

COMP210: Artificial Intelligence

Lecture 20. Efficient propositional reasoning

Trevor Bench-Capon

<http://www.csc.liv.ac.uk/~tbc/COMP210/>

Trevor Bench-Capon

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning—4.131

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*

Trevor Bench-Capon

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning—4.531

Overview

- Entailment, satisfiability, and validity.
- Conjunctive normal form.
- Satisfiability as a search problem.
- Efficient Heuristic Solvers

Trevor Bench-Capon

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning—4.201

Satisfiability Checking

- Given a knowledge base KB and a property α , check if $(KB \wedge \neg\alpha)$ is satisfiable
 - If not satisfiable, α is implied by KB
 - Otherwise, an interpretation of propositions would give us a *countermodel*
- Compare Prolog. Prolog attempts to satisfy the negation of the goal: the solutions returned are models of the goal (countermodels to the negation of the goal).

Trevor Bench-Capon

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning—4.531

Propositional Logic for KR

Given a knowledge base KB and a property α , check if $KB \models \alpha$

- Use truth tables
- Prove α from KB
- Relate with *validity* and *satisfiability*
- Davis-Putnam algorithm

Trevor Bench-Capon

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning—4.331

Countermodel (1)

To check if

$$\left. \begin{array}{l} (\text{hot} \wedge \text{smoky} \Rightarrow \text{fire}) \\ \wedge (\text{alarm_beeps} \Rightarrow \text{smoky}) \\ \wedge (\text{fire} \Rightarrow \text{switch_on_sprinklers}) \end{array} \right\} \not\models (\text{alarm_beeps} \Rightarrow \text{switch_on_sprinklers})$$

we form

$$\begin{array}{l} (\text{hot} \wedge \text{smoky} \Rightarrow \text{fire}) \\ \wedge (\text{alarm_beeps} \Rightarrow \text{smoky}) \\ \wedge (\text{fire} \Rightarrow \text{switch_on_sprinklers}) \\ \wedge \neg (\text{alarm_beeps} \Rightarrow \text{switch_on_sprinklers}) \end{array}$$

Trevor Bench-Capon

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning—4.731

Validity and Satisfiability

- A formula is said to be *valid* (or a *tautology*) iff it is true under *every* interpretation.
- A formula is said to be *satisfiable* (or *consistent*) iff it is true under *at least one* interpretation.
- A formula is said to be *unsatisfiable* (or *inconsistent* or *contradictory*) iff it is not made true under *any* interpretation.
- If a formula φ is a valid then $\neg\varphi$ is unsatisfiable.

Trevor Bench-Capon

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning—4.431

Countermodel (2)

But

$$\begin{array}{l} (\text{hot} \wedge \text{smoky} \Rightarrow \text{fire}) \\ \wedge (\text{alarm_beeps} \Rightarrow \text{smoky}) \\ \wedge (\text{fire} \Rightarrow \text{switch_on_sprinklers}) \\ \wedge \neg (\text{alarm_beeps} \Rightarrow \text{switch_on_sprinklers}) \end{array}$$

is true under the interpretation I :

$$\begin{array}{l} I(\text{hot}) = F \\ I(\text{smoky}) = T \\ I(\text{fire}) = F \\ I(\text{alarm_beeps}) = T \\ I(\text{switch_on_sprinklers}) = F \end{array}$$

Trevor Bench-Capon

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning—4.831

Example: Truth Tables and Satisfiability

Using a truth table show whether

$$(p \Rightarrow q) \vee (q \Rightarrow p)$$

is a tautology, consistent or inconsistent.

| p | q | $(p \Rightarrow q)$ | $(q \Rightarrow p)$ | $(p \Rightarrow q) \vee (q \Rightarrow p)$ |
|-----|-----|---------------------|---------------------|--------------------------------------------|
| T | T | | | |

