Recap

- Over the last three lectures, we have looked at:
  - Propositional logic, semantics and proof systems
  - Doing propositional logic proofs in Isabelle

- Today:
  - Syntax and Semantics of First-Order Logic (FOL)
  - Natural Deduction rules for FOL
  - Doing FOL proofs in Isabelle
Consider the following problem:

1. If someone cheats then everyone loses the game.
2. If everyone who cheats also loses, then I lose the game.
3. Did I lose the game?

Is Propositional Logic rich enough to formally represent and reason about this problem?

The finer logical structure of this problem would not be captured by the constructs we have so far encountered.

We need a richer language!
A Richer Language

First-order (predicate) logic (FOL) extends propositional logic:

- Atomic formulas are assertions about properties of individual(s).
  e.g. an individual might have the property of being a cheat.

- We can use variables to denote arbitrary individuals.
  e.g. $x$ is a cheater.

- We can bind variables with quantifiers $\forall$ (for all) and $\exists$ (exists).
  e.g. for all $x$, $x$ is a cheater.

- We can use connectives to compose formulas:
  e.g. for all $x$, if $x$ is a cheater then $x$ loses.

- We can use quantifiers on subformulas.
  e.g. we can formally distinguish between: “if anyone cheats we lose the game” and “if everyone cheats, we lose the game”.
Terms of First-Order Logic

Given:

- a countably infinite set of (individual) variables $\mathcal{V} = \{x, y, z, \ldots\}$;
- a finite or countably infinite set of function letters $\mathcal{F}$ each assigned a unique arity (possibly 0)

then the set of (well-formed) terms is the smallest set such that

- any variable $v \in \mathcal{V}$ is a term;
- if $f \in \mathcal{F}$ has arity $n$, and $t_1, \ldots t_n$ are terms, so is $f(t_1, \ldots, t_n)$.

Remarks

- If $f$ has arity 0, we usually write $f$ rather than $f()$, and call $f$ a constant
- In practice, we use infix notation when appropriate: e.g., $x + y$ instead of $+(x, y)$. 
Formulas of First-Order Logic

Given a countably infinite set of predicates $\mathcal{P}$, each assigned a unique arity (possibly 0), the set of wffs is the smallest set such that

- if $A \in \mathcal{P}$ has arity $n$, and $t_1, \ldots t_n$ are terms, then $A(t_1, \ldots, t_n)$ is a wff;
- if $P$ and $Q$ are wffs, so are $\neg P$, $P \lor Q$, $P \land Q$, $P \rightarrow Q$, $P \leftrightarrow Q$,
- if $P$ is a wff, so are $\exists x. P$ and $\forall x. P$ for any $x \in \mathcal{V}$;
- if $P$ is a wff, then $(P)$ is a wff.

Remarks

- If $A$ has arity 0, we usually write $A$ rather than $A()$, and call $A$ a **propositional variable**. This way, propositional logic wffs look like a subset of FOL wffs. Also, use infix notation where appropriate.

- We assume $\exists x$ and $\forall x$ bind more weakly than any of the propositional connectives.
  $\exists x. P \land Q$ is $\exists x. (P \land Q)$, not $(\exists x. P) \land Q$.
  (NB: H&H assume $\exists x$ and $\forall x$ bind like $\neg$.)
Example: Problem Revisited

We can now formally represent our problem in FOL:

- **Assumption 1**: If someone cheats then everyone loses the game: \((\exists x. \text{Cheats}(x)) \rightarrow \forall x. \text{Loses}(x)\).

- **Assumption 2**: If everyone who cheats also loses, then I lose the game: \((\forall x. \text{Cheats}(x) \rightarrow \text{Loses}(x)) \rightarrow \text{Loses}(\text{me})\).

To answer the question *Did I lose the game?* we need to prove either \(\text{Loses}(\text{me})\) or \(\neg \text{Loses}(\text{me})\) from these assumptions.

