

IA008: Computational Logic

1. Propositional Logic

Achim Blumensath
blumens@fi.muni.cz

Faculty of Informatics, Masaryk University, Brno

Basic Concepts

Propositional Logic

Syntax

- ▶ Variables $A, B, C, \dots, X, Y, Z, \dots$
- ▶ Operators $\wedge, \vee, \neg, \rightarrow, \leftrightarrow$

Semantics

$$\mathfrak{J} \models \varphi \quad \mathfrak{J} : \text{Variables} \rightarrow \{\text{true}, \text{false}\}$$

Examples

$$\varphi := A \wedge (A \rightarrow B) \rightarrow B,$$

$$\psi := \neg(A \wedge B) \leftrightarrow (\neg A \vee \neg B).$$

Terminology

- ▶ **entailment** $\varphi \vDash \psi$ (do not confuse with $\mathfrak{I} \models \varphi !$)
- ▶ **equivalence** $\varphi \equiv \psi$ (do not confuse with $\varphi = \psi !$)
- ▶ $\varphi \equiv \psi$ iff $\varphi \vDash \psi$ and $\psi \vDash \varphi$

Terminology

- ▶ **entailment** $\varphi \vDash \psi$ (do not confuse with $\mathfrak{I} \models \varphi !$)
- ▶ **equivalence** $\varphi \equiv \psi$ (do not confuse with $\varphi = \psi !$)
- ▶ $\varphi \equiv \psi$ iff $\varphi \vDash \psi$ and $\psi \vDash \varphi$
- ▶ **satisfiability** $\varphi \not\equiv \text{false}$
- ▶ **validity** $\varphi \equiv \text{true}$
- ▶ Every valid formula is satisfiable.
- ▶ φ is valid iff $\neg\varphi$ is not satisfiable.
- ▶ $\varphi \vDash \psi$ iff $\varphi \rightarrow \psi$ is valid.

Examples

- ▶ $A \wedge (A \rightarrow B) \rightarrow B$ is

Terminology

- ▶ **entailment** $\varphi \vDash \psi$ (do not confuse with $\mathfrak{I} \models \varphi !$)
- ▶ **equivalence** $\varphi \equiv \psi$ (do not confuse with $\varphi = \psi !$)
- ▶ $\varphi \equiv \psi$ iff $\varphi \vDash \psi$ and $\psi \vDash \varphi$
- ▶ **satisfiability** $\varphi \not\equiv \text{false}$
- ▶ **validity** $\varphi \equiv \text{true}$
- ▶ Every valid formula is satisfiable.
- ▶ φ is valid iff $\neg\varphi$ is not satisfiable.
- ▶ $\varphi \vDash \psi$ iff $\varphi \rightarrow \psi$ is valid.

Examples

- ▶ $A \wedge (A \rightarrow B) \rightarrow B$ is **valid**.
- ▶ $A \vee B$ is

Terminology

- ▶ **entailment** $\varphi \vDash \psi$ (do not confuse with $\mathfrak{I} \models \varphi !$)
- ▶ **equivalence** $\varphi \equiv \psi$ (do not confuse with $\varphi = \psi !$)
- ▶ $\varphi \equiv \psi$ iff $\varphi \vDash \psi$ and $\psi \vDash \varphi$
- ▶ **satisfiability** $\varphi \not\equiv \text{false}$
- ▶ **validity** $\varphi \equiv \text{true}$
- ▶ Every valid formula is satisfiable.
- ▶ φ is valid iff $\neg\varphi$ is not satisfiable.
- ▶ $\varphi \vDash \psi$ iff $\varphi \rightarrow \psi$ is valid.

Examples

- ▶ $A \wedge (A \rightarrow B) \rightarrow B$ is **valid**.
- ▶ $A \vee B$ is **satisfiable** but not **valid**.
- ▶ $\neg A \wedge A$ is

Terminology

- ▶ **entailment** $\varphi \vDash \psi$ (do not confuse with $\mathfrak{I} \models \varphi !$)
- ▶ **equivalence** $\varphi \equiv \psi$ (do not confuse with $\varphi = \psi !$)
- ▶ $\varphi \equiv \psi$ iff $\varphi \vDash \psi$ and $\psi \vDash \varphi$
- ▶ **satisfiability** $\varphi \not\equiv \text{false}$
- ▶ **validity** $\varphi \equiv \text{true}$
- ▶ Every valid formula is satisfiable.
- ▶ φ is valid iff $\neg\varphi$ is not satisfiable.
- ▶ $\varphi \vDash \psi$ iff $\varphi \rightarrow \psi$ is valid.

Examples

- ▶ $A \wedge (A \rightarrow B) \rightarrow B$ is **valid**.
- ▶ $A \vee B$ is **satisfiable** but not **valid**.
- ▶ $\neg A \wedge A$ is **not satisfiable**.

