Artificial Intelligence

Logic 2: Reasoning

Jendrik Seipp

Linköping University

based on slides by Thomas Keller and Malte Helmert (University of Basel)

Reasoning

Reasoning: Intuition

- often, we have a (set of) sentence(s) (a knowledge base)
- that represents our knowledge of the world
- knowledge bases usually only represent
  an incomplete description of the world
- $\rightsquigarrow$ we want to know if other sentences follow logically

What does this mean?

Reasoning: Intuition

- assume knowledge base $\Phi = \{P \vee Q, R \vee \neg P, S\}$
- Φ represents sentence $(P \vee Q) \wedge (R \vee \neg P) \wedge S$
- S holds in every interpretation where Φ is true
  What about P, Q and R?

⇝ consider all interpretations where Φ is true:

P	Q	R	S
F	**T**	**F**	**T**
F	**T**	**T**	**T**
T	**F**	**T**	**T**
T	**T**	**T**	**T**

Reasoning: Intuition

- assume knowledge base $\Phi = \{P \lor Q, R \lor \neg P, S\}$
- Φ represents sentence $(P \lor Q) \land (R \lor \neg P) \land S$
- S holds in every interpretation where Φ is true
  What about P, Q and R?

$\rightsquigarrow$ consider all interpretations where Φ is true:

P	Q	R	S
F	**T**	**F**	**T**
F	**T**	**T**	**T**
T	**F**	**T**	**T**
T	**T**	**T**	**T**

- the sentence $Q \lor R$ holds in all interpretations where Φ is true
- therefore, "$Q \lor R$ follows logically from Φ"

Reasoning: Formally

Definition (logical consequence)

Let Φ be a set of sentences. A sentence ψ follows logically from Φ (in symbols: $\Phi \models \psi$) if all models for Φ are also models are also models for ψ.

in other words: for each interpretation I,
if $I \models \varphi$ for all $\varphi \in \Phi$, then also $I \models \psi$

How can we automatically compute whether $\Phi \models \psi$?

- one possibility: build a truth table
- Are there “better” possibilities that (potentially) avoid generating the whole truth table?

Reasoning: Deduction Theorem

Proposition (deduction theorem)

Let Φ be a finite set of sentences and let ψ be a sentence. Then

$$\Phi \models \psi \quad \textit{iff} \quad \Big(\bigwedge_{\varphi \in \Phi} \varphi\Big) \to \psi \text{ is a tautology.}$$

Reasoning

consequence of deduction theorem:
reasoning can be reduced to testing validity

Algorithm

Question: Does $\Phi \models \psi$ hold?

1. test if $(\bigwedge_{\varphi \in \Phi} \varphi) \rightarrow \psi$ is tautology
2. if yes, then $\Phi \models \psi$, otherwise $\Phi \not\models \psi$

In the following: Can we test for validity "efficiently",
i.e., without computing the whole truth table?

Resolution

Sets of Clauses

for the rest of this chapter:

- we assume sentences in CNF
- clause represented as a set C of literals
- sentence represented as a set Δ of clauses

Example

Let $\varphi = (P \vee Q) \wedge \neg P$.

- φ in conjunctive normal form
- φ consists of clauses $(P \vee Q)$ and $\neg P$
- representation of φ as set of sets of literals: $\{\{P, Q\}, \{\neg P\}\}$

careful: distinguish $\square$ (empty clause) vs. $\emptyset$ (empty set of clauses)

Resolution: Idea

- consequence of deduction theorem:
  reasoning can be reduced to testing validity
- observation: sentence φ valid iff $\neg\varphi$ unsatisfiable
- testing for validity can be reduced to testing unsatisfiability

Resolution: Idea

- method to test sentence φ for unsatisfiability
- idea: derive new sentences from φ that follow logically from φ
- if empty clause $\square$ can be derived $\rightsquigarrow$ φ unsatisfiable

The Resolution Rule

$$\frac{C_1 \cup \{\ell\}, C_2 \cup \{\bar{\ell}\}}{C_1 \cup C_2}$$

- "from $C_1 \cup \{\ell\}$ and $C_2 \cup \{\bar{\ell}\}$, we can conclude $C_1 \cup C_2$"
- $C_1 \cup C_2$ is resolvent of parent clauses $C_1 \cup \{\ell\}$ and $C_2 \cup \{\bar{\ell}\}$.
- the literals ℓ and $\bar{\ell}$ are called resolution literals,
  the corresponding proposition is called resolution variable
- resolvent follows logically from parent clauses

Reasoning in the Pizzeria

Original formulas:

[each pizza has exactly one owner]

