

COMP60121: Automated Reasoning, Part II Advanced Topics

Renate Schmidt

School of Computer Science
The University of Manchester

schmidt@cs.man.ac.uk

<http://www.cs.man.ac.uk/~schmidt/COMP6012/>

Overview

- Handouts: These notes
Weekly assessed coursework exercises
Weekly unassessed exercises
Using SPASS (distributed next week)
Timetable (note: approximate!)
Exam definition sheet (let me know of any mistakes)
- Deadlines for assessed coursework: ~~9:00~~ 10:30 on next teaching day; see website
- Demonstrators: ~~Konstantin Korovin~~ Pavel Klinov
~~Nestan Tsiskaridze~~ Hilverd Reker
- Closed book exam

Content

- Emphasis on:
 - ▶ foundation of advanced automated theorem proving
 - ▶ both theory and practice (more theoretical, but motivated by considerations of practical aspects and efficiency)
- Treatment is formal and rigorous; small selection of important topics; many examples and exercises
- **All you need to know for the exam is in the lecture notes, and it is advisable that you can solve the exercises and can do the assignments**
- Good strategy: go through the material discussed in lectures after each lecture, do as many of the unassessed exercises, do the assignments and read ahead. And, ask questions!

Textbooks

- None cover all the material, but recommended are:
 - ▶ Schöning, U. (1989), *Logic for Computer Scientists*. Birkhäuser.
 - ▶ Fitting, M. (1990), *First-Order Logic and Automated Theorem Proving*. Springer.
- Also useful:
 - ▶ Socher-Ambrosius, R., Johann, P. (1997), *Deduction Systems*. Springer.
 - ▶ Goubault-Larrecq, J. and Mackie, I. (1997), *Proof Theory and Automated Deduction*. Kluwer.
 - ▶ Leitsch, A. (1997), *The Resolution Calculus*. Springer.
- All available in the Resources Centre Library and/or the main library. There is no need to buy a book.

COMP60121: Automated Reasoning II

Lecture 1

Previously ... (in Part I)

- Syntax and semantics of propositional logic and FOL
 - variables, constants, terms, quantifiers, ...
- Syntax and semantics of clause logic
 - atoms, literals, clauses, Skolem terms
- Transformation of formulae to clausal form
- Resolution calculus
 - resolution rule
 - factoring rule
 - unification – for first-order clauses

Aims

- To discuss and study optimised transformations into clausal form
- To introduce well-founded orderings
- To introduce ordered resolution with (well-behaved!) selection
- To study and show soundness and refutational completeness of resolution for ground case
- To introduce redundancy elimination
- To discuss applications

Well-Founded Orderings

- To show the refutational completeness of resolution, we make use of the concept of well-founded orderings.
- They are essential for making the search more deterministic and efficient.
- Are used in refinements of resolution.
- Reference:
Baader, F. and Nipkow, T. (1998), *Term rewriting and all that*. Cambridge Univ. Press, Chapter 2.

Basic Properties of Relations

Let R be a binary relation over a set X ($R \subseteq X \times X$).

- R is **transitive** iff
 $\forall x, y, z \in X$, if $R(x, y)$ and $R(y, z)$ then $R(x, z)$.
- R is **irreflexive** iff $\forall x \in X$, $\neg R(x, x)$
- R is **total**, or **linear**, iff
 $\forall x, y \in X$, if $x \neq y$ then $R(x, y)$ or $R(y, x)$
- R^* denotes the **reflexive-transitive closure** of R . I.e.

$$R^* = \bigcup_{n=0}^{\infty} R^n = R^0 \cup R \cup R^2 \cup R^3 \cup \dots$$

where $R^0(x, x)$ for any $x \in X$ and
 $R^{n+1}(x, y)$ iff $\exists z. R(x, z) \wedge R^n(z, y)$ for $n \geq 0$.