Equivalence Transformations

De Morgan's laws

$$\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi$$

$$\neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi$$

Equivalence Transformations

De Morgan's laws

$$\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi$$

$$\neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi$$

Distributive laws

$$\varphi \wedge (\psi \vee \theta) \equiv (\varphi \wedge \psi) \vee (\varphi \wedge \theta)$$

$$\varphi \vee (\psi \wedge \theta) \equiv (\varphi \vee \psi) \wedge (\varphi \vee \theta)$$

Normal Forms

Conjunctive Normal Form (CNF)

$$(A \vee \neg B) \wedge (\neg A \vee C) \wedge (A \vee \neg B \vee \neg C)$$

Disjunctive Normal Form (DNF)

$$(A \wedge C) \vee (\neg A \wedge \neg B) \vee (A \wedge \neg B \wedge \neg C)$$

Clauses

Definitions

- ▶ **literal** A or $\neg A$
- ▶ **clause** set of literals $\{A, B, \neg C\}$
short-hand for disjunction $A \vee B \vee \neg C$

Clauses

Definitions

- ▶ **literal** A or $\neg A$
- ▶ **clause** set of literals $\{A, B, \neg C\}$
short-hand for disjunction $A \vee B \vee \neg C$

Example

$$\text{CNF} \quad \varphi := (A \vee \neg B \vee C) \wedge (\neg A \vee C) \wedge B$$

clauses $\{A, \neg B, C\}, \{\neg A, C\}, \{B\}$

Clauses

Definitions

- ▶ **literal** A or $\neg A$
- ▶ **clause** set of literals $\{A, B, \neg C\}$
short-hand for disjunction $A \vee B \vee \neg C$

Example

$$\text{CNF} \quad \varphi := (A \vee \neg B \vee C) \wedge (\neg A \vee C) \wedge B$$

clauses $\{A, \neg B, C\}, \{\neg A, C\}, \{B\}$

Notation

$$\Phi[L := \text{true}] := \left\{ C \setminus \{\neg L\} \mid C \in \Phi, L \notin C \right\}.$$

The Satisfiability Problem

Davis-Putnam-Logemann-Loveland (DPLL) Algorithm

Input: a set of clauses Φ

Output: **true** if Φ is satisfiable, **false** otherwise.

DPLL(Φ)

for every singleton $\{L\}$ in Φ (* simplify Φ *)

$\Phi := \Phi[L := \text{true}]$

for every literal L whose negation does not occur in Φ

$\Phi := \Phi[L := \text{true}]$

if Φ contains the empty clause **then** (* are we done? *)

return **false**

if Φ is empty **then**

return **true**

choose some literal L in Φ (* try $L := \text{true}$ and $L := \text{false}$ *)

if $\text{DPLL}(\Phi[L := \text{true}])$ **then**

return **true**

else

return $\text{DPLL}(\Phi[L := \text{false}])$

Example

$$\Phi := \left\{ \{A, B, \neg C\}, \{\neg B, C, D\}, \{\neg A, \neg B, \neg D\}, \{B, C, D\}, \{\neg A, \neg B, \neg C\}, \{\neg A, \neg C, \neg D\} \right\}$$

Step 1: $A := \text{true}$

Example

$$\Phi := \left\{ \{A, B, \neg C\}, \{\neg B, C, D\}, \{\neg A, \neg B, \neg D\}, \{B, C, D\}, \{\neg A, \neg B, \neg C\}, \{\neg A, \neg C, \neg D\} \right\}$$

Step 1: $A := \text{true}$

$$\{\neg B, C, D\}, \{\neg B, \neg D\}, \{B, C, D\}, \{\neg B, \neg C\}, \{\neg C, \neg D\}$$

Step 2: $B := \text{true}$

Example

$$\Phi := \left\{ \{A, B, \neg C\}, \{\neg B, C, D\}, \{\neg A, \neg B, \neg D\}, \{B, C, D\}, \right. \\ \left. \{\neg A, \neg B, \neg C\}, \{\neg A, \neg C, \neg D\} \right\}$$

Step 1: $A := \text{true}$

$$\{\neg B, C, D\}, \{\neg B, \neg D\}, \{B, C, D\}, \{\neg B, \neg C\}, \{\neg C, \neg D\}$$

Step 2: $B := \text{true}$

$$\{C, D\}, \{\neg D\}, \{\neg C\}, \{\neg C, \neg D\}$$

Step 3: $C := \text{false}$ and $D := \text{false}$

Example

$$\Phi := \left\{ \{A, B, \neg C\}, \{\neg B, C, D\}, \{\neg A, \neg B, \neg D\}, \{B, C, D\}, \right. \\ \left. \{\neg A, \neg B, \neg C\}, \{\neg A, \neg C, \neg D\} \right\}$$

Step 1: $A := \text{true}$

$$\{\neg B, C, D\}, \{\neg B, \neg D\}, \{B, C, D\}, \{\neg B, \neg C\}, \{\neg C, \neg D\}$$

Step 2: $B := \text{true}$

$$\{C, D\}, \{\neg D\}, \{\neg C\}, \{\neg C, \neg D\}$$

Step 3: $C := \text{false}$ and $D := \text{false}$

$$\{D\}, \{\neg D\}$$

Example

$$\Phi := \left\{ \{A, B, \neg C\}, \{\neg B, C, D\}, \{\neg A, \neg B, \neg D\}, \{B, C, D\}, \right. \\ \left. \{\neg A, \neg B, \neg C\}, \{\neg A, \neg C, \neg D\} \right\}$$

Step 1: $A := \text{true}$

$$\{\neg B, C, D\}, \{\neg B, \neg D\}, \{B, C, D\}, \{\neg B, \neg C\}, \{\neg C, \neg D\}$$

Step 2: $B := \text{true}$

$$\{C, D\}, \{\neg D\}, \{\neg C\}, \{\neg C, \neg D\}$$

Step 3: $C := \text{false}$ and $D := \text{false}$

$$\{D\}, \{\neg D\}$$

\emptyset **failure**

Example

$$\Phi := \left\{ \{A, B, \neg C\}, \{\neg B, C, D\}, \{\neg A, \neg B, \neg D\}, \{B, C, D\}, \right. \\ \left. \{\neg A, \neg B, \neg C\}, \{\neg A, \neg C, \neg D\} \right\}$$

