This course

Propositional logic

- Syntax
- Semantics
- Normal Forms
- Propositional reasoning
- Propositional Resolution, Proofs
- DPLL and optimizations

First-Order Logic

- Syntax
- Semantics
- Normal Forms
- First-order reasoning
- Resolution and refinements
- Completeness

Section 1 Orderings, multi-sets, induction

Well-Founded Orderings

- Orderings will be used throughout this course.
- Well-founded orderings are crucial for induction proofs.
- Orderings are used for restricting search space in reasoning methods.
- Reference:

  Baader, F. and Nipkow, T. (1998), Term rewriting and all that.

  Cambridge Univ. Press, Chapter 2.

Basic Properties of Relations

Let R be a binary relation over a set X ($R \subseteq X \times X$).

- R is reflexive iff $\forall x \in X, \; R(x, x)$.
- R is irreflexive iff $\forall x \in X, \; \neg R(x, x)$.
- R is total, or linear, iff

  $\forall x, y \in X$, if $x \neq y$ then $R(x, y)$ or $R(y, x)$.
- R is transitive iff

  $\forall x, y, z \in X$, if $R(x, y)$ and $R(y, z)$ then $R(x, z)$.
- R^+ denotes the transitive closure of R:

  $$R^+ = \bigcup_{n=1}^{\infty} R^n = R^1 \cup R^2 \cup R^3 \cup \dots$$

  where $R^1 = R$ and

  $R^{n+1}(x, y)$ iff $\exists z.(R^n(x, z) \wedge R(z, y))$ for $n \geq 0$.
- R^- denotes the reflexive closure of R: $R^- = R \cup I$,

  where I is the set of all pairs (x, x) for $x \in X$.
- R^* denotes the transitive-reflexive closure of R: $R^* = (R^-)^+$

Basic Properties of Relations

Let R be a binary relation over a set X ($R \subseteq X \times X$).

- R is reflexive iff $\forall x \in X,\ R(x,x)$.
- R is irreflexive iff $\forall x \in X,\ \neg R(x,x)$.
- R is total, or linear, iff
  $\forall x, y \in X,$ if $x \neq y$ then $R(x,y)$ or $R(y,x)$.
- R is transitive iff
  $\forall x, y, z \in X,$ if $R(x,y)$ and $R(y,z)$ then $R(x,z)$.
- R^+ denotes the transitive closure of R:
  $$R^+ = \bigcup_{n=1}^{\infty} R^n = R^1 \cup R^2 \cup R^3 \cup \ldots$$
  where $R^1 = R$ and
  $R^{n+1}(x,y)$ iff $\exists z.(R^n(x,z) \wedge R(z,y))$ for $n \geq 0$.
- R^- denotes the reflexive closure of R: $R^- = R \cup I$,
  where I is the set of all pairs (x,x) for $x \in X$.
- R^* denotes the transitive-reflexive closure of R: $R^* = (R^-)^+$

Basic Properties of Relations

Let R be a binary relation over a set X ($R \subseteq X \times X$).

- R is reflexive iff $\forall x \in X, \ R(x,x)$.
- R is irreflexive iff $\forall x \in X, \ \neg R(x,x)$.
- R is total, or linear, iff

  $\forall x, y \in X$, if $x \neq y$ then $R(x,y)$ or $R(y,x)$.
- R is transitive iff

  $\forall x, y, z \in X$, if $R(x,y)$ and $R(y,z)$ then $R(x,z)$.
- R^+ denotes the transitive closure of R:

  $$R^+ = \bigcup_{n=1}^{\infty} R^n = R^1 \cup R^2 \cup R^3 \cup \ldots$$

  where $R^1 = R$ and

  $R^{n+1}(x,y)$ iff $\exists z.(R^n(x,z) \wedge R(z,y))$ for $n \geq 0$.
- R^- denotes the reflexive closure of R: $R^- = R \cup I$,

  where I is the set of all pairs (x,x) for $x \in X$.
- R^* denotes the transitive-reflexive closure of R: $R^* = (R^-)^+$

Basic Properties of Relations

Let R be a binary relation over a set X ($R \subseteq X \times X$).

- R is reflexive iff $\forall x \in X, \ R(x, x)$.
- R is irreflexive iff $\forall x \in X, \ \neg R(x, x)$.
- R is total, or linear, iff

  $\forall x, y \in X$, if $x \neq y$ then $R(x, y)$ or $R(y, x)$.
- R is transitive iff

  $\forall x, y, z \in X$, if $R(x, y)$ and $R(y, z)$ then $R(x, z)$.
- R^+ denotes the transitive closure of R:

  $$R^+ = \bigcup_{n=1}^{\infty} R^n = R^1 \cup R^2 \cup R^3 \cup \ldots$$

  where $R^1 = R$ and

  $R^{n+1}(x, y)$ iff $\exists z.(R^n(x, z) \wedge R(z, y))$ for $n \geq 0$.
- R^- denotes the reflexive closure of R: $R^- = R \cup I$,

  where I is the set of all pairs (x, x) for $x \in X$.
- R^* denotes the transitive-reflexive closure of R: $R^* = (R^-)^+$

Basic Properties of Relations

Let R be a binary relation over a set X ($R \subseteq X \times X$).

- R is reflexive iff $\forall x \in X, \; R(x,x)$.
- R is irreflexive iff $\forall x \in X, \; \neg R(x,x)$.
- R is total, or linear, iff

  $\forall x, y \in X$, if $x \neq y$ then $R(x,y)$ or $R(y,x)$.
- R is transitive iff

  $\forall x, y, z \in X$, if $R(x,y)$ and $R(y,z)$ then $R(x,z)$.
- R^+ denotes the transitive closure of R:

  $$R^+ = \bigcup_{n=1}^{\infty} R^n = R^1 \cup R^2 \cup R^3 \cup \dots$$

  where $R^1 = R$ and

  $R^{n+1}(x,y)$ iff $\exists z.(R^n(x,z) \wedge R(z,y))$ for $n \geq 0$.
- R^- denotes the reflexive closure of R: $R^- = R \cup I$,

  where I is the set of all pairs (x,x) for $x \in X$.
- R^* denotes the transitive-reflexive closure of R: $R^* = (R^-)^+$

Basic Properties of Relations

Let R be a binary relation over a set X ($R \subseteq X \times X$).

- R is reflexive iff $\forall x \in X, \; R(x, x)$.
- R is irreflexive iff $\forall x \in X, \; \neg R(x, x)$.
- R is total, or linear, iff

  $\forall x, y \in X$, if $x \neq y$ then $R(x, y)$ or $R(y, x)$.
- R is transitive iff

  $\forall x, y, z \in X$, if $R(x, y)$ and $R(y, z)$ then $R(x, z)$.
- R^+ denotes the transitive closure of R:

  $$R^+ = \bigcup_{n=1}^{\infty} R^n = R^1 \cup R^2 \cup R^3 \cup \ldots$$

  where $R^1 = R$ and

  $R^{n+1}(x, y)$ iff $\exists z.(R^n(x, z) \wedge R(z, y))$ for $n \geq 0$.
- R^- denotes the reflexive closure of R: $R^- = R \cup I$,

  where I is the set of all pairs (x, x) for $x \in X$.
- R^* denotes the transitive-reflexive closure of R: $R^* = (R^-)^+$

Basic Properties of Relations

Let R be a binary relation over a set X ($R \subseteq X \times X$).

- R is reflexive iff $\forall x \in X, \; R(x, x)$.
- R is irreflexive iff $\forall x \in X, \; \neg R(x, x)$.
- R is total, or linear, iff

  $\forall x, y \in X$, if $x \neq y$ then $R(x, y)$ or $R(y, x)$.
- R is transitive iff

  $\forall x, y, z \in X$, if $R(x, y)$ and $R(y, z)$ then $R(x, z)$.
- R^+ denotes the transitive closure of R:

  $$R^+ = \bigcup_{n=1}^{\infty} R^n = R^1 \cup R^2 \cup R^3 \cup \ldots$$

  where $R^1 = R$ and

  $R^{n+1}(x, y)$ iff $\exists z.(R^n(x, z) \wedge R(z, y))$ for $n \geq 0$.
- R^- denotes the reflexive closure of R: $R^- = R \cup I$,

  where I is the set of all pairs (x, x) for $x \in X$.
- R^* denotes the transitive-reflexive closure of R: $R^* = (R^-)^+$

Orderings

- A (strict) ordering on a set X is a transitive and irreflexive binary relation on X, here denoted by $\succ$.
- The pair $(X, \succ)$ is then called a (strictly) ordered set.
- An element x of X is minimal wrt. $\succ$, if there is no y in X such that $x \succ y$.
- A minimal element x in X is called the smallest (or strictly minimal) element, if for all $y \in X$ different from x, $y \succ x$.
- Maximal and largest (or strictly maximal) elements are defined analogously.
- **Notation**: $\prec$ for the inverse relation $\succ^{-1}$
  $\succeq$ for the reflexive closure $(\succ \cup =)$ of $\succ$, i.e.
  $x \succeq y$ iff either $x \succ y$ or $x = y$

Orderings

- A (strict) ordering on a set X is a transitive and irreflexive binary relation on X, here denoted by $\succ$.
- The pair $(X, \succ)$ is then called a (strictly) ordered set.
- An element x of X is minimal wrt. $\succ$, if there is no y in X such that $x \succ y$.
- A minimal element x in X is called the smallest (or strictly minimal) element, if for all $y \in X$ different from x, $y \succ x$.
- Maximal and largest (or strictly maximal) elements are defined analogously.
- **Notation**: $\prec$ for the inverse relation $\succ^{-1}$
  $\succeq$ for the reflexive closure $(\succ \cup =)$ of $\succ$, i.e.
  $x \succeq y$ iff either $x \succ y$ or $x = y$

Orderings

- A (strict) ordering on a set X is a transitive and irreflexive binary relation on X, here denoted by $\succ$.
- The pair $(X, \succ)$ is then called a (strictly) ordered set.
- An element x of X is minimal wrt. $\succ$, if there is no y in X such that $x \succ y$.
- A minimal element x in X is called the smallest (or strictly minimal) element, if for all $y \in X$ different from x, $\;y \succ x$.
- Maximal and largest (or strictly maximal) elements are defined analogously.
- **Notation**: $\prec$ for the inverse relation $\succ^{-1}$
  $\succeq$ for the reflexive closure $(\succ \cup =)$ of $\succ$, i.e.
  $x \succeq y$ iff either $x \succ y$ or $x = y$

Orderings

- A (strict) ordering on a set X is a transitive and irreflexive binary relation on X, here denoted by $\succ$.
- The pair $(X, \succ)$ is then called a (strictly) ordered set.
- An element x of X is minimal wrt. $\succ$, if there is no y in X such that $x \succ y$.
- A minimal element x in X is called the smallest (or strictly minimal) element, if for all $y \in X$ different from x, $y \succ x$.
- Maximal and largest (or strictly maximal) elements are defined analogously.
- **Notation**: $\prec$ for the inverse relation $\succ^{-1}$

  $\succeq$ for the reflexive closure $(\succ \cup =)$ of $\succ$, i.e.

  $x \succeq y$ iff either $x \succ y$ or $x = y$

Orderings

- A (strict) ordering on a set X is a transitive and irreflexive binary relation on X, here denoted by $\succ$.
- The pair $(X, \succ)$ is then called a (strictly) ordered set.
- An element x of X is minimal wrt. $\succ$, if there is no y in X such that $x \succ y$.
- A minimal element x in X is called the smallest (or strictly minimal) element, if for all $y \in X$ different from x, $y \succ x$.
- Maximal and largest (or strictly maximal) elements are defined analogously.
- **Notation**: $\prec$ for the inverse relation $\succ^{-1}$

  $\succeq$ for the reflexive closure $(\succ \cup =)$ of $\succ$, i.e.

  $x \succeq y$ iff either $x \succ y$ or $x = y$

Orderings

- A (strict) ordering on a set X is a transitive and irreflexive binary relation on X, here denoted by $\succ$.
- The pair $(X, \succ)$ is then called a (strictly) ordered set.
- An element x of X is minimal wrt. $\succ$, if there is no y in X such that $x \succ y$.
- A minimal element x in X is called the smallest (or strictly minimal) element, if for all $y \in X$ different from x, $y \succ x$.
- Maximal and largest (or strictly maximal) elements are defined analogously.
- **Notation**: $\prec$ for the inverse relation $\succ^{-1}$
  $\succeq$ for the reflexive closure $(\succ \cup =)$ of $\succ$, i.e.
  $x \succeq y$ iff either $x \succ y$ or $x = y$

Well-Foundedness

A (strict) ordering $\succ$ over X is called well-founded (or Noetherian or terminating), if there is no infinite decreasing chain $x_0 \succ x_1 \succ x_2 \succ \ldots$ of elements $x_i \in X$.

Emmy Noether

Lemma

$(X, \succ)$ is well-founded iff every non-empty subset Y of X has a minimal element.

Which of those orderings is well-founded?:

- $(\mathbb{N}, >)$
- $(\mathbb{Z}, >)$
- $(\mathbb{Z}^{\geq 0}, >)$
- $(\mathbb{Q}, >)$,
- $(\mathbb{R}, >)$,
- $(\mathbb{R}^{\geq 0}, >)$

Well-Foundedness

A (strict) ordering $\succ$ over X is called well-founded (or Noetherian or terminating), if there is no infinite decreasing chain $x_0 \succ x_1 \succ x_2 \succ \ldots$ of elements $x_i \in X$.

Emmy Noether

Lemma

$(X, \succ)$ is well-founded iff every non-empty subset Y of X has a minimal element.

Which of those orderings is well-founded?:

- $(\mathbb{N}, >)$
- $(\mathbb{Z}, >)$
- $(\mathbb{Z}^{\geq 0}, >)$
- $(\mathbb{Q}, >)$,
- $(\mathbb{R}, >)$,
- $(\mathbb{R}^{\geq 0}, >)$

Well-Foundedness

A (strict) ordering $\succ$ over X is called well-founded (or Noetherian or terminating), if there is no infinite decreasing chain $x_0 \succ x_1 \succ x_2 \succ \ldots$ of elements $x_i \in X$.


Emmy Noether

Lemma

$(X, \succ)$ is well-founded iff every non-empty subset Y of X has a minimal element.

Which of those orderings is well-founded?:

- $(\mathbb{N}, >)$
- $(\mathbb{Z}, >)$
- $(\mathbb{Z}^{\geq 0}, >)$
- $(\mathbb{Q}, >)$,
- $(\mathbb{R}, >)$,
- $(\mathbb{R}^{\geq 0}, >)$

Well-Foundedness

A (strict) ordering $\succ$ over X is called well-founded (or Noetherian or terminating), if there is no infinite decreasing chain $x_0 \succ x_1 \succ x_2 \succ \ldots$ of elements $x_i \in X$.

Emmy Noether

Lemma

$(X, \succ)$ is well-founded iff every non-empty subset Y of X has a minimal element.

Which of those orderings is well-founded?:

- $(\mathbb{N}, >)$
- $(\mathbb{Z}, >)$
- $(\mathbb{Z}^{\geq 0}, >)$
- $(\mathbb{Q}, >)$,
- $(\mathbb{R}, >)$,
- $(\mathbb{R}^{\geq 0}, >)$

Well-Foundedness

A (strict) ordering $\succ$ over X is called well-founded (or Noetherian or terminating), if there is no infinite decreasing chain $x_0 \succ x_1 \succ x_2 \succ \ldots$ of elements $x_i \in X$.

Emmy Noether

Lemma

$(X, \succ)$ is well-founded iff every non-empty subset Y of X has a minimal element.

Which of those orderings is well-founded?:

- $(\mathbb{N}, >)$
- $(\mathbb{Z}, >)$
- $(\mathbb{Z}^{\geq 0}, >)$
- $(\mathbb{Q}, >)$,
- $(\mathbb{R}, >)$,
- $(\mathbb{R}^{\geq 0}, >)$

Well-Foundedness

A (strict) ordering $\succ$ over X is called well-founded (or Noetherian or terminating), if there is no infinite decreasing chain $x_0 \succ x_1 \succ x_2 \succ \ldots$ of elements $x_i \in X$.

Emmy Noether

Lemma

$(X, \succ)$ is well-founded iff every non-empty subset Y of X has a minimal element.

Which of those orderings is well-founded?:

- $(\mathbb{N}, >)$
- $(\mathbb{Z}, >)$
- $(\mathbb{Z}^{\geq 0}, >)$
- $(\mathbb{Q}, >)$,
- $(\mathbb{R}, >)$,
- $(\mathbb{R}^{\geq 0}, >)$

Well-Foundedness

A (strict) ordering $\succ$ over X is called well-founded (or Noetherian or terminating), if there is no infinite decreasing chain $x_0 \succ x_1 \succ x_2 \succ \ldots$ of elements $x_i \in X$.

Emmy Noether

Lemma

$(X, \succ)$ is well-founded iff every non-empty subset Y of X has a minimal element.

Which of those orderings is well-founded?:

- $(\mathbb{N}, >)$
- $(\mathbb{Z}, >)$
- $(\mathbb{Z}^{\geq 0}, >)$
- $(\mathbb{Q}, >)$,
- $(\mathbb{R}, >)$,
- $(\mathbb{R}^{\geq 0}, >)$

Well-Foundedness

A (strict) ordering $\succ$ over X is called well-founded (or Noetherian or terminating), if there is no infinite decreasing chain $x_0 \succ x_1 \succ x_2 \succ \ldots$ of elements $x_i \in X$.

Emmy Noether

Lemma

$(X, \succ)$ is well-founded iff every non-empty subset Y of X has a minimal element.

Which of those orderings is well-founded?:

- $(\mathbb{N}, >)$
- $(\mathbb{Z}, >)$
- $(\mathbb{Z}^{\geq 0}, >)$
- $(\mathbb{Q}, >)$,
- $(\mathbb{R}, >)$,
- $(\mathbb{R}^{\geq 0}, >)$

Transition Relation

A binary relation $\Rightarrow \subseteq S \times S$ on a set (of states) S is called a transition relation.

Example: Consider a program P in an imperative language.

- The program state is defined by assigning values to all variables (including the program counter).
- P defines a transition relation on the set of states.

Definition. A transition relation $\Rightarrow$ on S is

- terminating if there is no infinite $s_1 \Rightarrow s_2 \Rightarrow \ldots \Rightarrow s_n \Rightarrow \ldots$.
- compatible with an ordering $\succ$ on S if $\Rightarrow \subseteq \succ$.

Lemma. A transition relation $\Rightarrow$ is terminating if and only if there is a well-founded ordering $\succ$ compatible with $\Rightarrow$.

State-of-the-art methods for proving termination of a programs are based on finding suitable well-founded ordering compatible with the program transition relation.

Transition Relation

A binary relation $\Rightarrow \subseteq S \times S$ on a set (of states) S is called a transition relation.

Example: Consider a program P in an imperative language.

- The program state is defined by assigning values to all variables (including the program counter).
- P defines a transition relation on the set of states.

Definition. A transition relation $\Rightarrow$ on S is

- terminating if there is no infinite $s_1 \Rightarrow s_2 \Rightarrow \ldots \Rightarrow s_n \Rightarrow \ldots$.
- compatible with an ordering $\succ$ on S if $\Rightarrow \subseteq \succ$.

Lemma. A transition relation $\Rightarrow$ is terminating if and only if there is a well-founded ordering $\succ$ compatible with $\Rightarrow$.

State-of-the-art methods for proving termination of a programs are based on finding suitable well-founded ordering compatible with the program transition relation.

Transition Relation

A binary relation $\Rightarrow \subseteq S \times S$ on a set (of states) S is called a transition relation.

Example: Consider a program P in an imperative language.

- The program state is defined by assigning values to all variables (including the program counter).
- P defines a transition relation on the set of states.

Definition. A transition relation $\Rightarrow$ on S is

- terminating if there is no infinite $s_1 \Rightarrow s_2 \Rightarrow \ldots \Rightarrow s_n \Rightarrow \ldots$.
- compatible with an ordering $\succ$ on S if $\Rightarrow \subseteq \succ$.

Lemma. A transition relation $\Rightarrow$ is terminating if and only if there is a well-founded ordering $\succ$ compatible with $\Rightarrow$.

State-of-the-art methods for proving termination of a programs are based on finding suitable well-founded ordering compatible with the program transition relation.

Transition Relation

A binary relation $\Rightarrow \subseteq S \times S$ on a set (of states) S is called a transition relation.

Example: Consider a program P in an imperative language.

- The program state is defined by assigning values to all variables (including the program counter).
- P defines a transition relation on the set of states.

Definition. A transition relation $\Rightarrow$ on S is

- terminating if there is no infinite $s_1 \Rightarrow s_2 \Rightarrow \ldots \Rightarrow s_n \Rightarrow \ldots$.
- compatible with an ordering $\succ$ on S if $\Rightarrow \subseteq \succ$.

Lemma. A transition relation $\Rightarrow$ is terminating if and only if there is a well-founded ordering $\succ$ compatible with $\Rightarrow$.

State-of-the-art methods for proving termination of a programs are based on finding suitable well-founded ordering compatible with the program transition relation.

Transition Relation

A binary relation $\Rightarrow \subseteq S \times S$ on a set (of states) S is called a transition relation.

Example: Consider a program P in an imperative language.

- The program state is defined by assigning values to all variables (including the program counter).
- P defines a transition relation on the set of states.

Definition. A transition relation $\Rightarrow$ on S is

- terminating if there is no infinite $s_1 \Rightarrow s_2 \Rightarrow \ldots \Rightarrow s_n \Rightarrow \ldots$.
- compatible with an ordering $\succ$ on S if $\Rightarrow \subseteq \succ$.

Lemma. A transition relation $\Rightarrow$ is terminating if and only if there is a well-founded ordering $\succ$ compatible with $\Rightarrow$.

State-of-the-art methods for proving termination of a programs are based on finding suitable well-founded ordering compatible with the program transition relation.

Noetherian Induction

Theorem (Noetherian Induction)

Let $(X, \succ)$ *be a well-founded ordering, let* Q *be a property of elements of* X.

If for all $x \in X$ *the following implication is satisfied*

> *if* $Q(y)$ *holds, for all* $y \in X$ *such that* $x \succ y$,[1]
> *then* $Q(x)$ *holds.*[2]

then

the property $Q(x)$ *holds for all* $x \in X$.

---

[1] induction hypothesis

[2] induction step/base case

Noetherian Induction (cont'd)

Proof.

By contradiction.

Thus, suppose for all $x \in X$ the implication above is satisfied, but $Q(x)$ does not hold for all $x \in X$.

Let $A = \{x \in X \mid Q(x) \text{ is false}\}$. Suppose $A \neq \emptyset$.

Since $(X, \succ)$ is well-founded, A has a minimal element x_1. Hence for all $y \in X$ with $x_1 \succ y$ the property $Q(y)$ holds.

On the other hand, the implication which is presupposed for this theorem holds in particular also for x_1, hence $Q(x_1)$ must be true so that x_1 cannot belong to A. *Contradiction*. □

Lexicographic Combination $\succ_{\text{lex}}$

Definition

Let $(X_1, \succ_1), (X_2, \succ_2)$ be two orderings.

Lexicographic combination of $(X_1, \succ_1), (X_2, \succ_2)$ is an ordering:

$$\succ_{\text{lex}} = (\succ_1, \succ_2)_{\text{lex}}$$

on $X_1 \times X_2$ such that

$(x_1, x_2) \succ_{\text{lex}} (y_1, y_2)$ iff (i) $x_1 \succ_1 y_1$, or else
(ii) $x_1 = y_1$ and $x_2 \succ_2 y_2$.

Note: We can iteratively combine any number of orderings.

Note: We combine an ordering with itself n times obtaining $\succ_{\text{lex}}^n$.

Lexicographic Combination $\succ_{\text{lex}}$

Definition

Let $(X_1, \succ_1), (X_2, \succ_2)$ be two orderings.

Lexicographic combination of $(X_1, \succ_1), (X_2, \succ_2)$ is an ordering:

$$\succ_{\text{lex}} = (\succ_1, \succ_2)_{\text{lex}}$$

on $X_1 \times X_2$ such that

$(x_1, x_2) \succ_{\text{lex}} (y_1, y_2)$ iff (i) $x_1 \succ_1 y_1$, or else
(ii) $x_1 = y_1$ and $x_2 \succ_2 y_2$.

Note: We can iteratively combine any number of orderings.

Note: We combine an ordering with itself n times obtaining $\succ_{\text{lex}}^n$.

Properties of Lexicographic Combination

Theorem

Let $(X_1, \succ_1)$ and $(X_2, \succ_2)$ be two orderings. Then

1. $\succ_{\mathrm{lex}}$ *is an ordering.*
2. *if both $\succ_1$ and $\succ_2$ well-founded then $\succ_{\mathrm{lex}}$ well-founded.*
3. *if both $\succ_1$ and $\succ_2$ total then $\succ_{\mathrm{lex}}$ total.*

Example: Lexicographic Combination

Example: Consider $(\mathbb{N}, >)$ then

$$(2, 5, 4) >^3_{\text{lex}} (1, 4, 3) >^3_{\text{lex}} (1, 3, 20)$$

Exercise: How many elements less than $(1, 2, 3)$?

Example: Lexicographic Combination

Example: Consider $(\mathbb{N}, >)$ then

$$(2,5,4) >^3_{\text{lex}} (1,4,3) >^3_{\text{lex}} (1,3,20)$$

Exercise: How many elements less than $(1,2,3)$?

Multi-Sets

- Multi-sets are "sets which allow repetition".
  E.g.: $\{a, a, b\}$, $\{a, b, a\}$, $\{a, b\}$
- Formally, let X be a set.
  A multi-set S over X is a mapping $S : X \to \mathbb{N}$.
- Intuitively, $S(x)$ specifies the number of occurrences of the element x (of the base set X) within S.
- **Example:** $S = \{a, a, a, b, b\}$ is a multi-set over $\{a, b, c\}$,
  where $S(a) = 3$, $S(b) = 2$, $S(c) = 0$.
- We say that x is an element of S, if $S(x) > 0$.

Multi-Sets (cont'd)

- We use set notation ($\in$, $\subset$, $\subseteq$, $\cup$, $\cap$, etc.) with analogous meaning also for multi-sets, e.g.,

$$\begin{aligned}
(S_1 \cup S_2)(x) &= S_1(x) + S_2(x) \\
(S_1 \cap S_2)(x) &= \min\{S_1(x), S_2(x)\} \\
(S_1 \backslash S_2)(x) &= S_1(x) \dot{-} S_2(x)
\end{aligned}$$

- A multi-set S over X is called finite, if

$$|\{x \in X |\ S(x) > 0\}| < \infty.$$

- From now on we consider finite multi-sets only.

Exercise

Suppose $S_1 = \{c, a, b\}$ and $S_2 = \{a, b, b, a\}$ are multi-sets over $\{a, b, c, d\}$.

Determine $S_1 \cup S_2$ and $S_1 \cap S_2$.

Exercise

Suppose $S_1 = \{c, a, b\}$ and $S_2 = \{a, b, b, a\}$ are multi-sets over $\{a, b, c, d\}$.
Determine $S_1 \cup S_2$ and $S_1 \cap S_2$.

Answer:

$$S_1 \cup S_2 = \{a, a, a, b, b, b, c\}$$

$$S_1 \cap S_2 = \{a, b\}$$

Multi-Set Orderings $\succ_{\mathrm{mul}}$

Definition

Let $(X, \succ)$ be an ordering. The multi-set extension $\succ_{\mathrm{mul}}$ of $\succ$ to (finite) multi-sets over X is defined by

$$S_1 \succ_{\mathrm{mul}} S_2 \iff S_1 \neq S_2 \text{ and } \forall x \in S_2 \backslash S_1.\ \exists y \in S_1 \backslash S_2.\ y \succ x$$

1. Remove common occurrences of elements from S_1 and S_2. Assume this gives $S_1' \neq S_2'$.
2. Then check that for every element x in S_2' there is an element $y \in S_1'$ that is greater than x. Then $S_1 \succ_{\mathrm{mul}} S_2$.

Multi-Set Orderings $\succ_{\mathrm{mul}}$

Definition

Let $(X, \succ)$ be an ordering. The multi-set extension $\succ_{\mathrm{mul}}$ of $\succ$ to (finite) multi-sets over X is defined by

$$
\begin{aligned}
S_1 \succ_{\mathrm{mul}} S_2 \iff\ & S_1 \neq S_2 \text{ and} \\
& \forall x \in S_2 \backslash S_1.\ \exists y \in S_1 \backslash S_2.\ y \succ x
\end{aligned}
$$

1. Remove common occurrences of elements from S_1 and S_2. Assume this gives $S_1' \neq S_2'$.
2. Then check that for every element x in S_2' there is an element $y \in S_1'$ that is greater than x. Then $S_1 \succ_{\mathrm{mul}} S_2$.

Example

- $S_1 = \{5, 5, 4, 3, 2\}$ $\quad$ $S_2 = \{5, 4, 4, 3, 3, 2\}$
  $S_1' = \{5\}$ $\quad$ $S_2' = \{4, 3\}$
  $5 > 4$ and $5 > 3$
  Therefore $S_1 >_{\mathrm{mul}} S_2$.
- $S_2 = \{5, 4, 4, 3, 3, 2\}$ $\quad$ $S_3 = \{5, 4, 3\}$
  $S_2' = \{4, 3, 2\}$ $\quad$ $S_3' = \emptyset$
  Therefore $S_2 >_{\mathrm{mul}} S_3$.
- Exercise: How does $S_4 = \{5, 3, 2\}$ compare with S_3?

Example

- $S_1 = \{\cancel{5}, 5, \cancel{4}, \cancel{3}, \cancel{2}\}$ $S_2 = \{\cancel{5}, \cancel{4}, 4, \cancel{3}, 3, \cancel{2}\}$
  $S_1' = \{5\}$ $S_2' = \{4, 3\}$
  $5 > 4$ and $5 > 3$
  Therefore $S_1 >_{\mathrm{mul}} S_2$.

- $S_2 = \{5, 4, 4, 3, 3, 2\}$ $S_3 = \{5, 4, 3\}$
  $S_2' = \{4, 3, 2\}$ $S_3' = \emptyset$
  Therefore $S_2 >_{\mathrm{mul}} S_3$.

- Exercise: How does $S_4 = \{5, 3, 2\}$ compare with S_3?

Example

- $S_1 = \{\cancel{5}, 5, \cancel{4}, \cancel{3}, \cancel{2}\}$ $\qquad$ $S_2 = \{\cancel{5}, \cancel{4}, 4, \cancel{3}, 3, \cancel{2}\}$
  $S_1' = \{5\}$ $\qquad$ $S_2' = \{4, 3\}$
  $5 > 4$ and $5 > 3$
  Therefore $S_1 >_{\mathrm{mul}} S_2$.

- $S_2 = \{5, 4, 4, 3, 3, 2\}$ $\qquad$ $S_3 = \{5, 4, 3\}$
  $S_2' = \{4, 3, 2\}$ $\qquad$ $S_3' = \emptyset$
  Therefore $S_2 >_{\mathrm{mul}} S_3$.

- Exercise: How does $S_4 = \{5, 3, 2\}$ compare with S_3?

Example

- $S_1 = \{\cancel{5}, 5, \cancel{4}, \cancel{3}, \cancel{2}\}$ $\quad$ $S_2 = \{\cancel{5}, \cancel{4}, 4, \cancel{3}, 3, \cancel{2}\}$
  $S_1' = \{5\}$ $\quad$ $S_2' = \{4, 3\}$
  $5 > 4$ and $5 > 3$
  Therefore $S_1 >_{\mathrm{mul}} S_2$.
- $S_2 = \{\cancel{5}, \cancel{4}, 4, \cancel{3}, 3, 2\}$ $\quad$ $S_3 = \{\cancel{5}, \cancel{4}, \cancel{3}\}$
  $S_2' = \{4, 3, 2\}$ $\quad$ $S_3' = \emptyset$
  Therefore $S_2 >_{\mathrm{mul}} S_3$.
- Exercise: How does $S_4 = \{5, 3, 2\}$ compare with S_3?

Example

- $S_1 = \{\cancel{5}, 5, \cancel{4}, \cancel{3}, \cancel{2}\}$ $\qquad$ $S_2 = \{\cancel{5}, \cancel{4}, 4, \cancel{3}, 3, \cancel{2}\}$
  $S_1' = \{5\}$ $\qquad$ $S_2' = \{4, 3\}$
  $5 > 4$ and $5 > 3$

  Therefore $S_1 >_{\mathrm{mul}} S_2$.

- $S_2 = \{\cancel{5}, \cancel{4}, 4, \cancel{3}, 3, 2\}$ $\qquad$ $S_3 = \{\cancel{5}, \cancel{4}, \cancel{3}\}$
  $S_2' = \{4, 3, 2\}$ $\qquad$ $S_3' = \emptyset$
  Therefore $S_2 >_{\mathrm{mul}} S_3$.

- Exercise: How does $S_4 = \{5, 3, 2\}$ compare with S_3?

Example

- $S_1 = \{\cancel{5}, 5, \cancel{4}, \cancel{3}, \cancel{2}\}$ $\quad$ $S_2 = \{\cancel{5}, \cancel{4}, 4, \cancel{3}, 3, \cancel{2}\}$
  $S_1' = \{5\}$ $\quad$ $S_2' = \{4, 3\}$
  $5 > 4$ and $5 > 3$
  Therefore $S_1 >_{\mathrm{mul}} S_2$.

- $S_2 = \{\cancel{5}, \cancel{4}, 4, \cancel{3}, 3, 2\}$ $\quad$ $S_3 = \{\cancel{5}, \cancel{4}, \cancel{3}\}$
  $S_2' = \{4, 3, 2\}$ $\quad$ $S_3' = \emptyset$
  Therefore $S_2 >_{\mathrm{mul}} S_3$.

- Exercise: How does $S_4 = \{5, 3, 2\}$ compare with S_3?

  Answer: $S_4 = \{\cancel{5}, \cancel{3}, 2\}$ $\quad$ $S_3 = \{\cancel{5}, 4, \cancel{3}\}$
  $S_4' = \{2\}$ $\quad$ $S_3' = \{4\}$

Example

- $S_1 = \{\cancel{5}, 5, \cancel{4}, \cancel{3}, \cancel{2}\}$ $\quad$ $S_2 = \{\cancel{5}, \cancel{4}, 4, \cancel{3}, 3, \cancel{2}\}$
  $S'_1 = \{5\}$ $\quad$ $S'_2 = \{4, 3\}$
  $5 > 4$ and $5 > 3$

  Therefore $S_1 >_{\mathrm{mul}} S_2$.

- $S_2 = \{\cancel{5}, \cancel{4}, 4, \cancel{3}, 3, 2\}$ $\quad$ $S_3 = \{\cancel{5}, \cancel{4}, \cancel{3}\}$
  $S'_2 = \{4, 3, 2\}$ $\quad$ $S'_3 = \emptyset$
  Therefore $S_2 >_{\mathrm{mul}} S_3$.

- Exercise: How does $S_4 = \{5, 3, 2\}$ compare with S_3?

  Answer: $S_4 = \{\cancel{5}, \cancel{3}, 2\}$ $\quad$ $S_3 = \{\cancel{5}, 4, \cancel{3}\}$
  $S'_4 = \{2\}$ $\quad$ $S'_3 = \{4\}$
  Therefore $S_3 >_{\mathrm{mul}} S_4$.

Properties of Multi-Set Orderings

Theorem

Let $\succ$ be an ordering. Then

1. $\succ_{\mathrm{mul}}$ *is an ordering.*
2. *if* $\succ$ *well-founded then* $\succ_{\mathrm{mul}}$ *well-founded.*
3. *if* $\succ$ *total then* $\succ_{\mathrm{mul}}$ *total*

Exercise: How many multi-sets less than $\{3\}$?

Properties of Multi-Set Orderings

Theorem

Let $\succ$ *be an ordering. Then*

1. $\succ_{\mathrm{mul}}$ *is an ordering.*
2. *if* $\succ$ *well-founded then* $\succ_{\mathrm{mul}}$ *well-founded.*
3. *if* $\succ$ *total then* $\succ_{\mathrm{mul}}$ *total*

Exercise: How many multi-sets less than $\{3\}$?

Summary

- (strict) orderings
- well-founded orderings
- Noetherian (well-founded) induction
- multi-sets
- multi-set ordering $\succ_{\mathrm{mul}}$

  $=$ multi-set extension of ordering $\succ$ on the elements

Section 3 Propositional Logic: Syntax and Semantics

What is logic?

- Syntax: formal language
- Semantics: meaning for the language
- Reasoning:
  - Proof theory
  - Model theory

Why Propositional Logic?

- Propositional logic is one of the simplest logics
- Propositional logic has direct applications e.g. circuit design
- There are efficient algorithms for reasoning in propositional logic
- Propositional logic is a foundation for most of the more expressive logics

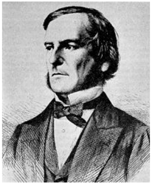
Our next goal is to study properties of propositional formulas and devise algorithms for reasoning in propositional logic.

Propositional (Boolean) Logic

Example: "If I study hard and I complete all assignments then I will get a good grade."

Atomic propositions (can be true or false):

- I study hard
- I complete all assignments
- I will get a good grade



George Boole

From atomic propositions we can construct more complex propositions (formulas) using Boolean connectives (and, or, not,...).

Next: Syntax and Semantics

Syntax: Propositional Formulas

Propositional (Boolean) variables usually denoted as $p, q, s, \ldots$.

Connectives: $\wedge$ "and", $\vee$ "or", $\neg$ "not", $\rightarrow$ "implies", $\leftrightarrow$ "equivalent"

Propositional formula:

- Every propositional variable is a formula, also called atomic formula, or simply atom.
- $\top$ (true) and $\perp$ (false) are formulas.
- If $A_1, \ldots, A_n$ are formulas, where $n \geq 2$, then $(A_1 \wedge \ldots \wedge A_n)$ and $(A_1 \vee \ldots \vee A_n)$ are formulas.
- If A is a formula, then $\neg A$ is a formula.
- If A and B are formulas, then $(A \rightarrow B)$ and $(A \leftrightarrow B)$ are formulas.

Subformulas

Example: $((p \wedge q) \rightarrow (q \vee \neg p \vee s))$

Immediate Subformulas:

$(p \wedge q)$ and $(q \vee \neg p \vee s)$

Subformulas:

$((p \wedge q) \rightarrow (q \vee \neg p \vee s))$;

$(p \wedge q)$ and $(q \vee \neg p \vee s)$;

p; q; $\neg p$; s

Notation: $A[B]$ means B occurs in A as a subformula.

Connectives

Example: $((p \land q) \to (q \lor \neg p \lor s))$ (too many brackets...)

Connective	Name	Priority
$\neg$	negation	5
$\lor$	disjunction	4
$\land$	conjunction	3
$\to$	implication	2
$\leftrightarrow$	equivalence	1

Now we can replace

$((p \land q) \to (q \lor \neg p \lor s))$ with $p \land q \to q \lor \neg p \lor s$.

Semantics: Interpretation

An interpretation I assigns truth values to propositional variables

$$I : P \to \{\mathbf{1}, \mathbf{0}\}$$

$\mathbf{1}, \mathbf{0}$ are called truth values or also boolean values.

- If $I(p) = \mathbf{1}$, then p is called true in I.
- If $I(p) = \mathbf{0}$, then p is called false in I.

Interpretations are also called truth assignments.

Example: $I(p) = \mathbf{0}$; $I(q) = \mathbf{1}$; $I(s) = \mathbf{0}$

Truth value

Extend I to all formulas:

1. $I(\top) = \mathbf{1}$ and $I(\bot) = \mathbf{0}$.
2. $I(A_1 \wedge \ldots \wedge A_n) = \mathbf{1}$ if and only if $I(A_i) = \mathbf{1}$ for all i.
3. $I(A_1 \vee \ldots \vee A_n) = \mathbf{1}$ if and only if $I(A_i) = \mathbf{1}$ for some i.
4. $I(\neg A) = \mathbf{1}$ if and only if $I(A) = \mathbf{0}$.
5. $I(A \to B) = \mathbf{1}$ if and only if $I(A) = \mathbf{0}$ or $I(B) = \mathbf{1}$.
6. $I(A \leftrightarrow B) = \mathbf{1}$ if and only if $I(A) = I(B)$.

