Logic and Discrete Structures -LDS

Course 10 – Propositional Logic

S. l. dr. ing. Cătălin Iapă

catalin.iapa@cs.upt.ro

From the last time:

Logic - general notions
Propositional Logic
Syntax
Semantics
Binary Decision Diagrams
Conjunctive normal form

Syntax of propositional logic

A language is defined by
its symbols
and the rules by which we correctly combine symbols (syntax)

Symbols of propositional logic:
sentences: usually denoted by the letters p, q, r , etc.
operators (logical connectors): negation ¬, implication → ,
parentheses ( )

Propositional logic formulas: defined by structural induction
(construct complex formulas from simpler ones)
A formula is:
any proposition (also called atomic formula)
$(\neg \alpha)$ if α is a formula
$(\alpha \rightarrow \beta)$ if α and β are formulas (α, β called subformulas)

Other logical operators (connectors)

We usually give minimal definitions (as few cases as possible) (any further reasoning must be done on all cases)

Known operators can be defined using ¬ and →:

$\alpha \wedge \beta \stackrel{def}{=} \neg(\alpha \rightarrow \neg\beta)$ (AND)

$\alpha \vee \beta \stackrel{def}{=} \neg\alpha \rightarrow \beta$ (OR)

$\alpha \leftrightarrow \beta \stackrel{def}{=} (\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$ (equivalence)

We skip redundant parentheses, defining operator precedence.

Order of precedence: ¬, ∧, ∨, →, ↔

The implication is associative to the right! $p \rightarrow q \rightarrow r = p \rightarrow (q \rightarrow r)$

The syntax does not define what a formula means. We will define the semantics later.

Semantics of a formula: truth functions

We define rigorously how we calculate the truth value of a formula = we give a semantics (meaning) to the formula (formula = syntactic notion)

A truth function v assigns to any formula

a truth value $\in \{T, F\}$ such that:

v (p) is defined for each atomic proposition p.

$$v(\neg\alpha) = \begin{cases} T & \text{if } v(\alpha) = F \quad F \\ & \text{if } v(\alpha) = T \end{cases}$$

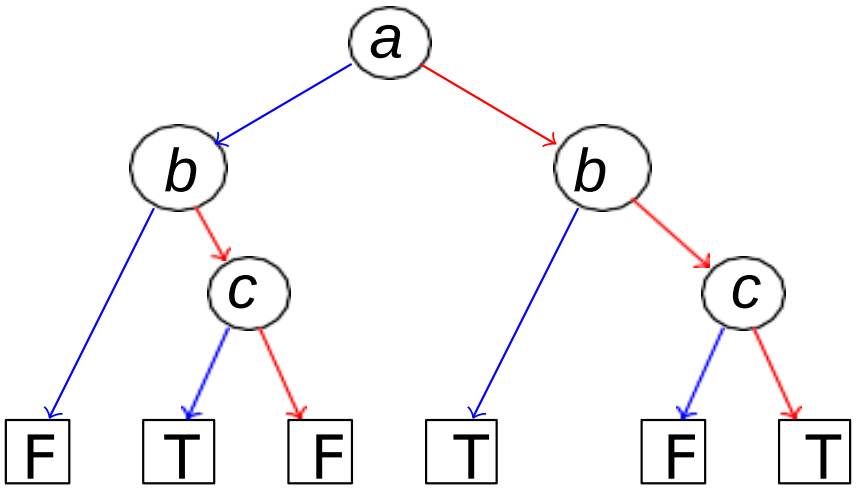
$$v(\alpha \rightarrow \beta) = \begin{cases} F & \text{if } v(\alpha) = T \text{ and } v(\beta) = F \\ T & \text{otherwise} \end{cases}$$

