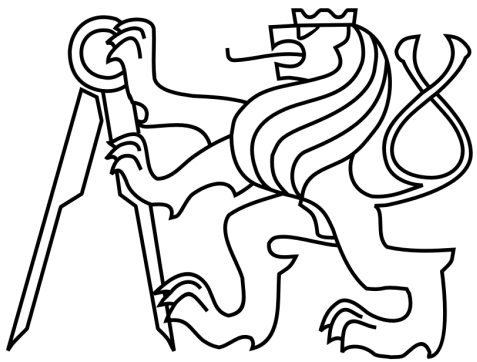
Basics of Practical Logic

Stefan Ratschan

Department of Digital Design
Faculty of Information Technology
Czech Technical University in Prague



Evropský sociální fond Praha & EU: Investujeme do vaší budoucnosti

Motivation

Fancy big question:

How can we describe and computationally analyze complex systems?

Basic language: predicate logic

Logic For Daily Survival

Personally, I use logic when I cook, wash, and clean.

But I admit that I am a little bit strange

> *"Life satisfaction positively correlates with level of education. On the contrary, dissatisfaction increases after the age of sixty "*
>
> *(Respekt 1/2017, Češi v číslech, spokojený život.)*

Correct usage of "on the contrary":

$$\neg \models A, \text{"on the contrary"}, \models \neg A$$

Whenever I read a newspaper,
I would like to send all journalists to a logic course
(that many logical mistakes they contain)

Journalists need logic, but computer scientists?

Logic for Computer Science

Almost every day you try to check
the correctness of some
computer program, web protocol, or other systems.

For doing this, you use reasoning based on common sense

The problem is that the corresponding module in the human brain developed a few thousand years ago—for some quite different purpose:

Logic For Computer Scientists

Because of this historic purpose,
this module is highly prone to mistakes
in modern, and especially computer-scientific life.

The main purpose of studying computer science is not
to learn programming languages, networking protocols etc., but to

permanently improve the logical module in your brain!

(still, those two goals can be nicely combined)

Goal of Today's Lecture

Today's lecture: Basis of practical logic
(i.e., adapted for everyday usage)

A few logical rules that
will still be valid if fall asleep for 1000 years and
wake up at a time when Java, HTML etc.
will have been dead for a long time.

How to Form Predicate Logical Expressions (Syntax)

Term (e.g., $2x + 3$, $\texttt{first}(\texttt{rest}(l))$, $a(i)$):

- Variables
- Function symbols of certain arity

Functions symbols of arity 0 are called *constants*.

We usually assign some meaning to function symbols:

- $+$ usually (but not always!) means addition
- `rest` usually means the rest of a list etc.

Moreover we often assign to function symbols certain types, for example:

- $+ : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$
- $\texttt{num\_elem} : \texttt{List} \to \mathbb{N}_0$.

There is a variant of logic ("many-sorted logic"), that describes the handling of types precisely. Here: intuitively.

How to Form Predicate Logical Expressions (Syntax)

Either true or false

Atomic formula (e.g. $2x + 3 > 0$, *empty*$(a(i))$):

- Predicate symbol of a certain arity with terms as arguments, especially the predicate $=$
- $\top$ (true, tautology), $\bot$ (false, contradiction)
  (alternative notation: **T**/**F**, 1/0, ...)

We usually assign some meaning to predicate symbols,
e.g., $<$ usually means "less than"

Moreover, we often assign some types to predicate symbols, e.g.:

- $<$ is a predicate on $\mathbb{R} \times \mathbb{R}$,
- `is_empty_list` is a predicate on lists

How to Form Predicate Logical Expressions (Syntax)

Formula: (e.g.: $\exists x \,.\, 2x + 3 > 0 \wedge x^2 \leq 0$)

- Atomic formulas
- Quantifiers: $\forall$, $\exists$
- Logical connectives: $\vee$, $\wedge$, $\neg$

Attention:

- Usually, $\neg$ has stronger precedence than $\wedge$ a $\vee$
- $\exists x \,.\, P(x) \wedge Q(x)$ here $\exists x \,.\, [P(x) \wedge Q(x)]$,
  but for others $[\exists x \,.\, P(x)] \wedge Q(x)$!

Many alternative notations for quantifiers, e.g.:

- $(\forall x) P(x) \wedge Q(x)$
- $P(x) \wedge Q(x) \quad \forall x$
- For all x, $P(x) \wedge Q(x)$.

