

1 Overview

Computing can trace its intellectual roots to logic. Questions such as: “what is a proof?” and the consequences (*e.g.*, the definition of computability) led to the theoretical foundations of computing. In this course, we have presented logic from a computational point of view, as is fitting for a computer science class. This has allowed us to take advantage of our intuitions about computation. We also saw, in a very direct way, how logic can help you gain intellectual control over the programs you write.

By using the ACL2 theorem prover, we saw that we can mechanize logic, *i.e.*, we can develop programs that reason about other programs. Systems such as ACL2 are programs whose goals are to augment human intelligence. We give them proof outlines and they fill in the details, possibly pointing out errors. We give them definitions and they check them for consistency. We program them and they can crank through proofs that no human could conceivably go through by hand.

We are now at the end of the semester, and the one remaining thing we want to do is to introduce you to traditional logic, as it traditionally taught and to show and highlight the connections with what we have been doing.

But, before we start, I would note that this is just the tip of the iceberg. You’ve seen how a theorem prover might behave from a user’s point of view. Things are much more interesting if you were to consider how to build and extend such a system. It is the difference between flying a plane and designing one.

2 Basic Set Theory

Before we start with logic, we briefly review set theory, since a basic understanding of set theory is required.

A set is a collection of items. For example: \emptyset and $\{\}$ both denote the empty set, $\{\mathbf{true}, \mathbf{false}\}$ is the set of Boolean values, and \mathbb{N} denotes the natural numbers, *i.e.*, $\{0, 1, 2, \dots\}$. We write $x \in A$ to denote that x is in (a member of) A , and $y \notin A$ to denote that y is not in A .

We can create new sets from existing sets in several ways. One is by using ordered pairs: the *ordered pair* whose first component is i and whose second component is j is denoted $\langle i, j \rangle$.

Ordered pairs are quite convenient. For example, they allow us to talk about relations and sets. R is a *binary relation* on set S if $R \subseteq S \times S = \{\langle x, y \rangle : x, y \in S\}$. We abbreviate $\langle s, w \rangle \in R$ by sRw . A function is a relation such that xRy and xRw implies $y = w$. Function application is usually written $f(x)$,

but is sometimes denoted by an infix dot “.”. That is, $f(x)$ and $f.x$ denote the unique y such that xfy .

Set comprehension is a convenient way of specifying sets. Several commonly used variations are shown below and used to specify the set of even natural numbers:

1. $\{x \in A \mid P(x)\}$: This set consists of all elements x of A such that x satisfies property P . For example, $\{x \in \mathbb{N} \mid x \text{ is even}\}$ denotes the set of even natural numbers.
2. $\{x \mid x \in A \text{ and } P(x)\}$: This set consists of all elements x such that x is in A and x satisfies property P . For example, $\{x \mid x \in \mathbb{N} \text{ and } x \text{ is even}\}$ denotes the set of even natural numbers.
3. $\{f(x) \mid x \in A\}$: This set consists of all elements $f(x)$ such that x is in A . For example, $\{2x \mid x \in \mathbb{N}\}$ denotes the set of even natural numbers.

Important Point It is possible to get into trouble when “defining” sets. This is shown by *Russell’s paradox*: consider the set of all sets that are not elements of themselves, $A = \{x \mid x \notin x\}$. Is $A \in A$? If yes, then by definition of A , $A \notin A$, so that can’t be. Therefore, $A \notin A$, but then by definition of A , $A \in A$. What have we done? We proved **false** by case analysis and the “definition” of A .

This was a *major* result discovered in 1901 by Russell. It led to much follow up work and to work on *axiomatic* set theory. What we are considering here is what is technically called *naive* set theory.

Historical note: variants of Russell’s paradox were discovered previously by others, *e.g.*, Cesare Burali-Forti discovered a similar paradox involving ordinal numbers.

You will note that the above variants of set comprehension do not allow you to “define” A because you need to start with a superset of the set you define using set comprehension. *And*, the “set of all sets” is *not* a set.

