Logic and Mechanized Reasoning

Propositional Logic

Marijn J.H. Heule

Carnegie Mellon University

Syntax

Semantics

Calculating with Propositions

Random Formulas

Syntax

Semantics

Calculating with Propositions

Random Formulas

Syntax: Definition

The set of propositional formulas is generated inductively:

- Each variable p_i is a formula.
- $\top$ and $\bot$ are formulas.
- If A is a formula, so is $\neg A$ ("not A").
- If A and B are formulas, so are
  - $A \wedge B$ ("A and B"),
  - $A \vee B$ ("A or B"),
  - $A \to B$ ("A implies B"), and
  - $A \leftrightarrow B$ ("A if and only if B").

Syntax: Complexity

Complexity: the number of connectives

$$
\begin{aligned}
complexity(p_i) &= 0 \\
complexity(\top) &= 0 \\
complexity(\bot) &= 0 \\
complexity(\neg A) &= complexity(A) + 1 \\
complexity(A \wedge B) &= complexity(A) + complexity(B) + 1 \\
complexity(A \vee B) &= complexity(A) + complexity(B) + 1 \\
complexity(A \to B) &= complexity(A) + complexity(B) + 1 \\
complexity(A \leftrightarrow B) &= complexity(A) + complexity(B) + 1
\end{aligned}
$$

Syntax: Depth

Depth of the parse tree

$$
\begin{array}{rcl}
depth(p_i) & = & 0 \\
depth(\top) & = & 0 \\
depth(\bot) & = & 0 \\
depth(\neg A) & = & depth(A) + 1 \\
depth(A \wedge B) & = & \max(depth(A), depth(B)) + 1 \\
depth(A \vee B) & = & \max(depth(A), depth(B)) + 1 \\
depth(A \rightarrow B) & = & \max(depth(A), depth(B)) + 1 \\
depth(A \leftrightarrow B) & = & \max(depth(A), depth(B)) + 1
\end{array}
$$

Syntax: Complexity and Depth

Theorem

For every formula A, we have $complexity(A) \leq 2^{depth(A)} - 1$.

Proof.

Base case: $complexity(p_i) = 0 = 2^0 - 1 = 2^{depth(p_i)} - 1$,

Inductive case (first $\neg$, afterwards $\wedge$):

$$complexity(\neg A) \quad =$$

Syntax: Complexity and Depth

Theorem

For every formula A, we have $complexity(A) \leq 2^{depth(A)} - 1$.

Proof.

Base case: $complexity(p_i) = 0 = 2^0 - 1 = 2^{depth(p_i)} - 1$,

Inductive case (first $\neg$, afterwards $\wedge$):

$$\begin{aligned} complexity(\neg A) &= complexity(A) + 1 \\ &\leq \end{aligned}$$

Syntax: Complexity and Depth

Theorem

For every formula A, we have $complexity(A) \leq 2^{depth(A)} - 1$.

Proof.

Base case: $complexity(p_i) = 0 = 2^0 - 1 = 2^{depth(p_i)} - 1$,

Inductive case (first $\neg$, afterwards $\wedge$):

$$\begin{aligned} complexity(\neg A) &= complexity(A) + 1 \\ &\leq 2^{depth(A)} - 1 + 1 \\ &\leq \end{aligned}$$

Syntax: Complexity and Depth

Theorem

For every formula A, we have $complexity(A) \leq 2^{depth(A)} - 1$.

Proof.

Base case: $complexity(p_i) = 0 = 2^0 - 1 = 2^{depth(p_i)} - 1$,

Inductive case (first $\neg$, afterwards $\wedge$):

$$\begin{aligned}
complexity(\neg A) &= complexity(A) + 1 \\
&\leq 2^{depth(A)} - 1 + 1 \\
&\leq 2^{depth(A)} + 2^{depth(A)} - 1 \\
&\leq
\end{aligned}$$

Syntax: Complexity and Depth

Theorem

For every formula A, we have $complexity(A) \leq 2^{depth(A)} - 1$.

Proof.

Base case: $complexity(p_i) = 0 = 2^0 - 1 = 2^{depth(p_i)} - 1$,

Inductive case (first $\neg$, afterwards $\wedge$):

$$\begin{aligned} complexity(\neg A) &= complexity(A) + 1 \\ &\leq 2^{depth(A)} - 1 + 1 \\ &\leq 2^{depth(A)} + 2^{depth(A)} - 1 \\ &\leq 2^{depth(A)+1} - 1 = 2^{depth(\neg A)} - 1. \end{aligned}$$

Syntax: Complexity and Depth

Theorem

For every formula A, we have $complexity(A) \leq 2^{depth(A)} - 1$.