Abbreviations, Syntactic Sugar etc.

- , : synonym for $\wedge$
- A block of quantifiers $\exists x, y, z$ means $\exists x \exists y \exists z$
- $\bigwedge_{i \in \{1,\ldots,n\}} \phi_i$ means $\phi_1 \wedge \cdots \wedge \phi_n$

Operator precedence
(e.g. $xy + z$ usually stands for $(xy) + z$ instead of $x(y + z)$).

Mathematical texts usually contain many further forms of syntactic sugar, variants, dialects etc., (e.g., $7!$, $f' = xyz$)

How to Form Logical Expressions (Syntax)

When writing formulas, we have to take care that

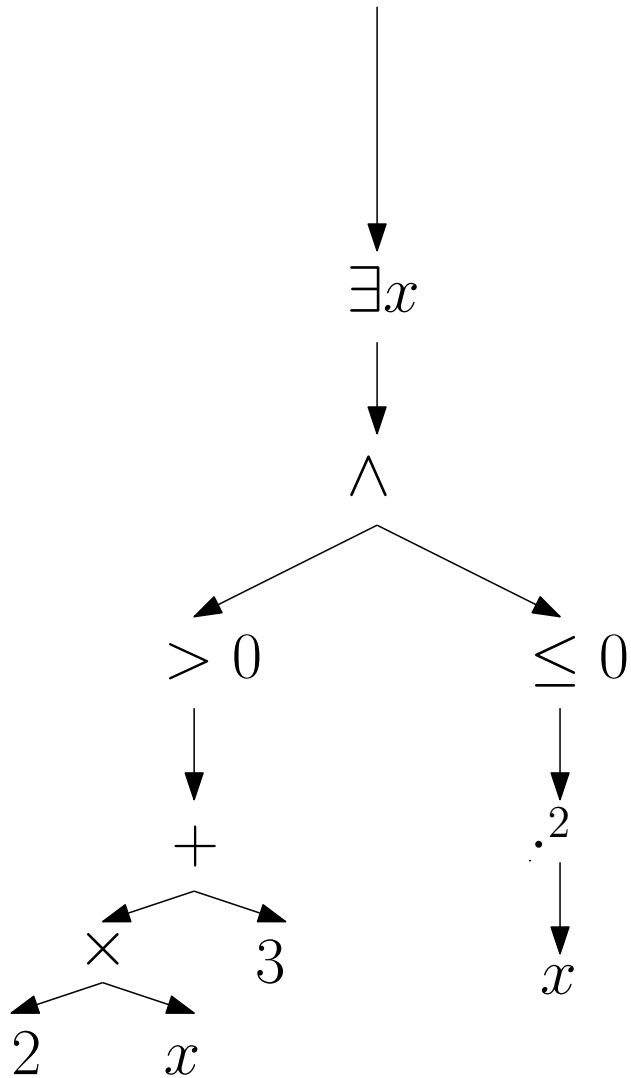
- all function and predicate symbols always
  have the correct number of arguments, and that
- the types of those arguments match.

In the case of programming languages this usually is done for use by the compiler (with the exception of JavaScript, LISP, Prolog, ...)

In other cases we have to be careful
(from your average newspaper: "Obama is the better president")

Syntax Tree

$\exists x \,.\, 2x + 3 > 0 \wedge x^2 \leq 0$



one out of many possibilities

Free/bound variable

The occurrence of variable x in a formula is called $bound$ *if on the path from the corresponding leaf to the root of the syntax tree of this formula there is a node marked with the quantifier* $\forall x$ *or* $\exists x$*. In the opposite case we call this occurrence* $free$*. A quantifier* $\forall x$ *or* $\exists x$ $binds$ *all occurrences of the variable x, that occur in the syntactic tree below this quantifier* [1]*.*

[1] Transl. from J. Velebil "Velmi jemný úvod do matematické logiky" (ČVUT FEL)

Substitution

We will write $A[x \leftarrow t]$ for

the result of replacing every free occurrence of the variable x with the term t in expression A

Attention:

- Name clashes: $[\forall x \,.\, P(x, y)][y \leftarrow f(x)] \;\equiv\; \forall x \,.\, P(x, f(x))$
- Substitution binds stronger than logical symbols (our convention)

$$A(x) \wedge B(x)[x \leftarrow f(x)] \;\text{ je }\; A(x) \wedge B(f(x))$$

- Substitution $\neq$ renaming!
  - Renaming x to y in $\forall x \,.\, f(x)$ results in $\forall y \,.\, f(y)$
  - But $[\forall x \,.\, f(x)][x \leftarrow y]$ is $\forall x \,.\, f(x)$

One More Abbreviation

$$\exists x \,.\, \phi \wedge \neg \exists y \,.\, \neg [x = y] \wedge \phi[x \leftarrow y]$$

where ϕ is an arbitrary formula

"there is precisely one"

Notation: $\exists^1 x \,.\, \phi$, $\exists!\, x \,.\, \phi$

Meaning of Logical Formulas (Semantics)

$1 + 1$???

An *interpretation* gives a certain meaning to every predicate and function symbol (arity and types must match!).

A *(variable) assignment* gives a certain value to every variable.

For a certain interpretation $\mathcal{I}$, variable assignment σ, term t
we can define the corresponding value of the term.

Example: the term $1 + 1$ has

- the value 2 under the usual interpretation in $\mathbb{Z}$, but
- the value 0 under the interpretation in $\mathbb{Z}_2$.

For a certain interpretation $\mathcal{I}$, variable assignment σ, formula ϕ,
the corresponding meaning is denoted by:

$$\mathcal{I}, \sigma \models \phi$$

Example: : $\mathbb{R} \models \exists x \,.\, x^2 = -1$, but not $\mathbb{C} \models \exists x \,.\, x^2 = -1$

This can be precisely defined, but we do not have time for that

Validity, Satisfiability

Assume: We do not know interpretation/variable assignment

What can we say about $P(x) \vee \neg P(x)$ or about $\top$?

$\models \phi$: The formula ϕ is *valid*
(holds independently of certain interpretation/variable assignment).

What can we say about $P(x) \wedge \neg P(x)$ or $\bot$?

A formula ϕ is *satisfiable* iff
there is an interpretation $\mathcal{I}$ and variable assignment σ s.t. $\mathcal{I}, \sigma \models \phi$

Logic in Practice

In practice it usually sufficient to know semantics intuitively

But it is important to know rules for handling logical formulas

Rest of today's lecture: logical rules, that

- hold in general (i.e., even in 1000 years), and that
- you can use every day.

Equivalences

"*is equivalent*": formulas have the same meaning, one formula can be replaced by the other (A, B, C, P are placeholders for arbitrary formulas):

- $A \wedge \top$ is equivalent to A
- $A \wedge \bot$ is equivalent to $\bot$
- $A \vee \top$ is equivalent to $\top$
- $A \vee \bot$ is equivalent to A
- $A \wedge B$ is equivalent to $B \wedge A$
- $A \vee B$ is equivalent to $B \vee A$
- $A \wedge [B \vee C]$ is equivalent to $[A \wedge B] \vee [A \wedge C]$
- $A \vee [B \wedge C]$ is equivalent to $[A \vee B] \wedge [A \vee C]$

Equivalences with Negation

- $\neg\top$ is equivalent to $\bot$
- $\top$ is equivalent to $\neg\bot$
- $\neg\neg A$ is equivalent to A
- $A \wedge \neg A$ is equivalent to $\bot$
- $A \vee \neg A$ is equivalent to $\top$
- $\neg[A \wedge B]$ is equivalent to $\neg A \vee \neg B$
- $\neg[A \vee B]$ is equivalent to $\neg A \wedge \neg B$

Further Equivalences

- $\exists x \exists y \,.\, A$ is equivalent to $\exists y \exists x \,.\, A$
- $\forall x \forall y \,.\, A$ is equivalent to $\forall y \forall x \,.\, A$
- $\neg \forall x \,.\, A$ is equivalent to $\exists x \,.\, \neg A$
- $\neg \exists x \,.\, A$ is equivalent to $\forall x \,.\, \neg A$
- $\exists x \,.\, A \vee B$ is equivalent to $[\exists x \,.\, A] \vee [\exists x \,.\, B]$
- $\forall x \,.\, A \wedge B$ is equivalent to $[\forall x \,.\, A] \wedge [\forall x \,.\, B]$
- $\exists x \,.\, A \wedge B$ is equivalent to $[\exists x \,.\, A] \wedge B$, if B does not contain x
- $\forall x \,.\, A \vee B$ is equivalent to $[\forall x \,.\, A] \vee B$, if B does not contain x
- $\exists x \,.\, A$ is equivalent to A if x does not occur freely in A
- $\forall x \,.\, A$ is equivalent to A if x does not occur freely in A
- A is equivalent to B, if A a B are the same up to renaming of bound variables without name clashes (α-conversion)