More on this later.
Free and Bound Variables

- An occurrence of a variable $x$ in a formula $P$ is **bound** if it is in the scope of a $\forall x$ or $\exists x$ quantifier.
- A variable occurrence $x$ is **in the scope of** a quantifier occurrence $\forall x$ or $\exists x$ if the quantifier occurrence is the first occurrence of a quantifier over $x$ in a traversal from the variable occurrence position to the root of the formula tree.
- If a variable occurrence is **not** bound, it is **free**

**Example**

In

$$P(x) \land (\forall x. P(y) \rightarrow P(x))$$

The first occurrence of $x$ and the occurrence of $y$ are free, while the second occurrence of $x$ is bound.
Free and Bound Variables

- An occurrence of a variable $x$ in a formula $P$ is **bound** if it is in the scope of a $\forall x$ or $\exists x$ quantifier.
- A variable occurrence $x$ is in the scope of a quantifier occurrence $\forall x$ or $\exists x$ if the quantifier occurrence is the first occurrence of a quantifier over $x$ in a traversal from the variable occurrence position to the root of the formula tree.
- If a variable occurrence is not bound, it is **free**

**Example**

In

\[
P(\text{free } x) \land (\forall x. P(y) \rightarrow P(x))
\]

The first occurrence of $x$ and the occurrence of $y$ are free, while the second occurrence of $x$ is bound.
Free and Bound Variables

✿ An occurrence of a variable $x$ in a formula $P$ is **bound** if it is in the scope of a $\forall x$ or $\exists x$ quantifier.

✿ A variable occurrence $x$ is **in the scope of** a quantifier occurrence $\forall x$ or $\exists x$ if the quantifier occurrence is the first occurrence of a quantifier over $x$ in a traversal from the variable occurrence position to the root of the formula tree.

✿ If a variable occurrence is **not** bound, it is **free**

**Example**

In

\[
P(x) \land (\forall x. P(y) \rightarrow P(x))
\]

The first occurrence of $x$ and the occurrence of $y$ are free, while the second occurrence of $x$ is bound.
Free and Bound Variables

- An occurrence of a variable $x$ in a formula $P$ is **bound** if it is in the scope of a $\forall x$ or $\exists x$ quantifier.

- A variable occurrence $x$ is **in the scope of** a quantifier occurrence $\forall x$ or $\exists x$ if the quantifier occurrence is the first occurrence of a quantifier over $x$ in a traversal from the variable occurrence position to the root of the formula tree.

- If a variable occurrence is **not** bound, it is **free**

**Example**

In

$$P(x) \land (\forall x. \ P(y) \rightarrow P(x))$$

The first occurrence of $x$ and the occurrence of $y$ are free, while the second occurrence of $x$ is bound.
Substitution Rules

If $P$ is a formula, $s$ is a term and $x$ is a variable, then

$$P[s/x]$$

is the formula obtained by **substituting** $s$ for all free occurrences of $x$ throughout $P$.

**Example**

$$(\exists x. P(x, y)) [3/y] \equiv \exists x. P(x, 3)$$
$$(\exists x. P(x, y)) [2/x] \equiv \exists x. P(x, y).$$

If necessary, bound variables in $P$ must be **renamed** to avoid capture of free variables in $s$.

$$(\exists x. P(x, y)) [f(x)/y] = \exists z. P(z, f(x))$$
Semantics of First-Order Logic Formulas

(Recall that an interpretation for propositional logic maps atomic propositions to truth values.)

Informally, an interpretation of a formula maps its function letters to actual functions, and its predicate symbols to actual predicates.

The interpretation also specifies some domain of discourse $\mathcal{D}$ (a non-empty set or universe) on which the functions and relations are defined.

A formal definition requires some work!
Definition (Interpretation)

An interpretation consists of a non-empty set $\mathcal{D}$, called the domain of the interpretation, together with the following assignments

1. each **predicate letter** of arity $n > 0$ is assigned to a subset of $\mathcal{D} \times \cdots \times \mathcal{D}$. Each **nullary predicate** is assigned either $T$ or $F$.

