13 Automated Reasoning

Chapter Objective

- Learn about general-purpose theorem proving in predicate calculus.

The problem

- Given:
  a knowledge base (a set of sentences)
- Prove:
  a sentence

Formally,

- Given:
  a Knowledge Base (KB),
  a sentence α
- Show whether:
  KB $\models \alpha$ (does KB entail α ? Or
  does α follow from KB ?)

The tool

- Modus ponens
  KB: $p \rightarrow q$
  p
  question: q

  answer: yes

  $\{p \rightarrow q, p\} \models \{q\}$
- We can form arbitrarily long “chains” of inference to prove a sentence
- We can use forward or backward reasoning

Example

- If Mary goes to a party, Jane also does.
  If Jane goes to a party, she cannot study.
  If Jane cannot study, she fails.
  Mary went to a party.
- Can we prove:
  Jane will fail.

Example

- If Mary goes to a party, Jane also does.
  M J
  If Jane goes to a party, she cannot study.
  J C
  If Jane cannot study, she fails.
  C F
  Mary went to a party.
  M
- Can we prove:
  Jane will fail.
  F

Does $\{M \rightarrow J, J \rightarrow C, C \rightarrow F, M\}$ entail $\{F\}$?

Example

1. $M \rightarrow J$
2. $J \rightarrow C$
3. $C \rightarrow F$
4. M

Modus ponens on 1 and 4:

5. J

Modus ponens on 2 and 5:

6. C

Modus ponens on 3 and 6:

7. F proven!

Another tool

- Modus tollens
  KB: $p \rightarrow q$
  $\neg q$
  entails $\neg p$.
- So, a theorem proving process involves applying such rules until the desired sentence is proven.
- We call this a "proof" because the rules we use are ***sound*** (correct).

Using modus ponens

• solves a lot of practical problems and is fairly efficient in terms of "searching" for a proof.

• Unfortunately, fails to prove some sentences which should be entailed by a KB (it is *incomplete*)

Example

If Mary goes to the party, Jane also will.
M J

If Mary does not go to the party, Jane will.
¬M J

{ M → J, ¬M → J} should entail {J} because either M is true, or ¬M is true and either way J is true.

But we cannot prove J using modus ponens.

We need a more general rule to cover such situations.

Resolution

• Unit resolution

{p ∨ q, ¬q} entails {p}

• Generalized resolution

{p ∨ q, ¬q ∨ r} entails {p ∨ r}

• Example
1. M → J ¬M ∨ J
2. ¬M → J M ∨ J

Resolution on 1 and 2
3. J ∨ J J proven!

Resolution refutation

• What we just did was to resolve the sentences in the KB (an any new sentences added) to see if they entail a particular sentence.

• The general technique is to <u>add the negation of the sentence to be proven</u> to the KB and see if this leads to a contradiction. In other words, if the KB becomes inconsistent with the addition of the negated sentence, then the original sentence must be true.

•This is called *resolution refutation*.

Resolution refutation (cont'd)

Redo the example:

KB contains
1. M → J ¬M ∨ J
2. ¬M → J M ∨ J

Question: Is J entailed?
Add its negation to the KB:
3. ¬ J

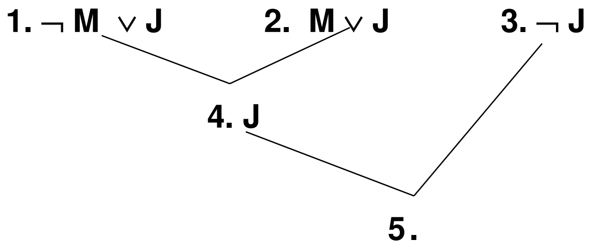
Resolve 1 and 2:
4. J ∨ J J

Resolve 4 and 3:

5. Contradiction (null result)
Hence J must be true.

Resolution refutation (cont'd)

Pictorially:



Steps for Resolution Refutation proofs

- **Put the premises or axioms into clause form (12.2.2)**
- **Add the negation of what is to be proved, in clause form, to the set of axioms**
- **Resolve these clauses together, producing new clauses that logically follow from them (12.2.3)**
- **Produce a contradiction by generating the empty clause**
- **The substitutions used to produce the empty clause are those under which the opposite of the negated goal is true (12.2.4)**

