Propositional Satisfiability

Pedro Cabalar

Dept. Computer Science
University of Corunna, SPAIN

April 8, 2025

Models of a formula

- We can define $M(\Gamma)$ = the set of models of a theory (or formula) Γ.
  Example: $M(a \vee b) = \{\{a, b\}, \{a\}, \{b\}\}$

Models of a formula

- We can define $M(\Gamma)$ = the set of models of a theory (or formula) Γ. Example: $M(a \vee b) = \{\{a, b\}, \{a\}, \{b\}\}$
- The models of a formula can be inspected by structural induction:

$$
\begin{aligned}
M(\alpha \vee \beta) &= M(\alpha) \cup M(\beta) \\
M(\alpha \wedge \beta) &= M(\alpha) \cap M(\beta) \\
M(\neg\alpha) &= 2^{\Sigma} \setminus M(\alpha)
\end{aligned}
$$

Models of a formula

- We can define $M(\Gamma)$ = the set of models of a theory (or formula) Γ.
  Example: $M(a \vee b) = \{\{a, b\}, \{a\}, \{b\}\}$
- The models of a formula can be inspected by structural induction:

$$\begin{aligned} M(\alpha \vee \beta) &= M(\alpha) \cup M(\beta) \\ M(\alpha \wedge \beta) &= M(\alpha) \cap M(\beta) \\ M(\neg \alpha) &= 2^{\Sigma} \setminus M(\alpha) \end{aligned}$$

- Two formulas α, β are equivalent if $M(\alpha) = M(\beta)$ (same models)

Models of a formula

- From a set S of interpretations: can we get a formula α s.t. $M(\alpha) = S$?

Models of a formula

- From a set S of interpretations: can we get a formula α s.t. $M(\alpha) = S$?
- Example: find α to cover $M(\alpha) = \{\{a, c\}, \{b, c\}, \{a, b, c\}\}$

Models of a formula

- From a set S of interpretations: can we get a formula α s.t. $M(\alpha) = S$?
- Example: find α to cover $M(\alpha) = \{\{a, c\}, \{b, c\}, \{a, b, c\}\}$
- Does this formula α always exist?

Models of a formula

- From a set S of interpretations: can we get a formula α s.t. $M(\alpha) = S$?
- Example: find α to cover $M(\alpha) = \{\{a, c\}, \{b, c\}, \{a, b, c\}\}$
- Does this formula α always exist?
- We will see a method (*minterms*) to obtain a minimal representation

Models of a formula

- Def. relation $\Gamma \models \alpha$ is called logical consequence or entailment and defined as $M(\Gamma) \subseteq M(\alpha)$.

Models of a formula

- Def. relation $\Gamma \models \alpha$ is called logical consequence or entailment and defined as $M(\Gamma) \subseteq M(\alpha)$.
  Example $\{happy, (rain \rightarrow \neg happy)\} \models \neg rain$

Models of a formula

- Def. relation $\Gamma \models \alpha$ is called logical consequence or entailment and defined as $M(\Gamma) \subseteq M(\alpha)$.
  Example $\{happy, (rain \rightarrow \neg happy)\} \models \neg rain$
- If $M(\alpha) = \emptyset$ (no models!), α is inconsistent or unsatisfiable

Models of a formula

- Def. relation $\Gamma \models \alpha$ is called logical consequence or entailment and defined as $M(\Gamma) \subseteq M(\alpha)$.
  Example $\{happy, (rain \rightarrow \neg happy)\} \models \neg rain$
- If $M(\alpha) = \emptyset$ (no models!), α is inconsistent or unsatisfiable
  Examples: $rain \wedge \neg rain$, $\bot$, ...

