UNIVERSITÉ CÔTE D'AZUR

Logic for AI

Master 1 Informatique

Andrea G. B. Tettamanzi
Laboratoire I3S – Pôle SPARKS
andrea.tettamanzi@univ-cotedazur.fr

Unit 2

Propositional Logic

Agenda

- Propositional Logic
  - Logical Entailment
  - Canonical Representation
  - Davis-Putnam Algorithm
  - Resolution
  - Formal Systems, Deduction, and Proof

Logical Entailment

- A set of premises Δ logically entails a conclusion φ, written

$$\Delta \models \phi$$

  if and only if every interpretation that satisfies the premises also satisfies the conclusion.
- Examples:

$$\{p\} \models p \vee q$$

$$\{p\} \not\models p \wedge q$$

$$\{p, q\} \models p \wedge q$$

Logical Entailment ≠ Logical Equivalence!

Truth Table Method

- Method for computing whether a set of premises logically entails a conclusion
  1) Form a truth table for the propositional constants occurring in the premises and conclusion; add a column for the premises and a column for the conclusion
  2) Evaluate the premises for each row in the table
  3) Evaluate the conclusion for each row in the table
  4) If every row that satisfies the premises also satisfies the conclusion, then the premises logically entail the conclusion

Logical Entailment and Satisfiability

- **Unsatisfiability Theorem**: $\Delta \models \phi$ if and only if $\Delta \cup \{\neg\phi\}$ is unsatisfiable.
- Proof:
  - $[\Rightarrow]$: Suppose that Δ $|=$ ϕ. If an interpretation satisfies **Δ,** then it must also satisfy ϕ. But then it cannot satisfy $\neg\phi$. Therefore, Δ $\cup$ $\{\neg\phi\}$ is unsatisfiable.
  - $[\Leftarrow]$: Suppose that Δ $\cup$ $\{\neg\phi\}$ is unsatisfiable. Then every interpretation that satisfies Δ must fail to satisfy $\neg\phi$, i.e., it must satisfy ϕ. Therefore, Δ $|=$ ϕ.
- Corollary: we can determine logical entailment by determining satisfiability (proof by refutation).

Satisfaction

- Method to find all propositional interpretations that satisfy a given set of sentences:
  1) Form a truth table for the propositional constants.
  2) For each sentence in the set and each row in the truth table, check whether the row satisfies the sentence. If not, cross out the row.
  3) Any row remaining satisfies all sentences in the set. (Note that there might be more than one.)

Canonical Representation

- Syntactically distinct sentences can be equivalent (i.e., semantically identical)
- Sometimes, that can be impractical
- Idea: why don't we reduce all sentences to a canonical form, so that checking them for equivalence becomes trivial?
- Conjunctive and Disjunctive Normal Form (resp. CNF and DNF)

Conjunctive Normal Form (CNF)

- A literal is a positive or negated constant, like p or ¬p
- A clause is the disjunction of a finite number of literals, i.e., a sentence of the form

$$(l_1 \vee l_2 \vee \ldots \vee l_n)$$

- A clause is valid if and only if it contains a pair of opposed literals, like p and ¬p.
- The empty clause F is the only unsatisfiable clause.
- A CNF is the conjunction of a finite number of clauses, i.e., a sentence of the form

$$(c_1 \wedge c_2 \wedge \ldots \wedge c_n)$$

Conjunctive Normal Form

- **Theorem**: for every propositional sentence, there exists an equivalent CNF
- Proof: we give an algorithm to transform any sentence into CNF

1) Eliminate the ⇔ and ⇒ operators:

$$(\phi \Leftrightarrow \psi) \rightarrow (\phi \Rightarrow \psi) \wedge (\psi \Rightarrow \phi) \qquad (\phi \Rightarrow \psi) \rightarrow (\neg \phi \vee \psi)$$

2) Apply as many times as possible the following rewrite rules:

$$\neg(\phi \vee \psi) \rightarrow (\neg \phi \wedge \neg \psi)$$
$$\neg(\phi \wedge \psi) \rightarrow (\neg \phi \vee \neg \psi)$$
$$\neg\neg\phi \rightarrow \phi$$

3) Apply as many times as possible the following rewrite rules:

$$\phi \vee (\psi \wedge \xi) \rightarrow (\phi \vee \psi) \wedge (\phi \vee \xi)$$
$$(\phi \wedge \psi) \vee \xi \rightarrow (\phi \vee \xi) \wedge (\psi \vee \xi)$$

The resulting CNF is equivalent to the initial sentence.

