>
<td>5 and ( \not\models \forall x. \text{fresh } o )</td>
</tr>
<tr>
<td>6a.</td>
<td>( S, \sigma[x \mapsto o] \not\models q(x) )</td>
<td>5 and ( \not\models \forall x. \text{fresh } o )</td>
</tr>
<tr>
<td>7a.</td>
<td>( S, \sigma[x \mapsto o] \not\models p(x) \lor q(x) )</td>
<td>4 and ( \not\models \exists x. \text{any } o )</td>
</tr>
<tr>
<td>8a.</td>
<td>( S, \sigma[x \mapsto o] \not\models p(x) \lor q(x) )</td>
<td>2 and ( \not\models \forall x. \text{any } o )</td>
</tr>
<tr>
<td>9a.</td>
<td>( S, \sigma[x \mapsto o] \not\models p(x) \lor q(x) )</td>
<td>8 and ( \lor )</td>
</tr>
<tr>
<td>10a.</td>
<td>( S, \sigma \models \bot )</td>
<td>7, 9a</td>
</tr>
<tr>
<td>10b.</td>
<td>( S, \sigma \models \bot )</td>
<td>6, 9b</td>
</tr>
</tbody>
</table>

Example 3

- Now, how do we formally prove this formula is not valid?
  
  \[ F : (\forall x. p(x, x)) \rightarrow (\exists x. \forall y. p(x, y)) \]

- We have to come up with \( U, I, \sigma \) such that \( U, I, \sigma \not\models F \)

- In this case, \( \sigma \) not necessary since no free variables

- Choose \( U = \{ \ast, \bullet \} \), and \( I(p) = \{ (\ast, \ast), (\bullet, \bullet) \} \)

- Clearly, under \( I, \forall x.p(x, x) \) evaluates to true.

- Furthermore, under \( I, (\exists x. \forall y.p(x, y)) \) evaluates to false.

- Thus, \( I \) is a falsifying interpretation of \( F \).
Soundness and Completeness of Proof Rules

- The proof rules we used are sound and complete.
- **Soundness**: If every branch of semantic argument proof derives a contradiction, then \( F \) is indeed valid.
- **Translation**: The proof system does not reach wrong conclusions
- **Completeness**: If formula \( F \) is valid, then there exists a finite-length proof in which every branch derives \( \bot \).
- **Translation**: There are no valid first-order formulas which we cannot prove to be valid using our proof rules.
- Completeness in this context also called refutational completeness

Important Properties of First Order Logic

- **Really important result**: It is undecidable whether a first-order formula is valid. (Church and Turing)
- **Review**: A problem is decidable iff there exists a procedure \( P \) such that, for any input:
  1. \( P \) halts and says “yes” if the answer is positive
  2. \( P \) halts and says “no” if the answer is negative
- But, what about the completeness result? Doesn’t this contradict undecidability?
- No, because completeness says we will find proof of validity if it exists, but if formula is invalid, we might search forever.

Semidecidability of First-Order Logic

- First-order logic is semidecidable
- A decision problem is semidecidable iff there exists a procedure \( P \) such that, for any input:
  1. \( P \) halts and says “yes” if the answer is positive
  2. \( P \) may not terminate if the answer is negative
- Thus, there exists an algorithm that always terminates and says if any arbitrary FOL formula is valid
- But no algorithm is guaranteed to terminate if the FOL formula is not valid

Decidable Fragments of First-Order Logic

- Although full-first order logic is not decidable, there are fragments of FOL that are decidable.
- A fragment of FOL is a syntactically restricted subset of full FOL: e.g., no functions, or only universal quantifiers, etc.
- Some decidable fragments:
  - Quantifier-free first order logic
  - Monadic first-order logic
  - Bernays-Schönfinkel class

Quantifier-Free Fragment of FOL

- The quantifier-free fragment of FOL is the syntactically restricted subset of FOL where formulas do not contain universal or existential quantifiers.
- Determining validity and satisfiability in quantifier-free FOL is decidable (NP-complete).
- This fragment can be reduced to a theory we will explore later, theory of equality with uninterpreted functions

Monadic First-Order Logic

- Pure monadic FOL: all predicates are monadic (i.e., arity 1) and no function constants.
- Impure monadic FOL: both monadic predicates and monadic function constants allowed
- Result: Monadic first-order logic is decidable (both versions)
- However, if we add even a single binary predicate, the logic becomes undecidable.
Another important property of FOL is compactness.