Equivalence Rules

- A is equivalent to A (*reflexivity*)
- If A is equivalent to B, then B is equivalent to A (*symmetry*)
- If A is equivalent to B and B is equivalent to C,
  then A is equivalent to C (*transitivity*)

Implication and Equivalence Signs

$A \Rightarrow B$ can be viewed as a short-cut for $\neg A \vee B$

$A \Leftrightarrow B$ can be viewed as a short-cut for $[A \Rightarrow B] \wedge [B \Rightarrow A]$

Attention: Various different conventions regarding operator precedence!

Consequences:

- $A \Rightarrow \top$ is equivalent to $\top$
- $\bot \Rightarrow B$ is equivalent to $\top$
- $[A \wedge B] \Rightarrow A$ is equivalent to $\top$
- $A \Rightarrow [A \vee B]$ is equivalent to $\top$
- $A \Rightarrow B$ is equivalent to $\neg B \Rightarrow \neg A$
- $A \Leftrightarrow B$ is equivalent to $\neg A \Leftrightarrow \neg B$

Further (*case distinction*):

- A is equivalent to $[B \Rightarrow A] \wedge [\neg B \Rightarrow A]$
- A is equivalent to $[B_1 \Rightarrow A] \wedge \cdots \wedge [B_n \Rightarrow A]$,
  if $B_1 \vee \cdots \vee B_n$ is equivalent to $\top$

... and now we are going to fulfill an old dream ...

Proofs

Proof: certificate that a certain formula ϕ is valid ($\models \phi$)

In propositional logic: truth tables

Problem? They grow exponentially in size, always!

Proofs for Humans

We can prove $A \Leftrightarrow B$ by
proving that A is equivalent to B.

For example: We want to prove $[P \vee Q] \Leftrightarrow P \vee [Q \wedge \neg P]$:

Due to transitivity of equivalences
it suffices to find a sequence of equivalent formulas:

$$P \vee Q$$
$$[P \vee Q] \wedge \top$$
$$[P \vee Q] \wedge [P \vee \neg P]$$
$$P \vee [Q \wedge \neg P]$$

We can prove A by proving that A is equivalent to $\top$.

But, in general, this is not enough.

See course on formal methods: general proof method

Extension of Proof System

You can add your own "private" rules:

As soon as you have a proof of a certain equivalence,
you can use it as an additional rule

Important:

- Other people do not know your rules
- So: If faced with the question "Why does this hold?"
  you always should be able to show the original proof

Equality

Usually we have available the special predicate symbol $=$.

We know (x, y, z can stand for any term):

- $x = x$ (*reflexivity*)
- If $x = y$, then $y = x$ (*symmetry*)
- If $x = y$ and $y = z$, then $x = z$ (*transitivity*)

If we know that a certain equality holds,
we can always replace the left-hand side by the right-hand side
and vice versa.

Notation: $x \neq y$ means $\neg[x = y]$

Set Theory

Axiomatization: Zermelo-Fraenkel set theory (ZFC) (C: axiom of choice)

Too complicated for this lectures

Some rules for everyday usage

We form sets as follows: $\{x \mid A\}$, where A is a logical formula

For sets S and T,

- $S \cap T \doteq \{x \mid x \in S \wedge x \in T\}$
- $S \cup T \doteq \{x \mid x \in S \vee x \in T\}$
- $S \setminus T \doteq \{x \mid x \in S \wedge \neg[x \in T]\}$
- The empty set is a set $\emptyset$ s.t. $\neg\exists x \,.\, x \in \emptyset$

Equivalences: For sets S and T,

- $S \subseteq T$ is equivalent to $\forall x \,.\, x \in S \Rightarrow x \in T$
- $S = T$ is equivalent to $\forall x \,.\, x \in S \Leftrightarrow x \in T$
- $a \in \{x \mid A\}$ is equivalent to $A[x \leftarrow a]$
- $\exists x \,.\, x \in \emptyset$ is equivalent to $\bot$