– p.9

Orderings

- A **(strict) ordering** on a set X is a transitive and irreflexive binary relation on X , here denoted by $\succ$.
- The pair $(X, \succ)$ is then called a **(strictly) ordered set**.
- An element x of X is **minimal** wrt. $\succ$, if there is no y in X such that $x \succ y$.
- A minimal element x in X is called the **smallest** (or **strictly minimal**) element, if for all $y \in X$ different from x , $y \succ x$.
- **Maximal** and **largest** (or **strictly maximal**) elements are defined analogously.
- **Notation:** $\prec$ for the inverse relation $\succ^{-1}$
 $\succeq$ for the reflexive closure ($\succ \cup =$) of $\succ$, i.e.
 $x \succeq y$ iff either $x \succ y$ or $x = y$

– p.10

Well-Foundedness

- A (strict) ordering $\succ$ over X is called **well-founded** (or **Noetherian** or **terminating**), if there is no infinite decreasing chain $x_0 \succ x_1 \succ x_2 \succ \dots$ of elements $x_i \in X$.

Lemma 1

$(X, \succ)$ is well-founded iff every non-empty subset Y of X has a minimal element.

- **Examples:**

- ▶ Natural numbers: $(\mathbb{N}, >)$

- **Counterexamples:**

- ▶ $(\mathbb{Z}, >)$

- ▶ $(\mathbb{N}, <)$

– p.11

Noetherian Induction (optional)

Property 2 (Noetherian Induction)

Let $(X, \succ)$ be a well-founded ordering, let Q be a property of elements of X .

If for all $x \in X$ the following implication is satisfied

if $Q(y)$ holds, for all $y \in X$ such that $x \succ y$,^a
then $Q(x)$ holds.^b

then

the property $Q(x)$ holds for all $x \in X$.

^ainduction hypothesis

^binduction step

– p.12

Noetherian Induction (optional) (cont'd)

Proof: By contradiction.

Thus, suppose for all $x \in X$ the implication above is satisfied, but $Q(x)$ does not hold for all $x \in X$.

Let $A = \{x \in X \mid Q(x) \text{ is false}\}$. Suppose $A \neq \emptyset$.

Since $(X, \succ)$ is well-founded, A has a minimal element x_1 . Hence for all $y \in X$ with $x_1 \succ y$ the property $Q(y)$ holds.

On the other hand, the implication which is presupposed for this theorem holds in particular also for x_1 , hence $Q(x_1)$ must be true so that x_1 cannot belong to A . *Contradiction*.

– p.13

Multi-Sets

- Multi-sets are “sets which allow repetition”.

E.g.: $\{a, a, b\}$, $\{a, b, a\}$, $\{a, b\}$

- Formally, let X be a set.

A **multi-set** S over X is a mapping $S : X \rightarrow \mathbb{N}$.

- Intuitively, $S(x)$ specifies the number of occurrences of the element x (of the base set X) within S .
- Example:** $S = \{a, a, a, b, b\}$ is a multi-set over $\{a, b, c\}$, where $S(a) = 3$, $S(b) = 2$, $S(c) = 0$.
- We say that x is an **element** of S , if $S(x) > 0$.

– p.14

Multi-Sets (cont'd)

- We use set notation ($\in$, $\subset$, $\subseteq$, $\cup$, $\cap$, etc.) with analogous meaning also for multi-sets, e.g.,

$$(S_1 \cup S_2)(x) = S_1(x) + S_2(x)$$

$$(S_1 \cap S_2)(x) = \min\{S_1(x), S_2(x)\}$$

- A multi-set S over X is called **finite**, if

$$|\{x \in X \mid S(x) > 0\}| < \infty.$$

- From now on we consider finite multi-sets only.**

– p.15

Exercise

Suppose $S_1 = \{c, a, b\}$ and $S_2 = \{a, b, b, a\}$ are multi-sets over $\{a, b, c, d\}$.

Determine $S_1 \cup S_2$ and $S_1 \cap S_2$.

– p.16

Exercise

Suppose $S_1 = \{c, a, b\}$ and $S_2 = \{a, b, b, a\}$ are multi-sets over $\{a, b, c, d\}$.

Determine $S_1 \cup S_2$ and $S_1 \cap S_2$.

