Lecture notes 1.0

Introduction

COMP 2411, session 1, 2004

Logical truth (1)

Consider the following statements.

1. If John has been waiting for the bus for more than 10 minutes and it is raining, then John is in an extremely bad mood.
2. If John is in an extremely bad mood whenever he has been waiting for the bus for more than 10 minutes and if John is not in an extremely bad mood, then John has not been waiting for the bus for more than 10 minutes.
3. If John is in an extremely bad mood whenever he has been waiting for the bus for more than 10 minutes and if John has been waiting for the bus for more than 10 minutes, then someone is in an extremely bad mood.

Logical truth (2)

The first statement might be true, but it has no logical justification.

The second statement is true, which can be justified in the propositional calculus: $((p \rightarrow q) \wedge \neg q) \rightarrow \neg p$ is valid (will be explained soon).

The third statement is true, which cannot be justified in the propositional calculus, but which can be justified in the predicate calculus: $((p(a) \rightarrow q(a)) \wedge p(a)) \rightarrow \exists x q(x)$is valid (will be explained a bit later).

The core of the logic part of the course is devoted to logical justifications in the propositional and in the predicate calculus.

Aristotle

The Greek philosopher Aristotle developed a study of Logic in a group of works collectively known as The Organon, written around 350 BC.

The object of Aristotle's Logic is to discover the laws that govern valid reasoning, as observed in deductions.

> *A deduction is speech in which, certain things having been supposed, something different from those supposed results of necessity because of their being so.* (*Prior Analytics I.2, 24b18-20*)

A syllogism is, in Aristotle's theory, one of the laws that govern valid reasoning.

Syllogisms

The following is the most famous instance of a syllogism:

Premises	All men are mortal	Socrates is a man
Conclusion	Socrates is mortal	

Taken together, syllogisms constitute a theory of inference.

In Aristotle's theory, reasoning means applying syllogisms to a set of premises until the desired conclusion is reached.

Aristotle's notion of inference is good, but the syllogisms do not account for every kind of valid form of reasoning.

Still, till the first half of the 19th century, the dominant view was that of Kant, who claimed that Aristotle had discovered everything there was to know about logic.

Leibniz's dream (1)

The mathematician and philosopher Leibniz was probably not too impressed by Aristotle's work when he wrote in 1667 what is referred to today as Leibniz's dream.

> *If we could find characters or signs appropriate for expressing all our thoughts as definitely and as exactly as arithmetic expresses numbers or geometric analysis expresses lines, we could in all subjects in so far as they are amenable to reasoning accomplish what is done in Arithmetic and Geometry.*
> *For all inquiries which depend on reasoning would be performed by the transposition of characters and by a kind of calculus, which would immediately facilitate the discovery of beautiful results.*

Leibniz's dream (2)

> *And if someone would doubt my results, I should say to him: "Let us calculate, Sir," and thus by taking to pen and ink, we should soon settle the question. Now the characters which express all our thoughts will constitute a new language. . . this language will be. . . very easy to learn. It will be quickly accepted by everybody on account of its great utility and its surprising facility, and it will serve wonderfully in communication among various peoples.*
> *There will be no equivocations or amphibolies, and everything which will be said intelligibly in the language will be said with propriety.*

Towards modern logic

Using modern concepts, we can interpret some of Leibniz claims or hopes as follows.

- Natural languages are messy, and the existence of valid forms of reasoning presupposes the existence of a clean formal language on which they can operate.
- A key property of logical inferences is that they can be performed effectively by a computer, as least in principle. Reasoning is running some program.

Frege

Leibniz wanted to embed Logic into Mathematics, by reducing the former to that part of the latter that deals with calculations (like solving differential equations, transforming algebraic equations).

The mathematician and philosopher Frege reversed the approach and tried to embed Mathematics into Logic.

In the Begriffsschrift, written in 1879, Frege invented the predicate calculus.

It is a remarkable fact that the whole of Mathematics can—but only in principle—be written in the language of the predicate calculus.

Frege also formalized the notion of proof, which is basically a counterpart to the Aristotelian theory of inference.