Putting sentences into clause form

1. Eliminate → using
$a \rightarrow b \equiv \neg a \lor b$

2. Reduce the scope of negations. Transformations include:

$\neg(\neg a) \equiv a$

$\neg(\exists X)\, a(X) \equiv (\forall X) \neg a(X)$

$\neg(\forall X)\, a(X) \equiv (\exists X) \neg a(X)$

$\neg(a \lor b) \equiv \neg a \land \neg b$

$\neg(a \land b) \equiv \neg a \lor \neg b$

Putting sentences into clause form (cont'd)

3. Standardize variables apart: rename all variables so that variables bound by different quantifiers have unique names

4. Move all quantifiers to the left without changing their order

5. Eliminate all existential quantifiers using *Skolemization*.

It's the process of giving a name to an object that must exist.

6. Drop all universal quantifiers (allright to do so now)

Putting sentences into clause form (cont'd)

7. Convert the expression into a conjunct of disjuncts form

Eventually each part of an $\wedge$'ed sentence will be separated, and we want the separated sentences to be disjuncts. So,

$a \wedge (b \vee c)$

is fine, whereas

$a \vee (b \wedge c)$ must be distributed to form
$(a \vee b) \wedge (a \vee c)$

Putting sentences into clause form (cont'd)

8. Call each conjunct a separate clause.

9. Standardize the variables apart again.

Using this procedure, <u>any</u> set of statements can be converted to the canonical form.

Resolution refutation is *complete*, i.e., if a sentence can be entailed (proven) it will be.

More on Skolemization

• It is a simple matter to replace every existentially quantified variable with a unique, new constant and drop the quantifier:

$\exists$X (happy (X)) may be replaced by any of the following:

happy(no-name)
happy(X#123)
happy(k1)

no-name, X#123, and k1 are *Skolem constants*. They should not appear in any other sentence in the KB .

Example

- All people who are graduating are happy.
  All happy people smile.
  John-doe is graduating.

  Is John-doe smiling?
- First convert to predicate logic
  $\forall X$ graduating(X) $\rightarrow$ happy(X)
  $\forall X$ happy(X) $\rightarrow$ smiling(X)
  graduating (john-doe)

  smiling(john-doe) negate this: $\neg$ smiling(john-doe)
- Then convert to canonical form

Example (cont'd)

1. $\forall X$ graduating(X) $\rightarrow$ happy(X)
2. $\forall X$ happy(X) $\rightarrow$ smiling(X)
3. graduating (john-doe)
4. $\neg$ smiling(john-doe)

Then convert to canonical form:

Step 1. Eliminate $\rightarrow$

1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
2. $\forall X \neg$ happy (X) $\vee$ smiling (X)
3. graduating (john-doe)
4. $\neg$ smiling (john-doe)

Example (cont'd)

1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
2. $\forall X \neg$ happy (X) $\vee$ smiling (X)
3. graduating (john-doe)
4. $\neg$ smiling (john-doe)

Step 2. Reduce the scope of $\neg$

Step 3. Standardize variables apart

1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
2. $\forall Y \neg$ happy (Y) $\vee$ smiling (Y)
3. graduating (john-doe)
4. $\neg$ smiling (john-doe)

Example (cont'd)

1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
2. $\forall Y \neg$ happy (Y) $\vee$ smiling (Y)
3. graduating (john-doe)
4. $\neg$ smiling (john-doe)

Step 4. Move all quantifiers to the left

Step 5. Eliminate $\exists$

Step 6. Drop all $\forall$

1. $\neg$ graduating (X) $\vee$ happy (X)
2. $\neg$ happy (Y) $\vee$ smiling (Y)
3. graduating (john-doe)
4. $\neg$ smiling (john-doe)

Example (cont'd)

1. $\neg$ graduating (X) $\vee$ happy (X)
2. $\neg$ happy (Y) $\vee$ smiling (Y)
3. graduating (john-doe)
4. $\neg$ smiling (john-doe)

Step 7. Convert to conjunct of disjuncts form

Step 8. Make each conjunct a separate clause.

Step 9. Standardize variables apart again.

Ready for resolution!

Example (cont'd)

4.$\neg$ smiling (john-doe)   2. $\neg$ happy (Y) $\vee$ smiling (Y)

{john-doe/Y}

5. $\neg$ happy (john-doe)

1. $\neg$ graduating (X) $\vee$ happy (X)

{john-doe/X}

6. $\neg$ graduating (john-doe)

3. graduating (john-doe)

7.