Proof.

Base case: $complexity(p_i) = 0 = 2^0 - 1 = 2^{depth(p_i)} - 1$,

Inductive case (first $\neg$, afterwards $\wedge$):

$$\begin{array}{rcl} complexity(\neg A) & = & complexity(A) + 1 \\ & \leq & 2^{depth(A)} - 1 + 1 \\ & \leq & 2^{depth(A)} + 2^{depth(A)} - 1 \\ & \leq & 2^{depth(A)+1} - 1 = 2^{depth(\neg A)} - 1. \end{array}$$

$$complexity(A \wedge B) \quad =$$

Syntax: Complexity and Depth

Theorem

For every formula A, we have $complexity(A) \leq 2^{depth(A)} - 1$.

Proof.

Base case: $complexity(p_i) = 0 = 2^0 - 1 = 2^{depth(p_i)} - 1$,

Inductive case (first $\neg$, afterwards $\wedge$):

$$\begin{aligned}
complexity(\neg A) &= complexity(A) + 1 \\
&\leq 2^{depth(A)} - 1 + 1 \\
&\leq 2^{depth(A)} + 2^{depth(A)} - 1 \\
&\leq 2^{depth(A)+1} - 1 = 2^{depth(\neg A)} - 1.
\end{aligned}$$

$$\begin{aligned}
complexity(A \wedge B) &= complexity(A) + complexity(B) + 1 \\
&\leq
\end{aligned}$$

Syntax: Complexity and Depth

Theorem

For every formula A*, we have* $complexity(A) \leq 2^{depth(A)} - 1$.

Proof.

Base case: $complexity(p_i) = 0 = 2^0 - 1 = 2^{depth(p_i)} - 1$,

Inductive case (first $\neg$, afterwards $\wedge$):

$$\begin{aligned}
complexity(\neg A) &= complexity(A) + 1 \\
&\leq 2^{depth(A)} - 1 + 1 \\
&\leq 2^{depth(A)} + 2^{depth(A)} - 1 \\
&\leq 2^{depth(A)+1} - 1 = 2^{depth(\neg A)} - 1.
\end{aligned}$$

$$\begin{aligned}
complexity(A \wedge B) &= complexity(A) + complexity(B) + 1 \\
&\leq 2^{depth(A)} - 1 + 2^{depth(B)} - 1 + 1 \\
&\leq
\end{aligned}$$

Syntax: Complexity and Depth

Theorem

For every formula A, we have $complexity(A) \leq 2^{depth(A)} - 1$.

Proof.

Base case: $complexity(p_i) = 0 = 2^0 - 1 = 2^{depth(p_i)} - 1$,

Inductive case (first $\neg$, afterwards $\wedge$):

$$
\begin{aligned}
complexity(\neg A) &= complexity(A) + 1 \\
&\leq 2^{depth(A)} - 1 + 1 \\
&\leq 2^{depth(A)} + 2^{depth(A)} - 1 \\
&\leq 2^{depth(A)+1} - 1 = 2^{depth(\neg A)} - 1.
\end{aligned}
$$

$$
\begin{aligned}
complexity(A \wedge B) &= complexity(A) + complexity(B) + 1 \\
&\leq 2^{depth(A)} - 1 + 2^{depth(B)} - 1 + 1 \\
&\leq 2 \cdot 2^{\max(depth(A), depth(B))} - 1 \\
&=
\end{aligned}
$$

Syntax: Complexity and Depth

Theorem

For every formula A, we have $complexity(A) \leq 2^{depth(A)} - 1$.

Proof.

Base case: $complexity(p_i) = 0 = 2^0 - 1 = 2^{depth(p_i)} - 1$,

Inductive case (first $\neg$, afterwards $\wedge$):

$$\begin{aligned}
complexity(\neg A) &= complexity(A) + 1 \\
&\leq 2^{depth(A)} - 1 + 1 \\
&\leq 2^{depth(A)} + 2^{depth(A)} - 1 \\
&\leq 2^{depth(A)+1} - 1 = 2^{depth(\neg A)} - 1.
\end{aligned}$$

$$\begin{aligned}
complexity(A \wedge B) &= complexity(A) + complexity(B) + 1 \\
&\leq 2^{depth(A)} - 1 + 2^{depth(B)} - 1 + 1 \\
&\leq 2 \cdot 2^{\max(depth(A), depth(B))} - 1 \\
&= 2^{\max(depth(A), depth(B)) + 1} - 1 \\
&=
\end{aligned}$$

Syntax: Complexity and Depth

Theorem

For every formula A, we have $complexity(A) \leq 2^{depth(A)} - 1$.