Binary decision tree

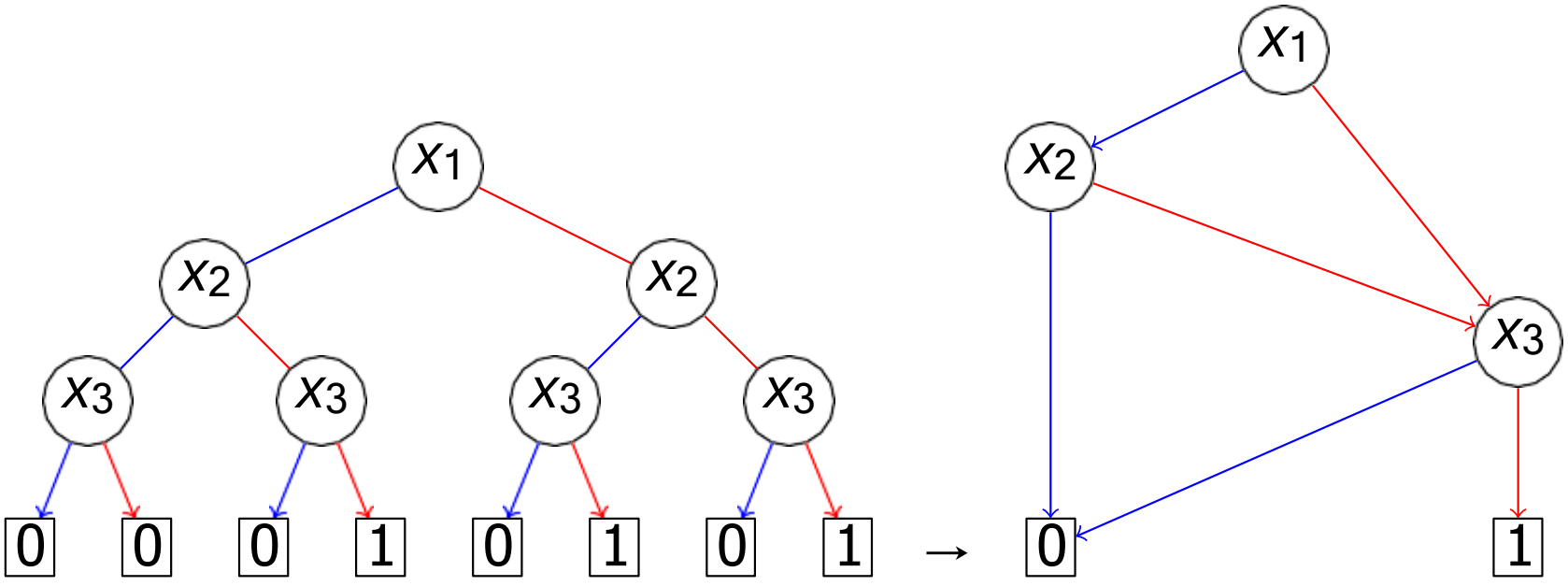
$$f = (a \vee b) \wedge (a \vee \neg c) \wedge (\neg a \vee \neg b \vee c)$$

$$f|_{a=T} = T \wedge T \wedge (\neg b \vee c) = \neg b \vee c$$

$$f|_{a=F} = b \wedge \neg c \wedge T = b \wedge \neg c$$



From tree to binary decision diagram



binary decision tree

binary decision diagram

Conjunctive normal form

Conjunctive normal form used to determine whether a formula is satisfiable (can be T)

Def: *Conjunctive normal form*
= *conjunction* $\wedge$ of clauses
clause = *disjunction* $\vee$ of *literals*
literal = sentance or its negation
(p or $\neg p$)

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

Example: conjunctive normal form

1) we take negations inside to sentences *de Morgan*

2) we replace the implications from the outside when we get to them

$p \to q = \neg p \vee q$

3) we take the disjunction $\vee$ inside the conjunction $\wedge$

distributivity

Example:

$$\begin{aligned} & \neg((a \wedge b) \vee ((a \to (b \wedge c)) \to c)) \\ = \quad & \neg(a \wedge b) \wedge \neg((a \to (b \wedge c)) \to c)) \\ = \quad & (\neg a \vee \neg b) \wedge ((a \to (b \wedge c)) \wedge \neg c) \\ = \quad & (\neg a \vee \neg b) \wedge (\neg a \vee (b \wedge c)) \wedge \neg c \\ = \quad & (\neg a \vee \neg b) \wedge (\neg a \vee b) \wedge (\neg a \vee c) \wedge \neg c \end{aligned}$$

In today's course

How do we determine if a formula is satisfiable?

- algorithm used in solving many problems

What is a logical proof?

The satisfiability of a formula in propositional logic (SAT-problem)

Proof vs logical consequence

The satisfiability of a formula in propositional logic

A formula is given in propositional logic.

Is there any truth-value assignment that makes it true?

= Is the formula satisfiable?

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

Find an attribution that satisfies the formula?

The formula is in conjunctive normal form

Rules in determining satisfiability

We simplify the problem, knowing that we want the true formula.
(does NOT apply to simplifying formulas into equivalent formulas!)

