Automated Reasoning

Georg Moser

Institute of Computer Science @ UIBK

Winter 2013

Organisation

Time and Place

Computational Logic	Wednesday, 13:15–15:00	3W03
Automated Theorem Proving	Wednesday, 15:15–17:00	3W03
exercise class	Wednesday, 17:15–18:00	3W03

Schedule

week 1	October 2	week 8	November 27
week 2	October 9	week 9	December 4
week 3	October 16	week 10	December 11
week 4	October 23	week 11	January 8
week 5	no lecture	week 12	January 15
week 6	November 13	week 13	January 22
week 7	November 20	first exam	January 29

Office Hours

Thursday, 9:00–11:00, 3M09, IfI Building

Outline of the Module

Advanced Topics in Logic

for example

- compactness
- model existence theorem
- Herbrand's Theorem
- Curry-Howard Isomorphism

Automated Reasoning

for example

- tableau provers
- redundancy and deletion
- superposition
- Robbins problem

Outline of the Lecture "Computational Logic"

Propositional Logic

short reminder of propositional logic, soundness and completeness theorem, natural deduction, propositional resolution

First Order Logic

introduction, syntax, semantics, Löwenheim-Skolem, compactness, model existence theorem, natural deduction, completeness, normalisation

Properties of First Order Logic

Craig's Interpolation Theorem, Robinson's Joint Consistency Theorem, Herbrand's Theorem

Limits and Extensions of First Order Logic

Intuitionistic Logic, Curry-Howard Isomorphism, Limits, Second-Order Logic

Outline of the Lecture "Automated Theorem Proving"

Early Approaches in Automated Reasoning

short recollection of Herbrand's theorem, Gilmore's prover, method of Davis and Putnam

Starting Points

resolution, tableau provers, structural Skolemisation, redundancy and deletion

Automated Reasoning with Equality

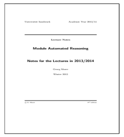
ordered resolution, paramodulation, ordered completion and proof orders, superposition

Applications of Automated Reasoning

Neuman-Stubblebinde Key Exchange Protocol, Robbins problem, resolution and paramodulation as decision procedure, . . .

Literature

- lecture notes
  (2nd edition)



Additional Reading

- G.S. Boolos, J.P. Burgess, and R.C. Jeffrey
  Computability and Logic
  Cambridge University Press, 2007
- H.-D. Ebbinghaus, J. Flum, and W. Thomas
  Einführung in die mathematische Logik
  Spektrum Akademischer Verlag, 2007
- A. Leitsch
  The Resolution Calculus
  Springer-Verlag, 2007
- papers, distributed during the course

Time and Place (cont'd)

Computational Logic	Wednesday, 13:15–15:00	3W03
Automated Theorem Proving	Wednesday, 15:15–17:00	3W03
exercise class	Wednesday, 17:15–18:00	3W03

Automated Reasoning Block

block	Wednesday, 13:15–18:00	3W03

Comments

- officially there are two lectures and one exercise group
- this is nonsense, as the course on theorem proving is based on the course on logic
- suggestion: we start with logic, if we are finished, we continue with theorem proving
- typical scheduling of the block: lecture, exercises, lecture

Introduction

What is Logic?

Argument ①

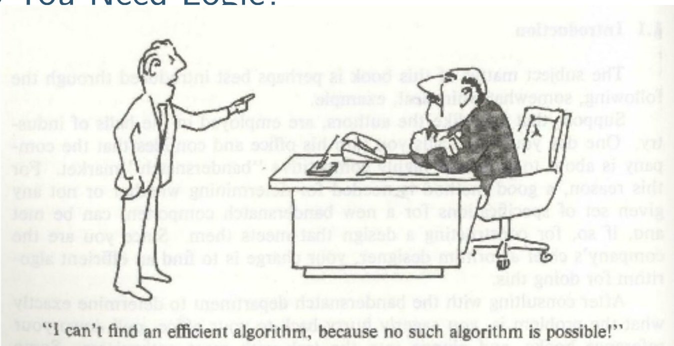
1. a mother or father of a person is an ancestor of that person
2. an ancestor of an ancestor of a person is an ancestor of a person
3. Sarah is the mother of Isaac, Isaac is the father of Jacob
4. Thus, Sarah is an ancestor of Jacob