Proof.

Base case: $complexity(p_i) = 0 = 2^0 - 1 = 2^{depth(p_i)} - 1$,

Inductive case (first $\neg$, afterwards $\wedge$):

$$\begin{aligned}
complexity(\neg A) &= complexity(A) + 1 \\
&\leq 2^{depth(A)} - 1 + 1 \\
&\leq 2^{depth(A)} + 2^{depth(A)} - 1 \\
&\leq 2^{depth(A)+1} - 1 = 2^{depth(\neg A)} - 1.
\end{aligned}$$

$$\begin{aligned}
complexity(A \wedge B) &= complexity(A) + complexity(B) + 1 \\
&\leq 2^{depth(A)} - 1 + 2^{depth(B)} - 1 + 1 \\
&\leq 2 \cdot 2^{\max(depth(A), depth(B))} - 1 \\
&= 2^{\max(depth(A), depth(B)) + 1} - 1 \\
&= 2^{depth(A \wedge B)} - 1
\end{aligned}$$

Syntax: Subformulas

$$\begin{aligned} \mathit{subformulas}(A) &= \{A\} \quad \text{if } A \text{ is atomic} \\ \mathit{subformulas}(\neg A) &= \{\neg A\} \cup \mathit{subformulas}(A) \\ \mathit{subformulas}(A \star B) &= \{A \star B\} \cup \mathit{subformulas}(A) \cup \\ &\quad\ \mathit{subformulas}(B) \end{aligned}$$

Syntax: Subformulas

$$
\begin{array}{rcl}
\mathit{subformulas}(A) & = & \{A\} \quad \text{if } A \text{ is atomic} \\
\mathit{subformulas}(\neg A) & = & \{\neg A\} \cup \mathit{subformulas}(A) \\
\mathit{subformulas}(A \star B) & = & \{A \star B\} \cup \mathit{subformulas}(A) \cup \\
 & & \mathit{subformulas}(B)
\end{array}
$$

Example

Consider the formula $(\neg A \wedge C) \to \neg(B \vee C)$.
The *subformulas* function returns

Syntax: Subformulas

$$\begin{array}{rcl} subformulas(A) & = & \{A\} \quad \text{if } A \text{ is atomic} \\ subformulas(\neg A) & = & \{\neg A\} \cup subformulas(A) \\ subformulas(A \star B) & = & \{A \star B\} \cup subformulas(A) \cup \\ & & subformulas(B) \end{array}$$

Example

Consider the formula $(\neg A \wedge C) \rightarrow \neg(B \vee C)$.
The *subformulas* function returns
$\{(\neg A \wedge C) \rightarrow \neg(B \vee C), \neg A \wedge C, \neg A, A, C, \neg(B \vee C), B \vee C, B)\}$

Syntax: Proposition

Proposition

For every pair of formulas A *and* B*, if* $B \in subformulas(A)$ *and* $A \in subformulas(B)$ *then* A *and* B *are atomic.*

True or false?

Syntax: Proposition

Proposition

For every pair of formulas A and B, if $B \in subformulas(A)$ and $A \in subformulas(B)$ then A and B are atomic.

True or false?

Proof.

False. A counterexample is $A = B = \neg p$. □

Syntax

Semantics

Calculating with Propositions

Random Formulas

Semantics: Introduction

Consider the formula $p \wedge (\neg q \vee r)$. Is it true?

Semantics: Introduction

Consider the formula $p \wedge (\neg q \vee r)$. Is it true?

It depends on the truth of p, q, and r.

Semantics: Introduction

Consider the formula $p \wedge (\neg q \vee r)$. Is it true?

It depends on the truth of p, q, and r.

Once we specify which of p, q, and r are true and which are false, the truth value of $p \wedge (\neg q \vee r)$ is completely determined.

Semantics: Introduction

Consider the formula $p \wedge (\neg q \vee r)$. Is it true?

It depends on the truth of p, q, and r.

Once we specify which of p, q, and r are true and which are false, the truth value of $p \wedge (\neg q \vee r)$ is completely determined.

A truth assignment τ provides this specification by mapping propositional variables to the constants $\top$ and $\bot$.