End Important Point

Some of the basic operations on sets are:

1. Subset: $A \subseteq B$ iff every element in A is in B .
2. Equality: $A = B$ iff every $A \subseteq B$ and $B \subseteq A$, *i.e.*, A and B have exactly the same elements.
3. Set union: $x \in A \cup B$ iff $x \in A$ or $x \in B$.
4. Set intersection: $x \in A \cap B$ iff $x \in A$ and $x \in B$.
5. Set difference: $x \in A \setminus B$ iff $x \in A$ but $x \notin B$.
6. Cross product: $x \in A \times B$ iff $x = \langle a, b \rangle$ and $a \in A$ and $b \in B$.
7. Set cardinality: the size of a set S is denoted by $|S|$.

8. Powerset: $\mathcal{P}(S)$ denotes the powerset of S , *i.e.*, the set $= \{A \mid A \subseteq S\}$.

Set show up everywhere. In fact, they are what *all* of modern mathematics are based on. For example, last week we discussed equivalence relations and congruence-based reasoning in ACL2. That allowed us to extend the idea of substituting equals for equals. In order to do that, we had to discuss equivalence relations and ideas such as reflexivity, symmetry, and transitivity. So, a relation B over some universe U is:

- *reflexive*: if for all $x \in U$, xBx .
- *symmetric*: if for all $x, y \in U$, if xBy then yBx .
- *transitive*: if for all $x, y, z \in U$, if xBy and yBz , then xBz .
- *an equivalence relation*: if it is reflexive, symmetric, and transitive.

Here are some definitions and theorems for you to consider.

- The identity relation, B^0 is $\{(x, x) \mid x \in U\}$.
- The *composition* of B and C is denoted $B;C$ and is the set $\{(b, c) \mid \text{there exists } x \text{ such that } bBx \text{ and } xCc\}$.
- For all natural numbers i , B^{i+1} is $B^i;B$.

Exercise 1 *Prove the following.*

1. B is reflexive iff $B^0 \subseteq B$.
2. $B^1 = B$.
3. B is transitive iff $B^2 \subseteq B$.
4. $\cup_{i \in \mathbb{N}} B^i$ is transitive.
5. $|\mathcal{P}(S)| > |S|$.

3 Intro to Logic

Logic forms the foundation of mathematics. We already have seen throughout the semester how to use logic to define the semantics of a programming language and how to develop a proof theory for that language. So, let's look at this from a more traditional point of view, with a classic example.

A *group* is a triple $\langle G, \circ, e \rangle$ such that

- (G1) For all x, y, z : $(x \circ y) \circ z = x \circ (y \circ z)$.
- (G2) For all x : $x \circ e = x$.
- (G3) For all x there is a y such that: $x \circ y = e$.

The following are groups: $\langle \mathbb{Z}, +, 0 \rangle$ and $\langle \mathbb{R}, +, 0 \rangle$. The following are not: $\langle \mathbb{N}, +, 0 \rangle$ and $\langle \mathbb{R}, \cdot, 1 \rangle$.

Here is a theorem about groups.

Theorem 1 *For every x , there is a y such that: $y \circ x = e$.*

The axioms only directly mention a right inverse, but the above claims that left inverses also exist.

Proof By (G3), there is a y such that (1) $x \circ y = e$ and a z such that (2) $y \circ z = e$. Taking associativity (G1) into account, we have:

$$\begin{aligned}
 & y \circ x \\
 = & \{ \text{(G2)} \} \\
 & y \circ x \circ e \\
 = & \{ \text{(2)} \} \\
 & y \circ x \circ y \circ z \\
 = & \{ \text{(1)} \} \\
 & y \circ e \circ z \\
 = & \{ \text{(G2)} \} \\
 & y \circ z \\
 = & \{ \text{(2)} \} \\
 & e \quad \square
 \end{aligned}$$

This is very similar to the proofs we have been studying during the semester.

This example highlights many of the issues we want to address. In mathematics, we study the properties of various objects, *e.g.*, groups. The properties that these objects enjoy are captured with “non-logical” axioms, *e.g.*, in the case of group theory, (G1)-(G3). The theory of groups consists of all theorems that are derivable from the “non-logical axioms” via logical reasoning alone. This separation is really fundamental. We cannot appeal to intuition or “obvious truths” about groups (or geometry, or the reals, or programming, etc).