2. Each **function letter** of arity $n > 0$ is assigned to a function $(\mathcal{D} \times \cdots \times \mathcal{D}) \rightarrow \mathcal{D}$. Each **nullary function (constant)** is assigned to a value in $\mathcal{D}$. 
Consider the formula:

\[ P(a) \land \exists x. Q(a, x) \quad (\ast) \]

In one possible interpretation:

- the domain is the set of natural numbers \( \mathbb{N} = \{0, 1, 2, 3, \ldots\} \);
- assign 2 to \( a \), assign the property of being even to \( P \), and the relation of being greater than to \( Q \), i.e. \( Q(x, y) \) means \( x \) is greater than \( y \);
- under this interpretation: \((\ast)\) affirms that 2 is even and there exists a natural number that 2 is greater than. Is \((\ast)\) satisfied under this interpretation?
- Such a satisfying interpretation is sometimes known as a **model**.

**NB:** In H&R, a model is *any* interpretation.
Example of Interpretation

Consider the formula:

\[ P(a) \land \exists x. Q(a, x) \quad (\ast) \]

In one possible interpretation:

- the domain is the set of natural numbers \( \mathbb{N} = \{0, 1, 2, 3, \ldots\} \);
- assign 2 to \( a \), assign the property of being even to \( P \), and the relation of being greater than to \( Q \), i.e. \( Q(x, y) \) means \( x \) is greater than \( y \);
- under this interpretation: (\ast) affirms that 2 is even and there exists a natural number that 2 is greater than. Is (\ast) satisfied under this interpretation? — Yes.
- Such a satisfying interpretation is sometimes known as a model.

NB: In H&R, a model is any interpretation.
Definition (Assignment)

Given an interpretation $M$, an assignment $s$ assigns a value from the domain $\mathcal{D}$ to each variable in $\mathcal{V}$.

We extend this assignment to all terms inductively by saying that

1. if $M$ maps the $n$-ary function letter $f$ to the function $F$, and
2. if terms $t_1, \ldots, t_n$ have been assigned values $a_1, \ldots, a_n \in \mathcal{D}$

then we can assign value $F(a_1, \ldots, a_n) \in \mathcal{D}$ to the term $f(t_1, \ldots, t_n)$.

An assignment $s$ of values to variables is also commonly known as an environment and we denote by $s[x \mapsto a]$ the environment that maps $x \in \mathcal{D}$ to $a$ (and any other variable $y \in \mathcal{D}$ to $s(y)$).

Remark: The interpretation $M$ is fixed before we interpret a formula, but the assignment $s$ will vary as we interpret the quantifiers.
Semantics of FOL Formulas (IV)

Definition (Satisfaction)
Given an interpretation $M$ and an assignment $s$ from $V$ to $D$

1. any wff which is a nullary predicate letter $A$ is satisfied if and only if the interpretation in $M$ of $A$ is $T$;

2. suppose we have a wff $P$ of the form $A(t_1 \ldots t_n)$, where $A$ is interpreted as relation $R$ and $t_1, \ldots, t_n$ have been assigned values $a_1, \ldots, a_n$ by $s$. Then $P$ is satisfied if and only if $(a_1, \ldots, a_n) \in R$;

3. any wff of the form $\forall x. P$ is satisfied if and only if $P$ is satisfied with respect to assignment $s[x \mapsto a]$ for all $a \in D$;

4. any wff of the form $\exists x. P$ is satisfied if and only if $P$ is satisfied with respect to assignment $s[x \mapsto a]$ for some $a \in D$;

5. any wffs of the form $P \lor Q, P \land Q, P \rightarrow Q, P \leftrightarrow Q, \neg P$ are satisfied according to the truth-tables for each connective (e.g. $P \lor Q$ is satisfied if and only if $\phi$ is satisfied or $\psi$ is satisfied).
Semantics of FOL Formulas (V)

Definition (Entailment)

We write $M \models_s P$ to mean that wff $P$ is satisfied by interpretation $M$ and assignment $s$.

We say that the wffs $P_1, P_2, \ldots, P_n$ entail wff $Q$ and write

$$P_1, P_2, \ldots, P_n \models Q$$

if, for any interpretation $M$ and assignment $s$ for which $M \models_s P_i$ for all $i$, we also have $M \models_s Q$.

As with propositional logic, we must ensure that our inference rules are valid. That is, if

$$\begin{array}{cccc}
P_1 & P_2 & \cdots & P_n \\
\hline
Q
\end{array}$$

then we must have $P_1, P_2, \ldots, P_n \models Q$. 
We now consider the additional natural deduction rules for FOL.