Teodor Banch-Capron

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning – 0.931

Example: Truth Tables and Satisfiability

Using a truth table show whether

$$(p \Rightarrow q) \vee (q \Rightarrow p)$$

is a tautology, consistent or inconsistent.

| p | q | $(p \Rightarrow q)$ | $(q \Rightarrow p)$ | $(p \Rightarrow q) \vee (q \Rightarrow p)$ |
|-----|-----|---------------------|---------------------|--------------------------------------------|
| T | T | T | T | T |
| T | F | F | T | T |
| F | T | | | |

Teodor Banch-Capron

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning – 0.931

Example: Truth Tables and Satisfiability

Using a truth table show whether

$$(p \Rightarrow q) \vee (q \Rightarrow p)$$

is a tautology, consistent or inconsistent.

| p | q | $(p \Rightarrow q)$ | $(q \Rightarrow p)$ | $(p \Rightarrow q) \vee (q \Rightarrow p)$ |
|-----|-----|---------------------|---------------------|--------------------------------------------|
| T | T | T | | |

Teodor Banch-Capron

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning – 0.931

Example: Truth Tables and Satisfiability

Using a truth table show whether

$$(p \Rightarrow q) \vee (q \Rightarrow p)$$

is a tautology, consistent or inconsistent.

| p | q | $(p \Rightarrow q)$ | $(q \Rightarrow p)$ | $(p \Rightarrow q) \vee (q \Rightarrow p)$ |
|-----|-----|---------------------|---------------------|--------------------------------------------|
| T | T | T | T | T |
| T | F | F | T | T |
| F | T | T | F | T |
| F | F | | | |

Teodor Banch-Capron

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning – 0.931

Example: Truth Tables and Satisfiability

Using a truth table show whether

$$(p \Rightarrow q) \vee (q \Rightarrow p)$$

is a tautology, consistent or inconsistent.

| p | q | $(p \Rightarrow q)$ | $(q \Rightarrow p)$ | $(p \Rightarrow q) \vee (q \Rightarrow p)$ |
|-----|-----|---------------------|---------------------|--------------------------------------------|
| T | T | T | T | |

Teodor Banch-Capron

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning – 0.931

Example: Truth Tables and Satisfiability

Using a truth table show whether

$$(p \Rightarrow q) \vee (q \Rightarrow p)$$

is a tautology, consistent or inconsistent.

| p | q | $(p \Rightarrow q)$ | $(q \Rightarrow p)$ | $(p \Rightarrow q) \vee (q \Rightarrow p)$ |
|-----|-----|---------------------|---------------------|--------------------------------------------|
| T | T | T | T | T |
| T | F | F | T | T |
| F | T | T | F | T |
| F | F | T | T | T |

Teodor Banch-Capron

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning – 0.931

Example: Truth Tables and Satisfiability

Using a truth table show whether

$$(p \Rightarrow q) \vee (q \Rightarrow p)$$

is a tautology, consistent or inconsistent.

| p | q | $(p \Rightarrow q)$ | $(q \Rightarrow p)$ | $(p \Rightarrow q) \vee (q \Rightarrow p)$ |
|-----|-----|---------------------|---------------------|--------------------------------------------|
| T | T | T | T | T |
| T | F | | | |

Teodor Banch-Capron

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning – 0.931

Efficiency

- A truth table contains 2^n rows
- Its construction requires 2^n steps...

Can we do better than that?