Notation: $I \models A$ if $I(A) = \mathbf{1}$ (A is true in I)
$I \not\models A$ if $I(A) = \mathbf{0}$ (A is false in I)

Truth Tables

A	B	$A \wedge B$	$A \vee B$	$\neg A$	$A \rightarrow B$	$A \leftrightarrow B$
0	**0**	0	0	1	1	1
1	**0**	0	1	0	0	0
0	**1**	0	1	1	1	0
1	**1**	1	1	0	1	1

Truth Tables

A	B	$A \wedge B$	$A \vee B$	$\neg A$	$A \rightarrow B$	$A \leftrightarrow B$
0	**0**	**0**	**0**	**1**	**1**	**1**
1	**0**	**0**	**1**	**0**	**0**	**0**
0	**1**	**0**	**1**	**1**	**1**	**0**
1	**1**	**1**	**1**	**0**	**1**	**1**

Truth Tables

A	B	$A \wedge B$	$A \vee B$	$\neg A$	$A \rightarrow B$	$A \leftrightarrow B$
0	**0**	**0**	**0**	1	1	1
1	**0**	**0**	**1**	0	0	0
0	**1**	**0**	**1**	1	1	0
1	**1**	**1**	**1**	0	1	1

Truth Tables

A	B	$A \wedge B$	$A \vee B$	$\neg A$	$A \rightarrow B$	$A \leftrightarrow B$
0	**0**	**0**	**0**	**1**	1	1
1	**0**	**0**	**1**	**0**	0	0
0	**1**	**0**	**1**	**1**	1	0
1	**1**	**1**	**1**	**0**	1	1

Truth Tables

A	B	$A \wedge B$	$A \vee B$	$\neg A$	$A \rightarrow B$	$A \leftrightarrow B$
0	0	0	0	1	1	1
1	0	0	1	0	0	0
0	1	0	1	1	1	0
1	1	1	1	0	1	1

Truth Tables

A	B	$A \wedge B$	$A \vee B$	$\neg A$	$A \rightarrow B$	$A \leftrightarrow B$
0	**0**	**0**	**0**	**1**	**1**	**1**
1	**0**	**0**	**1**	**0**	**0**	**0**
0	**1**	**0**	**1**	**1**	**1**	**0**
1	**1**	**1**	**1**	**0**	**1**	**1**

Operation tables

$\wedge$	1	0
1	1	0
0	0	0

$\vee$	1	0
1	1	1
0	1	0

$\neg$	
1	0
0	1

$\rightarrow$	1	0
1	1	0
0	1	1

$\leftrightarrow$	1	0
1	1	0
0	0	1

How to evaluate a formula?

Let's evaluate the formula

$$(p \to q) \land (p \land q \to r) \to (p \to r)$$

in the interpretation

$$I = \{p \mapsto \mathbf{1}, q \mapsto \mathbf{0}, r \mapsto \mathbf{1}\}.$$

Evaluating a formula.

	formula	value
1	$(p \to q) \wedge (p \wedge q \to r) \to (p \to r)$	
2	$p \to r$	
3	$(p \to q) \wedge (p \wedge q \to r)$	
4	$p \wedge q \to r$	
5	$p \to q$	
6	$p \wedge q$	
7	$p \quad p \quad p$	
8	$q \quad q$	
9	$r \quad r$	

Evaluating a formula.

	formula	value
1	$(p \to q) \land (p \land q \to r) \to (p \to r)$	
2	$p \to r$	
3	$(p \to q) \land (p \land q \to r)$	
4	$p \land q \to r$	
5	$p \to q$	
6	$p \land q$	
7	p $\quad$ p $\quad$ p	
8	q $\quad$ q	
9	r $\quad$ r	

Evaluating a formula.

	formula	value
1	$(p \to q) \land (p \land q \to r) \to (p \to r)$	
2	$p \to r$	
3	$(p \to q) \land (p \land q \to r)$	
4	$p \land q \to r$	
5	$p \to q$	
6	$p \land q$	
7	p $\quad$ p $\quad$ p	
8	q $\quad$ q	
9	r $\quad$ r	

Evaluating a formula.

	formula	value
1	$(p \to q) \land (p \land q \to r) \to (p \to r)$	
2	$p \to r$	
3	$(p \to q) \land (p \land q \to r)$	
4	$p \land q \to r$	
5	$p \to q$	
6	$p \land q$	
7	p p p	
8	q q	
9	r r	

Evaluating a formula.

	formula	value
1	$(p \to q) \land (p \land q \to r) \to (p \to r)$	
2	$p \to r$	
3	$(p \to q) \land (p \land q \to r)$	
4	$p \land q \to r$	
5	$p \to q$	
6	$p \land q$	
7	$p \quad p \quad p$	
8	$q \quad q$	
9	$r \quad r$	

Evaluating a formula.

	formula	value
1	$(p \to q) \wedge (p \wedge q \to r) \to (p \to r)$	
2	$p \to r$	
3	$(p \to q) \wedge (p \wedge q \to r)$	
4	$p \wedge q \to r$	
5	$p \to q$	
6	$p \wedge q$	
7	p p p	
8	q q	
9	r r	

Evaluating a formula.

	formula	value
1	$(p \to q) \land (p \land q \to r) \to (p \to r)$	
2	$p \to r$	
3	$(p \to q) \land (p \land q \to r)$	
4	$p \land q \to r$	
5	$p \to q$	
6	$p \land q$	
7	p p p	
8	q q	
9	r r	

Evaluating a formula.

	formula	value
1	$(p \to q) \land (p \land q \to r) \to (p \to r)$	
2	$p \to r$	
3	$(p \to q) \land (p \land q \to r)$	
4	$p \land q \to r$	
5	$p \to q$	
6	$p \land q$	
7	p p p	
8	q q	
9	r r	

Evaluating a formula.

	formula	value
1	$(p \to q) \land (p \land q \to r) \to (p \to r)$	
2	$p \to r$	
3	$(p \to q) \land (p \land q \to r)$	
4	$p \land q \to r$	
5	$p \to q$	
6	$p \land q$	
7	$p \quad p \quad p$	
8	$q \quad q$	
9	$r \quad r$	

Evaluating a formula.

	formula	value
1	$(p \to q) \wedge (p \wedge q \to r) \to (p \to r)$	
2	$p \to r$	
3	$(p \to q) \wedge (p \wedge q \to r)$	
4	$p \wedge q \to r$	
5	$p \to q$	
6	$p \wedge q$	
7	$p \quad p \quad p$	**1**
8	$q \quad q$	**0**
9	$r \quad r$	**1**

Evaluating a formula.

	formula	value
1	$(p \to q) \land (p \land q \to r) \to (p \to r)$	
2	$p \to r$	
3	$(p \to q) \land (p \land q \to r)$	
4	$p \land q \to r$	
5	$p \to q$	
6	$p \land q$	**0**
7	p p p	**1**
8	q q	**0**
9	r r	**1**

Evaluating a formula.

	formula	value
1	$(p \to q) \wedge (p \wedge q \to r) \to (p \to r)$	
2	$p \to r$	
3	$(p \to q) \wedge (p \wedge q \to r)$	
4	$p \wedge q \to r$	
5	$p \to q$	**0**
6	$p \wedge q$	**0**
7	p p p	**1**
8	q q	**0**
9	r r	**1**

Evaluating a formula.

	formula	value
1	$(p \to q) \land (p \land q \to r) \to (p \to r)$	
2	$p \to r$	
3	$(p \to q) \land (p \land q \to r)$	
4	$p \land q \to r$	**1**
5	$p \to q$	**0**
6	$p \land q$	**0**
7	$p \quad p \quad p$	**1**
8	$q \quad q$	**0**
9	$r \quad r$	**1**

Evaluating a formula.

	formula	value
1	$(p \to q) \wedge (p \wedge q \to r) \to (p \to r)$	
2	$p \to r$	
3	$(p \to q) \wedge (p \wedge q \to r)$	**0**
4	$p \wedge q \to r$	**1**
5	$p \to q$	**0**
6	$p \wedge q$	**0**
7	$p \quad p \quad p$	**1**
8	$q \quad q$	**0**
9	$r \quad r$	**1**

Evaluating a formula.

	formula	value
1	$(p \to q) \wedge (p \wedge q \to r) \to (p \to r)$	
2	$p \to r$	$\mathbf{1}$
3	$(p \to q) \wedge (p \wedge q \to r)$	$\mathbf{0}$
4	$p \wedge q \to r$	$\mathbf{1}$
5	$p \to q$	$\mathbf{0}$
6	$p \wedge q$	$\mathbf{0}$
7	p $\quad$ p $\quad$ p	$\mathbf{1}$
8	q $\quad$ q	$\mathbf{0}$
9	r $\quad$ r	$\mathbf{1}$

Evaluating a formula.

	formula	value
1	$(p \to q) \land (p \land q \to r) \to (p \to r)$	**1**
2	$p \to r$	**1**
3	$(p \to q) \land (p \land q \to r)$	**0**
4	$p \land q \to r$	**1**
5	$p \to q$	**0**
6	$p \land q$	**0**
7	$p \quad p \quad p$	**1**
8	$q \quad q$	**0**
9	$r \quad r$	**1**

Summary

We started studying propositional logic:

- Syntax – propositional formulas
- Semantics – interpretations assigning truth values

Next: satisfiability, validity, equivalence

Satisfiability, validity

- If a formula A is true in I we say that I *satisfies* A and that I is a model of A, denoted by $I \models A$.
- A is *satisfiable* if A is true in some interpretation.
- A is *unsatisfiable* (denoted $A \models \perp$) if A is false in all interpretations.
- A is *valid* (or a tautology) if A true in every interpretation (denoted $\models A$).
- A formula A entails B, (denoted $A \models B$) if all models of A are models of B.
- Two formulas A and B are called equivalent, (denoted $A \equiv B$) if they have the same models.

We will study: algorithms for evaluating formulas on interpretations, for checking satisfiability, validity and equivalence of formulas.

Satisfiability, validity

- If a formula A is true in I we say that I *satisfies* A and that I is a model of A, denoted by $I \models A$.
- A is *satisfiable* if A is true in some interpretation.
- A is *unsatisfiable* (denoted $A \models \bot$) if A is false in all interpretations.
- A is *valid* (or a tautology) if A true in every interpretation (denoted $\models A$).
- A formula A entails B, (denoted $A \models B$) if all models of A are models of B.
- Two formulas A and B are called equivalent, (denoted $A \equiv B$) if they have the same models.

We will study: algorithms for evaluating formulas on interpretations, for checking satisfiability, validity and equivalence of formulas.

Satisfiability, validity

- If a formula A is true in I we say that I *satisfies* A and that I is a model of A, denoted by $I \models A$.
- A is *satisfiable* if A is true in some interpretation.
- A is *unsatisfiable* (denoted $A \models \bot$) if A is false in all interpretations.
- A is *valid* (or a tautology) if A true in every interpretation (denoted $\models A$).
- A formula A entails B, (denoted $A \models B$) if all models of A are models of B.
- Two formulas A and B are called equivalent, (denoted $A \equiv B$) if they have the same models.

We will study: algorithms for evaluating formulas on interpretations, for checking satisfiability, validity and equivalence of formulas.

Satisfiability, validity

- If a formula A is true in I we say that I *satisfies* A and that I is a model of A, denoted by $I \models A$.
- A is *satisfiable* if A is true in some interpretation.
- A is *unsatisfiable* (denoted $A \models \perp$) if A is false in all interpretations.
- A is *valid* (or a tautology) if A true in every interpretation (denoted $\models A$).
- A formula A entails B, (denoted $A \models B$) if all models of A are models of B.
- Two formulas A and B are called equivalent, (denoted $A \equiv B$) if they have the same models.

We will study: algorithms for evaluating formulas on interpretations, for checking satisfiability, validity and equivalence of formulas.

Satisfiability, validity

- If a formula A is true in I we say that I *satisfies* A and that I is a model of A, denoted by $I \models A$.
- A is *satisfiable* if A is true in some interpretation.
- A is *unsatisfiable* (denoted $A \models \perp$) if A is false in all interpretations.
- A is *valid* (or a tautology) if A true in every interpretation (denoted $\models A$).
- A formula A entails B, (denoted $A \models B$) if all models of A are models of B.
- Two formulas A and B are called equivalent, (denoted $A \equiv B$) if they have the same models.

We will study: algorithms for evaluating formulas on interpretations, for checking satisfiability, validity and equivalence of formulas.

Satisfiability, validity

- If a formula A is true in I we say that I *satisfies* A and that I is a model of A, denoted by $I \models A$.
- A is *satisfiable* if A is true in some interpretation.
- A is *unsatisfiable* (denoted $A \models \perp$) if A is false in all interpretations.
- A is *valid* (or a tautology) if A true in every interpretation (denoted $\models A$).
- A formula A entails B, (denoted $A \models B$) if all models of A are models of B.
- Two formulas A and B are called equivalent, (denoted $A \equiv B$) if they have the same models.

We will study: algorithms for evaluating formulas on interpretations, for checking satisfiability, validity and equivalence of formulas.

Satisfiability, validity

- If a formula A is true in I we say that I *satisfies* A and that I is a model of A, denoted by $I \models A$.
- A is *satisfiable* if A is true in some interpretation.
- A is *unsatisfiable* (denoted $A \models \bot$) if A is false in all interpretations.
- A is *valid* (or a tautology) if A true in every interpretation (denoted $\models A$).
- A formula A entails B, (denoted $A \models B$) if all models of A are models of B.
- Two formulas A and B are called equivalent, (denoted $A \equiv B$) if they have the same models.

We will study: algorithms for evaluating formulas on interpretations, for checking satisfiability, validity and equivalence of formulas.

Satisfiability, validity

- If a formula A is true in I we say that I *satisfies* A and that I is a model of A, denoted by $I \models A$.
- A is *satisfiable* if A is true in some interpretation.
- A is *unsatisfiable* (denoted $A \models \perp$) if A is false in all interpretations.
- A is *valid* (or a tautology) if A true in every interpretation (denoted $\models A$).
- A formula A entails B, (denoted $A \models B$) if all models of A are models of B.
- Two formulas A and B are called equivalent, (denoted $A \equiv B$) if they have the same models.

We will study: algorithms for evaluating formulas on interpretations, for checking satisfiability, validity and equivalence of formulas.

Truth Tables

Consider $A = p \wedge \neg q \to q \vee \neg p$.

We know how to calculate the truth value of A in a given interpretation I.

If $I = \{p \mapsto \mathbf{0}; q \mapsto \mathbf{0}\}$ then

Now we consider all possible interpretations:

p	q	$p \wedge \neg q \to q \vee \neg p$
$\mathbf{0}$	$\mathbf{0}$	
$\mathbf{0}$	$\mathbf{1}$	
$\mathbf{1}$	$\mathbf{0}$	
$\mathbf{1}$	$\mathbf{1}$	

Truth Tables

Consider $A = p \land \neg q \to q \lor \neg p$.

We know how to calculate the truth value of A in a given interpretation I.

If $I = \{p \mapsto \mathbf{0}; q \mapsto \mathbf{0}\}$ then $I(A) = \mathbf{1}$.

Now we consider all possible interpretations:

p	q	$p \land \neg q \to q \lor \neg p$
$\mathbf{0}$	$\mathbf{0}$	
$\mathbf{0}$	$\mathbf{1}$	
$\mathbf{1}$	$\mathbf{0}$	
$\mathbf{1}$	$\mathbf{1}$	

Is this formula satisfiable?

Is this formula valid?

Truth Tables

Consider $A = p \land \neg q \to q \lor \neg p$.

We know how to calculate the truth value of A in a given interpretation I.

If $I = \{p \mapsto \mathbf{0}; q \mapsto \mathbf{0}\}$ then $I(A) = \mathbf{1}$.

Now we consider all possible interpretations:

p	q	$p \land \neg q \to q \lor \neg p$
$\mathbf{0}$	$\mathbf{0}$	
$\mathbf{0}$	$\mathbf{1}$	
$\mathbf{1}$	$\mathbf{0}$	
$\mathbf{1}$	$\mathbf{1}$	

Is this formula satisfiable?

Is this formula valid?

Truth Tables

Consider $A = p \wedge \neg q \rightarrow q \vee \neg p$.

We know how to calculate the truth value of A in a given interpretation I.

If $I = \{p \mapsto \mathbf{0}; q \mapsto \mathbf{0}\}$ then $I(A) = \mathbf{1}$.

Now we consider all possible interpretations:

p	q	$p \wedge \neg q \rightarrow q \vee \neg p$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{1}$
$\mathbf{0}$	$\mathbf{1}$	
$\mathbf{1}$	$\mathbf{0}$	
$\mathbf{1}$	$\mathbf{1}$	

Is this formula satisfiable?

Is this formula valid?

Truth Tables

Consider $A = p \wedge \neg q \rightarrow q \vee \neg p$.

We know how to calculate the truth value of A in a given interpretation I.

If $I = \{p \mapsto \mathbf{0}; q \mapsto \mathbf{0}\}$ then $I(A) = \mathbf{1}$.

Now we consider all possible interpretations:

p	q	$p \wedge \neg q \rightarrow q \vee \neg p$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{1}$
$\mathbf{0}$	$\mathbf{1}$	$\mathbf{1}$
$\mathbf{1}$	$\mathbf{0}$	
$\mathbf{1}$	$\mathbf{1}$	

Is this formula satisfiable?

Is this formula valid?

Truth Tables

Consider $A = p \wedge \neg q \to q \vee \neg p$.

We know how to calculate the truth value of A in a given interpretation I.

If $I = \{p \mapsto \mathbf{0}; q \mapsto \mathbf{0}\}$ then $I(A) = \mathbf{1}$.

Now we consider all possible interpretations:

p	q	$p \wedge \neg q \to q \vee \neg p$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{1}$
$\mathbf{0}$	$\mathbf{1}$	$\mathbf{1}$
$\mathbf{1}$	$\mathbf{0}$	$\mathbf{0}$
$\mathbf{1}$	$\mathbf{1}$	

Is this formula satisfiable?

Is this formula valid?

Truth Tables

Consider $A = p \land \neg q \rightarrow q \lor \neg p$.

We know how to calculate the truth value of A in a given interpretation I.

If $I = \{p \mapsto \mathbf{0}; q \mapsto \mathbf{0}\}$ then $I(A) = \mathbf{1}$.

Now we consider all possible interpretations:

p	q	$p \land \neg q \rightarrow q \lor \neg p$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{1}$
$\mathbf{0}$	$\mathbf{1}$	$\mathbf{1}$
$\mathbf{1}$	$\mathbf{0}$	$\mathbf{0}$
$\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$

Is this formula satisfiable?

Is this formula valid?

Truth Tables

Consider $A = p \land \neg q \rightarrow q \lor \neg p$.

We know how to calculate the truth value of A in a given interpretation I.

If $I = \{p \mapsto \mathbf{0}; q \mapsto \mathbf{0}\}$ then $I(A) = \mathbf{1}$.

Now we consider all possible interpretations:

p	q	$p \land \neg q \rightarrow q \lor \neg p$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{1}$
$\mathbf{0}$	$\mathbf{1}$	$\mathbf{1}$
$\mathbf{1}$	$\mathbf{0}$	$\mathbf{0}$
$\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$

Is this formula satisfiable?

Is this formula valid?

Truth Tables

Consider $A = p \wedge \neg q \rightarrow q \vee \neg p$.

We know how to calculate the truth value of A in a given interpretation I.

If $I = \{p \mapsto \mathbf{0}; q \mapsto \mathbf{0}\}$ then $I(A) = \mathbf{1}$.

Now we consider all possible interpretations:

p	q	$p \wedge \neg q \rightarrow q \vee \neg p$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{1}$
$\mathbf{0}$	$\mathbf{1}$	$\mathbf{1}$
$\mathbf{1}$	$\mathbf{0}$	$\mathbf{0}$
$\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$

Is this formula satisfiable? Yes

Is this formula valid?

Truth Tables

Consider $A = p \wedge \neg q \to q \vee \neg p$.

We know how to calculate the truth value of A in a given interpretation I.

If $I = \{p \mapsto \mathbf{0}; q \mapsto \mathbf{0}\}$ then $I(A) = \mathbf{1}$.

Now we consider all possible interpretations:

p	q	$p \wedge \neg q \to q \vee \neg p$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{1}$
$\mathbf{0}$	$\mathbf{1}$	$\mathbf{1}$
$\mathbf{1}$	$\mathbf{0}$	$\mathbf{0}$
$\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$

Is this formula satisfiable? Yes

Is this formula valid?

Truth Tables

Consider $A = p \land \neg q \to q \lor \neg p$.

We know how to calculate the truth value of A in a given interpretation I.

If $I = \{p \mapsto \mathbf{0}; q \mapsto \mathbf{0}\}$ then $I(A) = \mathbf{1}$.

Now we consider all possible interpretations:

p	q	$p \land \neg q \to q \lor \neg p$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{1}$
$\mathbf{0}$	$\mathbf{1}$	$\mathbf{1}$
$\mathbf{1}$	$\mathbf{0}$	$\mathbf{0}$
$\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$

Is this formula satisfiable? Yes

Is this formula valid? No

Truth Tables

p	q	$p \wedge \neg q \to q \vee \neg p$	$\neg p \vee q$
0	**0**	**1**	**1**
0	**1**	**1**	**1**
1	**0**	**0**	**0**
1	**1**	**1**	**1**

We have $p \wedge \neg q \to q \vee \neg p$ is equivalent to $\neg p \vee q$

Summary: Using truth tables we can check satisfiability, validity and equivalence.

Limitations: For modest number of variables truth tables are unacceptably huge!

Later: more practical algorithms.

Truth Tables

p	q	$p \land \neg q \to q \lor \neg p$	$\neg p \lor q$
0	**0**	**1**	**1**
0	**1**	**1**	**1**
1	**0**	**0**	**0**
1	**1**	**1**	**1**

We have $p \land \neg q \to q \lor \neg p$ is equivalent to $\neg p \lor q$

Summary: Using truth tables we can check satisfiability, validity and equivalence.

Limitations: For modest number of variables truth tables are unacceptably huge!

Later: more practical algorithms.

Truth Tables

p	q	$p \wedge \neg q \rightarrow q \vee \neg p$	$\neg p \vee q$
0	**0**	**1**	**1**
0	**1**	**1**	**1**
1	**0**	**0**	**0**
1	**1**	**1**	**1**

We have $p \wedge \neg q \rightarrow q \vee \neg p$ is equivalent to $\neg p \vee q$

Summary: Using truth tables we can check satisfiability, validity and equivalence.

Limitations: For modest number of variables truth tables are unacceptably huge!

Later: more practical algorithms.

Truth Tables

p	q	$p \wedge \neg q \rightarrow q \vee \neg p$	$\neg p \vee q$
0	**0**	**1**	**1**
0	**1**	**1**	**1**
1	**0**	**0**	**0**
1	**1**	**1**	**1**

We have $p \wedge \neg q \rightarrow q \vee \neg p$ is equivalent to $\neg p \vee q$

Summary: Using truth tables we can check satisfiability, validity and equivalence.

Limitations: For modest number of variables truth tables are unacceptably huge!

Later: more practical algorithms.

Truth Tables

p	q	$p \wedge \neg q \to q \vee \neg p$	$\neg p \vee q$
0	**0**	**1**	**1**
0	**1**	**1**	**1**
1	**0**	**0**	**0**
1	**1**	**1**	**1**

We have $p \wedge \neg q \to q \vee \neg p$ is equivalent to $\neg p \vee q$

Summary: Using truth tables we can check satisfiability, validity and equivalence.

Limitations: For modest number of variables truth tables are unacceptably huge!

Later: more practical algorithms.

Connections between these notions

- A formula A is valid if and only if $\neg A$ is unsatisfiable.
- A formula A is satisfiable if and only if $\neg A$ is not valid.
- A formula A is valid if and only if A is equivalent to $\top$.
- Formulas A and B are equivalent if and only if the formula $A \leftrightarrow B$ is valid.

- Propositional satisfiability is NP-complete.
- Propositional validity is coNP-complete.

Connections between these notions

- A formula A is valid if and only if $\neg A$ is unsatisfiable.
- A formula A is satisfiable if and only if $\neg A$ is not valid.
- A formula A is valid if and only if A is equivalent to $\top$.
- Formulas A and B are equivalent if and only if the formula $A \leftrightarrow B$ is valid.

- Propositional satisfiability is NP-complete.
- Propositional validity is coNP-complete.

Connections between these notions

- A formula A is valid if and only if $\neg A$ is unsatisfiable.
- A formula A is satisfiable if and only if $\neg A$ is not valid.
- A formula A is valid if and only if A is equivalent to $\top$.
- Formulas A and B are equivalent if and only if the formula $A \leftrightarrow B$ is valid.

- Propositional satisfiability is NP-complete.
- Propositional validity is coNP-complete.

Connections between these notions

- A formula A is valid if and only if $\neg A$ is unsatisfiable.
- A formula A is satisfiable if and only if $\neg A$ is not valid.
- A formula A is valid if and only if A is equivalent to $\top$.
- Formulas A and B are equivalent if and only if the formula $A \leftrightarrow B$ is valid.

- Propositional satisfiability is NP-complete.
- Propositional validity is coNP-complete.

Connections between these notions

- A formula A is valid if and only if $\neg A$ is unsatisfiable.
- A formula A is satisfiable if and only if $\neg A$ is not valid.
- A formula A is valid if and only if A is equivalent to $\top$.
- Formulas A and B are equivalent if and only if the formula $A \leftrightarrow B$ is valid.

- Propositional satisfiability is NP-complete.
- Propositional validity is coNP-complete.

Connections between these notions

- A formula A is valid if and only if $\neg A$ is unsatisfiable.
- A formula A is satisfiable if and only if $\neg A$ is not valid.
- A formula A is valid if and only if A is equivalent to $\top$.
- Formulas A and B are equivalent if and only if the formula $A \leftrightarrow B$ is valid.

- Propositional satisfiability is NP-complete.
- Propositional validity is coNP-complete.

Some useful equivalences

$$\neg\neg A \equiv A \tag{1}$$

$$A \wedge B \equiv B \wedge A \tag{2}$$

$$A \vee B \equiv B \vee A \tag{3}$$

$$(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C) \tag{4}$$

$$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C) \tag{5}$$

$$A \to B \equiv \neg A \vee B \tag{6}$$

$$\neg(A \wedge B) \equiv \neg A \vee \neg B \tag{7}$$

$$\neg(A \vee B) \equiv \neg A \wedge \neg B \tag{8}$$

$$A \leftrightarrow B \equiv (A \to B) \wedge (B \to A) \tag{9}$$

Some useful equivalences

$$\neg\neg A \equiv A \tag{1}$$

$$A \wedge B \equiv B \wedge A \tag{2}$$

$$A \vee B \equiv B \vee A \tag{3}$$

$$(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C) \tag{4}$$

$$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C) \tag{5}$$

$$A \rightarrow B \equiv \neg A \vee B \tag{6}$$

$$\neg(A \wedge B) \equiv \neg A \vee \neg B \tag{7}$$

$$\neg(A \vee B) \equiv \neg A \wedge \neg B \tag{8}$$

$$A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A) \tag{9}$$

Some useful equivalences

$$\neg\neg A \equiv A \tag{1}$$

$$A \wedge B \equiv B \wedge A \tag{2}$$

$$A \vee B \equiv B \vee A \tag{3}$$

$$(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C) \tag{4}$$

$$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C) \tag{5}$$

$$A \rightarrow B \equiv \neg A \vee B \tag{6}$$

$$\neg(A \wedge B) \equiv \neg A \vee \neg B \tag{7}$$

$$\neg(A \vee B) \equiv \neg A \wedge \neg B \tag{8}$$

$$A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A) \tag{9}$$

Some useful equivalences

$$\neg\neg A \equiv A \tag{1}$$

$$A \wedge B \equiv B \wedge A \tag{2}$$

$$A \vee B \equiv B \vee A \tag{3}$$

$$(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C) \tag{4}$$

$$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C) \tag{5}$$

$$A \rightarrow B \equiv \neg A \vee B \tag{6}$$

$$\neg(A \wedge B) \equiv \neg A \vee \neg B \tag{7}$$

$$\neg(A \vee B) \equiv \neg A \wedge \neg B \tag{8}$$

$$A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A) \tag{9}$$

Some useful equivalences

$$\neg\neg A \equiv A \tag{1}$$

$$A \wedge B \equiv B \wedge A \tag{2}$$

$$A \vee B \equiv B \vee A \tag{3}$$

$$(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C) \tag{4}$$

$$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C) \tag{5}$$

$$A \rightarrow B \equiv \neg A \vee B \tag{6}$$

$$\neg(A \wedge B) \equiv \neg A \vee \neg B \tag{7}$$

$$\neg(A \vee B) \equiv \neg A \wedge \neg B \tag{8}$$

$$A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A) \tag{9}$$

Some useful equivalences

$$\neg\neg A \equiv A \tag{1}$$

$$A \land B \equiv B \land A \tag{2}$$

$$A \lor B \equiv B \lor A \tag{3}$$

$$(A \lor B) \land C \equiv (A \land C) \lor (B \land C) \tag{4}$$

$$(A \land B) \lor C \equiv (A \lor C) \land (B \lor C) \tag{5}$$

$$A \to B \equiv \neg A \lor B \tag{6}$$

$$\neg(A \land B) \equiv \neg A \lor \neg B \tag{7}$$

$$\neg(A \lor B) \equiv \neg A \land \neg B \tag{8}$$

$$A \leftrightarrow B \equiv (A \to B) \land (B \to A) \tag{9}$$

Some useful equivalences

$$\neg\neg A \equiv A \tag{1}$$

$$A \wedge B \equiv B \wedge A \tag{2}$$

$$A \vee B \equiv B \vee A \tag{3}$$

$$(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C) \tag{4}$$

$$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C) \tag{5}$$

$$A \rightarrow B \equiv \neg A \vee B \tag{6}$$

$$\neg(A \wedge B) \equiv \neg A \vee \neg B \tag{7}$$

$$\neg(A \vee B) \equiv \neg A \wedge \neg B \tag{8}$$

$$A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A) \tag{9}$$

Some useful equivalences

$$\neg\neg A \equiv A \tag{1}$$

$$A \wedge B \equiv B \wedge A \tag{2}$$

$$A \vee B \equiv B \vee A \tag{3}$$

$$(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C) \tag{4}$$

$$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C) \tag{5}$$

$$A \rightarrow B \equiv \neg A \vee B \tag{6}$$

$$\neg(A \wedge B) \equiv \neg A \vee \neg B \tag{7}$$

$$\neg(A \vee B) \equiv \neg A \wedge \neg B \tag{8}$$

$$A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A) \tag{9}$$

Some useful equivalences

$$\neg\neg A \equiv A \tag{1}$$

$$A \wedge B \equiv B \wedge A \tag{2}$$

$$A \vee B \equiv B \vee A \tag{3}$$

$$(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C) \tag{4}$$

$$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C) \tag{5}$$

$$A \rightarrow B \equiv \neg A \vee B \tag{6}$$

$$\neg(A \wedge B) \equiv \neg A \vee \neg B \tag{7}$$

$$\neg(A \vee B) \equiv \neg A \wedge \neg B \tag{8}$$

$$A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A) \tag{9}$$

A few more equivalences

$$A \lor \bot \equiv \tag{10}$$

$$A \lor \top \equiv \tag{11}$$

$$A \land \bot \equiv \tag{12}$$

$$A \land \top \equiv \tag{13}$$

$$A \to \bot \equiv \tag{14}$$

$$\bot \to A \equiv \tag{15}$$

$$A \leftrightarrow \top \equiv \tag{16}$$

$$A \leftrightarrow \bot \equiv \tag{17}$$

A few more equivalences

$$A \vee \bot \equiv A \tag{10}$$

$$A \vee \top \equiv \tag{11}$$

$$A \wedge \bot \equiv \tag{12}$$

$$A \wedge \top \equiv \tag{13}$$

$$A \rightarrow \bot \equiv \tag{14}$$

$$\bot \rightarrow A \equiv \tag{15}$$

$$A \leftrightarrow \top \equiv \tag{16}$$

$$A \leftrightarrow \bot \equiv \tag{17}$$

A few more equivalences

$$A \vee \bot \equiv A \tag{10}$$

$$A \vee \top \equiv \top \tag{11}$$

$$A \wedge \bot \equiv \tag{12}$$

$$A \wedge \top \equiv \tag{13}$$

$$A \to \bot \equiv \tag{14}$$

$$\bot \to A \equiv \tag{15}$$

$$A \leftrightarrow \top \equiv \tag{16}$$

$$A \leftrightarrow \bot \equiv \tag{17}$$

A few more equivalences

$$A \vee \bot \equiv A \tag{10}$$

$$A \vee \top \equiv \top \tag{11}$$

$$A \wedge \bot \equiv \bot \tag{12}$$

$$A \wedge \top \equiv \tag{13}$$

$$A \to \bot \equiv \tag{14}$$

$$\bot \to A \equiv \tag{15}$$

$$A \leftrightarrow \top \equiv \tag{16}$$

$$A \leftrightarrow \bot \equiv \tag{17}$$

A few more equivalences

$$A \vee \bot \equiv A \tag{10}$$

$$A \vee \top \equiv \top \tag{11}$$

$$A \wedge \bot \equiv \bot \tag{12}$$

$$A \wedge \top \equiv A \tag{13}$$

$$A \rightarrow \bot \equiv \tag{14}$$

$$\bot \rightarrow A \equiv \tag{15}$$

$$A \leftrightarrow \top \equiv \tag{16}$$

$$A \leftrightarrow \bot \equiv \tag{17}$$

A few more equivalences

$$A \vee \bot \equiv A \tag{10}$$

$$A \vee \top \equiv \top \tag{11}$$

$$A \wedge \bot \equiv \bot \tag{12}$$

$$A \wedge \top \equiv A \tag{13}$$

$$A \rightarrow \bot \equiv \neg A \tag{14}$$

$$\bot \rightarrow A \equiv \tag{15}$$

$$A \leftrightarrow \top \equiv \tag{16}$$

$$A \leftrightarrow \bot \equiv \tag{17}$$

A few more equivalences

$$A \vee \bot \equiv A \tag{10}$$

$$A \vee \top \equiv \top \tag{11}$$

$$A \wedge \bot \equiv \bot \tag{12}$$

$$A \wedge \top \equiv A \tag{13}$$

$$A \to \bot \equiv \neg A \tag{14}$$

$$\bot \to A \equiv \top \tag{15}$$

$$A \leftrightarrow \top \equiv \tag{16}$$

$$A \leftrightarrow \bot \equiv \tag{17}$$

A few more equivalences

$$A \lor \bot \equiv A \tag{10}$$

$$A \lor \top \equiv \top \tag{11}$$

$$A \land \bot \equiv \bot \tag{12}$$

$$A \land \top \equiv A \tag{13}$$

$$A \rightarrow \bot \equiv \neg A \tag{14}$$

$$\bot \rightarrow A \equiv \top \tag{15}$$

$$A \leftrightarrow \top \equiv A \tag{16}$$

$$A \leftrightarrow \bot \equiv \tag{17}$$

A few more equivalences

$$A \vee \bot \equiv A \tag{10}$$

$$A \vee \top \equiv \top \tag{11}$$

$$A \wedge \bot \equiv \bot \tag{12}$$

$$A \wedge \top \equiv A \tag{13}$$

$$A \rightarrow \bot \equiv \neg A \tag{14}$$

$$\bot \rightarrow A \equiv \top \tag{15}$$

$$A \leftrightarrow \top \equiv A \tag{16}$$

$$A \leftrightarrow \bot \equiv \neg A \tag{17}$$

Substitution Lemma

- Propositional substitution (or substitution) is a mapping from propositional variables into propositional formulas.
- For a propositional formula A, $A\Theta$ denote a formula obtained from A by replacing variables by formulas according to Θ.
- Example:

$$\begin{array}{rcl} \Theta & = & \{p \mapsto s \rightarrow m \vee u, q \mapsto \top, r \mapsto m \leftrightarrow u\}. \\ A & = & p \wedge q \leftrightarrow r \\ A\Theta & = & (s \rightarrow m \vee u) \wedge \top \leftrightarrow (m \leftrightarrow u) \end{array}$$

Lemma. For any substitution Θ and a valid formula A, $A\Theta$ is valid.

Substitution Lemma

- Propositional substitution (or substitution) is a mapping from propositional variables into propositional formulas.
- For a propositional formula A, $A\Theta$ denote a formula obtained from A by replacing variables by formulas according to Θ.
- Example:

$$\begin{aligned}\Theta &= \{p \mapsto s \to m \vee u, q \mapsto \top, r \mapsto m \leftrightarrow u\}.\\ A &= p \wedge q \leftrightarrow r\\ A\Theta &= (s \to m \vee u) \wedge \top \leftrightarrow (m \leftrightarrow u)\end{aligned}$$

Lemma. For any substitution Θ and a valid formula A, $A\Theta$ is valid.

Substitution Lemma

- Propositional substitution (or substitution) is a mapping from propositional variables into propositional formulas.
- For a propositional formula A, $A\Theta$ denote a formula obtained from A by replacing variables by formulas according to Θ.
- Example:

$$\begin{array}{rcl} \Theta & = & \{p \mapsto s \to m \vee u, q \mapsto \top, r \mapsto m \leftrightarrow u\}. \\ A & = & p \wedge q \leftrightarrow r \\ A\Theta & = & (s \to m \vee u) \wedge \top \leftrightarrow (m \leftrightarrow u) \end{array}$$

Lemma. For any substitution Θ and a valid formula A, $A\Theta$ is valid.

Substitution Lemma

- Propositional substitution (or substitution) is a mapping from propositional variables into propositional formulas.
- For a propositional formula A, $A\Theta$ denote a formula obtained from A by replacing variables by formulas according to Θ.
- Example:

$$\begin{aligned}
\Theta &= \{p \mapsto s \to m \vee u, q \mapsto \top, r \mapsto m \leftrightarrow u\}. \\
A &= p \wedge q \leftrightarrow r \\
A\Theta &= (s \to m \vee u) \wedge \top \leftrightarrow (m \leftrightarrow u)
\end{aligned}$$

Lemma. For any substitution Θ and a valid formula A, $A\Theta$ is valid.

Substitution Lemma

- Propositional substitution (or substitution) is a mapping from propositional variables into propositional formulas.
- For a propositional formula A, $A\Theta$ denote a formula obtained from A by replacing variables by formulas according to Θ.
- Example:

$$
\begin{aligned}
\Theta &= \{p \mapsto s \rightarrow m \vee u, q \mapsto \top, r \mapsto m \leftrightarrow u\}. \\
A &= p \wedge q \leftrightarrow r \\
A\Theta &= (s \rightarrow m \vee u) \wedge \top \leftrightarrow (m \leftrightarrow u)
\end{aligned}
$$

Lemma. For any substitution Θ and a valid formula A, $A\Theta$ is valid.

Equivalent replacement

Lemma

Let I be an interpretation and $I \models A_1 \leftrightarrow A_2$. Then $I \models B[A_1] \leftrightarrow B[A_2]$.

Theorem

(Equivalent Replacement) Let $A_1 \equiv A_2$. Then $B[A_1] \equiv B[A_2]$.

Example: $(D \wedge C \to \neg D) \to C$	Apply:	$A \to B$	$\equiv$	$\neg A \vee B$
$(D \wedge C \to \neg D) \to C$				
$\neg(D \wedge C) \vee \neg D \to C$	Apply:	$\neg(A \wedge B)$	$\equiv$	$\neg A \vee \neg B$
$\neg D \vee \neg C \vee \neg D \to C$	Apply:	$A \to B$	$\equiv$	$\neg A \vee B$
$\neg(\neg D \vee \neg C \vee \neg D) \vee C$	Apply...	Apply...		
$(D \wedge C) \vee C$				

Equivalent replacement

Lemma

Let I be an interpretation and $I \models A_1 \leftrightarrow A_2$. Then $I \models B[A_1] \leftrightarrow B[A_2]$.

Theorem

(Equivalent Replacement) Let $A_1 \equiv A_2$. Then $B[A_1] \equiv B[A_2]$.

Example: $(D \wedge C \rightarrow \neg D) \rightarrow C$ Apply: $A \rightarrow B \equiv \neg A \vee B$

$(D \wedge C \rightarrow \neg D) \rightarrow C$

$\neg(D \wedge C) \vee \neg D \rightarrow C$ Apply: $\neg(A \wedge B) \equiv \neg A \vee \neg B$

$\neg D \vee \neg C \vee \neg D \rightarrow C$ Apply: $A \rightarrow B \equiv \neg A \vee B$

$\neg(\neg D \vee \neg C \vee \neg D) \vee C$ Apply... Apply...

$(D \wedge C) \vee C$

Equivalent replacement

Lemma

Let I be an interpretation and $I \models A_1 \leftrightarrow A_2$. Then $I \models B[A_1] \leftrightarrow B[A_2]$.

Theorem

(Equivalent Replacement) Let $A_1 \equiv A_2$. Then $B[A_1] \equiv B[A_2]$.

