Automated Deduction

Laura Kovács
TU Wien

Outline

General

- All information online at:
  www.cse.chalmers.se/~laurako/links/ADuct.html.
- Lectures:
  - During March 7- May 14, 2019;
  - Tuesdays between 10:15-11:45, in EI 8 Pötzl HS;
  - Thursdays between 9:15-10:45, in EI 3 Sahulka HS;
- Homeworks/exercises:
  - all together 3 homeworks;
  - handed out online March 21, April 4 and May 7 (tentative dates);
  - solution deadlines: upcoming lecture of the next week.
- Assessment: written exam (60%), homeworks (40%).
- Exam date: in May/June (to be decided).

Computer Systems and Correctness

Suppose we design a (complex) system, which may contain various components, for example, sensors, networks, computers. All of these components are using software.

We have requirements on how the system should function, for example safety, reliability, security, availability, absence of deadlocks etc.

How can one ensure that the system satisfies these requirements?

Modern computer systems are unreliable.

Small Example: Software

Consider the following fragment of a C program:

```
/* Returns a new array of integers of a given
 length initialised by a non-zero value */
int* allocateArray(int length)
{
  int i;
  int* array;
  array = malloc(sizeof(int)*length); // may return 0!
  if (!array) return 0;
  for (i = 0;i <=< length;i++)
    array[i] = 0;
  return array;
}
```

Is this program correct?

Hardly: it writes into memory that has not been allocated. No: it may write to the null address. No: it initialises the array by zeros

We discussed correctness of a program without ever defining what it means.

So what is correctness?

Notes

- We could spot the first two errors without knowing anything about the intended meaning of the program. But we had to understand the meaning of C programs in general and some specific properties of programming in C.
- To understand the last “error” we had to know something about the the intended behaviour of the program.

How to establish correctness

- Consider the program/system as a mathematical object. To do this, we will have to build a formal model of the system.
- Find a formal language for expressing intended properties.
- The language must have a semantics that explains what are possible interpretations of the sentences of the formal language. The semantics is normally based on notions is true, is false, satisfies.
- Write a specification, that is, intended properties of the system in this language.
- Prove formally that the system model also satisfies the specification.

What is logic?

Mathematical logic is a branch of science that deals with notions such as

- syntax and semantics;
- proof theory and model theory;
- reasoning.

Computational Logic

Computational logic deals with applications of logic in computer science and computer engineerings, including

- software and hardware verification;
- circuit design;
- constraint satisfaction;
- knowledge representation and reasoning;
- semantic Web;
- theorem proving in mathematics;
- ...

This course - tentative list of topics

Part 1: satisfiability checking in propositional logic: splitting, DPLL, randomized algorithms;

Part 2: satisfiability checking in the combined theory of linear arithmetic, uninterpreted functions and arrays (SMT);

Part 3: validity proving in first-order logic: resolution, superposition, unification, redundancy elimination;

- tools: MiniSat (Part 1) , Z3 (Part 2), and Vampire (Part 3).

Outline

Propositional logic

Formalises the notion of a proposition, that is a statement that can be either true or false.

Propositional logic: syntax

Assume a countable set of boolean variables.
Propositional formula:

- Every boolean variable is a formula, also called atomic formula, or simply atom.
- $\top$ and $\bot$ are formulas.
- If $A_1, \ldots, A_n$ are formulas, where $n \geq 2$, then $(A_1 \wedge \ldots \wedge A_n)$ and $(A_1 \vee \ldots \vee A_n)$ are formulas.
- If A is a formula, then $(\neg A)$ is a formula.
- If A and B are formulas, then $(A \rightarrow B)$ and $(A \leftrightarrow B)$ are formulas.

The symbols $\top, \bot, \wedge, \vee, \neg, \rightarrow, \leftrightarrow$ are called connectives.

Parsing Formulas

We want to avoid expressions cluttered with parentheses. The standard way to avoid them is to assign precedence to operators and use the precedence to disambiguate expressions.
For example, in arithmetic we know that the expression

$$x \cdot y + 2 \cdot z$$

is equivalent to

$$(x \cdot y) + (2 \cdot z),$$

since $\cdot$ has a higher precedence than $+$.