Answer:

$$S_1 \cup S_2 = \{a, a, a, b, b, b, c\}$$

$$S_1 \cap S_2 = \{a, b\}$$

– p.16

Multi-Set Orderings

- Let $(X, \succ)$ be an ordering. The multi-set extension $\succ_{\text{mul}}$ of $\succ$ to (finite) multi-sets over X is defined by

$$S_1 \succ_{\text{mul}} S_2 \text{ iff } S_1 \neq S_2 \text{ and}$$

$$\forall x \in X, \text{ if } S_2(x) > S_1(x) \text{ then}$$

$$\exists y \in X : y \succ x \text{ and } S_1(y) > S_2(y)$$

- Example:** Over $(\mathbb{N}, >)$:

$$\{5, 5, 4, 3, 2\} \succ_{\text{mul}} \{5, 4, 4, 3, 3, 2\} \succ_{\text{mul}} \{5, 4, 3\}$$

Exercise: How is the set $\{5, 5\}$ related to each of these?

– p.17

Method for Determining $\succ_{\text{mul}}$ for a Total Ordering $\succ$

1. Remove common occurrences of elements from S_1 and S_2 . Assume this gives S'_1 and S'_2 .
2. Then check that for every element x in S'_2 there is an element $y \in S'_1$ that is larger than x . Then $S_1 \succ_{\text{mul}} S_2$.

- This method facilitates this equivalent definition:

$$S_1 \succ_{\text{mul}} S_2 \text{ iff } S_1 \neq S_2 \text{ and}$$

$$\forall x \in S_2 \setminus S_1. \exists y \in S_1 \setminus S_2. y \succ x$$

– p.18

Reconsider Example

- $S_1 = \{\emptyset, 5, 4, 3, 2\}$ $S_2 = \{\emptyset, 4, 4, 3, 2\}$
 $S'_1 = \{5\}$ $S'_2 = \{4, 3\}$
 $5 > 4$ and $5 > 3$
 Therefore $S_1 \succ_{\text{mul}} S_2$.
- $S_2 = \{\emptyset, 4, 4, 3, 2\}$ $S_3 = \{\emptyset, 4, 3\}$
 $S'_2 = \{4, 3, 2\}$ $S'_3 = \emptyset$
 Therefore $S_2 \succ_{\text{mul}} S_3$.
- Exercise: How does $S_4 = \{5, 3, 2\}$ compare with S_3 ?

– p.19

Reconsider Example

- $S_1 = \{\emptyset, 5, A, \emptyset, \emptyset\}$ $S_2 = \{\emptyset, A, 4, \emptyset, 3, \emptyset\}$
 $S'_1 = \{5\}$ $S'_2 = \{4, 3\}$
 $5 > 4$ and $5 > 3$
Therefore $S_1 >_{\text{mul}} S_2$.
- $S_2 = \{\emptyset, A, 4, \emptyset, 3, 2\}$ $S_3 = \{\emptyset, A, \emptyset\}$
 $S'_2 = \{4, 3, 2\}$ $S'_3 = \emptyset$
Therefore $S_2 >_{\text{mul}} S_3$.
- Exercise: How does $S_4 = \{5, 3, 2\}$ compare with S_3 ?

Answer: $S_4 = \{\emptyset, \emptyset, 2\}$ $S_3 = \{\emptyset, 4, \emptyset\}$
 $S'_4 = \{2\}$ $S'_3 = \{4\}$

– p.19

Reconsider Example

- $S_1 = \{\emptyset, 5, A, \emptyset, \emptyset\}$ $S_2 = \{\emptyset, A, 4, \emptyset, 3, \emptyset\}$
 $S'_1 = \{5\}$ $S'_2 = \{4, 3\}$
 $5 > 4$ and $5 > 3$
Therefore $S_1 >_{\text{mul}} S_2$.
- $S_2 = \{\emptyset, A, 4, \emptyset, 3, 2\}$ $S_3 = \{\emptyset, A, \emptyset\}$
 $S'_2 = \{4, 3, 2\}$ $S'_3 = \emptyset$
Therefore $S_2 >_{\text{mul}} S_3$.
- Exercise: How does $S_4 = \{5, 3, 2\}$ compare with S_3 ?