Argument ②

1. a square or cube of a number is a power of that number
2. a power of a power of a number is a power of that number
3. 64 is the cube of 4, four is the square of 2
4. Thus, 64 is a power of 2

logic tells us that argument ① = argument ②

Why Do You Need Logic?



"I can't find an efficient algorithm, because no such algorithm is possible!"

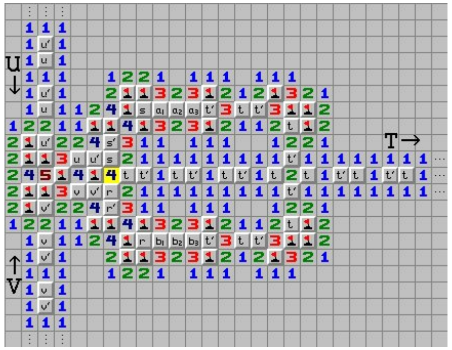
Recall

(for example) the equivalence of programs is undecidable

Proof.

reduction from the undecidability of the *Entscheidungsproblem* (Alonzo Church) ∎

Another Picture



SAT technology

- a Minesweeper solver can be coded by using a SAT solver
- verifying the correctness of a given Minesweeper configuration is an NP-complete problem

A More Serious Answer

Application ①: Program Analysis

- abstract interpretations represent the behaviour of programs
- logical products of interpretations allows the automated combination of simple interpreters
- based on Nelson-Oppen methodology

Application ②: Databases

- datalog is a declarative language and syntactically it is a subset of Prolog; used in knowledge representation systems
- disjunctive datalog is an extension of datalog that allows disjunctions in heads of rules
- disjunctive datalog is a strict extension of SQL
- (disjunctive) datalog forms the basis of semantic web applications and has connections to description logics and ontologies

(Disjunctive) Datalog

- datalog is a subset of Horn logic; hence minimal model of any datalog program is unique
- datalog rules can be translated into inclusions in relational databases
- datalog extends positive relational algebras
- disjunctive datalog extends datalog, but remains decidable
- disjunctive datalog can be extended with negation

Complexity results

- expression complexity of datalog is EXPTIME-complete
- expression complexity for disjunctive datalog (with $\neg$) is $\mathrm{NEXPTIME}^{\mathrm{NP}}$-complete
- problems complete for $\mathrm{NEXPTIME}^{\mathrm{NP}}$ can (only) be solved on an NTM with NP-oracle running in exponential time

Application ③: Types as Formulas

- the type checking in simple λ-calculus is equivalent to derivability in intuitionistic logic
- intuitionistic logic is a (sort of) constructive restriction of classical logic
- this correspondence is called Curry-Howard isomorphism
- the Curry-Howard correspondence can be extended to richer programming languages (and logics)

Application ④: Complexity Theory

- NP is the class of problems decidable by a NTM that runs in polynomial time
- this characterisation explicitly refers to a bound
- alternatively NP can be characterised as the class of existential second-order sentence
- completeness for NP of SAT becomes trivial

A Quiz

Questions

- what is the truth value of the following propositional formula

$$(p \to \neg q) \to (\neg q \to \neg p)$$

- give an informal explanation of the following two first-order formulas

$$\forall x \exists y A(x, y) \qquad \exists y \forall x (x < y)$$

- consider Skolemisation, is the following formula the Skolem normal form of $\forall x \exists y A(x, y)$?

$$\forall x \forall y A(x, f(x))$$

- do the following equivalences hold (and how to verify this)?

