MAU11602
Lecture 13
2026-02-19

Soundness and completeness

Is every theorem a tautology?
Is every tautology a theorem?
Every theorem is a tautology, since each of our rules of inference is sound, i.e. can never derive a statement which is not a tautology from statements which are.
Not every (classical) tautology is a theorem though. The following two statements are both tautologies, but only the first of them is a theorem:

$$(p \rightarrow (q \rightarrow \bot)) \rightarrow (q \rightarrow (p \rightarrow \bot))$$
$$((q \rightarrow \bot) \rightarrow p) \rightarrow ((p \rightarrow \bot) \rightarrow q)$$

Using the negation symbol, these are two different forms of the contrapositive rule, $(p \rightarrow \neg q) \rightarrow (q \rightarrow \neg p)$ and $(\neg q \rightarrow p) \rightarrow (\neg p \rightarrow q)$.
The first is actually a special case of the more general tautology $(p \rightarrow (q \rightarrow r)) \rightarrow (q \rightarrow (p \rightarrow r))$ and can proved without using the principle of explosion while the second one can't be proved with our current rules of inference.

Different logics

There is more than one version of zeroeth order logic.
- Our orginal rules of inference, without explosion, give what's called *minimal logic*.
- Those rules plus explosion give what's called *intuitionistic logic*.
- To get *classical logic* we need to add a further rule of inference.

There are a number of choices for this extra rule of inference, but one choice is

$$\frac{\Gamma, P \vdash R \qquad \Gamma, P \to \bot \vdash R}{\Gamma \vdash R}$$

In other words, if we can prove a statement R in a context Γ by showing that it holds regardless of whether some other statement P or its negation holds.
The corresponding logical principle is usually called the *law of the excluded middle*.
There's no obvious counterpart on the type theory side for this rule of inference though.
The logic governing booleans in programming is classical logic but the logic governing types of expressions seems to be intuitionistic or minimal, depending on whether you like functions with empty domain or not.

Quantifiers

The language developed so far for elementary arithmetic has the the useful feature that all statements can be mechanically checked.
Unfortunately there are many statements we'd like to make which can't be checked mechanically, like

$$\forall e.\, e < 0 \vee \exists a.\, \exists b.\, \exists c.\, \exists d.\, a \cdot a + b \cdot b + c \cdot c + d \cdot d = e.$$

Since we're leaving the realm of computation I'm switching to more standard mathematical notation, e.g. $\cdot$ for multiplication rather than `*` and $\vee$ for disjunction rather than `orelse`.
The $\forall$ and $\exists$ symbols are quantifiers, usually read as "for all" and "there exists".
So the statement above says that every non-negative integer can be written as a sum of four squares.
This is true, and provable with an appropriate set of axioms and rules of inference for arithmetic, but there can be no algorithm for checking such statements.

Quantifiers, continued

Grammatically, quantifiers behave like the λ used for defining anonymous functions. The typing rules are more restrictive though:

$$\frac{x\colon \sigma \vdash e\colon \texttt{bool}}{\forall x.\, e\colon \texttt{bool}} \qquad \frac{x\colon \sigma \vdash e\colon \texttt{bool}}{\exists x.\, e\colon \texttt{bool}}$$

For λ expressions the corresponding rule is

$$\frac{x\colon \sigma \vdash e\colon \tau}{\lambda x.\, e\colon \tau}$$

The real difference though is in evaluation. We can't just write reduction rules for quantifiers like we did λ expressions and all the other language features so far.
The rule for evaluating $\forall x.\, e$, assuming the variable x has type `int`, involves substituting each integer value for all free occurences of x in the expression e and then returning false if any of them evaluated to false and true otherwise.
This isn't an algorithm though, because there are infinitely many things to check.
At least the rules substitution and for free and bound variables are the same though.

First order logic

Some statements involving quantifiers, like

$$\forall e.\, \exists a.\, \exists b.\, \exists c.\, \exists d.\, e < 0 \vee a \cdot a + b \cdot b + c \cdot c + d \cdot d = e.$$

require a detailed analysis of the quantified expressions to evaluate but others, like

$$((\forall n.\, (n \cdot n < 17)) \wedge (\exists n.\, (n \cdot n < 17) \rightarrow (n \cdot (n + 1) > 42))) \rightarrow (\exists n.\, (n \cdot (n + 1) > 42))$$

don't.
The second statement is true, but it would also be true if we substituted any other boolean expressions with a free variable n of integer type in place of $n \cdot n < 17$ and $n \cdot (n + 1) > 42$.
The first statement is true, but would not necessarily be true if we substituted some other boolean expression with free variables a, b, c, d, and e for $e < 0 \vee a \cdot a + b \cdot b + c \cdot c + d \cdot d = e$.