Answer: $S_4 = \{\emptyset, \emptyset, 2\}$ $S_3 = \{\emptyset, 4, \emptyset\}$
 $S'_4 = \{2\}$ $S'_3 = \{4\}$
Therefore $S_3 >_{\text{mul}} S_4$.

– p.19

Properties of Multi-Set Orderings

Property 3

Let $\succ$ be an ordering. Then

- $\succ_{\text{mul}}$ is an ordering.
- if $\succ$ well-founded then $\succ_{\text{mul}}$ well-founded.
- if $\succ$ total then $\succ_{\text{mul}}$ total

– p.20

Summary

- (strict) orderings
- well-founded orderings
- Noetherian (well-founded) induction
- multi-sets
- multi-set ordering $\succ_{\text{mul}}$
= multi-set extension of ordering $\succ$ on the elements

– p.21

COMP60121: Automated Reasoning II

Lecture 2

Previously ...

- (strict) ordering
- well-founded orderings
- multi-set orderings
- multi-sets

Recall from Part I: Literals, Clauses

- Literals

$L ::= A$ (atom, positive literal)
| $\neg A$ (negative literal)

- Clauses

$C, D ::= \perp$ (empty clause)
| $L_1 \vee \dots \vee L_k, k \geq 1$ (non-empty clause)

- Note:

- ▶ We assume $\vee$ is associative and commutative, repetitions matter.
- ▶ I.e. from now on we regard clauses as multi-sets of literals and interpret, and denote, them as disjunctions.
- ▶ Thus, $C = P \vee P \vee \neg Q$ is identical to $C' = P \vee \neg Q \vee P$. But neither C nor C' are the same as $D = P \vee \neg Q$.

– p.24

Notation in Part II

- Notation:

A, B	atoms
L	literals
C, D	clauses
$\perp$	the empty clause / falsum
N	sets of clauses
F, G	formulae

- $N \models C$ means 'any model which satisfies N also satisfies C '
What does $N \models \perp$ mean?
- $N \vdash_{Res} C$ means ' C is derivable from N using the rules of Res '
i.e. there is a proof of C from N in the calculus Res

Notation in Part II

- Notation:

A, B	atoms
L	literals
C, D	clauses
$\perp$	the empty clause / falsum
N	sets of clauses
F, G	formulae

- $N \models C$ means 'any model which satisfies N also satisfies C '
What does $N \models \perp$ mean? **Answer: N is unsatisfiable**
- $N \vdash_{Res} C$ means ' C is derivable from N using the rules of Res '
i.e. there is a proof of C from N in the calculus Res

– p.25

Unrestricted Resolution Calculus (Variation from Part I)

- Propositional/ground resolution calculus Res**

$$\frac{C \vee A \quad \neg A \vee D}{C \vee D} \quad (\text{resolution})$$

$$\frac{C \vee A \vee A}{C \vee A} \quad ((\text{positive !}) \text{ factoring})$$

- Terminology: $C \vee D$ is the **resolvent**; $C \vee A$ is the **(positive) factor**
 A is the atom **resolved upon**, resp. **factored upon**
- These are **schematic inference rules**; for each substitution of the **schematic variables** C , D , and A , respectively, by ground clauses and ground atoms we obtain an inference rule.
- Since we assume $\vee$ is associative and commutative, note that A and $\neg A$ can occur anywhere in their respective clauses.

– p.26

Recap: Transformation to clausal form

- Basic (naive) algorithm:
 - Transform into **prenex normal form (PNF)**: move quantifiers to the front
 $\rightsquigarrow Q_1 x_1 \dots Q_n x_n G$ (G quantifier-free)
 - Skolemisation**: eliminate quantifiers
 $\rightsquigarrow$ quantifier-free formula
 - Transform into **conjunctive normal form (CNF)**
 $\rightsquigarrow C_1 \wedge \dots \wedge C_n$
 - Clausify**
 $\rightsquigarrow$ set of clauses $N = \{C_1, \dots, C_n\}$
- Let $Cl(F)$ denote **clausal form** N of F