$$\forall x \forall y A(x, f(x)) \approx \forall x \exists y A(x, y) \approx \exists x \forall y A(x, y)$$

- can we express the following statement about a given graph $\mathcal{G}$ in first-order logic?

  let s and t be nodes in $\mathcal{G}$, then there exists a path of length at most 3 from s to t

BoolTool

- BoolTool is a web-interfaced based tool for manipulation and transformation of formulas in propositional logic
- available at http://cl-informatik.uibk.ac.at/software/booltool/

Propositional Logic

Outline of the Lecture

Propositional Logic

short reminder of propositional logic, soundness and completeness theorem, natural deduction, propositional resolution

First Order Logic

introduction, syntax, semantics, Löwenheim-Skolem, compactness, model existence theorem, natural deduction, completeness, normalisation

Properties of First Order Logic

Craig's Interpolation Theorem, Robinson's Joint Consistency Theorem, Herbrand's Theorem

Limits and Extensions of First Order Logic

Intuitionistic Logic, Curry-Howard Isomorphism, Limits, Second-Order Logic let $p_1, p_2, \dots, p_j, \dots$ denote an infinite set of propositional atoms, denoted by p, q, r; the set of atoms is denoted by AT; $\top$, $\bot$ are truth constants

Definition

the propositional connectives are

$$\neg \qquad \wedge \qquad \vee \qquad \rightarrow$$

Definition

(propositional) formulas are defined as follows

- a propositional atom p and a truth constant is a formula
- if A, B are formulas, then

  $$\neg A \qquad (A \wedge B) \qquad (A \vee B) \qquad (A \rightarrow B)$$

  are formulas

we use precedence: $\neg > \vee, \wedge > \rightarrow$; right-associativity of $\rightarrow$

The Semantics of Propositional Logic

Example

the following expression A is a formula

$$(p \rightarrow \neg q) \rightarrow (\neg q \rightarrow \neg p)$$

Definition

- we write T, F for the two truth values
- an assignment $\mathsf{v}\colon \mathrm{AT} \rightarrow \{\mathrm{T}, \mathrm{F}\}$ maps atoms to truth values
- we write $\mathsf{v}(A)$ for valuation of A, the extension of the assignment to formulas

Definition

the consequence relation $\models$ asserts that $\mathsf{v}(B) = \mathrm{T}$, whenever $\mathsf{v}(A_1), \dots, \mathsf{v}(A_n)$ is true for any assignment v, denoted as $A_1, \dots, A_n \models B$

Definition

we write $\models A$, instead of $\varnothing \models A$ and call A a tautology or valid

Example

let $\mathsf{v}(p) = \mathrm{T}$, $\mathsf{v}(q) = \mathrm{F}$, then

$$\mathsf{v}(A) = \mathsf{v}((p \rightarrow \neg q) \rightarrow (\neg q \rightarrow \neg p)) = \mathrm{F}$$

Definition

- the provability relation $\vdash$ asserts that B is derived from $A_1, \dots, A_n$ in a formal calculus for propositional logic
- (propositional) natural deduction is an example of such a calculus
- we write $A_1, \dots, A_n \vdash B$, if B is derivable from $A_1, \dots, A_n$
- we write $\vdash B$ instead of $\varnothing \vdash B$
- B is called provable, if $\vdash B$ holds

Why Syntax & Semantics?

Question ①

why is it not enough to know when a formula is true, why do we need a "formal calculus"?

