

Lecture 21: Propositional Resolution

- Last time
 - Satisfiability as a search problem; Conjunctive Normal Form; DPLL algorithm

- Today
 - Propositional resolution
 - Characterisation
 - Algorithm
 - Automated reasoning
 - Recap of first-order logic

- Learning outcomes covered today:

Distinguish the characteristics, and advantages and disadvantages, of the major knowledge representation paradigms that have been used in AI, such as production rules, semantic networks, propositional logic and first-order logic;

Solve simple knowledge-based problems using the AI representations studied;

1

2

Resolution

- Computer methods are needed to deal with huge knowledge bases
- Enumeration of models is not feasible in propositional logic
- Natural deduction contains too many rules; hard to implement search
- Resolution is a proof method for classical propositional and first-order logic; requires formulae to be in CNF
- Given a formula φ resolution will decide whether the formula is *unsatisfiable or not*
- Resolution was suggested by John Robinson in the 1960s and he claimed it to be *machine oriented* as it had only one rule of inference

3

Validity, Satisfiability and Entailment

- Implications for Knowledge Representation
- *Deduction Theorem*:
 $KB \models \alpha$ if and only if $(KB \Rightarrow \alpha)$ is valid
- Or, . . .
 $KB \models \alpha$ if and only if $(KB \wedge \neg\alpha)$ is unsatisfiable
reductio ad absurdum
- For propositional, predicate and many other logics

4

Resolution

The method involves:

- Translation to a normal form (CNF)
- At each step, a new clause is derived from two clauses you already have
- Proof steps all use the same rule
 - resolution rule
- Repeat until false is derived (i.e. the formula contains a literal **and** its negation) or no new formulae can be derived
- We first introduce the method for propositional logic and then (next lecture) extend it to first-order predicate logic

5

Resolution Rule

- Each A_i is known as a *clause* and we consider the set of clauses $\{A_1, A_2 \dots A_k\}$

- The (propositional) resolution rule is as follows

$$\begin{array}{r} A \vee p \\ B \vee \neg p \\ \hline A \vee B \end{array}$$

- $A \vee B$ is called the *resolvent*
- $A \vee p$ and $B \vee \neg p$ are called *parents of the resolvent*
- p and $\neg p$ are called *complementary literals*
- Note in the above A or B can be empty

6

Resolution applied to sets of clauses

- Show by resolution that the following set of clauses is unsatisfiable

$$\{p \vee q, p \vee \neg q, \neg p \vee q, \neg p \vee \neg q\}$$

1. $p \vee q$
2. $p \vee \neg q$
3. $\neg p \vee q$
4. $\neg p \vee \neg q$
5. p [1, 2]

7

Resolution applied to sets of clauses

- Show by resolution that the following set of clauses is unsatisfiable

$$\{p \vee q, p \vee \neg q, \neg p \vee q, \neg p \vee \neg q\}$$

1. $p \vee q$
2. $p \vee \neg q$
3. $\neg p \vee q$
4. $\neg p \vee \neg q$
5. p [1, 2]
6. $\neg p$ [3, 4]

8

Resolution applied to sets of clauses

- Show by resolution that the following set of clauses is unsatisfiable

$$\{p \vee q, p \vee \neg q, \neg p \vee q, \neg p \vee \neg q\}$$

1. $p \vee q$
2. $p \vee \neg q$
3. $\neg p \vee q$
4. $\neg p \vee \neg q$
5. p [1, 2]
6. $\neg p$ [3, 4]
7. **false** [5, 6]

9

Exercise

- use resolution: is the following set of clauses satisfiable?

$$\{\neg p \vee q \vee r, p, \neg q, \neg r\}$$

10

Resolution Algorithm

- Proof if $KB \models \alpha$ by contradiction (i.e. show that $KB \wedge \neg\alpha$ is unsatisfiable)

```
function PL-Resolution(KB,  $\alpha$ ) returns true or false
  inputs: KB, the knowledge base, a sentence in propositional logic
          $\alpha$ , the query, a sentence in propositional logic
  clauses  $\leftarrow$  the set of clauses in the CNF representation of  $KB \wedge \neg\alpha$ 
  new  $\leftarrow \{\}$ 
  loop do
    for each pair of clauses  $C_i, C_j$  in clauses do
      resolvents  $\leftarrow$  PL-Resolve( $C_i, C_j$ )
      if resolvents contains the empty clause then return true //<- contradiction
      new  $\leftarrow$  new  $\cup$  resolvents
    if new  $\subseteq$  clauses then return false //<- no new clauses; no contradiction
  clauses  $\leftarrow$  clauses  $\cup$  new
```