– p.27

Recap: Conversion into Conjunctive Normal Form

- The **conjunctive normal form** $CNF(F)$ of a formula F can be computed by applying these rewrite rules:

$$\begin{array}{lll}
 F \leftrightarrow G & \Rightarrow_{CNF} & (F \rightarrow G) \wedge (G \rightarrow F) \\
 F \rightarrow G & \Rightarrow_{CNF} & (\neg F \vee G) \\
 \neg(F \vee G) & \Rightarrow_{CNF} & (\neg F \wedge \neg G) \\
 \neg(F \wedge G) & \Rightarrow_{CNF} & (\neg F \vee \neg G) \\
 \neg\neg F & \Rightarrow_{CNF} & F \\
 (F \wedge G) \vee H & \Rightarrow_{CNF} & (F \vee H) \wedge (G \vee H) \\
 F \wedge \top & \Rightarrow_{CNF} & F \\
 F \vee \top & \Rightarrow_{CNF} & \top \\
 \neg\top & \Rightarrow_{CNF} & \perp \\
 F \wedge \perp & \Rightarrow_{CNF} & \perp \\
 F \vee \perp & \Rightarrow_{CNF} & F \\
 \neg\perp & \Rightarrow_{CNF} & \top
 \end{array}$$

- The rules may be applied in any order.
- The rules are to be applied modulo associativity and commutativity of $\wedge$ and $\vee$.

– p.28

Conversion into Conjunctive Normal Form (cont'd)

- The first five rules compute the **negation normal form (NNF)** of a formula.
- For every formula F :
 - ▶ $\models F \leftrightarrow \text{CNF}(F)$
 - ▶ $\models F \leftrightarrow \text{NNF}(F)$
- Conversion to CNF (and therefore clause form) may produce a formula whose size is **exponential** in the size of the original formula.

– p.29

Relaxing the Requirements

- The goal
“find a formula G in CNF such that: $\models F \leftrightarrow G$ ”
is therefore impractical.
- For every closed f.o. formula F :
 - ▶ $\models \text{Cls}(F) \rightarrow F$, but not conversely.
 - ▶ F is (un)satisfiable iff $\text{Cls}(F)$ is (un)satisfiable.
- Since we can only preserve satisfiability-equivalence anyway, there is lots of room for optimisation.
- We therefore relax the requirement to
“find a formula G in CNF such that:
 F is satisfiable iff G is satisfiable”
and can get efficient transformations.

– p.30

Structural transformation

- Assume the context is propositional logic.
- **Structural transformation** (or **renaming**) exploits $(\dagger)$, left-to-right:

Property 4

$$F[G] \text{ is satisfiable iff } F[Q] \wedge (Q \leftrightarrow G) \text{ is satisfiable.} \quad (\dagger)$$

provided Q is a (fresh) propositional variable not occurring in $F[G]$.

- Use this as a rule which introduces a new propositional symbol Q for any subformula G of F . View Q as an abbreviation for G .
- We can use this rule recursively for any subformulas in the original formula (this introduces a linear number of new symbols).
- Conversion of the resulting formula to CNF increases the size only by an additional constant factor, **provided we have introduced a new symbol for each subformula** (each formula $Q \leftrightarrow G$ gives rise to at most one application of the distributivity law).

– p.31

Example

- $(P \vee R) \rightarrow P$ is satisfiable iff

$$\begin{aligned} & (Q_0 \rightarrow P) && Q_0 \text{ substituted for } (P \vee R) \\ & \wedge (Q_0 \leftrightarrow (P \vee R)) \end{aligned}$$

is satisfiable iff

$$\begin{aligned} & Q_1 && Q_1 \text{ substituted for } (Q_0 \rightarrow P) \\ & \wedge (Q_1 \leftrightarrow (Q_0 \rightarrow P)) \\ & \wedge (Q_0 \leftrightarrow (P \vee R)) \end{aligned}$$

is satisfiable

– p.32

Exercise

- Compute the structural transformation of $\neg P \vee (R \wedge P)$ in which a new symbol is introduced for every non-literal subformula (in the original formula).