Example: $(D \wedge C \rightarrow \neg D) \rightarrow C$ Apply: $A \rightarrow B \equiv \neg A \vee B$

$(D \wedge C \rightarrow \neg D) \rightarrow C$

$\neg(D \wedge C) \vee \neg D \rightarrow C$ Apply: $\neg(A \wedge B) \equiv \neg A \vee \neg B$

$\neg D \vee \neg C \vee \neg D \rightarrow C$ Apply: $A \rightarrow B \equiv \neg A \vee B$

$\neg(\neg D \vee \neg C \vee \neg D) \vee C$ Apply... Apply...

$(D \wedge C) \vee C$

Equivalent replacement

Lemma

Let I be an interpretation and $I \models A_1 \leftrightarrow A_2$. Then $I \models B[A_1] \leftrightarrow B[A_2]$.

Theorem

(Equivalent Replacement) Let $A_1 \equiv A_2$. Then $B[A_1] \equiv B[A_2]$.

Example: $(D \wedge C \to \neg D) \to C$ Apply: $A \to B \equiv \neg A \vee B$

$(D \wedge C \to \neg D) \to C$

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

$\neg D \vee \neg C \vee \neg D \to C$ Apply: $A \to B \equiv \neg A \vee B$

$\neg(\neg D \vee \neg C \vee \neg D) \vee C$ Apply... Apply...

$(D \wedge C) \vee C$

Equivalent replacement

Lemma

Let I be an interpretation and $I \models A_1 \leftrightarrow A_2$. Then $I \models B[A_1] \leftrightarrow B[A_2]$.

Theorem

(Equivalent Replacement) Let $A_1 \equiv A_2$. Then $B[A_1] \equiv B[A_2]$.

Example:	$(D \wedge C \rightarrow \neg D) \rightarrow C$	Apply:	$A \rightarrow B$	$\equiv$	$\neg A \vee B$
	$(D \wedge C \rightarrow \neg D) \rightarrow C$				
	$\neg(D \wedge C) \vee \neg D \rightarrow C$	Apply:	$\neg(A \wedge B)$	$\equiv$	$\neg A \vee \neg B$
	$\neg D \vee \neg C \vee \neg D \rightarrow C$	Apply:	$A \rightarrow B$	$\equiv$	$\neg A \vee B$
	$\neg(\neg D \vee \neg C \vee \neg D) \vee C$	Apply...	Apply...		
	$(D \wedge C) \vee C$				

Equivalent replacement

Lemma

Let I be an interpretation and $I \models A_1 \leftrightarrow A_2$. Then $I \models B[A_1] \leftrightarrow B[A_2]$.

Theorem

(Equivalent Replacement) Let $A_1 \equiv A_2$. Then $B[A_1] \equiv B[A_2]$.

Example:	$(D \wedge C \rightarrow \neg D) \rightarrow C$	Apply:	$A \rightarrow B$	$\equiv$	$\neg A \vee B$
	$(D \wedge C \rightarrow \neg D) \rightarrow C$				
	$\neg(D \wedge C) \vee \neg D \rightarrow C$	Apply:	$\neg(A \wedge B)$	$\equiv$	$\neg A \vee \neg B$
	$\neg D \vee \neg C \vee \neg D \rightarrow C$	Apply:	$A \rightarrow B$	$\equiv$	$\neg A \vee B$
	$\neg(\neg D \vee \neg C \vee \neg D) \vee C$	Apply...	Apply...		
	$(D \wedge C) \vee C$				

Equivalent replacement

Lemma

Let I be an interpretation and $I \models A_1 \leftrightarrow A_2$. Then $I \models B[A_1] \leftrightarrow B[A_2]$.

Theorem

(Equivalent Replacement) Let $A_1 \equiv A_2$. Then $B[A_1] \equiv B[A_2]$.

Example: $(D \wedge C \rightarrow \neg D) \rightarrow C$	Apply:	$A \rightarrow B \equiv \neg A \vee B$
$(D \wedge C \rightarrow \neg D) \rightarrow C$		
$\neg(D \wedge C) \vee \neg D \rightarrow C$	Apply:	$\neg(A \wedge B) \equiv \neg A \vee \neg B$
$\neg D \vee \neg C \vee \neg D \rightarrow C$	Apply:	$A \rightarrow B \equiv \neg A \vee B$
$\neg(\neg D \vee \neg C \vee \neg D) \vee C$	Apply...	Apply...
$(D \wedge C) \vee C$		

Equivalent replacement

Lemma

Let I be an interpretation and $I \models A_1 \leftrightarrow A_2$. Then $I \models B[A_1] \leftrightarrow B[A_2]$.

Theorem

(Equivalent Replacement) Let $A_1 \equiv A_2$. Then $B[A_1] \equiv B[A_2]$.

Example: $(D \land C \to \neg D) \to C$ Apply: $A \to B \equiv \neg A \lor B$

$(D \land C \to \neg D) \to C$

$\neg(D \land C) \lor \neg D \to C$ Apply: $\neg(A \land B) \equiv \neg A \lor \neg B$

$\neg D \lor \neg C \lor \neg D \to C$ Apply: $A \to B \equiv \neg A \lor B$

$\neg(\neg D \lor \neg C \lor \neg D) \lor C$ Apply... Apply...

$(D \land C) \lor C$

Equivalent replacement

Lemma

Let I be an interpretation and $I \models A_1 \leftrightarrow A_2$. Then $I \models B[A_1] \leftrightarrow B[A_2]$.

Theorem

(Equivalent Replacement) Let $A_1 \equiv A_2$. Then $B[A_1] \equiv B[A_2]$.

Example: $(D \land C \to \neg D) \to C$	Apply:	$A \to B$	$\equiv$	$\neg A \lor B$
$(D \land C \to \neg D) \to C$				
$\neg(D \land C) \lor \neg D \to C$	Apply:	$\neg(A \land B)$	$\equiv$	$\neg A \lor \neg B$
$\neg D \lor \neg C \lor \neg D \to C$	Apply:	$A \to B$	$\equiv$	$\neg A \lor B$
$\neg(\neg D \lor \neg C \lor \neg D) \lor C$	Apply...	Apply...		
$(D \land C) \lor C$				

Boolean functions

Boolean function of an arity n maps n sequences of truth values (boolean values) to $\{\mathbf{0}, \mathbf{1}\}$:

$$f : \{\mathbf{0}, \mathbf{1}\}^n \to \{\mathbf{0}, \mathbf{1}\}$$

We can define such a function by a value table:

p	q	$p + q$ mod 2
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$
$\mathbf{0}$	$\mathbf{1}$	$\mathbf{1}$
$\mathbf{1}$	$\mathbf{0}$	$\mathbf{1}$
$\mathbf{1}$	$\mathbf{1}$	$\mathbf{0}$

Boolean functions

Boolean function of an arity n maps n sequences of truth values (boolean values) to $\{\mathbf{0}, \mathbf{1}\}$:

$$f : \{\mathbf{0}, \mathbf{1}\}^n \to \{\mathbf{0}, \mathbf{1}\}$$

We can define such a function by a value table:

p	q	$p + q \ mod \ 2$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$
$\mathbf{0}$	$\mathbf{1}$	$\mathbf{1}$
$\mathbf{1}$	$\mathbf{0}$	$\mathbf{1}$
$\mathbf{1}$	$\mathbf{1}$	$\mathbf{0}$

Boolean functions and formulas

Any propositional formula represents a boolean function.

Assume A on variables $p_1, \ldots, p_n$ then define:

$$f_A(I(p_1), \ldots, I(p_n)) = I(A)$$

Example: $A = p \wedge q$

p	q	$f_\wedge(p, q)$
0	**0**	**0**
0	**1**	**0**
1	**0**	**0**
1	**1**	**1**

Later: we will see that the converse is also true:
Every boolean function is represented by a propositional formula.
We will often use this correspondence.

Boolean functions and formulas

Any propositional formula represents a boolean function.

Assume A on variables $p_1, \ldots, p_n$ then define:

$$f_A(I(p_1), \ldots, I(p_n)) = I(A)$$

Example: $A = p \wedge q$

p	q	$f_\wedge(p, q)$
0	**0**	**0**
0	**1**	**0**
1	**0**	**0**
1	**1**	**1**

Later: we will see that the converse is also true:

Every boolean function is represented by a propositional formula.

We will often use this correspondence.

Summary

We have studied notions of:

- satisfiability, validity, equivalence
- Using a semantic method of truth tables we can solve the above problems for a small number of variables
  - for a large number of variables truth tables are impractical

Next: more practical methods for satisfiability.

Section 4 Reasoning Methods

Reasoning methods

Aim: Prove validity/satisfiability of propositional formulas.

Reasoning Methods:

- Splitting
- Resolution
- DPLL
- Tableaux

Efficiency is the major problem.

Refutational reasoning

In reasoning methods we study, the validity problem is reformulated in terms of unsatisfiability. Proof by contradiction.

A is valid iff $\neg A$ is unsatisfiable.

In other words:

$$\models A \text{ iff } \neg A \models \bot$$

Example. The are an infinite number of prime numbers.

Other common problems:

$$\models \mathrm{Axioms} \rightarrow \mathrm{Theorem} \text{ iff } \mathrm{Axioms} \wedge \neg\mathrm{Theorem} \models \bot$$

$$\models A \leftrightarrow B \text{ iff } A \leftrightarrow \neg B \models \bot$$

Refutational reasoning

In reasoning methods we study, the validity problem is reformulated in terms of unsatisfiability. Proof by contradiction.

A is valid iff $\neg A$ is unsatisfiable.

In other words:

$$\models A \text{ iff } \neg A \models \bot$$

Example. The are an infinite number of prime numbers.

Other common problems:

$$\models \mathrm{Axioms} \rightarrow \mathrm{Theorem} \text{ iff } \mathrm{Axioms} \wedge \neg \mathrm{Theorem} \models \bot$$

$$\models A \leftrightarrow B \text{ iff } A \leftrightarrow \neg B \models \bot$$

Refutational reasoning

In reasoning methods we study, the validity problem is reformulated in terms of unsatisfiability. Proof by contradiction.

$$A \text{ is valid iff } \neg A \text{ is unsatisfiable.}$$

In other words:

$$\models A \text{ iff } \neg A \models \bot$$

Example. The are an infinite number of prime numbers.

Other common problems:

$$\models \mathrm{Axioms} \rightarrow \mathrm{Theorem} \text{ iff } \mathrm{Axioms} \wedge \neg\mathrm{Theorem} \models \bot$$

$$\models A \leftrightarrow B \text{ iff } A \leftrightarrow \neg B \models \bot$$

Refutational reasoning

In reasoning methods we study, the validity problem is reformulated in terms of unsatisfiability. Proof by contradiction.

$$A \text{ is valid iff } \neg A \text{ is unsatisfiable.}$$

In other words:

$$\models A \text{ iff } \neg A \models \bot$$

Example. The are an infinite number of prime numbers.

Other common problems:

$$\models \mathrm{Axioms} \rightarrow \mathrm{Theorem} \text{ iff } \mathrm{Axioms} \wedge \neg \mathrm{Theorem} \models \bot$$

$$\models A \leftrightarrow B \text{ iff } A \leftrightarrow \neg B \models \bot$$

Refutational reasoning

In reasoning methods we study, the validity problem is reformulated in terms of unsatisfiability. Proof by contradiction.

$$A \text{ is valid iff } \neg A \text{ is unsatisfiable.}$$

In other words:

$$\models A \text{ iff } \neg A \models \bot$$

Example. The are an infinite number of prime numbers.

Other common problems:

$$\models \mathrm{Axioms} \rightarrow \mathrm{Theorem} \text{ iff } \mathrm{Axioms} \wedge \neg\mathrm{Theorem} \models \bot$$

$$\models A \leftrightarrow B \text{ iff } A \leftrightarrow \neg B \models \bot$$

Soundness

A refutational reasoning method (or just reasoning method RM) is an algorithm (not necessarily terminating) which given as an input a set of formulas S outputs either "satisfiable", "unsatisfiable" or "don't know".

Consider a set of formulas Φ (usually called a fragment).

A reasoning method RM is sound for Φ if for any set $S \subseteq \Phi$:

- if $\mathrm{RM}(S)$ is "satisfiable" then there is an interpretation satisfying all formulas in S
- if $\mathrm{RM}(S)$ is "unsatisfiable" then there is no interpretation satisfying all formulas in S.

Remark: A trivial RM which on all inputs returns "don't know" is a sound reasoning method.

Soundness

A refutational reasoning method (or just reasoning method RM) is an algorithm (not necessarily terminating) which given as an input a set of formulas S outputs either "satisfiable", "unsatisfiable" or "don't know".

Consider a set of formulas Φ (usually called a fragment).

A reasoning method RM is sound for Φ if for any set $S \subseteq \Phi$:

- if $\mathrm{RM}(S)$ is "satisfiable" then there is an interpretation satisfying all formulas in S
- if $\mathrm{RM}(S)$ is "unsatisfiable" then there is no interpretation satisfying all formulas in S.

Remark: A trivial RM which on all inputs returns "don't know" is a sound reasoning method.

Soundness

A refutational reasoning method (or just reasoning method RM) is an algorithm (not necessarily terminating) which given as an input a set of formulas S outputs either "satisfiable", "unsatisfiable" or "don't know".

Consider a set of formulas Φ (usually called a fragment).

A reasoning method RM is sound for Φ if for any set $S \subseteq \Phi$:

- if $\mathrm{RM}(S)$ is "satisfiable" then there is an interpretation satisfying all formulas in S
- if $\mathrm{RM}(S)$ is "unsatisfiable" then there is no interpretation satisfying all formulas in S.

Remark: A trivial RM which on all inputs returns "don't know" is a sound reasoning method.

Completeness, Decision Procedures

Consider a set of formulas Φ.

A reasoning method RM is (refutationally) complete for Φ if for any set $S \subseteq \Phi$:

- if S is unsatisfiable then $\mathrm{RM}(S)$ is terminating and returns "unsatisfiable".

A reasoning method RM is terminating for Φ if $\mathrm{RM}(S)$ is terminating for any finite set of formulas $S \subseteq \Phi$.

A reasoning method RM is a decision procedure for Φ if RM is sound, refutationally complete and terminating for Φ.

Lemma

The truth table method is a decision procedure for propositional logic.

Completeness, Decision Procedures

Consider a set of formulas Φ.

A reasoning method RM is (refutationally) complete for Φ if for any set $S \subseteq \Phi$:

- if S is unsatisfiable then $\mathrm{RM}(S)$ is terminating and returns "unsatisfiable".

A reasoning method RM is terminating for Φ if $\mathrm{RM}(S)$ is terminating for any finite set of formulas $S \subseteq \Phi$.

A reasoning method RM is a decision procedure for Φ if RM is sound, refutationally complete and terminating for Φ.

Lemma

The truth table method is a decision procedure for propositional logic.

Completeness, Decision Procedures

Consider a set of formulas Φ.

A reasoning method RM is (refutationally) complete for Φ if for any set $S \subseteq \Phi$:

- if S is unsatisfiable then $\mathrm{RM}(S)$ is terminating and returns "unsatisfiable".

A reasoning method RM is terminating for Φ if $\mathrm{RM}(S)$ is terminating for any finite set of formulas $S \subseteq \Phi$.

A reasoning method RM is a decision procedure for Φ if RM is sound, refutationally complete and terminating for Φ.

Lemma

The truth table method is a decision procedure for propositional logic.

Splitting: the theoretical basis

$A_p^\perp$ and $A_p^\top$: the formulas obtained by replacing in A all occurrences of p by $\perp$ and $\top$, respectively.

Lemma. Let p be an atom, A be a formula, and I be a partial interpretation.

1. If $I \models p$, then A is equivalent to $A_p^\top$ in I.
2. If $I \models \neg p$, then A is equivalent to $A_p^\perp$ in I.

- Pick a variable p and perform case analysis on this variable:
  - In the case p is true, replace p by $\top$;
  - in the case p is false, replace p by $\perp$.
- When a formula contains occurrences of $\top$ or $\perp$, simplify it using rewrite rules.

Splitting: the theoretical basis

$A_p^{\perp}$ and $A_p^{\top}$: the formulas obtained by replacing in A all occurrences of p by $\perp$ and $\top$, respectively.

Lemma. Let p be an atom, A be a formula, and I be a partial interpretation.

1. If $I \models p$, then A is equivalent to $A_p^{\top}$ in I.
2. If $I \models \neg p$, then A is equivalent to $A_p^{\perp}$ in I.

- Pick a variable p and perform case analysis on this variable:
  - In the case p is true, replace p by $\top$;
  - in the case p is false, replace p by $\perp$.
- When a formula contains occurrences of $\top$ or $\perp$, simplify it using rewrite rules.

Splitting: the theoretical basis

$A_p^\perp$ and $A_p^\top$: the formulas obtained by replacing in A all occurrences of p by $\perp$ and $\top$, respectively.

Lemma. Let p be an atom, A be a formula, and I be a partial interpretation.

1. If $I \models p$, then A is equivalent to $A_p^\top$ in I.
2. If $I \models \neg p$, then A is equivalent to $A_p^\perp$ in I.

- Pick a variable p and perform case analysis on this variable:
  - In the case p is true, replace p by $\top$;
  - in the case p is false, replace p by $\perp$.
- When a formula contains occurrences of $\top$ or $\perp$, simplify it using rewrite rules.

Splitting: the theoretical basis

$A_p^\perp$ and $A_p^\top$: the formulas obtained by replacing in A all occurrences of p by $\perp$ and $\top$, respectively.

Lemma. Let p be an atom, A be a formula, and I be a partial interpretation.

1. If $I \models p$, then A is equivalent to $A_p^\top$ in I.
2. If $I \models \neg p$, then A is equivalent to $A_p^\perp$ in I.

- Pick a variable p and perform case analysis on this variable:
  - In the case p is true, replace p by $\top$;
  - in the case p is false, replace p by $\perp$.
- When a formula contains occurrences of $\top$ or $\perp$, simplify it using rewrite rules.

Simplification rules for $\top$ and $\bot$

Note: we need new simplification rules since formulas we simplify may contain propositional variables.

Simplification rules for $\top$:

$$\neg\top \Rightarrow \bot$$
$$\top \wedge A_1 \wedge \ldots \wedge A_n \Rightarrow A_1 \wedge \ldots \wedge A_n$$
$$\top \vee A_1 \vee \ldots \vee A_n \Rightarrow \top$$
$$A \rightarrow \top \Rightarrow \top \qquad \top \rightarrow A \Rightarrow A$$
$$A \leftrightarrow \top \Rightarrow A \qquad \top \leftrightarrow A \Rightarrow A$$

Simplification rules for $\bot$:

$$\neg\bot \Rightarrow \top$$
$$\bot \wedge A_1 \wedge \ldots \wedge A_n \Rightarrow \bot$$
$$\bot \vee A_1 \vee \ldots \vee A_n \Rightarrow A_1 \vee \ldots \vee A_n$$
$$A \rightarrow \bot \Rightarrow \neg A \qquad \bot \rightarrow A \Rightarrow \top$$
$$A \leftrightarrow \bot \Rightarrow \neg A \qquad \bot \leftrightarrow A \Rightarrow \neg A$$

Note that they cover all cases when $\bot$ or $\top$ occurs in the formula apart from the trivial ones.

If we apply these rules until they are no more applicable we obtain either a formula without $\bot$ or $\top$, or $\bot$, or $\top$.

Simplification rules for $\top$ and $\bot$

Note: we need new simplification rules since formulas we simplify may contain propositional variables.

Simplification rules for $\top$:

$$\neg\top \Rightarrow \bot$$
$$\top \wedge A_1 \wedge \ldots \wedge A_n \Rightarrow A_1 \wedge \ldots \wedge A_n$$
$$\top \vee A_1 \vee \ldots \vee A_n \Rightarrow \top$$
$$A \rightarrow \top \Rightarrow \top \qquad \top \rightarrow A \Rightarrow A$$
$$A \leftrightarrow \top \Rightarrow A \qquad \top \leftrightarrow A \Rightarrow A$$

Simplification rules for $\bot$:

$$\neg\bot \Rightarrow \top$$
$$\bot \wedge A_1 \wedge \ldots \wedge A_n \Rightarrow \bot$$
$$\bot \vee A_1 \vee \ldots \vee A_n \Rightarrow A_1 \vee \ldots \vee A_n$$
$$A \rightarrow \bot \Rightarrow \neg A \qquad \bot \rightarrow A \Rightarrow \top$$
$$A \leftrightarrow \bot \Rightarrow \neg A \qquad \bot \leftrightarrow A \Rightarrow \neg A$$

Note that they cover all cases when $\bot$ or $\top$ occurs in the formula apart from the trivial ones.

If we apply these rules until they are no more applicable we obtain either a formula without $\bot$ or $\top$, or $\bot$, or $\top$.

Simplification rules for $\top$ and $\bot$

Note: we need new simplification rules since formulas we simplify may contain propositional variables.

Simplification rules for $\top$:	Simplification rules for $\bot$:
$\neg\top \;\Rightarrow\; \bot$	$\neg\bot \;\Rightarrow\; \top$
$\top \wedge A_1 \wedge \ldots \wedge A_n \;\Rightarrow\; A_1 \wedge \ldots \wedge A_n$	$\bot \wedge A_1 \wedge \ldots \wedge A_n \;\Rightarrow\; \bot$
$\top \vee A_1 \vee \ldots \vee A_n \;\Rightarrow\; \top$	$\bot \vee A_1 \vee \ldots \vee A_n \;\Rightarrow\; A_1 \vee \ldots \vee A_n$
$A \rightarrow \top \;\Rightarrow\; \top \qquad \top \rightarrow A \;\Rightarrow\; A$	$A \rightarrow \bot \;\Rightarrow\; \neg A \qquad \bot \rightarrow A \;\Rightarrow\; \top$
$A \leftrightarrow \top \;\Rightarrow\; A \qquad \top \leftrightarrow A \;\Rightarrow\; A$	$A \leftrightarrow \bot \;\Rightarrow\; \neg A \qquad \bot \leftrightarrow A \;\Rightarrow\; \neg A$

Note that they cover all cases when $\bot$ or $\top$ occurs in the formula apart from the trivial ones.

If we apply these rules until they are no more applicable we obtain either a formula without $\bot$ or $\top$, or $\bot$, or $\top$.

Splitting method

procedure *split*(G)
parameters: function *select*
input: formula G
output: "satisfiable" or "unsatisfiable"
begin
  G := *simplify*(G)
  if $G = \top$ **then** **return** "satisfiable"
  if $G = \bot$ **then** **return** "unsatisfiable"
  (p, b) := *select*(G)
  case b **of**
  $\top \Rightarrow$
    if *split*($G_p^\top$) = "satisfiable"
      then **return** "satisfiable"
      else **return** *split*($G_p^\bot$)
  $\bot \Rightarrow$
    if *split*($G_p^\bot$) = "satisfiable"
      then **return** "satisfiable"
      else **return** *split*($G_p^\top$)
end

Theorem

Splitting method is a decision procedure for propositional logic.

Splitting method

procedure *split*(G)
parameters: function *select*
input: formula G
output: "satisfiable" or "unsatisfiable"
begin
  $G := \mathit{simplify}(G)$
  if $G = \top$ **then** **return** "satisfiable"
  if $G = \bot$ **then** **return** "unsatisfiable"
  $(p, b) := \mathit{select}(G)$
  case b **of**
  $\top \Rightarrow$
    if $\mathit{split}(G_p^\top) =$ "satisfiable"
      then **return** "satisfiable"
      else **return** $\mathit{split}(G_p^\bot)$
  $\bot \Rightarrow$
    if $\mathit{split}(G_p^\bot) =$ "satisfiable"
      then **return** "satisfiable"
      else **return** $\mathit{split}(G_p^\top)$
end

Theorem

Splitting method is a decision procedure for propositional logic.

Splitting method

procedure *split*(G)
parameters: function *select*
input: formula G
output: "satisfiable" or "unsatisfiable"
begin
  G := *simplify*(G)
  if $G = \top$ **then return** "satisfiable"
  if $G = \bot$ **then return** "unsatisfiable"
  (p, b) := *select*(G)
  case b **of**
  $\top \Rightarrow$
    if *split*$(G_p^\top)$ = "satisfiable"
      then return "satisfiable"
      else return *split*$(G_p^\bot)$
  $\bot \Rightarrow$
    if *split*$(G_p^\bot)$ = "satisfiable"
      then return "satisfiable"
      else return *split*$(G_p^\top)$
end

Theorem

Splitting method is a decision procedure for propositional logic.

Splitting method, example

$$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$$

The formula is unsatisfiable.

Splitting method, example

$$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$$

$q \mapsto \bot$ $\qquad$ $q \mapsto \top$

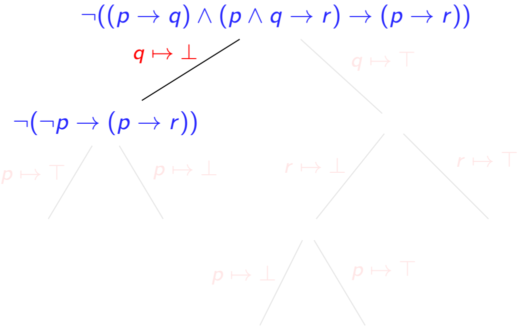
$$\neg((p \to \bot) \land (p \land \bot \to r) \to (p \to r))$$

$p \mapsto \top$ $\qquad$ $p \mapsto \bot$ $\qquad$ $r \mapsto \bot$ $\qquad$ $r \mapsto \top$

$p \mapsto \bot$ $\qquad$ $p \mapsto \top$

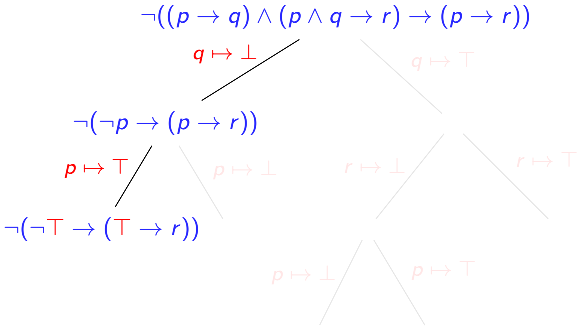
The formula is unsatisfiable.

Splitting method, example



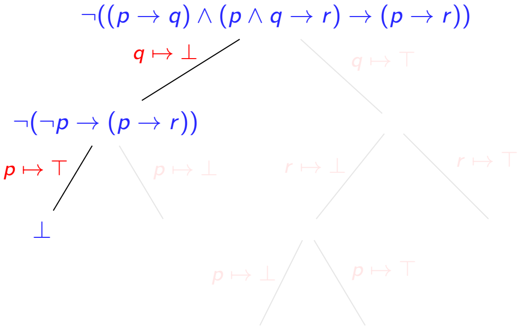
The formula is unsatisfiable.

Splitting method, example



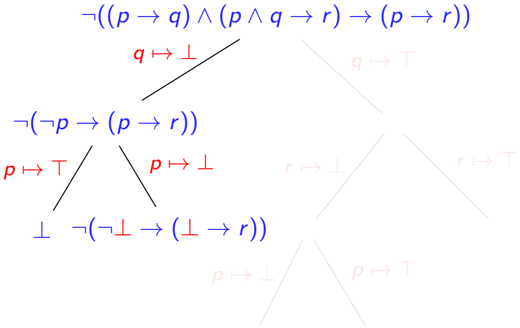
The formula is unsatisfiable.

Splitting method, example



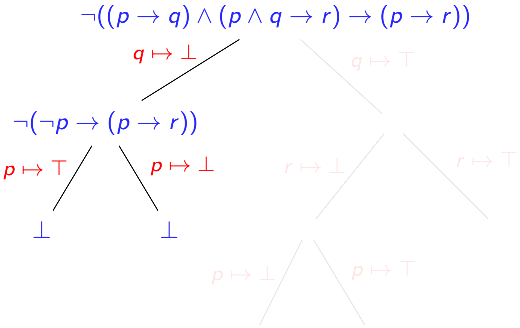
The formula is unsatisfiable.

Splitting method, example



The formula is unsatisfiable.

Splitting method, example



The formula is unsatisfiable.

Splitting method, example

$$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$$

$q \mapsto \bot$: $\neg(\neg p \to (p \to r))$

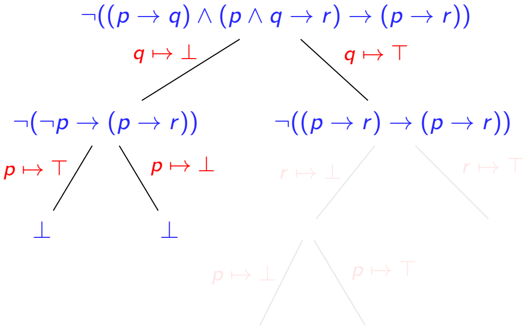
- $p \mapsto \top$: $\bot$
- $p \mapsto \bot$: $\bot$

$q \mapsto \top$: $\neg((p \to \top) \land (p \land \top \to r) \to (p \to r))$

- $r \mapsto \bot$
  - $p \mapsto \bot$
  - $p \mapsto \top$
- $r \mapsto \top$

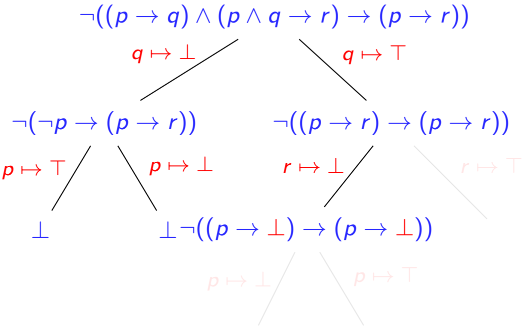
The formula is unsatisfiable.

Splitting method, example



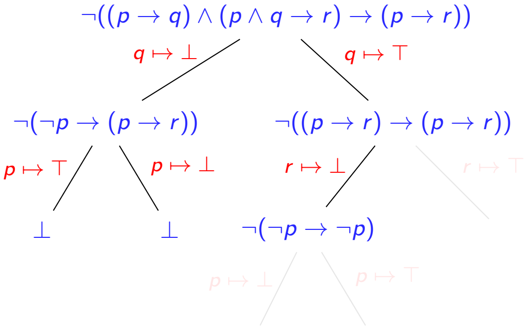
The formula is unsatisfiable.

Splitting method, example



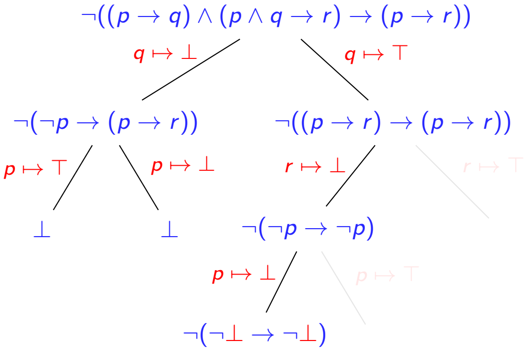
The formula is unsatisfiable.

Splitting method, example



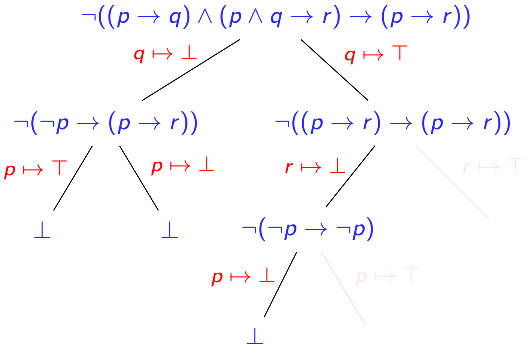
The formula is unsatisfiable.

Splitting method, example



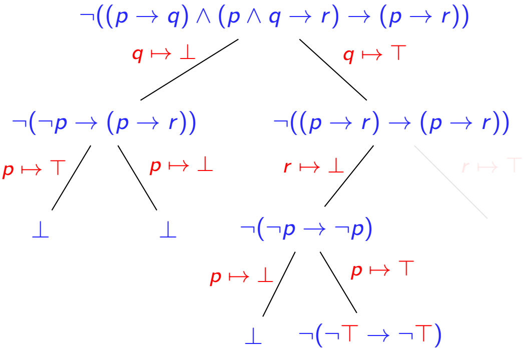
The formula is unsatisfiable.

Splitting method, example



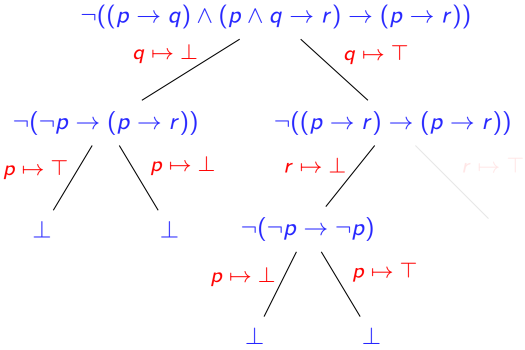
The formula is unsatisfiable.

Splitting method, example



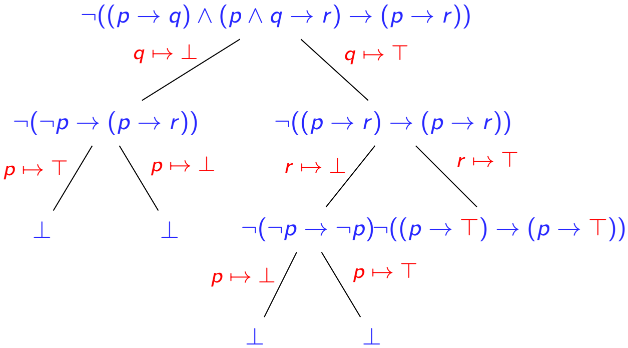
The formula is unsatisfiable.

Splitting method, example



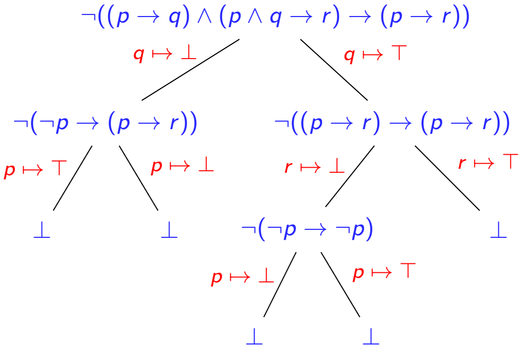
The formula is unsatisfiable.

Splitting method, example



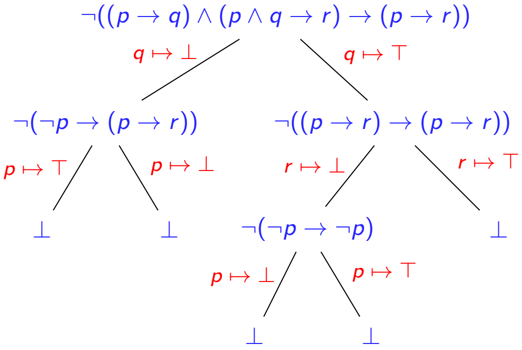
The formula is unsatisfiable.

Splitting method, example



The formula is unsatisfiable.

Splitting method, example



The formula is unsatisfiable.

Splitting method, example 2

$$\neg((p \to q) \land (p \land q \to r) \to (\neg p \to r))$$

$p \mapsto \bot$

$r \mapsto \bot$

The formula is satisfiable.

To find a model of this formula, we should simply collect choices made on the branch terminating at $\top$.

Any interpretation I such that $I(p) = I(r) = \mathbf{0}$ satisfies the formula, for example the interpretation $\{p \mapsto \mathbf{0}, q \mapsto \mathbf{0}, r \mapsto \mathbf{0}\}$.

Splitting method, example 2

$$\neg((p \to q) \land (p \land q \to r) \to (\neg p \to r))$$

$p \mapsto \bot$

$$\neg((\bot \to q) \land (\bot \land \neg q \to r) \to (\neg\bot \to r))$$

$r \mapsto \bot$

The formula is satisfiable.

To find a model of this formula, we should simply collect choices made on the branch terminating at $\top$.

Any interpretation I such that $I(p) = I(r) = \mathbf{0}$ satisfies the formula, for example the interpretation $\{p \mapsto \mathbf{0}, q \mapsto \mathbf{0}, r \mapsto \mathbf{0}\}$.

Splitting method, example 2

$$\neg((p \to q) \land (p \land q \to r) \to (\neg p \to r))$$

$$p \mapsto \perp$$

$$\neg r$$

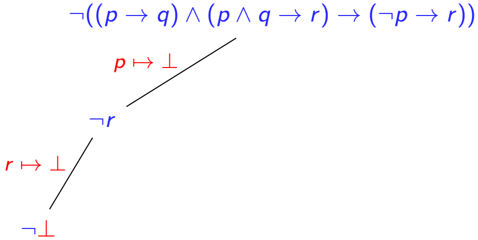
$$r \mapsto \perp$$

The formula is satisfiable.

To find a model of this formula, we should simply collect choices made on the branch terminating at $\top$.

Any interpretation I such that $I(p) = I(r) = \mathbf{0}$ satisfies the formula, for example the interpretation $\{p \mapsto \mathbf{0}, q \mapsto \mathbf{0}, r \mapsto \mathbf{0}\}$.

Splitting method, example 2



The formula is satisfiable.

To find a model of this formula, we should simply collect choices made on the branch terminating at $\top$.

Any interpretation I such that $I(p) = I(r) = \mathbf{0}$ satisfies the formula, for example the interpretation $\{p \mapsto \mathbf{0}, q \mapsto \mathbf{0}, r \mapsto \mathbf{0}\}$.

Splitting method, example 2

$$\neg((p \to q) \land (p \land q \to r) \to (\neg p \to r))$$

$p \mapsto \bot$

$\neg r$

$r \mapsto \bot$

$\top$

The formula is satisfiable.

To find a model of this formula, we should simply collect choices made on the branch terminating at $\top$.

Any interpretation I such that $I(p) = I(r) = \mathbf{0}$ satisfies the formula, for example the interpretation $\{p \mapsto \mathbf{0}, q \mapsto \mathbf{0}, r \mapsto \mathbf{0}\}$.

Splitting method, example 2

$$\neg((p \to q) \land (p \land q \to r) \to (\neg p \to r))$$

$$p \mapsto \bot$$

$$\neg r$$

$$r \mapsto \bot$$

$$\top$$

The formula is satisfiable.

To find a model of this formula, we should simply collect choices made on the branch terminating at $\top$.

Any interpretation I such that $I(p) = I(r) = \mathbf{0}$ satisfies the formula, for example the interpretation $\{p \mapsto \mathbf{0}, q \mapsto \mathbf{0}, r \mapsto \mathbf{0}\}$.

Splitting method, example 2

$$\neg((p \to q) \land (p \land q \to r) \to (\neg p \to r))$$

$p \mapsto \bot$

$$\neg r$$

$r \mapsto \bot$

$$\top$$

The formula is satisfiable.

To find a model of this formula, we should simply collect choices made on the branch terminating at $\top$.

Any interpretation I such that $I(p) = I(r) = \mathbf{0}$ satisfies the formula, for example the interpretation $\{p \mapsto \mathbf{0}, q \mapsto \mathbf{0}, r \mapsto \mathbf{0}\}$.

Splitting method, example 2

$$\neg((p \to q) \land (p \land q \to r) \to (\neg p \to r))$$

$p \mapsto \bot$

$$\neg r$$

$r \mapsto \bot$

$$\top$$

The formula is satisfiable.

To find a model of this formula, we should simply collect choices made on the branch terminating at $\top$.

Any interpretation I such that $I(p) = I(r) = \mathbf{0}$ satisfies the formula, for example the interpretation $\{p \mapsto \mathbf{0}, q \mapsto \mathbf{0}, r \mapsto \mathbf{0}\}$.

Summary

Reasoning methods:

- soundness, completeness, termination, decision procedure

Next: Normal forms, CNF, resolution, DPLL.

Summary

Reasoning methods:

- soundness, completeness, termination, decision procedure

Next: Normal forms, CNF, resolution, DPLL.

Section 5 Normal Forms

Normal Forms

In order to optimise satisfiability algorithms we need to consider formulas in a particular normal form.

In order that our algorithm can be used for all formulas we need to:

1. transform any given formula in to its normal form
   - such a normal form will be satisfiable if and only if the original formula is satisfiable
2. apply our satisfiability algorithm to this normal form

The most used normal forms for satisfiability are conjunctive normal form (clausal normal form) introduced below.

Normal Forms

In order to optimise satisfiability algorithms we need to consider formulas in a particular normal form.

In order that our algorithm can be used for all formulas we need to:

1. transform any given formula in to its normal form
   - such a normal form will be satisfiable if and only if the original formula is satisfiable
2. apply our satisfiability algorithm to this normal form

The most used normal forms for satisfiability are conjunctive normal form (clausal normal form) introduced below.

Normal Forms

In order to optimise satisfiability algorithms we need to consider formulas in a particular normal form.

In order that our algorithm can be used for all formulas we need to:

1. transform any given formula in to its normal form
   - such a normal form will be satisfiable if and only if the original formula is satisfiable
2. apply our satisfiability algorithm to this normal form

The most used normal forms for satisfiability are conjunctive normal form (clausal normal form) introduced below.