The introduction rules for the quantifiers are:

- Universal quantification: (Provided that $x_0$ is not free in the assumptions.)
  \[
  \frac{P[x_0/x]}{\forall x. P} \quad \text{(allI)}
  \]

- Existential quantification:
  \[
  \frac{P[t/x]}{\exists x. P} \quad \text{(exI)}
  \]
Existential Elimination

\[
\dfrac{
\exists x. P \\
\vdots \\
\exists x. P}{Q} \quad (\text{exE})
\]

Provided \(x_0\) does not occur in \(Q\) or any assumption other than \(P[x_0/x]\) on which the derivation of \(Q\) from \(P[x_0/x]\) depends.
Universal Elimination

Specialisation rule:

\[ \forall x. P \]
\[ \frac{}{P[t/x]} \quad \text{(spec)} \]

An alternative universal elimination rule is \texttt{allE}:

\[ [P[t/x]] \]
\[ \vdots \]
\[ \forall x. P \]
\[ Q \]
\[ \frac{}{Q} \quad \text{(allE)} \]
Example Proof

Prove that $\exists y. P(y)$ is true, given that $\forall x. P(x)$ holds.

$$
\begin{align*}
\forall x. P(x) & \quad \text{(spec)} \\
\frac{P(a)}{\exists y. P(y)} & \quad \text{(exI)}
\end{align*}
$$
Example Proof

Prove that $\exists y. P(y)$ is true, given that $\forall x. P(x)$ holds.

$\begin{align*}
\forall x. P(x) & \quad \text{(spec)} \\
\hline
P(a) & \\
\hline
\exists y. P(y) & \quad \text{(exI)}
\end{align*}$

**Side note:** semantically, we implicitly use here the fact that our domain of discourse is non-empty. It doesn’t matter what $a$ is, but we have to have something.
Why the side conditions on `allI` and `exE`?

A “proof” of $P(y) \vdash \forall x. P(x)$:

\[
P(y) \equiv P(x)[y/x] \\
\frac{}{\forall x. P(x)} \quad (\text{allI})
\]

“If $P$ is true for some $y$, then $P$ is true for all $x$”

But: The variable $y$ is free in the assumption $P(y)$!

A “proof” of $\exists x. P(x) \vdash P(y)$ (i.e., for any $y$)

\[
\frac{\exists x. P(x) \quad [P(x)[y/x]]}{P(y)}
\]

The variable $y$ should not occur in the conclusion, nor in any other assumptions.

**Machine assistance:** Isabelle keeps track of which variable names are allowed where, so we can only apply the rules in a sound way.
Example Proof (II)

Prove that $\forall x. Q(x)$ is true, given $\forall x. P(x)$ and $(\forall x. P(x) \rightarrow Q(x))$.

$$\begin{align*}
\forall x. P(x) & \rightarrow Q(x) \\
\forall x. P(x) & \rightarrow Q(y) \\
\forall x. P(x) & \rightarrow Q(y) \\
\forall x. P(x) & \rightarrow Q(y) \\
\forall x. Q(x) & \rightarrow Q(y) \\
\forall x. Q(x) & \rightarrow Q(y) \\
\forall x. Q(x) & \rightarrow Q(y) \\
\forall x. Q(x) & \rightarrow Q(y) \\
\forall x. Q(x) & \rightarrow Q(y)
\end{align*}$$
Problem (III)

Prove that $\text{Loses}(me)$ given that

1. $(\exists x. \text{Cheats}(x)) \rightarrow \forall x. \text{Loses}(x)$
2. $(\forall x. \text{Cheats}(x) \rightarrow \text{Loses}(x)) \rightarrow \text{Loses}(me)$

\[
\begin{align*}
\text{assumption1} & \quad \left[ \text{Cheats}(y) \right]_1 \quad \text{(exI)} \\
& \quad \exists x. \text{Cheats}(x) \\
& \quad \forall x. \text{Loses}(x) \quad \text{(mp)} \\
& \quad \text{Loses}(y) \quad \text{(spec)} \\
& \quad \text{Cheats}(y) \rightarrow \text{Loses}(y) \quad \text{(impI)} \\
\text{assumption2} & \quad \forall x. \text{Cheats}(x) \rightarrow \text{Loses}(x) \quad \text{(allI)} \\
& \quad \text{Loses}(me) \quad \text{(mp)}
\end{align*}
\]
Isabelle’s HOL object logic is richer than the FOL so far presented. One difference is that all variables, terms and formulas have **types**. The type language is built using