Proving an existentially quantified sentence

- **All people who are graduating are happy.
  All happy people smile.
  Someone is graduating.**

  Is someone smiling?

- **First convert to predicate logic**
  $\forall$X graduating(X) $\rightarrow$ happy(X)
  $\forall$X happy(X) $\rightarrow$ smiling(X)
  $\exists$ X graduating (X)

  $\exists$ X smiling(X)   negate this: $\neg\exists$ X smiling(X)

- **Then convert to canonical form**

Example

1. $\forall$X graduating(X) $\rightarrow$ happy(X)
2. $\forall$X happy(X) $\rightarrow$ smiling(X)
3. $\exists$ X graduating (X)
4. $\neg\exists$ X smiling (X)

Then convert to canonical form:

Step 1. Eliminate $\rightarrow$

1. $\forall$X $\neg$ graduating (X) $\vee$ happy (X)
2. $\forall$X $\neg$ happy (X) $\vee$ smiling (X)
3. $\exists$ X graduating (X)
4. $\neg\exists$ X smiling (X)

Example (cont'd)

1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
2. $\forall X \neg$ happy (X) $\vee$ smiling (X)
3. $\exists$ X graduating (X)
4. $\neg \exists$ X smiling (X)

Step 2. Reduce the scope of negation.

1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
2. $\forall X \neg$ happy (X) $\vee$ smiling (X)
3. $\exists$ X graduating (X)
4. $\forall$ X $\neg$ smiling (X)

Example (cont'd)

1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
2. $\forall X \neg$ happy (X) $\vee$ smiling (X)
3. $\exists$ X graduating (X)
4. $\forall$ X $\neg$ smiling (X)

Step 3. Standardize variables apart

1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
2. $\forall Y \neg$ happy (Y) $\vee$ smiling (Y)
3. $\exists$ Z graduating (Z)
4. $\forall$ W $\neg$ smiling (W)

Example (cont'd)

1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
2. $\forall Y \neg$ happy (Y) $\vee$ smiling (Y)
3. $\exists$ Z graduating (Z)
4. $\forall$ W $\neg$ smiling (W)

Step 4. Move all quantifiers to the left

Step 5. Eliminate $\exists$

1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
2. $\forall Y \neg$ happy (Y) $\vee$ smiling (Y)
3. graduating (no-name1)
4. $\forall$ W $\neg$ smiling (W)

Example (cont'd)

1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
2. $\forall Y \neg$ happy (Y) $\vee$ smiling (Y)
3. graduating (no-name1)
4. $\forall$ W $\neg$ smiling (W)

Step 6. Drop all $\forall$

1. $\neg$ graduating (X) $\vee$ happy (X)
2. $\neg$ happy (Y) $\vee$ smiling (Y)
3. graduating (no-name1)
4. $\neg$ smiling (W)

Step 7. Convert to conjunct of disjuncts form

Step 8. Make each conjunct a separate clause.

Step 9. Standardize variables apart again.

Example (cont'd)

4. $\neg$ smiling (W)

2. $\neg$ happy (Y) $\vee$ smiling (Y)

{W/Y}

5. $\neg$ happy (W)

1. $\neg$ graduating (X) $\vee$ happy (X)

{W/X}

6. $\neg$ graduating (W)

{no-name1/W}

3. graduating (no-name1)

7.

Proving a universally quantified sentence

- All people who are graduating are happy.
  All happy people smile.
  Everybody is graduating.

  Is everybody smiling?

- First convert to predicate logic
  $\forall$X graduating(X) $\rightarrow$ happy(X)
  $\forall$X happy(X) $\rightarrow$ smiling(X)
  $\forall$X graduating (X)

  $\forall$X smiling(X)   negate this: $\neg$ $\forall$ X smiling(X)

- Then convert to canonical form

Example

1. $\forall$X graduating(X) $\rightarrow$ happy(X)
2. $\forall$X happy(X) $\rightarrow$ smiling(X)
3. $\forall$X graduating (X)
4. $\neg$ $\forall$ X smiling (X)

Then convert to canonical form:

Step 1. Eliminate $\rightarrow$

1. $\forall$X $\neg$ graduating (X) $\vee$ happy (X)
2. $\forall$X $\neg$ happy (X) $\vee$ smiling (X)
3. $\forall$X graduating (X)
4. $\neg$ $\forall$ X smiling (X)

Example (cont'd)

1. $\forall$X $\neg$ graduating (X) $\vee$ happy (X)
2. $\forall$X $\neg$ happy (X) $\vee$ smiling (X)
3. $\forall$X graduating (X)
4. $\neg$ $\forall$ X smiling(X)

Step 2. Reduce the scope of negation.