Models of a formula

- Def. relation $\Gamma \models \alpha$ is called logical consequence or entailment and defined as $M(\Gamma) \subseteq M(\alpha)$.
  Example $\{happy, (rain \rightarrow \neg happy)\} \models \neg rain$
- If $M(\alpha) = \emptyset$ (no models!), α is inconsistent or unsatisfiable
  Examples: $rain \wedge \neg rain$, $\bot$, ...
- If $M(\alpha) = 2^{\Sigma}$ (all interpretations are models), α, is valid or a tautology.

Models of a formula

- Def. relation $\Gamma \models \alpha$ is called logical consequence or entailment and defined as $M(\Gamma) \subseteq M(\alpha)$.
  Example $\{happy, (rain \rightarrow \neg happy)\} \models \neg rain$
- If $M(\alpha) = \emptyset$ (no models!), α is inconsistent or unsatisfiable
  Examples: $rain \wedge \neg rain$, $\bot$, ...
- If $M(\alpha) = 2^{\Sigma}$ (all interpretations are models), α, is valid or a tautology. Examples: $rain \vee \neg rain$, $\top$, $b \wedge c \wedge d \rightarrow (d \rightarrow b)$, ...

Models of a formula

- Def. relation $\Gamma \models \alpha$ is called logical consequence or entailment and defined as $M(\Gamma) \subseteq M(\alpha)$.
  Example $\{happy, (rain \rightarrow \neg happy)\} \models \neg rain$
- If $M(\alpha) = \emptyset$ (no models!), α is inconsistent or unsatisfiable
  Examples: $rain \wedge \neg rain$, $\bot$, ...
- If $M(\alpha) = 2^{\Sigma}$ (all interpretations are models), α, is valid or a tautology. Examples: $rain \vee \neg rain$, $\top$, $b \wedge c \wedge d \rightarrow (d \rightarrow b)$, ...
- We write $\models \alpha$ to mean that α is a tautology

Models of a formula

- Def. relation $\Gamma \models \alpha$ is called logical consequence or entailment and defined as $M(\Gamma) \subseteq M(\alpha)$.
  Example $\{happy, (rain \rightarrow \neg happy)\} \models \neg rain$
- If $M(\alpha) = \emptyset$ (no models!), α is inconsistent or unsatisfiable
  Examples: $rain \wedge \neg rain$, $\bot$, ...
- If $M(\alpha) = 2^\Sigma$ (all interpretations are models), α, is valid or a tautology. Examples: $rain \vee \neg rain$, $\top$, $b \wedge c \wedge d \rightarrow (d \rightarrow b)$, ...
- We write $\models \alpha$ to mean that α is a tautology
  Note: this is $\emptyset \models \alpha$, so we require $M(\emptyset) = 2^\Sigma$

Models of a formula

- Def. relation $\Gamma \models \alpha$ is called logical consequence or entailment and defined as $M(\Gamma) \subseteq M(\alpha)$.
  Example $\{happy, (rain \rightarrow \neg happy)\} \models \neg rain$
- If $M(\alpha) = \emptyset$ (no models!), α is inconsistent or unsatisfiable
  Examples: $rain \wedge \neg rain$, $\bot$, ...
- If $M(\alpha) = 2^{\Sigma}$ (all interpretations are models), α, is valid or a tautology. Examples: $rain \vee \neg rain$, $\top$, $b \wedge c \wedge d \rightarrow (d \rightarrow b)$, ...
- We write $\models \alpha$ to mean that α is a tautology
  Note: this is $\emptyset \models \alpha$, so we require $M(\emptyset) = 2^{\Sigma} \subseteq M(\alpha)$

Propositional Logic: Semantics

Theorem

$\models \alpha \rightarrow \beta$ *is equivalent to* $\alpha \models \beta$.

Definition (Weaker/stronger formula)

When $\models \alpha \rightarrow \beta$*, or just* $M(\alpha) \subseteq M(\beta)$*, we say that* α *is stronger than* β *(or* β *is weaker* α*).*

Propositional Logic: Semantics

Theorem

$\models \alpha \to \beta$ *is equivalent to* $\alpha \models \beta$.

