CS 245: Logic and Computation

Alice Gao

Lecture 2, September 12, 2017

Based on slides by Jonathan Buss, Lila Kari, Anna Lubiw and Steve Wolfman with thanks to B. Bonakdarpour, D. Maftuleac, C. Roberts, R. Trefler, and P. Van Beek

Please come and sit in the front. I won't pick on you.

We will begin when the music stops.

Let's begin!

Announcements

Office hours

Which time slot would you most likely be available?

a. Monday 11am-12pm
b. Monday 1-2pm
c. Monday 2-3pm
d. Tuesday 4-5pm
e. Wednesday 11am-12pm

Previously on CS 245

- The roadmaps of CS 245
- What is logic?
- What are the applications of logic in computer science?
- What is a proposition?
- How do we translate English sentences into compound propositions?

A conjecture that is neither true nor false

Thank you, Christopher Lo for this awesome story!

The Continuum Hypothesis by Georg Cantor: there is no set with cardinality strictly between the integers and the real numbers.

- Cantor failed to prove (or disprove) this conjecture.
- Kurt Godel proved that it cannot be disproved (1940).
- Paul Kohen proved that it cannot be proved (1963).
- Christopher's conclusion: there is no way to know for sure the truth value of this conjecture, or even that there is one.

Translating from English to Propositional Logic

Translate the following sentences to propositional logic formulas.

1. She is clever but not hard working.
2. I will eat an apple or an orange but not both.
3. If he does not study hard, then he will fail.
4. He will fail unless he studies hard.
5. He will not fail only if he studies hard.

English can be ambiguous.

Give multiple translations of the following sentences into propositional logic. Are these translations logically equivalent?

1. Pigs can fly and the grass is red or the sky is blue.

2. If it is sunny tomorrow, then I will play golf, provided that I do not feel stressed.

On an assignment, we may ask you to translate English sentences with ambiguity into propositional logic. I highly recommend that you explain the reasoning behind your solution.

Using propositional logic to model the real world

Consider the following argument, drawn from an article by Julian Baggini (http://www.butterfliesandwheels.org/2004/tu-quoque/). (The onnagata are male actors portraying female characters in kabuki theatre.)

Premise 1: If women are too close to femininity to portray women, then men must be too close to masculinity to play men, and vice versa.

Premise 2: And yet, if the onnagata are correct, women are too close to femininity to portray women and yet men are not too close to masculinity to play men.

Conclusion: Therefore, the onnagata are incorrect, and women are not too close to femininity to portray women.

Translate the two premises and the conclusion into propositional logic. I will post the solution before next class.

Thought question: Is the argument valid? Why or why not?

Learning goals — revisited

By the end of the lecture, you should be able to

- Give a high-level description of logic.
- Give examples of applications of logic in computer science.
- Define propositions.
- Classify English sentences into propositions and non-propositions.
- Give multiple translations of English sentences with ambiguity.
- Translate English sentences with no ambiguity into compound propositions.

Propositional Logic: Syntax

Learning goals

By the end of this lecture, you should be able to:

- Describe the three types of symbols in propositional logic.
- Describe the recursive definition of well-formed formulas.
- Write the parse tree for a well-formed formula.
- Determine and give reasons for whether a given formula is well formed or not.
- Identify the recursive structure in a recursive definition.
- Explain how to use structural induction to prove properties of a recursively defined concept.

Atomic and compound propositions

An *atomic* proposition (also called an atom or an atomic formula) is a statement or an assertion that must be true or false. It is represented by a single propositional variable.

We construct a *compound* proposition by connecting atomic propositions using logical connectives.

Symbols and expressions

Propositions in English are represented by *formulas*.

A formula consists of a string of *symbols*.

There are three kinds of symbols.

Propositional variables: Lowercase Latin letters possibly with subscripts; e.g., p, q, r, p_1, p_2, q_{27}, etc.

Connectives: $\neg$, $\wedge$, $\vee$, $\rightarrow$ and $\leftrightarrow$.

Punctuation: '(' and ')'.

Expressions

An *expression* is a string of symbols.

Examples of expressions:

- α: $(\neg)()\vee pq\rightarrow$
- β: $a \vee b \wedge c$
- γ: $((a \rightarrow b) \vee c)$

Expressions

An *expression* is a string of symbols.

Examples of expressions:

- α: $(\neg)()\vee pq\rightarrow$
- β: $a \vee b \wedge c$
- γ: $((a \rightarrow b) \vee c)$

What does each expression mean? In how many ways can we interpret each expression?