1. $\forall$X $\neg$ graduating (X) $\vee$ happy (X)
2. $\forall$X $\neg$ happy (X) $\vee$ smiling (X)
3. $\forall$ X graduating (X)
4. $\exists$ X $\neg$ smiling (X)

Example (cont'd)

1. ∀X ¬ graduating (X) ∨ happy (X)
2. ∀X ¬ happy (X) ∨ smiling (X)
3. ∀ X graduating (X)
4. ∃ X ¬ smiling (X)

Step 3. Standardize variables apart

1. ∀X ¬ graduating (X) ∨ happy (X)
2. ∀Y ¬ happy (Y) ∨ smiling (Y)
3. ∀ Z graduating (Z)
4. ∃ W ¬ smiling (W)

Example (cont'd)

1. ∀X ¬ graduating (X) ∨ happy (X)
2. ∀Y ¬ happy (Y) ∨ smiling (Y)
3. ∀ Z graduating (Z)
4. ∃ W ¬ smiling (W)

Step 4. Move all quantifiers to the left

Step 5. Eliminate ∃

1. ∀X ¬ graduating (X) ∨ happy (X)
2. ∀Y ¬ happy (Y) ∨ smiling (Y)
3. ∀ Z graduating (Z)
4. ¬ smiling (no-name1)

Example (cont'd)

1. ∀X ¬ graduating (X) ∨ happy (X)
2. ∀Y ¬ happy (Y) ∨ smiling (Y)
3. ∀ Z graduating (Z)
4. ¬ smiling (no-name1)

Step 6. Drop all ∀

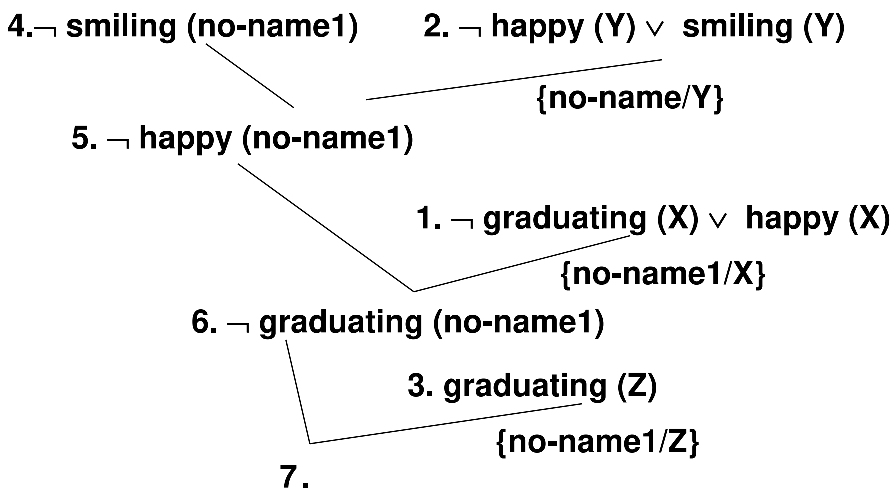
1. ¬ graduating (X) ∨ happy (X)
2. ¬ happy (Y) ∨ smiling (Y)
3. graduating (Z)
4. ¬ smiling (no-name1)

Step 7. Convert to conjunct of disjuncts form

Step 8. Make each conjunct a separate clause.

Step 9. Standardize variables apart again.

Example (cont'd)



Exercise

All people who are graduating are happy.
All happy people smile.

Prove that all people who are graduating smile.

More on Skolemization (cont'd)

- **If the existentially quantified variable is in the scope of universally quantified variables, then the existentially quantified variable must be a function of those other variables. We introduce a new, unique function called *Skolem function*.**

$\forall$X $\exists$Y (loves (X,Y)) may be replaced with any of the following:

$\forall$X loves (X, no-name(X))
$\forall$X loves (X, loved-one(X))
$\forall$X loves (X, k1(X))

no-name, loved-one, k1 are Skolem functions. They should not appear in any other sentence in the KB. They should also not have any other parameter than X.