$AM \leftrightarrow \neg BM$

$AQ \leftrightarrow \neg BQ$

[each person ordered exactly one pizza]

$AM \leftrightarrow \neg AQ$

$BM \leftrightarrow \neg BQ$

Reasoning in the Pizzeria

DNF:

[each pizza has exactly one owner]

$AM \vee BM, \neg AM \vee \neg BM$

$AQ \vee BQ, \neg AQ \vee \neg BQ$

[each person ordered exactly one pizza]

$AM \vee AQ, \neg AM \vee \neg AQ$

$BM \vee BQ, \neg BM \vee \neg BQ$

Reasoning in the Pizzeria

DNF:

[each pizza has exactly one owner]

$AM \vee BM, \neg AM \vee \neg BM$

$AQ \vee BQ, \neg AQ \vee \neg BQ$

[each person ordered exactly one pizza]

$AM \vee AQ, \neg AM \vee \neg AQ$

$BM \vee BQ, \neg BM \vee \neg BQ$

Adam ordered Margherita $\rightsquigarrow$ add to KB:

AM

Resolution over knowledge base:

$\neg AM \vee \neg AQ$ with $AM \rightsquigarrow \neg AQ$

$AQ \vee BQ$ with $\neg AQ \rightsquigarrow BQ$

Waiter knows that Berta ordered the Quattro Stagioni pizza.

Example

Let $\Delta = \{\{A, \neg B, C\}, \{A, \neg C\}, \{\neg A, E\}, \{B, E\}\}$.
Does $\Delta \models E$ hold?

solution:

- test if the following is a tautology:
  $(A \vee \neg B \vee C) \wedge (A \vee \neg C) \wedge (\neg A \vee E) \wedge (B \vee E) \rightarrow E$
- equivalently: test if the following is unsatisfiable:
  $(A \vee \neg B \vee C) \wedge (A \vee \neg C) \wedge (\neg A \vee E) \wedge (B \vee E) \wedge \neg E$
- ... (resolution steps: $\rightsquigarrow$ blackboard)

Example

Let $\Delta = \{\{A, \neg B, C\}, \{A, \neg C\}, \{\neg A, E\}, \{B, E\}\}$.
Does $\Delta \models E$ hold?

solution:

- test if the following is a tautology:
  $(A \vee \neg B \vee C) \wedge (A \vee \neg C) \wedge (\neg A \vee E) \wedge (B \vee E) \rightarrow E$
- equivalently: test if the following is unsatisfiable:
  $(A \vee \neg B \vee C) \wedge (A \vee \neg C) \wedge (\neg A \vee E) \wedge (B \vee E) \wedge \neg E$
- ... (resolution steps: $\rightsquigarrow$ blackboard)
- observation: empty clause $\square$ can be derived,
  hence Δ' unsatisfiable
- consequently $\Delta \models E$

Exercise

Use the resolution method to show that $\psi = C \land \neg D$ follows logically from $\phi = \{\{A, B, C\}, \{\neg A, \neg B, D\}, \{A, \neg B, C\}, \{B, C, D\}, \{\neg D, F\}, \{E, \neg F\}, \{\neg D, \neg E\}\}$, i.e., $\phi \models \psi$.

Compare the number of required resolution steps to the size (number of rows) of a truth table that verifies the same statement.

Exercise Solution

$\phi = \{\{A, B, C\}, \{\neg A, \neg B, D\}, \{A, \neg B, C\}, \{B, C, D\}, \{\neg D, F\}, \{E, \neg F\}, \{\neg D, \neg E\}\}$
$\psi = C \land \neg D$

Testing if $\phi \models \psi$ is equivalent to testing if $\phi \rightarrow \psi$ is a tautology. We use resolution to show that ϕ and the negation of ψ is not satisfiable. Hence, first add $\neg\psi$ to ϕ, i.e., $\phi' = \phi \cup \{\neg C, D\}$.
(1) From $\{A, \neg B, C\}$ and $\{\neg A, \neg B, D\}$ we get $\{\neg B, C, D\}$,
(2) from which with $\{B, C, D\}$ we get $\{C, D\}$.
(3) From $\{\neg D, F\}$ and $\{E, \neg F\}$ we get $\{\neg D, E\}$,
(4) from which with $\{\neg D, \neg E\}\}$ we get $\{\neg D\}$,
(5) from which with $\{\neg C, D\}$ we get $\{\neg C\}$.
(6) from which with $\{C, D\}$ we get $\{D\}$,
(7) from which with $\{\neg D\}$ we get the empty clause $\square$.
Therefore, ϕ' is unsatisfiable and hence $\phi \models \psi$.

