Logik für Informatiker
Tools for propositional logic

Till Mossakowski

WiSe 2007/08

Recall: Strategies in Fitch

- Always try to match the situation in your proof with the rules in the book (see book appendix for a complete list)
- Look at the main connective in a premise, apply the corresponding elimination rule (forwards)
- Or: look at the main connective in the conclusion, apply the corresponding introduction rule (backwards)

Recall: Conjunctive Normal Form (CNF)

For each propositional sentence, there is an equivalent sentence of form

$$(\varphi_{1,1} \vee \ldots \vee \varphi_{1,m_1}) \wedge \ldots \wedge (\varphi_{n,1} \vee \ldots \vee \varphi_{n,m_n})$$

where the $\varphi_{i,j}$ are literals, i.e. atomic sentences or negations of atomic sentences.

A sentence in CNF is called a Horn sentence, if each disjunction of literals contains at most one positive literal.

Examples of Horn sentences

$$\neg Home(claire) \wedge (\neg Home(max) \vee Happy(carl))$$

$$Home(claire) \wedge Home(max) \wedge \neg Home(carl)$$

$$Home(claire) \vee \neg Home(max) \vee \neg Home(carl)$$

$$Home(claire) \wedge Home(max) \wedge (\neg Home(max) \vee \neg Home(max))$$

Examples of non-Horn sentences

$$\neg Home(claire) \wedge (Home(max) \vee Happy(carl))$$

$$(Home(claire) \vee Home(max) \vee \neg Happy(claire)) \wedge Happy(carl)$$

$$Home(claire) \vee (Home(max) \vee \neg Home(carl)$$

Alternative notation for the conjuncts in Horn sentences

$\neg A_1 \vee \ldots \vee \neg A_n \vee B$	$(A_1 \wedge \ldots \wedge A_n) \rightarrow B$
$\neg A_1 \vee \ldots \vee \neg A_n$	$(A_1 \wedge \ldots \wedge A_n) \rightarrow \bot$
B	$\top \rightarrow B$
$\bot$	$\square$

Any Horn sentence is equivalent to a conjunction of conditional statements of the above four forms.

Satisfaction algorithm for Horn sentences

1. For any conjunct $\top \rightarrow B$, assign TRUE to B.
2. If for some conjunct $(A_1 \wedge \ldots \wedge A_n) \rightarrow B$, you have assigned TRUE to $A_1, \ldots, A_n$ then assign TRUE to B.
3. Repeat step 2 as often as possible.
4. If there is some conjunct $(A_1 \wedge \ldots \wedge A_n) \rightarrow \bot$ with TRUE assigned to $A_1, \ldots, A_n$, the Horn sentence is not satisfiable. Otherwise, assigning FALSE to the yet unassigned atomic sentences makes all the conditionals (and hence also the Horn sentence) true.

Correctness of the satisfaction algorithm

Theorem The algorithm for the satisfiability of Horn sentences is correct, in that it classifies as tt-satisfiable exactly the tt-satisfiable Horn sentences.

Propositional Prolog

$$AncestorOf(a,b) :- MotherOf(a,b).$$
$$AncestorOf(b,c) :- MotherOf(b,c).$$
$$AncestorOf(a,b) :- FatherOf(a,b).$$
$$AncestorOf(b,c) :- FatherOf(b,c).$$
$$AncestorOf(a,c) :- AncestorOf(a,b), AncestorOf(b,c).$$
$$MotherOf(a,b). \qquad FatherOf(b,c). \qquad FatherOf(b,d).$$

To ask whether this database entails B, Prolog adds $\bot \leftarrow B$ and runs the Horn algorithm. If the algorithm fails, Prolog answers "yes", otherwise "no".

Clauses

A clause is a finite set of literals.

Examples:

$$C_1 = \{Small(a), Cube(a), BackOf(b, a)\}$$
$$C_2 = \{Small(a), Cube(b)\}$$
$$C_3 = \emptyset \quad (\text{ also written } \square)$$

Any set $\mathcal{T}$ of sentences in CNF can be replaced by an equivalent set $\mathcal{S}$ of clauses: each conjunct leads to a clause.

Resolution

A clause R is a resolvent of clauses C_1, C_2 if there is an atomic sentence A with $A \in C_1$ and $(\neg A) \in C_2$, such that

$$R = C_1 \cup C_2 \setminus \{A, \neg A\}.$$

Resolution algorithm: Given a set $\mathcal{S}$ of clauses, systematically add resolvents. If you add $\square$ at some point, then $\mathcal{S}$ is not satisfiable. Otherwise, it is satisfiable.

Example

We start with the CNF sentence:

$$\neg A \wedge (B \vee C \vee B) \wedge (\neg C \vee \neg D) \wedge (A \vee D) \wedge (\neg B \vee \neg D)$$

In Clause form:

$$\{\neg A\},\ \{B, C\},\ \{\neg C, \neg D\},\ \{A, D\},\ \{\neg B, \neg D\}$$

Apply resolution:

$$\dfrac{\dfrac{\{A,D\} \quad \{\neg A\}}{\{D\}} \qquad \dfrac{\dfrac{\{B,C\} \quad \{\neg C,\neg D\}}{\{B,\neg D\}} \qquad \{\neg B,\neg D\}}{\{\neg D\}}}{\square}$$

Soundness and completeness

Theorem Resolution is sound and complete. That is, given a set $\mathcal{S}$ of clauses, it is possible to arrive at $\square$ by successive resolutions if and only if $\mathcal{S}$ is not satisfiable.