Definition (Weaker/stronger formula)

When $\models \alpha \to \beta$, *or just* $M(\alpha) \subseteq M(\beta)$, *we say that* α *is* ***stronger*** *than* β *(or* β *is* ***weaker*** α*).*

- Which are the strongest and weakest possible formulae?

Theorem

$\models \alpha \to \beta$ *is equivalent to* $\alpha \models \beta$.

Definition (Weaker/stronger formula)

When $\models \alpha \to \beta$*, or just* $M(\alpha) \subseteq M(\beta)$*, we say that* α *is* ***stronger*** *than* β *(or* β *is* ***weaker*** α*).*

- Which are the strongest and weakest possible formulae?
- Examples: for each pair, which is the strongest?

p	$p \wedge q$
p	$p \vee \neg q$
$p \vee q$	$p \wedge q$
p	$(q \to p)$
$p \wedge \neg q$	$\neg p \wedge q$

Theorem

$\models \alpha \to \beta$ *is equivalent to* $\alpha \models \beta$.

Definition (Weaker/stronger formula)

When $\models \alpha \to \beta$*, or just* $M(\alpha) \subseteq M(\beta)$*, we say that* α *is* ***stronger*** *than* β *(or* β *is* ***weaker*** α*).*

- Which are the strongest and weakest possible formulae?
- Examples: for each pair, which is the strongest?

p	$p \wedge q$
p	$p \vee \neg q$
$p \vee q$	$p \wedge q$
p	$(q \to p)$
$p \wedge \neg q$	$\neg p \wedge q$

Theorem

$\models \alpha \rightarrow \beta$ *is equivalent to* $\alpha \models \beta$.

Definition (Weaker/stronger formula)

When $\models \alpha \rightarrow \beta$, *or just* $M(\alpha) \subseteq M(\beta)$, *we say that* α *is* ***stronger*** *than* β *(or* β *is* ***weaker*** α*).*

- Which are the strongest and weakest possible formulae?
- Examples: for each pair, which is the strongest?

p	$\leftarrow$	$p \wedge q$
p		$p \vee \neg q$
$p \vee q$		$p \wedge q$
p		$(q \rightarrow p)$
$p \wedge \neg q$		$\neg p \wedge q$

Propositional Logic: Semantics

Theorem

$\models \alpha \to \beta$ *is equivalent to* $\alpha \models \beta$.

Definition (Weaker/stronger formula)

When $\models \alpha \to \beta$, *or just* $M(\alpha) \subseteq M(\beta)$, *we say that* α *is* ***stronger*** *than* β *(or* β *is* ***weaker*** α*).*

- Which are the strongest and weakest possible formulae?
- Examples: for each pair, which is the strongest?

$$
\begin{array}{ccc}
p & \leftarrow & p \wedge q \\
p & \rightarrow & p \vee \neg q \\
p \vee q & & p \wedge q \\
p & & (q \to p) \\
p \wedge \neg q & & \neg p \wedge q
\end{array}
$$

Propositional Logic: Semantics

Theorem

$\models \alpha \rightarrow \beta$ *is equivalent to* $\alpha \models \beta$.

Definition (Weaker/stronger formula)

When $\models \alpha \rightarrow \beta$*, or just* $M(\alpha) \subseteq M(\beta)$*, we say that* α *is stronger than* β *(or* β *is weaker* α*).*

- Which are the strongest and weakest possible formulae?
- Examples: for each pair, which is the strongest?

$$
\begin{array}{ccc}
p & \leftarrow & p \wedge q \\
p & \rightarrow & p \vee \neg q \\
p \vee q & \leftarrow & p \wedge q \\
p & & (q \rightarrow p) \\
p \wedge \neg q & & \neg p \wedge q
\end{array}
$$

Propositional Logic: Semantics

Theorem

$\models \alpha \rightarrow \beta$ *is equivalent to* $\alpha \models \beta$.