We use 7 resolution steps compared to a truth table with $2^6 = 64$ interpretations (= rows).

Resolution: Discussion

- if a sentence φ can be derived from Δ, then $\Delta \models \varphi$
  (resolution is sound)
- but $\Delta \models \varphi$ does not imply that φ can be derived from Δ
  (resolution is not complete)
- however: resolution is a complete proof method to test sentences for unsatisfiability
  (i.e., Δ is unsatisfiable iff $\square$ can be derived from Δ)
- in the worst case, resolution proofs can take exponential time
- a good strategy to determine next resolution step is needed

DPLL

Propositional Logic: Algorithmic Problems

main problems:

- reasoning ($\Theta \models \varphi$?):
  Does the sentence φ follow logically from the sentences Θ?
- equivalence ($\varphi \equiv \psi$):
  Are the sentences φ and ψ logically equivalent?
- satisfiability (SAT):
  Is sentence φ satisfiable? If yes, find a model for φ.

Propositional Logic: Algorithmic Problems

main problems:

- reasoning ($\Theta \models \varphi$?):
  Does the sentence φ follow logically from the sentences Θ?
  Is $\Theta \cup \{\neg\varphi\}$ is unsatisfiable?
- equivalence ($\varphi \equiv \psi$):
  Are the sentences φ and ψ logically equivalent?
  Are both $\varphi \land \neg\psi$ and $\psi \land \neg\varphi$ unsatisfiable?
- satisfiability (SAT):
  Is sentence φ satisfiable? If yes, find a model for φ.

The Satisfiability Problem

The Satisfiability Problem (SAT)

given:

sentence in conjunctive normal form

usually represented as pair $\langle V, \Delta \rangle$:

- V set of propositional variables (propositions)
- Δ set of clauses over V

find:

- satisfying model
- or proof that no model exists

SAT is a famous NP-complete problem (Cook 1971; Levin 1973).

SAT vs. CSP

SAT can be considered as constraint satisfaction problem:

- CSP variables = propositions
- domains = $\{\mathbf{F}, \mathbf{T}\}$
- constraints = clauses

However, we often have constraints that affect > 2 variables.

Due to this relationship, all ideas for CSPs are applicable to SAT:

- search
- inference
- variable and value orders

The DPLL Algorithm

The DPLL algorithm (Davis/Putnam/Logemann/Loveland) corresponds to backtracking with inference for CSPs.

- recursive call DPLL(Δ, I)
  for clause set Δ and partial interpretation I
- result is consistent extension of I;
  unsatisfiable if no such extension exists
- first call DPLL$(\Delta, \emptyset)$

Inference and Orders in DPLL

- simplify: after assigning value d to variable v, simplify all clauses that contain v
  $\rightsquigarrow$ forward checking (for constraints of potentially higher arity)
- unit clause heuristic: variables that occur in clauses without other variables (unit clauses) are assigned immediately
  $\rightsquigarrow$ minimum remaining values variable order
- pure symbol heuristic: variables that always occur with the same "sign" (pure variables) are assigned immediately

The DPLL Algorithm: Pseudo-Code

function DPLL(Δ, I):

if $\square \in \Delta$: [empty clause exists $\rightsquigarrow$ unsatisfiable]
    return unsatisfiable
else if $\Delta = \emptyset$: [no clauses left $\rightsquigarrow$ interpretation I satisfies sentence]
    return I
else if there is a pure variable $\{v\}$ in Δ [pure symbol heuristic]
    or a unit clause $\{v\}$ or $\{\neg v\}$ in Δ: [unit clause heuristic]
    let v be such a variable and d the associated truth value
    return DPLL(simplify(Δ, v, d), $I \cup \{v \mapsto d\}$)
else:
    select some variable v which occurs in Δ
    for each $d \in \{\mathbf{F}, \mathbf{T}\}$ in some order:
        $\Delta' :=$ simplify(Δ, v, d)
        $I' :=$ DPLL(Δ', $I \cup \{v \mapsto d\}$)
        if $I' \neq$ **unsatisfiable**
            return I'
    return unsatisfiable

The DPLL Algorithm: simplify

function simplify(Δ, v, d)

let ℓ be the literal for v that is satisfied by $v \mapsto d$

$\Delta' := \{C \mid C \in \Delta \text{ such that } \ell \notin C\}$

$\Delta'' := \{C \setminus \{\bar{\ell}\} \mid C \in \Delta'\}$

return Δ''

- Remove clauses containing ℓ
  $\rightsquigarrow$ clause is satisfied by $v \mapsto d$
- Remove $\bar{\ell}$ from remaining clauses
  $\rightsquigarrow$ clause has to be satisfied with other variable