Step 1: $A := \text{true}$

$$\{\neg B, C, D\}, \{\neg B, \neg D\}, \{B, C, D\}, \{\neg B, \neg C\}, \{\neg C, \neg D\}$$

Backtrack to step 2: $B := \text{false}$

Example

$$\Phi := \left\{ \{A, B, \neg C\}, \{\neg B, C, D\}, \{\neg A, \neg B, \neg D\}, \{B, C, D\}, \right. \\ \left. \{\neg A, \neg B, \neg C\}, \{\neg A, \neg C, \neg D\} \right\}$$

Step 1: $A := \text{true}$

$$\{\neg B, C, D\}, \{\neg B, \neg D\}, \{B, C, D\}, \{\neg B, \neg C\}, \{\neg C, \neg D\}$$

Backtrack to step 2: $B := \text{false}$

$$\{C, D\}, \{\neg C, \neg D\}$$

Step 3: $C := \text{true}$

Example

$$\Phi := \left\{ \{A, B, \neg C\}, \{\neg B, C, D\}, \{\neg A, \neg B, \neg D\}, \{B, C, D\}, \right. \\ \left. \{\neg A, \neg B, \neg C\}, \{\neg A, \neg C, \neg D\} \right\}$$

Step 1: $A := \text{true}$

$$\{\neg B, C, D\}, \{\neg B, \neg D\}, \{B, C, D\}, \{\neg B, \neg C\}, \{\neg C, \neg D\}$$

Backtrack to step 2: $B := \text{false}$

$$\{C, D\}, \{\neg C, \neg D\}$$

Step 3: $C := \text{true}$

$$\{\neg D\} \quad \text{satisfiable}$$

Solution: $A = \text{true}$, $B = \text{false}$, $C = \text{true}$, $D = \text{false}$

Expressing graph problems

Vertex cover

Variables:

C_v vertex v belongs to the cover

Expressing graph problems

Vertex cover

Variables:

C_v vertex v belongs to the cover

Formulae:

$C_u \vee C_v$ for every edge $\langle u, v \rangle \in E$

Size_k^{\leq} "At most k of the C_v are true."

Expressing graph problems

Vertex cover

Variables:

C_v vertex v belongs to the cover

Formulae:

$C_u \vee C_v$ for every edge $\langle u, v \rangle \in E$

Size_k^{\leq} "At most k of the C_v are true."

Clique

Variables:

C_v vertex v belongs to the clique

Expressing graph problems

Vertex cover

Variables:

C_v vertex v belongs to the cover

Formulae:

$C_u \vee C_v$ for every edge $\langle u, v \rangle \in E$

Size_k^{\leq} "At most k of the C_v are true."

Clique

Variables:

C_v vertex v belongs to the clique

Formulae:

$\neg C_u \vee \neg C_v$ for every non-edge $\langle u, v \rangle \notin E$

Size_k^{\geq} "At least k of the C_v are true."

Expressing graph problems

The Size_k^{\geq} formulae

Fix an enumeration v_0, \dots, v_{n-1} of V .

Variables:

S_m^k at least k variables C_{v_i} with $i < m$ are true

Expressing graph problems

The Size_k^{\geq} formulae

Fix an enumeration v_0, \dots, v_{n-1} of V .

Variables:

S_m^k at least k variables C_{v_i} with $i < m$ are true

Formulae:

S_m^0

$\neg S_0^k$ for $k > 0$

$C_{v_i} \rightarrow [S_i^k \leftrightarrow S_{i+1}^{k+1}]$

$\neg C_{v_i} \rightarrow [S_i^k \leftrightarrow S_{i+1}^k]$

S_n^k

Expressing graph problems

The Size_k^{\geq} formulae

Fix an enumeration v_0, \dots, v_{n-1} of V .

Variables:

S_m^k at least k variables C_{v_i} with $i < m$ are true

Formulae:

S_m^0

$\neg S_0^k$ for $k > 0$

$C_{v_i} \rightarrow [S_i^k \leftrightarrow S_{i+1}^{k+1}]$

$\neg C_{v_i} \rightarrow [S_i^k \leftrightarrow S_{i+1}^k]$

S_n^k

	v_0	v_1	v_2	
C_{v_i}	1	0	1	
S_i^0	1	1	1	1
S_i^1	0	1	1	1
S_i^2	0	0	0	1
S_i^3	0	0	0	0

Expressing graph problems

The Size_k^{\geq} formulae

Fix an enumeration v_0, \dots, v_{n-1} of V .

Variables:

S_m^k at least k variables C_{v_i} with $i < m$ are true

Formulae:

S_m^0

$\neg S_0^k$ for $k > 0$

$C_{v_i} \rightarrow [S_i^k \leftrightarrow S_{i+1}^{k+1}]$

$\neg C_{v_i} \rightarrow [S_i^k \leftrightarrow S_{i+1}^k]$

S_n^k

	v_0	v_1	v_2	
C_{v_i}	1	0	1	
S_i^0	1	1	1	1
S_i^1	0	1	1	1
S_i^2	0	0	0	1
S_i^3	0	0	0	0

A similar construction works for Size_k^{\leq} .

The Satisfiability Problem

Theorem

\exists -SAT (satisfiability for formulae in \exists -CNF) is NP-complete.