First order logic, continued

Just as zeroeth order logic is the study of statements connected by conjunction, disjunction, and implication, without examining the internal structure of those statements, first order logic is the study of statements connected by universal and existential quantifiers, conjunction, disjunction, and implication, without examining the internal structure of those statements.

From the point of view of first order logic the second statement on the previous slide is true because all statements of the form

$$((\forall n.\, p(n)) \wedge (\exists n.\, p(n) \to q(n))) \to (\exists n.\, q(n))$$

are true.

This doesn't even require the variable n to have integer type.

A choice of set for the quantifiers to range over is called a *domain*. The p and q above are called *predicates*. These are unary predicates because they involve only one variable, but we could have n-ary predicates for any n.

Interpretations, validity

An *interpretation* of an expression in first order logic is a choice of domain togther with a choice for each predicate and each combination of assignments of variables in that predicate to members of the domain, of a boolean value.
Having chosen and interpretation we can say whether a statement is true or not. For most statements this will depend on which interpretation was chosen.
An interpretation for the statement

$$((\forall n.\, p(n)) \wedge (\exists n.\, p(n) \rightarrow q(n))) \rightarrow (\exists n.\, q(n))$$

is the one which takes n to range over the integers and takes $p(n)$ to be true when $n \cdot n < 17$ and $q(n)$ to be true when $n \cdot (n+1) > 42$.
A statement in first order logic which is true under all interpretations is called *valid*.
Valid statements in first order logic play are like tautologies in zeroeth order logic.
We can then say that

$$((\forall n.\, (n \cdot n < 17)) \wedge (\exists n.\, (n \cdot n < 17) \rightarrow (n \cdot (n+1) > 42))) \rightarrow (\exists n.\, (n \cdot (n+1) > 42))$$

is true because it's an instance of the statement above, and that statement is valid.

First order logics

Zeroeth order logic isn't really a single logical system but a collection of related ones.
There are some presentational differences, e.g. do we have constants for true and false, and which logical operators are regarded as basic?
There are also more substantive differences, e.g. is our logic minimal, intuitionistic, or classical?
This is even more true of first order logic.
The main new presentational choices are:
- Is equality part of our language?
- Do we have a notation for values in the domain?
- Do we have a notation for functions in the domain?

The main new substantive differences are:
- Do we implicitly assume all domains are non-empty?
- Do we implicitly assume every expression can be evaluated, e.g. if our version of first order logic has functions, could we safely substitute x/y for $f(x,y)$?

Properties of first order logics

Unless we make a mistake somewhere, all versions of first order logic should be *consistent*, so no statement and its negation are both theorems.

Unless we make a mistake, the theory should be sound, i.e. all theorems are actually true in any interpretation, i.e. are valid.

Unless we make a mistake, the theory should be complete, i.e. all valid statements are actually theorems.

For non-classical versions of the theory we many need to extend the notion of interpretation to make this true.

Validity is *semi-decidable*. There is an algorithm which takes a statement and will, if the statement is valid, eventually generate a proof. If the statement is invalid it might produce a refutation, i.e. an interpretation which makes it false, but it might also just run forever.

Classical zeroeth order logic, by contrast, is *decidable*. There is an algorithm which takes a statement and will generate a proof it was given a tautology and a refutation, i.e. values of the variables which make the statement false, otherwise.

Inference rules for $\forall$ (informal)

Just as we had introduction and elimination rules for $\wedge$, $\vee$ and $\rightarrow$ there are introduction and elimination rules for $\forall$ and $\exists$.

We should be able to infer a universal rule from an arbitrary example. The informal version of this looks like "Suppose n is an integer. Then ... so $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$. Since n was arbitrary we conclude that $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$ for all integers n."

When we say arbitrary, what we mean is that n does not appear in the context, so none of our statements about it depend on any hidden assumptions.

From a universal rule we should also be able to derive specific instances. For example, once we have $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$ for all n we should be able to derive examples like $\sum_{i=0}^{7} i = \frac{7\cdot(7+1)}{2}$ or $\sum_{i=0}^{m+1} i = \frac{(m+1)(m+1+1)}{2}$ from it.

This involves substitution, so all of our rules for capture avoiding substitution apply.

There may be an additional requirement to ensure that the expression being substituted is evaluable if the language allows unevaluable expressions, e.g. division by zero.

Inference rules for $\exists$ (informal)

From a single example we should be able to deduce existence. For example, from the specific case $\sum_{i=0}^{7} i = \frac{7\cdot(7+1)}{2}$ we should be able to conclude that there is an n such that $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$.

We should also be able attach a name to an object known to exist. The name should be fresh, to avoid any assumptions on the object beyond those asserted by the existence statement.

The name is fresh, but the object itself needn't be. It might already have one or more names. It's just that we aren't allowed to assume it does.

In particular, we can apply this rule more than once, each time with a fresh variable.

This is one of the reasons validity is decidable in zeroeth order logic but only semi-decidable in first order logic.