Literal, clause

- Literal: either an atom p (*positive literal*) or its negation $\neg p$ (*negative literal*).
- The complementary literal to L:

$$\overline{L} \stackrel{\text{def}}{=} \begin{cases} \neg L, & \text{if } L \text{ is positive;} \\ p, & \text{if } L \text{ has the form } \neg p. \end{cases}$$

  In other words, p and $\neg p$ are complementary.
- Clause: a disjunction $L_1 \vee \ldots \vee L_n$, $n \geq 0$ of literals.
  A clause can be seen as a mulit-set of literals $\{L_1, \ldots, L_n\}$.
- Empty clause, denoted by $\perp$: $n = 0$ (also denoted as $\square$, the empty clause is false in every interpretation).
- Unit clause: $n = 1$.

Literal, clause

- Literal: either an atom p (*positive literal*) or its negation $\neg p$ (*negative literal*).
- The complementary literal to L:

$$\overline{L} \stackrel{\text{def}}{=} \begin{cases} \neg L, & \text{if } L \text{ is positive;} \\ p, & \text{if } L \text{ has the form } \neg p. \end{cases}$$

  In other words, p and $\neg p$ are complementary.

- Clause: a disjunction $L_1 \lor \ldots \lor L_n$, $n \geq 0$ of literals.

  A clause can be seen as a mulit-set of literals $\{L_1, \ldots, L_n\}$.

- Empty clause, denoted by $\perp$: $n = 0$ (also denoted as $\Box$, the empty clause is false in every interpretation).
- Unit clause: $n = 1$.

Literal, clause

- Literal: either an atom p (*positive literal*) or its negation $\neg p$ (*negative literal*).
- The complementary literal to L:

$$\overline{L} \stackrel{\text{def}}{=} \begin{cases} \neg L, & \text{if } L \text{ is positive;} \\ p, & \text{if } L \text{ has the form } \neg p. \end{cases}$$

  In other words, p and $\neg p$ are complementary.
- Clause: a disjunction $L_1 \vee \ldots \vee L_n$, $n \geq 0$ of literals.
  A clause can be seen as a mulit-set of literals $\{L_1, \ldots, L_n\}$.
- Empty clause, denoted by $\perp$: $n = 0$ (also denoted as $\Box$, the empty clause is false in every interpretation).
- Unit clause: $n = 1$.

Literal, clause

- Literal: either an atom p (*positive literal*) or its negation $\neg p$ (*negative literal*).
- The complementary literal to L:

$$\overline{L} \stackrel{\text{def}}{=} \begin{cases} \neg L, & \text{if } L \text{ is positive;} \\ p, & \text{if } L \text{ has the form } \neg p. \end{cases}$$

  In other words, p and $\neg p$ are complementary.
- Clause: a disjunction $L_1 \lor \ldots \lor L_n$, $n \geq 0$ of literals.
  A clause can be seen as a mulit-set of literals $\{L_1, \ldots, L_n\}$.
- Empty clause, denoted by $\perp$: $n = 0$ (also denoted as $\square$, the empty clause is false in every interpretation).
- Unit clause: $n = 1$.

Literal, clause

- Literal: either an atom p (*positive literal*) or its negation $\neg p$ (*negative literal*).
- The complementary literal to L:

$$\overline{L} \stackrel{\text{def}}{=} \begin{cases} \neg L, & \text{if } L \text{ is positive;} \\ p, & \text{if } L \text{ has the form } \neg p. \end{cases}$$

  In other words, p and $\neg p$ are complementary.
- Clause: a disjunction $L_1 \vee \ldots \vee L_n$, $n \geq 0$ of literals.
  A clause can be seen as a mulit-set of literals $\{L_1, \ldots, L_n\}$.
- Empty clause, denoted by $\perp$: $n = 0$ (also denoted as $\Box$, the empty clause is false in every interpretation).
- Unit clause: $n = 1$.

CNF

- A formula A is in conjunctive normal form, or simply CNF, if it is either $\top$, or $\bot$, or a conjunction of disjunctions of literals:

$$A = \bigwedge_i \bigvee_j L_{i,j}.$$

  That is, a conjunction of clauses.
- A formula B is a conjunctive normal form of a formula A if B is equivalent to A and B is in conjunctive normal form.

Example: $(p \vee \neg q \vee \neg s) \wedge \neg q \wedge (s \vee \neg p \vee \neg p)$

Notation (Set of clauses): $\{p \vee \neg q \vee \neg s, \neg q, s \vee \neg p \vee \neg p\}$, or
$\{\{p, \neg q, \neg s\}, \{\neg q\}, \{s, \neg p, \neg p\}\}$.

CNF

- A formula A is in conjunctive normal form, or simply CNF, if it is either $\top$, or $\bot$, or a conjunction of disjunctions of literals:

  $$A = \bigwedge_i \bigvee_j L_{i,j}.$$

  That is, a conjunction of clauses.
- A formula B is a conjunctive normal form of a formula A if B is equivalent to A and B is in conjunctive normal form.

Example: $(p \lor \neg q \lor \neg s) \land \neg q \land (s \lor \neg p \lor \neg p)$

Notation (Set of clauses): $\{p \lor \neg q \lor \neg s, \neg q, s \lor \neg p \lor \neg p\}$, or
$\{\{p, \neg q, \neg s\}, \{\neg q\}, \{s, \neg p, \neg p\}\}$.

CNF: Truth Tables

Theorem. For every formula there is an equivalent CNF.

Algorithm 1. (Truth Tables). If all rows have value **1** then $A \equiv \top$.

p	q	$A(p,q)$
0	**0**	**1**
0	**1**	**0**
1	**0**	**1**
1	**1**	**0**

Goal: find a set of disjunctions equiv. to $A(p,q)$

Consider a row with **0** value: **0** **1** **0**

Add for such row: $p \vee \neg q$

Next row: **1** **1** **0**

$\neg p \vee \neg q$

Resulting formula: $A(p,q) \equiv (p \vee \neg q) \wedge (\neg p \vee \neg q)$

Check $A(p,q) \equiv (p \vee \neg q) \wedge (\neg p \vee \neg q)$:

Consider a row with **0** value – then the added disjunct is false.

Consider a row with **1** value – then all disjuncts are true.

Corollary. Every boolean function can be represented by a CNF.

Corollary. Every propositional formula has a conjunctive normal form.

CNF: Truth Tables

Theorem. For every formula there is an equivalent CNF.

Algorithm 1. (Truth Tables). If all rows have value **1** then $A \equiv \top$.

p	q	$A(p,q)$
0	**0**	**1**
0	**1**	**0**
1	**0**	**1**
1	**1**	**0**

Goal: find a set of disjunctions equiv. to $A(p,q)$

Consider a row with **0** value: **0** **1** **0**

Add for such row: $p \vee \neg q$

Next row: **1** **1** **0**

$\neg p \vee \neg q$

Resulting formula: $A(p,q) \equiv (p \vee \neg q) \wedge (\neg p \vee \neg q)$

Check $A(p,q) \equiv (p \vee \neg q) \wedge (\neg p \vee \neg q)$:

Consider a row with **0** value – then the added disjunct is false.

Consider a row with **1** value – then all disjuncts are true.

Corollary. Every boolean function can be represented by a CNF.

Corollary. Every propositional formula has a conjunctive normal form.

CNF: Truth Tables

Theorem. For every formula there is an equivalent CNF.

Algorithm 1. (Truth Tables). If all rows have value **1** then $A \equiv \top$.

p	q	$A(p,q)$
0	**0**	**1**
0	**1**	**0**
1	**0**	**1**
1	**1**	**0**

Goal: find a set of disjunctions equiv. to $A(p,q)$

Consider a row with **0** value: **0** **1** **0**

Add for such row: $p \vee \neg q$

Next row: **1** **1** **0**

$\neg p \vee \neg q$

Resulting formula: $A(p,q) \equiv (p \vee \neg q) \wedge (\neg p \vee \neg q)$

Check $A(p,q) \equiv (p \vee \neg q) \wedge (\neg p \vee \neg q)$:

Consider a row with **0** value – then the added disjunct is false.

Consider a row with **1** value – then all disjuncts are true.

Corollary. Every boolean function can be represented by a CNF.

Corollary. Every propositional formula has a conjunctive normal form.

CNF: Truth Tables

Theorem. For every formula there is an equivalent CNF.

Algorithm 1. (Truth Tables). If all rows have value **1** then $A \equiv \top$.

p	q	$A(p,q)$
0	**0**	**1**
0	**1**	**0**
1	**0**	**1**
1	**1**	**0**

Goal: find a set of disjunctions equiv. to $A(p,q)$

Consider a row with **0** value: **0** **1** **0**

Add for such row: $p \vee \neg q$

Next row: **1** **1** **0**

$\neg p \vee \neg q$

Resulting formula: $A(p,q) \equiv (p \vee \neg q) \wedge (\neg p \vee \neg q)$

Check $A(p,q) \equiv (p \vee \neg q) \wedge (\neg p \vee \neg q)$:

Consider a row with **0** value – then the added disjunct is false.

Consider a row with **1** value – then all disjuncts are true.

Corollary. Every boolean function can be represented by a CNF.

Corollary. Every propositional formula has a conjunctive normal form.

CNF: Truth Tables

Theorem. For every formula there is an equivalent CNF.

Algorithm 1. (Truth Tables). If all rows have value **1** then $A \equiv \top$.

p	q	$A(p,q)$
0	**0**	**1**
0	**1**	**0**
1	**0**	**1**
1	**1**	**0**

Goal: find a set of disjunctions equiv. to $A(p,q)$

Consider a row with **0** value: **0** **1** **0**

Add for such row: $p \vee \neg q$

Next row: **1** **1** **0**

$\neg p \vee \neg q$

Resulting formula: $A(p,q) \equiv (p \vee \neg q) \wedge (\neg p \vee \neg q)$

Check $A(p,q) \equiv (p \vee \neg q) \wedge (\neg p \vee \neg q)$:

Consider a row with **0** value – then the added disjunct is false.

Consider a row with **1** value – then all disjuncts are true.

Corollary. Every boolean function can be represented by a CNF.

Corollary. Every propositional formula has a conjunctive normal form.

CNF: Truth Tables

Theorem. For every formula there is an equivalent CNF.

Algorithm 1. (Truth Tables). If all rows have value **1** then $A \equiv \top$.

p	q	$A(p,q)$
0	**0**	**1**
0	**1**	**0**
1	**0**	**1**
1	**1**	**0**

Goal: find a set of disjunctions equiv. to $A(p,q)$

Consider a row with **0** value: **0** **1** **0**

Add for such row: $p \lor \neg q$

Next row: **1** **1** **0**

$\neg p \lor \neg q$

Resulting formula: $A(p,q) \equiv (p \lor \neg q) \land (\neg p \lor \neg q)$

Check $A(p,q) \equiv (p \lor \neg q) \land (\neg p \lor \neg q)$:

Consider a row with **0** value – then the added disjunct is false.

Consider a row with **1** value – then all disjuncts are true.

Corollary. Every boolean function can be represented by a CNF.

Corollary. Every propositional formula has a conjunctive normal form.

CNF: Truth Tables

Theorem. For every formula there is an equivalent CNF.

Algorithm 1. (Truth Tables). If all rows have value **1** then $A \equiv \top$.

p	q	$A(p,q)$
0	**0**	**1**
0	**1**	**0**
1	**0**	**1**
1	**1**	**0**

Goal: find a set of disjunctions equiv. to $A(p,q)$

Consider a row with **0** value: **0** **1** **0**

Add for such row: $p \vee \neg q$

Next row: **1** **1** **0**

$\neg p \vee \neg q$

Resulting formula: $A(p,q) \equiv (p \vee \neg q) \wedge (\neg p \vee \neg q)$

Check $A(p,q) \equiv (p \vee \neg q) \wedge (\neg p \vee \neg q)$:

Consider a row with **0** value – then the added disjunct is false.

Consider a row with **1** value – then all disjuncts are true.

Corollary. Every boolean function can be represented by a CNF.

Corollary. Every propositional formula has a conjunctive normal form.

Conversion the conjunctive normal form

Algorithm 2. (Syntactic transformation).

$$
\begin{array}{rcl}
F \leftrightarrow G & \Rightarrow_{\mathrm{CNF}} & (F \rightarrow G) \wedge (G \rightarrow F) \\
F \rightarrow G & \Rightarrow_{\mathrm{CNF}} & (\neg F \vee G) \\
\neg(F \vee G) & \Rightarrow_{\mathrm{CNF}} & (\neg F \wedge \neg G) \\
\neg(F \wedge G) & \Rightarrow_{\mathrm{CNF}} & (\neg F \vee \neg G) \\
\neg\neg F & \Rightarrow_{\mathrm{CNF}} & F
\end{array}
$$

$$
\begin{array}{rclrcl}
F \wedge \top & \Rightarrow_{\mathrm{CNF}} & F & F \wedge \bot & \Rightarrow_{\mathrm{CNF}} & \bot \\
F \vee \top & \Rightarrow_{\mathrm{CNF}} & \top & F \vee \bot & \Rightarrow_{\mathrm{CNF}} & F \\
\neg\top & \Rightarrow_{\mathrm{CNF}} & \bot & \neg\bot & \Rightarrow_{\mathrm{CNF}} & \top
\end{array}
$$

$$
(F \wedge G) \vee H \Rightarrow_{\mathrm{CNF}} (F \vee H) \wedge (G \vee H)
$$

These rules are applied modulo associativity and commutativity of $\wedge$ and $\vee$.

Theorem. For any formula F after a finite number of applications $\Rightarrow_{\mathrm{CNF}}$ we obtain a CNF of F.

The first five rules compute the negation normal form (NNF) of a formula.

Conversion the conjunctive normal form

Algorithm 2. (Syntactic transformation).

$$
\begin{array}{rcl}
F \leftrightarrow G & \Rightarrow_{\mathrm{CNF}} & (F \rightarrow G) \wedge (G \rightarrow F) \\
F \rightarrow G & \Rightarrow_{\mathrm{CNF}} & (\neg F \vee G) \\
\neg(F \vee G) & \Rightarrow_{\mathrm{CNF}} & (\neg F \wedge \neg G) \\
\neg(F \wedge G) & \Rightarrow_{\mathrm{CNF}} & (\neg F \vee \neg G) \\
\neg\neg F & \Rightarrow_{\mathrm{CNF}} & F
\end{array}
$$

$$
\begin{array}{rclrcl}
F \wedge \top & \Rightarrow_{\mathrm{CNF}} & F & F \wedge \bot & \Rightarrow_{\mathrm{CNF}} & \bot \\
F \vee \top & \Rightarrow_{\mathrm{CNF}} & \top & F \vee \bot & \Rightarrow_{\mathrm{CNF}} & F \\
\neg \top & \Rightarrow_{\mathrm{CNF}} & \bot & \neg \bot & \Rightarrow_{\mathrm{CNF}} & \top
\end{array}
$$

$$
(F \wedge G) \vee H \Rightarrow_{\mathrm{CNF}} (F \vee H) \wedge (G \vee H)
$$

These rules are applied modulo associativity and commutativity of $\wedge$ and $\vee$.

Theorem. For any formula F after a finite number of applications $\Rightarrow_{\mathrm{CNF}}$ we obtain a CNF of F.

The first five rules compute the negation normal form (NNF) of a formula.

Conversion the conjunctive normal form

Algorithm 2. (Syntactic transformation).

$$
\begin{array}{rcl}
F \leftrightarrow G & \Rightarrow_{\mathrm{CNF}} & (F \rightarrow G) \wedge (G \rightarrow F) \\
F \rightarrow G & \Rightarrow_{\mathrm{CNF}} & (\neg F \vee G) \\
\neg(F \vee G) & \Rightarrow_{\mathrm{CNF}} & (\neg F \wedge \neg G) \\
\neg(F \wedge G) & \Rightarrow_{\mathrm{CNF}} & (\neg F \vee \neg G) \\
\neg\neg F & \Rightarrow_{\mathrm{CNF}} & F
\end{array}
$$

$$
\begin{array}{rclrcl}
F \wedge \top & \Rightarrow_{\mathrm{CNF}} & F & F \wedge \bot & \Rightarrow_{\mathrm{CNF}} & \bot \\
F \vee \top & \Rightarrow_{\mathrm{CNF}} & \top & F \vee \bot & \Rightarrow_{\mathrm{CNF}} & F \\
\neg\top & \Rightarrow_{\mathrm{CNF}} & \bot & \neg\bot & \Rightarrow_{\mathrm{CNF}} & \top
\end{array}
$$

$$
(F \wedge G) \vee H \Rightarrow_{\mathrm{CNF}} (F \vee H) \wedge (G \vee H)
$$

These rules are applied modulo associativity and commutativity of $\wedge$ and $\vee$.

Theorem. For any formula F after a finite number of applications $\Rightarrow_{\mathrm{CNF}}$ we obtain a CNF of F.

The first five rules compute the negation normal form (NNF) of a formula.

CNF, example

$$\neg((p \to q) \land (p \land q \to r) \to (p \to r)) \Rightarrow$$
$$\neg(\neg((p \to q) \land (p \land q \to r)) \lor (p \to r)) \Rightarrow_{\mathrm{CNF}}$$
$$\neg\neg((p \to q) \land (p \land q \to r)) \land \neg(p \to r) \Rightarrow_{\mathrm{CNF}}$$
$$(p \to q) \land (p \land q \to r) \land \neg(p \to r) \Rightarrow_{\mathrm{CNF}}$$
$$(p \to q) \land (p \land q \to r) \land \neg(\neg p \lor r) \Rightarrow_{\mathrm{CNF}}$$
$$(p \to q) \land (p \land q \to r) \land \neg\neg p \land \neg r \Rightarrow_{\mathrm{CNF}}$$
$$(p \to q) \land (p \land q \to r) \land p \land \neg r \Rightarrow_{\mathrm{CNF}}$$
$$(p \to q) \land (\neg(p \land q) \lor r) \land p \land \neg r \Rightarrow_{\mathrm{CNF}}$$
$$(p \to q) \land (\neg p \lor \neg q \lor r) \land p \land \neg r \Rightarrow_{\mathrm{CNF}}$$
$$(\neg p \lor q) \land (\neg p \lor \neg q \lor r) \land p \land \neg r$$

CNF, example

$$\neg((p \to q) \land (p \land q \to r) \to (p \to r)) \Rightarrow$$
$$\neg(\neg((p \to q) \land (p \land q \to r)) \lor (p \to r)) \Rightarrow_{\mathrm{CNF}}$$
$$\neg\neg((p \to q) \land (p \land q \to r)) \land \neg(p \to r) \Rightarrow_{\mathrm{CNF}}$$
$$(p \to q) \land (p \land q \to r) \land \neg(p \to r) \Rightarrow_{\mathrm{CNF}}$$
$$(p \to q) \land (p \land q \to r) \land \neg(\neg p \lor r) \Rightarrow_{\mathrm{CNF}}$$
$$(p \to q) \land (p \land q \to r) \land \neg\neg p \land \neg r \Rightarrow_{\mathrm{CNF}}$$
$$(p \to q) \land (p \land q \to r) \land p \land \neg r \Rightarrow_{\mathrm{CNF}}$$
$$(p \to q) \land (\neg(p \land q) \lor r) \land p \land \neg r \Rightarrow_{\mathrm{CNF}}$$
$$(p \to q) \land (\neg p \lor \neg q \lor r) \land p \land \neg r \Rightarrow_{\mathrm{CNF}}$$
$$(\neg p \lor q) \land (\neg p \lor \neg q \lor r) \land p \land \neg r$$

CNF, example

$$\neg((p \to q) \land (p \land q \to r) \to (p \to r)) \Rightarrow$$
$$\neg(\neg((p \to q) \land (p \land q \to r)) \lor (p \to r)) \Rightarrow_{\mathrm{CNF}}$$
$$\neg\neg((p \to q) \land (p \land q \to r)) \land \neg(p \to r) \Rightarrow_{\mathrm{CNF}}$$
$$(p \to q) \land (p \land q \to r) \land \neg(p \to r) \Rightarrow_{\mathrm{CNF}}$$
$$(p \to q) \land (p \land q \to r) \land \neg(\neg p \lor r) \Rightarrow_{\mathrm{CNF}}$$
$$(p \to q) \land (p \land q \to r) \land \neg\neg p \land \neg r \Rightarrow_{\mathrm{CNF}}$$
$$(p \to q) \land (p \land q \to r) \land p \land \neg r \Rightarrow_{\mathrm{CNF}}$$
$$(p \to q) \land (\neg(p \land q) \lor r) \land p \land \neg r \Rightarrow_{\mathrm{CNF}}$$
$$(p \to q) \land (\neg p \lor \neg q \lor r) \land p \land \neg r \Rightarrow_{\mathrm{CNF}}$$
$$(\neg p \lor q) \land (\neg p \lor \neg q \lor r) \land p \land \neg r$$

CNF, example

$$\neg((p \to q) \wedge (p \wedge q \to r) \to (p \to r)) \Rightarrow$$

$$\neg(\neg((p \to q) \wedge (p \wedge q \to r)) \vee (p \to r)) \Rightarrow_{\mathrm{CNF}}$$

$$\neg\neg((p \to q) \wedge (p \wedge q \to r)) \wedge \neg(p \to r) \Rightarrow_{\mathrm{CNF}}$$

$$(p \to q) \wedge (p \wedge q \to r) \wedge \neg(p \to r) \Rightarrow_{\mathrm{CNF}}$$

$$(p \to q) \wedge (p \wedge q \to r) \wedge \neg(\neg p \vee r) \Rightarrow_{\mathrm{CNF}}$$

$$(p \to q) \wedge (p \wedge q \to r) \wedge \neg\neg p \wedge \neg r \Rightarrow_{\mathrm{CNF}}$$

$$(p \to q) \wedge (p \wedge q \to r) \wedge p \wedge \neg r \Rightarrow_{\mathrm{CNF}}$$

$$(p \to q) \wedge (\neg(p \wedge q) \vee r) \wedge p \wedge \neg r \Rightarrow_{\mathrm{CNF}}$$

$$(p \to q) \wedge (\neg p \vee \neg q \vee r) \wedge p \wedge \neg r \Rightarrow_{\mathrm{CNF}}$$

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

CNF, example

$$\neg((p \to q) \wedge (p \wedge q \to r) \to (p \to r)) \Rightarrow$$

$$\neg(\neg((p \to q) \wedge (p \wedge q \to r)) \vee (p \to r)) \Rightarrow_{\mathrm{CNF}}$$

$$\neg\neg((p \to q) \wedge (p \wedge q \to r)) \wedge \neg(p \to r) \Rightarrow_{\mathrm{CNF}}$$

$$(p \to q) \wedge (p \wedge q \to r) \wedge \neg(p \to r) \Rightarrow_{\mathrm{CNF}}$$

$$(p \to q) \wedge (p \wedge q \to r) \wedge \neg(\neg p \vee r) \Rightarrow_{\mathrm{CNF}}$$

$$(p \to q) \wedge (p \wedge q \to r) \wedge \neg\neg p \wedge \neg r \Rightarrow_{\mathrm{CNF}}$$

$$(p \to q) \wedge (p \wedge q \to r) \wedge p \wedge \neg r \Rightarrow_{\mathrm{CNF}}$$

$$(p \to q) \wedge (\neg(p \wedge q) \vee r) \wedge p \wedge \neg r \Rightarrow_{\mathrm{CNF}}$$

$$(p \to q) \wedge (\neg p \vee \neg q \vee r) \wedge p \wedge \neg r \Rightarrow_{\mathrm{CNF}}$$

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

CNF, example

$$\neg((p \to q) \land (p \land q \to r) \to (p \to r)) \Rightarrow$$

$$\neg(\neg((p \to q) \land (p \land q \to r)) \lor (p \to r)) \Rightarrow_{\mathrm{CNF}}$$

$$\neg\neg((p \to q) \land (p \land q \to r)) \land \neg(p \to r) \Rightarrow_{\mathrm{CNF}}$$

$$(p \to q) \land (p \land q \to r) \land \neg(p \to r) \Rightarrow_{\mathrm{CNF}}$$

$$(p \to q) \land (p \land q \to r) \land \neg(\neg p \lor r) \Rightarrow_{\mathrm{CNF}}$$

$$(p \to q) \land (p \land q \to r) \land \neg\neg p \land \neg r \Rightarrow_{\mathrm{CNF}}$$

$$(p \to q) \land (p \land q \to r) \land p \land \neg r \Rightarrow_{\mathrm{CNF}}$$

$$(p \to q) \land (\neg(p \land q) \lor r) \land p \land \neg r \Rightarrow_{\mathrm{CNF}}$$

$$(p \to q) \land (\neg p \lor \neg q \lor r) \land p \land \neg r \Rightarrow_{\mathrm{CNF}}$$

$$(\neg p \lor q) \land (\neg p \lor \neg q \lor r) \land p \land \neg r$$

CNF, example

$\neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)) \Rightarrow$

$\neg(\neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r)) \vee (p \rightarrow r)) \Rightarrow_{\mathrm{CNF}}$

$\neg\neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r)) \wedge \neg(p \rightarrow r) \Rightarrow_{\mathrm{CNF}}$

$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \wedge \neg(p \rightarrow r) \Rightarrow_{\mathrm{CNF}}$

$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \wedge \neg(\neg p \vee r) \Rightarrow_{\mathrm{CNF}}$

$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \wedge \neg\neg p \wedge \neg r \Rightarrow_{\mathrm{CNF}}$

$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \wedge p \wedge \neg r \Rightarrow_{\mathrm{CNF}}$

$(p \rightarrow q) \wedge (\neg(p \wedge q) \vee r) \wedge p \wedge \neg r \Rightarrow_{\mathrm{CNF}}$

$(p \rightarrow q) \wedge (\neg p \vee \neg q \vee r) \wedge p \wedge \neg r \Rightarrow_{\mathrm{CNF}}$

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

CNF, example

$$\neg((p \to q) \land (p \land q \to r) \to (p \to r)) \Rightarrow$$

$$\neg(\neg((p \to q) \land (p \land q \to r)) \lor (p \to r)) \Rightarrow_{\mathrm{CNF}}$$

$$\neg\neg((p \to q) \land (p \land q \to r)) \land \neg(p \to r) \Rightarrow_{\mathrm{CNF}}$$

$$(p \to q) \land (p \land q \to r) \land \neg(p \to r) \Rightarrow_{\mathrm{CNF}}$$

$$(p \to q) \land (p \land q \to r) \land \neg(\neg p \lor r) \Rightarrow_{\mathrm{CNF}}$$

$$(p \to q) \land (p \land q \to r) \land \neg\neg p \land \neg r \Rightarrow_{\mathrm{CNF}}$$

$$(p \to q) \land (p \land q \to r) \land p \land \neg r \Rightarrow_{\mathrm{CNF}}$$

$$(p \to q) \land (\neg(p \land q) \lor r) \land p \land \neg r \Rightarrow_{\mathrm{CNF}}$$

$$(p \to q) \land (\neg p \lor \neg q \lor r) \land p \land \neg r \Rightarrow_{\mathrm{CNF}}$$

$$(\neg p \lor q) \land (\neg p \lor \neg q \lor r) \land p \land \neg r$$

CNF, example

$$\neg((p \to q) \land (p \land q \to r) \to (p \to r)) \Rightarrow$$

$$\neg(\neg((p \to q) \land (p \land q \to r)) \lor (p \to r)) \Rightarrow_{\mathrm{CNF}}$$

$$\neg\neg((p \to q) \land (p \land q \to r)) \land \neg(p \to r) \Rightarrow_{\mathrm{CNF}}$$

$$(p \to q) \land (p \land q \to r) \land \neg(p \to r) \Rightarrow_{\mathrm{CNF}}$$

$$(p \to q) \land (p \land q \to r) \land \neg(\neg p \lor r) \Rightarrow_{\mathrm{CNF}}$$

$$(p \to q) \land (p \land q \to r) \land \neg\neg p \land \neg r \Rightarrow_{\mathrm{CNF}}$$

$$(p \to q) \land (p \land q \to r) \land p \land \neg r \Rightarrow_{\mathrm{CNF}}$$

$$(p \to q) \land (\neg(p \land q) \lor r) \land p \land \neg r \Rightarrow_{\mathrm{CNF}}$$

$$(p \to q) \land (\neg p \lor \neg q \lor r) \land p \land \neg r \Rightarrow_{\mathrm{CNF}}$$

$$(\neg p \lor q) \land (\neg p \lor \neg q \lor r) \land p \land \neg r$$

CNF, example

$\neg((p \to q) \wedge (p \wedge q \to r) \to (p \to r)) \Rightarrow$

$\neg(\neg((p \to q) \wedge (p \wedge q \to r)) \vee (p \to r)) \Rightarrow_{\mathrm{CNF}}$

$\neg\neg((p \to q) \wedge (p \wedge q \to r)) \wedge \neg(p \to r) \Rightarrow_{\mathrm{CNF}}$

$(p \to q) \wedge (p \wedge q \to r) \wedge \neg(p \to r) \Rightarrow_{\mathrm{CNF}}$

$(p \to q) \wedge (p \wedge q \to r) \wedge \neg(\neg p \vee r) \Rightarrow_{\mathrm{CNF}}$

$(p \to q) \wedge (p \wedge q \to r) \wedge \neg\neg p \wedge \neg r \Rightarrow_{\mathrm{CNF}}$

$(p \to q) \wedge (p \wedge q \to r) \wedge p \wedge \neg r \Rightarrow_{\mathrm{CNF}}$

$(p \to q) \wedge (\neg(p \wedge q) \vee r) \wedge p \wedge \neg r \Rightarrow_{\mathrm{CNF}}$

$(p \to q) \wedge (\neg p \vee \neg q \vee r) \wedge p \wedge \neg r \Rightarrow_{\mathrm{CNF}}$

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

CNF, example

Therefore, the formula

$$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$$

has the same models as the set consisting of four clauses

$$\begin{gathered}\neg p \lor q \\ \neg p \lor \neg q \lor r \\ p \\ \neg r\end{gathered}$$

The CNF transformation allows one to reduce the satisfiability problem for formulas to the satisfiability problem for sets of clauses.

CNF can be Exponential

Consider:

$$F = (p_1^1 \wedge p_1^2) \vee (p_2^1 \wedge p_2^2) \vee \cdots \vee (p_k^1 \wedge p_k^2).$$

CNF of F is:

$$\mathrm{CNF}(F) = \bigwedge_{i_j \in \{1,2\}} p_1^{i_1} \vee \ldots \vee p_k^{i_k}$$

exponential in size (w.r.t. the size of F).

Idea: Introduce names for subformulas:

$$\begin{array}{lcl} n_1 & \leftrightarrow & (p_1^1 \wedge p_1^2) \\ n_2 & \leftrightarrow & (p_2^1 \wedge p_2^2) \\ & \cdots & \\ n_k & \leftrightarrow & (p_k^1 \wedge p_k^2) \end{array}$$

Replace subformulas with their definitions in F:

$$n \leftrightarrow (n_1 \vee \ldots \vee n_k)$$
$$n$$

obtaining an equi-satisfiable formula in CNF.

CNF can be Exponential

Consider:

$$F = (p_1^1 \wedge p_1^2) \vee (p_2^1 \wedge p_2^2) \vee \cdots \vee (p_k^1 \wedge p_k^2).$$

CNF of F is:

$$\mathrm{CNF}(F) = \bigwedge_{i_j \in \{1,2\}} p_1^{i_1} \vee \ldots \vee p_k^{i_k}$$

exponential in size (w.r.t. the size of F).

Idea: Introduce names for subformulas:

$$\begin{array}{ccc} n_1 & \leftrightarrow & (p_1^1 \wedge p_1^2) \\ n_2 & \leftrightarrow & (p_2^1 \wedge p_2^2) \\ & \ldots & \\ n_k & \leftrightarrow & (p_k^1 \wedge p_k^2) \end{array}$$

Replace subformulas with their definitions in F:

$$n \leftrightarrow (n_1 \vee \ldots \vee n_k)$$
$$n$$

obtaining an equi-satisfiable formula in CNF.

CNF can be Exponential

Consider:

$$F = (p_1^1 \wedge p_1^2) \vee (p_2^1 \wedge p_2^2) \vee \cdots \vee (p_k^1 \wedge p_k^2).$$

CNF of F is:

$$\mathrm{CNF}(F) = \bigwedge_{i_j \in \{1,2\}} p_1^{i_1} \vee \ldots \vee p_k^{i_k}$$

exponential in size (w.r.t. the size of F).

Idea: Introduce names for subformulas:

$$\begin{array}{rcl} n_1 & \leftrightarrow & (p_1^1 \wedge p_1^2) \\ n_2 & \leftrightarrow & (p_2^1 \wedge p_2^2) \\ & \ldots & \\ n_k & \leftrightarrow & (p_k^1 \wedge p_k^2) \end{array}$$

Replace subformulas with their definitions in F:

$$n \leftrightarrow (n_1 \vee \ldots \vee n_k)$$
$$n$$

obtaining an equi-satisfiable formula in CNF.

Structural (definitional) CNF transformation

Theorem

$$F[G] \text{ is satisfiable} \Leftrightarrow F[n_G] \wedge (n_G \leftrightarrow G) \text{ is satisfiable.}$$

provided n_G is a (fresh) propositional variable not occurring in $F[G]$.
n_G can be seen as a name for G.

Structural CNF Transformation:

- introduce names recursively for every non-literal subformula in the original formula (this introduces a linear number of new symbols).
- Conversion of the resulting formula into CNF increases the size only by an additional constant factor
- resulting formula is in CNF and is equi-satisfiable to the original formula.

Structural (definitional) CNF transformation

Theorem

$$F[G] \text{ is satisfiable} \Leftrightarrow F[n_G] \wedge (n_G \leftrightarrow G) \text{ is satisfiable}.$$

provided n_G is a (fresh) propositional variable not occurring in $F[G]$.
n_G can be seen as a name for G.

Structural CNF Transformation:

- introduce names recursively for every non-literal subformula in the original formula (this introduces a linear number of new symbols).
- Conversion of the resulting formula into CNF increases the size only by an additional constant factor
- resulting formula is in CNF and is equi-satisfiable to the original formula.

Example

name	subformula	name definitions	clauses
			n_1
n_1	$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$	$n_1 \leftrightarrow \neg n_2$	$\neg n_1 \lor \neg n_2$
			$n_1 \lor n_2$
n_2	$(p \to q) \land (p \land q \to r) \to (p \to r)$	$n_2 \leftrightarrow (n_3 \to n_7)$	$\neg n_2 \lor \neg n_3 \lor n_7$
			$n_3 \lor n_2$
			$\neg n_7 \lor n_2$
n_3	$(p \to q) \land (p \land q \to r)$	$n_3 \leftrightarrow (n_4 \land n_5)$	$\neg n_3 \lor n_4$
			$\neg n_3 \lor n_5$
			$\neg n_4 \lor \neg n_5 \lor n_3$
n_4	$p \to q$	$n_4 \leftrightarrow (p \to q)$	$\neg n_4 \lor \neg p \lor q$
			$p \lor n_4$
			$\neg q \lor n_4$

Example

name	subformula	name definitions	clauses
n_5	$p \land q \rightarrow r$	$n_5 \leftrightarrow (n_6 \rightarrow r)$	$\neg n_5 \lor \neg n_6 \lor r$ $n_6 \lor n_5$ $\neg r \lor n_5$
n_6	$p \land q$	$n_6 \leftrightarrow (p \land q)$	$\neg n_6 \lor p$ $\neg n_6 \lor q$ $\neg p \lor \neg q \lor n_6$
n_7	$p \rightarrow r$	$n_7 \leftrightarrow (p \rightarrow r)$	$\neg n_7 \lor \neg p \lor r$ $p \lor n_7$ $\neg r \lor n_7$

Note: There are at most three literals in each clause!

Theorem

Any propositional formula can be transformed in linear time into an equi-satisfiable CNF. Moreover each clause in such CNF contains at most three literals (3-CNF).

Example

name	subformula	name definitions	clauses
n_5	$p \wedge q \rightarrow r$	$n_5 \leftrightarrow (n_6 \rightarrow r)$	$\neg n_5 \vee \neg n_6 \vee r$ $n_6 \vee n_5$ $\neg r \vee n_5$
n_6	$p \wedge q$	$n_6 \leftrightarrow (p \wedge q)$	$\neg n_6 \vee p$ $\neg n_6 \vee q$ $\neg p \vee \neg q \vee n_6$
n_7	$p \rightarrow r$	$n_7 \leftrightarrow (p \rightarrow r)$	$\neg n_7 \vee \neg p \vee r$ $p \vee n_7$ $\neg r \vee n_7$

Note: There are at most three literals in each clause!

Theorem

Any propositional formula can be transformed in linear time into an equi-satisfiable CNF. Moreover each clause in such CNF contains at most three literals (3-CNF).

Example

name	subformula	name definitions	clauses
n_5	$p \wedge q \to r$	$n_5 \leftrightarrow (n_6 \to r)$	$\neg n_5 \vee \neg n_6 \vee r$
			$n_6 \vee n_5$
			$\neg r \vee n_5$
n_6	$p \wedge q$	$n_6 \leftrightarrow (p \wedge q)$	$\neg n_6 \vee p$
			$\neg n_6 \vee q$
			$\neg p \vee \neg q \vee n_6$
n_7	$p \to r$	$n_7 \leftrightarrow (p \to r)$	$\neg n_7 \vee \neg p \vee r$
			$p \vee n_7$
			$\neg r \vee n_7$

Note: There are at most three literals in each clause!

Theorem

Any propositional formula can be transformed in linear time into an equi-satisfiable CNF. Moreover each clause in such CNF contains at most three literals (3-CNF).

Optimised Structural Transformation

A further improvement is possible by taking the polarity of the subformula F into account.

A subformula occurrence in F has

- a neutral polarity if it occurs in the scope of $\leftrightarrow$, otherwise
- a positive polarity if it occurs in the scope of an even number of negations (including left-hand sides of $\rightarrow$).
- a negative polarity if it occurs in the scope of an odd number of negations (including left-hand sides of $\rightarrow$).

Definition. (Opposite to)

- positive polarity is opposite to negative polarity
- negative polarity is opposite to positive polarity
- neutral polarity is opposite to neutral polarity

Optimised Structural Transformation

A further improvement is possible by taking the polarity of the subformula F into account.

A subformula occurrence in F has

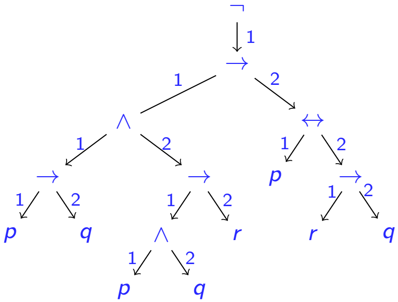
- a neutral polarity if it occurs in the scope of $\leftrightarrow$, otherwise
- a positive polarity if it occurs in the scope of an even number of negations (including left-hand sides of $\rightarrow$).
- a negative polarity if it occurs in the scope of an odd number of negations (including left-hand sides of $\rightarrow$).

Definition. (Opposite to)

- positive polarity is opposite to negative polarity
- negative polarity is opposite to positive polarity
- neutral polarity is opposite to neutral polarity

Parse tree

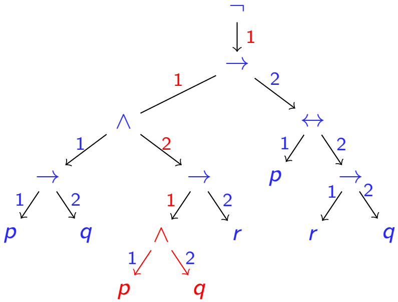
$A \stackrel{\text{def}}{=} \neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \leftrightarrow (r \rightarrow q)))$.



- Position in the formula: 1.1.2.1;
- Subformula at this position: $p \wedge q$; denoted $A|_{1.1.2.1} = p \wedge q$.
- Position of A is ϵ.

Parse tree

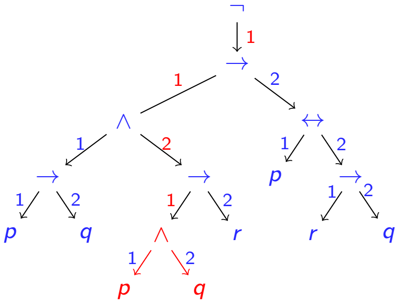
$A \stackrel{\text{def}}{=} \neg((p \to q) \land (p \land q \to r) \to (p \leftrightarrow (r \to q)))$.



- Position in the formula: 1.1.2.1;
- Subformula at this position: $p \land q$; denoted $A|_{1.1.2.1} = p \land q$.
- Position of A is ϵ.