The Satisfiability Problem

Theorem

\exists -SAT (satisfiability for formulae in \exists -CNF) is NP-complete.

Proof

Given Turing machine M and input w , construct formula φ such that

M accepts w iff φ is satisfiable.

Proof

Turing machine $\mathcal{M} = \langle Q, \Sigma, \Delta, q_0, F_+, F_- \rangle$

Q set of states

Σ tape alphabet

Δ set of transitions $\langle p, a, b, m, q \rangle \in Q \times \Sigma \times \Sigma \times \{-1, 0, 1\} \times Q$

q_0 initial state

F_+ accepting states

F_- rejecting states

nondeterministic, runtime bounded by the polynomial $r(n)$

Proof

Turing machine $\mathcal{M} = \langle Q, \Sigma, \Delta, q_0, F_+, F_- \rangle$

Q set of states

Σ tape alphabet

Δ set of transitions $\langle p, a, b, m, q \rangle \in Q \times \Sigma \times \Sigma \times \{-1, 0, 1\} \times Q$

q_0 initial state

F_+ accepting states

F_- rejecting states

nondeterministic, runtime bounded by the polynomial $r(n)$

Encoding in PL

$S_{t,q}$ state q at time t

$H_{t,k}$ head in field k at time t

$W_{t,k,a}$ letter a in field k at time t

$$\varphi_w := \bigwedge_{t < r(n)} [\text{ADM}_t \wedge \text{INIT} \wedge \text{TRANS}_t \wedge \text{ACC}]$$

Proof

$S_{t,q}$ state q at time t

$H_{t,k}$ head in field k at time t

$W_{t,k,a}$ letter a in field k at time t

Admissibility formula

$$\text{ADM}_t := \bigwedge_{p \neq q} [\neg S_{t,p} \vee \neg S_{t,q}] \quad \text{unique state}$$

$$\wedge \bigwedge_{k \neq l} [\neg H_{t,k} \vee \neg H_{t,l}] \quad \text{unique head position}$$

$$\wedge \bigwedge_k \bigwedge_{a \neq b} [\neg W_{t,k,a} \vee \neg W_{t,k,b}] \quad \text{unique letter}$$

Proof

$S_{t,q}$ state q at time t

$H_{t,k}$ head in field k at time t

$W_{t,k,a}$ letter a in field k at time t

Initialisation formula for input: $a_0 \dots a_{n-1}$

$$\begin{aligned} \text{INIT} := & S_{0,q_0} && \text{initial state} \\ & \wedge H_{0,0} && \text{initial head position} \\ & \wedge \bigwedge_{k < n} W_{0,k,a_k} \wedge \bigwedge_{n \leq k \leq r(n)} W_{0,k,\square} && \text{initial tape content} \end{aligned}$$

Acceptance formula

$$\text{ACC} := \bigvee_{q \in F_+} \bigvee_{t \leq r(n)} S_{t,q} \quad \text{accepting state}$$

Proof

$S_{t,q}$ state q at time t

$H_{t,k}$ head in field k at time t

$W_{t,k,a}$ letter a in field k at time t

Transition formula

$$\text{TRANS}_t := \bigvee_{\langle p, a, b, m, q \rangle \in \Delta} \bigvee_{k \leq r(n)} [S_{t,p} \wedge H_{t,k} \wedge W_{t,k,a} \wedge S_{t+1,q} \wedge H_{t+1,k+m} \wedge W_{t+1,k,b}]$$

effect of transition

$$\wedge \bigwedge_{k \leq r(n)} \bigwedge_{a \in \Sigma} [\neg H_{t,k} \wedge W_{t,k,a} \rightarrow W_{t+1,k,a}]$$

rest of tape remains unchanged

Proof

$$\text{TRANS}_t := \bigvee_{\langle p, a, b, m, q \rangle \in \Delta} \bigvee_{k \leq r(n)} [S_{t,p} \wedge H_{t,k} \wedge W_{t,k,a} \wedge S_{t+1,q} \wedge H_{t+1,k+m} \wedge W_{t+1,k,b}] \wedge \dots$$

Proof

$$\text{TRANS}_t := \bigvee_{\langle p, a, b, m, q \rangle \in \Delta} \bigvee_{k \leq r(n)} [S_{t,p} \wedge H_{t,k} \wedge W_{t,k,a} \wedge S_{t+1,q} \wedge H_{t+1,k+m} \wedge W_{t+1,k,b}] \wedge \dots$$

equivalently:

$$\bigwedge_{k \leq r(n)} \bigwedge_{p \in Q} \bigwedge_{a \in \Sigma} \left[S_{t,p} \wedge H_{t,k} \wedge W_{t,k,a} \rightarrow \bigvee_{q \in TS(p,a)} S_{t+1,q} \right]$$

$$TS(p, a) := \{ q \in Q \mid \langle p, a, b, m, q \rangle \in \Delta \}$$

Proof

$$\text{TRANS}_t := \bigvee_{\langle p, a, b, m, q \rangle \in \Delta} \bigvee_{k \leq r(n)} [S_{t,p} \wedge H_{t,k} \wedge W_{t,k,a} \wedge S_{t+1,q} \wedge H_{t+1,k+m} \wedge W_{t+1,k,b}] \wedge \dots$$

equivalently:

$$\begin{aligned} & \bigwedge_{k \leq r(n)} \bigwedge_{p \in Q} \bigwedge_{a \in \Sigma} \left[S_{t,p} \wedge H_{t,k} \wedge W_{t,k,a} \rightarrow \bigvee_{q \in TS(p,a)} S_{t+1,q} \right] \\ & \wedge \bigwedge_{k \leq r(n)} \bigwedge_{p,q \in Q} \bigwedge_{a \in \Sigma} \left[S_{t,p} \wedge H_{t,k} \wedge W_{t,k,a} \wedge S_{t+1,q} \rightarrow \bigvee_{m \in TH(p,a,q)} H_{t+1,k+m} \right] \end{aligned}$$

$$TH(p, a, q) := \{ m \mid \langle p, a, b, m, q \rangle \in \Delta \}$$

Proof

$$\text{TRANS}_t := \bigvee_{\langle p, a, b, m, q \rangle \in \Delta} \bigvee_{k \leq r(n)} [S_{t,p} \wedge H_{t,k} \wedge W_{t,k,a} \wedge S_{t+1,q} \wedge H_{t+1,k+m} \wedge W_{t+1,k,b}] \wedge \dots$$

equivalently:

$$\begin{aligned} & \bigwedge_{k \leq r(n)} \bigwedge_{p \in Q} \bigwedge_{a \in \Sigma} \left[S_{t,p} \wedge H_{t,k} \wedge W_{t,k,a} \rightarrow \bigvee_{q \in TS(p,a)} S_{t+1,q} \right] \\ & \wedge \bigwedge_{k \leq r(n)} \bigwedge_{p,q \in Q} \bigwedge_{a \in \Sigma} \left[S_{t,p} \wedge H_{t,k} \wedge W_{t,k,a} \wedge S_{t+1,q} \rightarrow \bigvee_{m \in TH(p,a,q)} H_{t+1,k+m} \right] \\ & \wedge \bigwedge_{k \leq r(n)} \bigwedge_{p,q \in Q} \bigwedge_{a \in \Sigma} \bigwedge_{m \in \{-1, 0, 1\}} \left[S_{t,p} \wedge H_{t,k} \wedge W_{t,k,a} \wedge S_{t+1,q} \wedge H_{t+1,k+m} \rightarrow \right. \\ & \quad \left. \bigvee_{b \in TW(p,a,m,q)} W_{t+1,k,b} \right] \end{aligned}$$

$$TW(p, a, m, q) := \{ b \in Q \mid \langle p, a, b, m, q \rangle \in \Delta \}$$

Proof

Properties of φ_w

- ▶ It is in CNF.
- ▶ It has length $\sim r(n)^3$.
- ▶ It is satisfiable if, and only if, the Turing machine accepts w .

Consequently, the satisfiability problem for PL-formulae in CNF is NP-complete.

Proof

Properties of φ_w

- ▶ It is in CNF.
- ▶ It has length $\sim r(n)^3$.
- ▶ It is satisfiable if, and only if, the Turing machine accepts w .

Consequently, the satisfiability problem for PL-formulae in CNF is NP-complete.

Reduction to 3-CNF

$$\{L_0, L_1, L_2, \dots, L_n\} \quad \mapsto \quad \{L_0, L_1, X\}, \quad \{\neg X, L_2, \dots, L_n\}$$

(X new variable)

Resolution

Resolution

Resolution Step

The **resolvent** of two clauses

$$C = \{L, A_0, \dots, A_m\} \quad \text{and} \quad C' = \{\neg L, B_0, \dots, B_n\}$$

is the clause

$$\{A_0, \dots, A_m, B_0, \dots, B_n\}.$$

Lemma

Let C be the resolvent of two clauses in Φ . Then

$$\Phi \models \Phi \cup \{C\}.$$

Resolution

Resolution Step

The **resolvent** of two clauses

$$C = \{L, A_0, \dots, A_m\} \quad \text{and} \quad C' = \{\neg L, B_0, \dots, B_n\}$$

is the clause

$$\{A_0, \dots, A_m, B_0, \dots, B_n\}.$$

(This is the inverse of the splitting trick from the last slide.)

Lemma

Let C be the resolvent of two clauses in Φ . Then

$$\Phi \vDash \Phi \cup \{C\}.$$

The Resolution Method

Observation

If Φ contains the empty clause \emptyset , then Φ is not satisfiable.

Resolution Method

Input: a set of clauses Φ

Output: **true** if Φ is satisfiable, **false** otherwise.

$RM(\Phi)$

add to Φ all possible resolvents

repeat until no new clauses are generated

if $\emptyset \in \Phi$ **then**

return **false**

else

return **true**

Theorem

The resolution method for propositional logic is **sound** and **complete**.