Connectives and Their Precedence

Connective	Name	Precedence
$\top$	verum	
$\bot$	falsum	
$\neg$	negation	5
$\wedge$	conjunction	4
$\vee$	disjunction	3
$\rightarrow$	implication	2
$\leftrightarrow$	equivalence	1

Parsing Formulas

Let us parse

$$\neg A \wedge B \to C \vee D \leftrightarrow E.$$

Connective	Precedence
$\top$	
$\bot$	
$\neg$	5
$\wedge$	4
$\vee$	3
$\to$	2
$\leftrightarrow$	1

Inside-out (starting with the highest precedence connectives):

$$(((\neg A) \wedge B) \to (C \vee D)) \leftrightarrow E.$$

Outside-in (starting with the lowest precedence connectives):

$$(((\neg A) \wedge B) \to (C \vee D)) \leftrightarrow E.$$

Semantics, Interpretation

Consider an arithmetical expression, for example

$$x \cdot y + 2 \cdot z.$$

In arithmetic the meaning of expressions with variables is defined as follows.
Take a mapping from variables to (integer) values, for example

$$\{x \mapsto 1, y \mapsto 7, z \mapsto -3\}.$$

Then, under this mapping the expression has the value 1. In other words, when we interpret variables as values, we can compute the value of the expression.

Semantics, Interpretation

Likewise, the semantics of propositional formulas can be defined by assigning values to variables.

- There are two boolean values, also called truth values: true (denoted 1) and false (denoted 0).
- An interpretation for a set P of boolean variables is a mapping $I : P \to \{1, 0\}$.
- Interpretations are also called truth assignments.

Interpreting formulas

The truth value of a complex formula is determined by the truth values of its components.

Given an interpretation I, extend I to a mapping from all formulas to truth values as follows.

1. $I(\top) = 1$ and $I(\bot) = 0$.
2. $I(A_1 \wedge \ldots \wedge A_n) = 1$ if and only if $I(A_i) = 1$ for all i.
3. $I(A_1 \vee \ldots \vee A_n) = 1$ if and only if $I(A_i) = 1$ for some i.
4. $I(\neg A) = 1$ if and only if $I(A) = 0$.
5. $I(A_1 \rightarrow A_2) = 1$ if and only if $I(A_1) = 0$ or $I(A_2) = 1$.
6. $I(A_1 \leftrightarrow A_2) = 1$ if and only if $I(A_1) = I(A_2)$.

Operation tables

$I(A_1 \vee A_2) = 1$ if and only if $I(A_1) = 1$ or $I(A_2) = 1$.
$I(A_1 \leftrightarrow A_2) = 1$ if and only if $I(A_1) = I(A_2)$.

$\wedge$	1	0
1	1	0
0	0	0

$\vee$	1	0
1	1	1
0	1	0

$\neg$	
1	0
0	1

$\rightarrow$	1	0
1	1	0
0	1	1

$\leftrightarrow$	1	0
1	1	0
0	0	1

Therefore, every connective can be considered as a function on truth values.

Satisfiability, validity, equivalence

Let A be a formula.

- If $I(A) = 1$, then we say that the formula A is true in I and that I satisfies A and that I is a model of A, denoted by $I \models A$.
- If $I(A) = 0$, then we say that the formula A is false in I.
- A is satisfiable if it is true in some interpretation.
- A is valid (or a tautology) if it is true in every interpretation.
- Two formulas A and B are called equivalent, denoted by $A \equiv B$ if they have the same models.

Examples

$A \to A$ and $A \vee \neg A$ are valid for all formulas A.

Evidently, every valid formula is also satisfiable.

$A \wedge \neg A$ is unsatisfiable.

Formula p, where p is a boolean variable, is satisfiable but not valid.

Examples: equivalences

For all formulas A and B, the following equivalences hold.

$$A \to \bot \equiv \neg A; \tag{1}$$
$$\top \to A \equiv A; \tag{2}$$
$$A \to B \equiv \neg(A \wedge \neg B); \tag{3}$$
$$A \wedge B \equiv \neg(\neg A \vee \neg B); \tag{4}$$
$$A \vee B \equiv \neg A \to B. \tag{5}$$

Connections between these notions

1. A formula A is valid if and only if $\neg A$ is unsatisfiable.
2. A formula A is satisfiable if and only if $\neg A$ is not valid.

3. A formula A is valid if and only if A is equivalent to $\top$.
4. Formulas A and B are equivalent if and only if the formula $A \leftrightarrow B$ is valid.

5. Formulas A and B are equivalent if and only if the formula $\neg(A \leftrightarrow B)$ is unsatisfiable.
6. A formula A is satisfiable if and only if A is not equivalent to $\bot$.

How to evaluate a formula?

Let's evaluate the formula

$$(p \to q) \wedge (p \wedge q \to r) \to (p \to r)$$

in the interpretation

$$\{p \mapsto 1, q \mapsto 0, r \mapsto 1\}.$$

Evaluating a formula

formula	value
$(p \to q) \wedge (p \wedge q \to r) \to (p \to r)$	1
$p \to r$	1
$(p \to q) \wedge (p \wedge q \to r)$	0
$p \wedge q \to r$	1
$p \to q$	0
$p \wedge q$	0
p p p	1
q q	0
r r	1

$$\{p \mapsto 1, q \mapsto 0, r \mapsto 1\}$$

So the formula is true in this interpretation.