12

Full Circle Example

- Using resolution show

$$((q \wedge p) \Rightarrow r) \models (\neg p \vee \neg q \vee r)$$

- show that

$$((q \wedge p) \Rightarrow r) \wedge \neg(\neg p \vee \neg q \vee r)$$

- is unsatisfiable

- Translate to CNF
- Apply the resolution algorithm

13

1) Transformation to CNF

$$((q \wedge p) \Rightarrow r) \wedge \neg(\neg p \vee \neg q \vee r)$$

14

1) Transformation to CNF

$$\begin{aligned} & ((q \wedge p) \Rightarrow r) \wedge \neg(\neg p \vee \neg q \vee r) \\ \equiv & (\neg(q \wedge p) \vee r) \wedge \neg(\neg p \vee \neg q \vee r) \end{aligned}$$

15

1) Transformation to CNF

$$\begin{aligned} & ((q \wedge p) \Rightarrow r) \wedge \neg(\neg p \vee \neg q \vee r) \\ \equiv & (\neg(q \wedge p) \vee r) \wedge \neg(\neg p \vee \neg q \vee r) \\ \equiv & ((\neg q \vee \neg p) \vee r) \wedge \neg(\neg p \vee \neg q \vee r) \end{aligned}$$

16

1) Transformation to CNF

$$\begin{aligned} & ((q \wedge p) \Rightarrow r) \wedge \neg(\neg p \vee \neg q \vee r) \\ \equiv & (\neg(q \wedge p) \vee r) \wedge \neg(\neg p \vee \neg q \vee r) \\ \equiv & ((\neg q \vee \neg p) \vee r) \wedge \neg(\neg p \vee \neg q \vee r) \\ \equiv & (\neg q \vee \neg p \vee r) \wedge (\neg\neg p \wedge \neg\neg q \wedge \neg r) \end{aligned}$$

17

1) Transformation to CNF

$$\begin{aligned} & ((q \wedge p) \Rightarrow r) \wedge \neg(\neg p \vee \neg q \vee r) \\ \equiv & (\neg(q \wedge p) \vee r) \wedge \neg(\neg p \vee \neg q \vee r) \\ \equiv & ((\neg q \vee \neg p) \vee r) \wedge \neg(\neg p \vee \neg q \vee r) \\ \equiv & (\neg q \vee \neg p \vee r) \wedge (\neg\neg p \wedge \neg\neg q \wedge \neg r) \\ \equiv & (\neg q \vee \neg p \vee r) \wedge (p \wedge q \wedge \neg r) \end{aligned}$$

18

2) Resolution

1. $\neg q \vee \neg p \vee r$
2. p
3. q
4. $\neg r$

- Finally, apply the resolution rule.

1) Transformation to CNF

$$\begin{aligned} & ((q \wedge p) \Rightarrow r) \wedge \neg(\neg p \vee \neg q \vee r) \\ \equiv & (\neg(q \wedge p) \vee r) \wedge \neg(\neg p \vee \neg q \vee r) \\ \equiv & ((\neg q \vee \neg p) \vee r) \wedge \neg(\neg p \vee \neg q \vee r) \\ \equiv & (\neg q \vee \neg p \vee r) \wedge (\neg\neg p \wedge \neg\neg q \wedge \neg r) \\ \equiv & (\neg q \vee \neg p \vee r) \wedge (p \wedge q \wedge \neg r) \\ \equiv & (\neg q \vee \neg p \vee r) \wedge p \wedge q \wedge \neg r \end{aligned}$$

19

2) Resolution

1. $\neg q \vee \neg p \vee r$
2. p
3. q
4. $\neg r$

- Finally, apply the resolution rule.

$$5. \neg q \vee r \quad [1, 2]$$

20

21

2) Resolution

1. $\neg q \vee \neg p \vee r$
2. p
3. q
4. $\neg r$

- Finally, apply the resolution rule.

