Propositional Logic

Propositional Logic: Basic Ideas

The elementary building blocks of propositional logic are

- atomic propositions (or simply atoms) that cannot be decomposed any further: E.g.,
  - "The block is red"
  - "It is raining"
- logical connectives "and", "or", "not", by which we can build propositional formulas

Propositional Logic: syntax

Atomic Propositions

- $\bot$ (denoting false)
- $\top$ (denoting true)
- Any letter of the alphabet, e.g.: p
- Any letter of the alphabet with a numeric subscript and/or superscript, e.g.: q_4, p^7, r_2'
- Any alphanumeric string, e.g.: “Tom is the driver”

is an atomic proposition (or simply and atom)

Well-Formed Propositions (WFPs)

1. Every atomic proposition is a wfp
2. If α is a wfp, then so is $(\neg\alpha)$
3. If α and β are wfps, then so are

(conjunction)	$(\alpha \wedge \beta)$	(disjunction)	$(\alpha \vee \beta)$
(implication)	$(\alpha \rightarrow \beta)$	(equivalence)	$(\alpha \leftrightarrow \beta)$

4. Nothing else is a wfp

- Parentheses may be omitted
  - we allow $(p_1 \wedge \cdots \wedge p_n)$ and $(p_1 \vee \cdots \vee p_n)$
- Square brackets may be used instead of parentheses
- The symbols $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ are called logical connectives

Examples of (WFPs)

$$((p \wedge (q \vee c)) \rightarrow d)$$

(“Betty drives Tom” $\rightarrow$ ($\neg$ “Tom is the driver”))

Summary: Syntax of Propositional Logic

Countable alphabet Σ of atomic propositions: $a, b, c, \ldots$

$$
\begin{array}{lcll}
\alpha, \beta & \longrightarrow & a & \text{(atom)} \\
 & | & \bot & \text{(false)} \\
 & | & \top & \text{(true)} \\
 & | & (\neg\alpha) & \text{(negation)} \\
 & | & (\alpha \wedge \beta) & \text{(conjunction)} \\
 & | & (\alpha \vee \beta) & \text{(disjunction)} \\
 & | & (\alpha \rightarrow \beta) & \text{(implication)} \\
 & | & (\alpha \leftrightarrow \beta) & \text{(equivalence)}
\end{array}
$$

Atom : atomic proposition

Literal : atomic proposition or negated atomic proposition (e.g., a, $\neg b$)

Semantics: Intuition

- Atomic statements can be true (T) or false (F)
- The truth value of formulas is determined by the truth values of the atoms

Example: $(a \vee b) \wedge c$

- If a and b are false and c is true, then the formula is false
- If a and c are true, then the formula is true

Semantics: formally

- A truth value assignment (or interpretation) of the atoms in Σ is a function $\mathcal{I}$:

$$\mathcal{I} : \Sigma \to \{\mathsf{T}, \mathsf{F}\}$$

- Instead of $\mathcal{I}(a)$ we also write $a^{\mathcal{I}}$
- A formula α is satisfied by an interpretation $\mathcal{I}$, denoted $\mathcal{I} \models \alpha$ iff

$$\mathcal{I} \models \top \quad \mathcal{I} \not\models \bot$$

$\mathcal{I} \models a$	iff	$a^{\mathcal{I}} = \mathsf{T}$
$\mathcal{I} \models \neg\alpha$	iff	$\mathcal{I} \not\models \alpha$
$\mathcal{I} \models \alpha \wedge \beta$	iff	$\mathcal{I} \models \alpha$ and $\mathcal{I} \models \beta$
$\mathcal{I} \models \alpha \vee \beta$	iff	$\mathcal{I} \models \alpha$ or $\mathcal{I} \models \beta$
$\mathcal{I} \models \alpha \to \beta$	iff	if $\mathcal{I} \models \alpha$ then $\mathcal{I} \models \beta$
$\mathcal{I} \models \alpha \leftrightarrow \beta$	iff	$\mathcal{I} \models \alpha$ if and only if $\mathcal{I} \models \beta$

Example

- Consider the formula α

$$(a \vee b) \wedge c$$

- Let $\mathcal{I}_1$ be the interpretation

$$\begin{aligned} a^{\mathcal{I}_1} &= \mathsf{T} \\ b^{\mathcal{I}_1} &= \mathsf{F} \\ c^{\mathcal{I}_1} &= \mathsf{T} \end{aligned}$$

then $\mathcal{I}_1 \models \alpha$

$$\begin{aligned} \mathcal{I}_1 \models (a \vee b) \wedge c \quad \text{iff} \quad & \mathcal{I}_1 \models (a \vee b) \text{ and } \mathcal{I}_1 \models c \\ \text{iff} \quad & \mathcal{I}_1 \models (a \vee b) \text{ and } c^{\mathcal{I}_1} = \mathsf{T} \\ \text{iff} \quad & (\mathcal{I}_1 \models a \text{ or } \mathcal{I}_1 \models b) \text{ and } c^{\mathcal{I}_1} = \mathsf{T} \\ \text{iff} \quad & (a^{\mathcal{I}_1} = \mathsf{T} \text{ or } b^{\mathcal{I}_1} = \mathsf{T}) \text{ and } c^{\mathcal{I}_1} = \mathsf{T} \end{aligned}$$

Example

- Consider the formula α

$$(a \vee b) \wedge c$$

- Let $\mathcal{I}_2$ be the interpretation

$$\begin{aligned} a^{\mathcal{I}_2} &= \mathsf{F} \\ b^{\mathcal{I}_2} &= \mathsf{F} \\ c^{\mathcal{I}_2} &= \mathsf{T} \end{aligned}$$

then $\mathcal{I}_2 \not\models \alpha$

Truth Tables

The truth of a formula γ in an interpretation $\mathcal{I}$ (denoted $\gamma^{\mathcal{I}}$) can also be determined using truth tables