So, what exactly is a “proof”, then? A proof is something that preserves truth and is so clear and unambiguous that it can be checked by a machine. By the way, checking proofs is a lot simpler than developing a theorem proving systems like ACL2, because ACL2 tries to also *discover* proofs.

Other questions naturally arise. When we prove theorems about groups, then the results apply to every instance of a group, *e.g.*, $\langle \mathbb{Z}, +, 0 \rangle$ and $\langle \mathbb{R}, +, 0 \rangle$, but if some formula φ holds in every group (denoted $\{(G1), (G2), (G3)\} \models \varphi$), then does there necessarily exist a proof (denoted $\{(G1), (G2), (G3)\} \vdash \varphi$)? Note that proofs are finite, machine checkable things, whereas there are many groups; how many? The answer is an uncountable number. In fact, remember what we saw previously, that some “collections” are too large to be a set? The collection of groups is too large to be a set, too!

We can make this question precise by fixing a proof theory, like we did with ACL2. Then, we will see that for any set of sentences Φ and any sentence φ , $\Phi \models \varphi$ iff $\Phi \vdash \varphi$, (where $\Phi \vdash \varphi$ denotes that there is a proof of φ from Φ). This is Gödel's completeness theorem, perhaps the most important result in logic, as it relates syntax with semantics.

4 Syntax of FOL

When one presents a mathematical language to a mature audience, one starts with the syntax and then the semantics. The syntax tells us what markings, what sequences of symbols, belong to the language. If we were to think about programming languages, this corresponds to the syntax checker. We will insist that the problem of checking whether a sequence of symbols is syntactically well-formed is decidable, that is there exists a program that can say “yes” or “no” when presented with a sequence of symbols. Syntax can be presented using BNF or any other precise method. When presenting the syntax, there is no need to mention the meaning or semantics of the strings; all that we do is that we determine what is and what is not a “statement” in the language. The semantics, or meaning, is given later.

We do not want to look at a specific language, instead, we want to describe the syntax of any first-order language (FOL). All FOLs have the following in common.

Definition 1 *A contains the following symbols:*

1. $x, y, z \dots$ (variables, which are sometimes subscripted);
2. $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$ (Boolean connectives);
3. \forall, \exists (quantifiers);
4. $=$ (equality);
5. $), ($ (parenthesis);

Depending on the first-order theory (FOT) in question, there may be other symbols in a FOL, *e.g.*, in the theory of groups we had \circ , a 2-ary function symbol and e , a constant. In set theory we have \in , a 2-ary relation symbol, and so on.

Definition 2 *The symbol set S of a FOL contains*

1. for every $n \geq 1$ a (possibly empty) set of n -ary relation symbols.
2. for every $n \geq 1$ a (possibly empty) set of n -ary function symbols.
3. a (possibly empty) set of constant symbols.

S may be empty and the symbols mentioned in the definition of S must be distinct from each other and from the symbols in \mathcal{A} . S determines a FOL and $\mathcal{A}_S := \mathcal{A} \cup S$ is the alphabet of this language.

We shall use the letters P, Q, R, \dots for relation symbols, f, g, h, \dots for function symbols, and c, c_0, c_1, \dots for constants.

4.1 Terms

We now need a way to name objects in our universe and to make statements about such objects. That is what terms and formulas are for.

FOLs are interpreted over structures, *e.g.*, in the FOL of groups, \circ corresponds to group multiplication say of group G . Terms are expressions that denote elements of G . Formulas are expressions that make statements about G , *e.g.*, that all elements of a certain type have a certain property.

Consider an analogy with English.

Bill, the father of John, etc. all denote elements in our universe. Similarly, $x, c, f(x, y)$, etc. denote elements of the universe of a first-order theory.

Similarly, statements such as “Bob has three siblings” are statements about the universe. They are either true or false. That is the role played by formulas.