Semantics: Evaluation

$$
\begin{aligned}
[\![p_i]\!]_\tau &= \tau(p_i) \\
[\![\top]\!]_\tau &= \top \\
[\![\bot]\!]_\tau &= \bot \\
[\![\neg A]\!]_\tau &= \begin{cases} \top & \text{if } [\![A]\!]_\tau = \bot \\ \bot & \text{otherwise} \end{cases} \\
[\![A \wedge B]\!]_\tau &= \begin{cases} \top & \text{if } [\![A]\!]_\tau = \top \text{ and } [\![B]\!]_\tau = \top \\ \bot & \text{otherwise} \end{cases} \\
[\![A \vee B]\!]_\tau &= \begin{cases} \top & \text{if } [\![A]\!]_\tau = \top \text{ or } [\![B]\!]_\tau = \top \\ \bot & \text{otherwise} \end{cases} \\
[\![A \to B]\!]_\tau &= \begin{cases} \top & \text{if } [\![A]\!]_\tau = \bot \text{ or } [\![B]\!]_\tau = \top \\ \bot & \text{otherwise} \end{cases} \\
[\![A \leftrightarrow B]\!]_\tau &= \begin{cases} \top & \text{if } [\![A]\!]_\tau = [\![B]\!]_\tau \\ \bot & \text{otherwise} \end{cases}
\end{aligned}
$$

Semantics: Satisfiable, Unsatisfiable, and Valid

- If $[\![A]\!]_\tau = \top$, then A is satisfied by τ. In that case, τ is a satisfying assignment of A.
- A propositional formula A is satisfiable iff there exists an assignment τ that satisfies it and unsatisfiable otherwise.
- A propositional formula A is valid iff every assignment satisfies it.

Semantics: Satisfiable, Unsatisfiable, and Valid

- If $[\![A]\!]_\tau = \top$, then A is satisfied by τ. In that case, τ is a satisfying assignment of A.
- A propositional formula A is satisfiable iff there exists an assignment τ that satisfies it and unsatisfiable otherwise.
- A propositional formula A is valid iff every assignment satisfies it.

Example

Which one(s) of the formulas is satisfiable/unsatisfiable/valid?

- $(A \leftrightarrow B) \vee (\neg C)$
- $(A) \vee (\neg B) \vee (\neg A \wedge B)$
- $(A) \wedge (\neg B) \wedge (A \rightarrow B)$

Semantics: Relation Valid and Unsatisfiable

Theorem

A propositional formula A is valid if and only if $\neg A$ is unsatisfiable.

Semantics: Relation Valid and Unsatisfiable

Theorem

A propositional formula A is valid if and only if $\neg A$ is unsatisfiable.

Proof.

A is valid if and only if $[\![A]\!]_\tau = \top$ for every assignment τ.

Semantics: Relation Valid and Unsatisfiable

Theorem

A propositional formula A is valid if and only if $\neg A$ is unsatisfiable.

Proof.

A is valid if and only if $[\![A]\!]_\tau = \top$ for every assignment τ.

By the def of $[\![\neg A]\!]_\tau$, this happens iff $[\![\neg A]\!]_\tau = \bot$ for every τ.

Semantics: Relation Valid and Unsatisfiable

Theorem

A propositional formula A is valid if and only if $\neg A$ is unsatisfiable.

Proof.

A is valid if and only if $[\![A]\!]_\tau = \top$ for every assignment τ.

By the def of $[\![\neg A]\!]_\tau$, this happens iff $[\![\neg A]\!]_\tau = \bot$ for every τ.

This is the same as saying that $\neg A$ is unsatisfiable. □

Semantics: Proposition 1

Proposition

For every pair of formulas A and B, $A \wedge B$ is valid if and only if A is valid and B is valid.

True or false?

Semantics: Proposition 1

Proposition

For every pair of formulas A and B, $A \wedge B$ is valid if and only if A is valid and B is valid.

True or false?

Proof.

True. $A \wedge B$ is valid means that for every assignment τ we have $[\![A \wedge B]\!]_\tau = \top$. By the definition of $[\![A \wedge B]\!]$, this happens if and only if $[\![A]\!]_\tau = \top$ and $[\![B]\!]_\tau = \top$ for every τ, i.e. if and only if A and B are both valid. □

Semantics: Proposition 2

Proposition

For every pair of formulas A and B, $A \wedge B$ is satisfiable if and only if A is satisfiable and B is satisfiable.

True or false?

Semantics: Proposition 2

Proposition

For every pair of formulas A and B, $A \wedge B$ is satisfiable if and only if A is satisfiable and B is satisfiable.

True or false?

Proof.

False. Consider the formula $A \wedge B$ with $A = p$ and $B = \neg p$. Clearly both A and B are satisfiable, while $A \wedge B$ is unsatisfiable. □

Semantics: Proposition 3

Proposition

For every pair of formulas A and B, $A \vee B$ is valid if and only if A is valid or B is valid.

True or false?

Semantics: Proposition 3

Proposition

For every pair of formulas A and B, $A \vee B$ is valid if and only if A is valid or B is valid.

True or false?

Proof.