α	$\neg\alpha$
T	F
F	T

α	β	$\alpha \wedge \beta$
F	F	F
F	T	F
T	F	F
T	T	T

α	β	$\alpha \vee \beta$
F	F	F
F	T	T
T	F	T
T	T	T

α	β	$\alpha \rightarrow \beta$
F	F	T
F	T	T
T	F	F
T	T	T

α	β	$\alpha \leftrightarrow \beta$
F	F	T
F	T	F
T	F	F
T	T	T

Example

- Consider the formula α

$$(a \vee b) \wedge c$$

- Let $\mathcal{I}_1$ be the interpretation

$$\begin{aligned} a^{\mathcal{I}_1} &= \mathsf{T} \\ b^{\mathcal{I}_1} &= \mathsf{F} \\ c^{\mathcal{I}_1} &= \mathsf{T} \end{aligned}$$

  then $\mathcal{I}_1 \models \alpha$

- In fact, $\alpha^{\mathcal{I}_1} = \mathsf{T}$

$$\begin{aligned} \alpha^{\mathcal{I}_1} &= (a^{\mathcal{I}_1} \vee b^{\mathcal{I}_1}) \wedge c^{\mathcal{I}_1} \\ &= (\mathsf{T} \vee \mathsf{F}) \wedge \mathsf{T} \\ &= \mathsf{T} \wedge \mathsf{T} \\ &= \mathsf{T} \end{aligned}$$

Example

- Consider the formula α

$$(a \vee b) \wedge c$$

- Let $\mathcal{I}_2$ be the interpretation

$$\begin{aligned} a^{\mathcal{I}_2} &= \mathsf{F} \\ b^{\mathcal{I}_2} &= \mathsf{F} \\ c^{\mathcal{I}_2} &= \mathsf{T} \end{aligned}$$

  then $\mathcal{I}_2 \not\models \alpha$

- In fact, $\alpha^{\mathcal{I}_1} = \mathsf{F}$

$$\begin{aligned} \alpha^{\mathcal{I}_2} &= (a^{\mathcal{I}_2} \vee b^{\mathcal{I}_2}) \wedge c^{\mathcal{I}_2} \\ &= (\mathsf{F} \vee \mathsf{F}) \wedge \mathsf{T} \\ &= \mathsf{F} \wedge \mathsf{T} \\ &= \mathsf{F} \end{aligned}$$

Semantics: Interpretations as $0-1$ functions

- An interpretation can also be specified as a function $\mathcal{I}\colon \Sigma \to \{0,1\}$
- The intuition is that $a^{\mathcal{I}} = 1$ means that a is True, while $a^{\mathcal{I}} = 0$ means that a is False:

$$\mathcal{I} \models a \quad \text{iff} \quad a^{\mathcal{I}} = 1$$

- The truth $\alpha^{\mathcal{I}}$ of a formula α in $\mathcal{I}$ can be established using the rules:

$$\begin{array}{lcl}
(\neg\alpha)^{\mathcal{I}} & = & 1 - \alpha^{\mathcal{I}} \\
(\alpha \vee \beta)^{\mathcal{I}} & = & \max(\alpha^{\mathcal{I}}, \beta^{\mathcal{I}}) \\
(\alpha \wedge \beta)^{\mathcal{I}} & = & \min(\alpha^{\mathcal{I}}, \beta^{\mathcal{I}}) \\
(\alpha \rightarrow \beta)^{\mathcal{I}} & = & \max(1 - \alpha^{\mathcal{I}}, \beta^{\mathcal{I}}) \\
(\alpha \leftrightarrow \beta)^{\mathcal{I}} & = & 1 - |\alpha^{\mathcal{I}} - \beta^{\mathcal{I}}|
\end{array}$$

Example

- Consider the formula α

$$(a \vee b) \wedge c$$

- Let $\mathcal{I}_1$ be the interpretation

$$\begin{aligned} a^{\mathcal{I}_1} &= 1 \\ b^{\mathcal{I}_1} &= 0 \\ c^{\mathcal{I}_1} &= 1 \end{aligned}$$

  then $\mathcal{I}_1 \models \alpha$

- In fact, $\alpha^{\mathcal{I}_1} = 1$

$$\begin{aligned} \alpha^{\mathcal{I}_1} &= (a^{\mathcal{I}_1} \vee b^{\mathcal{I}_1}) \wedge c^{\mathcal{I}_1} \\ &= \min(\max(1,0),1) \\ &= \min(1,1) \\ &= 1 \end{aligned}$$

Example

- Consider the formula α

$$(a \vee b) \wedge c$$

- Let $\mathcal{I}_2$ be the interpretation

$$\begin{aligned} a^{\mathcal{I}_2} &= 0 \\ b^{\mathcal{I}_2} &= 0 \\ c^{\mathcal{I}_2} &= 1 \end{aligned}$$

  then $\mathcal{I}_2 \not\models \alpha$

- In fact, $\alpha^{\mathcal{I}_1} = 0$

$$\begin{aligned} \alpha^{\mathcal{I}_2} &= (a^{\mathcal{I}_2} \vee b^{\mathcal{I}_2}) \wedge c^{\mathcal{I}_2} \\ &= \min(\max(0,0),1) \\ &= \min(0,1) \\ &= 0 \end{aligned}$$

Semantics: Interpretations as sets

- An interpretation can also be specified as a subset of Σ, i.e. $\mathcal{I} \subseteq \Sigma$
- The intuition is that the atoms in $\mathcal{I}$ are considered True, while the others are considered False:

$$\mathcal{I} \models a \quad \text{iff} \quad a \in \mathcal{I}$$

- For instance, the interpretation $\mathcal{I}$

$$\begin{aligned} a^{\mathcal{I}} &= \mathsf{T} \\ b^{\mathcal{I}} &= \mathsf{F} \\ c^{\mathcal{I}} &= \mathsf{T} \end{aligned}$$

can be represented as

$$\mathcal{I} = \{a, b\}$$

How many interpretations do exists?