Parse tree

$A \stackrel{\text{def}}{=} \neg((p \to q) \wedge (p \wedge q \to r) \to (p \leftrightarrow (r \to q)))$.



- Position in the formula: $1.1.2.1$;
- Subformula at this position: $p \wedge q$; denoted $A|_{1.1.2.1} = p \wedge q$.
- Position of A is ϵ.

Formal definition of polarity

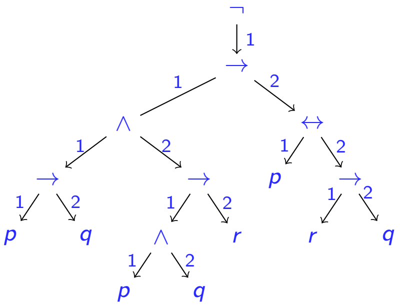
The notion of (positive, negative and neutral) polarity can be inductively defined as follows.

- F has positive polarity in F.
- Suppose G is a subformula of F.
  - If $G = \neg G'$ then G' has polarity opposite to G.
  - If $G = G_1 \star G_2$ where $\star \in \{\vee, \wedge\}$ then G_1 and G_2 have the same polarity as G in F.
  - If $G = G_1 \rightarrow G_2$ then G_1 has polarity opposite to G and G_2 has the same polarity as G.
  - $G = G_1 \leftrightarrow G_2$ then both G_1 and G_2 have neutral polarities.

The coloring algorithm for determining polarity

$\neg((p \to q) \land (p \land q \to r) \to (p \leftrightarrow (r \to q)))$.

- Color in blue all arcs below an equivalence.
- Color in red all uncolored arcs going down from a negation or left-hand side of an implication.

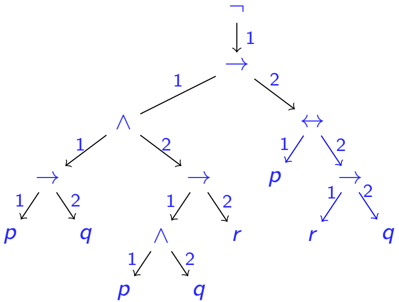


- If a position has at least one blue arc above it, its polarity is 0.
- Otherwise, its polarity is -1 if it has an odd number of red arcs above it and 1 if even.

The coloring algorithm for determining polarity

$\neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \leftrightarrow (r \rightarrow q)))$.

- Color in blue all arcs below an equivalence.
- Color in red all uncolored arcs going down from a negation or left-hand side of an implication.

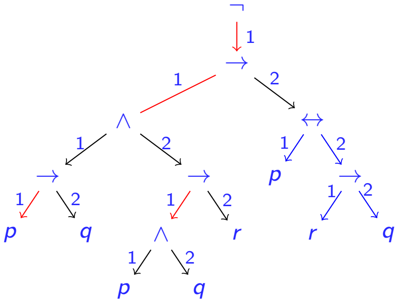


- If a position has at least one blue arc above it, its polarity is 0.
- Otherwise, its polarity is -1 if it has an odd number of red arcs above it and 1 if even.

The coloring algorithm for determining polarity

$\neg((p \to q) \land (p \land q \to r) \to (p \leftrightarrow (r \to q)))$.

- Color in blue all arcs below an equivalence.
- Color in red all uncolored arcs going down from a negation or left-hand side of an implication.

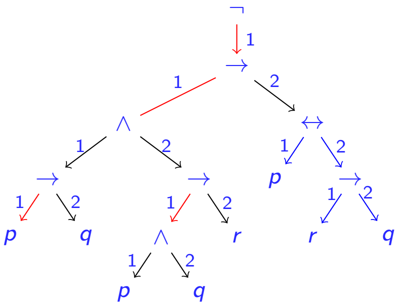


- If a position has at least one blue arc above it, its polarity is 0.
- Otherwise, its polarity is -1 if it has an odd number of red arcs above it and 1 if even.

The coloring algorithm for determining polarity

$\neg((p \to q) \land (p \land q \to r) \to (p \leftrightarrow (r \to q)))$.

- Color in blue all arcs below an equivalence.
- Color in red all uncolored arcs going down from a negation or left-hand side of an implication.

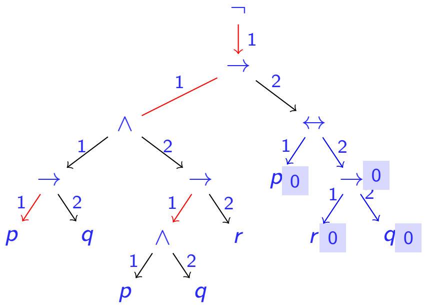


- If a position has at least one blue arc above it, its polarity is 0.
- Otherwise, its polarity is -1 if it has an odd number of red arcs above it and 1 if even.

The coloring algorithm for determining polarity

$\neg((p \to q) \land (p \land q \to r) \to (p \leftrightarrow (r \to q)))$.

- Color in blue all arcs below an equivalence.
- Color in red all uncolored arcs going down from a negation or left-hand side of an implication.

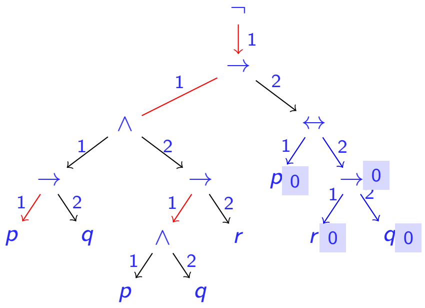


- If a position has at least one blue arc above it, its polarity is 0.
- Otherwise, its polarity is -1 if it has an odd number of red arcs above it and 1 if even.

The coloring algorithm for determining polarity

$\neg((p \to q) \land (p \land q \to r) \to (p \leftrightarrow (r \to q)))$.

- Color in blue all arcs below an equivalence.
- Color in red all uncolored arcs going down from a negation or left-hand side of an implication.

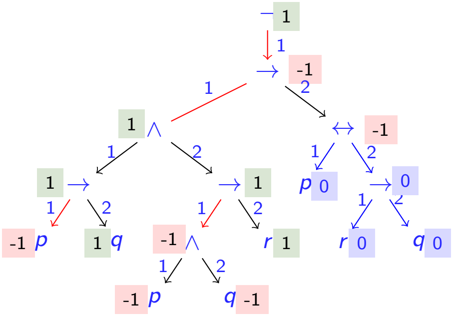


- If a position has at least one blue arc above it, its polarity is 0.
- Otherwise, its polarity is -1 if it has an odd number of red arcs above it and 1 if even.

The coloring algorithm for determining polarity

$\neg((p \to q) \wedge (p \wedge q \to r) \to (p \leftrightarrow (r \to q)))$.

- Color in blue all arcs below an equivalence.
- Color in red all uncolored arcs going down from a negation or left-hand side of an implication.



- If a position has at least one blue arc above it, its polarity is 0.
- Otherwise, its polarity is -1 if it has an odd number of red arcs above it and 1 if even.

Position and polarity

position	subformula	polarity
ε	$\neg((p \to q) \wedge (p \wedge q \to r) \to (p \to r))$	1
1	$(p \to q) \wedge (p \wedge q \to r) \to (p \to r)$	-1
1.1	$(p \to q) \wedge (p \wedge q \to r)$	1
1.1.1	$p \to q$	1
1.1.1.1	p	-1
1.1.1.2	q	1
1.1.2	$p \wedge q \to r$	1
1.1.2.1	$p \wedge q$	-1
1.1.2.1.1	p	-1
1.1.2.1.2	q	-1
1.1.2.2	r	1
1.2	$p \to r$	-1
1.2.1	p	1
1.2.2	r	-1

Position and polarity

position	subformula	polarity
ε	$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$	1
1	$(p \to q) \land (p \land q \to r) \to (p \to r)$	-1
1.1	$(p \to q) \land (p \land q \to r)$	1
1.1.1	$p \to q$	1
1.1.1.1	p	-1
1.1.1.2	q	1
1.1.2	$p \land q \to r$	1
1.1.2.1	$p \land q$	-1
1.1.2.1.1	p	-1
1.1.2.1.2	q	-1
1.1.2.2	r	1
1.2	$p \to r$	-1
1.2.1	p	1
1.2.2	r	-1

Position and polarity

position	subformula	polarity
ε	$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$	1
1	$(p \to q) \land (p \land q \to r) \to (p \to r)$	-1
1.1	$(p \to q) \land (p \land q \to r)$	1
1.1.1	$p \to q$	1
1.1.1.1	p	-1
1.1.1.2	q	1
1.1.2	$p \land q \to r$	1
1.1.2.1	$p \land q$	-1
1.1.2.1.1	p	-1
1.1.2.1.2	q	-1
1.1.2.2	r	1
1.2	$p \to r$	-1
1.2.1	p	1
1.2.2	r	-1

Position and polarity

position	subformula	polarity
ε	$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$	1
1	$(p \to q) \land (p \land q \to r) \to (p \to r)$	-1
1.1	$(p \to q) \land (p \land q \to r)$	1
1.1.1	$p \to q$	1
1.1.1.1	p	-1
1.1.1.2	q	1
1.1.2	$p \land q \to r$	1
1.1.2.1	$p \land q$	-1
1.1.2.1.1	p	-1
1.1.2.1.2	q	-1
1.1.2.2	r	1
1.2	$p \to r$	-1
1.2.1	p	1
1.2.2	r	-1

Position and polarity

position	subformula	polarity
ε	$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$	1
1	$(p \to q) \land (p \land q \to r) \to (p \to r)$	-1
1.1	$(p \to q) \land (p \land q \to r)$	1
1.1.1	$p \to q$	1
1.1.1.1	p	-1
1.1.1.2	q	1
1.1.2	$p \land q \to r$	1
1.1.2.1	$p \land q$	-1
1.1.2.1.1	p	-1
1.1.2.1.2	q	-1
1.1.2.2	r	1
1.2	$p \to r$	-1
1.2.1	p	1
1.2.2	r	-1

Position and polarity

position	subformula	polarity
ε	$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$	1
1	$(p \to q) \land (p \land q \to r) \to (p \to r)$	-1
1.1	$(p \to q) \land (p \land q \to r)$	1
1.1.1	$p \to q$	1
1.1.1.1	p	-1
1.1.1.2	q	1
1.1.2	$p \land q \to r$	1
1.1.2.1	$p \land q$	-1
1.1.2.1.1	p	-1
1.1.2.1.2	q	-1
1.1.2.2	r	1
1.2	$p \to r$	-1
1.2.1	p	1
1.2.2	r	-1

Position and polarity

position	subformula	polarity
ε	$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$	1
1	$(p \to q) \land (p \land q \to r) \to (p \to r)$	-1
1.1	$(p \to q) \land (p \land q \to r)$	1
1.1.1	$p \to q$	1
1.1.1.1	p	-1
1.1.1.2	q	1
1.1.2	$p \land q \to r$	1
1.1.2.1	$p \land q$	-1
1.1.2.1.1	p	-1
1.1.2.1.2	q	-1
1.1.2.2	r	1
1.2	$p \to r$	-1
1.2.1	p	1
1.2.2	r	-1

Position and polarity

position	subformula	polarity
ε	$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$	1
1	$(p \to q) \land (p \land q \to r) \to (p \to r)$	-1
1.1	$(p \to q) \land (p \land q \to r)$	1
$1.1.1$	$p \to q$	1
$1.1.1.1$	p	-1
$1.1.1.2$	q	1
$1.1.2$	$p \land q \to r$	1
$1.1.2.1$	$p \land q$	-1
$1.1.2.1.1$	p	-1
$1.1.2.1.2$	q	-1
$1.1.2.2$	r	1
1.2	$p \to r$	-1
$1.2.1$	p	1
$1.2.2$	r	-1

Position and polarity

position	subformula	polarity
ε	$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$	1
1	$(p \to q) \land (p \land q \to r) \to (p \to r)$	-1
1.1	$(p \to q) \land (p \land q \to r)$	1
1.1.1	$p \to q$	1
1.1.1.1	p	-1
1.1.1.2	q	1
1.1.2	$p \land q \to r$	1
1.1.2.1	$p \land q$	-1
1.1.2.1.1	p	-1
1.1.2.1.2	q	-1
1.1.2.2	r	1
1.2	$p \to r$	-1
1.2.1	p	1
1.2.2	r	-1

Optimising Structural Transformation

Theorem

Let n_G be a propositional variable not occurring in $F[G]$.

1. $F[G]$ *is satisfiable* $\iff$ $F[n_G] \wedge (n_G \rightarrow G)$ *is satisfiable, provided* G *has positive polarity in* F.
2. $F[G]$ *is satisfiable* $\iff$ $F[n_G] \wedge (G \rightarrow n_G)$ *is satisfiable, provided* G *has negative polarity in* F.
3. $F[G]$ *is satisfiable* $\iff$ $F[n_G] \wedge (n_G \leftrightarrow G)$ *is satisfiable, provided* G *has neutral polarity in* F.

Example

name	subformula	polarity	name definitions	clauses
				n_1
n_1	$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$	$+1$	$n_1 \to \neg n_2$	$\neg n_1 \lor \neg n_2$
n_2	$(p \to q) \land (p \land q \to r) \to (p \to r)$	-1	$(n_3 \to n_7) \to n_2$	$n_3 \lor n_2$ $\neg n_7 \lor n_2$
n_3	$(p \to q) \land (p \land q \to r)$	$+1$	$n_3 \to (n_4 \land n_5)$	$\neg n_3 \lor n_4$ $\neg n_3 \lor n_5$
n_4	$p \to q$	$+1$	$n_4 \to (p \to q)$	$\neg n_4 \lor \neg p \lor q$
n_5	$p \land q \to r$	$+1$	$n_5 \to (n_6 \to r)$	$\neg n_5 \lor \neg n_6 \lor r$
n_6	$p \land q$	-1	$(p \land q) \to n_6$	$\neg p \lor \neg q \lor n_6$
n_7	$p \to r$	-1	$(p \to r) \to n_7$	$p \lor n_7$ $\neg r \lor n_7$

Summary

We have studied algorithms for transforming formulas into:

- conjunctive (clause) normal form (CNF)
  - truth tables
  - syntactic transformations
- structural transformation into equi-satisfiable CNF
  - optimised structural transformation

Next: Reasoning methods for proving (un)satisfiability of sets of clauses.

Section Inference Systems, Proofs and Propositional Resolution

The Reasoning Problem

Given: S – set of clauses.

Example: $S = \{q \vee \neg p, p \vee q, \neg q\}$

We want to prove that S is unsatisfiable.

Methods studied before: Truth tables

Next: Inference systems, propositional resolution

General Idea:

- use a set of simple rules for deriving new logical consequences from S.
- use these inference rules to derive the contradiction signified by the empty clause $\perp$

The Reasoning Problem

Given: S – set of clauses.

Example: $S = \{q \vee \neg p, p \vee q, \neg q\}$

We want to prove that S is unsatisfiable.

Methods studied before: Truth tables

Next: Inference systems, propositional resolution

General Idea:

- use a set of simple rules for deriving new logical consequences from S.
- use these inference rules to derive the contradiction signified by the empty clause $\perp$

The Reasoning Problem

Given: S – set of clauses.

Example: $S = \{q \vee \neg p, p \vee q, \neg q\}$

We want to prove that S is unsatisfiable.

Methods studied before: Truth tables

Next: Inference systems, propositional resolution

General Idea:

- use a set of simple rules for deriving new logical consequences from S.
- use these inference rules to derive the contradiction signified by the empty clause $\perp$

Propositional Resolution

Propositional Resolution inference system $\mathbb{BR}$, consists of the following inference rules:

- Binary Resolution Rule (BR):

$$\frac{C \vee p \qquad \neg p \vee D}{C \vee D} \; (BR)$$

- Binary Factoring Rule (BF):

$$\frac{C \vee L \vee L}{C \vee L} \; (BF)$$

  where L is a literal.

Note: Conclusions of BR and BF are logically implied by the premises.

- $\{C \vee p, \neg p \vee D\} \models C \vee D$
- $\{C \vee L \vee L\} \models C \vee L$

Propositional Resolution

Propositional Resolution inference system $\mathbb{BR}$, consists of the following inference rules:

- Binary Resolution Rule (BR):

$$\frac{C \vee p \qquad \neg p \vee D}{C \vee D}\ (BR)$$

- Binary Factoring Rule (BF):

$$\frac{C \vee L \vee L}{C \vee L}\ (BF)$$

  where L is a literal.

Note: Conclusions of BR and BF are logically implied by the premises.

- $\{C \vee p, \neg p \vee D\} \models C \vee D$
- $\{C \vee L \vee L\} \models C \vee L$

Example

Given: $S = \{q \vee \neg p, p \vee q, \neg q\}$

A proof in resolution calculus:

$$\dfrac{\dfrac{\dfrac{q \vee \neg p \qquad p \vee q}{q \vee q}\ \text{(BR)}}{q}\ \text{(BF)} \qquad \neg q}{\bot}\ \text{(BR)}$$

Another proof in resolution calculus:

$$\dfrac{\dfrac{\dfrac{q \vee \neg p \qquad \neg q}{\neg p}\ \text{(BR)} \qquad p \vee q}{q}\ \text{(BR)} \qquad \neg q}{\bot}\ \text{(BR)}$$

Example

Given: $S = \{q \vee \neg p, p \vee q, \neg q\}$

A proof in resolution calculus:

$$\dfrac{\dfrac{\dfrac{q \vee \neg p \qquad p \vee q}{q \vee q}\,\text{(BR)}}{q}\,\text{(BF)} \qquad \neg q}{\bot}\,\text{(BR)}$$

Another proof in resolution calculus:

$$\dfrac{\dfrac{\dfrac{q \vee \neg p \qquad \neg q}{\neg p}\,\text{(BR)} \qquad p \vee q}{q}\,\text{(BR)} \qquad \neg q}{\bot}\,\text{(BR)}$$

Example

Given: $S = \{q \vee \neg p, p \vee q, \neg q\}$

A proof in resolution calculus:

$$\dfrac{\dfrac{\dfrac{q \vee \neg p \qquad p \vee q}{q \vee q}\,\text{(BR)}}{q}\,\text{(BF)} \qquad \neg q}{\bot}\,\text{(BR)}$$

Another proof in resolution calculus:

$$\dfrac{\dfrac{\dfrac{q \vee \neg p \qquad \neg q}{\neg p}\,\text{(BR)} \qquad p \vee q}{q}\,\text{(BR)} \qquad \neg q}{\bot}\,\text{(BR)}$$

Example

Given: $S = \{q \lor \neg p, p \lor q, \neg q\}$

A proof in resolution calculus:

$$\dfrac{\dfrac{\dfrac{q \lor \neg p \qquad p \lor q}{q \lor q}\,\text{(BR)}}{q}\,\text{(BF)} \qquad \neg q}{\bot}\,\text{(BR)}$$

Another proof in resolution calculus:

$$\dfrac{\dfrac{\dfrac{q \lor \neg p \qquad \neg q}{\neg p}\,\text{(BR)} \qquad p \lor q}{q}\,\text{(BR)} \qquad \neg q}{\bot}\,\text{(BR)}$$

Example

Given: $S = \{q \lor \neg p, p \lor q, \neg q\}$

A proof in resolution calculus:

$$\dfrac{\dfrac{\dfrac{q \lor \neg p \qquad p \lor q}{q \lor q}\,\text{(BR)}}{q}\,\text{(BF)} \qquad \neg q}{\bot}\,\text{(BR)}$$

Another proof in resolution calculus:

$$\dfrac{\dfrac{\dfrac{q \lor \neg p \qquad \neg q}{\neg p}\,\text{(BR)} \qquad p \lor q}{q}\,\text{(BR)} \qquad \neg q}{\bot}\,\text{(BR)}$$

Example

Given: $S = \{q \lor \neg p, p \lor q, \neg q\}$

A proof in resolution calculus:

$$\dfrac{\dfrac{\dfrac{q \lor \neg p \qquad p \lor q}{q \lor q}\ \text{(BR)}}{q}\ \text{(BF)} \qquad \neg q}{\bot}\ \text{(BR)}$$

Another proof in resolution calculus:

$$\dfrac{\dfrac{\dfrac{q \lor \neg p \qquad \neg q}{\neg p}\ \text{(BR)} \qquad p \lor q}{q}\ \text{(BR)} \qquad \neg q}{\bot}\ \text{(BR)}$$

Example

Given: $S = \{q \lor \neg p, p \lor q, \neg q\}$

A proof in resolution calculus:

$$\cfrac{\cfrac{\cfrac{q \lor \neg p \qquad p \lor q}{q \lor q}\,\text{(BR)}}{q}\,\text{(BF)} \qquad \neg q}{\bot}\,\text{(BR)}$$

Another proof in resolution calculus:

$$\cfrac{\cfrac{\cfrac{q \lor \neg p \qquad \neg q}{\neg p}\,\text{(BR)} \qquad p \lor q}{q}\,\text{(BR)} \qquad \neg q}{\bot}\,\text{(BR)}$$

Example

Given: $S = \{q \vee \neg p, p \vee q, \neg q\}$

A proof in resolution calculus:

$$\dfrac{\dfrac{\dfrac{q \vee \neg p \qquad p \vee q}{q \vee q}\,(\mathrm{BR})}{q}\,(\mathrm{BF}) \qquad \neg q}{\bot}\,(\mathrm{BR})$$

Another proof in resolution calculus:

$$\dfrac{\dfrac{\dfrac{q \vee \neg p \qquad \neg q}{\neg p}\,(\mathrm{BR}) \qquad p \vee q}{q}\,(\mathrm{BR}) \qquad \neg q}{\bot}\,(\mathrm{BR})$$

Inference System

An inference has the form:

$$\frac{F_1 \qquad \dots \qquad F_n}{G}$$

where $n \geq 0$, $F_1, \dots, F_n, G$ are formulas.

- $F_1 \dots F_n$ are called premises.
- G is called conclusion.

An inference rule R is a set of inferences.

An inference system, (or a calculus) $\mathbb{I}$ is a set of inference rules.

Inference System

An inference has the form:

$$\frac{F_1 \qquad \dots \qquad F_n}{G}$$

where $n \geq 0$, $F_1, \dots, F_n, G$ are formulas.

- $F_1 \dots F_n$ are called premises.
- G is called conclusion.

An inference rule R is a set of inferences.

An inference system, (or a calculus) $\mathbb{I}$ is a set of inference rules.

Derivation, proofs

- A derivation tree in $\mathbb{I}$ is a tree built from inferences.
- A proof of F (in $\mathbb{I}$) from $F_1, \ldots, F_n$ is a tree with leaves in $F_1, \ldots, F_n$ and the root F.
- A refutation proof is a proof of $\perp$.
- F is derivable, (or provable) in $\mathbb{I}$ from a set of formulas S, denoted $S \vdash_{\mathbb{I}} F$, if there is a proof of F from formulas in S.

Linear Proofs

Tree Proof:

$$\dfrac{\dfrac{\dfrac{q \vee \neg p \qquad p \vee q}{q \vee q}\ \text{(BR)}}{q}\ \text{(BF)} \qquad \neg q}{\bot}\ \text{(BR)}$$

Linear Proof:

1.	$q \vee \neg p$	input
2.	$p \vee q$	input
3.	$\neg q$	input
4.	$q \vee q$	BR (1,2)
5.	q	BF (4)
6.	$\bot$	BR (3,5)

Soundness

- An inference is sound if the conclusion of this inference logically follows from the premises ($\models$).
- An inference rule is sound if all its inferences are sound.
- An inference system is sound if all its inference rules are sound.

Lemma

If an inference system $\mathbb{I}$ is sound then for any set of formulas S:

$$S \vdash_{\mathbb{I}} \bot \quad \textit{implies} \quad S \models \bot$$

Theorem (Soundness)

The resolution inference system $\mathbb{BR}$ is sound.

Soundness

- An inference is sound if the conclusion of this inference logically follows from the premises ($\models$).
- An inference rule is sound if all its inferences are sound.
- An inference system is sound if all its inference rules are sound.

Lemma

If an inference system $\mathbb{I}$ is sound then for any set of formulas S:

$$S \vdash_{\mathbb{I}} \bot \quad \textit{implies} \quad S \models \bot$$

Theorem (Soundness)

The resolution inference system $\mathbb{BR}$ is sound.

Soundness

- An inference is sound if the conclusion of this inference logically follows from the premises ($\models$).
- An inference rule is sound if all its inferences are sound.
- An inference system is sound if all its inference rules are sound.

Lemma

If an inference system $\mathbb{I}$ is sound then for any set of formulas S:

$$S \vdash_{\mathbb{I}} \bot \quad \textit{implies} \quad S \models \bot$$

Theorem (Soundness)

The resolution inference system $\mathbb{BR}$ is sound.

Completeness

An inference system $\mathbb{I}$ is refutationally complete if for any set of formulas S we have:

$$S \models \bot \quad \text{implies} \quad S \vdash_{\mathbb{I}} \bot.$$

Theorem (Completness)

The resolution inference system $\mathbb{BR}$ is complete.

The proof is given later in the course.

Completeness

An inference system $\mathbb{I}$ is refutationally complete if for any set of formulas S we have:

$$S \models \bot \quad \text{implies} \quad S \vdash_{\mathbb{I}} \bot.$$

Theorem (Completness)

The resolution inference system $\mathbb{BR}$ is complete.

The proof is given later in the course.

Applications of inference systems

Formal Proofs:

- each step of a proof is easy to check
- proofs – certificates of correctness
- independent proof checking

Reasoning methods based on inference systems:

- efficient proof search
- restrictions on applicability of inference rules
- proof search strategies

Applications of inference systems

Formal Proofs:

- each step of a proof is easy to check
- proofs – certificates of correctness
- independent proof checking

Reasoning methods based on inference systems:

- efficient proof search
- restrictions on applicability of inference rules
- proof search strategies

Reasoning methods based on inference systems

Basic Idea. A Saturation Process:

Given set of clauses S we exhaustively apply all inference rules adding the conclusions to this set until the contradiction ($\perp$) is derived.

$$S_0 \Rightarrow S_1 \Rightarrow \ldots S_n \Rightarrow \ldots$$

Three outcomes:

1. $\perp$ is derived ($\perp \in S_n$ for some n), then S is unsatisfiable (provided $\mathbb{I}$ is sound);
2. no new clauses can be derived from S and $\perp \not\in S$, then S is saturated; in this case S is satisfiable, (provided $\mathbb{I}$ is complete).
3. S grows ad infinitum, the process does not terminate.

Goal: speed up the first two cases and reduce non-termination.

Reasoning methods based on inference systems

Basic Idea. A Saturation Process:

Given set of clauses S we exhaustively apply all inference rules adding the conclusions to this set until the contradiction ($\perp$) is derived.

$$S_0 \Rightarrow S_1 \Rightarrow \dots S_n \Rightarrow \dots$$

Three outcomes:

1. $\perp$ is derived ($\perp \in S_n$ for some n), then S is unsatisfiable (provided $\mathbb{I}$ is sound);
2. no new clauses can be derived from S and $\perp \notin S$, then S is saturated; in this case S is satisfiable, (provided $\mathbb{I}$ is complete).
3. S grows ad infinitum, the process does not terminate.

Goal: speed up the first two cases and reduce non-termination.

Reasoning methods based on inference systems

Basic Idea. A Saturation Process:

Given set of clauses S we exhaustively apply all inference rules adding the conclusions to this set until the contradiction ($\perp$) is derived.

$$S_0 \Rightarrow S_1 \Rightarrow \dots S_n \Rightarrow \dots$$

Three outcomes:

1. $\perp$ is derived ($\perp \in S_n$ for some n), then S is unsatisfiable (provided $\mathbb{I}$ is sound);
2. no new clauses can be derived from S and $\perp \not\in S$, then S is saturated; in this case S is satisfiable, (provided $\mathbb{I}$ is complete).
3. S grows ad infinitum, the process does not terminate.

Goal: speed up the first two cases and reduce non-termination.

Saturation ingredients

- simplification inferences
- inference restrictions
- saturation strategies

Saturation ingredients

- simplification inferences
- inference restrictions
- saturation strategies

Simplification rules

Simplification rules allow to remove clauses in the saturation process without affecting neither soundness nor completeness.

Tautology elimination (TE):

$$S \Rightarrow S \setminus \{C\}$$

where C is a tautology ($\models C$).

- when a clause is a tautology?

Tautology elimination is a simplification rule.

Simplification rules

Simplification rules allow to remove clauses in the saturation process without affecting neither soundness nor completeness.

Tautology elimination (TE):

$$S \Rightarrow S \setminus \{C\}$$

where C is a tautology ($\models C$).

- when a clause is a tautology?

Tautology elimination is a simplification rule.

Simplification rules

Simplification rules allow to remove clauses in the saturation process without affecting neither soundness nor completeness.

Tautology elimination (TE):

$$S \Rightarrow S \setminus \{C\}$$

where C is a tautology ($\models C$).

- when a clause is a tautology?
- $p \vee \neg p \vee C$, why?

Tautology elimination is a simplification rule.

Simplification rules

Simplification rules allow to remove clauses in the saturation process without affecting neither soundness nor completeness.

Tautology elimination (TE):

$$S \Rightarrow S \setminus \{C\}$$

where C is a tautology ($\models C$).

- when a clause is a tautology?
- $p \vee \neg p \vee C$, why?

Tautology elimination is a simplification rule.

Simplification rules

Simplification rules allow to remove clauses in the saturation process without affecting neither soundness nor completeness.

Tautology elimination (TE):

$$S \Rightarrow S \setminus \{C\}$$

where C is a tautology ($\models C$).

- when a clause is a tautology?
- $p \vee \neg p \vee C$, why?

Tautology elimination is a simplification rule.

Simplification rules

Simplification rules allow to remove clauses in the saturation process without affecting neither soundness nor completeness.

Tautology elimination (TE):

$$S \Rightarrow S \setminus \{C\}$$

where C is a tautology ($\models C$).

- when a clause is a tautology?
- $p \vee \neg p \vee C$, why?

Tautology elimination is a simplification rule.

Subsumption Elimination

A clause C subsumes a clause D if $C \subset D$.

Example:

- $p \lor \neg q$ subsumes $p \lor s \lor \neg q \lor d \lor d$,
- $p \lor \neg q$ subsumes $p \lor \neg q \lor \neg q$,
- does $\perp$ subsume $p \lor \neg q \lor \neg q$?
- does $p \lor q \lor s$ subsume $p \lor s$?

Subsumption Elimination (SE):

$$S \Rightarrow S \setminus \{D\}$$

where there is $C \in S$ such that $C \subset D$.

Subsumption elimination is a simplification rule.

Subsumption Elimination

A clause C subsumes a clause D if $C \subset D$.

Example:

- $p \vee \neg q$ subsumes $p \vee s \vee \neg q \vee d \vee d$,
- $p \vee \neg q$ subsumes $p \vee \neg q \vee \neg q$,
- does $\perp$ subsume $p \vee \neg q \vee \neg q$?
- does $p \vee q \vee s$ subsume $p \vee s$?

Subsumption Elimination (SE):

$$S \Rightarrow S \setminus \{D\}$$

where there is $C \in S$ such that $C \subset D$.

Subsumption elimination is a simplification rule.

Subsumption Elimination

A clause C subsumes a clause D if $C \subset D$.

Example:

- $p \vee \neg q$ subsumes $p \vee s \vee \neg q \vee d \vee d$,
- $p \vee \neg q$ subsumes $p \vee \neg q \vee \neg q$,
- does $\perp$ subsume $p \vee \neg q \vee \neg q$?
- does $p \vee q \vee s$ subsume $p \vee s$?

Subsumption Elimination (SE):

$$S \Rightarrow S \setminus \{D\}$$

where there is $C \in S$ such that $C \subset D$.

Subsumption elimination is a simplification rule.

Subsumption Elimination

A clause C subsumes a clause D if $C \subset D$.

Example:

- $p \vee \neg q$ subsumes $p \vee s \vee \neg q \vee d \vee d$,
- $p \vee \neg q$ subsumes $p \vee \neg q \vee \neg q$,
- does $\perp$ subsume $p \vee \neg q \vee \neg q$?
- does $p \vee q \vee s$ subsume $p \vee s$?

Subsumption Elimination (SE):

$$S \Rightarrow S \setminus \{D\}$$

where there is $C \in S$ such that $C \subset D$.

Subsumption elimination is a simplification rule.

Subsumption Elimination

A clause C subsumes a clause D if $C \subset D$.

Example:

- $p \lor \neg q$ subsumes $p \lor s \lor \neg q \lor d \lor d$,
- $p \lor \neg q$ subsumes $p \lor \neg q \lor \neg q$,
- does $\perp$ subsume $p \lor \neg q \lor \neg q$?
- does $p \lor q \lor s$ subsume $p \lor s$?

Subsumption Elimination (SE):

$$S \Rightarrow S \setminus \{D\}$$

where there is $C \in S$ such that $C \subset D$.

Subsumption elimination is a simplification rule.

Subsumption Elimination

A clause C subsumes a clause D if $C \subset D$.

Example:

- $p \lor \neg q$ subsumes $p \lor s \lor \neg q \lor d \lor d$,
- $p \lor \neg q$ subsumes $p \lor \neg q \lor \neg q$,
- does $\perp$ subsume $p \lor \neg q \lor \neg q$?
- does $p \lor q \lor s$ subsume $p \lor s$?

Subsumption Elimination (SE):

$$S \Rightarrow S \setminus \{D\}$$

where there is $C \in S$ such that $C \subset D$.

Subsumption elimination is a simplification rule.

Example

1.	$\neg s \vee p$	input
2.	$q \vee \neg p \vee s$	input
3.	$s \vee p$	input
4.	$p \vee \neg q$	input

Is this set (un)satisfiable, why?

Example

1.	$\neg s \vee p$	input
2.	$q \vee \neg p \vee s$	input
3.	$s \vee p$	input
4.	$p \vee \neg q$	input

Is this set (un)satisfiable, why?

Example

1.	$\neg s \vee p$	input
2.	$q \vee \neg p \vee s$	input
3.	$s \vee p$	input
4.	$p \vee \neg q$	input
5.	$q \vee \neg p \vee p$	BR (1,2)

Is this set (un)satisfiable, why?

Example

1.	$\neg s \vee p$	input
2.	$q \vee \neg p \vee s$	input
3.	$s \vee p$	input
4.	$p \vee \neg q$	input
5.	$\cancel{q \vee \neg p \vee p}$	BR (1,2), TE (5)

Is this set (un)satisfiable, why?

Example

1.	$\neg s \vee p$	input
2.	$q \vee \neg p \vee s$	input
3.	$s \vee p$	input
4.	$p \vee \neg q$	input
5.	$\cancel{q \vee \neg p \vee p}$	BR (1,2), TE (5)

Is this set (un)satisfiable, why?

Example

1.	$\neg s \vee p$	input
2.	$q \vee \neg p \vee s$	input
3.	$s \vee p$	input
4.	$p \vee \neg q$	input
5.	$\cancel{q \vee \neg p \vee p}$	BR (1,2), TE (5)
6.	$p \vee p$	BR (1,3)

Is this set (un)satisfiable, why?

Example

1.	$\neg s \vee p$	input
2.	$q \vee \neg p \vee s$	input
3.	$s \vee p$	input
4.	$p \vee \neg q$	input
5.	$\cancel{q \vee \neg p \vee p}$	BR (1,2), TE (5)
6.	$p \vee p$	BR (1,3)

Is this set (un)satisfiable, why?

Example

1.	$\neg s \vee p$	input
2.	$q \vee \neg p \vee s$	input
3.	$s \vee p$	input
4.	$p \vee \neg q$	input
5.	$\cancel{q \vee \neg p \vee p}$	BR (1,2), TE (5)
6.	$p \vee p$	BR (1,3)
7.	p	BF (6)

Is this set (un)satisfiable, why?

Example

1.	$\neg s \vee p$	input
2.	$q \vee \neg p \vee s$	input
3.	$s \vee p$	input
4.	$p \vee \neg q$	input
5.	$\cancel{q \vee \neg p \vee p}$	BR (1,2), TE (5)
6.	$\cancel{p \vee p}$	BR (1,3), SE (7)
7.	p	BF (6)

Is this set (un)satisfiable, why?

Example

1.	$\neg s \vee p$	input
2.	$q \vee \neg p \vee s$	input
3.	$s \vee p$	input
4.	$p \vee \neg q$	input
5.	$\cancel{q \vee \neg p \vee p}$	BR (1,2), TE (5)
6.	$\cancel{p \vee p}$	BR (1,3), SE (7)
7.	p	BF (6)

Is this set (un)satisfiable, why?

Example

1.	$\neg s \vee p$	input
2.	$q \vee \neg p \vee s$	input
3.	$s \vee p$	input
4.	$\cancel{p \vee \neg q}$	input, SE (7)
5.	$\cancel{q \vee \neg p \vee p}$	BR (1,2), TE (5)
6.	$\cancel{p \vee p}$	BR (1,3), SE (7)
7.	p	BF (6)

Is this set (un)satisfiable, why?

Example

1.	$\neg s \vee p$	input
2.	$q \vee \neg p \vee s$	input
3.	$s \vee p$	input
4.	$\cancel{p \vee \neg q}$	input, SE (7)
5.	$\cancel{q \vee \neg p \vee p}$	BR (1,2), TE (5)
6.	$\cancel{p \vee p}$	BR (1,3), SE (7)
7.	p	BF (6)

Is this set (un)satisfiable, why?

Example

1.	$\neg s \vee p$	input
2.	$q \vee \neg p \vee s$	input
3.	$\cancel{s \vee p}$	input, SE (7)
4.	$\cancel{p \vee \neg q}$	input, SE (7)
5.	$\cancel{q \vee \neg p \vee p}$	BR (1,2), TE (5)
6.	$\cancel{p \vee p}$	BR (1,3), SE (7)
7.	p	BF (6)

Is this set (un)satisfiable, why?

Example

1.	$\neg s \vee p$	input
2.	$q \vee \neg p \vee s$	input
3.	$\cancel{s \vee p}$	input, SE (7)
4.	$\cancel{p \vee \neg q}$	input, SE (7)
5.	$\cancel{q \vee \neg p \vee p}$	BR (1,2), TE (5)
6.	$\cancel{p \vee p}$	BR (1,3), SE (7)
7.	p	BF (6)

Is this set (un)satisfiable, why?

Example

1.	$\cancel{\neg s \lor p}$	input, SE (7)
2.	$q \lor \neg p \lor s$	input
3.	$\cancel{s \lor p}$	input, SE (7)
4.	$\cancel{p \lor \neg q}$	input, SE (7)
5.	$\cancel{q \lor \neg p \lor p}$	BR (1,2), TE (5)
6.	$\cancel{p \lor p}$	BR (1,3), SE (7)
7.	p	BF (6)

Is this set (un)satisfiable, why?

Example

1.	$\cancel{\neg s \vee p}$	input, SE (7)
2.	$q \vee \neg p \vee s$	input
3.	$\cancel{s \vee p}$	input, SE (7)
4.	$\cancel{p \vee \neg q}$	input, SE (7)
5.	$\cancel{q \vee \neg p \vee p}$	BR (1,2), TE (5)
6.	$\cancel{p \vee p}$	BR (1,3), SE (7)
7.	p	BF (6)

Is this set (un)satisfiable, why?

Example

1.	$\cancel{\neg s \vee p}$	input, SE (7)
2.	$q \vee \neg p \vee s$	input
3.	$\cancel{s \vee p}$	input, SE (7)
4.	$\cancel{p \vee \neg q}$	input, SE (7)
5.	$\cancel{q \vee \neg p \vee p}$	BR (1,2), TE (5)
6.	$\cancel{p \vee p}$	BR (1,3), SE (7)
7.	p	BF (6)
8.	$q \vee s$	BR (2,7)

Is this set (un)satisfiable, why?

Example

1.	$\cancel{\neg s \vee p}$	input, SE (7)
2.	$q \vee \neg p \vee s$	input
3.	$\cancel{s \vee p}$	input, SE (7)
4.	$\cancel{p \vee \neg q}$	input, SE (7)
5.	$\cancel{q \vee \neg p \vee p}$	BR (1,2), TE (5)
6.	$\cancel{p \vee p}$	BR (1,3), SE (7)
7.	p	BF (6)
8.	$q \vee s$	BR (2,7)

Is this set (un)satisfiable, why?

Example

1.	$\cancel{\neg s \vee p}$	input, SE (7)
2.	$\cancel{q \vee \neg p \vee s}$	input, SE (8)
3.	$\cancel{s \vee p}$	input, SE (7)
4.	$\cancel{p \vee \neg q}$	input, SE (7)
5.	$\cancel{q \vee \neg p \vee p}$	BR (1,2), TE (5)
6.	$\cancel{p \vee p}$	BR (1,3), SE (7)
7.	p	BF (6)
8.	$q \vee s$	BR (2,7)

Is this set (un)satisfiable, why?