Definition (Weaker/stronger formula)

When $\models \alpha \rightarrow \beta$, *or just* $M(\alpha) \subseteq M(\beta)$, *we say that* α *is stronger than* β *(or* β *is weaker* α*).*

- Which are the strongest and weakest possible formulae?
- Examples: for each pair, which is the strongest?

$$
\begin{array}{ccc}
p & \leftarrow & p \wedge q \\
p & \rightarrow & p \vee \neg q \\
p \vee q & \leftarrow & p \wedge q \\
p & \rightarrow & (q \rightarrow p) \\
p \wedge \neg q & & \neg p \wedge q
\end{array}
$$

Satisfiability

Definition (SAT decision problem)

Decision problem $SAT(\alpha) \in \{yes, no\}$ checks whether a formula α has some model. That is: $SAT(\alpha) = yes$ iff $M(\alpha) \neq \emptyset$.

- Time complexity: NP-complete problem.

Satisfiability

Definition (SAT decision problem)

Decision problem $SAT(\alpha) \in \{yes, no\}$ checks whether a formula α has some model. That is: $SAT(\alpha) = yes$ iff $M(\alpha) \neq \emptyset$.

- Time complexity: NP-complete problem. Furthermore, it was the first problem identified problem in this class, and crucial for proving that other problems belong to it.

Satisfiability

Definition (SAT decision problem)

Decision problem $SAT(\alpha) \in \{yes, no\}$ checks whether a formula α has some model. That is: $SAT(\alpha) = yes$ iff $M(\alpha) \neq \emptyset$.

- Time complexity: NP-complete problem. Furthermore, it was the first problem identified problem in this class, and crucial for proving that other problems belong to it.
- Nowadays, SAT is an outstanding state-of-the-art research area for search algorithms. There exist many efficient tools and commercial applications. See `www.satlive.org`

Satisfiability

Definition (SAT decision problem)

Decision problem $SAT(\alpha) \in \{yes, no\}$ checks whether a formula α has some model. That is: $SAT(\alpha) = yes$ iff $M(\alpha) \neq \emptyset$.

- Time complexity: NP-complete problem. Furthermore, it was the first problem identified problem in this class, and crucial for proving that other problems belong to it.
- Nowadays, SAT is an outstanding state-of-the-art research area for search algorithms. There exist many efficient tools and commercial applications. See `www.satlive.org`
- SAT keypoint: instead of designing an *ad hoc* search algorithm, encode the problem into propositional logic and use SAT as a backend.

Exercise

- Programming the satisfaction relation $\models$ in Prolog:

```
:- op(210, yfx, &).
:- op(220, yfx, v).
:- op(1060, yfx, <->).
sat(_I, false) :- !, fail.
sat(_I, true) :- !.
sat(I, P) :- atom(P),!,member(P,I),!.
sat(I, -A) :- \+ sat(I, A).
sat(I, A & B) :- sat(I, A), sat(I, B).
sat(I, A v B) :- sat(I,A),! ; sat(I, B).
sat(I, A -> B) :- sat(I,-A v B).
sat(I, A <-> B) :- sat(I,(A -> B)&(B -> A)).
```

Exercise

- Testing whether a formula is a tautology:

```
tautology(S, F) :- \+ (subset(S,I), \+ sat(I,F)).

subset([],[]) :- !.
subset([X | Xs],S) :- subset(Xs,S).
subset([X | Xs],[X | S]) :- subset(Xs,S).
```

Conjunctive Normal Form

- The input for most SAT solvers is a formula α in conjunctive normal form.