- Not really: Theory says that this is a very hard problem
- In practice, not so bad, if we can use heuristics to identify lines we don't need to check
- But we have to transform $(KB \wedge \neg\alpha)$ into a normal form

Teodor Banch-Capron

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning – 0.1031

Equivalent Formulae

- Two formulae A and B are equivalent, written $A \equiv B$ iff A and B have the same truth values for every interpretation.
- Show

$$(p \Rightarrow q) \equiv (\neg p \vee q).$$

Draw up a truth table for $(p \Rightarrow q)$ and $(\neg p \vee q)$ and check their truth values are the same.

| p | q | $p \Rightarrow q$ | $\neg p$ | $\neg p \vee q$ |
|-----|-----|-------------------|----------|-----------------|
| T | T | T | F | T |

Teodor Banch-Capron

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning – 0, 1131

Equivalent Transformations III

- Identity laws

$$\begin{aligned} A \wedge T &\equiv A \\ A \vee F &\equiv A \\ A \wedge F &\equiv F \\ A \vee T &\equiv T \end{aligned}$$

- Complement laws

$$\begin{aligned} A \wedge \neg A &\equiv F \\ A \vee \neg A &\equiv T \\ \neg(\neg A) &\equiv A \\ \neg T &\equiv F \\ \neg F &\equiv T \end{aligned}$$

Teodor Banch-Capron

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning – 0, 1431

Equivalent Formulae

- Two formulae A and B are equivalent, written $A \equiv B$ iff A and B have the same truth values for every interpretation.
- Show

$$(p \Rightarrow q) \equiv (\neg p \vee q).$$

Draw up a truth table for $(p \Rightarrow q)$ and $(\neg p \vee q)$ and check their truth values are the same.

| p | q | $p \Rightarrow q$ | $\neg p$ | $\neg p \vee q$ |
|-----|-----|-------------------|----------|-----------------|
| T | T | T | F | T |
| T | F | F | F | F |
| F | T | T | T | T |
| F | F | T | T | T |

Teodor Banch-Capron

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning – 0, 1131

Equivalent Transformations IV

- de Morgan's laws

$$\begin{aligned} \neg(A \wedge B) &\equiv \neg A \vee \neg B \\ \neg(A \vee B) &\equiv \neg A \wedge \neg B \end{aligned}$$

- laws for \Rightarrow and \Leftrightarrow

$$\begin{aligned} A \Rightarrow B &\equiv \neg A \vee B \\ A \Leftrightarrow B &\equiv (A \Rightarrow B) \wedge (B \Rightarrow A) \end{aligned}$$

We can use these laws to simplify expressions and to prove equivalences.

Teodor Banch-Capron

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning – 0, 1531

Equivalent Transformations I

Where A , B and C are propositions or propositional formulae and T and F are true and false respectively.

- Idempotent laws

$$\begin{aligned} A \wedge A &\equiv A \\ A \vee A &\equiv A \end{aligned}$$

- Associative laws

$$\begin{aligned} (A \wedge B) \wedge C &\equiv A \wedge (B \wedge C) \\ (A \vee B) \vee C &\equiv A \vee (B \vee C) \end{aligned}$$

Teodor Banch-Capron

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning – 0, 1231

Example

Simplify the following expression.

$$\neg(\neg P \wedge \neg Q)$$

$$\begin{aligned} &\neg(\neg P \wedge \neg Q) && \text{given} \\ \equiv &(\neg\neg P \vee \neg\neg Q) && \text{de Morgan's laws} \\ \equiv &(P \vee Q) && \text{Complement laws} \end{aligned}$$

Teodor Banch-Capron

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning – 0, 1631

Equivalent Transformations II

- Commutative laws

$$\begin{aligned} A \wedge B &\equiv B \wedge A \\ A \vee B &\equiv B \vee A \end{aligned}$$

- Distributive laws

$$\begin{aligned} A \wedge (B \vee C) &\equiv (A \wedge B) \vee (A \wedge C) \\ A \vee (B \wedge C) &\equiv (A \vee B) \wedge (A \vee C) \end{aligned}$$