– p.33

Exercise

- Compute the structural transformation of $\neg P \vee (R \wedge P)$ in which a new symbol is introduced for every non-literal subformula (in the original formula).

$$\neg P \vee (R \wedge P)$$

$$Q_0 \quad Q_1$$

Answer: Q_0

$$\wedge (Q_0 \leftrightarrow (\neg P \vee Q_1))$$

$$\wedge (Q_1 \leftrightarrow (R \wedge P))$$

Any answer equivalent modulo the use of introduced symbols with different names is correct.

– p.33

Optimising Structural Transformation

- A further improvement is possible by taking the **polarity** of the subformula F into account.
- Assume that F contains neither $\rightarrow$ nor $\leftrightarrow$.
- A subformula G of F has **positive polarity** in F , if it occurs in the scope of an even number of negation symbols. It has **negative polarity** in F , if it occurs in the scope of an odd number of negation symbols.

– p.34

Formal definition of polarity

- The notion of (positive and negative) polarity can be inductively defined as follows.
 - ▶ F has **positive polarity** in F .
 - ▶ Suppose G is a subformula of F .
 - If $G = \neg G'$ then G' has **positive (negative)** polarity in F if G has **negative (positive)** polarity in F .
 - If $G = G_1 \star G_2$ where $\star \in \{\vee, \wedge\}$ then G_1 and G_2 have **positive (negative)** polarity if G has **positive (negative)** polarity.

– p.35

Exercise

- What is the polarity of each subformula in each of the following?

1. $\neg P \vee (R \wedge P)$

2. $\neg P \vee \neg(R \wedge P)$

– p.36

Exercise

- What is the polarity of each subformula in each of the following?

1. $\neg P \vee (R \wedge P)$

$\neg P \vee (R \wedge P)$

$+ - + + + +$

$R \wedge P$ does not occur in the scope of $\neg$.

2. $\neg P \vee \neg(R \wedge P)$

$\neg P \vee \neg(R \wedge P)$

$+ - + + - - -$

$R \wedge P$ occurs in the scope of one $\neg$.

– p.36

Optimising Structural Transformation (cont'd)

Property 5

Let $F[G]$ be a formula containing neither $\rightarrow$ nor $\leftrightarrow$;
let Q be a propositional variable not occurring in $F[G]$.

- $F[G]$ is satisfiable iff $F[Q] \wedge (Q \rightarrow G)$ is satisfiable, provided G has positive polarity in F .
- $F[G]$ is satisfiable iff $F[Q] \wedge (G \rightarrow Q)$ is satisfiable, provided G has negative polarity in F .

Structural transformation has many uses and can be generalised to include $\rightarrow$ and $\leftrightarrow$, as well as first-order logic.

– p.37

Exercise

- What is the optimised structural transformation of $\neg P \vee (R \wedge P)$ in which a new symbol is introduced for every non-literal subformula (in the original formula)?

– p.38

Exercise

- What is the optimised structural transformation of $\neg P \vee (R \wedge P)$ in which a new symbol is introduced for every non-literal subformula (in the original formula)?

$$\begin{aligned} & \neg P \vee (R \wedge P) \\ & \quad Q_0 \quad Q_1 \\ \rightsquigarrow & Q_0 \\ & \wedge (Q_0 \rightarrow (\neg P \vee Q_1)) \\ & \wedge (Q_1 \rightarrow (R \wedge P)) \end{aligned}$$

– p.38

Exercise (cont'd)

- What is it for $\neg P \vee \neg(R \wedge P)$?

$$\begin{aligned} & \neg P \vee \neg(R \wedge P) \\ & \quad Q_0 \quad Q_1 \quad Q_2 \\ \rightsquigarrow & Q_0 \\ & \wedge (Q_0 \rightarrow (\neg P \vee Q_1)) \\ & \wedge (Q_1 \rightarrow \neg Q_2) \\ & \wedge (Q_2 \leftarrow (R \wedge P)) \end{aligned}$$

Note that when we write ‘introduce new symbols for every non-literal subformula’ we are referring to the subformulae of the original formula.

– p.39

Summary

- language of resolution:
 - atoms
 - literals (positive & negative)
 - clauses (= multi-sets)
- calculus Res:
 - resolution & positive factoring
- optimised conversion to clause form:
 - structural transformation

– p.40

COMP60121: Automated Reasoning II

Lecture 3