False. Consider the formula $A \vee B$ with $A = p$ and $B = \neg p$. The formula $A \vee B$ is valid, while either A nor B is valid. □

Semantics: Proposition 4

Proposition

For every pair of formulas A and B, $A \vee B$ is satisfiable if and only if A is satisfiable or B is satisfiable.

True or false?

Semantics: Proposition 4

Proposition

For every pair of formulas A and B, $A \vee B$ is satisfiable if and only if A is satisfiable or B is satisfiable.

True or false?

Proof.

True. Suppose $A \vee B$ is satisfied by τ. By definition it must be the case that $[\![A]\!]_\tau = \top$ or $[\![B]\!]_\tau = \top$, so τ satisfies A or B. Conversely, if an assignment τ satisfies either A or B, then $[\![A]\!]_\tau = \top$ or $[\![B]\!]_\tau = \top$. In either case, $[\![A \vee B]\!]_\tau = \top$. So if A is satisfiable or B is satisfiable, so is $A \vee B$. □

Semantics: Entailment and Equivalence

- If every satisfying assignment of a formula A, also satisfies formula B, the A entails B, denoted by $A \models B$.
- If $A \models B$ and $B \models A$, then A and B are logically equivalent, denoted by $A \equiv B$.

Semantics: Entailment and Equivalence

- If every satisfying assignment of a formula A, also satisfies formula B, the A entails B, denoted by $A \models B$.
- If $A \models B$ and $B \models A$, then A and B are logically equivalent, denoted by $A \equiv B$.

Example

Which formula entails which other formula?

- A
- $\neg A \to B$
- $\neg(\neg A \vee \neg B)$

Semantics: Proposition 5

Proposition

Suppose A *and* B *are formulas and* $A \models B$*.*
If A *is valid, then* B *is valid.*

True or false?

Semantics: Proposition 5

Proposition

Suppose A and B are formulas and $A \models B$.
If A is valid, then B is valid.

True or false?

Proof.

True. Suppose $A \models B$, and suppose A is valid. Let τ be any truth assignment. Since A is valid, $[\![A]\!]_\tau = \top$. Since $A \models B$, $[\![B]\!]_\tau = \top$. We have shown $[\![B]\!]_\tau = \top$ for every τ, i.e. B is valid. □

Semantics: Proposition 6

Proposition

Suppose A *and* B *are formulas and* $A \models B$*.*
If B *is satisfiable, then* A *is satisfiable.*

True or false?

Semantics: Proposition 6

Proposition

Suppose A and B are formulas and $A \models B$.
If B is satisfiable, then A is satisfiable.

True or false?

Proof.
False. A counterexample is $A = p \wedge \neg p$ and $B = p$. □

Semantics: Proposition 7

Proposition

For every triple of formulas A, B, and C, if $A \models B \models C \models A$ then $A \equiv B \equiv C$.

True or false?

Semantics: Proposition 7

Proposition

For every triple of formulas A, B, and C, if $A \models B \models C \models A$ then $A \equiv B \equiv C$.

True or false?

Proof.

True. Suppose $A \models B \models C \models A$. Let τ be any truth assignment. We need to show $[\![A]\!]_\tau = [\![B]\!]_\tau = [\![C]\!]_\tau$. Suppose $[\![A]\!]_\tau = \top$. Since $A \models B$, $[\![B]\!]_\tau = \top$, and since $B \models C$, we have $[\![C]\!]_\tau = \top$. So, in that case, $[\![A]\!]_\tau = [\![B]\!]_\tau = [\![C]\!]_\tau$.
The other possibility is $[\![A]\!]_\tau = \bot$. Since $C \models A$, we must have $[\![C]\!]_\tau = \bot$, and since $B \models C$, we have $[\![B]\!]_\tau = \bot$. So, in that case also, $[\![A]\!]_\tau = [\![B]\!]_\tau = [\![C]\!]_\tau$. □

Semantics: Diplomacy Problem

"You are chief of protocol for the embassy ball. The crown prince instructs you either to invite *Peru* or to exclude *Qatar*. The queen asks you to invite either *Qatar* or *Romania* or both. The king, in a spiteful mood, wants to snub either *Romania* or *Peru* or both. Is there a guest list that will satisfy the whims of the entire royal family?"

Semantics: Diplomacy Problem

"You are chief of protocol for the embassy ball. The crown prince instructs you either to invite *Peru* or to exclude *Qatar*. The queen asks you to invite either *Qatar* or *Romania* or both. The king, in a spiteful mood, wants to snub either *Romania* or *Peru* or both. Is there a guest list that will satisfy the whims of the entire royal family?"