Expressions

An *expression* is a string of symbols.

Examples of expressions:

- α: $(\neg)()\vee pq\rightarrow$
- β: $a \vee b \wedge c$
- γ: $((a \rightarrow b) \vee c)$

What does each expression mean? In how many ways can we interpret each expression?

Ideally, we would like *one and only one way* to interpret each expression.

Can we focus on a set of expressions where each expression in this set has a unique interpretation?

Definition of well-formed formulas

Let $\mathcal{P}$ be a set of propositional variables. We define the set of *well-formed formulas* over $\mathcal{P}$ inductively as follows.

1. A single symbol of $\mathcal{P}$ is well-formed.
2. If α is well-formed, then $(\neg\alpha)$ is well-formed.
3. If α and β are well-formed, then each of $(\alpha \wedge \beta), (\alpha \vee \beta), (\alpha \rightarrow \beta), (\alpha \leftrightarrow \beta)$ is well-formed.
4. Nothing else is a well-formed formula.

Understanding well-formed formulas

Which symbol(s) can appear as the first one in a well-formed formula? Choose the largest set of possible symbols. (p denotes any propositional variable.)

a. $\neg$, $($
b. $\neg$
c. p, $\neg$, $($
d. p, $\neg$
e. p, $($

Let $\mathcal{P}$ be a set of propositional variables. We define the set of *well-formed formulas* over $\mathcal{P}$ inductively as follows.

1. A single symbol of $\mathcal{P}$ is well-formed.
2. If α is well-formed, then $(\neg\alpha)$ is well-formed.
3. If α and β are well-formed, then each of $(\alpha \wedge \beta), (\alpha \vee \beta), (\alpha \rightarrow \beta), (\alpha \leftrightarrow \beta)$ is well-formed.
4. Nothing else is a well-formed formula.

The parse tree of a well-formed formula

A parse tree is another way to represent a well-formed formula. The parse tree makes the structure of the formula explicit.

Write the parse tree of the following well-formed formulas:

1. $((a \vee b) \wedge (\neg(a \wedge b)))$
2. $(((\neg p) \wedge q) \rightarrow (p \wedge (q \vee (\neg r))))$.

Examples of well-formed formulas

Consider each formula below. Determine whether it's well-formed or not. If it is, write its parse tree. If it is not, explain why and describe ways to make it well-formed with minimum changes.

a. $\neg a$

b. $(a \to b)$

c. $(a \wedge b \wedge c)$

d. $(a \to b \to c)$

e. $(a \vee b \wedge c)$

(In case you are bored...) Challenge question: Alas, the instructors need your help coming up with exam questions on well-formed formulas. Create an exam question to test the concept of well-formed formulas. Please provide the question description, the correct answer, and explain what this question is testing. Email it to me (alice.gao@uwaterloo.ca) and I'll share the good ones with everyone.

Unique Readability of Formulas

Does every well-formed formula have a unique meaing? Yes.

Theorem. Every well-formed formula has a unique derivation as a well-formed formula. That is, each well-formed formula has exactly one of the following forms:

(1) an atom, (2) $(\neg\alpha)$, (3) $(\alpha \wedge \beta)$, (4) $(\alpha \vee \beta)$, (5) $(\alpha \rightarrow \beta)$, or (6) $(\alpha \leftrightarrow \beta)$.

In each case, it is of that form in exactly one way.

(Why) Is the Theorem True?

As an example, consider $((p \wedge q) \rightarrow r)$. It can be formed from the two formulas $(p \wedge q)$ and r using the connective $\rightarrow$.

If we tried to form it using $\wedge$, the two parts would need to be "$(p$" and "$q) \rightarrow r$". But neither of those is a formula!

The statement holds for this example.

We will prove this theorem using structural induction (a type of mathematical induction).

How much do you remember about induction?

1. Why and when do we use mathematical induction to prove a theorem?
2. What are the two main steps in an induction proof?
3. In each step, what do you need to prove?

Induction over natural numbers

You may remember proving properties of natural numbers (0, 1, 2, 3, ...) using induction from MATH 135.

Let P be some property. We want to prove that every natural number has property P. That is, $P(0)$, $P(1)$, $P(2)$, $P(3)$, $P(4)$, ..., are all true.