(BF)+(SE) is sufficient for termination of $\mathbb{BR}$

Consider (BF):

$$\frac{C \vee L \vee L}{C \vee L}$$

Note: Using (SE) we can eliminate the premise in the presence of the conclusion.

We say a clause C to is in a set-reduced form if every literal occurs no more than once in C. A clause C in a set-reduced from can be seen as a set of literals (rather than a multi-set).

Remark: if we eagerly apply (BF) and (SE) then we can reduce any clause into the set reduced form.

Theorem

$\mathbb{BR}$ with eager subsumption elimination is a decision procedure for propositional logic.

(BF)+(SE) is sufficient for termination of $\mathbb{BR}$

Consider (BF):

$$\frac{C \vee L \vee L}{C \vee L}$$

Note: Using (SE) we can eliminate the premise in the presence of the conclusion.

We say a clause C to is in a set-reduced form if every literal occurs no more than once in C. A clause C in a set-reduced from can be seen as a set of literals (rather than a multi-set).

Remark: if we eagerly apply (BF) and (SE) then we can reduce any clause into the set reduced form.

Theorem

$\mathbb{BR}$ with eager subsumption elimination is a decision procedure for propositional logic.

(BF)+(SE) is sufficient for termination of $\mathbb{BR}$

Consider (BF):

$$\frac{C \vee L \vee L}{C \vee L}$$

Note: Using (SE) we can eliminate the premise in the presence of the conclusion.

We say a clause C to is in a set-reduced form if every literal occurs no more than once in C. A clause C in a set-reduced from can be seen as a set of literals (rather than a multi-set).

Remark: if we eagerly apply (BF) and (SE) then we can reduce any clause into the set reduced form.

Theorem

$\mathbb{BR}$ with eager subsumption elimination is a decision procedure for propositional logic.

(BF)+(SE) is sufficient for termination of $\mathbb{BR}$

Consider (BF):

$$\frac{C \vee L \vee L}{C \vee L}$$

Note: Using (SE) we can eliminate the premise in the presence of the conclusion.

We say a clause C to is in a set-reduced form if every literal occurs no more than once in C. A clause C in a set-reduced from can be seen as a set of literals (rather than a multi-set).

Remark: if we eagerly apply (BF) and (SE) then we can reduce any clause into the set reduced form.

Theorem

$\mathbb{BR}$ *with eager subsumption elimination is a decision procedure for propositional logic.*

Summary

Inference systems:

- soundness, completeness, proofs
- inference systems for reasoning methods

The resolution inference system ($\mathbb{BR}$)

- soundness, completeness (later)
- simplification rules:
  - tautology elimination (TE) and
  - subsumption elimination (SE)
- decision procedure with eager (SE).

Summary

Inference systems:

- soundness, completeness, proofs
- inference systems for reasoning methods

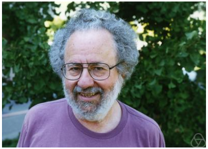
The resolution inference system ($\mathbb{BR}$)

- soundness, completeness (later)
- simplification rules:
  - tautology elimination (TE) and
  - subsumption elimination (SE)
- decision procedure with eager (SE).

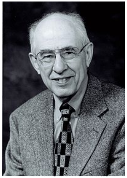
Section DPLL

DPLL Inventors

DPLL: Davis, Putnam, Loveland and Logemann



Martin Davis



Hilary Putnam

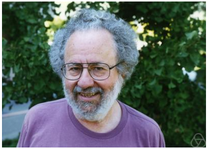
Donald Loveland

George Logemann

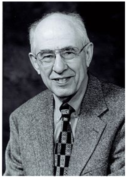
DPLL Algorithm: A reasoning method for propositional logic.

DPLL Inventors

DPLL: Davis, Putnam, Loveland and Logemann


Martin Davis


Donald Loveland


Hilary Putnam


George Logemann

DPLL Algorithm: A reasoning method for propositional logic.

DPLL

DPLL properties:

- Applies to clausal logic, (like resolution),
- Splitting+Unit Propagation

The most efficient RM for propositional logic known up to now:

- space efficient
- backjumping
- lemma learning
- two watch literals

DPLL

DPLL properties:

- Applies to clausal logic, (like resolution),
- Splitting+Unit Propagation

The most efficient RM for propositional logic known up to now:

- space efficient
- backjumping
- lemma learning
- two watch literals

Unit Propagation

Unit Resolution (one step Unit Propagation).

Consider a set of clauses S:

1. ℓ
2. $\ell \vee C$
3. $\bar{\ell} \vee D$
4.

Unit Propagation.

1. $\ell_1, \ldots, \ell_n$
2.
3.

$\ldots$

n+2.

Unit Propagation

Unit Resolution (one step Unit Propagation).

Consider a set of clauses S:

1. ℓ
2. $\ell \vee C$
3. $\bar{\ell} \vee D$
4.

Unit Propagation.

1. $\ell_1, \ldots, \ell_n$
2.
3.

$\ldots$

n+2.

Unit Propagation

Unit Resolution (one step Unit Propagation).

Consider a set of clauses S:

1. ℓ
2. $\cancel{\ell \vee C}$ SE(1)
3. $\bar{\ell} \vee D$
4.

Unit Propagation.

1. $\ell_1, \ldots, \ell_n$
2.
3.

$\ldots$

n+2.

Unit Propagation

Unit Resolution (one step Unit Propagation).

Consider a set of clauses S:

1. ℓ
2. $\cancel{\ell \vee C}$ SE(1)
3. $\bar{\ell} \vee D$
4.

Unit Propagation.

1. $\ell_1, \ldots, \ell_n$
2.
3.

$\ldots$

n+2.

Unit Propagation

Unit Resolution (one step Unit Propagation).

Consider a set of clauses S:

1. ℓ
2. $\cancel{\ell \vee C}$ SE(1)
3. $\bar{\ell} \vee D$
4. D BR(1,3)

Unit Propagation.

1. $\ell_1, \ldots, \ell_n$

2.

3.

$\ldots$

n+2.

Unit Propagation

Unit Resolution (one step Unit Propagation).

Consider a set of clauses S:

1.	ℓ	
2.	$\cancel{\ell \vee C}$	SE(1)
3.	$\cancel{\bar{\ell} \vee D}$	SE(4)
4.	D	BR(1,3)

Unit Propagation.

1.	$\ell_1, \ldots, \ell_n$
2.	
3.	
	$\ldots$
n+2.	

Unit Propagation

Unit Resolution (one step Unit Propagation).

Consider a set of clauses S:

1.	ℓ	
2.	$\cancel{\ell \vee C}$	SE(1)
3.	$\cancel{\bar{\ell} \vee D}$	SE(4)
4.	D	BR(1,3)

1.	ℓ	
2.	$\cancel{\ell \vee C}$	SE(1)
3.	$\cancel{\bar{\ell}} \vee D$	UR(1)

Unit Propagation.

1. $\ell_1, \ldots, \ell_n$

2.

3.

$\ldots$

n+2.

Unit Propagation

Unit Resolution (one step Unit Propagation).

Consider a set of clauses S:

1.	ℓ	
2.	$\cancel{\ell \vee C}$	SE(1)
3.	$\cancel{\bar{\ell} \vee D}$	SE(4)
4.	D	BR(1,3)

1.	ℓ	
2.	$\cancel{\ell \vee C}$	SE(1)
3.	$\cancel{\bar{\ell}} \vee D$	UR(1)

Unit Propagation.

1.	$\ell_1, \ldots, \ell_n$
2.	$\overline{\ell_1} \vee \ldots \vee \overline{\ell_n} \vee \ell$
3.	$\ell_1 \vee D_1$
	$\ldots$
n+2.	$\ell_n \vee D_n$

Unit Propagation

Unit Resolution (one step Unit Propagation).

Consider a set of clauses S:

1.	ℓ	
2.	$\cancel{\ell \vee C}$	SE(1)
3.	$\cancel{\bar{\ell} \vee D}$	SE(4)
4.	D	BR(1,3)

1.	ℓ	
2.	$\cancel{\ell \vee C}$	SE(1)
3.	$\cancel{\bar{\ell}} \vee D$	UR(1)

Unit Propagation.

1.	$\ell_1, \ldots, \ell_n$
2.	$\overline{\ell_1} \vee \ldots \vee \overline{\ell_n} \vee \ell$
3.	$\ell_1 \vee D_1$
	$\ldots$
n+2.	$\ell_n \vee D_n$

Unit Propagation

Unit Resolution (one step Unit Propagation).

Consider a set of clauses S:

1.	ℓ	
2.	$\cancel{\ell \vee C}$	SE(1)
3.	$\cancel{\bar{\ell} \vee D}$	SE(4)
4.	D	BR(1,3)

1.	ℓ	
2.	$\cancel{\ell \vee C}$	SE(1)
3.	$\cancel{\bar{\ell}} \vee D$	UR(1)

Unit Propagation.

1.	$\ell_1, \ldots, \ell_n, \ell$	
2.	$\cancel{\overline{\ell_1} \vee \ldots \vee \overline{\ell_n} \vee} \ \ell$	UP (1,2)
3.	$\ell_1 \vee D_1$	
	$\ldots$	
n+2.	$\ell_n \vee D_n$	

Unit Propagation

Unit Resolution (one step Unit Propagation).

Consider a set of clauses S:

1.	ℓ	
2.	$\cancel{\ell \vee C}$	SE(1)
3.	$\cancel{\bar{\ell} \vee D}$	SE(4)
4.	D	BR(1,3)

1.	ℓ	
2.	$\cancel{\ell \vee C}$	SE(1)
3.	$\cancel{\bar{\ell}} \vee D$	UR(1)

Unit Propagation.

1.	$\ell_1, \ldots, \ell_n, \ell$	
2.	$\cancel{\overline{\ell_1} \vee \ldots \vee \overline{\ell_n} \vee} \ \ell$	UP (1,2)
3.	$\cancel{\ell_1 \vee D_1}$	SE(1)
	$\ldots$	
n+2.	$\cancel{\ell_n \vee D_n}$	SE(1)

Example (UP)

$$\begin{array}{|l|}\hline \parallel \\ \hline p,\ \neg q \\ \neg p \lor q \lor s \\ \neg s \lor \neg p \lor u \\ \neg u \lor q \\ p \lor \neg s \\ s \lor u \\ \hline \end{array} \Rightarrow_{\mathrm{UP}} \begin{array}{|l|}\hline p \parallel \\ \hline \cancel{p},\ \neg q \\ \cancel{\neg p} \lor q \lor s \\ \neg s \lor \cancel{\neg p} \lor u \\ \neg u \lor q \\ \cancel{p \lor \neg s} \\ s \lor u \\ \hline \end{array} \Rightarrow_{\mathrm{UP}} \begin{array}{|l|}\hline p, \neg q \parallel \\ \hline \cancel{p},\ \cancel{\neg q} \\ \cancel{\neg p} \lor \cancel{q} \lor s \\ \neg s \lor \cancel{\neg p} \lor u \\ \neg u \lor \cancel{q} \\ \cancel{p \lor \neg s} \\ s \lor u \\ \hline \end{array} \Rightarrow_{\mathrm{UP}}$$

$$\begin{array}{|l|}\hline p, \neg q, s \parallel \\ \hline \cancel{p},\ \cancel{\neg q} \\ \cancel{\neg p \lor q \lor s} \\ \cancel{\neg s} \lor \cancel{\neg p} \lor u \\ \neg u \lor q \\ \cancel{p \lor \neg s} \\ \cancel{s \lor u} \\ \hline \end{array} \Rightarrow_{\mathrm{UP}} \begin{array}{|l|}\hline p, \neg q, s, u \parallel \\ \hline \cancel{p},\ \cancel{\neg q} \\ \cancel{\neg p \lor q \lor s} \\ \cancel{\neg s} \lor \cancel{\neg p} \lor u \\ \cancel{\neg u} \lor \cancel{q} \\ \cancel{p \lor \neg s} \\ \cancel{s \lor u} \\ \hline \end{array} \Rightarrow_{\perp} \begin{array}{|l|}\hline \perp \parallel \\ \hline p, \neg q \\ \neg p \lor q \lor s \\ \neg s \lor \neg p \lor u \\ \neg u \lor q \\ p \lor \neg s \\ s \lor u \\ \hline \end{array} \quad \text{Unsat}$$

Example (UP)

$\|$
$p, \neg q$
$\neg p \vee q \vee s$
$\neg s \vee \neg p \vee u$
$\neg u \vee q$
$p \vee \neg s$
$s \vee u$

$\Rightarrow_{\mathrm{UP}}$

$p \,\|$
$\cancel{p}, \neg q$
$\cancel{\neg p} \vee q \vee s$
$\neg s \vee \cancel{\neg p} \vee u$
$\neg u \vee q$
$\cancel{p \vee \neg s}$
$s \vee u$

$\Rightarrow_{\mathrm{UP}}$

$p, \neg q \,\|$
$\cancel{p}, \cancel{\neg q}$
$\cancel{\neg p} \vee \cancel{q} \vee s$
$\neg s \vee \cancel{\neg p} \vee u$
$\neg u \vee \cancel{q}$
$\cancel{p \vee \neg s}$
$s \vee u$

$\Rightarrow_{\mathrm{UP}}$

$p, \neg q, s \,\|$
$\cancel{p}, \cancel{\neg q}$
$\cancel{\neg p \vee q \vee s}$
$\cancel{\neg s} \vee \cancel{\neg p} \vee u$
$\neg u \vee q$
$\cancel{p \vee \neg s}$
$\cancel{s \vee u}$

$\Rightarrow_{\mathrm{UP}}$

$p, \neg q, s, u \,\|$
$\cancel{p}, \cancel{\neg q}$
$\cancel{\neg p \vee q \vee s}$
$\cancel{\neg s} \vee \cancel{\neg p} \vee u$
$\cancel{\neg u} \vee \cancel{q}$
$\cancel{p \vee \neg s}$
$\cancel{s \vee u}$

$\Rightarrow_{\perp}$

$\perp \,\|$
$p, \neg q$
$\neg p \vee q \vee s$
$\neg s \vee \neg p \vee u$
$\neg u \vee q$
$p \vee \neg s$
$s \vee u$

Unsat

Example (UP)

$\Vert$
$p, \ \neg q$
$\neg p \vee q \vee s$
$\neg s \vee \neg p \vee u$
$\neg u \vee q$
$p \vee \neg s$
$s \vee u$

$\Rightarrow_{\text{UP}}$

$p \ \Vert$
$\cancel{p}, \ \neg q$
$\cancel{\neg p} \vee q \vee s$
$\neg s \vee \cancel{\neg p} \vee u$
$\neg u \vee q$
$\cancel{p \vee \neg s}$
$s \vee u$

$\Rightarrow_{\text{UP}}$

$p, \neg q \ \Vert$
$\cancel{p}, \cancel{\neg q}$
$\cancel{\neg p} \vee \cancel{q} \vee s$
$\neg s \vee \cancel{\neg p} \vee u$
$\neg u \vee \cancel{q}$
$\cancel{p \vee \neg s}$
$s \vee u$

$\Rightarrow_{\text{UP}}$

$p, \neg q, s \ \Vert$
$\cancel{p}, \cancel{\neg q}$
$\cancel{\neg p \vee q \vee s}$
$\cancel{\neg s} \vee \cancel{\neg p} \vee u$
$\neg u \vee q$
$\cancel{p \vee \neg s}$
$\cancel{s \vee u}$

$\Rightarrow_{\text{UP}}$

$p, \neg q, s, u \ \Vert$
$\cancel{p}, \cancel{\neg q}$
$\cancel{\neg p \vee q \vee s}$
$\cancel{\neg s} \vee \cancel{\neg p} \vee u$
$\cancel{\neg u} \vee \cancel{q}$
$\cancel{p \vee \neg s}$
$\cancel{s \vee u}$

$\Rightarrow_{\perp}$

$\perp \ \Vert$
$p, \neg q$
$\neg p \vee q \vee s$
$\neg s \vee \neg p \vee u$
$\neg u \vee q$
$p \vee \neg s$
$s \vee u$

Unsat

Example (UP)

$\parallel$		$p \parallel$		$p, \neg q \parallel$	
$p, \ \neg q$		$\cancel{p}, \ \neg q$		$\cancel{p}, \cancel{\neg q}$	
$\neg p \lor q \lor s$		$\cancel{\neg p} \lor q \lor s$		$\cancel{\neg p} \lor \cancel{q} \lor s$	
$\neg s \lor \neg p \lor u$	$\Rightarrow_{\mathrm{UP}}$	$\neg s \lor \cancel{\neg p} \lor u$	$\Rightarrow_{\mathrm{UP}}$	$\neg s \lor \cancel{\neg p} \lor u$	$\Rightarrow_{\mathrm{UP}}$
$\neg u \lor q$		$\neg u \lor q$		$\neg u \lor \cancel{q}$	
$p \lor \neg s$		$\cancel{p \lor \neg s}$		$\cancel{p \lor \neg s}$	
$s \lor u$		$s \lor u$		$s \lor u$	

$p, \neg q, s \parallel$		$p, \neg q, s, u \parallel$		$\perp \parallel$	
$\cancel{p}, \cancel{\neg q}$		$\cancel{p}, \cancel{\neg q}$		$p, \neg q$	
$\cancel{\neg p \lor q \lor s}$		$\cancel{\neg p \lor q \lor s}$		$\neg p \lor q \lor s$	
$\cancel{\neg s} \lor \cancel{\neg p} \lor u$	$\Rightarrow_{\mathrm{UP}}$	$\cancel{\neg s} \lor \cancel{\neg p} \lor u$	$\Rightarrow_{\perp}$	$\neg s \lor \neg p \lor u$	Unsat
$\neg u \lor q$		$\cancel{\neg u} \lor \cancel{q}$		$\neg u \lor q$	
$\cancel{p \lor \neg s}$		$\cancel{p \lor \neg s}$		$p \lor \neg s$	
$\cancel{s \lor u}$		$\cancel{s \lor u}$		$s \lor u$	

Example (UP)

$\Vert$		$p \Vert$		$p, \neg q \Vert$	
$p, \ \neg q$		$\cancel{p}, \ \neg q$		$\cancel{p}, \cancel{\neg q}$	
$\neg p \vee q \vee s$		$\cancel{\neg p} \vee q \vee s$		$\cancel{\neg p} \vee \cancel{q} \vee s$	
$\neg s \vee \neg p \vee u$	$\Rightarrow_{\mathrm{UP}}$	$\neg s \vee \cancel{\neg p} \vee u$	$\Rightarrow_{\mathrm{UP}}$	$\neg s \vee \cancel{\neg p} \vee u$	$\Rightarrow_{\mathrm{UP}}$
$\neg u \vee q$		$\neg u \vee q$		$\neg u \vee \cancel{q}$	
$p \vee \neg s$		$\cancel{p \vee \neg s}$		$\cancel{p \vee \neg s}$	
$s \vee u$		$s \vee u$		$s \vee u$	

$p, \neg q, s \Vert$		$p, \neg q, s, u \Vert$		$\bot \Vert$	
$\cancel{p}, \cancel{\neg q}$		$\cancel{p}, \cancel{\neg q}$		$p, \neg q$	
$\cancel{\neg p \vee q \vee s}$		$\cancel{\neg p \vee q \vee s}$		$\neg p \vee q \vee s$	
$\cancel{\neg s} \vee \cancel{\neg p} \vee u$	$\Rightarrow_{\mathrm{UP}}$	$\cancel{\neg s} \vee \cancel{\neg p} \vee u$	$\Rightarrow_{\bot}$	$\neg s \vee \neg p \vee u$	Unsat
$\neg u \vee q$		$\cancel{\neg u} \vee \cancel{q}$		$\neg u \vee q$	
$\cancel{p \vee \neg s}$		$\cancel{p \vee \neg s}$		$p \vee \neg s$	
$\cancel{s \vee u}$		$\cancel{s \vee u}$		$s \vee u$	

Example (UP)

$\|$

$p, \neg q$

$\neg p \lor q \lor s$

$\neg s \lor \neg p \lor u$

$\neg u \lor q$

$p \lor \neg s$

$s \lor u$

$\Rightarrow_{\mathrm{UP}}$

$p \,\|$

$\cancel{p}, \neg q$

$\cancel{\neg p} \lor q \lor s$

$\neg s \lor \cancel{\neg p} \lor u$

$\neg u \lor q$

$\cancel{p \lor \neg s}$

$s \lor u$

$\Rightarrow_{\mathrm{UP}}$

$p, \neg q \,\|$

$\cancel{p}, \cancel{\neg q}$

$\cancel{\neg p} \lor \cancel{q} \lor s$

$\neg s \lor \cancel{\neg p} \lor u$

$\neg u \lor \cancel{q}$

$\cancel{p \lor \neg s}$

$s \lor u$

$\Rightarrow_{\mathrm{UP}}$

$p, \neg q, s \,\|$

$\cancel{p}, \cancel{\neg q}$

$\cancel{\neg p \lor q \lor s}$

$\cancel{\neg s} \lor \cancel{\neg p} \lor u$

$\neg u \lor q$

$\cancel{p \lor \neg s}$

$\cancel{s \lor u}$

$\Rightarrow_{\mathrm{UP}}$

$p, \neg q, s, u \,\|$

$\cancel{p}, \cancel{\neg q}$

$\cancel{\neg p \lor q \lor s}$

$\cancel{\neg s} \lor \cancel{\neg p} \lor u$

$\cancel{\neg u} \lor \cancel{q}$

$\cancel{p \lor \neg s}$

$\cancel{s \lor u}$

$\Rightarrow_{\perp}$

$\perp \,\|$

$p, \neg q$

$\neg p \lor q \lor s$

$\neg s \lor \neg p \lor u$

$\neg u \lor q$

$p \lor \neg s$

$s \lor u$

Unsat

Example (UP)

$\|$
$p, \neg q$
$\neg p \vee q \vee s$
$\neg s \vee \neg p \vee u$
$\neg u \vee q$
$p \vee \neg s$
$s \vee u$

$\Rightarrow_{\text{UP}}$

$p \;\|$
$\cancel{p}, \neg q$
$\cancel{\neg p} \vee q \vee s$
$\neg s \vee \cancel{\neg p} \vee u$
$\neg u \vee q$
$\cancel{p \vee \neg s}$
$s \vee u$

$\Rightarrow_{\text{UP}}$

$p, \neg q \;\|$
$\cancel{p}, \cancel{\neg q}$
$\cancel{\neg p} \vee \cancel{q} \vee s$
$\neg s \vee \cancel{\neg p} \vee u$
$\neg u \vee \cancel{q}$
$\cancel{p \vee \neg s}$
$s \vee u$

$\Rightarrow_{\text{UP}}$

$p, \neg q, s \;\|$
$\cancel{p}, \cancel{\neg q}$
$\cancel{\neg p \vee q \vee s}$
$\cancel{\neg s} \vee \cancel{\neg p} \vee u$
$\neg u \vee q$
$\cancel{p \vee \neg s}$
$\cancel{s \vee u}$

$\Rightarrow_{\text{UP}}$

$p, \neg q, s, u \;\|$
$\cancel{p}, \cancel{\neg q}$
$\cancel{\neg p \vee q \vee s}$
$\cancel{\neg s} \vee \cancel{\neg p} \vee u$
$\cancel{\neg u} \vee \cancel{q}$
$\cancel{p \vee \neg s}$
$\cancel{s \vee u}$

$\Rightarrow_{\perp}$

$\perp \;\|$
$p, \neg q$
$\neg p \vee q \vee s$
$\neg s \vee \neg p \vee u$
$\neg u \vee q$
$p \vee \neg s$
$s \vee u$

Unsat

Example (UP)

$\|$
$p, \neg q$
$\neg p \vee q \vee s$
$\neg s \vee \neg p \vee u$
$\neg u \vee q$
$p \vee \neg s$
$s \vee u$

$\Rightarrow_{\mathrm{UP}}$

$p \;\|$
$\cancel{p}, \neg q$
$\cancel{\neg p} \vee q \vee s$
$\neg s \vee \cancel{\neg p} \vee u$
$\neg u \vee q$
$\cancel{p \vee \neg s}$
$s \vee u$

$\Rightarrow_{\mathrm{UP}}$

$p, \neg q \;\|$
$\cancel{p}, \cancel{\neg q}$
$\cancel{\neg p} \vee \cancel{q} \vee s$
$\neg s \vee \cancel{\neg p} \vee u$
$\neg u \vee \cancel{q}$
$\cancel{p \vee \neg s}$
$s \vee u$

$\Rightarrow_{\mathrm{UP}}$

$p, \neg q, s \;\|$
$\cancel{p}, \cancel{\neg q}$
$\cancel{\neg p \vee q \vee s}$
$\cancel{\neg s} \vee \cancel{\neg p} \vee u$
$\neg u \vee q$
$\cancel{p \vee \neg s}$
$\cancel{s \vee u}$

$\Rightarrow_{\mathrm{UP}}$

$p, \neg q, s, u \;\|$
$\cancel{p}, \cancel{\neg q}$
$\cancel{\neg p \vee q \vee s}$
$\cancel{\neg s} \vee \cancel{\neg p} \vee u$
$\cancel{\neg u} \vee \cancel{q}$
$\cancel{p \vee \neg s}$
$\cancel{s \vee u}$

$\Rightarrow_{\perp}$

$\perp \;\|$
$p, \neg q$
$\neg p \vee q \vee s$
$\neg s \vee \neg p \vee u$
$\neg u \vee q$
$p \vee \neg s$
$s \vee u$

Unsat

Example (UP)

$\|$
$p, \; \neg q$
$\neg p \vee q \vee s$
$\neg s \vee \neg p \vee u$
$\neg u \vee q$
$p \vee \neg s$
$s \vee u$

$\Rightarrow_{\mathrm{UP}}$

$p \; \|$
$\cancel{p}, \; \neg q$
$\cancel{\neg p} \vee q \vee s$
$\neg s \vee \cancel{\neg p} \vee u$
$\neg u \vee q$
$\cancel{p \vee \neg s}$
$s \vee u$

$\Rightarrow_{\mathrm{UP}}$

$p, \neg q \; \|$
$\cancel{p}, \; \cancel{\neg q}$
$\cancel{\neg p} \vee \cancel{q} \vee s$
$\neg s \vee \cancel{\neg p} \vee u$
$\neg u \vee \cancel{q}$
$\cancel{p \vee \neg s}$
$s \vee u$

$\Rightarrow_{\mathrm{UP}}$

$p, \neg q, s \; \|$
$\cancel{p}, \; \cancel{\neg q}$
$\cancel{\neg p \vee q \vee s}$
$\cancel{\neg s} \vee \cancel{\neg p} \vee u$
$\neg u \vee q$
$\cancel{p \vee \neg s}$
$\cancel{s \vee u}$

$\Rightarrow_{\mathrm{UP}}$

$p, \neg q, s, u \; \|$
$\cancel{p}, \; \cancel{\neg q}$
$\cancel{\neg p \vee q \vee s}$
$\cancel{\neg s} \vee \cancel{\neg p} \vee u$
$\cancel{\neg u} \vee \cancel{q}$
$\cancel{p \vee \neg s}$
$\cancel{s \vee u}$

$\Rightarrow_{\perp}$

$\perp \; \|$
$p, \neg q$
$\neg p \vee q \vee s$
$\neg s \vee \neg p \vee u$
$\neg u \vee q$
$p \vee \neg s$
$s \vee u$

Unsat

Horn Clauses

A clause is called Horn if it contains at most one positive literal.

Examples (Horn): $p, \quad \neg q \vee \neg s \vee q, \quad \neg p \vee \neg s.$

Examples (non-Horn): $p \vee q, \quad \neg s \vee \neg q \vee s \vee u.$

Theorem

Unit Propagation is a polynomial-time decision procedure for the fragment of Horn clauses.

Remark. UP is not complete for the fragment of all clauses.

Horn Clauses

A clause is called Horn if it contains at most one positive literal.

Examples (Horn): $p, \quad \neg q \vee \neg s \vee q, \quad \neg p \vee \neg s.$

Examples (non-Horn): $p \vee q, \quad \neg s \vee \neg q \vee s \vee u.$

Theorem

Unit Propagation is a polynomial-time decision procedure for the fragment of Horn clauses.

Remark. UP is not complete for the fragment of all clauses.

Horn Clauses

A clause is called Horn if it contains at most one positive literal.

Examples (Horn): p, $\neg q \vee \neg s \vee q$, $\neg p \vee \neg s$.

Examples (non-Horn): $p \vee q$, $\neg s \vee \neg q \vee s \vee u$.

Theorem

Unit Propagation is a polynomial-time decision procedure for the fragment of Horn clauses.

Remark. UP is not complete for the fragment of all clauses.

Decide

To remedy UP incompleteness we need new rules:

Decide (D) and Backtrack (B).

$$
\begin{array}{|l|}\hline \| \\ \hline p \vee q \\ \neg p \vee q \\ p \vee \neg q \\ \neg p \vee \neg q \\ \hline \end{array}
\Rightarrow_D
\begin{array}{|c|}\hline p^d \; \| \\ \hline \\ \vee q \\ \\ \vee \neg q \\ \hline \end{array}
\Rightarrow_{\mathrm{UP}}
\begin{array}{|c|}\hline p^d \, q \; \| \\ \hline \\ \\ \\ \vee \\ \hline \end{array}
\Rightarrow_B
\begin{array}{|c|}\hline \neg p \; \| \\ \hline \vee q \\ \\ \vee \neg q \\ \\ \hline \end{array}
\Rightarrow_{\mathrm{UP}}
$$

$$
\begin{array}{|c|}\hline \neg p q \; \| \\ \hline \\ \\ \vee \\ \\ \hline \end{array}
\Rightarrow_\bot
\begin{array}{|l|}\hline \bot \; \| \\ \hline p \vee q \\ \neg p \vee q \\ p \vee \neg q \\ \neg p \vee \neg q \\ \hline \end{array}
$$

Unsatisfiable!

Decide

To remedy UP incompleteness we need new rules:

Decide (D) and Backtrack (B).

$$
\begin{array}{|l|}\hline \;\| \\ \hline p \vee q \\ \neg p \vee q \\ p \vee \neg q \\ \neg p \vee \neg q \\ \hline \end{array}
\Rightarrow_D
\begin{array}{|l|}\hline p^d \;\| \\ \hline p \vee q \\ \neg p \vee q \\ p \vee \neg q \\ \neg p \vee \neg q \\ \hline \end{array}
\Rightarrow_{\mathrm{UP}}
\begin{array}{|l|}\hline p^d q \;\| \\ \hline \\ \\ \\ \vee \\ \hline \end{array}
\Rightarrow_B
\begin{array}{|l|}\hline \neg p \;\| \\ \hline \vee q \\ \\ \vee \neg q \\ \\ \hline \end{array}
\Rightarrow_{\mathrm{UP}}
$$

$$
\begin{array}{|l|}\hline \neg p q \;\| \\ \hline \\ \\ \vee \\ \\ \hline \end{array}
\Rightarrow_\bot
\begin{array}{|l|}\hline \bot \;\| \\ \hline p \vee q \\ \neg p \vee q \\ p \vee \neg q \\ \neg p \vee \neg q \\ \hline \end{array}
$$

Unsatisfiable!

Decide

To remedy UP incompleteness we need new rules:

Decide (D) and Backtrack (B).

$\Vert$
$p \vee q$
$\neg p \vee q$
$p \vee \neg q$
$\neg p \vee \neg q$

$\Rightarrow_D$

$p^d \ \Vert$
$\cancel{p \vee q}$
$\cancel{\neg p} \vee q$
$\cancel{p \vee \neg q}$
$\cancel{\neg p} \vee \neg q$

$\Rightarrow_{\mathrm{UP}}$

$p^d q \ \Vert$
$\vee$

$\Rightarrow_B$

$\neg p \ \Vert$
$\vee q$
$\vee \neg q$

$\Rightarrow_{\mathrm{UP}}$

$\neg p q \ \Vert$
$\vee$

$\Rightarrow_{\perp}$

$\perp \ \Vert$
$p \vee q$
$\neg p \vee q$
$p \vee \neg q$
$\neg p \vee \neg q$

Unsatisfiable!

Decide

To remedy UP incompleteness we need new rules:

Decide (D) and Backtrack (B).

$\|$
$p \vee q$
$\neg p \vee q$
$p \vee \neg q$
$\neg p \vee \neg q$

$\Rightarrow_D$

$p^d \;\|$
$\cancel{p \vee q}$
$\cancel{\neg p} \vee q$
$\cancel{p \vee \neg q}$
$\cancel{\neg p} \vee \neg q$

$\Rightarrow_{\mathrm{UP}}$

$p^d q \;\|$
$p \vee q$
$\neg p \vee q$
$p \vee \neg q$
$\neg p \vee \neg q$

$\Rightarrow_B$

$\neg p \;\|$
$\vee q$
$\vee \neg q$

$\Rightarrow_{\mathrm{UP}}$

$\neg p q \;\|$
$\vee$

$\Rightarrow_\bot$

$\bot \;\|$
$p \vee q$
$\neg p \vee q$
$p \vee \neg q$
$\neg p \vee \neg q$

Unsatisfiable!

Decide

To remedy UP incompleteness we need new rules:

Decide (D) and Backtrack (B).

$\|$		$p^d \|$		$p^d q \|$		$\neg p \|$	
$p \lor q$		$\cancel{p \lor q}$		$\cancel{p \lor q}$		$\lor q$	
$\neg p \lor q$	$\Rightarrow_D$	$\cancel{\neg p} \lor q$	$\Rightarrow_{\mathrm{UP}}$	$\cancel{\neg p \lor q}$	$\Rightarrow_B$		$\Rightarrow_{\mathrm{UP}}$
$p \lor \neg q$		$\cancel{p \lor \neg q}$		$\cancel{p \lor \neg q}$		$\lor \neg q$	
$\neg p \lor \neg q$		$\cancel{\neg p} \lor \neg q$		$\cancel{\neg p} \lor \cancel{\neg q}$			

$\neg p q \|$		$\bot \|$	
		$p \lor q$	
	$\Rightarrow_\bot$	$\neg p \lor q$	Unsatisfiable!
$\lor$		$p \lor \neg q$	
		$\neg p \lor \neg q$	

Decide

To remedy UP incompleteness we need new rules:

Decide (D) and Backtrack (B).

$$
\begin{array}{|l|}\hline \; \| \\ \hline p \vee q \\ \neg p \vee q \\ p \vee \neg q \\ \neg p \vee \neg q \\ \hline \end{array}
\Rightarrow_D
\begin{array}{|l|}\hline p^d \; \| \\ \hline \cancel{p \vee q} \\ \cancel{\neg p} \vee q \\ \cancel{p \vee \neg q} \\ \cancel{\neg p} \vee \neg q \\ \hline \end{array}
\Rightarrow_{\mathrm{UP}}
\begin{array}{|l|}\hline p^d \, q \; \| \\ \hline \cancel{p \vee q} \\ \cancel{\neg p \vee q} \\ \cancel{p \vee \neg q} \\ \cancel{\neg p} \vee \cancel{\neg q} \\ \hline \end{array}
\Rightarrow_B
\begin{array}{|l|}\hline \neg p \; \| \\ \hline p \vee q \\ \neg p \vee q \\ p \vee \neg q \\ \neg p \vee \neg q \\ \hline \end{array}
\Rightarrow_{\mathrm{UP}}
$$

$$
\begin{array}{|c|}\hline \neg p q \; \| \\ \hline \\ \\ \vee \\ \\ \hline \end{array}
\Rightarrow_{\perp}
\begin{array}{|l|}\hline \perp \; \| \\ \hline p \vee q \\ \neg p \vee q \\ p \vee \neg q \\ \neg p \vee \neg q \\ \hline \end{array}
$$

Unsatisfiable!

Decide

To remedy UP incompleteness we need new rules:

Decide (D) and Backtrack (B).

$\|$		$p^d \ \|$		$p^d q \ \|$		$\neg p \ \|$	
$p \lor q$		$\cancel{p \lor q}$		$\cancel{p \lor q}$		$\cancel{p} \lor q$	
$\neg p \lor q$	$\Rightarrow_D$	$\cancel{\neg p} \lor q$	$\Rightarrow_{\mathrm{UP}}$	$\cancel{\neg p \lor q}$	$\Rightarrow_B$	$\cancel{\neg p \lor q}$	$\Rightarrow_{\mathrm{UP}}$
$p \lor \neg q$		$\cancel{p \lor \neg q}$		$\cancel{p \lor \neg q}$		$\cancel{p} \lor \neg q$	
$\neg p \lor \neg q$		$\cancel{\neg p} \lor \neg q$		$\cancel{\neg p} \lor \cancel{\neg q}$		$\cancel{\neg p \lor \neg q}$	

$\neg p q \ \|$		$\bot \ \|$	
		$p \lor q$	
	$\Rightarrow_{\bot}$	$\neg p \lor q$	Unsatisfiable!
$\lor$		$p \lor \neg q$	
		$\neg p \lor \neg q$	

Decide

To remedy UP incompleteness we need new rules:

Decide (D) and Backtrack (B).

$$
\begin{array}{|l|}\hline \; \| \\ \hline p \lor q \\ \neg p \lor q \\ p \lor \neg q \\ \neg p \lor \neg q \\ \hline \end{array}
\Rightarrow_D
\begin{array}{|l|}\hline p^d \; \| \\ \hline \cancel{p \lor q} \\ \cancel{\neg p} \lor q \\ \cancel{p \lor \neg q} \\ \cancel{\neg p} \lor \neg q \\ \hline \end{array}
\Rightarrow_{\mathrm{UP}}
\begin{array}{|l|}\hline p^d q \; \| \\ \hline \cancel{p \lor q} \\ \cancel{\neg p \lor q} \\ \cancel{p \lor \neg q} \\ \cancel{\neg p} \lor \cancel{\neg q} \\ \hline \end{array}
\Rightarrow_B
\begin{array}{|l|}\hline \neg p \; \| \\ \hline \cancel{p} \lor q \\ \cancel{\neg p \lor q} \\ \cancel{p} \lor \neg q \\ \cancel{\neg p \lor \neg q} \\ \hline \end{array}
\Rightarrow_{\mathrm{UP}}
$$

$$
\begin{array}{|l|}\hline \neg p q \; \| \\ \hline p \lor q \\ \neg p \lor q \\ p \lor \neg q \\ \neg p \lor \neg q \\ \hline \end{array}
\Rightarrow_{\perp}
\begin{array}{|l|}\hline \perp \; \| \\ \hline p \lor q \\ \neg p \lor q \\ p \lor \neg q \\ \neg p \lor \neg q \\ \hline \end{array}
$$

Unsatisfiable!

Decide

To remedy UP incompleteness we need new rules:

Decide (D) and Backtrack (B).

$$
\begin{array}{|c|}\hline \| \\ \hline p \vee q \\ \neg p \vee q \\ p \vee \neg q \\ \neg p \vee \neg q \\ \hline \end{array}
\Rightarrow_D
\begin{array}{|c|}\hline p^d \; \| \\ \hline \cancel{p \vee q} \\ \cancel{\neg p} \vee q \\ \cancel{p \vee \neg q} \\ \cancel{\neg p} \vee \neg q \\ \hline \end{array}
\Rightarrow_{\mathrm{UP}}
\begin{array}{|c|}\hline p^d q \; \| \\ \hline \cancel{p \vee q} \\ \cancel{\neg p \vee q} \\ \cancel{p \vee \neg q} \\ \cancel{\neg p} \vee \cancel{\neg q} \\ \hline \end{array}
\Rightarrow_B
\begin{array}{|c|}\hline \neg p \; \| \\ \hline \cancel{p} \vee q \\ \cancel{\neg p \vee q} \\ \cancel{p} \vee \neg q \\ \cancel{\neg p \vee \neg q} \\ \hline \end{array}
\Rightarrow_{\mathrm{UP}}
$$

$$
\begin{array}{|c|}\hline \neg p q \; \| \\ \hline \cancel{p \vee q} \\ \cancel{\neg p \vee q} \\ \cancel{p} \vee \cancel{\neg q} \\ \cancel{\neg p \vee \neg q} \\ \hline \end{array}
\Rightarrow_\perp
\begin{array}{|c|}\hline \perp \; \| \\ \hline p \vee q \\ \neg p \vee q \\ p \vee \neg q \\ \neg p \vee \neg q \\ \hline \end{array}
$$

Unsatisfiable!