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

Semantics: Truth Table

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

p	q	r	falsifies	$[\![\Gamma]\!]_\tau$
$\bot$	$\bot$	$\bot$	$(q \vee r)$	$\bot$
$\bot$	$\bot$	$\top$	—	$\top$
$\bot$	$\top$	$\bot$	$(p \vee \neg q)$	$\bot$
$\bot$	$\top$	$\top$	$(p \vee \neg q)$	$\bot$
$\top$	$\bot$	$\bot$	$(q \vee r)$	$\bot$
$\top$	$\bot$	$\top$	$(\neg r \vee \neg p)$	$\bot$
$\top$	$\top$	$\bot$	—	$\top$
$\top$	$\top$	$\top$	$(\neg r \vee \neg p)$	$\bot$

Syntax

Semantics

Calculating with Propositions

Random Formulas

Calculating with Propositions: Laws

Some propositional laws (more in the textbook):

$$
\begin{aligned}
A \vee \top &\equiv \top \\
A \wedge \top &\equiv A \\
A \vee B &\equiv B \vee A \\
(A \vee B) \vee C &\equiv A \vee (B \vee C) \\
A \wedge (B \vee C) &\equiv (A \wedge B) \vee (A \wedge C) \\
A \vee (B \wedge C) &\equiv (A \vee B) \wedge (A \vee C) \\
A \wedge (A \vee B) &\equiv A
\end{aligned}
$$

Calculating with Propositions: Laws

Some propositional laws (more in the textbook):

$$
\begin{aligned}
A \vee \top &\equiv \top \\
A \wedge \top &\equiv A \\
A \vee B &\equiv B \vee A \\
(A \vee B) \vee C &\equiv A \vee (B \vee C) \\
A \wedge (B \vee C) &\equiv (A \wedge B) \vee (A \wedge C) \\
A \vee (B \wedge C) &\equiv (A \vee B) \wedge (A \vee C) \\
A \wedge (A \vee B) &\equiv A
\end{aligned}
$$

De Morgan's laws:

$$
\begin{aligned}
\neg(A \wedge B) &\equiv \neg A \vee \neg B \\
\neg(A \vee B) &\equiv \neg A \wedge \neg B
\end{aligned}
$$

Calculating with Propositions: Example

Theorem
For any propositional formulas A and B, we have
$(A \wedge \neg B) \vee B \equiv A \vee B$.

Proof.

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

Calculating with Propositions: Example

Theorem

For any propositional formulas A and B, we have
$(A \wedge \neg B) \vee B \equiv A \vee B$.

Proof.

$$
\begin{aligned}
(A \wedge \neg B) \vee B &\equiv (A \vee B) \wedge (\neg B \vee B) \\
&\equiv
\end{aligned}
$$

Calculating with Propositions: Example

Theorem

For any propositional formulas A and B, we have $(A \wedge \neg B) \vee B \equiv A \vee B$.

Proof.

$$
\begin{aligned}
(A \wedge \neg B) \vee B &\equiv (A \vee B) \wedge (\neg B \vee B) \\
&\equiv (A \vee B) \wedge \top \\
&\equiv
\end{aligned}
$$

Calculating with Propositions: Example

Theorem

For any propositional formulas A and B, we have $(A \wedge \neg B) \vee B \equiv A \vee B$.

Proof.

$$
\begin{aligned}
(A \wedge \neg B) \vee B &\equiv (A \vee B) \wedge (\neg B \vee B) \\
&\equiv (A \vee B) \wedge \top \\
&\equiv (A \vee B).
\end{aligned}
$$

□

Calculating with Propositions: A Harder Example

Theorem

For any propositional formulas A, B, and C, we have
$\neg((A \vee B) \wedge (B \to C)) \equiv (\neg A \vee B) \wedge (\neg A \vee \neg C) \wedge (\neg B \vee \neg C)$.

Proof.

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

Calculating with Propositions: A Harder Example

Theorem

For any propositional formulas A, B, and C, we have
$\neg((A \vee B) \wedge (B \to C)) \equiv (\neg A \vee B) \wedge (\neg A \vee \neg C) \wedge (\neg B \vee \neg C)$.

Proof.

$$
\begin{aligned}
\neg((A \vee B) \wedge (B \to C)) &\equiv \neg((A \vee B) \wedge (\neg B \vee C)) \\
&\equiv
\end{aligned}
$$

Calculating with Propositions: A Harder Example

Theorem

For any propositional formulas A, B, and C, we have
$\neg((A \vee B) \wedge (B \to C)) \equiv (\neg A \vee B) \wedge (\neg A \vee \neg C) \wedge (\neg B \vee \neg C)$.