- Suppose there are has n different atoms
- Each atom is either T or F, $\rightarrow$ there are 2^n interpretations
- Example: given α as the formula $(a \vee b) \wedge c$, there are $2^3 = 8$ different interpretations for α

Interpretation	a	b	c	Binay Representation	Set Representation
$\mathcal{I}_1$	F	F	F	$\langle 0,0,0 \rangle$	$\emptyset$
$\mathcal{I}_2$	F	F	T	$\langle 0,0,1 \rangle$	$\{c\}$
$\mathcal{I}_3$	F	T	F	$\langle 0,1,0 \rangle$	$\{b\}$
$\mathcal{I}_4$	F	T	T	$\langle 0,1,1 \rangle$	$\{b,c\}$
$\mathcal{I}_5$	T	F	F	$\langle 1,0,0 \rangle$	$\{a\}$
$\mathcal{I}_6$	T	F	T	$\langle 1,0,1 \rangle$	$\{a,c\}$
$\mathcal{I}_7$	T	T	F	$\langle 1,1,0 \rangle$	$\{a,b\}$
$\mathcal{I}_8$	T	T	T	$\langle 1,1,1 \rangle$	$\{a,b,c\}$

- The interpretations correspond to all possible subsets of $\{a,b,c\}$
- Note: $\mathcal{I}_j \models \alpha$ iff $j \in \{4,6,8\}$

Satisfiability and Validity

- An interpretation $\mathcal{I}$ is a model of α iff $\mathcal{I} \models \alpha$
- An interpretation $\mathcal{I}$ is a model of set *KB* of formulae *KB* iff $\mathcal{I} \models \alpha$ for all $\alpha \in$ *KB*
- A formula α (a set of formulae *KB*) is
  - satisfiable, if there is some $\mathcal{I}$ that satisfies α (*KB*)
  - unsatisfiable, if α is not satisfiable
  - falsifiable, if there is some $\mathcal{I}$ that does not satisfy α
  - valid (i.e. a tautology), if every $\mathcal{I}$ is a model of α
- Two formluae α, β are logically equivalent (denoted $\alpha \equiv \beta$), if for all $\mathcal{I}$:

$$\mathcal{I} \models \alpha \text{ iff } \mathcal{I} \models \beta$$

Examples

- Satisfiable: $a \vee (b \wedge c)$
- Unsatisfiable: $(a \vee b) \wedge (\neg a \vee c) \wedge (\neg b \vee \neg c)$
- Falsifiable: $a \vee (b \wedge c)$
- Valid: $(a \wedge (a \rightarrow b)) \rightarrow b)$
- Logically equivalent: $a \vee (b \wedge c) \equiv (a \vee b) \wedge (a \vee c)$

Some Consequences

Proposition:

- α is valid iff $\neg\alpha$ is unsatisfiable
- α is unsatisfiable iff $\neg\alpha$ is valid

Proposition: $\alpha \equiv \beta$ iff $\alpha \leftrightarrow \beta$ is valid

Proposition: If $\alpha \equiv \beta$, and δ is the result of replacing α in γ by β, then $\gamma \equiv \delta$.

Equivalences (I)

Commutativity

$$\alpha \vee \beta \equiv \beta \vee \alpha$$
$$\alpha \wedge \beta \equiv \beta \wedge \alpha$$
$$\alpha \leftrightarrow \beta \equiv \beta \leftrightarrow \alpha$$

Associativity

$$(\alpha \vee \beta) \vee \gamma \equiv \alpha \vee (\beta \vee \gamma)$$
$$(\alpha \wedge \beta) \wedge \gamma \equiv \alpha \wedge (\beta \wedge \gamma)$$

Idempotence

$$\alpha \vee \alpha \equiv \alpha$$
$$\alpha \wedge \alpha \equiv \alpha$$

Absorption

$$\alpha \vee (\alpha \wedge \beta) \equiv \alpha$$
$$\alpha \wedge (\alpha \vee \beta) \equiv \alpha$$

Distributivity

$$\alpha \vee (\beta \wedge \gamma) \equiv (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$$
$$\alpha \wedge (\beta \vee \gamma) \equiv (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$$

Equivalences (II)

Tautology	$\alpha \vee \mathsf{T}$	$\equiv$	T
	$\alpha \vee \neg\alpha$	$\equiv$	T
Unsatisfiability	$\alpha \wedge \mathsf{F}$	$\equiv$	F
	$\alpha \wedge \neg\alpha$	$\equiv$	F
Neutrality	$\alpha \wedge \mathsf{T}$	$\equiv$	α
	$\alpha \vee \mathsf{F}$	$\equiv$	α
Double Negation	$\neg\neg\alpha$	$\equiv$	α
De Morgan Law	$\neg(\alpha \vee \beta)$	$\equiv$	$(\neg\alpha) \wedge (\neg\beta)$
	$\neg(\alpha \wedge \beta)$	$\equiv$	$(\neg\alpha) \vee (\neg\beta)$
Implication	$\alpha \to \beta$	$\equiv$	$(\neg\alpha) \vee \beta$
	$\neg(\alpha \to \beta)$	$\equiv$	$\alpha \wedge (\neg\beta)$
Equivalence	$\alpha \leftrightarrow \beta$	$\equiv$	$(\alpha \to \beta) \wedge (\beta \to \alpha)$
	$\neg(\alpha \leftrightarrow \beta)$	$\equiv$	$(\neg\alpha \wedge \beta) \vee (\neg\beta \wedge \alpha)$

Normal Forms

There exists some standardized forms of formulae:

- Negation Normal Form (NNF): only atoms can be negated
  Example: $(a \vee (\neg b)) \wedge ((\neg c) \rightarrow ((\neg b) \wedge d))$
- Conjunctive Normal Form (CNF): conjunction of disjunctions of literals (called clauses)

