Automated Theorem Proving

Georg Moser

Institute of Computer Science @ UIBK

Winter 2015

Organisation

Time and Place

Automated Theorem Proving	Wednesday, 13:15–14:45	3W03
exercise class	Wednesday, 15:00–15:45	3W03

Time and Place

Automated Theorem Proving	Wednesday, 13:15–14:45	3W03
exercise class	Wednesday, 15:00–15:45	3W03

Schedule

week 1	October 9	week 8	November 27
week 2	October 16	week 9	December 4
week 3	October 23	week 10	December 11
week 4	October 30	week 11	December 18
week 5	November 6	week 12	January 15
week 6	November 13	week 13	January 22
week 7	November 20	week 14	January 29
first exam	February 5		

Time and Place

Automated Theorem Proving	Wednesday, 13:15–14:45	3W03
exercise class	Wednesday, 15:00–15:45	3W03

Schedule

week 1	October 9	week 8	November 27
week 2	October 16	week 9	December 4
week 3	October 23	week 10	December 11
week 4	October 30	week 11	December 18
week 5	November 6	week 12	January 15
week 6	November 13		no lecture
week 7	November 20	week 13	January 29
first exam	February 5		

Time and Place

Automated Theorem Proving	Wednesday, 13:15–14:45	3W03
exercise class	Wednesday, 15:00–15:45	3W03

Schedule

week 1	October 9	week 8	November 27
week 2	October 16	week 9	December 4
week 3	October 23	week 10	December 11
week 4	October 30	week 11	December 18
week 5	November 6	week 12	January 15
week 6	November 13		no lecture
week 7	November 20	week 13	January 29
first exam	February 5		

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

Outline of the Module

Advanced Topics in Logic

for example

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

Automated Reasoning

for example

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

Outline of the Lecture

Early Approaches in Automated Reasoning

Herbrand's theorem for dummies, Gilmore's prover, method of Davis and Putnam

Starting Points

resolution, tableau provers, Skolemisation, ordered resolution, redundancy and deletion

Automated Reasoning with Equality

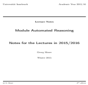
paramodulation, ordered completion and proof orders, superposition

Applications of Automated Reasoning

Neuman-Stubblebinde Key Exchange Protocol, Robbins problem

Literature

- lecture notes
  (3rd edition)


Universität Innsbruck Academic Year 2015/16

Lecture Notes

Module Automated Reasoning

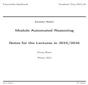
Notes for the Lectures in 2015/2016

Georg Moser

Winter 2015

Literature

- lecture notes
  (3rd edition)


Universität Innsbruck Academic Year 2015/16

Lecture Notes

Module Automated Reasoning

Notes for the Lectures in 2015/2016

Georg Moser

Winter 2015

© G. Moser 3rd 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

Time and Place (cont'd)

Automated Theorem Proving	Friday, 13:15–14:45	3W03
exercise class	Friday, 14:45–15:40	3W03

Time and Place (cont'd)

Automated Theorem Proving	Friday, 13:15–14:45	3W03
exercise class	Friday, 14:45–15:40	3W03

Comments

- officially there are two lectures and one exercise group
- this is not too bright, as the course on theorem proving is based on the course on logic
- however, (budget) constraints require that the courses are held in one term
- implemenation: logic on Wednesday, automated theorem proving on Friday
- exercises will cover both topics

Motivation

Applications of Automated Reasoning

1. Program Analysis

   logical products of interpretations allows the automated combination of simple interpreters

Applications of Automated Reasoning

1. Program Analysis
   logical products of interpretations allows the automated combination of simple interpreters
2. Databases, in particular datalog
   datalog is a declarative language and syntactically it is a subset of Prolog; used in knowledge representation systems

Applications of Automated Reasoning

1. Program Analysis
   logical products of interpretations allows the automated combination of simple interpreters
2. Databases, in particular datalog
   datalog is a declarative language and syntactically it is a subset of Prolog; used in knowledge representation systems
3. Types as Formulas
   the type checking in simple λ-calculus is equivalent to derivability in intuitionistic logic

Applications of Automated Reasoning

1. Program Analysis
   logical products of interpretations allows the automated combination of simple interpreters
2. Databases, in particular datalog
   datalog is a declarative language and syntactically it is a subset of Prolog; used in knowledge representation systems
3. Types as Formulas
   the type checking in simple λ-calculus is equivalent to derivability in intuitionistic logic
4. Complexity Theory
   NP can be characterised as the class of existential second-order sentence