Decide

To remedy UP incompleteness we need new rules:

Decide (D) and Backtrack (B).

$$
\begin{array}{|l|}\hline \parallel \\ \hline p \lor q \\ \neg p \lor q \\ p \lor \neg q \\ \neg p \lor \neg q \\ \hline \end{array}
\Rightarrow_D
\begin{array}{|l|}\hline p^d \parallel \\ \hline \cancel{p \lor q} \\ \cancel{\neg p} \lor q \\ \cancel{p \lor \neg q} \\ \cancel{\neg p} \lor \neg q \\ \hline \end{array}
\Rightarrow_{\mathrm{UP}}
\begin{array}{|l|}\hline p^d q \parallel \\ \hline \cancel{p \lor q} \\ \cancel{\neg p \lor q} \\ \cancel{p \lor \neg q} \\ \cancel{\neg p} \lor \cancel{\neg q} \\ \hline \end{array}
\Rightarrow_B
\begin{array}{|l|}\hline \neg p \parallel \\ \hline \cancel{p} \lor q \\ \cancel{\neg p \lor q} \\ \cancel{p} \lor \neg q \\ \cancel{\neg p \lor \neg q} \\ \hline \end{array}
\Rightarrow_{\mathrm{UP}}
$$

$$
\begin{array}{|l|}\hline \neg p q \parallel \\ \hline \cancel{p \lor q} \\ \cancel{\neg p \lor q} \\ \cancel{p} \lor \cancel{\neg q} \\ \cancel{\neg p \lor \neg q} \\ \hline \end{array}
\Rightarrow_{\perp}
\begin{array}{|l|}\hline \perp \parallel \\ \hline p \lor q \\ \neg p \lor q \\ p \lor \neg q \\ \neg p \lor \neg q \\ \hline \end{array}
$$

Unsatisfiable!

DPLL state

A DPLL state is a pair $U \parallel S$, where

- U is either $\perp$ or a sequence of literals s.t. if $\ell \in U$ then $\bar{\ell} \notin U$
- S is a set of clauses.

With the sequence of literals U we associate a partial interpretation:

$$I_U = \begin{cases} p \mapsto \mathbf{1} & \text{if } p \in U \\ p \mapsto \mathbf{0} & \text{if } \neg p \in U \end{cases}$$

A literal ℓ is undefined in I_U if neither ℓ nor $\bar{\ell}$ belongs to U.

A clause C is true in I_U if there is $\ell \in C$, ℓ defined in I_U and $I_U \models \ell$.

A DPLL derivation form the state $\parallel S$ is a sequence of the form:

$$\parallel S \Rightarrow U_1 \parallel S \Rightarrow \ldots \Rightarrow U_n \parallel S \ldots$$

where each $\Rightarrow$ is an application of one of the DPLL Rules.

DPLL state

A DPLL state is a pair $U \parallel S$, where

- U is either $\perp$ or a sequence of literals s.t. if $\ell \in U$ then $\overline{\ell} \notin U$
- S is a set of clauses.

With the sequence of literals U we associate a partial interpretation:

$$I_U = \begin{cases} p \mapsto \mathbf{1} & \text{if } p \in U \\ p \mapsto \mathbf{0} & \text{if } \neg p \in U \end{cases}$$

A literal ℓ is undefined in I_U if neither ℓ nor $\overline{\ell}$ belongs to U.

A clause C is true in I_U if there is $\ell \in C$, ℓ defined in I_U and $I_U \models \ell$.

A DPLL derivation form the state $\parallel S$ is a sequence of the form:

$$\parallel S \Rightarrow U_1 \parallel S \Rightarrow \ldots \Rightarrow U_n \parallel S \ldots$$

where each $\Rightarrow$ is an application of one of the DPLL Rules.

DPLL state

A DPLL state is a pair $U \parallel S$, where

- U is either $\perp$ or a sequence of literals s.t. if $\ell \in U$ then $\bar{\ell} \notin U$
- S is a set of clauses.

With the sequence of literals U we associate a partial interpretation:

$$I_U = \begin{cases} p \mapsto \mathbf{1} & \text{if } p \in U \\ p \mapsto \mathbf{0} & \text{if } \neg p \in U \end{cases}$$

A literal ℓ is undefined in I_U if neither ℓ nor $\bar{\ell}$ belongs to U.

A clause C is true in I_U if there is $\ell \in C$, ℓ defined in I_U and $I_U \models \ell$.

A DPLL derivation form the state $\parallel S$ is a sequence of the form:

$$\parallel S \Rightarrow U_1 \parallel S \Rightarrow \ldots \Rightarrow U_n \parallel S \ldots$$

where each $\Rightarrow$ is an application of one of the DPLL Rules.

DPLL state

A DPLL state is a pair $U \parallel S$, where

- U is either $\perp$ or a sequence of literals s.t. if $\ell \in U$ then $\overline{\ell} \notin U$
- S is a set of clauses.

With the sequence of literals U we associate a partial interpretation:

$$I_U = \begin{cases} p \mapsto \mathbf{1} & \text{if } p \in U \\ p \mapsto \mathbf{0} & \text{if } \neg p \in U \end{cases}$$

A literal ℓ is undefined in I_U if neither ℓ nor $\overline{\ell}$ belongs to U.

A clause C is true in I_U if there is $\ell \in C$, ℓ defined in I_U and $I_U \models \ell$.

A DPLL derivation form the state $\parallel S$ is a sequence of the form:

$$\parallel S \Rightarrow U_1 \parallel S \Rightarrow \ldots \Rightarrow U_n \parallel S \ldots$$

where each $\Rightarrow$ is an application of one of the DPLL Rules.

DPLL Rules

Unit Propagate (UP):

$$U \parallel S, \quad \Rightarrow_{\mathrm{UP}} \quad U\ell \parallel S \qquad \text{if } \begin{cases} I_U \models \neg C, \text{ for } C \vee \ell \in S \\ \ell \text{ is undefined in } I_U \end{cases}$$

Decide (D):

$$U \parallel S \quad \Rightarrow_D \quad U\ell^d \parallel S \qquad \text{if } \{\ \ell \text{ is undefined in } I_U$$

Backtrack (B)

$$U\ell^d V \parallel S \quad \Rightarrow_B \quad U\bar{\ell} \parallel S \qquad \text{if } \begin{cases} I_{U\ell^d V} \models \neg C, \text{ for } C \in S, \\ V \text{ contains no decision literals} \end{cases}$$

Unsat ($\perp$)

$$U \parallel S \quad \Rightarrow_\perp \quad \perp \parallel S \qquad \text{if } \begin{cases} I_U \models \neg C, \text{ for } C \in S, \\ U \text{ contains no decision literals} \end{cases}$$

Eager unit propagation:

Decide rule is applied only if no other rule is applicable.

DPLL Rules

Unit Propagate (UP):

$$U \parallel S, \quad \Rightarrow_{\mathrm{UP}} \quad U\ell \parallel S \qquad \text{if } \begin{cases} I_U \models \neg C, \text{ for } C \vee \ell \in S \\ \ell \text{ is undefined in } I_U \end{cases}$$

Decide (D):

$$U \parallel S \quad \Rightarrow_{D} \quad U\ell^d \parallel S \qquad \text{if } \left\{ \ell \text{ is undefined in } I_U \right.$$

Backtrack (B)

$$U\ell^d V \parallel S \quad \Rightarrow_{B} \quad U\bar{\ell} \parallel S \qquad \text{if } \begin{cases} I_{U\ell^d V} \models \neg C, \text{ for } C \in S, \\ V \text{ contains no decision literals} \end{cases}$$

Unsat ($\perp$)

$$U \parallel S \quad \Rightarrow_{\perp} \quad \perp \parallel S \qquad \text{if } \begin{cases} I_U \models \neg C, \text{ for } C \in S, \\ U \text{ contains no decision literals} \end{cases}$$

Eager unit propagation:

Decide rule is applied only if no other rule is applicable.

DPLL Rules

Unit Propagate (UP):

$$U \parallel S, \quad \Rightarrow_{\mathrm{UP}} \quad U\ell \parallel S \qquad \text{if } \begin{cases} I_U \models \neg C, \text{ for } C \vee \ell \in S \\ \ell \text{ is undefined in } I_U \end{cases}$$

Decide (D):

$$U \parallel S \quad \Rightarrow_D \quad U\ell^d \parallel S \qquad \text{if } \left\{ \ell \text{ is undefined in } I_U \right.$$

Backtrack (B)

$$U\ell^d V \parallel S \quad \Rightarrow_B \quad U\bar{\ell} \parallel S \qquad \text{if } \begin{cases} I_{U\ell^d V} \models \neg C, \text{ for } C \in S, \\ V \text{ contains no decision literals} \end{cases}$$

Unsat ($\perp$)

$$U \parallel S \quad \Rightarrow_\perp \quad \perp \parallel S \qquad \text{if } \begin{cases} I_U \models \neg C, \text{ for } C \in S, \\ U \text{ contains no decision literals} \end{cases}$$

Eager unit propagation:

Decide rule is applied only if no other rule is applicable.

DPLL Rules

Unit Propagate (UP):

$$U \parallel S, \quad \Rightarrow_{\mathrm{UP}} \quad U\ell \parallel S \qquad \text{if } \begin{cases} I_U \models \neg C, \text{ for } C \vee \ell \in S \\ \ell \text{ is undefined in } I_U \end{cases}$$

Decide (D):

$$U \parallel S \quad \Rightarrow_D \quad U\ell^d \parallel S \qquad \text{if } \begin{cases} \ell \text{ is undefined in } I_U \end{cases}$$

Backtrack (B)

$$U\ell^d V \parallel S \quad \Rightarrow_B \quad U\overline{\ell} \parallel S \qquad \text{if } \begin{cases} I_{U\ell^d V} \models \neg C, \text{ for } C \in S, \\ V \text{ contains no decision literals} \end{cases}$$

Unsat ($\perp$)

$$U \parallel S \quad \Rightarrow_\perp \quad \perp \parallel S \qquad \text{if } \begin{cases} I_U \models \neg C, \text{ for } C \in S, \\ U \text{ contains no decision literals} \end{cases}$$

Eager unit propagation:

Decide rule is applied only if no other rule is applicable.

DPLL Rules

Unit Propagate (UP):

$$U \parallel S, \quad \Rightarrow_{\mathrm{UP}} \quad U\ell \parallel S \qquad \text{if } \begin{cases} I_U \models \neg C, \text{ for } C \vee \ell \in S \\ \ell \text{ is undefined in } I_U \end{cases}$$

Decide (D):

$$U \parallel S \quad \Rightarrow_D \quad U\ell^d \parallel S \qquad \text{if } \{\ \ell \text{ is undefined in } I_U$$

Backtrack (B)

$$U\ell^d V \parallel S \quad \Rightarrow_B \quad U\bar{\ell} \parallel S \qquad \text{if } \begin{cases} I_{U\ell^d V} \models \neg C, \text{ for } C \in S, \\ V \text{ contains no decision literals} \end{cases}$$

Unsat ($\perp$)

$$U \parallel S \quad \Rightarrow_\perp \quad \perp \parallel S \qquad \text{if } \begin{cases} I_U \models \neg C, \text{ for } C \in S, \\ U \text{ contains no decision literals} \end{cases}$$

Eager unit propagation:

Decide rule is applied only if no other rule is applicable.

DPLL Rules

Unit Propagate (UP):

$$U \parallel S, \quad \Rightarrow_{\mathrm{UP}} \quad U\ell \parallel S \qquad \text{if } \begin{cases} I_U \models \neg C, \text{ for } C \vee \ell \in S \\ \ell \text{ is undefined in } I_U \end{cases}$$

Decide (D):

$$U \parallel S \quad \Rightarrow_D \quad U\ell^d \parallel S \qquad \text{if } \left\{ \ell \text{ is undefined in } I_U \right.$$

Backtrack (B)

$$U\ell^d V \parallel S \quad \Rightarrow_B \quad U\bar{\ell} \parallel S \qquad \text{if } \begin{cases} I_{U\ell^d V} \models \neg C, \text{ for } C \in S, \\ V \text{ contains no decision literals} \end{cases}$$

Unsat ($\perp$)

$$U \parallel S \quad \Rightarrow_\perp \quad \perp \parallel S \qquad \text{if } \begin{cases} I_U \models \neg C, \text{ for } C \in S, \\ U \text{ contains no decision literals} \end{cases}$$

Eager unit propagation:

Decide rule is applied only if no other rule is applicable.

DPLL Decision Procedure

DPLL final state of a derivation $\| \, S \Rightarrow \ldots \Rightarrow U_n \, \| \, S$

- $U_n = \bot$ then S is unsatisfiable, otherwise
- $I_{U_n} \models S$ and S is satisfiable
- any DPLL derivation terminates

Theorem

DPLL is a decision procedure for propositional clausal logic.

Reference: R. Nieuwenhuis, A. Oliveras and C. Tinelli
Solving SAT and SAT Modulo Theories: From an Abstract
Davis-Putnam-Logemann-Loveland Procedure to DPLL(T).
Journal of the ACM, Vol.53 Nov. 2006, pp. 937-977.

DPLL space efficient: requires linear space.

DPLL Decision Procedure

DPLL final state of a derivation $\| S \Rightarrow \ldots \Rightarrow U_n \| S$

- $U_n = \bot$ then S is unsatisfiable, otherwise
- $I_{U_n} \models S$ and S is satisfiable
- any DPLL derivation terminates

Theorem

DPLL is a decision procedure for propositional clausal logic.

Reference: R. Nieuwenhuis, A. Oliveras and C. Tinelli
Solving SAT and SAT Modulo Theories: From an Abstract
Davis-Putnam-Logemann-Loveland Procedure to DPLL(T).
Journal of the ACM, Vol.53 Nov. 2006, pp. 937-977.

DPLL space efficient: requires linear space.

DPLL Decision Procedure

DPLL final state of a derivation $\| S \Rightarrow \ldots \Rightarrow U_n \| S$

- $U_n = \perp$ then S is unsatisfiable, otherwise
- $I_{U_n} \models S$ and S is satisfiable
- any DPLL derivation terminates

Theorem

DPLL is a decision procedure for propositional clausal logic.

Reference: R. Nieuwenhuis, A. Oliveras and C. Tinelli
Solving SAT and SAT Modulo Theories: From an Abstract
Davis-Putnam-Logemann-Loveland Procedure to DPLL(T).
Journal of the ACM, Vol.53 Nov. 2006, pp. 937-977.

DPLL space efficient: requires linear space.

DPLL Example

$$\begin{array}{|l|}
\hline
\;\| \\
\hline
p \lor q \lor \neg r \\
\neg p \lor q \\
p \lor \neg q \lor r \\
\neg p \lor \neg q \\
\hline
\end{array}$$

DPLL Example

$$
\begin{array}{|l|}
\hline
\parallel \\
\hline
p \lor q \lor \neg r \\
\neg p \lor q \\
p \lor \neg q \lor r \\
\neg p \lor \neg q \\
\hline
\end{array}
\Rightarrow_D
\begin{array}{|l|}
\hline
{\color{red} p^d} \parallel \\
\hline
p \lor q \lor \neg r \\
\neg p \lor q \\
p \lor \neg q \lor r \\
\neg p \lor \neg q \\
\hline
\end{array}
\Rightarrow_{\mathrm{UP}}
\begin{array}{|l|}
\hline
p^d q \parallel \\
\hline
\\
\\
\\
\lor \\
\hline
\end{array}
\Rightarrow_B
$$

$$
\begin{array}{|l|}
\hline
\neg p \parallel \\
\hline
\lor q \lor \neg r \\
\\
\lor \neg q \lor r \\
\\
\hline
\end{array}
\Rightarrow_D
\begin{array}{|l|}
\hline
\neg p q^d \parallel \\
\hline
\\
\\
\lor \quad \lor r \\
\\
\hline
\end{array}
\Rightarrow_{\mathrm{UP}}
\begin{array}{|l|}
\hline
\neg p q^d r \parallel \\
\hline
\cancel{p \lor q \lor \neg r} \\
\cancel{\neg p \lor q} \\
\cancel{p \lor \neg q \lor r} \\
\cancel{\neg p \lor \neg q} \\
\hline
\end{array}
$$

Satisfiable!

Model $I = \{p \mapsto 0, q \mapsto 1, r \mapsto 1\}$

DPLL Example

$\Vert$		$p^d \Vert$
$p \vee q \vee \neg r$		$\cancel{p \vee q \vee \neg r}$
$\neg p \vee q$	$\Rightarrow_D$	$\cancel{\neg p} \vee q$
$p \vee \neg q \vee r$		$\cancel{p \vee \neg q \vee r}$
$\neg p \vee \neg q$		$\cancel{\neg p} \vee \neg q$

DPLL Example

$\parallel$
$p \vee q \vee \neg r$
$\neg p \vee q$
$p \vee \neg q \vee r$
$\neg p \vee \neg q$

$\Rightarrow_D$

$p^d \parallel$
$\cancel{p \vee q \vee \neg r}$
$\cancel{\neg p} \vee q$
$\cancel{p \vee \neg q \vee r}$
$\cancel{\neg p} \vee \neg q$

$\Rightarrow_{\mathrm{UP}}$

$p^d q \parallel$
$p \vee q \vee \neg r$
$\neg p \vee q$
$p \vee \neg q \vee r$
$\neg p \vee \neg q$

$\Rightarrow_B$

$\neg p \parallel$
$\vee q \vee \neg r$
$\vee \neg q \vee r$

$\Rightarrow_D$

$\neg p q^d \parallel$
$\vee \quad \vee r$

$\Rightarrow_{\mathrm{UP}}$

$\neg p q^d r \parallel$
$\cancel{p \vee q \vee \neg r}$
$\cancel{\neg p \vee q}$
$\cancel{p \vee \neg q \vee r}$
$\cancel{\neg p \vee \neg q}$

Satisfiable!

Model $I = \{p \mapsto 0, q \mapsto 1, r \mapsto 1\}$

DPLL Example

$\parallel$		$p^d \parallel$		$p^d\, q \parallel$	
$p \lor q \lor \neg r$		$\cancel{p \lor q \lor \neg r}$		$\cancel{p \lor q \lor \neg r}$	
$\neg p \lor q$	$\Rightarrow_D$	$\cancel{\neg p} \lor q$	$\Rightarrow_{\mathrm{UP}}$	$\cancel{\neg p \lor q}$	$\Rightarrow_B$
$p \lor \neg q \lor r$		$\cancel{p \lor \neg q \lor r}$		$\cancel{p \lor \neg q \lor r}$	
$\neg p \lor \neg q$		$\cancel{\neg p} \lor \neg q$		$\cancel{\neg p} \lor \cancel{\neg q}$	

$\neg p \parallel$		$\neg p q^d \parallel$		$\neg p q^d r \parallel$
$\lor q \lor \neg r$				$\cancel{p \lor q \lor \neg r}$
	$\Rightarrow_D$		$\Rightarrow_{\mathrm{UP}}$	$\cancel{\neg p \lor q}$
$\lor \neg q \lor r$		$\lor \quad \lor r$		$\cancel{p \lor \neg q \lor r}$
				$\cancel{\neg p \lor \neg q}$

Satisfiable!

Model $I = \{p \mapsto 0; q \mapsto 1; r \mapsto 1\}$

DPLL Example

$\Vert$
$p \lor q \lor \neg r$
$\neg p \lor q$
$p \lor \neg q \lor r$
$\neg p \lor \neg q$

$\Rightarrow_D$

$p^d \; \Vert$
$\cancel{p \lor q \lor \neg r}$
$\cancel{\neg p} \lor q$
$\cancel{p \lor \neg q \lor r}$
$\cancel{\neg p} \lor \neg q$

$\Rightarrow_{\text{UP}}$

$p^d q \; \Vert$
$\cancel{p \lor q \lor \neg r}$
$\cancel{\neg p \lor q}$
$\cancel{p \lor \neg q \lor r}$
$\cancel{\neg p} \lor \cancel{\neg q}$

$\Rightarrow_B$

$\neg p \; \Vert$
$p \lor q \lor \neg r$
$\neg p \lor q$
$p \lor \neg q \lor r$
$\neg p \lor \neg q$

$\Rightarrow_D$

$\neg p q^d \; \Vert$
$\lor \quad \lor r$

$\Rightarrow_{\text{UP}}$

$\neg p q^d r \; \Vert$
$\cancel{p \lor q \lor \neg r}$
$\cancel{\neg p \lor q}$
$\cancel{p \lor \neg q \lor r}$
$\cancel{\neg p \lor \neg q}$

Satisfiable!

Model $I = \{p \mapsto 0, q \mapsto 1, r \mapsto 1\}$

DPLL Example

$\Vert$
$p \vee q \vee \neg r$
$\neg p \vee q$
$p \vee \neg q \vee r$
$\neg p \vee \neg q$

$\Rightarrow_D$

$p^d \Vert$
$\cancel{p \vee q \vee \neg r}$
$\cancel{\neg p} \vee q$
$\cancel{p \vee \neg q \vee r}$
$\cancel{\neg p} \vee \neg q$

$\Rightarrow_{\mathrm{UP}}$

$p^d q \Vert$
$\cancel{p \vee q \vee \neg r}$
$\cancel{\neg p \vee q}$
$\cancel{p \vee \neg q \vee r}$
$\cancel{\neg p} \vee \cancel{\neg q}$

$\Rightarrow_B$

$\neg p \Vert$
$\cancel{p} \vee q \vee \neg r$
$\cancel{\neg p \vee q}$
$\cancel{p} \vee \neg q \vee r$
$\cancel{\neg p \vee \neg q}$

$\Rightarrow_D$

$\neg p q^d \Vert$
$\vee \quad \vee r$

$\Rightarrow_{\mathrm{UP}}$

$\neg p q^d r \Vert$
$\cancel{p \vee q \vee \neg r}$
$\cancel{\neg p \vee q}$
$\cancel{p \vee \neg q \vee r}$
$\cancel{\neg p \vee \neg q}$

Satisfiable!

Model $I = \{p \mapsto 0, q \mapsto 1, r \mapsto 1\}$

DPLL Example

$\|$
$p \lor q \lor \neg r$
$\neg p \lor q$
$p \lor \neg q \lor r$
$\neg p \lor \neg q$

$\Rightarrow_D$

$p^d \;\|$
$\cancel{p \lor q \lor \neg r}$
$\cancel{\neg p} \lor q$
$\cancel{p \lor \neg q \lor r}$
$\cancel{\neg p} \lor \neg q$

$\Rightarrow_{\mathrm{UP}}$

$p^d q \;\|$
$\cancel{p \lor q \lor \neg r}$
$\cancel{\neg p \lor q}$
$\cancel{p \lor \neg q \lor r}$
$\cancel{\neg p} \lor \cancel{\neg q}$

$\Rightarrow_B$

$\neg p \;\|$
$\cancel{p} \lor q \lor \neg r$
$\cancel{\neg p \lor q}$
$\cancel{p} \lor \neg q \lor r$
$\cancel{\neg p \lor \neg q}$

$\Rightarrow_D$

$\neg p q^d \;\|$
$p \lor q \lor \neg r$
$\neg p \lor q$
$p \lor \neg q \lor r$
$\neg p \lor \neg q$

$\Rightarrow_{\mathrm{UP}}$

$\neg p q^d r \;\|$
$\cancel{p \lor q \lor \neg r}$
$\cancel{\neg p \lor q}$
$\cancel{p \lor \neg q \lor r}$
$\cancel{\neg p \lor \neg q}$

Satisfiable!

Model $I = \{p \mapsto 0; q \mapsto 1; r \mapsto 1\}$

DPLL Example

$\|$
$p \vee q \vee \neg r$
$\neg p \vee q$
$p \vee \neg q \vee r$
$\neg p \vee \neg q$

$\Rightarrow_D$

$p^d \;\|$
$\cancel{p \vee q \vee \neg r}$
$\cancel{\neg p} \vee q$
$\cancel{p \vee \neg q \vee r}$
$\cancel{\neg p} \vee \neg q$

$\Rightarrow_{\mathrm{UP}}$

$p^d q \;\|$
$\cancel{p \vee q \vee \neg r}$
$\cancel{\neg p \vee q}$
$\cancel{p \vee \neg q \vee r}$
$\cancel{\neg p} \vee \cancel{\neg q}$

$\Rightarrow_B$

$\neg p \;\|$
$\cancel{p} \vee q \vee \neg r$
$\cancel{\neg p \vee q}$
$\cancel{p} \vee \neg q \vee r$
$\cancel{\neg p \vee \neg q}$

$\Rightarrow_D$

$\neg p q^d \;\|$
$\cancel{p \vee q \vee \neg r}$
$\cancel{\neg p \vee q}$
$\cancel{p} \vee \cancel{\neg q} \vee r$
$\cancel{\neg p \vee \neg q}$

$\Rightarrow_{\mathrm{UP}}$

$\neg p q^d r \;\|$
$\cancel{p \vee q \vee \neg r}$
$\cancel{\neg p \vee q}$
$\cancel{p \vee \neg q \vee r}$
$\cancel{\neg p \vee \neg q}$

Satisfiable!

Model $I = \{p \mapsto 0, q \mapsto 1, r \mapsto 1\}$

DPLL Example

$\|$
$p \lor q \lor \neg r$
$\neg p \lor q$
$p \lor \neg q \lor r$
$\neg p \lor \neg q$

$\Rightarrow_D$

$p^d \;\|$
$\cancel{p \lor q \lor \neg r}$
$\cancel{\neg p} \lor q$
$\cancel{p \lor \neg q \lor r}$
$\cancel{\neg p} \lor \neg q$

$\Rightarrow_{\text{UP}}$

$p^d q \;\|$
$\cancel{p \lor q \lor \neg r}$
$\cancel{\neg p \lor q}$
$\cancel{p \lor \neg q \lor r}$
$\cancel{\neg p} \lor \cancel{\neg q}$

$\Rightarrow_B$

$\neg p \;\|$
$\cancel{p} \lor q \lor \neg r$
$\cancel{\neg p \lor q}$
$\cancel{p} \lor \neg q \lor r$
$\cancel{\neg p \lor \neg q}$

$\Rightarrow_D$

$\neg p q^d \;\|$
$\cancel{p \lor q \lor \neg r}$
$\cancel{\neg p \lor q}$
$\cancel{p} \lor \cancel{\neg q} \lor r$
$\cancel{\neg p \lor \neg q}$

$\Rightarrow_{\text{UP}}$

$\neg p q^d r \;\|$
$\cancel{p \lor q \lor \neg r}$
$\cancel{\neg p \lor q}$
$\cancel{p \lor \neg q \lor r}$
$\cancel{\neg p \lor \neg q}$

Satisfiable!

Model $I = \{p \mapsto 0; q \mapsto 1; r \mapsto 1\}$

DPLL Example

$\|$		$p^d \;\|$		$p^d q \;\|$	
$p \lor q \lor \neg r$		$\cancel{p \lor q \lor \neg r}$		$\cancel{p \lor q \lor \neg r}$	
$\neg p \lor q$	$\Rightarrow_D$	$\cancel{\neg p} \lor q$	$\Rightarrow_{\text{UP}}$	$\cancel{\neg p \lor q}$	$\Rightarrow_B$
$p \lor \neg q \lor r$		$\cancel{p \lor \neg q \lor r}$		$\cancel{p \lor \neg q \lor r}$	
$\neg p \lor \neg q$		$\cancel{\neg p} \lor \neg q$		$\cancel{\neg p} \lor \cancel{\neg q}$	

$\neg p \;\|$		$\neg p q^d \;\|$		$\neg p q^d r \;\|$
$\cancel{p} \lor q \lor \neg r$		$\cancel{p \lor q \lor \neg r}$		$\cancel{p \lor q \lor \neg r}$
$\cancel{\neg p \lor q}$	$\Rightarrow_D$	$\cancel{\neg p \lor q}$	$\Rightarrow_{\text{UP}}$	$\cancel{\neg p \lor q}$
$\cancel{p} \lor \neg q \lor r$		$\cancel{p} \lor \cancel{\neg q} \lor r$		$\cancel{p \lor \neg q \lor r}$
$\cancel{\neg p \lor \neg q}$		$\cancel{\neg p \lor \neg q}$		$\cancel{\neg p \lor \neg q}$

Satisfiable!

Model $I = \{p \mapsto 0; q \mapsto 1; r \mapsto 1\}$

DPLL Termination

Let n be the number of propositional variables in our problem.

Consider a set $\mathrm{LitType} = \{u, d, i\}$ consisting of three elements.

Informally: u corresponds to an undefined variable, d to a decision variable and i to an implied variable.

Associate with any DPLL state $\ell_1 \dots \ell_m \parallel S$ an n-tuple $(t_1, \dots, t_n)$ of elements from $\mathrm{LitType}$ such that

- for $1 \leq k \leq m$
  - if ℓ_k is a decision variable then $t_k = d$
  - if ℓ_k is an implied variable then $t_k = i$
- for $m + 1 \leq k \leq n$, $t_k = u$.

Define an ordering $u \succ d \succ i$. Note: $\succ$ is obviously well-founded.

By Theorem on the lexicographic combination, $\succ^n_{\mathrm{lex}}$ is also well-founded.

It is straightforward to check that all DPLL rules are compatible with $\succ^n_{\mathrm{lex}}$.

Therefore any DPLL derivation terminates!

DPLL Backjumping

Expensive branching occurs upon decision literals.

Reducing the number of decision literals is crucial for efficiency.

$u[\ell_1 \dots \ell_n]$ – u is implied by $\ell_1, \dots, \ell_n$, i.e. $S \wedge \ell_1 \wedge \dots \wedge \ell_n \models u$

$\|$
$\neg t \vee \neg q \vee v$
$\neg t \vee \neg v$
$\neg p \vee q$
$s \vee u$
$t \vee \neg s$

$\neg t \vee \neg v$ is a conflict clause.

Analyse which decisions imply the conflict.

- t^d is already a decision literal
- $v[t^d q]$, we have $q[p^d]$ therefore $v[t^d p^d]$

1) $S \wedge t \wedge v \models \bot$ 2) $S \wedge t \wedge t \wedge q \models \bot$

3) $S \wedge p \wedge t \models \bot$

$S \wedge p \models \neg t$ hence $\neg t$ is implied by p!

Backjump to p^d and assign $\neg t[p^d]$

Backjump over several decision levels!

DPLL Backjumping

Expensive branching occurs upon decision literals.

Reducing the number of decision literals is crucial for efficiency.

$u[\ell_1 \dots \ell_n]$ – u is implied by $\ell_1, \dots, \ell_n$, i.e. $S \wedge \ell_1 \wedge \dots \wedge \ell_n \models u$

$p^d \parallel$
$\neg t \vee \neg q \vee v$
$\neg t \vee \neg v$
$\neg p \vee q$
$s \vee u$
$t \vee \neg s$

$\neg t \vee \neg v$ is a conflict clause.

Analyse which decisions imply the conflict.

- t^d is already a decision literal
- $v[t^d q]$, we have $q[p^d]$ therefore $v[t^d p^d]$

1) $S \wedge t \wedge v \models \bot$ 2) $S \wedge t \wedge t \wedge q \models \bot$

3) $S \wedge p \wedge t \models \bot$

$S \wedge p \models \neg t$ hence $\neg t$ is implied by p!

Backjump to p^d and assign $\neg t[p^d]$

Backjump over several decision levels!

DPLL Backjumping

Expensive branching occurs upon decision literals.

Reducing the number of decision literals is crucial for efficiency.

$u[\ell_1 \ldots \ell_n]$ – u is implied by $\ell_1, \ldots, \ell_n$, i.e. $S \wedge \ell_1 \wedge \ldots \wedge \ell_n \models u$

$p^d \, q[p^d] \parallel$
$\neg t \vee \neg q \vee v$
$\neg t \vee \neg v$
$\neg p \vee q$
$s \vee u$
$t \vee \neg s$

$\neg t \vee \neg v$ is a conflict clause.

Analyse which decisions imply the conflict.

- t^d is already a decision literal
- $v[t^d q]$, we have $q[p^d]$ therefore $v[t^d p^d]$

1) $S \wedge t \wedge v \models \bot$ 2) $S \wedge t \wedge t \wedge q \models \bot$

3) $S \wedge p \wedge t \models \bot$

$S \wedge p \models \neg t$ hence $\neg t$ is implied by p!

Backjump to p^d and assign $\neg t[p^d]$

Backjump over several decision levels!

DPLL Backjumping

Expensive branching occurs upon decision literals.

Reducing the number of decision literals is crucial for efficiency.

$u[\ell_1 \dots \ell_n]$ – u is implied by $\ell_1, \dots, \ell_n$, i.e. $S \wedge \ell_1 \wedge \dots \wedge \ell_n \models u$

$p^d q[p^d] \neg s^d \parallel$
$\neg t \vee \neg q \vee v$
$\neg t \vee \neg v$
$\neg p \vee q$
$s \vee u$
$t \vee \neg s$

$\neg t \vee \neg v$ is a conflict clause.

Analyse which decisions imply the conflict.

- t^d is already a decision literal
- $v[t^d q]$, we have $q[p^d]$ therefore $v[t^d p^d]$

1) $S \wedge t \wedge v \models \bot$ 2) $S \wedge t \wedge t \wedge q \models \bot$

3) $S \wedge p \wedge t \models \bot$

$S \wedge p \models \neg t$ hence $\neg t$ is implied by p!

Backjump to p^d and assign $\neg t[p^d]$

Backjump over several decision levels!

DPLL Backjumping

Expensive branching occurs upon decision literals.

Reducing the number of decision literals is crucial for efficiency.

$u[\ell_1 \dots \ell_n]$ – u is implied by $\ell_1, \dots, \ell_n$, i.e. $S \wedge \ell_1 \wedge \dots \wedge \ell_n \models u$

$p^d q[p^d] \neg s^d \parallel$
$\neg t \vee \neg q \vee v$
$\neg t \vee \neg v$
$\neg p \vee q$
$s \vee u$
$t \vee \neg s$

$\neg t \vee \neg v$ is a conflict clause.

Analyse which decisions imply the conflict.

- t^d is already a decision literal
- $v[t^d q]$, we have $q[p^d]$ therefore $v[t^d p^d]$

1) $S \wedge t \wedge v \models \bot$ 2) $S \wedge t \wedge t \wedge q \models \bot$

3) $S \wedge p \wedge t \models \bot$

$S \wedge p \models \neg t$ hence $\neg t$ is implied by p!

Backjump to p^d and assign $\neg t[p^d]$

Backjump over several decision levels!

DPLL Backjumping

Expensive branching occurs upon decision literals.

Reducing the number of decision literals is crucial for efficiency.

$u[\ell_1 \ldots \ell_n]$ – u is implied by $\ell_1, \ldots, \ell_n$, i.e. $S \wedge \ell_1 \wedge \ldots \wedge \ell_n \models u$

$p^d q[p^d] \neg s^d u[\neg s^d] \parallel$
$\neg t \vee \neg q \vee v$
$\neg t \vee \neg v$
$\neg p \vee q$
$s \vee u$
$t \vee \neg s$

$\neg t \vee \neg v$ is a conflict clause.

Analyse which decisions imply the conflict.

- t^d is already a decision literal
- $v[t^d q]$, we have $q[p^d]$ therefore $v[t^d p^d]$

1) $S \wedge t \wedge v \models \bot$ 2) $S \wedge t \wedge t \wedge q \models \bot$

3) $S \wedge p \wedge t \models \bot$

$S \wedge p \models \neg t$ hence $\neg t$ is implied by p!

Backjump to p^d and assign $\neg t[p^d]$

Backjump over several decision levels!

DPLL Backjumping

Expensive branching occurs upon decision literals.

Reducing the number of decision literals is crucial for efficiency.

$u[\ell_1 \ldots \ell_n]$ – u is implied by $\ell_1, \ldots, \ell_n$, i.e. $S \wedge \ell_1 \wedge \ldots \wedge \ell_n \models u$

$p^d q[p^d] \neg s^d u[\neg s^d] t^d \parallel$
$\neg t \vee \neg q \vee v$
$\neg t \vee \neg v$
$\neg p \vee q$
$s \vee u$
$t \vee \neg s$

$\neg t \vee \neg v$ is a conflict clause.

Analyse which decisions imply the conflict.

- t^d is already a decision literal
- $v[t^d q]$, we have $q[p^d]$ therefore $v[t^d p^d]$

1) $S \wedge t \wedge v \models \bot$ 2) $S \wedge t \wedge t \wedge q \models \bot$

3) $S \wedge p \wedge t \models \bot$

$S \wedge p \models \neg t$ hence $\neg t$ is implied by p!

Backjump to p^d and assign $\neg t[p^d]$

Backjump over several decision levels!

DPLL Backjumping

Expensive branching occurs upon decision literals.

Reducing the number of decision literals is crucial for efficiency.

$u[\ell_1 \dots \ell_n]$ – u is implied by $\ell_1, \dots, \ell_n$, i.e. $S \wedge \ell_1 \wedge \dots \wedge \ell_n \models u$

$p^d q[p^d] \neg s^d u[\neg s^d] t^d \;\|$
$\neg t \vee \neg q \vee v$
$\neg t \vee \neg v$
$\neg p \vee q$
$s \vee u$
$t \vee \neg s$

$\neg t \vee \neg v$ is a conflict clause.

Analyse which decisions imply the conflict.

- t^d is already a decision literal
- $v[t^d q]$, we have $q[p^d]$ therefore $v[t^d p^d]$

1) $S \wedge t \wedge v \models \bot$ 2) $S \wedge t \wedge t \wedge q \models \bot$

3) $S \wedge p \wedge t \models \bot$

$S \wedge p \models \neg t$ hence $\neg t$ is implied by p!

Backjump to p^d and assign $\neg t[p^d]$

Backjump over several decision levels!

DPLL Backjumping

Expensive branching occurs upon decision literals.

Reducing the number of decision literals is crucial for efficiency.

$u[\ell_1 \dots \ell_n]$ – u is implied by $\ell_1, \dots, \ell_n$, i.e. $S \wedge \ell_1 \wedge \dots \wedge \ell_n \models u$

$p^d q[p^d] \neg s^d u[\neg s^d] t^d v[t^d q] \;\|$
$\neg t \vee \neg q \vee v$
$\neg t \vee \neg v$
$\neg p \vee q$
$s \vee u$
$t \vee \neg s$

$\neg t \vee \neg v$ is a conflict clause.

Analyse which decisions imply the conflict.

- t^d is already a decision literal
- $v[t^d q]$, we have $q[p^d]$ therefore $v[t^d p^d]$

1) $S \wedge t \wedge v \models \bot$ 2) $S \wedge t \wedge t \wedge q \models \bot$

3) $S \wedge p \wedge t \models \bot$

$S \wedge p \models \neg t$ hence $\neg t$ is implied by p!

Backjump to p^d and assign $\neg t[p^d]$

Backjump over several decision levels!

DPLL Backjumping

Expensive branching occurs upon decision literals.

Reducing the number of decision literals is crucial for efficiency.

$u[\ell_1 \ldots \ell_n]$ – u is implied by $\ell_1, \ldots, \ell_n$, i.e. $S \wedge \ell_1 \wedge \ldots \wedge \ell_n \models u$

$p^d q[p^d] \neg s^d u[\neg s^d] t^d v[t^d q] \parallel$
$\neg t \vee \neg q \vee v$
$\neg t \vee \neg v$
$\neg p \vee q$
$s \vee u$
$t \vee \neg s$

$\neg t \vee \neg v$ is a conflict clause.

Analyse which decisions imply the conflict.

- t^d is already a decision literal
- $v[t^d q]$, we have $q[p^d]$ therefore $v[t^d p^d]$

1) $S \wedge t \wedge v \models \bot$ 2) $S \wedge t \wedge t \wedge q \models \bot$

3) $S \wedge p \wedge t \models \bot$

$S \wedge p \models \neg t$ hence $\neg t$ is implied by p!

Backjump to p^d and assign $\neg t[p^d]$

Backjump over several decision levels!

DPLL Backjumping

Expensive branching occurs upon decision literals.

Reducing the number of decision literals is crucial for efficiency.

$u[\ell_1 \dots \ell_n]$ – u is implied by $\ell_1, \dots, \ell_n$, i.e. $S \wedge \ell_1 \wedge \dots \wedge \ell_n \models u$

$p^d q[p^d] \neg s^d u[\neg s^d] t^d v[t^d q] \parallel$
$\neg t \vee \neg q \vee v$
$\boxed{\neg t \vee \neg v}$
$\neg p \vee q$
$s \vee u$
$t \vee \neg s$

$\neg t \vee \neg v$ is a conflict clause.

Analyse which decisions imply the conflict.

- t^d is already a decision literal
- $v[t^d q]$, we have $q[p^d]$ therefore $v[t^d p^d]$

1) $S \wedge t \wedge v \models \bot$ 2) $S \wedge t \wedge t \wedge q \models \bot$

3) $S \wedge p \wedge t \models \bot$

$S \wedge p \models \neg t$ hence $\neg t$ is implied by p!

Backjump to p^d and assign $\neg t[p^d]$

Backjump over several decision levels!

DPLL Backjumping

Expensive branching occurs upon decision literals.

Reducing the number of decision literals is crucial for efficiency.