  $(l_{11} \vee l_{12} \vee \ldots \vee l_{1n_1}) \wedge (l_{21} \vee l_{22} \vee \ldots \vee l_{2n_2}) \wedge \ldots \wedge (l_{m1} \vee l_{m2} \vee \ldots \vee l_{mn_m})$

  Example: $(a \vee (\neg b)) \wedge ((\neg c) \vee (\neg b) \vee c) \wedge (c \vee a \vee (\neg d))$
- Disjunctive Normal Form (DNF): disjunction of conjunctions of literals

  $(l_{11} \wedge l_{12} \wedge \ldots \wedge l_{1n_1}) \vee (l_{21} \wedge l_{22} \wedge \ldots \wedge l_{2n_2}) \vee \ldots \vee (l_{m1} \wedge l_{m2} \wedge \ldots \wedge l_{mn_m})$

  Example: $(a \wedge (\neg b)) \vee ((\neg c) \wedge (\neg b) \wedge c) \vee (c \wedge a \wedge (\neg d))$

Normal Forms, cont.

- Horn Form: conjunction of *Horn clauses* (clauses with at most 1 atom) Example:

$$(a \vee (\neg b)) \wedge ((\neg c) \vee (\neg b) \vee c) \wedge (c \vee (\neg d))$$

Proposition: For every formula, there exists an equivalent formula in NNF, one in CNF and one in DNF.

Transformation into NNF

Apply the following equivalences

$$
\begin{array}{lcl}
\neg\neg\alpha & \equiv & \alpha \\
\neg(\alpha \vee \beta) & \equiv & (\neg\alpha) \wedge (\neg\beta) \\
\neg(\alpha \wedge \beta) & \equiv & (\neg\alpha) \vee (\neg\beta) \\
\neg(\alpha \rightarrow \beta) & \equiv & \alpha \wedge (\neg\beta) \\
\neg(\alpha \leftrightarrow \beta) & \equiv & (\neg\alpha \wedge \beta) \vee (\neg\beta \wedge \alpha)
\end{array}
$$

Exercise: convert into NNF:

$$(\neg(a \vee (\neg b))) \wedge (\neg(c \rightarrow ((\neg b) \wedge d)))$$

Transformation into CNF

1. Transform into NNF; then
2. Apply the following equivalences

$$\begin{array}{lcl} \alpha \vee (\beta \wedge \gamma) & \equiv & (\alpha \vee \beta) \wedge (\alpha \vee \gamma) \\ \alpha \rightarrow \beta & \equiv & (\neg\alpha) \vee \beta \\ \alpha \leftrightarrow \beta & \equiv & (\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha) \end{array}$$

Exercise: convert into CNF:

$$(\neg(a \vee (\neg b))) \wedge (\neg(c \rightarrow ((\neg b) \wedge d)))$$

Transformation into DNF

1. Transform into NNF
2. Apply the following equivalences

$$\begin{array}{lcl} \alpha \wedge (\beta \vee \gamma) & \equiv & (\alpha \wedge \beta) \vee (\alpha \wedge \gamma) \\ \alpha \rightarrow \beta & \equiv & (\neg\alpha) \vee \beta \\ \alpha \leftrightarrow \beta & \equiv & (\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha) \end{array}$$

Exercise: convert into CNF:

$$(\neg(a \vee (\neg b))) \wedge (\neg(c \rightarrow ((\neg b) \wedge d)))$$

Why Normal Forms?

- We can transform propositional formulas, in particular, we can construct their NNF, CNF and DNF
- DNF tells us something as to whether a formula is satisable. If all disjuncts contain F or complementary literals, then no model exists. Otherwise, the formula is satisfiable
- CNF tells us something as to whether a formula is a tautology. If all clauses (i.e., conjuncts) contain T or complementary literals, then the formula is a tautology. Otherwise, the formula is falsifiable

But,

- the transformation into DNF or CNF may be expensive (in time/space) Example:

$$(a \wedge b) \vee (c \wedge d) \overset{CNF}{\mapsto} (a \vee c) \wedge (a \vee d) \wedge (b \vee c) \wedge (b \vee d)$$

Satisfiability of KBs

- A set *KB* of formulae is satisfied iff $\mathcal{I} \models \alpha$ for all $\alpha \in KB$
- An interpretation $\mathcal{I}$ is a model of set *KB* of formulae (denoted $\mathcal{I} \models KB$) iff $\mathcal{I} \models \alpha$ for all $\alpha \in KB$
- A set *KB* of formulae is
  - satisfiable, if there is some $\mathcal{I}$ that satisfies *KB*
  - unsatisfiable, if *KB* is not satisfiable
- A set *KB* of formulae entails a formula α iff α is true in all models of *KB*, i.e.

$$KB \models \alpha \quad \text{iff} \quad \mathcal{I} \models \alpha \text{ for all models of } KB$$

Examples

- Satisfiable:

$$\{(a \vee b), (a \vee c)\}$$

- Unsatisfiable:

$$\{(a \vee b), (\neg a \vee b), (a \vee \neg b), (\neg a \vee \neg b)\}$$

- Entailment: assume $KB = \{a, a \rightarrow b\}$. Then

$$KB \models b$$

Some Properties of Entailment

Deduction Theorem :

$$KB \cup \{\alpha\} \models \beta \text{ iff } KB \models \alpha \rightarrow \beta$$

Contraposition Theorem :

$$KB \cup \{\alpha\} \models \neg\beta \text{ iff } KB \cup \{\beta\} \models \neg\alpha$$

Contradiction Theorem :

$$KB \models \alpha \text{ iff } KB \cup \{\neg\alpha\} \text{ is unsatisfiable}$$