A logic is called compact if an infinite set of sentences $\Gamma$ is satisfiable iif every finite subset of $\Gamma$ is satisfiable.

Theorem (due to Gödel): First-order logic is compact.

Proof of compactness of FOL follows from the completeness of proof rules.

Recall: Completeness means that if a formula is unsatisfiable, then there exists a finite-length proof of unsatisfiability.

Suppose FOL was not compact, i.e., there is an infinite set of sentences $\Gamma$ that are unsat, but every finite subset $\Sigma$ is sat.

By completeness of proof rules, if $\Gamma$ is unsat, there exists a finite-length proof of unsatisfiability.

But this means the proof must use a finite subset of sentences $\Sigma$ of $\Gamma$, otherwise proof could not be finite.

But this implies there is also a proof of unsatisfiability of $\Sigma$.

Thus, by soundness of proof rules, $\Sigma$ must be unsat. \qed
Consequences of Compactness

- Proof of compactness might look like a useless property, but it has very interesting consequences!
- Compactness can be used to show that a variety of interesting properties are not expressible in first-order logic.
- For instance, we can use compactness theorem to show that transitive closure is not expressible in first order logic.

"Expressing" Transitive Closure in FOL

- Given a directed graph \( G = (V, E) \), the transitive closure of \( G \) is defined as the graph \( G^* = (V, E^*) \) where:
  \[
  E^* = \{(n, n') \mid \text{if there is a path from vertex } n \text{ to } n' \}
  \]

- Observe: A binary predicate \( p(t, t') \) be viewed as a graph containing an edge from node \( t \) to \( t' \)

- Thus, the concept of transitive closure applies to binary predicates as well.

- A binary predicate \( T \) is the transitive closure of predicate \( p \) if \( \langle t_0, t_n \rangle \in T \) iff there exists some sequence \( t_0, t_1, \ldots, t_n \) such that \( (t_i, t_{i+1}) \in p \)

Proof I

- \( \Psi^n(a, b) \) encode the proposition: there is no path of length \( n \) from \( a \) to \( b \).

- In particular, \( \Psi^1 = \neg p(a, b) \)

- Similarly,
  \[
  \Psi^n = \neg \exists x_1, \ldots, x_{n-1}. (p(a, x_1) \land p(x_1, x_2) \land \ldots \land p(x_{n-1}, b))
  \]

Transitive Closure and FOL

- In fact, no matter how hard we try to correct this definition, we cannot express transitive closure in FOL.

- Will use compactness theorem to show that transitive closure is not expressible in FOL.

- Compactness: An infinite set of sentences \( \Gamma \) is satisfiable iff every finite subset of \( \Gamma \) is satisfiable.

- For contradiction, suppose transitive closure is expressible in first order logic.

- Let \( \Gamma' \) be a (possibly infinite) set of sentences expressing that \( T \) is the transitive closure of \( p \).

Proof II

- Recall: \( \Gamma \) is a set of propositions encoding \( T \) is transitive closure of \( p \).

- Now, construct \( \Gamma' \) as follows:
  \[
  \Gamma' = \Gamma \cup \{ T(a, b), \Psi^1, \Psi^2, \Psi^3, \ldots \}
  \]

- Observe: \( \Gamma' \) is unsatisfiable because:
  1. Since \( \Gamma \) encodes that \( T \) is transitive closure of \( p \), \( T(a, b) \) says there is some path from \( a \) to \( b \)
  2. The infinite set of propositions \( \Psi^1, \Psi^2, \ldots \) say that there is no path of any length from \( a \) to \( b \)
Proof III

- Now, consider any finite subset of $\Gamma'$:
  \[ \Gamma' = \Gamma \cup \{ T(a, b), \Psi_1, \Psi_2, \Psi_3, \ldots \} \]
- Clearly, any finite subset does not contain $\Psi_i$ for some $i$.
- Observe: This finite subset is satisfied by a model where there is a path of length $i$ from $a$ to $b$.
- Thus, every finite subset of $\Gamma'$ is satisfiable.
- By the compactness theorem, this would imply $\Gamma'$ is also satisfiable.
- But we just showed that $\Gamma'$ is unsatisfiable!
- Thus, transitive closure cannot be expressed in FOL!

Summary

- Semantic argument method for proving validity in FOL
- Soundness and completeness of semantic argument method
- Important properties of FOL: undecidability, semidecidability, compactness
- Compactness: useful to show what is not expressible in FOL
- Next lecture: Basics of automated first-order theorem provers (much less theoretical)