R1) A single literal in a clause has only one useful value:

in $a \wedge (\neg a \vee b \vee c) \wedge (\neg a \vee \neg b \vee \neg c)$ a must be T

in $(a \vee b) \wedge \neg b \wedge (\neg a \vee \neg b \vee c)$ b must be F

(otherwise the formula has the value F)

Rules in determining satisfiability

R2a) If a literal is T, the clauses in which it appears can be deleted (they are true, we have solved them)

R2b) If a literal is F, it can be deleted from the clauses in which it appears (cannot make the clause true)

The previous examples simplify:

$$a \wedge (\neg a \vee b \vee c) \wedge (\neg a \vee \neg b \vee \neg c) \xrightarrow{a=\mathsf{T}} (b \vee c) \wedge (\neg b \vee \neg c)$$

$$(a \vee b) \wedge \neg b \wedge (\neg a \vee \neg b \vee c) \xrightarrow{b=\mathsf{F}} a$$

(and here can be a = T , so the formula is feasible)

Rules in determining satisfiability

R3) If there are no more clauses, the formula is satisfiable
(with constructed assignment)

If we get an empty clause, the formula is not satisfiable
(being emptyclause, we cannot make it T)

$(a \vee b) \wedge a \wedge (a \vee \neg b \vee c) \overset{a=T}{\rightarrow} (T \vee b) \wedge T \wedge (T \vee \neg b \vee c) \overset{R2a}{\rightarrow}$
delete all clauses (contain T, we have solved them)
$\Rightarrow$ satisfiable formula (with a = T)

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

$\overset{a=T}{\rightarrow}$ $b \wedge (\neg b \vee c) \wedge (\neg b \vee \neg c)$

$\overset{b=T}{\rightarrow}$ $c \wedge \neg c$ $\overset{c=T}{\rightarrow}$ $\varnothing$(¬c becomes empty clause $\Rightarrow$ not satisfiable)

Rules in determining satisfiability

What if we can't make reductions according to these rules?

$a \wedge (\neg a \vee b \vee c) \wedge (\neg b \vee \neg c) \xrightarrow{a=\mathsf{T}} (b \vee c) \wedge (\neg b \vee \neg c)$?

R4) We choose a proposition and split by cases (try):

- with value F
- with value T

A solution for any case is good (we are not looking for a specific solution).

If no case has a solution, the formula is not satisfiable.

The satisfiability of a formula in propositional logic (SAT-problem)

Proof vs logical consequence

Syntax and semantics

For propositional logic, we have discussed:

Syntax: a formula has form:
proposition or $(\neg$ *formula*$)$ or $($*formula* $\rightarrow$ *formula*$)$

Semantics: we calculate the truth value (meaning) of the formula from the truth value (meaning) of the sentences

$$v(\neg\alpha) = \begin{cases} \text{T} & \text{if } v(\alpha) = \text{F} \quad \text{F} \\ & \text{if } v(\alpha) = \text{T} \end{cases}$$

$$v(\alpha \rightarrow \beta) = \begin{cases} \text{F} & \text{if } v(\alpha) = \text{T and } v(\beta) = \text{F} \quad \text{T} \\ & \text{else} \end{cases}$$

Logical inferences

Inference allows us to prove a formula syntactically
(using only its structure)

It is based on a rule of inference (deduction)

$$\frac{A \qquad A \rightarrow B}{B}$$ *modus ponens*

(from A and $A \rightarrow B$ we deduce (infer) B; A, B whatever formula)
and a set of axioms (formulas that can be used as premises)

A1: $\alpha \rightarrow (\beta \rightarrow \alpha)$
A2: $(\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow ((\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \gamma))$
A3: $(\neg\beta \rightarrow \neg\alpha) \rightarrow (\alpha \rightarrow \beta)$
where α, β etc. can be replaced by any formula

*) A1 - A3 are tautologies

Deduction (demonstration)

Informally, a deduction (demonstration) is a series of statements in which each follows from (can be derived from) the previous ones.
Rigorously, we define:
Let H be a set of formulas (hypotheses).