Checking Entailment by Enumeration

- How can we verify whether $KB \models \alpha$?
- We enumerate all interpretations $\mathcal{I}$ and verify that:
  1. if $\mathcal{I}$ is a model of KB then $\mathcal{I}$ is also a model of α; or equivalently
  2. $KB \cup \{\neg\alpha\}$ is not satisfied by $\mathcal{I}$ (contradiction theorem)

Example

Consider $KB = \{a, a \rightarrow b\}$ and $\alpha = b$. Let us show that $KB \models \alpha$

	a	b	$a \rightarrow b$	KB	α	$KB \cup \{\neg\alpha\}$
$\mathcal{I}_1$	F	F	T	F	F	F
$\mathcal{I}_2$	F	T	T	F	T	F
$\mathcal{I}_3$	T	F	F	F	F	F
$\mathcal{I}_4$	T	T	T	T	T	F

Hence

- α is true in all models of KB; or equivalently
- $KB \cup \{\neg\alpha\}$ is unsatisfiable

Therefore, $KB \models \alpha$

Example

If a dog is a pet, and a pet is an animal, then is a dog an animal?

Atoms : dog, pet, animal

Representation :
- "a dog is a pet" $\mapsto$ dog $\rightarrow$ pet
- "a pet is an animal" $\mapsto$ pet $\rightarrow$ animal
- "a dog is an animal" $\mapsto$ dog $\rightarrow$ animal

Knowledge Base : $KB = \{\text{dog} \rightarrow \text{pet}, \text{pet} \rightarrow \text{animal}\}$

Query : $KB \models \alpha$, where $\alpha = \text{dog} \rightarrow \text{animal}$?

	dog	pet	animal	KB	α	$KB \cup \{\neg\alpha\}$
$\mathcal{I}_1$	F	F	F	T	T	F
$\mathcal{I}_2$	F	F	T	T	T	F
$\mathcal{I}_3$	F	T	F	F	T	F
$\mathcal{I}_4$	F	T	T	T	T	F
$\mathcal{I}_5$	T	F	F	F	F	F
$\mathcal{I}_6$	T	F	T	F	T	F
$\mathcal{I}_7$	T	T	F	F	F	F
$\mathcal{I}_8$	T	T	T	T	T	F

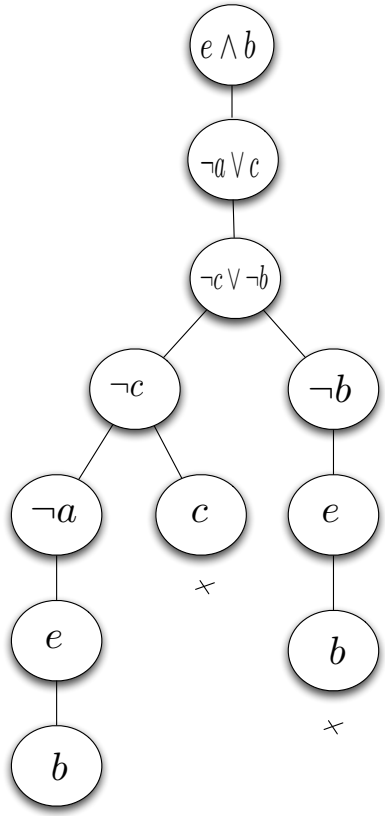
Hence

- α is true in all models of KB; or equivalently
- $KB \cup \{\neg\alpha\}$ is unsatisfiable

Therefore, $KB \models \alpha$

Checking KB Satisfiability using analytic tableaux

- A tableaux is tree, where each node is a formula α
- Tableau Inference Rules
  1. If a path contains $\alpha \wedge \beta$ then it contains α and β
  2. If a path contains $\alpha \vee \beta$ then it contains either α or β
- A clash is a path containing α and $\neg\alpha$
- A tableau is clash free if there is path not being a clash
- A tableau is complete if no rule can be applied to a path
- A KB is satisfiable iff there is a clash-free and complete tableau for it



Checking KB Satisfiability using DPLL algorithm

DPLL: Davis-Putnam-Logemann-Loveland

1. Let $C = \bigwedge_{F \in KB} CNF(F)$, where $CNF(F)$ transforms F into CNF
2. Return $DPLL(C)$

Function DPLL(C)

input : A formula C in CNF

output: True if C satisfiable, false otherwise

repeat
  if *C literal consistent* **then**
    return *true*
  end
  if *(C contains a conjunct that is* $\bot$*)* **OR** *(C not literal consistent)* **then**
    return *false*
  end
  foreach *conjunct in C being a literal l* **do** $C = C[l/\top]$;
  foreach *literal l that occurs pure in C* **do** $C = C[l/\top]$

until *none of the previous steps is applicable*;

$l := chooseLiteral(C)$;

return $DPLL(C[l/\top])$ **OR** $DPLL(C[l/\bot])$ *;*

- For $C = l_1 \wedge \ldots \wedge l_n$, C literal consistent iff not both l and $\neg l$ occur in C for some letter l
- literal l occurs pure in C iff $\neg l$ does not occur in C
- $C[l/\top]$ is as C in which any occurrence of l is replaced with $\top$, $C[l/\bot]$ is as C in which any occurrence of l is replaced with $\bot$, and
  - $F \vee \top$ is replaced with $\top$, $F \vee \bot$ is replaced with F

Checking KB Satisfiability using Resolution

- A formula $C = C_1 \wedge \ldots \wedge C_i \wedge \ldots \wedge C_n$ in CNF, where $C_i = l_{i_1} \vee \ldots \vee l_{i_{k_i}}$ can be represented as a set of clauses, where a clause is a set of literals
  - $C_i = \{l_{i_1}, \ldots, l_{i_{k_i}}\}$, $C = \{C_1, \ldots, C_n\}$
- Resolution rule: from clauses $C = \{\ldots, l, \ldots\}$ and $C' = \{\ldots, \neg l, \ldots\}$ infer $C \cup C' \setminus \{l, \neg l\}$
- We have: for a *KB* being a set of clauses *KB* unsatisfiable iff the empty clause can be inferred