Sets

Cartesian product: $S \times T \doteq \{(x, y) \mid x \in S, y \in T\}$

Power set: $\mathcal{P}(T) \doteq \{S \mid S \subseteq T\}$, sometimes also 2^T

Moreover:

- For a set S, $\{x \mid x \in S\} = S$

Very often the following notation is used:

- $\{a\}$ for $\{x \mid x = a\}$
- $\{a_1, \ldots, a_n\}$ for $\{x \mid x = a_1 \vee \cdots \vee x = a_n\}$
- $\forall x \in S \,.\, A$ for $\forall x \,.\, x \in S \Rightarrow A$
- $\exists x \in S \,.\, A$ for $\exists x \,.\, x \in S \wedge A$
- $x \notin A$ for $\neg[x \in A]$
- $\bigcap$, $\bigcup$, $\ldots$

Proofs in Set Theory

$$A \setminus (B \cap A) \subseteq A$$
$$\forall x \,.\, [x \in A \setminus (B \cap A)] \Rightarrow x \in A$$
$$\forall x \,.\, [x \in \{x \mid x \in A \wedge \neg x \in (B \cap A)\}] \Rightarrow x \in A$$
$$\forall x \,.\, [x \in A \wedge \neg x \in (B \cap A)] \Rightarrow x \in A$$
$$\forall x \,.\, \top$$
$$\top$$

Discussion

First-order predicate logic + set theory suffices for
building all of mathematics!

A *relation* on sets $S_1, \ldots, S_n$ is
a subset of the Cartesian product $S_1 \times \cdots \times S_n$.

Example: The relation on the sets $\mathbb{N}$ and $\mathbb{N}$,
that formalizes the property "is a prime factor of ":

$$\{(2,2),(3,3),(2,4),(5,5),(2,6),(3,6),\ldots\}$$

A *function* from a set S to a set T is a relation F on S a T such that
for every $s \in S$ there is exactly one $t \in T$ with $F(s,t)$.

Example: The function length (of a string)

$$\{("ab",2),("cdx",3),("",0),\ldots\}$$

Usage of Predicate Logic

Predicate logic is usually hidden in prose:

Example: Theorem 2.3 from Hopcroft and Ullman [1979]:

> *"Let r be a regular expression. Then there exists an NFA with ϵ-transitions that accepts $L(r)$"*

$$\forall r \,.\, \mathit{regexp}(r) \Rightarrow \exists a \,.\, \mathit{NFAeps}(a) \wedge \mathrm{accepts}(a, L(r))$$

The predicates *regexp*, *NFAeps*, *accepts* and the function symbol L are defined before.

Conclusions: There exists a NFA with ϵ-transitions that accepts $L(01*)$.

In general:

If we know that a formula with a leading universal quantifier holds, we can substitute concrete values for the corresponding variables.

Predicate Logic in Everyday Life

Interview with Petr Fiala, Respekt 2017/50:

> *When I hear some politicians say: "If we will not be in the Eurozone, then we will not belong to the core of the European union", I have to smile. Since when is Greece in the core of the European union?*

Statement:

$$\forall x \,.\, \neg E(x) \Rightarrow \neg C(x)$$

Attempt to refute the statement:

$$E(GR) \wedge \neg C(GR)$$

Does this really refute the statement?

Analysis

To refute $\forall x \,.\, \neg E(x) \Rightarrow \neg C(x)$:

$$\neg \forall x \,.\, \neg E(x) \Rightarrow \neg C(x)$$

$$\exists x \,.\, \neg [\neg E(x) \Rightarrow \neg C(x)]$$

$$\exists x \,.\, \neg [E(x) \vee \neg C(x)]$$

$$\exists x \,.\, \neg E(x) \wedge C(x)$$

But:

$$E(GR) \wedge \neg C(GR)$$

Further Material

Books (thousands of them)

Online material:

- Courses BI-MLO, BI-ZDM
- Starý, J., Introduction to Mathematical Logic
- `http://people.ucalgary.ca/~rzach/static/open-logic/courses/phil379/phil379-screen.pdf`
- `https://github.com/OpenLogicProject/OpenLogic/wiki/Other-Logic-Textbooks`

Next Lecture

How can we describe and computationally analyze complex systems?

The structure of complex systems, finite automata

Literature

J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, 1979.