Additional Applications

Application ⑤: Issues of Security

- security protocols are small programs that aim at securing communications over a public network
- design of such protocols is difficult and error-prone
- we will study the use of a first-order theorem prover to show that the Neuman-Stubblebine key exchange protocol can be broken

Additional Applications

Application ⑤: Issues of Security

- security protocols are small programs that aim at securing communications over a public network
- design of such protocols is difficult and error-prone
- we will study the use of a first-order theorem prover to show that the Neuman-Stubblebine key exchange protocol can be broken

Application ⑥: Software Verification

- termination of programs is undecidable (Alan Turing)
- so what: termination of imperative programs can be shown by

  AProVE, Terminator, Julia, COSTA, . . .

  fully automatically . . .
- Terminator uses model-checking

Software Verification

- in the early years of model-checking mainly hardware was analysed like integrated circuits

Software Verification

- in the early years of model-checking mainly hardware was analysed like integrated circuits
- in the last decade the approach was extended to the verification of properties of software

Software Verification

- in the early years of model-checking mainly hardware was analysed like integrated circuits
- in the last decade the approach was extended to the verification of properties of software
- initially only safety properties could be analysed ("nothing bad happens")

Software Verification

- in the early years of model-checking mainly hardware was analysed like integrated circuits
- in the last decade the approach was extended to the verification of properties of software
- initially only safety properties could be analysed ("nothing bad happens")
- recently liveness properties ("something good will happen") became of interest
- termination of a program is a liveness property

Software Verification

- in the early years of model-checking mainly hardware was analysed like integrated circuits
- in the last decade the approach was extended to the verification of properties of software
- initially only safety properties could be analysed ("nothing bad happens")
- recently liveness properties ("something good will happen") became of interest
- termination of a program is a liveness property

Terminator research project

Software Verification

- in the early years of model-checking mainly hardware was analysed like integrated circuits
- in the last decade the approach was extended to the verification of properties of software
- initially only safety properties could be analysed ("nothing bad happens")
- recently liveness properties ("something good will happen") became of interest
- termination of a program is a liveness property

Terminator research project

- developed by Microsoft Research Cambridge

Software Verification

- in the early years of model-checking mainly hardware was analysed like integrated circuits
- in the last decade the approach was extended to the verification of properties of software
- initially only safety properties could be analysed ("nothing bad happens")
- recently liveness properties ("something good will happen") became of interest
- termination of a program is a liveness property

Terminator research project

- developed by Microsoft Research Cambridge
- employs transition invariants, given a program step relation $\to_{\mathsf{P}}$ find finitely many well-founded relations $\mathsf{U}_1, \ldots, \mathsf{U}_n$ whose union contains the transitive closure of $\to_{\mathsf{P}}$

A Bit More on Java

Example

```java
public static int div(int x, int y) {
  int res = 0;
  while (x >= y && y > 0) {
    x = x-y;
    res = res + 1;
  }
  return res;
}
```

A Bit More on Java

Example

```
public static int div(int x, int y) {
  int res = 0;
  while (x >= y && y > 0) {
    x = x-y;
    res = res + 1;
  }
  return res;
}
```

Termination of the example could be proven.

A Bit More on Java (cont'd)

Example

```
public static void test(int n, int m){
  if (0 < n && n < m) {
    int j = n+1;
    while(j<n || j > n){
      if (j>m) j=0 else j=j+1;
    }
  }
}
```

A Bit More on Java (cont'd)

Example

```
public static void test(int n, int m){
  if (0 < n && n < m) {
    int j = n+1;
    while(j<n || j > n){
      if (j>m) j=0 else j=j+1;
    }
  }
}
```

We were unable to show termination of the example.

Outline of the Lecture

Early Approaches in Automated Reasoning

Herbrand's theorem for dummies, Gilmore's prover, method of Davis and Putnam

Starting Points

resolution, tableau provers, Skolemisation, ordered resolution, redundancy and deletion

Automated Reasoning with Equality

paramodulation, ordered completion and proof orders, superposition

Applications of Automated Reasoning

Neuman-Stubblebinde Key Exchange Protocol, Robbins problem

Outline of the Lecture

Early Approaches in Automated Reasoning

Herbrand's theorem for dummies, Gilmore's prover, method of Davis and Putnam

Starting Points

resolution, tableau provers, Skolemisation, ordered resolution, redundancy and deletion