Example

Consider $C_1 = \{a, b\}$, $C_2 = \{\neg a, c\}$, $C_3 = \{\neg b\}$, $C_4 = \{\neg c\}$

1. from C_1 and C_2 infer $C_5 = \{b, c\}$
2. from C_4 and C_3 infer $C_6 = \{c\}$
3. from C_6 and C_4 infer $C_7 = \emptyset$

Therefore, $C = \{C_1, C_2, C_3, C_4\}$ is not satisfiable

Checking KB Satisfiability using ILP

- An alternative method for satisfiability checking consists on relying on Integer Linear Programming (ILP)
- Basic idea:
  - For a formula ϕ consider a variable x_ϕ taking values in {0,1}
  - The intuition is that ϕ is true iff $x_\phi = 1$
  - Apply semantic preserving transformations, generating ILP equations
  - Check if the set of inequations has a solution

Consider a knowledge base KB

1. $EQ_{KB} = \emptyset$
2. For all $\phi \in KB$, $EQ_{KB} := EQ_{KB} \cup \{x_\phi = 1, \sigma(\phi)\}$

$$\sigma(\phi) = \begin{cases} x_p \in \{0,1\} & \text{if} \quad \phi = p \\ x_\phi = 1 - x_{\phi'}, \sigma(\phi'), x_\phi \in \{0,1\} & \text{if} \quad \phi = \neg\phi' \\ x_\phi = \min(x_{\phi_1}, x_{\phi_2}) \\ \sigma(\phi_1), \sigma(\phi_2), x_\phi \in \{0,1\} & \text{if} \quad \phi = \phi_1 \wedge \phi_2 \\ x_\phi = \max(x_{\phi_1}, x_{\phi_2}) \\ \sigma(\phi_1), \sigma(\phi_2), x_\phi \in \{0,1\} & \text{if} \quad \phi = \phi_1 \vee \phi_2 \\ \sigma(\neg\phi_1 \vee \phi_2) & \text{if} \quad \phi = \phi_1 \rightarrow \phi_2 \\ \sigma((\phi_1 \rightarrow \phi_2) \wedge (\phi_2 \rightarrow \phi_1)) & \text{if} \quad \phi = \phi_1 \leftrightarrow \phi_2 \; . \end{cases}$$

Note1: $x = \min(y, z)$ is $x \leq y, x \leq z, x \geq y + z - 1$

Note2: $x = \max(y, z)$ is $x \geq y, x \geq z, x \leq y + z$

3. KB satisfiable iff EQ_{KB} has a solution

Example

Consider $KB = \{a, a \rightarrow b\}$. Let us show that $KB \models b$

1. Consider $KB' = KB \cup \{\neg b\}$
2. We have to show that KB' not satisfiable
3. Compute $EQ_{KB'}$

$$\begin{aligned} EQ_{KB'} &= \{x_a = 1, x_{\neg a \vee b} = 1, x_{\neg b} = 1\} \\ &\cup \{x_{\neg b} = 1 - x_b, x_b \in \{0, 1\}\} \\ &\cup \{x_{\neg a} + x_b \geq x_{\neg a \vee b}, x_{\neg a} \leq x_{\neg a \vee b}, x_b \leq x_{\neg a \vee b}\} \\ &\cup \{x_{\neg a} = 1 - x_a, x_{\neg a} \in \{0, 1\}\} \end{aligned}$$

4. It can be verified that $EQ_{KB'}$ does not have a solution

Exercises: Propositional Logic

Using a SAT Solver

- A SAT solver is a program that determines whether a set S of formulae in CNF is satisfiable or not
- Usually, if S is satisfiable then a SAT solver provides also a model $\mathcal{I}$ of S
- Hence, for instance, we may use a SAT solver to
  - check if $KB \models \alpha$ by verifying whether $KB \cup \{\neg\alpha\}$ is unsatisfiable
  - find a model $\mathcal{I}$ (solution) to a problem encoded as a set of formulae S

Some SAT Solvers

- For the exercises, use a SAT solver, i.e. a program determining the satisfiability or unsatisfiability of a KB in CNF. For instance,

sat4j	`http://www.sat4j.org/` (java)
zchaff	`http://www.princeton.edu/~chaff/zchaff.html` (C)
minisat	`http://www.cs.chalmers.se/Cs/Research/FormalMethods/MiniSat/` (C)

SAT Solver Input File Format

- The input file format of a SAT solver is usually a formula in CNF
- Typically, the DIMACS format is used

Comment line a comment line starts with the letter `c`

```
c Example of a comment line
```

Problem line a problem line has the following format

```
p cnf VARIABLES CLAUSES
```

where *VARIABLES* is the number of variables of the problem and *CLAUSES* is the number of clauses of the problem

Clause line atom are positive integers, negated atoms are negative integers. Each clause line terminates with the value 0. For instance,

```
1 -2 3 0
```

represents the clause $(a_1 \vee (\neg a_2) \vee a_3)$

Example File

An input file for the formula

$$(a_1 \vee a_3 \vee (\neg a_4)) \wedge (a_4) \wedge (a_2 \vee (\neg a_3))$$

is

```
c
c Example CNF format file
c
p cnf 4 3
1 3 -4 0
4 0
2 -3 0
```

Example calls:

sat4j	java -jar sat4j-1.6.jar myfile.cnf
zchaff	zchaff myfile.cnf
minisat	minisat myfile.cnf

Example 1

- Assume that
$$KB = \{a, a \implies b\}$$
- Prove, using sat4j, that
$$KB \models b$$
$$KB \not\models c$$
- The input problem has to be a satisfiability problem of a CNF formula
- So, we have to rewrite the $KB \models b$ decision problem into a satisfiability problem
- But, we know that $KB \models b$ iff $KB \cup \{\neg b\}$ is unsatisfiable