Proof.

$$\begin{aligned} \neg((A \vee B) \wedge (B \to C)) &\equiv \neg((A \vee B) \wedge (\neg B \vee C)) \\ &\equiv \neg(A \vee B) \vee \neg(\neg B \vee C) \\ &\equiv \end{aligned}$$

Calculating with Propositions: A Harder Example

Theorem

For any propositional formulas A, B, and C, we have
$\neg((A \vee B) \wedge (B \to C)) \equiv (\neg A \vee B) \wedge (\neg A \vee \neg C) \wedge (\neg B \vee \neg C)$.

Proof.

$$
\begin{aligned}
\neg((A \vee B) \wedge (B \to C)) &\equiv \neg((A \vee B) \wedge (\neg B \vee C)) \\
&\equiv \neg(A \vee B) \vee \neg(\neg B \vee C) \\
&\equiv (\neg A \wedge \neg B) \vee (B \wedge \neg C) \\
&\equiv
\end{aligned}
$$

Calculating with Propositions: A Harder Example

Theorem

For any propositional formulas A, B, and C, we have
$\neg((A \vee B) \wedge (B \to C)) \equiv (\neg A \vee B) \wedge (\neg A \vee \neg C) \wedge (\neg B \vee \neg C)$.

Proof.

$$\begin{aligned}
\neg((A \vee B) \wedge (B \to C)) &\equiv \neg((A \vee B) \wedge (\neg B \vee C)) \\
&\equiv \neg(A \vee B) \vee \neg(\neg B \vee C) \\
&\equiv (\neg A \wedge \neg B) \vee (B \wedge \neg C) \\
&\equiv (\neg A \vee (B \wedge \neg C)) \wedge (\neg B \vee (B \wedge \neg C)) \\
&\equiv
\end{aligned}$$

Calculating with Propositions: A Harder Example

Theorem

For any propositional formulas A, B, and C, we have
$\neg((A \vee B) \wedge (B \to C)) \equiv (\neg A \vee B) \wedge (\neg A \vee \neg C) \wedge (\neg B \vee \neg C)$.

Proof.

$$
\begin{aligned}
\neg((A \vee B) \wedge (B \to C)) &\equiv \neg((A \vee B) \wedge (\neg B \vee C)) \\
&\equiv \neg(A \vee B) \vee \neg(\neg B \vee C) \\
&\equiv (\neg A \wedge \neg B) \vee (B \wedge \neg C) \\
&\equiv (\neg A \vee (B \wedge \neg C)) \wedge (\neg B \vee (B \wedge \neg C)) \\
&\equiv (\neg A \vee (B \wedge \neg C)) \wedge (\neg B \vee B) \wedge (\neg B \vee \neg C) \\
&\equiv
\end{aligned}
$$

Calculating with Propositions: A Harder Example

Theorem

For any propositional formulas A, B, and C, we have $\neg((A \vee B) \wedge (B \to C)) \equiv (\neg A \vee B) \wedge (\neg A \vee \neg C) \wedge (\neg B \vee \neg C)$.

Proof.

$$
\begin{aligned}
\neg((A \vee B) \wedge (B \to C)) &\equiv \neg((A \vee B) \wedge (\neg B \vee C)) \\
&\equiv \neg(A \vee B) \vee \neg(\neg B \vee C) \\
&\equiv (\neg A \wedge \neg B) \vee (B \wedge \neg C) \\
&\equiv (\neg A \vee (B \wedge \neg C)) \wedge (\neg B \vee (B \wedge \neg C)) \\
&\equiv (\neg A \vee (B \wedge \neg C)) \wedge (\neg B \vee B) \wedge (\neg B \vee \neg C) \\
&\equiv (\neg A \vee (B \wedge \neg C)) \wedge \top \wedge (\neg B \vee \neg C) \\
&\equiv
\end{aligned}
$$

Calculating with Propositions: A Harder Example

Theorem

For any propositional formulas A, B, and C, we have $\neg((A \vee B) \wedge (B \to C)) \equiv (\neg A \vee B) \wedge (\neg A \vee \neg C) \wedge (\neg B \vee \neg C)$.