Resolution refutation algorithm

Resolution-refutation (KB, α)

KB $\leftarrow$ KB U { $\neg\,\alpha$ }

repeat until the null clause is derived

find two sentences to resolve (should have opposite terms under the mgu)

KB $\leftarrow$ KB U { the result of resolution }

Example

- **All people who are graduating are happy.**
  All happy people smile.
  John-doe is graduating.

  Who is smiling?
- **First convert to predicate logic**
  $\forall$X graduating(X) $\rightarrow$ happy(X)
  $\forall$X happy(X) $\rightarrow$ smiling(X)
  graduating (john-doe)

  $\exists$X smiling(X)   negate this: $\neg\,\exists$X smiling(X)
- **Then convert to canonical form**

Example (cont'd)

1. $\forall X$ graduating(X) $\rightarrow$ happy(X)
2. $\forall X$ happy(X) $\rightarrow$ smiling(X)
3. graduating (john-doe)
4. $\neg \exists X$ smiling(X)

Then convert to canonical form:

Step 1. Eliminate $\rightarrow$

1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
2. $\forall X \neg$ happy (X) $\vee$ smiling (X)
3. graduating (john-doe)
4. $\neg \exists X$ smiling(X)

Example (cont'd)

1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
2. $\forall X \neg$ happy (X) $\vee$ smiling (X)
3. graduating (john-doe)
4. $\neg \exists X$ smiling(X)

Step 2. Reduce the scope of $\neg$

1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
2. $\forall X \neg$ happy (X) $\vee$ smiling (X)
3. graduating (john-doe)
4. $\forall X \neg$ smiling(X)

Example (cont'd)

1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
2. $\forall X \neg$ happy (X) $\vee$ smiling (X)
3. graduating (john-doe)
4. $\forall X \neg$ smiling(X)

Step 3. Standardize variables apart

1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
2. $\forall Y \neg$ happy (Y) $\vee$ smiling (Y)
3. graduating (john-doe)
4. $\forall Z \neg$ smiling (Z)

Example (cont'd)

1. $\forall X \neg$ graduating (X) $\vee$ happy (X)
2. $\forall Y \neg$ happy (Y) $\vee$ smiling (Y)
3. graduating (john-doe)
4. $\forall Z \neg$ smiling (Z)

Step 4. Move all quantifiers to the left

Step 5. Eliminate $\exists$

Step 6. Drop all $\forall$

1. $\neg$ graduating (X) $\vee$ happy (X)
2. $\neg$ happy (Y) $\vee$ smiling (Y)
3. graduating (john-doe)
4. $\neg$ smiling (Z)

Ready for resolution.

Example (cont'd)

4. ¬ smiling (Z)

2. ¬ happy (Y) ∨ smiling (Y)

{Z/Y}

5. ¬ happy (Z)

1. ¬ graduating (X) ∨ happy (X)

{Z/X}

6. ¬ graduating (Z)

{john-doe/Z}

3. graduating (john-doe)

7.

The substitution for Z is the answer.
John-doe is smiling!

Question

Which one of the following is a "search problem?"

1) conversion into canonical form

2) proof by resolution

Answer:

a. only 1

b. only 2

c. both 1 and 2

d. none

Strategies/heuristics for searching for resolution proofs

- ***breadth-first search:*** **each clause in the clause space is compared for resolution with every clause in the clause space in the first round. The clauses at the second level of the search space are generated by resolving the clauses at the first level with all the original clauses. For level n: resolve level (n -1) with the original set and all clauses previously produced.**

- ***the set of support*****: For a set of input clauses, S, we can specify a subset, T of S, called the set of support. The strategy requires that one of the resolvents in each resolution have an ancestor in the set of support.**

Strategies/heuristics for searching for resolution proofs (cont'd)

- ***unit preference:*** **each clauses with one literal or as few literals as possible (why?)**
Note that unit resolution where one of the resolvents is required to be a unit clause is not complete

- ***the linear input form*****: Take the negated goal and resolve with one of the original clauses. Take the result and resolve with another axiom. Continue taking the result and resolving with another axiom until the null clause is produced.**

Strategies/heuristics for searching for resolution proofs (cont'd)

Example:

Choose one as the negation of the goal

$\neg a \vee \neg b$

$a \vee \neg b$

$\neg a \vee b$

$a \vee b$

Both unit resolution and linear input form would fail to produce a proof.