- Hence, it suffices to represent $KB \cup \{\neg b\}$ in CNF and check it for unsatisfiability

$$KB \cup \{\neg b\} \quad \equiv \quad \{a, \neg a \vee b, \neg b\}$$

- Assignment to propositional atoms

$$\begin{array}{lcl} a & \mapsto & 1 \\ b & \mapsto & 2 \end{array}$$

- The file encoding may be

```
c Exercise 1a CNF format file
p cnf 3 2
1 0
-1 2 0
-2 0
```

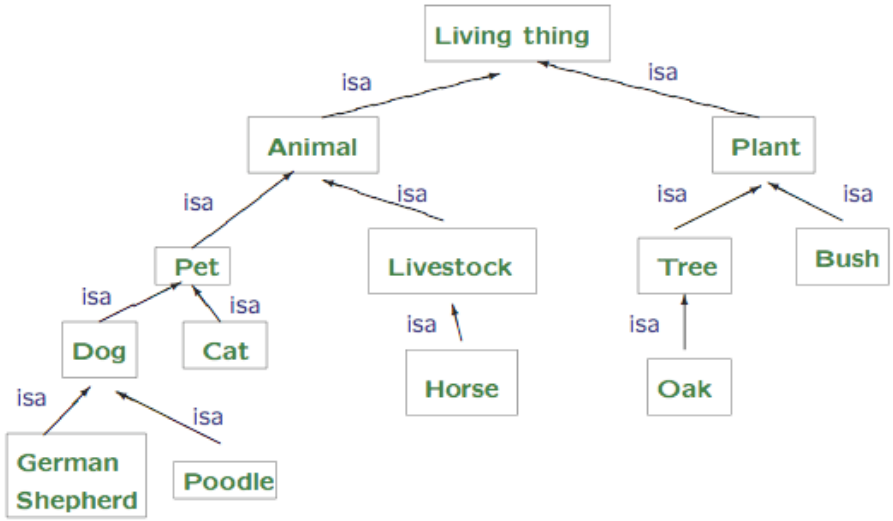
- java -jar sat4j-1.6.jar Exercise1a.cnf returns “s UNSATISFIABLE”
- Hence, $KB \models b$

- For the second problem, we know that $KB \models c$ iff $KB \cup \{\neg c\}$ is unsatisfiable
- Hence, it suffices to represent $KB \cup \{\neg c\}$ in CNF and check it for unsatisfiability ($KB \cup \{\neg c\} \equiv \{a, \neg a \vee b, \neg c\}$)
- Assignment to propositional atoms $a \mapsto 1, b \mapsto 2, c \mapsto 3$
- The file encoding may be

```
c Exercise 1b CNF format file
p cnf 3 3
1 0
-1 2 0
-3 0
```

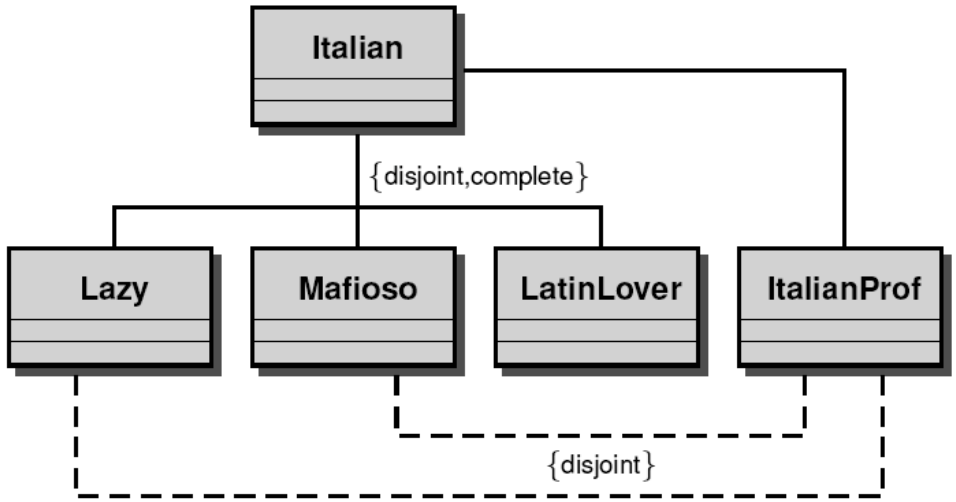
- java -jar sat4j-1.6.jar Exercise1b.cnf returns
"s SATISFIABLE v 1 2 -3" with model "1 and 2 are true, while 3 is false", i.e. $\mathcal{I} = \{a, b\}, \mathcal{I} \models KB \cup \{\neg c\}$
- Hence, $KB \not\models c$
- In fact, $\mathcal{I} \models KB$, but $\mathcal{I} \not\models c$

Example ISA Hierarchy



- Encode the hierarchy and
  - check if “a dog is an animal”
  - check if “a dog is a plant”
  - check if “a dog is not a plant”

Example “Latin Lover”



- Encode the example
  - check if “an ItalianProf is a LatinLover”

Example: Expert System for Automobile Diagnosis

Knowledge base

- “If there is gas in the tank and the fuel line is ok then there is gas in the engine”
- “if there is gas in the engine and the sparks are ok then the engine runs”
- “if power reaches the plugs and the plugs are clean then the sparks are ok”
- “if the battery is charged and the cables are ok then the power reaches the plugs”

Observed

- “the engine does not run”
- “there is gas in the tank, the plugs are clean and the battery is charged”

Prove

- “either the fuel line or the cable line are broken”

Encoding: Expert System for Automobile Diagnosis

Knowledge base

$$(\text{GasInTank} \wedge \text{FuelLineOK}) \implies \text{GasInEngine}$$
$$(\text{GasInEngine} \wedge \text{GoodSpark}) \implies \text{EngineRuns}$$
$$(\text{PowerToPlugs} \wedge \text{PlugsClean}) \implies \text{GoodSpark}$$
$$(\text{BatteryCharged} \wedge \text{CablesOK}) \implies \text{PowerToPlugs}$$