A deduction (proof) from H is a string of formulas A1, A2,... , An, for $\forall i \in [1, n]$

1. A_i is an axiom, or
2. A_i is a hypothesis (a formula from H), or
3. A_i follows by modus ponens from A_j, A_k prior ($j, k < i$)

We say that A_n *follows from H (it's deductible, it's a consequence)*.
Denoted: $H \vdash A_n$

Example of inference (deduction)

We prove that A $\rightarrow$ A for any formula A:

(1) $A \rightarrow ((A \rightarrow A) \rightarrow A))$ A1 cu $\alpha = A$, $\beta = A \rightarrow A$

(2) $A \rightarrow ((A \rightarrow A) \rightarrow A)) \rightarrow ((A \rightarrow (A \rightarrow A)) \rightarrow (A \rightarrow A))$ A2 cu $\alpha = \gamma = A$, $\beta = A \rightarrow A$

(3) $(A \rightarrow (A \rightarrow A)) \rightarrow (A \rightarrow A)$ MP(1,2)

(4) $A \rightarrow (A \rightarrow A)$ A1 cu $\alpha = \beta = A$

(5) $A \rightarrow A$ MP(3,4)

Checking a proof is a simple, mechanical process (we check the reason given for each statement; a simple comparison of strings of symbols).

Finding a proof is a more difficult process.

Other deduction (inference) rules

Modus ponens is sufficient to formalize propositional logic but there are other rules of deduction that simplify the proofs

$$\frac{p \rightarrow q \qquad \neg q}{\neg p}$$ *modus tollens (indirect proof)*

$$\frac{p}{p \vee q}$$ *generalisation (introduction of disjunction)*

$$\frac{p \wedge q}{p}$$ *specialisation (simplification)*

$$\frac{p \vee q \qquad \neg p}{q}$$ *eliminate (disjunctive syllogism)*

$$\frac{p \rightarrow q \qquad q \rightarrow r}{p \rightarrow r}$$ *transitivity (hypothetical syllogism)*

Deduction (example)

Let $H = \{a,\ \neg b \vee d,\ a \rightarrow (b \wedge c),\ (c \wedge d\ ) \rightarrow (\neg a \vee e)\}$.
Show that $H \vdash e$.

(1) a	hypothesis, H_1
(2) $a \rightarrow (b \wedge c)$	hypothesis, H_3
(3) $b \wedge c$	modus ponens (1, 2)
(4) b	specialization (3)
(5) d	eliminate (4, H_2)
(6) c	specialization (3)
(7) $c \wedge d$	(5) și (6)
(8) $\neg a \vee e$	modus ponens (7, H_4)
(9) e	eliminate (1, 8)

Logical (semantic) consequence

Interpretation = assignment of truth to the sentences of a formula. A formula can be true or false in an interpretation.

Def.: A set of formulas H = {H1, . . . , Hn} implies a formula C if any interpretation satisfying (the formulas in) H satisfies C

Denoted: $H \models C$

(C is a logical consequence/semantic consequence of the hypotheses H)

Logical (semantic) consequence

To establish the semantic consequence we need to interpret formulas (with truth values/functions)

$\Rightarrow$ we work with the semantics (meaning) of formulas

Exemple: we show that $\{A \vee B,\ C \vee \neg B\} \models A \vee C$

Case 1: $v(B) = \mathrm{T}$. Then $v(A \vee B) = \mathrm{T}$ and $v(C \vee \neg B) = v(C)$. If $v(C) = T$, then $v(A \vee C) = T$, so the statement is true.

Case 2: $v(B) = \mathrm{F}$. Similarly, we reduce to $\{A\} \models A \vee C$ (true).

Soundness and completeness of the rules

$H \vdash C$: inference (purely syntactic, from axioms and inference rules)

$H \models C$: implication, semantic consequence (truth values)

Soundness:

If H is a set of formulas, and C is a formula such that $H \vdash C$, then $H \models C$

(Any theorem is valid;

any statement obtained by deduction is always true).

Soundness and completeness of the rules

$H \vdash C$: inference (purely syntactic, from axioms and inference rules)

$H \models C$: implication, semantic consequence (truth values)

Completeness:

If H is a set of formulas, and C is a formula such that $H \models C$, then $H \vdash C$.

(Every tautology is a theorem,

any semantic consequence can be deduced from the same hypothesis).

Soundness and completeness of the rules

$H \vdash C$: inference (purely syntactic, from axioms and inference rules)

$H \models C$: implication, semantic consequence (truth values)

Propositional logic is sound and complete:

To prove a formula, we can show that it is valid.

To do this, we check that its negation is not satisfiable.

Thank you!

Bibliography

The content of the course is based on the material from the LSD course taught by Prof. Dr. Eng. Marius Minea and S.l. Dr. Eng. Casandra Holotescu (http://staff.cs.upt.ro/~marius/curs/lsd/index.html)