Theorem: $P(k)$ is true where $k = 0, 1, 2, 3, ...$

Induction over natural numbers

Theorem: $P(k)$ is true where $k = 0, 1, 2, 3, \ldots$

Examples of induction proofs that you may remember:

a. Base step: Prove $P(0)$. Inductive step: Consider an arbitrary $k \geq 0$. Assume that $P(k)$ is true. Prove that $P(k+1)$ is true.

b. Base step: Prove $P(0)$. Inductive step: Consider an arbitrary $k \geq 0$. Assume that $P(0)$, $P(1)$, ..., $P(k)$ are true. Prove that $P(k+1)$ is true.

c. Base step: Prove $P(0)$ and $P(1)$. Inductive step: Consider an arbitrary $k \geq 0$. Assume that $P(k)$ is true. Prove that $P(k+2)$ is true.

An incorrect induction proof:

- Base step: Prove $P(0)$. Inductive step: Consider an arbitrary $k \geq 0$. Assume that $P(k)$ is true. Prove that $P(k+2)$ is true.

Structural induction (1)

Step 1: Identify the recursive structure in the problem.

Theorem 1: Every well-formed formula has an equal number of left and right brackets.

Theorem 2: In any binary tree, the number of nodes with two children is one less than the number of leaf nodes.

Theorem 3: The sum of the natural numbers up to n is equal to $n(n+1)/2$.

Structural induction (1)

Step 1: Identify the recursive structure in the problem.

Theorem 1: Every *well-formed formula* has an equal number of left and right brackets.

Theorem 2: In any *binary tree*, the number of nodes with two children is one less than the number of leaf nodes.

Theorem 3: *The sum of the natural numbers up to* n is equal to $n(n+1)/2$.

Structural induction (2)

Step 2: Identify each recursive appearance of the structure inside its definition.

Example 1: A well-formed (propositional) formula is one of

- A single propositional variable,
- $(\neg\alpha)$ where α is a well-formed formula,
- $(\alpha \wedge \beta), (\alpha \vee \beta), (\alpha \rightarrow \beta)$, and $(\alpha \leftrightarrow \beta)$, where α and β are both well-formed formulas.

Example 2: A binary tree is one of

- An empty tree,
- A node with two subtrees, each of which is a binary tree.

Example 3: The sum of the natural numbers up to n is equal to n plus the sum of natural numbers up to $n-1$.

Structural induction (2)

Step 2: Identify each recursive appearance of the structure inside its definition.

Example 1: A well-formed (propositional) formula is one of

- A single propositional variable,
- $(\neg\alpha)$ where α is a *well-formed formula*,
- $(\alpha \wedge \beta), (\alpha \vee \beta), (\alpha \rightarrow \beta)$, and $(\alpha \leftrightarrow \beta)$, where α and β are both *well-formed formulas*.

Example 2: A binary tree is one of

- An empty tree,
- A node with two subtrees, each of which is a *binary tree*.

Example 3: The sum of the natural numbers up to n is equal to n plus *the sum of natural numbers up to* $n-1$.

Structural induction (3)

Step 3: Divide the cases into: those without recursive appearances ("base cases") and those with ("recursive" or "inductive" cases).

Example 1: A well-formed (propositional) formula is one of

- A single propositional variable,
- $(\neg\alpha)$ where α is a well-formed formula,
- $(\alpha \wedge \beta), (\alpha \vee \beta), (\alpha \rightarrow \beta)$, and $(\alpha \leftrightarrow \beta)$, where α and β are well-formed formulas.

Example 2: A binary tree is one of

- An empty tree,
- A node with two subtrees, each of which is a binary tree.

Structural induction (3)

Step 4: Divide the cases into: those without recursive appearances ("base cases") and those with ("recursive" or "inductive" cases).

Example 1: A well-formed (propositional) formula is one of

- A single propositional variable, *(base case)*
- $(\neg\alpha)$ where α is a well-formed formula, *(inductive case)*
- $(\alpha \wedge \beta), (\alpha \vee \beta), (\alpha \rightarrow \beta)$, and $(\alpha \leftrightarrow \beta)$, where α and β are both well-formed formulas. *(inductive case)*

Example 2: A binary tree is one of

- An empty tree, *(base case)*
- A node with two subtrees, each of which is a binary tree. *(inductive case)*

The principle of structural induction

Define $P(\varphi)$ to be some property.

Theorem: For all *recursive structure* φ, $P(\varphi)$ holds.