5. $\neg q \vee r$ [1, 2]
6. r [5, 3]

22

Theoretical Issues

- Resolution is *refutation complete*. That is, if given an unsatisfiable set of clauses the procedure is guaranteed to produce **false**
- Resolution is *sound*. That is, if we derive **false** from a set of clauses then the set of clauses is unsatisfiable
- The resolution method *terminates*. That is, we apply the resolution rule until we derive false or no new clauses can be derived, and it will always stop

24

2) Resolution

1. $\neg q \vee \neg p \vee r$
2. p
3. q
4. $\neg r$

- Finally, apply the resolution rule.

5. $\neg q \vee r$ [1, 2]
6. r [5, 3]
7. **false** [4, 6]

23

Reducing the Search Space (I)

- Although the basic resolution method is complete, it is not very efficient. This is due to the search space that has to be explored
- A lot of effort has been applied in trying to reduce the search space
 - The elimination of tautologies (e.g. clauses such as $p \vee q \vee \neg q$)
 - Subsumption (if a clause set contains the clauses p and $p \vee q$, $p \vee q$ may be discarded); removes useless or redundant rules.

25

Reducing the Search Space (II)

- Some forms of resolution restrict which clauses may be resolved together e.g. *unit resolution* (always resolve using at least one unit clause) or *set of support* (after the first step, use at most one original clause)
- Heuristics may be applied to guide the proof search e.g. *weighting strategies*
- Applying strategies such as set of support or heuristics may affect completeness

26

Automated Reasoning

- The resolution proof method may be automated, i.e. carried out by a computer program
- Theorem provers based on resolution have been developed
 - https://en.wikipedia.org/wiki/Automated_theorem_proving
- Prolog also uses resolution, but only for a subset of FOL: Horn Clauses
 - **At most one positive literal** in any clause.
 - $p :- q, r$ is equivalent to...
 - $(q \wedge r) \Rightarrow p$
 - $p \vee \neg(q \wedge r)$
 - $p \vee \neg q \vee \neg r$
 - This greatly improves efficiency, making Prolog usable as a programming language.

27

Resolution in Prolog

(1) $p :- q, r.$ i.e. $p \vee \neg q \vee \neg r$

(2) $q :- t.$ i.e. $q \vee \neg t$

(3) $r :- u.$ i.e. $r \vee \neg u$

(4) $t.$

(5) $u.$

We want to show that (1–5) entails p

First add:

(6) $\neg p.$

Now prove (1-6) is unsatisfiable...

Resolve (6) and (1) to get (7) $\neg q \vee \neg r$

Resolve (7) and (2) to get (8) $\neg t \vee \neg r$

Resolve (4) and (8) to get (9) $\neg r$

Resolve (9) and (3) to get (10) $\neg u$

Resolve (10) and (5) to get empty clause.

$\neg p$ is unsatisfiable and hence p is true.

28

Pros and Cons of Propositional Logic

Benefits over most programming languages, data structures and databases:

- Propositional logic is *declarative*
 - separates knowledge and inference
- Propositional logic allows partial/disjunctive/negated information
 - allows to specify uncertainty about complex cases
- Propositional logic is *compositional*
 - Meaning of $q \wedge r$ depends (only!) on meaning of q and r
 - i.e., it is *context-independent* (unlike natural language, where meaning depends on context)

But...

- Propositional logic has very limited expressive power (unlike natural language)

29

Example

- Consider

$$\frac{\begin{array}{l} \text{Kitty is a cat} \\ \text{cats are mammals} \end{array}}{\text{Kitty is a mammal}}$$

- In propositional logic this would be represented as

$$\frac{c, m}{k}$$

- This derivation is not valid in propositional logic. If it were then from any c and m could derive any k .
 - We need to capture the connection between c and m .
- To do this, we will use *first-order (or predicate) logic*.