Conjunctive Normal Form

- The input for most SAT solvers is a formula α in conjunctive normal form.
- This is a conjunction of clauses = disjunctions of literals. Example: $(p_1 \vee \neg p_2) \wedge (\neg p_3 \vee p_1 \vee p_2) \wedge \neg p_1 \wedge (p_2 \vee p_4)$

Conjunctive Normal Form

- The input for most SAT solvers is a formula α in conjunctive normal form.
- This is a conjunction of clauses = disjunctions of literals. Example: $(p_1 \vee \neg p_2) \wedge (\neg p_3 \vee p_1 \vee p_2) \wedge \neg p_1 \wedge (p_2 \vee p_4)$
- This is represented as a text file in DIMACS format. For instance, the formula above becomes

```
p cnf 3 4        3 variables, 4 clauses
1 -2 0           0 marks the end of a clause
-3 1 2 0
-1 0
2 4 0
```

Reduction to CNF

- Reduction to CNF: several methods can be used (for instance, semantic tableaux)

Reduction to CNF

- Reduction to CNF: several methods can be used (for instance, semantic tableaux)
- Reducing a formula α to CNF causes an exponential cost
- Distributivity blows up

$$(p_1 \wedge q_1) \vee (p_2 \wedge q_2) \vee \cdots \vee (p_n \wedge q_n)$$

2^n disjunctions depending on whether we take p or q for each i

Theorem

Reducing $\varphi \longrightarrow CNF(\varphi)$ in classical logic is NP-hard.

Reduction to CNF (with auxiliary atoms)

- [Tseytin 1968] proposed a polynomial reduction but. . .
- Key idea: introduce auxiliary variables per each non-atomic subformula, then add equivalences to fix their truth

Reduction to CNF (with auxiliary atoms)

- [Tseytin 1968] proposed a polynomial reduction but…
- Key idea: introduce auxiliary variables per each non-atomic subformula, then add equivalences to fix their truth

$$(p_1 \wedge q_1) \vee (p_2 \wedge q_2) \vee \cdots \vee (p_n \wedge q_n)$$

Reduction to CNF (with auxiliary atoms)

- [Tseytin 1968] proposed a polynomial reduction but...
- Key idea: introduce auxiliary variables per each non-atomic subformula, then add equivalences to fix their truth

$$\underbrace{(p_1 \wedge q_1)}_{a_1} \vee \underbrace{(p_2 \wedge q_2)}_{a_2} \vee \cdots \vee \underbrace{(p_n \wedge q_n)}_{a_n}$$

Reduction to CNF (with auxiliary atoms)

- [Tseytin 1968] proposed a polynomial reduction but...
- Key idea: introduce auxiliary variables per each non-atomic subformula, then add equivalences to fix their truth

$$a_1 \vee a_2 \vee \cdots \vee a_n$$
$$a_i \leftrightarrow p_i \wedge q_i$$

Reduction to CNF (with auxiliary atoms)

- [Tseytin 1968] proposed a polynomial reduction but...
- Key idea: introduce auxiliary variables per each non-atomic subformula, then add equivalences to fix their truth

$$a_1 \vee a_2 \vee \cdots \vee a_n$$

$$a_i \rightarrow p_i \wedge q_i \qquad a_i \leftarrow p_i \wedge q_i$$

Reduction to CNF (with auxiliary atoms)

- [Tseytin 1968] proposed a polynomial reduction but. . .
- Key idea: introduce auxiliary variables per each non-atomic subformula, then add equivalences to fix their truth

$$a_1 \vee a_2 \vee \cdots \vee a_n$$

$$a_i \rightarrow p_i \qquad a_i \rightarrow q_i \qquad a_i \leftarrow p_i \wedge q_i$$

Reduction to CNF (with auxiliary atoms)

- [Tseytin 1968] proposed a polynomial reduction but...
- Key idea: introduce auxiliary variables per each non-atomic subformula, then add equivalences to fix their truth

$$a_1 \vee a_2 \vee \cdots \vee a_n$$

$$\neg a_i \vee p_i \qquad \neg a_i \vee q_i \qquad a_i \vee \neg p_i \vee \neg q_i$$

$1 + 3 \cdot n$ clauses. We have n new atoms: we would hide in models

Reduction to CNF (with auxiliary atoms)