Example

$\{A, C\}$

$\{B, \neg C\}$

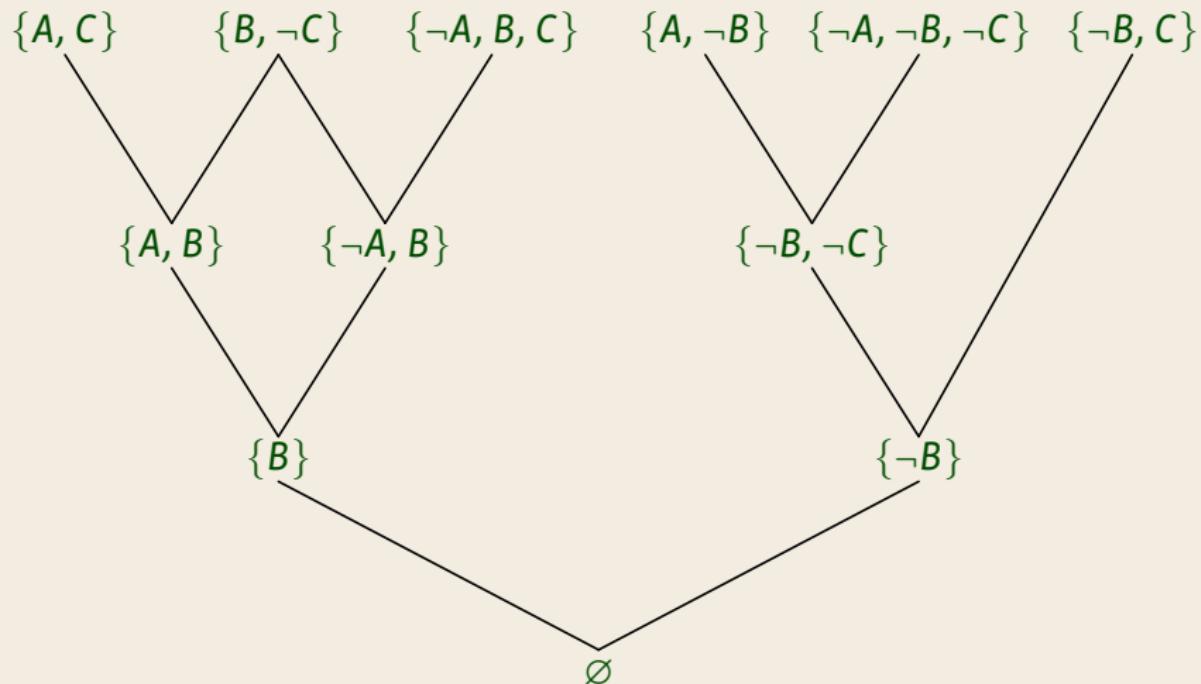
$\{\neg A, B, C\}$

$\{A, \neg B\}$

$\{\neg A, \neg B, \neg C\}$

$\{\neg B, C\}$

Example



Davis-Putnam Algorithm

Input: a set of clauses Φ

Output: **true** if Φ is satisfiable, **false** otherwise.

DP(Φ)

remove all tautological clauses from Φ

if $\Phi = \emptyset$ **then**

return **true**

if $\Phi = \{\emptyset\}$ **then**

return **false**

select a variable X

add to Φ all resolvents over X

remove all clauses containing X or $\neg X$ from Φ

repeat

Example

$\{A, C\}$ $\{B, \neg C\}$ $\{\neg A, B, C\}$ $\{A, \neg B\}$ $\{\neg A, \neg B, \neg C\}$ $\{\neg B, C\}$

Example

$\{A, C\}$ $\{B, \neg C\}$ $\{\neg A, B, C\}$ $\{A, \neg B\}$ $\{\neg A, \neg B, \neg C\}$ $\{\neg B, C\}$

select A : $\{B, C\}$ $\{\neg B, C, \neg C\}$ $\{B, \neg B, C\}$ $\{\neg B, \neg C\}$

Example

$\{A, C\} \{B, \neg C\} \{\neg A, B, C\} \{A, \neg B\} \{\neg A, \neg B, \neg C\} \{\neg B, C\}$

select A : $\{B, C\} \{\neg B, C, \neg C\} \{B, \neg B, C\} \{\neg B, \neg C\}$

removals: $\{B, \neg C\} \{\neg B, C\} \{B, C\} \{\neg B, \neg C\}$

Example

$\{A, C\} \{B, \neg C\} \{\neg A, B, C\} \{A, \neg B\} \{\neg A, \neg B, \neg C\} \{\neg B, C\}$

select A : $\{B, C\} \{\neg B, C, \neg C\} \{B, \neg B, C\} \{\neg B, \neg C\}$

removals: $\{B, \neg C\} \{\neg B, C\} \{B, C\} \{\neg B, \neg C\}$

select B : $\{C, \neg C\} \{\neg C\} \{C\} \{C, \neg C\}$

Example

$\{A, C\} \{B, \neg C\} \{\neg A, B, C\} \{A, \neg B\} \{\neg A, \neg B, \neg C\} \{\neg B, C\}$

select A : $\{B, C\} \{\neg B, C, \neg C\} \{B, \neg B, C\} \{\neg B, \neg C\}$

removals: $\{B, \neg C\} \{\neg B, C\} \{B, C\} \{\neg B, \neg C\}$

select B : $\{C, \neg C\} \{\neg C\} \{C\} \{C, \neg C\}$

removals: $\{\neg C\} \{C\}$

Example

$\{A, C\} \{B, \neg C\} \{\neg A, B, C\} \{A, \neg B\} \{\neg A, \neg B, \neg C\} \{\neg B, C\}$