Definition 3 *The set of S -terms is the least set closed under the following rules.*

1. Every variable is an S -term.
2. Every constant in S is an S -term.
3. If t_1, \dots, t_n are S -terms and f is an n -ary function symbol in S , then $f(t_1, \dots, t_n)$ is an S -term.

4.2 Formulas

Definition 4 *An atomic formula of S is either of the form $t_1 = t_2$ or $R(t_1, \dots, t_n)$, where t_1, t_2, \dots, t_n are S -terms and R is an n -ary relation symbol in S .*

Definition 5 *The set of S -formulas is the least set closed under the following rules.*

1. Every atomic formula is an S -formula.
2. If φ, ψ are S -formulas and x is a variable, then $\neg\varphi$, $(\varphi \vee \psi)$, $\langle \exists x :: \varphi \rangle$, and $\langle \forall x :: \varphi \rangle$ are S -formulas.

We will see that $\langle \exists x :: \varphi \rangle$ iff $\neg\langle \forall x :: \neg\varphi \rangle$ and $\langle \forall x :: \varphi \rangle$ iff $\neg\langle \exists x :: \neg\varphi \rangle$, thus \exists and \forall can be defined in terms of each other. Also, the only Boolean connectives we need are \neg and \vee , as the other connectives can be defined in terms of these. How?

If we do use other Boolean connectives, then remember that they can be expanded into \neg and \vee s.

Are the following formulas?

- $\langle \forall x :: \circ(e, x) = e \rangle$ (yes)
- $\langle \forall x :: \circ(e, y) = e \rangle$ (yes)
- $\langle \forall x :: \circ(e, y) = e \rangle \vee \langle \forall x \circ(e, y) = e \rangle$ (no: parens)
- $\langle \forall x :: \langle \exists x :: \circ(x, x) = x \rangle \rangle$ (yes)
- $\langle \forall x :: \langle \exists x :: (\circ(x, x) \vee x) \rangle \rangle$ (no: \vee applied to formulas)

Is there a string that is both a formula and a term? (No)

Can you think of a formula that can be parsed in more than one way? (No)

Formulas without free variables are called *sentences*.

5 Semantics of FOL

We will now go beyond the grammatical, syntactic aspects of FOL—such as the notions of *free*, *term*, and *formula*—to discuss what terms and formulas mean.

Here is an example of something that isn't syntactic: what does $\langle \forall x :: R(x, y) \rangle$ mean? Well, it depends on what R means, *i.e.*, what relation is it and over what domain? and what x means, *i.e.*, what element of the domain is it? Say that R is $<$ on \mathbb{N} and x is 0, then the statement is false. If R is \geq , then it is true.

5.1 Structures and Interpretations

Definition 6 An S -structure is a pair $\mathbf{U} = \langle A, \mathbf{a} \rangle$, where A is a non-empty set, the domain or universe, and \mathbf{a} is a function with domain S such that:

1. If $c \in S$ is a constant symbol, then $\mathbf{a}.c \in A$.
2. If $f \in S$ is an n -ary function symbol, then $\mathbf{a}.f : A^n \rightarrow A$.
3. If $R \in S$ is an n -ary relation symbol, then $\mathbf{a}.R \subseteq A^n$.

Instead of $\mathbf{a}.R$, $\mathbf{a}.f$, and $\mathbf{a}.c$ we often write $R^{\mathbf{U}}$, $f^{\mathbf{U}}$, and $c^{\mathbf{U}}$.

Are we done? Can we give a precise meaning to terms and formulas?

What about $\langle \forall x :: <(x, x) \rangle$? (not true in \mathbb{R} nor in \mathbb{N})

What about $\langle \forall x :: \langle \exists y :: <(y, x) \rangle \rangle$? (not true in \mathbb{N} , true in \mathbb{R})

What about our initial example, $\langle \forall x :: <(x, y) \rangle$?

It depends on what y means, so let's go on.

Definition 7 An S -interpretation \mathcal{J} is a pair $\langle \mathbf{U}, \beta \rangle$, where $\mathbf{U} = \langle A, \mathbf{a} \rangle$ is an S -structure and $\beta : \text{Var} \rightarrow A$, is an assignment, a function that assigns values to the variables.