Example

$$\Delta = \{\{\neg W, X, \neg Z\}, \{\neg W, Y\}, \{X, Y, \neg Z\}, \{\neg X, \neg Y\}, \{Z\}, \{X, \neg Y\}\}$$

Example

$$\Delta = \{\{\neg W, X, \neg Z\}, \{\neg W, Y\}, \{X, Y, \neg Z\}, \{\neg X, \neg Y\}, \{Z\}, \{X, \neg Y\}\}$$

1. pure symbol heuristic: $W \mapsto \mathbf{F}$

Example

$$\Delta = \{\{\neg W, X, \neg Z\}, \{\neg W, Y\}, \{X, Y, \neg Z\}, \{\neg X, \neg Y\}, \{Z\}, \{X, \neg Y\}\}$$

1. pure symbol heuristic: $W \mapsto \mathbf{F}$
   $\{\{X, Y, \neg Z\}, \{\neg X, \neg Y\}, \{Z\}, \{X, \neg Y\}\}$

Example

$$\Delta = \{\{\neg W, X, \neg Z\}, \{\neg W, Y\}, \{X, Y, \neg Z\}, \{\neg X, \neg Y\}, \{Z\}, \{X, \neg Y\}\}$$

1. pure symbol heuristic: $W \mapsto \mathbf{F}$
   $\{\{X, Y, \neg Z\}, \{\neg X, \neg Y\}, \{Z\}, \{X, \neg Y\}\}$
2. unit clause heuristic: $Z \mapsto \mathbf{T}$

Example

$$\Delta = \{\{\neg W, X, \neg Z\}, \{\neg W, Y\}, \{X, Y, \neg Z\}, \{\neg X, \neg Y\}, \{Z\}, \{X, \neg Y\}\}$$

1. pure symbol heuristic: $W \mapsto \mathbf{F}$
   $\{\{X, Y, \neg Z\}, \{\neg X, \neg Y\}, \{Z\}, \{X, \neg Y\}\}$
2. unit clause heuristic: $Z \mapsto \mathbf{T}$
   $\{\{X, Y\}, \{\neg X, \neg Y\}, \{X, \neg Y\}\}$

Example

$$\Delta = \{\{\neg W, X, \neg Z\}, \{\neg W, Y\}, \{X, Y, \neg Z\}, \{\neg X, \neg Y\}, \{Z\}, \{X, \neg Y\}\}$$

1. pure symbol heuristic: $W \mapsto \mathbf{F}$
   $\{\{X, Y, \neg Z\}, \{\neg X, \neg Y\}, \{Z\}, \{X, \neg Y\}\}$
2. unit clause heuristic: $Z \mapsto \mathbf{T}$
   $\{\{X, Y\}, \{\neg X, \neg Y\}, \{X, \neg Y\}\}$
3. splitting on variable X:

Example

$$\Delta = \{\{\neg W, X, \neg Z\}, \{\neg W, Y\}, \{X, Y, \neg Z\}, \{\neg X, \neg Y\}, \{Z\}, \{X, \neg Y\}\}$$

1. pure symbol heuristic: $W \mapsto \mathbf{F}$
   $\{\{X, Y, \neg Z\}, \{\neg X, \neg Y\}, \{Z\}, \{X, \neg Y\}\}$
2. unit clause heuristic: $Z \mapsto \mathbf{T}$
   $\{\{X, Y\}, \{\neg X, \neg Y\}, \{X, \neg Y\}\}$
3. splitting on variable X:

2a. $X \mapsto \mathbf{F}$
$\{\{Y\}, \{\neg Y\}\}$

Example

$$\Delta = \{\{\neg W, X, \neg Z\}, \{\neg W, Y\}, \{X, Y, \neg Z\}, \{\neg X, \neg Y\}, \{Z\}, \{X, \neg Y\}\}$$

1. pure symbol heuristic: $W \mapsto \mathbf{F}$
   $\{\{X, Y, \neg Z\}, \{\neg X, \neg Y\}, \{Z\}, \{X, \neg Y\}\}$
2. unit clause heuristic: $Z \mapsto \mathbf{T}$
   $\{\{X, Y\}, \{\neg X, \neg Y\}, \{X, \neg Y\}\}$
3. splitting on variable X:

2a. $X \mapsto \mathbf{F}$
$\{\{Y\}, \{\neg Y\}\}$

3a. unit clause heuristic: $Y \mapsto \mathbf{T}$
$\{\square\}$

Example

$$\Delta = \{\{\neg W, X, \neg Z\}, \{\neg W, Y\}, \{X, Y, \neg Z\}, \{\neg X, \neg Y\}, \{Z\}, \{X, \neg Y\}\}$$