Teodor Banch-Capron

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning – 0, 1331

Example

Prove the following equivalence

$$\neg(\neg(P \wedge Q) \vee P) \equiv F$$

$$\begin{aligned} &\neg(\neg(P \wedge Q) \vee P) && \text{given} \\ \equiv &\neg((\neg P \vee \neg Q) \vee P) && \text{de Morgan's laws} \\ \equiv &\neg((\neg Q \vee \neg P) \vee P) && \text{Commutative laws} \\ \equiv &\neg(\neg Q \vee (\neg P \vee P)) && \text{Associative laws} \\ \equiv &\neg(\neg Q \vee T) && \text{Complement laws} \\ \equiv &\neg T && \text{Complement laws} \\ \equiv &F && \text{Complement laws} \end{aligned}$$

Teodor Banch-Capron

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning – 0, 1731

Normal Forms

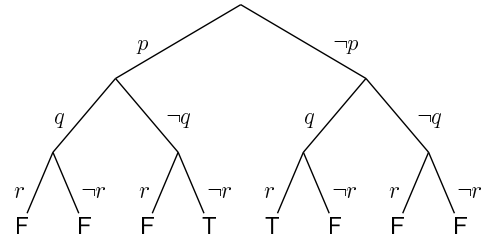
- It is often useful to transform formulae into *normal forms*. These are logically equivalent formulae but have syntactically different forms that may be more suitable for reasoning with.
- Recall two formulae A and B are equivalent, written $A \equiv B$ if and only if the truth values for A and B are the same for each interpretation.
- There are several normal forms: Negation normal form, Clausal form, Disjunctive Normal Form and Conjunctive Normal Form. We are interested in **Conjunctive Normal Form (CNF)**.

Teodor Banach-Capron

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning - d. 19/31

Search Space

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



- Every path in the tree represents a (partial) assignment

Teodor Banach-Capron

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning - d. 22/31

Conjunctive Normal Form

- A formula is in *Conjunctive Normal Form* if it is of the form

$$A_1 \wedge A_2 \wedge \dots \wedge A_k$$

where each A_i is a **disjunction** of propositions or their negations.

- Example**
 $(p \vee q) \wedge r \wedge (\neg p \vee \neg r \vee s)$ is in CNF.
 $\neg(p \vee q) \wedge r \wedge (\neg p \vee \neg r \vee s)$ is not in CNF.
 $(p \vee q) \wedge r \wedge (p \Rightarrow (\neg r \vee s))$ is not in CNF.
- Any well formed formula of propositional logic can be rewritten, using the above equivalences, as an **equivalent** formula of CNF

Teodor Banach-Capron

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning - d. 19/31

Complexity...

- ... For n propositions, the tree will have $(2^n - 1)$ nodes.
- Another idea: Simplify formula with a partial assignment
 - $(\neg p \vee \neg r) \wedge (p \vee q) \wedge (r \vee \neg q)$; let $p = True$
 - $(\neg True \vee \neg r) \wedge (True \vee q) \wedge (r \vee \neg q)$
 - $(False \vee \neg r) \wedge True \wedge (r \vee \neg q)$
 - $\neg r \wedge (r \vee \neg q)$ can only be true if $r = False$
 - $\neg False \wedge (False \vee q)$
 - q can only be true if $q = True$
 - $True$
- Given a good order for partial assignments this can be very helpful. DPLL provides heuristics for choosing partial assignments. Only in the worst case we will need to try everything.

Teodor Banach-Capron

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning - d. 23/31

Satisfiability as a Search Problem

- Given a formula in CNF, can we find an assignment of truth values to propositions which satisfies it?
- States are *partial assignments*—some propositions get values, some are possibly unassigned
- Operations are deciding whether a (yet unassigned) proposition is true or false
- Initial state: empty partial assignment
- Goal state: an assignment making the formula true

Teodor Banach-Capron

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning - d. 20/31

Likewise...