The truth value of the formula was derived from the truth values in the table of subformulas.

Equivalent replacement

We denote by $A[B]$ a formula A with a fixed occurrence of a subformula B. If we use this notation we can also write $A[B']$ to denote the formula obtained from A by replacing this occurrence of B by B'.

Lemma (Equivalent Replacement)

Let I be an interpretation and $I \models A_1 \leftrightarrow A_2$. Then $I \models B[A_1] \leftrightarrow B[A_2]$.

In other words, replacing, in a formula B, a subformula A_1 by a formula A_2 with the same value gives a formula with the same value.

Theorem (Equivalent Replacement)

Let $A_1 \equiv A_2$. Then $B[A_1] \equiv B[A_2]$.

In other words, replacing, in a formula B, a subformula A_1 by an equivalent formula A_2 gives an equivalent formula.

(thanks to compositionality!)

A purely syntactic algorithm

If $I \models p$, then $I \models p \leftrightarrow \top$;
If $I \not\models p$, then $I \models p \leftrightarrow \bot$;

Since we can replace a subformula by a formula with the same value, we can replace every atom p by either $\top$ or $\bot$, depending on the value of p in I.

Rewrite rules for evaluating a formula

Suppose that we have a formula consisting only of $\bot$ and $\top$.
One can note that every formula of this form different from $\bot$ and $\top$ can be "simplified" to a smaller equivalent formula.
For example, every formula of the form $A \to \top$ is equivalent to a simpler formula $\top$.
This simplification process can be formalised as a rewrite rule system:

$$\top \wedge \ldots \wedge \top \Rightarrow \top$$
$$\bot \wedge A_1 \wedge \ldots \wedge A_n \Rightarrow \bot$$

$$\top \vee A_1 \vee \ldots \vee A_n \Rightarrow \top$$
$$\bot \vee \ldots \vee \bot \Rightarrow \bot$$

$$\neg\top \Rightarrow \bot$$
$$\neg\bot \Rightarrow \top$$

$$A \to \top \Rightarrow \top$$
$$\bot \to A \Rightarrow \top$$
$$\top \to \bot \Rightarrow \bot$$

$$\top \leftrightarrow \top \Rightarrow \top$$
$$\top \leftrightarrow \bot \Rightarrow \bot$$
$$\bot \leftrightarrow \top \Rightarrow \bot$$
$$\bot \leftrightarrow \bot \Rightarrow \top$$

Algorithm for evaluating a formula

We can define a purely syntax algorithm for evaluating a formula using the rewrite rule system.

procedure *evaluate*(G, I)
input: formula G, interpretation I
output: the boolean value $I(G)$
begin
  forall atoms p occurring in G
    if $I \models p$
      then replace all occurrences of p in G by $\top$;
      else replace all occurrences of p in G by $\bot$;
  rewrite G into a normal form using the rewrite rules
  if $G = \top$ **then return** 1 **else return** 0
end

Example

Let us evaluate the formula

$$(p \to q) \wedge (p \wedge q \to r) \to (p \to r)$$

in the interpretation

$$\{p \mapsto 1, q \mapsto 0, r \mapsto 1\}.$$

Its value is equal to the value of

$$(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to (\top \to \top).$$

Apply rewrite rules

Inside-out, left-to-right:

$$(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow$$
$$\bot \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow$$
$$\bot \wedge (\bot \to \top) \to (\top \to \top) \Rightarrow$$
$$\bot \wedge \top \to (\top \to \top) \Rightarrow$$
$$\bot \to (\top \to \top) \Rightarrow$$
$$\bot \to \top \Rightarrow$$
$$\top$$

$$A \wedge \bot \Rightarrow \bot$$
$$\top \to \bot \Rightarrow \bot$$
$$A \to \top \Rightarrow \top$$

Outside-in, right-to-left:

$$(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow$$
$$(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to \top \Rightarrow$$
$$\top$$

The result will always be the same independently of the order of rewriting!

Outline

A puzzle

There are three persons: Stirlitz, Müller, and Eismann. It is known that exactly one of them is Russian, while the other two are Germans. Moreover, every Russian must be a spy.

When Stirlitz meets Müller in a corridor, he makes the following joke: “you know, Müller, you are as German as I am Russian”. It is known that Stirlitz always tells the truth when he is joking.

We have to show that Eismann is not a Russian spy.

How can we solve problems of this kind?

Propositional Satisfiability Problem

Given a propositional formula A, check whether it is satisfiable or unsatisfiable.