1. pure symbol heuristic: $W \mapsto \mathbf{F}$
   $\{\{X, Y, \neg Z\}, \{\neg X, \neg Y\}, \{Z\}, \{X, \neg Y\}\}$
2. unit clause heuristic: $Z \mapsto \mathbf{T}$
   $\{\{X, Y\}, \{\neg X, \neg Y\}, \{X, \neg Y\}\}$
3. splitting on variable X:

2a. $X \mapsto \mathbf{F}$
$\{\{Y\}, \{\neg Y\}\}$

3a. unit clause heuristic: $Y \mapsto \mathbf{T}$
$\{\square\}$

2b. $X \mapsto \mathbf{T}$
$\{\{\neg Y\}\}$

Example

$$\Delta = \{\{\neg W, X, \neg Z\}, \{\neg W, Y\}, \{X, Y, \neg Z\}, \{\neg X, \neg Y\}, \{Z\}, \{X, \neg Y\}\}$$

1. pure symbol heuristic: $W \mapsto \mathbf{F}$
   $\{\{X, Y, \neg Z\}, \{\neg X, \neg Y\}, \{Z\}, \{X, \neg Y\}\}$
2. unit clause heuristic: $Z \mapsto \mathbf{T}$
   $\{\{X, Y\}, \{\neg X, \neg Y\}, \{X, \neg Y\}\}$
3. splitting on variable X:

2a. $X \mapsto \mathbf{F}$
$\{\{Y\}, \{\neg Y\}\}$

3a. unit clause heuristic: $Y \mapsto \mathbf{T}$
$\{\square\}$

2b. $X \mapsto \mathbf{T}$
$\{\{\neg Y\}\}$

3b. unit clause heuristic: $Y \mapsto \mathbf{F}$
$\{\}$

Example

$$\Delta = \{\{\neg W, X, \neg Z\}, \{\neg W, Y\}, \{X, Y, \neg Z\}, \{\neg X, \neg Y\}, \{Z\}, \{X, \neg Y\}\}$$

1. pure symbol heuristic: $W \mapsto \mathbf{F}$
   $\{\{X, Y, \neg Z\}, \{\neg X, \neg Y\}, \{Z\}, \{X, \neg Y\}\}$
2. unit clause heuristic: $Z \mapsto \mathbf{T}$
   $\{\{X, Y\}, \{\neg X, \neg Y\}, \{X, \neg Y\}\}$
3. splitting on variable X:

2a. $X \mapsto \mathbf{F}$
$\{\{Y\}, \{\neg Y\}\}$

3a. unit clause heuristic: $Y \mapsto \mathbf{T}$
$\{\square\}$

2b. $X \mapsto \mathbf{T}$
$\{\{\neg Y\}\}$

3b. unit clause heuristic: $Y \mapsto \mathbf{F}$
$\{\}$

Properties of DPLL

- DPLL is sound and complete
- DPLL computes a model where φ is true if such a model exists
  - some variables possibly remain unassigned in the solution I; their values can be chosen arbitrarily
- time complexity in general exponential

$\rightsquigarrow$ important in practice: good variable order and additional inference methods (in particular clause learning)

- best known SAT algorithms are based on DPLL

DPLL on Horn Formulas

Horn Formulas

important special case: Horn formulas

Definition (Horn formula)

A Horn clause is a clause with at most one positive literal, i.e., of the form

$$\neg x_1 \lor \dots \lor \neg x_n \lor y \text{ or } \neg x_1 \lor \dots \lor \neg x_n$$

($n = 0$ is allowed.)

A Horn formula is a propositional formula in conjunctive normal form that only consists of Horn clauses.

- foundation of logic programming (e.g., PROLOG)
- critical in many kinds of practical reasoning problems

DPLL on Horn Formulas

Proposition (DPLL on Horn formulas)

If the input formula φ is a Horn formula, then the time complexity of DPLL is polynomial in the length of φ.

Summary

- Reasoning: the formula ψ follows from the set of formulas Φ if all models of Φ are also models of ψ.
- Reasoning can be reduced to testing validity (with the deduction theorem).
- Testing validity can be reduced to testing unsatisfiability.
- Resolution can be applied to formulas in conjunctive normal form.
  $\rightsquigarrow$ can be used to test if a set of clauses is unsatisfiable.
- DPLL: systematic backtracking search with unit propagation
- DPLL successful in practice, polynomial on Horn formulas