Example: reduce the formula below to CNF using Tseytin's technique

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

Reduction to CNF (with auxiliary atoms)

Example: reduce the formula below to CNF using Tseytin's technique

$$\neg(p \vee \underbrace{(q \wedge r)}_{a_1} \vee \neg\underbrace{(p \vee \neg r)}_{a_2})$$

Reduction to CNF (with auxiliary atoms)

Example: reduce the formula below to CNF using Tseytin's technique

$$\neg(p \vee \underbrace{(q \wedge r)}_{a_1} \vee \neg(\underbrace{p \vee \neg r}_{a_2}))$$

$$a_1 \leftrightarrow q \wedge r$$

$$a_2 \leftrightarrow p \vee \neg r$$

Reduction to CNF (with auxiliary atoms)

Example: reduce the formula below to CNF using Tseytin's technique

$$\neg(p \vee a_1 \vee \neg a_2)$$
$$a_1 \leftrightarrow q \wedge r$$
$$a_2 \leftrightarrow p \vee \neg r$$

Reduction to CNF (with auxiliary atoms)

Example: reduce the formula below to CNF using Tseytin's technique

$$\neg p \wedge \neg a_1 \wedge a_2$$

$$a_1 \rightarrow q \wedge r \qquad q \wedge r \rightarrow a_1$$

$$a_2 \rightarrow p \wedge \neg r \qquad p \wedge \neg r \rightarrow a_2$$

Reduction to CNF (with auxiliary atoms)

Example: reduce the formula below to CNF using Tseytin's technique

$$\neg p \land \neg a_1 \land a_2$$

$$a_1 \rightarrow q \qquad a_1 \rightarrow r \qquad q \land r \rightarrow a_1$$

$$a_2 \rightarrow p \qquad a_2 \rightarrow \neg r \qquad p \land \neg r \rightarrow a_2$$

Reduction to CNF (with auxiliary atoms)

Example: reduce the formula below to CNF using Tseytin's technique

$$
\begin{array}{ccc}
\neg p & \neg a_1 & a_2 \\
\neg a_1 \vee q & \neg a_1 \vee r & \neg q \vee \neg r \vee a_1 \\
\neg a_2 \vee p & \neg a_2 \vee \neg r & \neg p \vee \neg r \vee a_2
\end{array}
$$

SAT solvers

Basic Methods: (we will see them in detail later)

- DPLL (Davis-Putnam-Logemann-Loveland)
  - Backtracking algorithm: picks some atom p and tries two branches: one with $p = \mathit{true}$, one with $p = \mathit{false}$.

SAT solvers

Basic Methods: (we will see them in detail later)

- DPLL (Davis-Putnam-Logemann-Loveland)
  - Backtracking algorithm: picks some atom p and tries two branches: one with $p = \mathit{true}$, one with $p = \mathit{false}$.
  - Once a new assignment is made, it is exploited as much as possible (unit propagation)

SAT solvers

Basic Methods: (we will see them in detail later)

- DPLL (Davis-Putnam-Logemann-Loveland)
  - Backtracking algorithm: picks some atom p and tries two branches: one with $p = true$, one with $p = false$.
  - Once a new assignment is made, it is exploited as much as possible (unit propagation)
  - Keypoint: good heuristics to choose the most convenient atom p

SAT solvers

Basic Methods: (we will see them in detail later)

- DPLL (Davis-Putnam-Logemann-Loveland)
  - Backtracking algorithm: picks some atom p and tries two branches: one with $p = true$, one with $p = false$.
  - Once a new assignment is made, it is exploited as much as possible (unit propagation)
  - Keypoint: good heuristics to choose the most convenient atom p
- CDCL (Conflict-Driven Conflict Learning)
  - Maintains an implication graph (each node is a literal, each arrow an implication)
  - When an inconsistent assignment is reached, it extracts from the graph a new clause (reflecting the conflict)
  - back jump: it backtracks several steps backwards to the first-assigned variable involved in the conflict