select A : $\{B, C\} \{\neg B, C, \neg C\} \{B, \neg B, C\} \{\neg B, \neg C\}$

removals: $\{B, \neg C\} \{\neg B, C\} \{B, C\} \{\neg B, \neg C\}$

select B : $\{C, \neg C\} \{\neg C\} \{C\} \{C, \neg C\}$

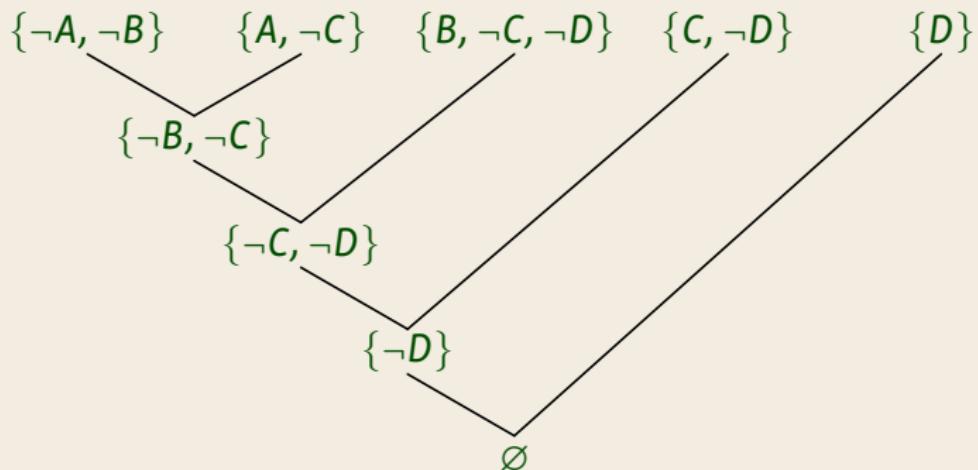
removals: $\{\neg C\} \{C\}$

select C : \emptyset

Horn formulae

Linear Resolution

A **linear resolution** is a sequence of resolution steps where in each step the resolvent of the previous step is used.



Horn formulae and linear resolution

Horn formulae

A **Horn clause** is a clause C that contains at most one positive literal.

Example

$$A_0 \wedge \cdots \wedge A_n \rightarrow B \quad \equiv \quad \{\neg A_0, \dots, \neg A_n, B\}$$

Horn formulae and linear resolution

Horn formulae

A **Horn clause** is a clause C that contains at most one positive literal.

Example

$$A_0 \wedge \cdots \wedge A_n \rightarrow B \quad \equiv \quad \{\neg A_0, \dots, \neg A_n, B\}$$

Theorem

A set of Horn clauses is unsatisfiable if, and only if, one can use linear resolution to derive the empty clause from it.

SLD Resolution

Linear resolution where the clauses are **sequences** instead of sets and we always resolve the **leftmost literal** of the current clause.

Minimal models

Lemma

Every satisfiable set of Horn-formulae has a minimal model.

Minimal models

Lemma

Every satisfiable set of Horn-formulae has a minimal model.

Algorithm to compute it:

Input: Φ set of Horn-formulae

$T := \emptyset$

repeat

for all $A_0 \wedge \dots \wedge A_{n-1} \rightarrow B \in \Phi$ **do**

if $A_0, \dots, A_{n-1} \in T$ **then**

$T := T \cup \{B\}$

until T does not change anymore

Minimal models

Lemma

Every satisfiable set of Horn-formulae has a minimal model.

Algorithm to compute it:

Input: Φ set of Horn-formulae

$T := \emptyset$

repeat

for all $A_0 \wedge \dots \wedge A_{n-1} \rightarrow B \in \Phi$ **do**

if $A_0, \dots, A_{n-1} \in T$ **then**

$T := T \cup \{B\}$

until T does not change anymore

Theorem

Satisfiability for sets of Horn-formulae can be checked in linear time.