30

Recap of First-Order Logic

- Whereas propositional logic assumes the world contains *facts*, first-order logic (like natural language) assumes the world contains
 - **Objects** (people, houses, numbers, colours...);
 - **Relations** (part of, after, prime, brother of, . . .);
 - **Functions** (best friend, one more than, end of . . .)
- Examples:


```
course_lecturer(Frans, COMP219)
male(Frans)
< (3, 4)
< (4, plustwo(1))
mammal(Kitty)
```

 - Frans, Kitty, COMP219, 3, 4 and 1 are **constants**.
 - course_lecturer, male, mammal, and < are **predicates**.
 - male, mammal have **arity** one and the other predicates have arity two.
 - Plustwo is a **function** (that refers to other objects),
 - e.g. plustwo(1) refers to the constant 3

31

Quantifiers

- Quantifiers allow us to express properties about collections of objects
- The **quantifiers** are
 - \forall universal quantifier 'For all . . .'
 - \exists existential quantifier 'There exists . . .'
- If $P(x)$ is a predicate then we can write
 - $\forall x \cdot P(x)$; and
 - $\exists x \cdot P(x)$;
 where x is a *variable* which can stand for any object in the domain

32

Interpretations

- We need a domain to which we are referring.


```
course_lecturer(Frans, COMP219)
```
- The name Frans is mapped to the object in the domain we are referring to (me)
- The name COMP219 is mapped to the object in the domain we are referring to (the course COMP219)
- The predicate name course_lecturer will be mapped to a set of pairs of objects where the first in the pair is the (real) person who teaches the second in the pair
- Hence the above evaluates to true

33

Syntax of Predicate Logic

- The formulas of predicate logic are constructed from the following symbols
 - a set PRED of predicate symbols with arity;
 - a set FUNC of function symbols with arity;
 - a set CONS of constant symbols;
 - a set VAR of variable symbols;
 - the quantifiers \forall and \exists ;
 - **true**, **false** and the connectives $\wedge, \vee, \Rightarrow, \neg, \Leftrightarrow, =$.
- Note propositions can be viewed as predicates with arity 0

34

Terms

- The set of terms, TERM, is constructed by the following rules
 - any constant is in TERM;
 - any variable is in TERM;
 - if t_1, \dots, t_n are in TERM and f is a function symbol of arity n then $f(t_1, \dots, t_n)$ is a term.
 - $f(x, y)$
 - $\text{add}(2, 4)$
 - $\text{mother_of}(\text{Katie})$

'terms' refer to objects

35

Well-Formed Formulae

- The set of sentences or *well-formed formulae* of predicate logic are:
 - **true**, **false** and propositional formulae are in WFF.
 - if t_1, \dots, t_n are in TERM and p is a predicate symbol of arity n then $p(t_1, \dots, t_n)$ is in WFF.
 - If A and B are in WFF then so is $\neg A, A \vee B, A \wedge B, A \Rightarrow B$ and $A \Leftrightarrow B$.
 - If A is in WFF and x is a variable then $\forall x \cdot A$ and $\exists x \cdot A$ are in WFF.
 - If A is in WFF then so is (A) .

36

Exercise (Nell)

- Which of the following are well-formed formulae of first-order logic?
 - 1) $\forall \exists \cdot p(x)$
 - 2) $\exists x \cdot p(\neg x)$
 - 3) $\forall x \cdot p(x) \wedge \exists y \cdot r(y)$

37

Domains and Interpretation

- Suppose we have a formula $\forall x \cdot P(x)$. What does x range over? *Physical objects, numbers, people, times, ...* ?
- Depends on the **domain** that we intend...
- In each domain we have an **interpretation** that specifies which objects correspond to which constants, etc.
- Often, we “name” a domain to make our intended interpretation clear. E.g.,
 - Suppose our intended interpretation is the “positive integers” and that $>, +, \times, \dots$ have “the usual mathematical interpretation”.
 - Is this formula *satisfiable* under the above interpretation?
 $\exists n \cdot n = (n \times n)$

38

Summary

- We have described how to apply the proof method *resolution* in propositional logic
 - First, formulae need to be in conjunctive normal form
 - There is only one rule of inference
- We have had a brief recap of first-order logic
 - We have looked at its syntax but we haven’t seen its formal semantics (see good AI and logic books)
 - Informally we’ve seen we need a domain of interest; constants, predicates, functions have mappings into this domain (‘the interpretation’)
 - To evaluate quantifiers we must check whether all objects in the domain satisfy the formula (\forall) or some object does (\exists)
- **Next time**
 - We will look at extending resolution to FOL

39