Proofs (1)

$$\dfrac{\dfrac{\forall y(0+y=y)}{0+s(0)=s(0)} \quad \dfrac{\forall x \forall y(s(x)+y=s(x+y))}{s(0)+s(0)=s(0+s(0))}}{s(0)+s(0)=s(s(0))}$$

is a proof that $1+1=2$, involving:

- the axiom stating that for all numbers y, $0+y=y$;
- the axiom stating that for all numbers x and y, $(x+1)+y=(x+y)+1$;
- 2 applications of the inference rule stating that if every number has property P, then number n has property P;
- 1 application of the inference rule stating that if n and n' are equal numbers and n has property P, then n' has property P.

Proofs (2)

More generally, a proof derives a conclusion from axioms, by applying rules of inference (a counterpart to the Aristotelian syllogisms) to axioms and formulas previously derived.

A proof can be represented as a labeled tree.

- The labels of the leaves are axioms.
- The label of the root is the conclusion.
- An internal node is labeled with the formula obtained by applying some inference rule to the formulas labeling the children of the node.

Axiomatisations

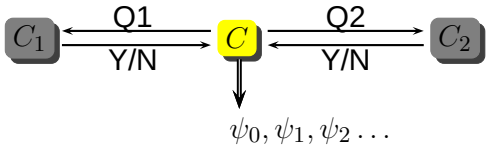
$1+1=2$ is more 'straightforward' than the fact that for all numbers x and y, $(x+1)+y=(x+y)+1$. So proving the former from the latter looks futile. Understanding that it is not futile is a great step towards understanding Logic...

The fundamental question after Frege's work was to try and come up with a 'reasonable' set of axioms and a 'reasonable' set of inference rules that would allow to derive all true arithmetic statements, $1+1=2$ as well as Fermat's theorem, expressed as formulas of the predicate calculus.

Such a set of axioms and inference rules would be a complete axiomatisation of arithmetics.

From Logic to Computation (1)

A complete axiomatisation would enable three computers C_1, C_2 and C to work as follows.



C would output $\psi_0, \psi_1, \psi_2 \ldots$, an enumeration of all true arithmetic statements, asking questions of the form:

- Q1: Is formula φ an axiom?
- Q2: Is formula φ the conclusion of a rule whose premises are among $\psi_0, \ldots, \psi_n$?

From Logic to Computation (2)

The previous picture is in accordance with Leibniz's intuition that logical inferences can be performed effectively...

...a notion that had to be defined formally in order to make the previous picture meaningful and find out whether a complete axiomatization of arithmetics is possible.

The efforts of Turing, Post, von Neuman, Gödel and others yielded:

- an abstract model of computation, the Turing machine;
- a theory of computable functions;
- computers;
- a proof that a complete axiomatization of arithmetics is impossible.

Declarative programming

Most programming languages are designed to instruct the computer in how to solve a problem. They belong to the paradigm of procedural or imperative programming.

A declarative programming language is designed to describe a problem, and let the computer find out how to solve it...at least in principle.

Suppose we want to concatenate $[1, 2, 3]$ and $[a, b]$. We would like to input

$$\texttt{concatenate}\ [1, 2, 3]\ \texttt{and}\ [a, b]$$

and just get the answer...but for that to be possible, the computer must know what we mean by *concatenate*.

Meaning of concatenate

The following captures the meaning of *concatenate*.

- For all lists y, y concatenated to the end of the empty list is identical to y.
- For all lists x and y, if x is nonempty then y concatenated to the end of x is identical to the list whose head is x's head, and whose tail is identical to y concatenated to the end of x's tail.

Translation into predicate calculus

The meaning of *concatenate* can be expressed as formulas of the predicate calculus:

(1) $\forall y(\mathrm{cat}(\epsilon, y) = y)$

(2) $\forall x \forall y \forall z \forall h \forall v((x = [h|v] \wedge \mathrm{cat}(v, y) = z) \rightarrow \mathrm{cat}(x, y) = [h|z])$

in the same way as

(1') $\forall y(0 + y = y)$ and

(2') $\forall v \forall y(s(v) + y = s(v + y))$

can be viewed as axioms about natural numbers and addition. (Note the formal analogy between (1')–(2') compared to (1)–(2).)