- **base types** such as `bool` (the type of truth values) and `nat` (the type of natural numbers).
- **type constructors** such as `list` and `set` which are written postfix, e.g., `nat list` or `nat set`.
- **function types** written using `⇒`; e.g.

\[ \text{nat} \times \text{nat} \Rightarrow \text{nat} \]

which is a function taking two arguments of type `nat` and returning an object of type `nat`.

- **type variables** such as `'a`, `'b` etc. These give rise to polymorphic types such as `'a ⇒ `'a`. 
Consider the mathematical predicate \( a = b \mod n \). We could formalise this operator as:

\[
\text{definition } \text{mod} :: " \text{int} \Rightarrow \text{int} \Rightarrow \text{int} \Rightarrow \text{bool}"
\]

where "\( \text{mod } a b n \equiv \exists k. a = k \times n + b \)"

Isabelle performs type inference, allowing us to write:

\[
\forall x y n. \text{mod } x y n \rightarrow \text{mod } y x n
\]

instead of

\[
\forall (x :: \text{int}) (y :: \text{int}) (n :: \text{int}). \text{mod } x y n \rightarrow \text{mod } y x n
\]
FOL L-System Sequent Calculus Rules

\[ \frac{\Gamma \vdash P[x_0/x]}{\Gamma \vdash \forall x. P} \] (allI) \hspace{1cm} \frac{\Gamma, P[t/x] \vdash Q}{\Gamma, \forall x. P \vdash Q} \] (e allE t) \hspace{1cm} \frac{\Gamma, \forall x. P, P[t/x] \vdash Q}{\Gamma, \forall x. P \vdash Q} \] (f spec t)

\[ \frac{\Gamma \vdash P[t/x]}{\Gamma \vdash \exists x.P} \] (exI) \hspace{1cm} \frac{\Gamma, P[x_0/x] \vdash Q}{\Gamma, \exists x. P \vdash Q} \] (e exE t) \hspace{1cm} \frac{\Gamma, \forall x. \neg P \vdash \bot}{\Gamma \vdash \exists x. P} \] (exCIF)

- Rule prefixes: e = erule, f = frule
- \( x_0 \) is some variable not free in hypotheses or conclusion. Isabelle automatically picks fresh names (to ensure soundness!)
- When \( t \) suffix is used above (e.g., as in “e allE t”), then the term \( t \) can be explicitly specified in Isabelle method using a variant of the existing method. e.g., apply (erule_tac x="t" in allE).
- Rule exCIF is a variation on the standard Isabelle rule exCI introduced in the FOL.thy file on the course webpage.
Example II as a FOL Sequent Proof

\[
\begin{align*}
\frac{P(y) \vdash P(y)}{P(y) \rightarrow Q(y), P(y) \vdash Q(y)} & \quad \text{(e impE)} \\
\frac{P(y), Q(y) \vdash Q(y)}{P(y) \rightarrow Q(y), P(y) \vdash Q(y)} & \quad \text{(e allE y)} \\
\frac{P(y) \rightarrow Q(y), (\forall x. P(x)) \vdash Q(y)}{(\forall x. P(x) \rightarrow Q(x)), (\forall x. P(x)) \vdash Q(y)} & \quad \text{(e allE y)} \\
\frac{(\forall x. P(x) \rightarrow Q(x)), (\forall x. P(x)) \vdash Q(y)}{(\forall x. P(x) \rightarrow Q(x)), (\forall x. P(x)) \vdash \forall x. Q(x)} & \quad \text{(allI)}
\end{align*}
\]
Summary

- Introduction to First-Order Logic (H&R 2.1-2.4)
  - Syntax and Semantics
  - Substitution
  - Natural Deduction rules for quantifiers

- Isabelle and First-Order Logic
  - Defining predicates
  - A brief look at types
  - Have a look at FOL.thy on the course webpage, and the exercises.

- Next time:
  - Representation and Higher-Order Logic (HOL)