Question ②

what is the connection here:

$$A_1, \dots, A_n \vdash B \iff A_1, \dots, A_n \models B$$

Answer

1. historically the proof systems were first
2. study of proof systems led to efficient SAT techniques
3. in designing new logics for applications one starts with the rules, typically the semantics comes later

Soundness & Completeness

Theorem

- $\exists$ *provability relations* $\vdash$ *such that the following holds:*

$$A_1, \dots, A_n \vdash B \iff A_1, \dots, A_n \models B$$

- *we say the calculus underlying* $\vdash$ *is sound and complete*
- *we say propositional logic is (finitely) axiomatised by such a (finite) formal system*

Example

natural deduction for propositional logic

Natural Deduction

	introduction	elimination
$\wedge$	$\dfrac{E \quad F}{E \wedge F}\ \wedge i$	$\dfrac{E \wedge F}{E}\ \wedge : e \qquad \dfrac{E \wedge F}{F}\ \wedge : e$
$\vee$	$\dfrac{E}{E \vee F}\ \vee : i \qquad \dfrac{F}{E \vee F}\ \vee : i$	$\dfrac{E \vee F \quad \boxed{\begin{array}{c}E\\ \vdots\\ G\end{array}} \quad \boxed{\begin{array}{c}F\\ \vdots\\ G\end{array}}}{G}\ \vee : e$
$\rightarrow$	$\dfrac{\boxed{\begin{array}{c}E\\ \vdots\\ F\end{array}}}{E \rightarrow F}\ \rightarrow : i$	$\dfrac{E \quad E \rightarrow F}{F}\ \rightarrow : e$

Natural Deduction (cont'd)

	introduction	elimination
$\neg$	$\dfrac{\boxed{\begin{array}{c}E\\ \vdots\\ \bot\end{array}}}{\neg E}\ \neg : i$	$\dfrac{F \quad \neg F}{\bot}\ \neg : e$
$\bot$		$\dfrac{\bot}{F}\ \bot : e$
$\neg\neg$		$\dfrac{\neg\neg F}{F}\ \neg\neg : e$

Example

derivation of Pierce's law $((p \rightarrow q) \rightarrow p) \rightarrow p$

1	$((p \rightarrow q) \rightarrow p)$	assumption
2	$\neg p$	assumption
3	p	assumption
4	$\bot$	2, $\neg$ elimination
5	q	$\bot$ elimination
6	$p \rightarrow q$	$\rightarrow$ introduction
7	p	1, $\rightarrow$ elimination
8	$\bot$	2, $\neg$ elimination
9	p	derived rule
10	$((p \rightarrow q) \rightarrow p) \rightarrow p$	1, $\rightarrow$ introduction

Theorem

let $A \rightarrow B$ *be valid,* $\exists$ C *such that* $A \rightarrow C$, $C \rightarrow B$ *valid*

interpolant C *contains only variables that occur in* A *and* B

Propositional Resolution

Definition

- a literal is a propositional atom p or its negation $\neg p$
- a formula F is in conjunctive normal form (CNF) if F is a conjunction of disjunctions of literals

formulas A, B are (logically) equivalent ($A \equiv B$) if $A \vDash B$ and $B \vDash A$

Lemma

$\forall$ formula A $\exists$ formula B in CNF such that $A \equiv B$

Definition

a clause is disjunction of literals

- $\square$ is a clause
- literals are clauses
- if C, D are clauses, then $C \lor D$ is a clause

Convention

we use: $p \equiv \neg\neg p$, $\square \lor \square \equiv \square$, $C \lor \square \lor D \equiv C \lor D \equiv D \lor C$

Example

$\neg p$, $p \lor q$, $p \lor \neg q \lor r$, $\square$, $\neg\neg p \lor q$ are clauses; $\neg\neg p \lor q \equiv q \lor p$

Definition

resolution

$$\frac{C \lor p \qquad D \lor \neg p}{C \lor D}$$

factoring

$$\frac{C \lor l \lor l}{C \lor l} \qquad l \text{ a literal}$$

let $\mathcal{C}$ be a set of clauses; define resolution operator $\mathrm{Res}(\mathcal{C})$