Proving existential statements

If the computer had reasoning abilities, *i.e.*, built-in inference rules, it should be able to prove from (1) and (2) that $\mathrm{cat}([1,2,3],[a,b]) = [1,2,3,a,b]$, like proving from (1') and (2') that $3 + 2 = 5$.

But proving $\mathrm{cat}([1,2,3],[a,b]) = [1,2,3,a,b]$ from (1) and (2) is not really interesting, since we want to provide $\mathrm{cat}([1,2,3],[a,b])$ as input and get $[1,2,3,a,b]$ as output.

We can ask the computer to try and prove a statement seemingly weaker than $\mathrm{cat}([1,2,3],[a,b]) = [1,2,3,a,b]$, namely:

`does there exist a list which is equal to` $[a,b]$ `concatenated to the end of` $[1,2,3]$`?`

Constructive proofs

Formally, we would ask the computer to prove:

$$\star \quad \exists z \, \mathrm{cat}([1,2,3],[a,b]) = z$$

It turns out the computer can prove $\star$ from (1) and (2) if and only if it can compute a witness for z, namely, compute the (necessarily unique) list $[1,2,3,a,b]$ such that:

$$\mathrm{cat}([1,2,3],[a,b]) = [1,2,3,a,b]$$

(provided that it has the right inference rules...).

In other words, the computer can produce a constructive proof.

'Typical' dialogue

So we can have the following dialogue.

User Try and prove that there exists a list which is equal to $[a,b]$ concatenated to the end of $[1,2,3]$.

Computer OK... Done, I've got a proof.

User In that case, you must have discovered more, you must actually know which list is equal to $[a,b]$ concatenated to the end of $[1,2,3]$. Would you be kind enough to tell me?

Computer No problem, mate. Indeed, I found out that $[a,b]$ concatenated to the end of $[1,2,3]$ is equal to the list $[1,2,3,a,b]$.

Nonconstructive proofs

Not all proofs are constructive. Consider for instance the following proof that there exists 2 irrational numbers a and b such that a^b is rational.

Remember that $\sqrt{2}$ is irrational.

If $\sqrt{2}^{\sqrt{2}}$ is rational, then we are done, taking $a = b = \sqrt{2}$.

If $\sqrt{2}^{\sqrt{2}}$is irrrational, then since $(\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = \sqrt{2}^2 = 2$, we are also done, taking $a = \sqrt{2}^{\sqrt{2}}$ and $b = \sqrt{2}$.

So we have shown the existence of irrational numbers a and b such that a^b is rational, but we have not exhibited irrational numbers a and b such that a^b is rational.

Concatenating lists in Prolog

The two axioms that capture the meaning of list concatenation are expressed as follows in Prolog.

(1^*) `cat([], Y, Y).`
(2^*) `cat([H|T], Y, [H|Z]) :- cat(T, Y, Z).`

The dialogue between user and computer takes the following form:

```
?- cat([1,2,3],[a,b],L).        (input)
L = [1, 2, 3, a, b]             (output)
```


One program, many queries (1)

Still using (1^*) and (2^*) only, the computer can answer many other queries.

E.g., we can ask the computer to try and prove that there exists a list L that, concatenated to the end of $[1, 2, 3]$, is equal to $[1, 2, 3, a, b]$:

```
?- cat([1,2,3],L,[1,2,3,a,b]).        (input)
L = [a, b]                            (output)
```


One program, many queries (2)

Or we can ask the computer to try and prove that there exists two lists L_1 and L_2 whose concatenation is equal to $[1, 2, 3, a, b]$:

```
?- cat(L1,L2,[1,2,3,a,b]).

L1 = []
L2 = [1, 2, 3, a, b];

L1 = [1]
L2 = [2, 3, a, b] ;

L1 = [1, 2]
L2 = [3, a, b]

Yes
```

A few general statements. . .

- Logic programming is the discipline that applies Logic and provides the theoretical foundations to semi-declarative programming languages.
- Prolog is a semi-declarative programming language, that implements the main concepts of Logic programming.
- Prolog is not the sole semi-declarative programming language, but it was created before the others and is still the most prominent one.
- No programming language is purely declarative, and Prolog is no exception. It is doubtful that a purely declarative programming language can ever be designed.