Proof by structural induction:

Base case: *Prove that the theorem is true for each base case you identified. This should be quite easy.*

Inductive step: *(For each recursive case, write out an inductive step)*

Consider an arbitrary *recursive case structure* φ.

Induction hypothesis: Assume the theorem holds for *each recursive appearance of the structure in this case*. We now show that the theorem holds for φ.

By the principle of structural induction, every recursive structure φ has property P.

Balanced brackets in well-formed formulas

Theorem: Every well-formed formula has an equal number of left and right brackets.

Unbalanced brackets in a proper prefix of a formula

Theorem: Every proper prefix of a well-formed formula has more left brackets than right brackets.

A *proper prefix* of φ is a non-empty expression x such that φ is xy for some non-empty expression y.

Question: Let α and β be well-formed formulas. Let a and b be proper prefixes of α and β respectively. List all the proper prefixes of $(\alpha \wedge \beta)$.

Proving the Unique Readability Theorem

Theorem. Every formula is exactly one of an atom, $(\neg\alpha)$, $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \to \beta)$ or $(\alpha \leftrightarrow \beta)$; and in each case it is of that form in exactly one way.

We want to prove this using structural induction. How will it go?

Example: Consider a formula of the form $(\alpha \to \beta)$. One such is our previous example $((p \wedge q) \to r)$, which has $(p \wedge q)$ for α and r for β.

Can we parse the formula by applying $\wedge$ last?

$$((p \wedge q) \to r) = (\alpha' \wedge \beta') \ ,$$

Then α' is the expression "$(p$" and β' is the expression "$q)\to r$". Fortunately, neither α' nor β' is a formula. Why?

How can we make sure that this argument works for every formula?

Proving Unique Readability

Property $P(\varphi)$:

A formula φ has property P iff it satisfies all three of the following.

A: The first symbol of φ is either '(' or a variable.

B: φ has an equal number of '(' and ')', and each proper prefix of φ has more '(' than ')'.

C: φ has a unique construction as a formula.

We prove property P for all formulas, by Structural Induction.

The basis (an atom) is trivial.

Starting the Inductive Step

Inductive hypothesis: Formulas β and γ each has property P.

Inductive step: Let α be $(\beta \star \gamma)$, for $\star$ a binary connective.

Clearly, α has property A; we must show it has B and C.

For property B, we consider all proper prefixes of α. They are

- The expressions "(", "$(\beta$", "$(\beta\star$", and "$(\beta \star \gamma$".
- Any expression "$(x$", where x is a proper prefix of β.
- Any expression "$(\beta \star x$", where x is a proper prefix of γ.

The formulas β and γ, by B of the inductive hypothesis, each have the same number of open and close parentheses. Also by B, an expression x in the second and third cases has more '(' than ')'. Thus, the prefix of α has more open than close parentheses in each case; α satisfies B.

Proving Property C

We have assumed that α is $(\beta \star \gamma)$, where both β and γ have property P.

For property C, we must show

If α is $(\beta' \star' \gamma')$ for **formulas** β' and γ', then $\beta = \beta'$, $\star = \star'$ and $\gamma = \gamma'$.

If $|\beta'| = |\beta|$, then $\beta' = \beta$ (both start at the second symbol of α).
Thus also $\star = \star'$ and $\gamma = \gamma'$, as required.

If $0 < |\beta'| < |\beta|$, then β' is a proper prefix of β.
Thus, by hypothesis [β has B], β' is not a formula; we have nothing to prove.

If $|\beta'| > |\beta|$,then $\beta' = \beta \star y$, where y is a proper prefix of γ (or is empty).
By hypothesis, β has equally many '(' and ')' [$P(\beta)$],
while y has more '(' than ')' [$P(\gamma)$]. Thus β' has more '(' than ')';
it is not a formula, and we have nothing to prove.

Therefore α has a unique derivation; it has property C, as required.

Two consequences of unique formation

We shall define the semantics (meaning) of a formula from its syntax.
Thus unique formation ensures unambiguous formulas.

Given a formula, determine its sub-formulas by counting parentheses.

1	2	...	m	$m+1$	$m+2$	...	$n-1$	n
(	$\underbrace{(\ \langle\texttt{rest of subformula}\rangle\ )}$			$\star$	$\langle\texttt{subformula 2}\rangle$			**)**

To determine m: count excess of '(' over ')'

When the count returns to zero, the subformula has ended.

(For efficient parsing of more-complicated formulas/programs, see CS 241.)