We now define the meaning of any term t in interpretation \mathcal{J} , denoted $\mathcal{J}.t$:

1. If $v \in Var$, then $\mathcal{J}.v = \beta.v$.
2. If $c \in S$ is a constant symbol, then $\mathcal{J}.c = c^{\mathbf{U}}$.
3. If $f(t_1, \dots, t_n)$ is a term, then $\mathcal{J}(f(t_1, \dots, t_n))$ is $(f^{\mathbf{U}})(\mathcal{J}.t_1, \dots, \mathcal{J}.t_n)$.

Let's look at an example. If $S = S_{gr}$ and $\mathcal{J} = \langle \mathbf{U}, \beta \rangle$, where $\mathbf{U} = \langle \mathbb{Z}, +, 0 \rangle$ and $\beta.x = 2, \beta.y = 4$, then what is $\mathcal{J}(\circ(x, \circ(e, y)))$?

$$\begin{aligned}
 &= +^{\mathbb{Z}}(\mathcal{J}.x, \mathcal{J}(\circ(e, y))) \\
 &= \beta.x + +^{\mathbb{Z}}(e^{\mathbb{Z}}, \mathcal{J}.y) \\
 &= 2 + (0 + \beta.y) \\
 &= 2 + (0 + 4) \\
 &= 6
 \end{aligned}$$

If β is an assignment, then $\beta_x^a(y)$ is a if $y = x$ and $\beta.y$ otherwise. For $\mathcal{J} = \langle \mathbf{U}, \beta \rangle$, \mathcal{J}_x^a denotes $\langle \mathbf{U}, \beta_x^a \rangle$.

We now define what it means for an interpretation to satisfy a formula.

1. $\mathcal{J} \models (t_1 = t_2)$ iff $\mathcal{J}.t_1 = \mathcal{J}.t_2$.
2. $\mathcal{J} \models R(t_1, \dots, t_n)$ iff $\langle \mathcal{J}.t_1, \dots, \mathcal{J}.t_n \rangle \in R^{\mathbf{U}}$.
3. $\mathcal{J} \models \neg\varphi$ iff not $\mathcal{J} \models \varphi$.
4. $\mathcal{J} \models (\varphi \vee \psi)$ iff $\mathcal{J} \models \varphi$ or $\mathcal{J} \models \psi$.
5. $\mathcal{J} \models \langle \exists x :: \varphi \rangle$ iff for some $a \in A$, $\mathcal{J}_x^a \models \varphi$.
6. $\mathcal{J} \models \langle \forall x :: \varphi \rangle$ iff for any $a \in A$, $\mathcal{J}_x^a \models \varphi$.

If $\mathcal{J} \models \varphi$ we say that φ holds in \mathcal{J} ; we also say that \mathcal{J} is a model of φ ; we also say that \mathcal{J} satisfies φ .

Given, Φ , a set of formulas, $\mathcal{J} \models \Phi$ (\mathcal{J} is a model of Φ) iff for every $\varphi \in \Phi$, $\mathcal{J} \models \varphi$.

You should convince yourself that $\mathcal{J} \models \varphi$ iff φ is *true* under interpretation \mathcal{J} .

Let's look at an example. If $S = S_{gr}$ and $\mathcal{J} = \langle \mathbf{U}, \beta \rangle$, where $\mathbf{U} = \langle \mathbb{Z}, +, 0 \rangle$ and $\beta.x = 2, \beta.y = 4$, as before, then what is the value of $\mathcal{J} \models \langle \forall x :: \langle \exists y :: \circ(x, e) = y \rangle \rangle$?