This gives us an alternative sound and complete proof calculus by putting

$$\mathcal{T} \vdash S$$

iff with resolution, we can obtain $\square$ from the clausal form of $\mathcal{T} \cup \{\neg S\}$.

Davis-Putnam-Logemann-Loveland algorithm

- backtracking algorithm:
  - select a literal,
  - assign a truth value to it,
  - simplify the formula,
  - recursively check if the simplified formula is satisfiable
    - ▷ if this is the case, the original formula is satisfiable;
    - ▷ otherwise, do the recursive check with the opposite truth value.
- Implementations: mChaff, zChaff, darwin
- Crucial: design of the literal selection function

Optimizations in DPLL

- If a clause is a unit clause, i.e. it contains only a single unassigned literal, this clause can only be satisfied by assigning the necessary value to make this literal true $\Rightarrow$ reduction of search space
- Pure literal elimination: If a propositional variable occurs with only one polarity in the formula, it is called pure $\Rightarrow$ the assignment is clear

DPLL in pseudo code

```
function DPLL(Φ)
    if Φ is a consistent set of literals
        then return true;
    if Φ contains an empty clause
        then return false;
    for every unit clause l in Φ
        Φ=unit-propagate(l, Φ);
    for every literal l that occurs pure in Φ
        Φ=pure-literal-assign(l, Φ);
    l := select-literal(Φ);
    return DPLL(Φ∧l) OR DPLL(Φ∧not(l));
```

Common Algebraic Specification Language

- strongly typed; types are declated using the sort keyword

```
sort Blocks
```

- predicates have to be declared with their types

```
preds Cube, Dodec, Tet : Blocks
```

- propositional variables = nullary predicates

```
preds A,B,C : ()
```

- constants have to be declared with their types

```
ops a,b,c : Blocks
```

Heterogeneous Tool Set

- Reads and checks CASL specifications
- Can prove %implied sentences using resolution provers
  - use "Prove" menu of a node, select SPASS
- Can find models of sets of sentences using DPLL
  - use "Check consistency" menu of a node, select darwin
- available at `http://www.dfki.de/sks/hets`.
  - use the daily version
  - Windows users: use the live CD

Example CASL specification: propositions

```
spec Props =
  preds A,B,C : ()
  . A
  . not (A /\ B)
  . C => B
  . not C %implied
end
```

Example CASL specification: blocks

```
spec Tarski1 =      sort Blocks
  preds Cube, Dodec, Tet, Small, Medium, Large : Blocks
  ops a,b,c : Blocks
  . not a=b . not a=c  . not b=c
  . Small(a) => Cube(a)     %(small_cube_a)%
  . Small(a) <=> Small(b)   %(small_a_b)%
  . Small(b) \/ Medium(b)   %(small_medium_b)%
  . Medium(b) => Medium(c) %(medium_b_c)%
  . Medium(c) => Tet(c)     %(medium_tet_c)%
  . not Tet(c)              %(not_tet_c)%
  . Cube(a)                 %(cube_a)% %implied
  . Cube(b)                 %(cube_b)% %implied
```

Exercises

Discussion: Mo. 26th November

Due: Mo. 3rd December, 10h

- book: 17.17 - 17.45
- Logelei (grade 1)
- Sudoku (grade 1)

Logelei

Translate the following into CASL and solve it with Hets.

Als ich unlängst mit der Bahn fuhr, stiegen in Bremen drei Teenager zu, ein Junge und zwei Mädchen. Der Junge erkundigte sich bei seinen Begleiterinnen- sie heißen Olga und Petra-, wer von den 14 Jungen aus ihrer Klasse an einer geplanten >>Superparty<< teilnehmen würde. Das war offensichtlich ein heißes Thema; denn die Mädchen schienen geradezu begierig zu sein, es zu diskutieren.
Olga begann mit der Auskunft: „Wenn weder Bernd noch Christian kommt, dann nimmt auch Norbert nicht an der Party teil." Darauf Petra: „Wenn Dieter und Norbert kommen, dann wird auch Elgar erscheinen."
Olga: „Nimmt Ingo teil, dann feiert, sofern Gerd nicht kommt, Haug ebenfalls mit."
Petra: „Falls Fabian nicht teilnimmt, wird Jürgen, sofern Martin mitmacht, nicht kommen."
Olga: „Kommen sowohl Haug als auch Axel, dann bleibt Christian der Party fern."
Petra: „Wenn Ingo kommt, wird Jürgen nicht teilnehmen."
Olga: „Aber wenn Kai kommt, wird auch Lars kommen."
Petra: „Feiert Martin mit, wird, sofern Lars nicht kommt, Bernd an der Party teilnehmen."
Olga: „Kommt weder Axel noch Kai, so wird auch Christian der Party fernbleiben."
Der Zug erreichte Osnabrück, wo die frei Teenager ausstiegen. Der Junge, der das Abteil als Letzter verließ, drehte sich noch einmal um und raunte mir zu:
„Unser Gespräch wird sie ein wenig verwirrt haben. Sie müssen dabei bedenken, dass eines der beiden Mädchen stets die Wahrheit sagt, hingegen das andere nie eine Aussage über die Lippen bringt, die der vollen Wahrheit entspricht."
Ehe ich mich erkundigen konnte, wer die Lügnerin war, hatte der Junge den Zug schon verlassen. Seither grüble ich darüber, wer wohl zu der Superparty kommen wird und wer nicht.
Wer??!

Sudoku

Translate the following into CASL and solve it with Hets.

		2					8	9
	4				8	2	7	
					6			4
	7		9					
3		8	5					
	9	7	3				6	5
	8		7	2				
4								8