Conjunctive Normal Form

- A few details complete the algorithm of the previous slide:
  - Valid clauses can be deleted as soon as they appear
  - Repeated literals in the same clause can be simplified
  - If a clause *c* is included in another clause *c'* (*c* subsumes *c'*), then clause *c'* can be deleted
  - A CNF including an empty clause can be reduced to just the empty clause F.

- The CNF thus obtained is said to be “pure”.
- The algorithm always terminates after a finite number of steps and returns a CNF

Davis-Putnam Algorithm

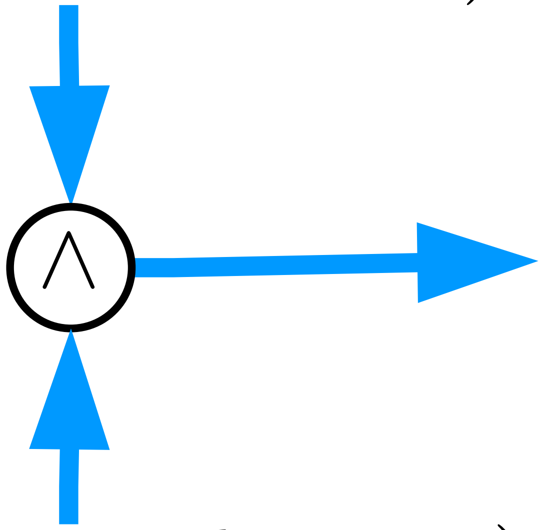
- DP(S : pure CNF) : Boolean // Test whether S is satisfiable
  1) If S = Ø, then return T; If S = {F}, then return F; Otherwise
  2) Select a propositional constant p in S, giving priority to those such that (a) p or $\neg p$ occurs alone in a clause or (b) only p or $\neg p$ occurs in S
  3) Let S_p be the set of clause containing p, $S_{\neg p}$ those not containing p, and S'' the remaining clauses
  4) $S'_p \leftarrow S_p$ where p is set to F (thus, deleted from each clause)
  5) $S'_{\neg p} \leftarrow S_{\neg p}$ where p is set to T (thus $\neg p$ is deleted)
  6) Return DP($S'_p \cup S''$) $\vee$ DP($S'_{\neg p} \cup S''$).

Deduction (Proofs)

- Deduction:
  - Symbolic manipulation of sentences, rather than enumeration of interpretations (= truth assignments)
- Benefits:
  - Usually smaller than truth tables
  - Can be often found with less work

Resolution Principle

$$(l_1 \vee l_2 \vee \ldots \vee l_n \vee p)$$



$$(l_1 \vee l_2 \vee \ldots \vee l_n)$$

Resolvent clause

$$(l_1 \vee l_2 \vee \ldots \vee l_n \vee \neg p)$$

Clausal Resolution

- To check whether a CNF S is satisfiable:
  1) Find two clauses in S, one containing literal *l* and the other containing ¬*l*, such that they have not yet been used together (if they cannot be found, terminate with result: “satisfiable”)
  2) Compute their resolvent (if it is the empty clause F, terminate with result: “unsatisfiable”)
  3) Add the resolvent to S
  4) Go back to Step 1.

- We can use resulution to construct proofs by refutation: to prove that S |= ϕ, prove that S$\cup \{\neg\phi\}$ is unsatisfiable.

Example

$$S = \{p \overset{1}{\vee} q, p \overset{2}{\vee} r, \neg q \overset{3}{\vee} \neg r, \overset{4}{\neg p}\}$$

#	clause	from
5	$p \vee \neg r$	(1, 3)
6	q	(1, 4)
7	$p \vee \neg q$	(2, 3)
8	r	(2, 4)
9	p	(2, 5)
10	$\neg r$	(3, 6)
11	$\neg q$	(3, 8)
12	$\neg r$	(4, 5)
13	$\neg q$	(4, 7)
14	F	(4, 9)

Thank you for your attention