Observed

$$\neg\text{EngineRuns},$$
$$\text{GasInTank}, \text{PlugsClean}, \text{BatteryCharged}$$

Prove

$$(\neg\text{FuelLineOK}) \vee (\neg\text{CablesOK})$$

Example Region Coloring

- Find an assignment of colors to the regions such that no region with common borderline have same color
  - Show that there is no solution with 3 colors
  - Find a solution with 4 colors

Encoding

- Use a variable x_{rc} for each region $r \in \mathcal{R}$ and each color $c \in \mathcal{C}$
- For each region $r \in \mathcal{R}$, consider the clause

$$\bigvee_{c \in \mathcal{C}} x_{rc}$$

  declaring that each region should have a color
- For each region r, for each pair of distinct colors $c_1 \neq c_2$, consider the clause

$$x_{rc_1} \implies \neg x_{rc_2}$$

  declaring that each region should not have two colors
- For each pair of distinct regions $r_1 \neq r_2$, which have a common borderline, and each color c consider the clause

$$x_{r_1 c} \implies \neg x_{r_2 c}$$

  declaring that these two regions should not have the same color

Example Italian Region Coloring

- Using a SAT solver, find an assignment of colors to the regions such that no region with common borderline have same color

The Zebra Puzzle

There are three houses in a row on street. Each house is inhabited by a man of a different nationality, who has a different pet, and drinks a different beverage. The Spaniard own a dog. The Ukranian drinks tea. The man in the third house drinks milk. The Norwegian lives next to the tea drinker. The juice drinker owns a fox. The fox is next door to the dog.

Question: Who owns the zebra?

Possible Encoding

$$
\begin{array}{ccc}
H_1 \mapsto \text{House1} & H_2 \mapsto \text{House2} & H_3 \mapsto \text{House3} \\
N_1 \mapsto \text{Spanier} & N_2 \mapsto \text{Ukrain} & N_3 \mapsto \text{Norvegian} \\
P_1 \mapsto \text{Dog} & P_2 \mapsto \text{Fox} & P_3 \mapsto \text{Zebra} \\
B_1 \mapsto \text{Milk} & B_2 \mapsto \text{Tea} & B_3 \mapsto \text{Juice}
\end{array}
$$

$$
\begin{aligned}
\text{Drinks}N_iB_j &\mapsto \text{“}N_i \text{ drinks } B_j\text{”} \\
\text{Owns}N_iP_j &\mapsto \text{“}N_i \text{ owns } P_j\text{”} \\
\text{Lives}N_iH_j &\mapsto \text{“}N_i \text{ lives } H_j\text{”} \\
\text{Next}H_iH_j &\mapsto \text{“}H_i \text{ is next to } H_j\text{”}
\end{aligned}
$$

- $\text{Next}H_iH_j$:

$$\text{Next}H_1H_2 \wedge \text{Next}H_2H_1$$
$$\text{Next}H_2H_3 \wedge \text{Next}H_3H_2$$
$$\neg\text{Next}H_1H_3 \wedge \neg\text{Next}H_3H_1$$

- $\forall i$

$$\text{Drinks}N_iB_1 \vee \text{Drinks}N_iB_2 \vee \text{Drinks}N_iB_3$$
$$\text{Owns}N_iP_1 \vee \text{Owns}N_iP_2 \vee \text{Owns}N_iP_3$$
$$\text{Lives}N_iH_1 \vee \text{Owns}N_iH_2 \vee \text{Owns}N_iH_3$$

- $\forall i, \forall j_1 \neq j_2$

$$\text{Drinks}N_iB_{j_1} \implies \neg\text{Drinks}N_iB_{j_2}$$
$$\text{Owns}N_iP_{j_1} \implies \neg\text{Owns}N_iP_{j_2}$$
$$\text{Lives}N_iH_{j_1} \implies \neg\text{Owns}N_iH_{j_2}$$

- $\forall j, \forall i_1 \neq i_2$

$$\begin{aligned} \text{DrinksN}_{i_1}\text{B}_j &\implies \neg\text{DrinksN}_{i_2}\text{B}_j \\ \text{OwnsN}_{i_1}\text{P}_j &\implies \neg\text{OwnsN}_{i_2}\text{P}_j \\ \text{LivesN}_{i_1}\text{H}_j &\implies \neg\text{OwnsN}_{i_2}\text{H}_j \end{aligned}$$

- $\text{OwnsN}_1\text{P}_1, \text{DrinksN}_2\text{B}_2$
- $\forall i$

$$\text{LivesN}_i\text{H}_3 \implies \text{DrinksN}_i\text{B}_1$$

- $\forall j, \forall i, \forall k \neq 3, |i - j| = 1$

$$\text{LivesN}_3\text{H}_j \wedge \text{LivesN}_k\text{H}_i \implies \text{DrinksN}_k\text{B}_2$$

- $\forall i$

$$\text{DrinksN}_i\text{B}_3 \implies \text{OwnsN}_i\text{P}_2$$

- $\forall i \neq 1$

$$\text{OwnsN}_i\text{P}_2 \wedge \text{LivesN}_i\text{H}_j \implies \text{LivesN}_1\text{H}_k$$

$$\begin{aligned}
\text{OwnsN}_i\text{P}_2 \wedge \text{LivesN}_i\text{H}_1 &\implies \text{LivesN}_1\text{H}_2 \\
\text{OwnsN}_i\text{P}_2 \wedge \text{LivesN}_i\text{H}_3 &\implies \text{LivesN}_1\text{H}_2 \\
\text{OwnsN}_i\text{P}_2 \wedge \text{LivesN}_i\text{H}_2 &\implies (\text{LivesN}_1\text{H}_1 \vee \text{LivesN}_1\text{H}_3)
\end{aligned}$$