Automated Reasoning with Equality

paramodulation, ordered completion and proof orders, superposition

Applications of Automated Reasoning

Neuman-Stubblebinde Key Exchange Protocol, Robbins problem

Herbrand's Theorem for Dummies

Jacques Herbrand (1908–1931) proposed to

- transform first-order into propositional logic
- basis of Gilmore's prover

Herbrand's Theorem for Dummies

Jacques Herbrand (1908–1931) proposed to

- transform first-order into propositional logic
- basis of Gilmore's prover

$\mathcal{G}$ a set of universal sentences (of $\mathcal{L}$) without $=$

Herbrand's Theorem for Dummies

Jacques Herbrand (1908–1931) proposed to

- transform first-order into propositional logic
- basis of Gilmore's prover

$\mathcal{G}$ a set of universal sentences (of $\mathcal{L}$) without $=$

Theorem

$\mathcal{G}$ is satisfiable iff $\mathcal{G}$ has a Herbrand model (over $\mathcal{L}$)

Herbrand's Theorem for Dummies

Jacques Herbrand (1908–1931) proposed to

- transform first-order into propositional logic
- basis of Gilmore's prover

$\mathcal{G}$ a set of universal sentences (of $\mathcal{L}$) without $=$

Theorem

$\mathcal{G}$ is satisfiable iff $\mathcal{G}$ has a Herbrand model (over $\mathcal{L}$)

see lecture notes

Herbrand's Theorem for Dummies

Jacques Herbrand (1908–1931) proposed to

- transform first-order into propositional logic
- basis of Gilmore's prover

$\mathcal{G}$ a set of universal sentences (of $\mathcal{L}$) without $=$

Theorem

$\mathcal{G}$ is satisfiable iff $\mathcal{G}$ has a Herbrand model (over $\mathcal{L}$)

Gilmore's Prover

Gilmore's Prover (declarative version)

1. F be an arbitrary sentence in base language $\mathcal{L}$

Gilmore's Prover

Gilmore's Prover (declarative version)

1. F be an arbitrary sentence in base language $\mathcal{L}$
2. consider its negation $\neg F$
   wlog $\neg F = \forall x_1 \cdots \forall x_n G(x_1, \ldots, x_n)$ in SNF

Gilmore's Prover

Gilmore's Prover (declarative version)

1. F be an arbitrary sentence in base language $\mathcal{L}$
2. consider its negation $\neg F$
   wlog $\neg F = \forall x_1 \cdots \forall x_n G(x_1, \dots, x_n)$ in SNF
3. consider all possible Herbrand interpretations of $\mathcal{L}$

Gilmore's Prover

Gilmore's Prover (declarative version)

1. F be an arbitrary sentence in base language $\mathcal{L}$
2. consider its negation $\neg F$
   wlog $\neg F = \forall x_1 \cdots \forall x_n G(x_1, \ldots, x_n)$ in SNF
3. consider all possible Herbrand interpretations of $\mathcal{L}$
4. F is valid if $\exists$ finite unsatisfiable subset $S \subseteq \mathrm{Gr}(\neg F)$

Gilmore's Prover

Gilmore's Prover (declarative version)

1. F be an arbitrary sentence in base language $\mathcal{L}$
2. consider its negation $\neg F$
   wlog $\neg F = \forall x_1 \cdots \forall x_n G(x_1, \ldots, x_n)$ in SNF
3. consider all possible Herbrand interpretations of $\mathcal{L}$
4. F is valid if $\exists$ finite unsatisfiable subset $S \subseteq \mathrm{Gr}(\neg F)$

Definition

$$\mathrm{Gr}(\mathcal{G}) = \{G(t_1, \ldots, t_n) \mid \forall x_1 \cdots \forall x_n G(x_1, \ldots, x_n) \in \mathcal{G}, t_i \text{ closed terms}\}$$

Definition

let

$$\mathcal{A} = \{A_0, A_1, A_2, \dots\}$$

be (ground) atomic formulas over Herbrand universe of $\mathcal{L}$

Definition

let

$$\mathcal{A} = \{A_0, A_1, A_2, \dots\}$$

be (ground) atomic formulas over Herbrand universe of $\mathcal{L}$

Definition (Semantic Tree)

the semantic tree T for F:

- the root is a semantic tree

Definition

let

$$\mathcal{A} = \{A_0, A_1, A_2, \dots\}$$

be (ground) atomic formulas over Herbrand universe of $\mathcal{L}$

Definition (Semantic Tree)

the semantic tree T for F:

- the root is a semantic tree
- two edges leaving the root are labelled by A_0 or $\neg A_0$, respectively

Definition

let

$$\mathcal{A} = \{A_0, A_1, A_2, \dots\}$$

be (ground) atomic formulas over Herbrand universe of $\mathcal{L}$

Definition (Semantic Tree)

the semantic tree T for F:

- the root is a semantic tree
- two edges leaving the root are labelled by A_0 or $\neg A_0$, respectively
- let I be a node in T of height n; then I is either a
  1. leaf node or
  2. the edges e_1, e_2 leaving node I are labelled by A_n and $\neg A_n$

Definition

let

$$\mathcal{A} = \{A_0, A_1, A_2, \dots\}$$

be (ground) atomic formulas over Herbrand universe of $\mathcal{L}$

Definition (Semantic Tree)

the semantic tree T for F:

- the root is a semantic tree
- two edges leaving the root are labelled by A_0 or $\neg A_0$, respectively
- let I be a node in T of height n; then I is either a
  1. leaf node or
  2. the edges e_1, e_2 leaving node I are labelled by A_n and $\neg A_n$

Fact

path in T gives rise to a (partial) Herbrand interpretation $\mathcal{I}$ over $\mathcal{L}$

Definition

- let $I \in T$, Herbrand interpretation induced by I is denoted as $\mathcal{I}$

Definition

- let $I \in T$, Herbrand interpretation induced by I is denoted as $\mathcal{I}$
- I is closed, if $\exists\ G \in \mathrm{Gr}(\neg F)$ such that $\mathcal{I} \not\models G$ and thus $\mathcal{I} \not\models \neg F$

Definition

- let $I \in T$, Herbrand interpretation induced by I is denoted as $\mathcal{I}$
- I is closed, if $\exists\ G \in \mathrm{Gr}(\neg F)$ such that $\mathcal{I} \not\models G$ and thus $\mathcal{I} \not\models \neg F$
- note that if I is closed, then $\mathcal{I}$ models the original formula F

Lemma

if all nodes in T are closed then F is valid

Definition

- let $I \in T$, Herbrand interpretation induced by I is denoted as $\mathcal{I}$
- I is closed, if $\exists\ G \in \mathrm{Gr}(\neg F)$ such that $\mathcal{I} \not\models G$ and thus $\mathcal{I} \not\models \neg F$
- note that if I is closed, then $\mathcal{I}$ models the original formula F

Lemma

if all nodes in T are closed then F is valid

Proof.

- suppose all nodes in T are closed
- $\exists$ finite unsatisfiable $S \subseteq \mathrm{Gr}(\neg F)$
- a simple corollary to Herbrand's theorem says that $\neg F$ is unsatisfiable if $\exists$ finite unsatisfiable $S \subseteq \mathrm{Gr}(\neg F)$
- hence $\neg F$ is unsatisfiable, thus F is valid ∎

Definition

the Herbrand universe for a language $\mathcal{L}$ can be constructed iteratively as follows:

$$H_0 := \begin{cases} \{c \mid c \text{ is a constant in } \mathcal{L}\} & \exists \text{ constants} \\ \{c\} & \text{otherwise} \end{cases}$$

$$H_{n+1} := \{f(t_1, \dots, t_k) \mid f^k \in \mathcal{L}, t_1, \dots, t_k \in H_n\}$$

finally $H := \bigcup_{n \geqslant 0} H_n$ denotes the Herbrand universe for $\mathcal{L}$

Definition

the Herbrand universe for a language $\mathcal{L}$ can be constructed iteratively as follows:

$$H_0 := \begin{cases} \{c \mid c \text{ is a constant in } \mathcal{L}\} & \exists \text{ constants} \\ \{c\} & \text{otherwise} \end{cases}$$

$$H_{n+1} := \{f(t_1, \dots, t_k) \mid f^k \in \mathcal{L}, t_1, \dots, t_k \in H_n\}$$

finally $H := \bigcup_{n \geqslant 0} H_n$ denotes the Herbrand universe for $\mathcal{L}$

Definitions

- let $\mathcal{C} = \{C_1, \dots, C_n\}$ be the set of clauses over $\mathcal{L}$, representing $\neg F$[a]
- define $\mathcal{C}'_n$ as the ground instances of $\mathcal{C}$ using only terms from H_n

[a] a clause is a disjunction of literals

Gilmore's Prover in Pseudo-Code

```
begin {
  contr := false;
  n := 0;
  while (not contr) do {
    D' := DNF(C'_n);
    contr := all constituents of D'
                 contain complementary literals;
    n := n + 1;
   }
 }
```

Gilmore's Prover in Pseudo-Code

```
begin {
  contr := false;
  n := 0;
  while (not contr) do {
    D' := DNF(C'_n);
    contr := all constituents of D'
             contain complementary literals;
    n := n + 1;
   }
 }
```

Disadvantages

- generation of all $\mathcal{C}'_n$
- transformation to DNF
- did not yield actual proofs of simple (predicate logic) formulas