Example

$$B \wedge C \rightarrow A$$

$$A \wedge D \rightarrow B$$

$$F \rightarrow C$$

$$E \rightarrow D$$

$$D \wedge E \rightarrow A$$

$$C \wedge F \rightarrow B$$

$$1 \rightarrow F$$

Example

$$B \wedge C \rightarrow A$$

$$A \wedge D \rightarrow B$$

$$\textcolor{red}{F} \rightarrow C$$

$$E \rightarrow D$$

$$D \wedge E \rightarrow A$$

$$C \wedge \textcolor{red}{F} \rightarrow B$$

$$1 \rightarrow \textcolor{red}{F}$$

Example

$$B \wedge \textcolor{red}{C} \rightarrow A$$

$$A \wedge D \rightarrow B$$

$$\textcolor{red}{F} \rightarrow \textcolor{red}{C}$$

$$E \rightarrow D$$

$$D \wedge E \rightarrow A$$

$$\textcolor{red}{C} \wedge \textcolor{red}{F} \rightarrow B$$

$$1 \rightarrow \textcolor{red}{F}$$

Example

$$B \wedge C \rightarrow A$$

$$A \wedge D \rightarrow B$$

$$F \rightarrow C$$

$$E \rightarrow D$$

$$D \wedge E \rightarrow A$$

$$C \wedge F \rightarrow B$$

$$1 \rightarrow F$$

Example

$$B \wedge C \rightarrow A$$

$$A \wedge D \rightarrow B$$

$$F \rightarrow C$$

$$E \rightarrow D$$

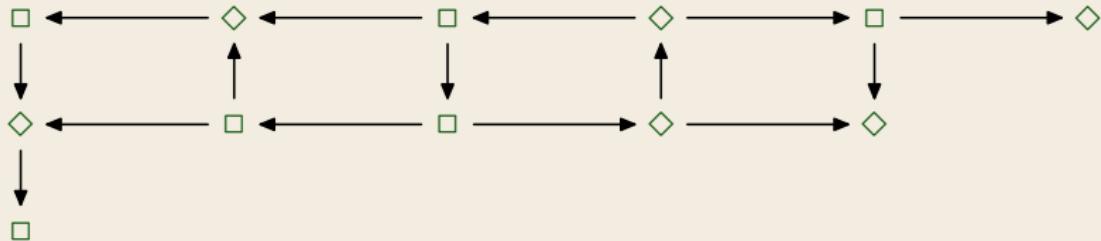
$$D \wedge E \rightarrow A$$

$$C \wedge F \rightarrow B$$

$$1 \rightarrow F$$

Finite Games $\mathcal{G} = \langle V_{\diamond}, V_{\square}, E \rangle$

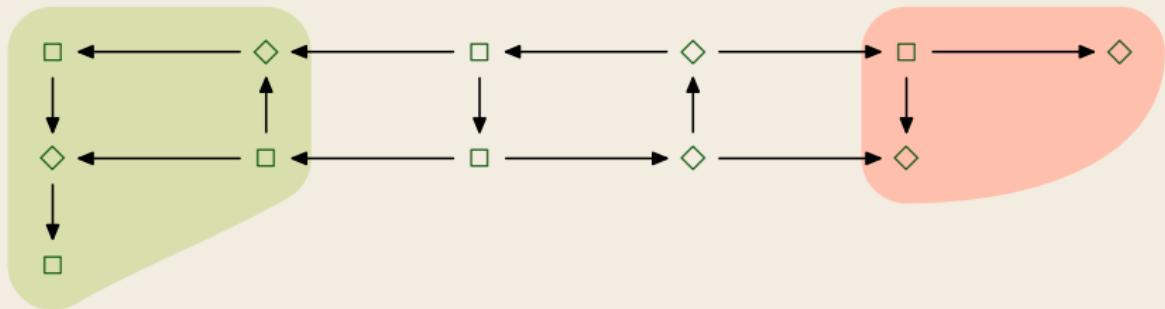
Players \diamond and \square



Winning regions: $W_{\diamond}, W_{\square}$

Finite Games $\mathcal{G} = \langle V_{\diamond}, V_{\square}, E \rangle$

Players \diamond and \square



Winning regions: $W_{\diamond}, W_{\square}$

Reduction

positions

$V_{\diamond} = \text{variables } \langle A \rangle \quad \text{and} \quad V_{\square} = \text{formulae } [A_0 \wedge \dots \wedge A_{n-1} \rightarrow B]$

edges

$$\begin{array}{ccc} \langle B \rangle & \rightarrow & [A_0 \wedge \dots \wedge A_{n-1} \rightarrow B] \\ [A_0 \wedge \dots \wedge A_{n-1} \rightarrow B] & \rightarrow & \langle A_i \rangle \end{array}$$

Lemma

A variable A belongs to V_{\diamond} iff it is true in the minimal model.

$$B \wedge C \rightarrow A$$

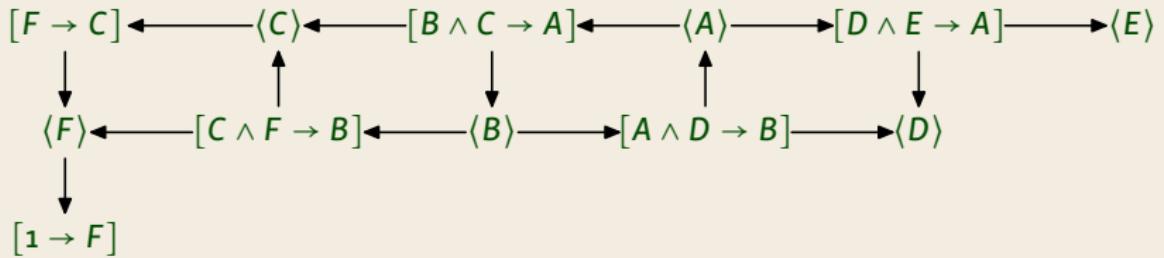
$$A \wedge D \rightarrow B$$

$$F \rightarrow C$$

$$D \wedge E \rightarrow A$$

$$C \wedge F \rightarrow B$$

$$1 \rightarrow F$$



Simple Algorithm

$\text{Win}(v, \sigma)$

```
if  $v \in V_\sigma$  then
    if there is an edge  $v \rightarrow u$  with  $\text{Win}(u, \sigma)$  then
        return true
    else
        return false

if  $v \in V_{\bar{\sigma}}$  then
    if for every edge  $v \rightarrow u$  we have  $\text{Win}(u, \sigma)$  then
        return true
    else
        return false
```

(* $\overline{\Diamond} := \square$ $\overline{\square} := \Diamond$ *)

Linear Algorithm

Input: game $\langle V_{\diamond}, V_{\square}, E \rangle$

forall $v \in V$ **do**

$win[v] := \perp$ (* winner of the position *)

$P[v] := \emptyset$ (* set of predecessors of v *)

$n[v] := 0$ (* number of successors of v *)

end

forall $\langle u, v \rangle \in E$ **do**

$P[v] := P[v] \cup \{u\}$

$n[u] := n[u] + 1$

end

forall $v \in V_{\diamond}$ **do**

if $n[v] = 0$ **then** Propagate(v, \square)

forall $v \in V_{\square}$ **do**

if $n[v] = 0$ **then** Propagate(v, \diamond)

return win

```
procedure Propagate( $v, \sigma$ ) =  
  if  $\text{win}[v] \neq \perp$  then return  
   $\text{win}[v] := \sigma$   
  forall  $u \in P[v]$  do  
     $n[u] := n[u] - 1$   
    if  $u \in V_\sigma$  or  $n[u] = 0$  then Propagate( $u, \sigma$ )  
  end  
end
```