- $\mathrm{Res}(\mathcal{C}) = \{D \mid D \text{ is resolvent or factor with premises in } \mathcal{C}\}$
- $\mathrm{Res}^0(\mathcal{C}) = \mathcal{C}$
- $\mathrm{Res}^{n+1}(\mathcal{C}) := \mathrm{Res}^n(\mathcal{C}) \cup \mathrm{Res}(\mathrm{Res}^n(\mathcal{C}))$
- $\mathrm{Res}^*(\mathcal{C}) := \bigcup_{n \geqslant 0} \mathrm{Res}^n(\mathcal{C})$

Observation

(propositional) resolution is a refutation based technique; not competitive with SAT solvers

Lemma

resolution is sound and complete; more precisely if F is a formula and $\mathcal{C}$ its clause form, then F is unsatisfiable iff $\square \in \mathrm{Res}^*(\mathcal{C})$

Example

$$\dfrac{\dfrac{\dfrac{q \lor r \qquad \dfrac{\dfrac{p \lor \neg q \lor r \qquad \neg p \lor r}{\neg q \lor r \lor r}}{\neg q \lor r}}{r \lor r}}{r} \qquad \neg r}{\square}$$

the clause set $\mathcal{C}$ is refutable, hence the CNF represented is unsatisfiable

$$(q \lor r) \land (p \lor \neg q \lor r) \land (\neg p \lor r) \land \neg r$$

Many-Valued Propositional Logics

Question

why do we have only two truth values?

Answer

value unknown

no reason, lets have three: 0, $\frac{1}{2}$, 1

$\neg$	
0	1
$\frac{1}{2}$	$\frac{1}{2}$
1	0

$\land$	0	$\frac{1}{2}$	1
0	0	0	0
$\frac{1}{2}$	0	$\frac{1}{2}$	$\frac{1}{2}$
1	0	$\frac{1}{2}$	1

$\lor$	0	$\frac{1}{2}$	1
0	0	$\frac{1}{2}$	1
$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	1
1	1	1	1

$\to$	0	$\frac{1}{2}$	1
0	1	1	1
$\frac{1}{2}$	$\frac{1}{2}$	1	1
1	0	$\frac{1}{2}$	1

Example

three-valued logic is employed in SQL to handle unknown values

Definition

- let $V \subseteq [0,1]$ be truth values containing 0,1
- a Lukasiewicz assignment (based on V) is a mapping $\mathsf{v}\colon \mathsf{AT} \to V$
- v is extended to a valuation of formulas as follows:

$$\mathsf{v}(\neg A) = 1 - \mathsf{v}(A)$$
$$\mathsf{v}(A \wedge B) = \min\{\mathsf{v}(A), \mathsf{v}(B)\} \qquad \mathsf{v}(A \vee B) = \max\{\mathsf{v}(A), \mathsf{v}(B)\}$$
$$\mathsf{v}(A \to B) \ = \min\{1, 1 - \mathsf{v}(A) + \mathsf{v}(B)\}$$

- A is valid if $\mathsf{v}(A) = 1$ for all assignments based on V

Theorem

1. *(finite- or infinite-valued) Lukasiewicz logic is finitely axiomatisable, that is, there exists a finite sound and complete proof system*
2. *validity for Lukasiewicz logic is decidable (it is* coNP*-complete)*

Application of Many-Valued Logics

- in databases a third truth value is useful to model unknown data
- let $[0,1]$ be the set of truth values: values denotes a probabilities
- finite or infinite-valued logic are often called fuzzy logics
- $\exists$ (subsets of first-order) infinite valued fuzzy logics based on $[0,1]$ that are finitely axiomatisable and decidable
- CADIA (Computer Assisted DIAGnosis) is a series of medical expert systems developed at the Vienna Medical University (since 1980's)

```
IF suspicion of liver metastases by liver palpation
THEN pancreatic cancer
with degree of confirmation 0.3
```

- inference system of CADIAG-2 can be expressed as a infinite valued fuzzy logics
- representation showed inconsistencies in CADIAG-2