$$\begin{aligned}
 &\mathcal{J} \models \langle \forall x :: \langle \exists y :: \circ(x, e) = y \rangle \rangle \\
 &\text{iff for all } i \in \mathbb{Z}, \mathcal{J}_x^i \models \exists y \circ xe = y \\
 &\text{iff for all } i \in \mathbb{Z}, \text{ there is a } j \in \mathbb{Z} \text{ such that } (\mathcal{J}_x^i)_y^j \models \circ xe = y \\
 &\text{iff for all } i \in \mathbb{Z}, \text{ there is a } j \in \mathbb{Z} \text{ such that } (\mathcal{J}_x^i)_y^j(\circ xe) = (\mathcal{J}_x^i)_y^j(y) \\
 &\text{iff for all } i \in \mathbb{Z}, \text{ there is a } j \in \mathbb{Z} \text{ such that } \circ^{\mathbf{U}}((\mathcal{J}_x^i)_y^j(x), (\mathcal{J}_x^i)_y^j(e)) = j \\
 &\text{iff for all } i \in \mathbb{Z}, \text{ there is a } j \in \mathbb{Z} \text{ such that } i + e^{\mathbf{U}} = j \\
 &\text{iff for all } i \in \mathbb{Z}, \text{ there is a } j \in \mathbb{Z} \text{ such that } i + 0 = j \\
 &\text{true, set } j \text{ to } i
 \end{aligned}$$

Note that the meaning of a sentence does not depend on the assignment. In general, we are interested in sentences, but to evaluate them, we have to evaluate subformulas, which may not be sentences, therefore, the need for assignments. This kind of thing comes up in programming a lot.

Using the notion of satisfaction, we define the notion of consequence.

Definition 8 *Let Φ be a set of formulas and φ a formula. Then $\Phi \models \varphi$ (φ is a consequence of Φ) iff for every interpretation, \mathcal{J} , which is a model of Φ , we have that $\mathcal{J} \models \varphi$.*

We have developed enough mathematical machinery to reconsider, in a more rigorous way, one of our initial goals. Recall, that we were interested in whether $\Phi \models \varphi$ iff $\Phi \vdash \varphi$. For example, we saw a proof that groups have a left inverse, i.e., $\Phi_{gr} \vdash \langle \forall x :: \langle \exists y :: (y \circ x) = e \rangle \rangle$, and you should be convinced that such a proof implies $\Phi_{gr} \models \langle \forall x :: \langle \exists y :: (y \circ x) = e \rangle \rangle$, where $\Phi_{gr} = \{ \langle \forall x :: \langle \forall y :: \langle \forall z :: (x \circ y) \circ z = x \circ (y \circ z) \rangle \rangle \rangle, \langle \forall x :: x \circ e = x \rangle, \langle \forall x :: \langle \exists y :: x \circ y = e \rangle \rangle \}$.

What is not as clear is whether the opposite direction holds. Gödel's completeness theorem establishes this.

Theorem 2 *For all Φ and φ , $\Phi \vdash \varphi$ iff $\Phi \models \varphi$.*

6 Gödel's First Incompleteness Theorem

Here is a variant of Gödel's incompleteness theorem, what is widely considered to be the major intellectual contribution of the twentieth century. A set S is *recursive* iff there is a program (in Scheme, say) that for any input returns yes or no, depending on whether the input is an element or not. Assuming $\text{Con}(\text{ZF})$ (that ZF is consistent), the set $\{ \varphi : \text{ZF} \vdash \varphi \}$ is not recursive. (Why do we assume $\text{Con}(\text{ZF})$? Otherwise, all formulas follow from ZF.) More generally, for any consistent extension C of ZF, we have $\{ \varphi : C \vdash \varphi \}$ is not recursive. We will not prove this, but it should be intuitively clear: we can embed programming languages in set theory and we can write a formula that holds iff the program it encodes terminates.

Theorem 3 *(Gödel's first incompleteness theorem.) If C is a recursive consistent extension of ZF, then it is incomplete, i.e., there is a formula φ such that $C \not\vdash \varphi$ and $C \not\vdash \neg\varphi$.*

Proof Outline: If not, then for every φ , either $C \vdash \varphi$ or $C \vdash \neg\varphi$. We can now decide $C \vdash \varphi$: enumerate all proofs of C . Stop when a proof for φ or $\neg\varphi$ is found. \square

In ZF, the axiom of choice is neither provable nor refutable. In ZFC, the continuum hypothesis is neither provable nor refutable. By Gödel's first incompleteness theorem, no matter how we extend ZFC, there will always be sentences which are neither provable nor refutable.