- $(\neg p \vee \neg r) \wedge (p \vee q) \wedge (r \vee \neg q)$; let $p = False$
- $(\neg False \vee \neg r) \wedge (False \vee q) \wedge (r \vee \neg q)$
- $(True \vee \neg r) \wedge q \wedge (r \vee \neg q)$
- $True \wedge q \wedge (r \vee \neg q)$
- $q \wedge (r \vee \neg q)$ can only be true if $q = True$
- $True \wedge (r \vee \neg True)$
- $(r \vee False)$
- r can only be true if $r = True$
- $True$

Teodor Banach-Capron

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning - d. 24/31

Algorithm for satisfiability for CNF

A complete backtracking algorithm (Davis-Putnam or DPLL)

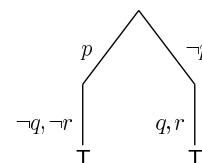
- Davis-Putnam paper (1960)
- The presented version is described in a paper by Davis, Logemann, and Loveland (1962)
- Naive Approach: every proposition is either true or false in any interpretation

Teodor Banach-Capron

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning - d. 21/31

Reduced Search Space

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



- Only 5 nodes!

Teodor Banach-Capron

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning - d. 25/31

Algorithm Structure

- Search through possible assignments of propositions
- Simplify formulae with partial assignments
 - Unit clause propagation
 - Pure literal elimination

Tevor Bench-Capron

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning – d. 26/31

Simplify(ϕ, l)

Given a propositional formula ϕ and a literal l , $\text{simplify}(\phi, l)$ does the following:

- Delete every clause that contains l from ϕ ;
- Delete $\neg l$ from all clauses in ϕ
- Apply exhaustively unit clause propagation and pure literal elimination;
- Fail (return False) if positive **and** negative unit clauses for the **same** literal

Tevor Bench-Capron

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning – d. 30/31

Unit Clause

- A clause with just one literal is called a *unit clause*
 - E.g. $\boxed{q} \wedge (\neg r \vee \neg q) \wedge (r \wedge s)$
- Literal's value can be *uniquely* assigned
 - q **must** be set to *True*
- Unit clause propagation:
Check if a formula in CNF has a unit clause, C .
Set the value of the literal in C such that C is *True*.
 - Notice that some other clauses may become unit
 - E.g. $q \wedge (\neg r \vee \neg q) \wedge (r \wedge s)$ reduces to $\boxed{\neg r} \wedge (r \wedge s)$
 - r **must** be set to *False*
 - $\neg r \wedge (r \wedge s)$ reduces to \boxed{s}

Tevor Bench-Capron

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning – d. 27/31

Applications

- There are now several efficient sat solvers based on DPLL available on the web. They are used in a number of applications.
 - Hardware and software verification
 - IBM, Intel, ...
 - Planning
 - Scheduling
 - ...

Tevor Bench-Capron

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning – d. 31/31

Pure Literal

- A *pure literal* is a literal that always appears with the same "sign" in all clauses.
 - E.g. $\boxed{p} \vee q \wedge (\neg q \vee \neg r) \wedge (\neg q \vee r)$
- Making a pure literal *True* makes *some* clauses *True*, but *no* clause *False*
 - E.g. $(p \vee q) \wedge (\neg q \vee \neg r) \wedge (\neg q \vee r)$ reduces to $(\boxed{\neg q} \vee \neg r) \wedge (\boxed{\neg q} \vee r)$
 - $(\neg q \vee \neg r) \wedge (\neg q \vee r)$ reduces to *True*
- Pure literal elimination:
Check if a formula in CNF has a pure literal, l
Set the value of l to *True*

Tevor Bench-Capron

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning – d. 28/31

DPLL(ϕ)

Given a propositional formula ϕ , $\text{DPLL}(\phi)$ does the following:

- If ϕ is *True* then return *True*;
- If ϕ is *False* then return *False*;
- Pick a proposition p in ϕ
 - If $\text{DPLL}(\text{Simplify}(\phi, p))$ is *True* then return *True*;
 - If $\text{DPLL}(\text{Simplify}(\phi, \neg p))$ is *True* then return *True*;
- Return *False*;

Tevor Bench-Capron

COMP210: Artificial Intelligence, Lecture 20, Efficient propositional reasoning – d. 29/31