Proof.

$$\begin{aligned}
\neg((A \vee B) \wedge (B \to C)) &\equiv \neg((A \vee B) \wedge (\neg B \vee C)) \\
&\equiv \neg(A \vee B) \vee \neg(\neg B \vee C) \\
&\equiv (\neg A \wedge \neg B) \vee (B \wedge \neg C) \\
&\equiv (\neg A \vee (B \wedge \neg C)) \wedge (\neg B \vee (B \wedge \neg C)) \\
&\equiv (\neg A \vee (B \wedge \neg C)) \wedge (\neg B \vee B) \wedge (\neg B \vee \neg C) \\
&\equiv (\neg A \vee (B \wedge \neg C)) \wedge \top \wedge (\neg B \vee \neg C) \\
&\equiv (\neg A \vee (B \wedge \neg C)) \wedge (\neg B \vee \neg C) \\
&\equiv
\end{aligned}$$

Calculating with Propositions: A Harder Example

Theorem

For any propositional formulas A, B, and C, we have
$\neg((A \vee B) \wedge (B \to C)) \equiv (\neg A \vee B) \wedge (\neg A \vee \neg C) \wedge (\neg B \vee \neg C)$.

Proof.

$$
\begin{aligned}
\neg((A \vee B) \wedge (B \to C)) &\equiv \neg((A \vee B) \wedge (\neg B \vee C)) \\
&\equiv \neg(A \vee B) \vee \neg(\neg B \vee C) \\
&\equiv (\neg A \wedge \neg B) \vee (B \wedge \neg C) \\
&\equiv (\neg A \vee (B \wedge \neg C)) \wedge (\neg B \vee (B \wedge \neg C)) \\
&\equiv (\neg A \vee (B \wedge \neg C)) \wedge (\neg B \vee B) \wedge (\neg B \vee \neg C)) \\
&\equiv (\neg A \vee (B \wedge \neg C)) \wedge \top \wedge (\neg B \vee \neg C) \\
&\equiv (\neg A \vee (B \wedge \neg C)) \wedge (\neg B \vee \neg C) \\
&\equiv (\neg A \vee B) \wedge (\neg A \vee \neg C) \wedge (\neg B \vee \neg C).
\end{aligned}
$$

□

Syntax

Semantics

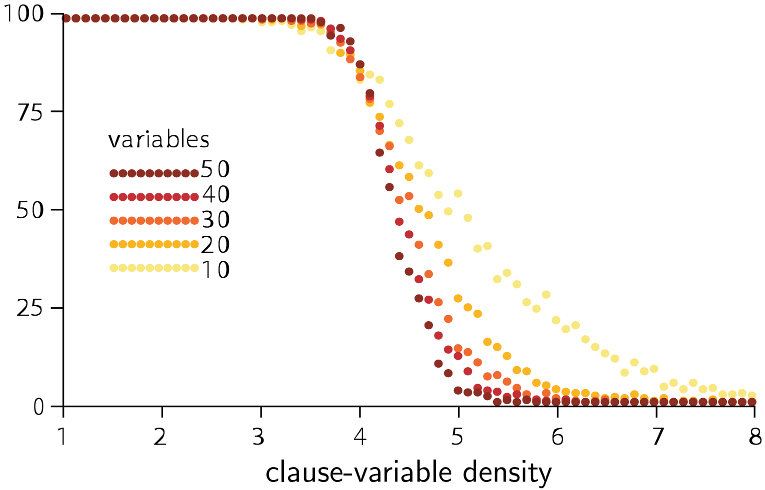
Calculating with Propositions

Random Formulas

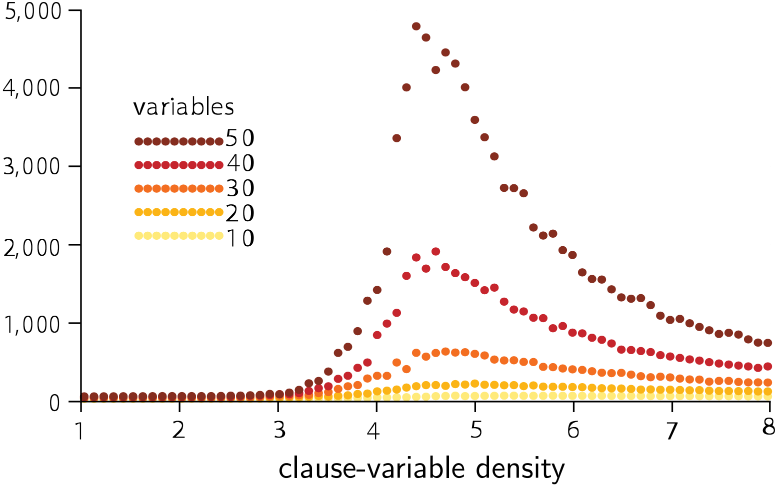
Random Formulas: Introduction

- Formulas in conjunctive normal form
- All clauses have length k
- Variables have the same probability to occur
- Each literal is negated with probability of 50%
- Density is ratio Clauses to Variables

Random Formulas: Phase Transition



Random Formulas: Exponential Runtime



SAT Game

by Olivier Roussel

http://www.cs.utexas.edu/~marijn/game/