If A is satisfiable, we also want to find a satisfying assignment for A, that is, a model of A.

It is one of the most famous combinatorial problems in computer science.

It is a very hard problem with a surprisingly large number of practical applications.

It is also the first ever problem to be proved NP-complete.

Russian spy puzzle

There are three persons: Stirlitz, Müller, and Eismann. It is known that exactly one of them is Russian, while the other two are Germans. Moreover, every Russian must be a spy.

When Stirlitz meets Müller in a corridor, he makes the following joke: “you know, Müller, you are as German as I am Russian”. It is known that Stirlitz always tells the truth when he is joking.

We have to show that Eismann is not a Russian spy.

How can we solve problems of this kind?

Formalisation in propositional logic

Introduce nine propositional variables as in the following table:

	Stirlitz	Müller	Eismann
Russian	RS	RM	RE
German	GS	GM	GE
Spy	SS	SM	SE

For example,

SE : Eismann is a Spy
RS : Stirlitz is Russian

Formalisation in propositional logic

There are three persons: Stirlitz, Müller, and Eismann. It is known that exactly one of them is Russian, while the other two are Germans.

$$(RS \wedge GM \wedge GE) \vee (GS \wedge RM \wedge GE) \vee (GS \wedge GM \wedge RE).$$

Moreover, every Russian must be a spy.

$$(RS \rightarrow SS) \wedge (RM \rightarrow SM) \wedge (RE \rightarrow SE).$$

When Stirlitz meets Müller in a corridor, he makes the following joke: "you know, Müller, you are as German as I am Russian".

$$RS \leftrightarrow GM.$$

Hidden: Russians are not Germans.

$$(RS \leftrightarrow \neg GS) \wedge (RM \leftrightarrow \neg GM) \wedge (RE \leftrightarrow \neg GE).$$

We have to show that Eismann is not a Russian spy.
To this end, we add the following formula

$$RE \wedge SE.$$

and check whether the resulting set of formulas is satisfiable. If it is unsatisfiable, then Eismann cannot be a Russian spy.

Idea: use formula evaluation methods

Consider $\neg((p \to q) \wedge (p \wedge q \to r) \to (p \to r))$.
We can evaluate it in any interpretation, for example,
$\{p \mapsto 0, q \mapsto 0, r \mapsto 0\}$:

	subformula							I_0
1	$\neg((p \to q) \wedge (p \wedge q \to r) \to (p \to r))$							0
2	$(p \to q) \wedge (p \wedge q \to r) \to (p \to r)$					$p \to r$		1
3						$p \to r$		1
4	$(p \to q) \wedge (p \wedge q \to r)$							1
5			$p \wedge q \to r$					1
6	$p \to q$							1
7			$p \wedge q$					0
8	p		p			p		0
9		q		q				0
10					r		r	0

Truth tables

$\neg((p \to q) \wedge (p \wedge q \to r) \to (p \to r)$.
Likewise, we can evaluate it in all interpretations:

	subformula	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8
1	$\neg((p \to q) \wedge (p \wedge q \to r) \to (p \to r))$	0	0	0	0	0	0	0	0
2	$(p \to q) \wedge (p \wedge q \to r) \to (p \to r)$	1	1	1	1	1	1	1	1
3	$p \to r$	1	1	1	1	0	1	0	1
4	$(p \to q) \wedge (p \wedge q \to r)$	1	1	1	1	0	0	0	1
5	$p \wedge q \to r$	1	1	1	1	1	1	0	1
6	$p \to q$	1	1	1	1	0	0	1	1
7	$p \wedge q$	0	0	0	0	0	0	1	1
8	p	0	0	0	0	1	1	1	1
9	q	0	0	1	1	0	0	1	1
10	r	0	1	0	1	0	1	0	1

The formula is unsatisfiable since it is false in every interpretation.

Problem: a formula with n propositional variables has 2^n different interpretations.

Compact truth table

Idea: we can sometimes evaluate a formula based on values of only a subset of all variables.

subformula	I_2	I_3	I_4	I_1
$\neg((p \to q) \wedge (p \wedge q \to r) \to (p \to r))$	0	0	0	0
$(p \to q) \wedge (p \wedge q \to r) \to (p \to r)$	1	1	1	1
$p \to r$	1	0	0	1
$(p \to q) \wedge (p \wedge q \to r)$		0	0	
$p \wedge q \to r$		1	0	1
$p \to q$		0	1	
$p \wedge q$		0	1	
p	0	1	1	
q		0	1	
r	0	0	0	1

The formula is unsatisfiable.

Note: the size of the compact table (but not the result) depends on the order of atoms!

The ideas of guessing variable values (or case analysis) and propagation are the key ideas in nearly all propositional satisfiability algorithms.