$u[\ell_1 \ldots \ell_n]$ – u is implied by $\ell_1, \ldots, \ell_n$, i.e. $S \wedge \ell_1 \wedge \ldots \wedge \ell_n \models u$

$p^d q[p^d] \neg s^d u[\neg s^d] t^d v[t^d q] \parallel$
$\neg t \vee \neg q \vee v$
$\boxed{\neg t \vee \neg v}$
$\neg p \vee q$
$s \vee u$
$t \vee \neg s$

$\neg t \vee \neg v$ is a conflict clause.

Analyse which decisions imply the conflict.

- t^d is already a decision literal
- $v[t^d q]$, we have $q[p^d]$ therefore $v[t^d p^d]$

1) $S \wedge t \wedge v \models \bot$ 2) $S \wedge t \wedge t \wedge q \models \bot$

3) $S \wedge p \wedge t \models \bot$

$S \wedge p \models \neg t$ hence $\neg t$ is implied by p!

Backjump to p^d and assign $\neg t[p^d]$

Backjump over several decision levels!

DPLL Backjumping

Expensive branching occurs upon decision literals.

Reducing the number of decision literals is crucial for efficiency.

$u[\ell_1 \ldots \ell_n]$ – u is implied by $\ell_1, \ldots, \ell_n$, i.e. $S \wedge \ell_1 \wedge \ldots \wedge \ell_n \models u$

$p^d q[p^d] \neg s^d u[\neg s^d] t^d v[t^d p^d] \parallel$
$\neg t \vee \neg q \vee v$
$\boxed{\neg t \vee \neg v}$
$\neg p \vee q$
$s \vee u$
$t \vee \neg s$

$\neg t \vee \neg v$ is a conflict clause.

Analyse which decisions imply the conflict.

- t^d is already a decision literal
- $v[t^d q]$, we have $q[p^d]$ therefore $v[t^d p^d]$

1) $S \wedge t \wedge v \models \bot$ 2) $S \wedge t \wedge t \wedge q \models \bot$

3) $S \wedge p \wedge t \models \bot$

$S \wedge p \models \neg t$ hence $\neg t$ is implied by p!

Backjump to p^d and assign $\neg t[p^d]$

Backjump over several decision levels!

DPLL Backjumping

Expensive branching occurs upon decision literals.

Reducing the number of decision literals is crucial for efficiency.

$u[\ell_1 \dots \ell_n]$ – u is implied by $\ell_1, \dots, \ell_n$, i.e. $S \wedge \ell_1 \wedge \dots \wedge \ell_n \models u$

$p^d q[p^d] \neg s^d u[\neg s^d] t^d v[t^d p^d] \parallel$
$\neg t \vee \neg q \vee v$
$\boxed{\neg t \vee \neg v}$
$\neg p \vee q$
$s \vee u$
$t \vee \neg s$

$\neg t \vee \neg v$ is a conflict clause.

Analyse which decisions imply the conflict.

- t^d is already a decision literal
- $v[t^d q]$, we have $q[p^d]$ therefore $v[t^d p^d]$

1) $S \wedge t \wedge v \models \bot$ 2) $S \wedge t \wedge t \wedge q \models \bot$

3) $S \wedge p \wedge t \models \bot$

$S \wedge p \models \neg t$ hence $\neg t$ is implied by p!

Backjump to p^d and assign $\neg t[p^d]$

Backjump over several decision levels!

DPLL Backjumping

Expensive branching occurs upon decision literals.

Reducing the number of decision literals is crucial for efficiency.

$u[\ell_1 \ldots \ell_n]$ – u is implied by $\ell_1, \ldots, \ell_n$, i.e. $S \wedge \ell_1 \wedge \ldots \wedge \ell_n \models u$

$p^d q[p^d] \neg s^d u[\neg s^d] t^d v[t^d p^d] \parallel$
$\neg t \vee \neg q \vee v$
$\boxed{\neg t \vee \neg v}$
$\neg p \vee q$
$s \vee u$
$t \vee \neg s$

$\neg t \vee \neg v$ is a conflict clause.

Analyse which decisions imply the conflict.

- t^d is already a decision literal
- $v[t^d q]$, we have $q[p^d]$ therefore $v[t^d p^d]$

1) $S \wedge t \wedge v \models \bot$ 2) $S \wedge t \wedge t \wedge q \models \bot$

3) $S \wedge p \wedge t \models \bot$

$S \wedge p \models \neg t$ hence $\neg t$ is implied by p!

Backjump to p^d and assign $\neg t[p^d]$

Backjump over several decision levels!

DPLL Backjumping

Expensive branching occurs upon decision literals.

Reducing the number of decision literals is crucial for efficiency.

$u[\ell_1 \dots \ell_n]$ – u is implied by $\ell_1, \dots, \ell_n$, i.e. $S \wedge \ell_1 \wedge \dots \wedge \ell_n \models u$

$p^d q[p^d] \neg t[p^d] \parallel$
$\neg t \vee \neg q \vee v$
$\neg t \vee \neg v$
$\neg p \vee q$
$s \vee u$
$t \vee \neg s$

$\neg t \vee \neg v$ is a conflict clause.

Analyse which decisions imply the conflict.

- t^d is already a decision literal
- $v[t^d q]$, we have $q[p^d]$ therefore $v[t^d p^d]$

1) $S \wedge t \wedge v \models \bot$ 2) $S \wedge t \wedge t \wedge q \models \bot$

3) $S \wedge p \wedge t \models \bot$

$S \wedge p \models \neg t$ hence $\neg t$ is implied by p!

Backjump to p^d and assign $\neg t[p^d]$

Backjump over several decision levels!

DPLL Backjumping

Expensive branching occurs upon decision literals.

Reducing the number of decision literals is crucial for efficiency.

$u[\ell_1 \ldots \ell_n]$ – u is implied by $\ell_1, \ldots, \ell_n$, i.e. $S \wedge \ell_1 \wedge \ldots \wedge \ell_n \models u$

$p^d q[p^d] \neg t[p^d] \parallel$
$\neg t \vee \neg q \vee v$
$\neg t \vee \neg v$
$\neg p \vee q$
$s \vee u$
$t \vee \neg s$

$\neg t \vee \neg v$ is a conflict clause.

Analyse which decisions imply the conflict.

- t^d is already a decision literal
- $v[t^d q]$, we have $q[p^d]$ therefore $v[t^d p^d]$

1) $S \wedge t \wedge v \models \bot$ 2) $S \wedge t \wedge t \wedge q \models \bot$

3) $S \wedge p \wedge t \models \bot$

$S \wedge p \models \neg t$ hence $\neg t$ is implied by p!

Backjump to p^d and assign $\neg t[p^d]$

Backjump over several decision levels!

DPLL Backjumping

Expensive branching occurs upon decision literals.

Reducing the number of decision literals is crucial for efficiency.

$u[\ell_1 \dots \ell_n]$ – u is implied by $\ell_1, \dots, \ell_n$, i.e. $S \wedge \ell_1 \wedge \dots \wedge \ell_n \models u$

$p^d q[p^d] \neg t[p^d] \neg s[\neg t] \parallel$
$\neg t \vee \neg q \vee v$
$\neg t \vee \neg v$
$\neg p \vee q$
$s \vee u$
$t \vee \neg s$

$\neg t \vee \neg v$ is a conflict clause.

Analyse which decisions imply the conflict.

- t^d is already a decision literal
- $v[t^d q]$, we have $q[p^d]$ therefore $v[t^d p^d]$

1) $S \wedge t \wedge v \models \bot$ 2) $S \wedge t \wedge t \wedge q \models \bot$

3) $S \wedge p \wedge t \models \bot$

$S \wedge p \models \neg t$ hence $\neg t$ is implied by p!

Backjump to p^d and assign $\neg t[p^d]$

Backjump over several decision levels!

DPLL Backjumping

Expensive branching occurs upon decision literals.

Reducing the number of decision literals is crucial for efficiency.

$u[\ell_1 \dots \ell_n]$ – u is implied by $\ell_1, \dots, \ell_n$, i.e. $S \wedge \ell_1 \wedge \dots \wedge \ell_n \models u$

$p^d q[p^d] \neg t[p^d] \neg s[\neg t] \parallel$
$\neg t \vee \neg q \vee v$
$\neg t \vee \neg v$
$\neg p \vee q$
$s \vee u$
$t \vee \neg s$

$\neg t \vee \neg v$ is a conflict clause.

Analyse which decisions imply the conflict.

- t^d is already a decision literal
- $v[t^d q]$, we have $q[p^d]$ therefore $v[t^d p^d]$

1) $S \wedge t \wedge v \models \bot$ 2) $S \wedge t \wedge t \wedge q \models \bot$

3) $S \wedge p \wedge t \models \bot$

$S \wedge p \models \neg t$ hence $\neg t$ is implied by p!

Backjump to p^d and assign $\neg t[p^d]$

Backjump over several decision levels!

DPLL Backjumping

Expensive branching occurs upon decision literals.

Reducing the number of decision literals is crucial for efficiency.

$u[\ell_1 \ldots \ell_n]$ – u is implied by $\ell_1, \ldots, \ell_n$, i.e. $S \wedge \ell_1 \wedge \ldots \wedge \ell_n \models u$

$p^d q[p^d] \neg t[p^d] \neg s[\neg t] u[\neg s] \parallel$
$\neg t \vee \neg q \vee v$
$\neg t \vee \neg v$
$\neg p \vee q$
$s \vee u$
$t \vee \neg s$

$\neg t \vee \neg v$ is a conflict clause.

Analyse which decisions imply the conflict.

- t^d is already a decision literal
- $v[t^d q]$, we have $q[p^d]$ therefore $v[t^d p^d]$

1) $S \wedge t \wedge v \models \bot$ 2) $S \wedge t \wedge t \wedge q \models \bot$

3) $S \wedge p \wedge t \models \bot$

$S \wedge p \models \neg t$ hence $\neg t$ is implied by p!

Backjump to p^d and assign $\neg t[p^d]$

Backjump over several decision levels!

DPLL Backjumping

Expensive branching occurs upon decision literals.

Reducing the number of decision literals is crucial for efficiency.

$u[\ell_1 \dots \ell_n]$ – u is implied by $\ell_1, \dots, \ell_n$, i.e. $S \wedge \ell_1 \wedge \dots \wedge \ell_n \models u$

$p^d q[p^d] \neg t[p^d] \neg s[\neg t] u[\neg s] v^d \parallel$
$\neg t \vee \neg q \vee v$
$\neg t \vee \neg v$
$\neg p \vee q$
$s \vee u$
$t \vee \neg s$

$\neg t \vee \neg v$ is a conflict clause.

Analyse which decisions imply the conflict.

- t^d is already a decision literal
- $v[t^d q]$, we have $q[p^d]$ therefore $v[t^d p^d]$

1) $S \wedge t \wedge v \models \perp$ 2) $S \wedge t \wedge t \wedge q \models \perp$

3) $S \wedge p \wedge t \models \perp$

$S \wedge p \models \neg t$ hence $\neg t$ is implied by p!

Backjump to p^d and assign $\neg t[p^d]$

Backjump over several decision levels!

DPLL Backjumping

Expensive branching occurs upon decision literals.

Reducing the number of decision literals is crucial for efficiency.

$u[\ell_1 \ldots \ell_n]$ – u is implied by $\ell_1, \ldots, \ell_n$, i.e. $S \wedge \ell_1 \wedge \ldots \wedge \ell_n \models u$

$p^d q[p^d] \neg t[p^d] \neg s[\neg t] u[\neg s] v^d \parallel$
$\neg t \vee \neg q \vee v$
$\neg t \vee \neg v$
$\neg p \vee q$
$s \vee u$
$t \vee \neg s$

Satisfiable!

$\neg t \vee \neg v$ is a conflict clause.

Analyse which decisions imply the conflict.

- t^d is already a decision literal
- $v[t^d q]$, we have $q[p^d]$ therefore $v[t^d p^d]$

1) $S \wedge t \wedge v \models \bot$ 2) $S \wedge t \wedge t \wedge q \models \bot$

3) $S \wedge p \wedge t \models \bot$

$S \wedge p \models \neg t$ hence $\neg t$ is implied by p!

Backjump to p^d and assign $\neg t[p^d]$

Backjump over several decision levels!

Backjump Rule

Backjump (BJ)

$$U \ell^d V \parallel S \quad \Rightarrow_{BJ} \quad U e \parallel S \qquad \text{if} \begin{cases} I_{U\ell^d V} \models \neg C, \ \text{for } C \in S, \\ U \wedge S \models e, \\ e \text{ is undefined in } U. \end{cases}$$

Note: Backtracking is a special case of backjumping.

Main difference: We can backjump over several decision variables.

Backjump Rule

Backjump (BJ)

$$U\ell^d V \parallel S \quad \Rightarrow_{BJ} \quad Ue \parallel S \quad \text{if} \begin{cases} I_{U\ell^d V} \models \neg C, \text{ for } C \in S, \\ U \wedge S \models e, \\ e \text{ is undefined in } U. \end{cases}$$

Note: Backtracking is a special case of backjumping.

Main difference: We can backjump over several decision variables.

Backjump Rule

Backjump (BJ)

$$U\ell^d V \parallel S \quad \Rightarrow_{BJ} \quad Ue \parallel S \qquad \text{if} \begin{cases} I_{U\ell^d V} \models \neg C, \text{ for } C \in S, \\ U \wedge S \models e, \\ e \text{ is undefined in } U. \end{cases}$$

Note: Backtracking is a special case of backjumping.

Main difference: We can backjump over several decision variables.

Lemma Learning

In backjumping we add implied literals.

Lemma learning: add lemmas so that implied literals can be inferred by unit propagation.

$\|$

$\neg t \vee \neg q \vee v$

$\neg t \vee \neg v$

$\neg p \vee q$

$s \vee u$

$t \vee \neg s$

$\neg t \vee \neg v$ is a conflict clause.

Analyse which decisions imply the conflict.

- t^d is already a decision literal
- $v[t^d q]$, we have $q[p^d]$ therefore $v[t^d p^d]$

$S \wedge p \wedge t \models \bot$

$S \wedge p \models \neg t$ hence $\neg t$ is implied by p!

Backjump to p^d and assign $\neg t[p^d]$

Lemma Learning: $S \wedge p \wedge t \models \bot$ therefore $S \models \neg p \vee \neg t$

- add backjump lemma $\neg p \vee \neg t$ to S
- $\neg t$ is implied by p from $S \cup \{\neg p \vee \neg t\}$ by UP!

Lemma Learning

In backjumping we add implied literals.

Lemma learning: add lemmas so that implied literals can be inferred by unit propagation.

$p^d q[p^d] \neg s^d u[\neg s^d] t^d v[t^d q] \parallel$
$\neg t \vee \neg q \vee v$
$\boxed{\neg t \vee \neg v}$
$\neg p \vee q$
$s \vee u$
$t \vee \neg s$

$\neg t \vee \neg v$ is a conflict clause.

Analyse which decisions imply the conflict.

- t^d is already a decision literal
- $v[t^d q]$, we have $q[p^d]$ therefore $v[t^d p^d]$

$S \wedge p \wedge t \models \bot$

$S \wedge p \models \neg t$ hence $\neg t$ is implied by p!

Backjump to p^d and assign $\neg t[p^d]$

Lemma Learning: $S \wedge p \wedge t \models \bot$ therefore $S \models \neg p \vee \neg t$

- add backjump lemma $\neg p \vee \neg t$ to S
- $\neg t$ is implied by p from $S \cup \{\neg p \vee \neg t\}$ by UP!

Lemma Learning

In backjumping we add implied literals.

Lemma learning: add lemmas so that implied literals can be inferred by unit propagation.

$p^d q[p^d] \neg s^d u[\neg s^d] t^d v[t^d q] \parallel$
$\neg t \vee \neg q \vee v$
$\boxed{\neg t \vee \neg v}$
$\neg p \vee q$
$s \vee u$
$t \vee \neg s$

$\neg t \vee \neg v$ is a conflict clause.

Analyse which decisions imply the conflict.

- t^d is already a decision literal
- $v[t^d q]$, we have $q[p^d]$ therefore $v[t^d p^d]$

$S \wedge p \wedge t \models \bot$

$S \wedge p \models \neg t$ hence $\neg t$ is implied by p!

Backjump to p^d and assign $\neg t[p^d]$

Lemma Learning: $S \wedge p \wedge t \models \bot$ therefore $S \models \neg p \vee \neg t$

- add backjump lemma $\neg p \vee \neg t$ to S
- $\neg t$ is implied by p from $S \cup \{\neg p \vee \neg t\}$ by UP!

Lemma Learning

In backjumping we add implied literals.

Lemma learning: add lemmas so that implied literals can be inferred by unit propagation.

$p^d q[p^d] \neg s^d u[\neg s^d] t^d v[t^d q] \parallel$
$\neg t \vee \neg q \vee v$
$\neg t \vee \neg v$
$\neg p \vee q$
$s \vee u$
$t \vee \neg s$
$\neg p \vee \neg t$

$\neg t \vee \neg v$ is a conflict clause.

Analyse which decisions imply the conflict.

- t^d is already a decision literal
- $v[t^d q]$, we have $q[p^d]$ therefore $v[t^d p^d]$

$S \wedge p \wedge t \models \bot$

$S \wedge p \models \neg t$ hence $\neg t$ is implied by p!

Backjump to p^d and assign $\neg t[p^d]$

Lemma Learning: $S \wedge p \wedge t \models \bot$ therefore $S \models \neg p \vee \neg t$

- add backjump lemma $\neg p \vee \neg t$ to S
- $\neg t$ is implied by p from $S \cup \{\neg p \vee \neg t\}$ by UP!

Lemma Learning

In backjumping we add implied literals.

Lemma learning: add lemmas so that implied literals can be inferred by unit propagation.

$p^d q[p^d] \neg s^d u[\neg s^d] t^d v[t^d q] \parallel$
$\neg t \vee \neg q \vee v$
$\neg t \vee \neg v$
$\neg p \vee q$
$s \vee u$
$t \vee \neg s$
$\neg p \vee \neg t$

$\neg t \vee \neg v$ is a conflict clause.

Analyse which decisions imply the conflict.

- t^d is already a decision literal
- $v[t^d q]$, we have $q[p^d]$ therefore $v[t^d p^d]$

$S \wedge p \wedge t \models \bot$

$S \wedge p \models \neg t$ hence $\neg t$ is implied by p!

Backjump to p^d and assign $\neg t[p^d]$

Lemma Learning: $S \wedge p \wedge t \models \bot$ therefore $S \models \neg p \vee \neg t$

- add backjump lemma $\neg p \vee \neg t$ to S
- $\neg t$ is implied by p from $S \cup \{\neg p \vee \neg t\}$ by UP!

Lemma Learning

In backjumping we add implied literals.

Lemma learning: add lemmas so that implied literals can be inferred by unit propagation.

$p^d q[p^d] \neg t[p] \parallel$
$\neg t \vee \neg q \vee v$
$\neg t \vee \neg v$
$\neg p \vee q$
$s \vee u$
$t \vee \neg s$
$\neg p \vee \neg t$

$\neg t \vee \neg v$ is a conflict clause.

Analyse which decisions imply the conflict.

- t^d is already a decision literal
- $v[t^d q]$, we have $q[p^d]$ therefore $v[t^d p^d]$

$S \wedge p \wedge t \models \bot$

$S \wedge p \models \neg t$ hence $\neg t$ is implied by p!

Backjump to p^d and assign $\neg t[p^d]$

Lemma Learning: $S \wedge p \wedge t \models \bot$ therefore $S \models \neg p \vee \neg t$

- add backjump lemma $\neg p \vee \neg t$ to S
- $\neg t$ is implied by p from $S \cup \{\neg p \vee \neg t\}$ by UP!

Lemma Learning Rule

Lemma Learn (LL):

$$U \parallel S \quad \Rightarrow_{LL} \quad U \parallel S \cup \{C\} \qquad \text{if} \begin{cases} S \models C \\ C \text{ is set-reduced} \end{cases}$$

Note:

- Lemmas help to avoid repeated computations.
- Lemmas are reused on different branches.
- Resolution proofs of backjump lemmas can be reconstructed based on backjump analysis.
- Resolution proof of the cotradiction can be obtained from the unsatisfiable state $\bot \parallel S$.

Theorem

DPLL(BJ) and DPLL(BJ,LL) are decision procedures for propositional clausal logic.

Lemma Learning Rule

Lemma Learn (LL):

$$U \parallel S \quad \Rightarrow_{LL} \quad U \parallel S \cup \{C\} \qquad \text{if } \begin{cases} S \models C \\ C \text{ is set-reduced} \end{cases}$$

Note:

- Lemmas help to avoid repeated computations.
- Lemmas are reused on different branches.
- Resolution proofs of backjump lemmas can be reconstructed based on backjump analysis.
- Resolution proof of the cotradiction can be obtained from the unsatisfiable state $\bot \parallel S$.

Theorem

DPLL(BJ) and DPLL(BJ,LL) are decision procedures for propositional clausal logic.

Lemma Learning Rule

Lemma Learn (LL):

$$U \parallel S \quad \Rightarrow_{LL} \quad U \parallel S \cup \{C\} \qquad \text{if } \begin{cases} S \models C \\ C \text{ is set-reduced} \end{cases}$$

Note:

- Lemmas help to avoid repeated computations.
- Lemmas are reused on different branches.
- Resolution proofs of backjump lemmas can be reconstructed based on backjump analysis.
- Resolution proof of the cotradiction can be obtained from the unsatisfiable state $\perp \parallel S$.

Theorem

DPLL(BJ) and DPLL(BJ,LL) are decision procedures for propositional clausal logic.

Summary

DPLL is the most efficient RM for propositional logic known up to now.

- efficient unit propagation
- backjumping
- lemma learning

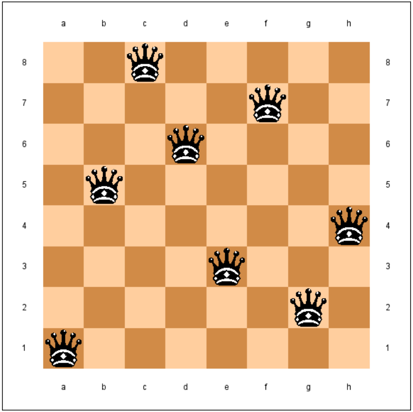
Section Formalising problems in propositional logic

N-Queens Problem

N-Queens Problem.

Place N queens on an $N \times N$ chess board such that no two queens attack each other.

Next: Formalising N-Queens Problem in propositional logic.

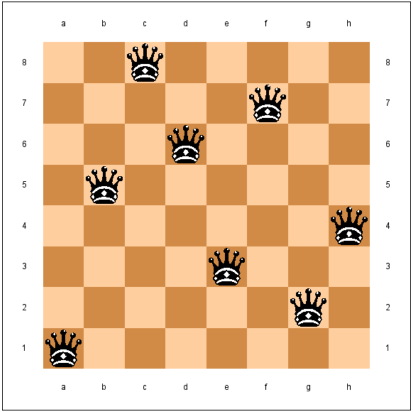


Formalising N-Queens Problem (I)

Propositional variables: q_{ij} – square (i, j) is occupied by a queen.

Rules: If q_{ij} is placed then there should be no other queen placed on

- row right: $(i, j + k)$ for $1 \leq k \leq n - j$,
- column up: $(i + k, j)$ for $1 \leq k \leq n - i$
- diag. up right: $(i + k, j + k)$ for $1 \leq k \leq \min\{n - i, n - j\}$
- diag. up left: $(i + k, j - k)$ for $1 \leq k \leq \min\{n - i, j - 1\}$

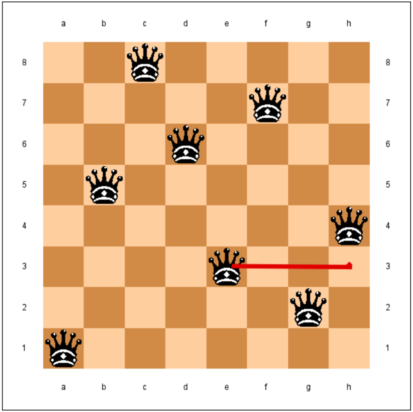


Formalising N-Queens Problem (I)

Propositional variables: q_{ij} – square (i, j) is occupied by a queen.

Rules: If q_{ij} is placed then there should be no other queen placed on

- row right: $(i, j + k)$
  for $1 \leq k \leq n - j$,
- column up: $(i + k, j)$
  for $1 \leq k \leq n - i$
- diag. up right: $(i + k, j + k)$
  for $1 \leq k \leq \min\{n - i, n - j\}$
- diag. up left: $(i + k, j - k)$
  for $1 \leq k \leq \min\{n - i, j - 1\}$



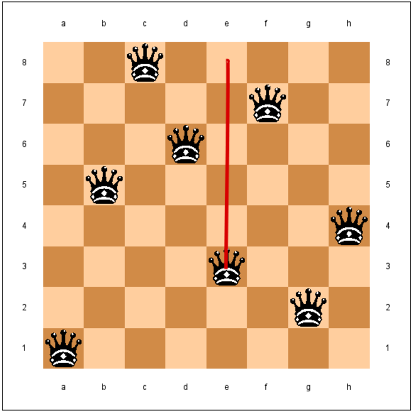
- $R_{ij} = \bigwedge_k (q_{ij} \to \neg q_{i,j+k})$
  for $1 \leq k \leq n - j$

Formalising N-Queens Problem (I)

Propositional variables: q_{ij} – square (i, j) is occupied by a queen.

Rules: If q_{ij} is placed then there should be no other queen placed on

- row right: $(i, j + k)$
  for $1 \leq k \leq n - j$,
- column up: $(i + k, j)$
  for $1 \leq k \leq n - i$
- diag. up right: $(i + k, j + k)$
  for $1 \leq k \leq \min\{n - i, n - j\}$
- diag. up left: $(i + k, j - k)$
  for $1 \leq k \leq \min\{n - i, j - 1\}$

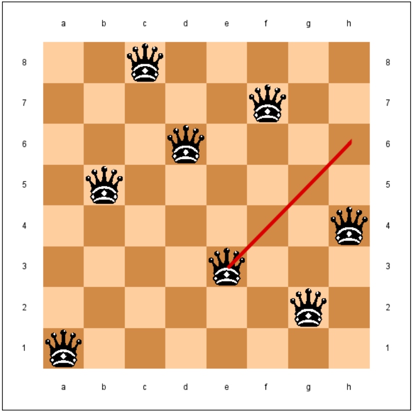


Formalising N-Queens Problem (I)

Propositional variables: q_{ij} – square (i, j) is occupied by a queen.

Rules: If q_{ij} is placed then there should be no other queen placed on

- row right: $(i, j + k)$
  for $1 \leq k \leq n - j$,
- column up: $(i + k, j)$
  for $1 \leq k \leq n - i$
- diag. up right: $(i + k, j + k)$
  for $1 \leq k \leq \min\{n - i, n - j\}$
- diag. up left: $(i + k, j - k)$
  for $1 \leq k \leq \min\{n - i, j - 1\}$

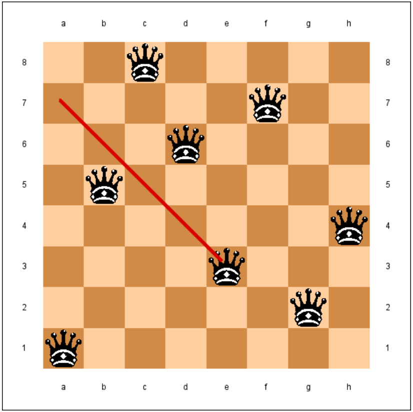


Formalising N-Queens Problem (I)

Propositional variables: q_{ij} – square (i, j) is occupied by a queen.

Rules: If q_{ij} is placed then there should be no other queen placed on

- row right: $(i, j + k)$
  for $1 \leq k \leq n - j$,
- column up: $(i + k, j)$
  for $1 \leq k \leq n - i$
- diag. up right: $(i + k, j + k)$
  for $1 \leq k \leq \min\{n - i, n - j\}$
- diag. up left: $(i + k, j - k)$
  for $1 \leq k \leq \min\{n - i, j - 1\}$



Formalising N-Queens Problem (I)

Propositional variables: q_{ij} – square (i, j) is occupied by a queen.

Rules: If q_{ij} is placed then there should be no other queen placed on

- row right: $(i, j + k)$
  for $1 \leq k \leq n - j$,
- column up: $(i + k, j)$
  for $1 \leq k \leq n - i$
- diag. up right: $(i + k, j + k)$
  for $1 \leq k \leq \min\{n - i, n - j\}$
- diag. up left: $(i + k, j - k)$
  for $1 \leq k \leq \min\{n - i, j - 1\}$

- $R_{ij} = \bigwedge_k (q_{ij} \rightarrow \neg q_{i,j+k})$
  for $1 \leq k \leq n - j$,
- $C_{ij} = \bigwedge_k (q_{ij} \rightarrow \neg q_{i+k,j})$
  for $1 \leq k \leq n - i$
- $\mathrm{DRU}_{ij} = \bigwedge_k (q_{ij} \rightarrow \neg q_{i+k,j+k})$
  for $1 \leq k \leq \min\{n - i, n - j\}$
- $\mathrm{DLU}_{ij} = \bigwedge_k (q_{ij} \rightarrow \neg q_{i-k,j+k})$
  for $1 \leq k \leq \min\{n - i, j - 1\}$

Formalising N-Queens Problem (I)

Propositional variables: q_{ij} – square (i, j) is occupied by a queen.

Rules: If q_{ij} is placed then there should be no other queen placed on

- row right: $(i, j + k)$
  for $1 \leq k \leq n - j$,
- column up: $(i + k, j)$
  for $1 \leq k \leq n - i$
- diag. up right: $(i + k, j + k)$
  for $1 \leq k \leq \min\{n - i, n - j\}$
- diag. up left: $(i + k, j - k)$
  for $1 \leq k \leq \min\{n - i, j - 1\}$

- $R_{ij} = \bigwedge_k (q_{ij} \to \neg q_{i,j+k})$
  for $1 \leq k \leq n - j$,
- $C_{ij} = \bigwedge_k (q_{ij} \to \neg q_{i+k,j})$
  for $1 \leq k \leq n - i$
- $\mathrm{DRU}_{ij} = \bigwedge_k (q_{ij} \to \neg q_{i+k,j+k})$
  for $1 \leq k \leq \min\{n - i, n - j\}$
- $\mathrm{DLU}_{ij} = \bigwedge_k (q_{ij} \to \neg q_{i-k,j+k})$
  for $1 \leq k \leq \min\{n - i, j - 1\}$

Formalising N-Queens Problem (I)

Propositional variables: q_{ij} – square (i, j) is occupied by a queen.

Rules: If q_{ij} is placed then there should be no other queen placed on

- row right: $(i, j + k)$
  for $1 \le k \le n - j$,
- column up: $(i + k, j)$
  for $1 \le k \le n - i$
- diag. up right: $(i + k, j + k)$
  for $1 \le k \le \min\{n - i, n - j\}$
- diag. up left: $(i + k, j - k)$
  for $1 \le k \le \min\{n - i, j - 1\}$

- $R_{ij} = \bigwedge_k (q_{ij} \to \neg q_{i,j+k})$
  for $1 \le k \le n - j$,
- $C_{ij} = \bigwedge_k (q_{ij} \to \neg q_{i+k,j})$
  for $1 \le k \le n - i$
- $\mathrm{DRU}_{ij} = \bigwedge_k (q_{ij} \to \neg q_{i+k,j+k})$
  for $1 \le k \le \min\{n - i, n - j\}$
- $\mathrm{DLU}_{ij} = \bigwedge_k (q_{ij} \to \neg q_{i-k,j+k})$
  for $1 \le k \le \min\{n - i, j - 1\}$

Formalising N-Queens Problem (I)

Propositional variables: q_{ij} – square (i, j) is occupied by a queen.

Rules: If q_{ij} is placed then there should be no other queen placed on

- row right: $(i, j + k)$
  for $1 \leq k \leq n - j$,
- column up: $(i + k, j)$
  for $1 \leq k \leq n - i$
- diag. up right: $(i + k, j + k)$
  for $1 \leq k \leq \min\{n - i, n - j\}$
- diag. up left: $(i + k, j - k)$
  for $1 \leq k \leq \min\{n - i, j - 1\}$

- $R_{ij} = \bigwedge_k (q_{ij} \to \neg q_{i,j+k})$
  for $1 \leq k \leq n - j$,
- $C_{ij} = \bigwedge_k (q_{ij} \to \neg q_{i+k,j})$
  for $1 \leq k \leq n - i$
- $\mathrm{DRU}_{ij} = \bigwedge_k (q_{ij} \to \neg q_{i+k,j+k})$
  for $1 \leq k \leq \min\{n - i, n - j\}$
- $\mathrm{DLU}_{ij} = \bigwedge_k (q_{ij} \to \neg q_{i-k,j+k})$
  for $1 \leq k \leq \min\{n - i, j - 1\}$

Formalising N-Queens Problem (II)

$$\begin{aligned}
\mathrm{QueenRules} &= \textstyle\bigwedge_{ij}(R_{ij} \wedge C_{ij} \wedge \mathrm{DRU}_{ij} \wedge \mathrm{DLU}_{ij}) \\
\mathrm{QueenPlaced}_i &= q_{i1} \vee \ldots \vee q_{in} \\
\mathrm{NQueensPlaced} &= \textstyle\bigwedge_i \mathrm{QueenPlaced}_i \\
\mathrm{NQueensProblem} &= \mathrm{QueenRules} \bigwedge \mathrm{NQueensPlaced}
\end{aligned}$$

Lemma

N-Queens Problem has a solution if and only if $\mathrm{NQueensProblem}$ *is satisfiable.*

Formalising N-Queens Problem (II)

$$\begin{aligned}
\text{QueenRules} &= \bigwedge_{ij}(R_{ij} \wedge C_{ij} \wedge \mathrm{DRU}_{ij} \wedge \mathrm{DLU}_{ij}) \\
\text{QueenPlaced}_i &= q_{i1} \vee \ldots \vee q_{in} \\
\text{NQueensPlaced} &= \bigwedge_i \text{QueenPlaced}_i \\
\text{NQueensProblem} &= \text{QueenRules} \bigwedge \text{NQueensPlaced}
\end{aligned}$$

Lemma

N-Queens Problem has a solution if and only if NQueensProblem *is satisfiable.*

Formalising N-Queens Problem (II)

$$\begin{aligned}
\mathrm{QueenRules} &= \bigwedge_{ij}(R_{ij} \wedge C_{ij} \wedge \mathrm{DRU}_{ij} \wedge \mathrm{DLU}_{ij}) \\
\mathrm{QueenPlaced}_i &= q_{i1} \vee \ldots \vee q_{in} \\
\mathrm{NQueensPlaced} &= \bigwedge_i \mathrm{QueenPlaced}_i \\
\mathrm{NQueensProblem} &= \mathrm{QueenRules} \bigwedge \mathrm{NQueensPlaced}
\end{aligned}$$

Lemma

N-Queens Problem has a solution if and only if $\mathrm{NQueensProblem}$ *is satisfiable.*

Formalising N-Queens Problem (II)

$$\begin{aligned}
\mathrm{QueenRules} &= \bigwedge_{ij}(R_{ij} \wedge C_{ij} \wedge \mathrm{DRU}_{ij} \wedge \mathrm{DLU}_{ij}) \\
\mathrm{QueenPlaced}_i &= q_{i1} \vee \ldots \vee q_{in} \\
\mathrm{NQueensPlaced} &= \bigwedge_i \mathrm{QueenPlaced}_i \\
\mathrm{NQueensProblem} &= \mathrm{QueenRules} \bigwedge \mathrm{NQueensPlaced}
\end{aligned}$$

Lemma

N-Queens Problem has a solution if and only if $\mathrm{NQueensProblem}$ is satisfiable.

Formalising N-Queens Problem (II)

$$
\begin{aligned}
\text{QueenRules} &= \bigwedge_{ij}(R_{ij} \wedge C_{ij} \wedge \text{DRU}_{ij} \wedge \text{DLU}_{ij}) \\
\text{QueenPlaced}_i &= q_{i1} \vee \ldots \vee q_{in} \\
\text{NQueensPlaced} &= \bigwedge_i \text{QueenPlaced}_i \\
\text{NQueensProblem} &= \text{QueenRules} \bigwedge \text{NQueensPlaced}
\end{aligned}
$$

Lemma

N-Queens Problem has a solution if and only if NQueensProblem *is satisfiable.*

Boolean functions

Boolean function of an arity n maps n sequences of truth values (Boolean values) to $\{\mathbf{0},\mathbf{1}\}$:

$$f:\{\mathbf{0},\mathbf{1}\}^n \to \{\mathbf{0},\mathbf{1}\}$$

We can define such a function by a value table:

p	q	$p+q$ *mod* 2
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$
$\mathbf{0}$	$\mathbf{1}$	$\mathbf{1}$
$\mathbf{1}$	$\mathbf{0}$	$\mathbf{1}$
$\mathbf{1}$	$\mathbf{1}$	$\mathbf{0}$

Boolean functions

Boolean function of an arity n maps n sequences of truth values (Boolean values) to $\{\mathbf{0}, \mathbf{1}\}$:

$$f : \{\mathbf{0}, \mathbf{1}\}^n \to \{\mathbf{0}, \mathbf{1}\}$$

We can define such a function by a value table:

p	q	$p + q \ mod\ 2$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$
$\mathbf{0}$	$\mathbf{1}$	$\mathbf{1}$
$\mathbf{1}$	$\mathbf{0}$	$\mathbf{1}$
$\mathbf{1}$	$\mathbf{1}$	$\mathbf{0}$

Boolean functions and formulas

We say that a Boolean function f is equivalent to a formula A if f and A have the same truth tables.

For any propositional formula there exists an equivalent Boolean function.

Assume A on variables $p_1, \ldots, p_n$ then define:

$$f_A(I(p_1), \ldots, I(p_n)) = I(F)$$

Example: $A = p \wedge q$

p	q	$f_\wedge(p, q)$
0	**0**	**0**
0	**1**	**0**
1	**0**	**0**
1	**1**	**1**

Next: show that the converse is also true.

For every Boolean function there exists an equivalent propositional formula.

Boolean functions and formulas

We say that a Boolean function f is equivalent to a formula A if f and A have the same truth tables.

For any propositional formula there exists an equivalent Boolean function.

Assume A on variables $p_1, \ldots, p_n$ then define:

$$f_A(I(p_1), \ldots, I(p_n)) = I(F)$$

Example: $A = p \wedge q$

p	q	$f_\wedge(p, q)$
0	**0**	**0**
0	**1**	**0**
1	**0**	**0**
1	**1**	**1**

Next: show that the converse is also true.

For every Boolean function there exists an equivalent propositional formula.

CNF: Truth Tables

Theorem. For every Boolean function there is an equivalent CNF.

Algorithm (Truth Tables). If all rows have value **1** then $f(p,q) \equiv \top$.

p	q	$f(p,q)$
0	**0**	**1**
0	**1**	**0**
1	**0**	**1**
1	**1**	**0**

Goal: find a set of disjunctions equiv. to $f(p,q)$

Consider a row with **0** value: **0** **1** **0**

Add for such row: $p \vee \neg q$

Next row: **1** **1** **0**

$\neg p \vee \neg q$

Resulting formula: $f(p,q) \equiv (p \vee \neg q) \wedge (\neg p \vee \neg q)$

Check $f(p,q) \equiv (p \vee \neg q) \wedge (\neg p \vee \neg q)$:

Consider a row with **0** value – then the added disjunct is false.

Consider a row with **1** value – then all disjuncts are true.

CNF: Truth Tables

Theorem. For every Boolean function there is an equivalent CNF.

Algorithm (Truth Tables). If all rows have value **1** then $f(p, q) \equiv \top$.

p	q	$f(p, q)$
0	**0**	**1**
0	**1**	**0**
1	**0**	**1**
1	**1**	**0**

Goal: find a set of disjunctions equiv. to $f(p, q)$

Consider a row with **0** value: **0** **1** **0**

Add for such row: $p \vee \neg q$

Next row: **1** **1** **0**

$\neg p \vee \neg q$

Resulting formula: $f(p, q) \equiv (p \vee \neg q) \wedge (\neg p \vee \neg q)$

Check $f(p, q) \equiv (p \vee \neg q) \wedge (\neg p \vee \neg q)$:

Consider a row with **0** value – then the added disjunct is false.

Consider a row with **1** value – then all disjuncts are true.

CNF: Truth Tables

Theorem. For every Boolean function there is an equivalent CNF.

Algorithm (Truth Tables). If all rows have value **1** then $f(p,q) \equiv \top$.

p	q	$f(p,q)$
0	**0**	**1**
0	**1**	**0**
1	**0**	**1**
1	**1**	**0**

Goal: find a set of disjunctions equiv. to $f(p,q)$

Consider a row with **0** value: **0** **1** **0**

Add for such row: $p \vee \neg q$

Next row: **1** **1** **0**

$\neg p \vee \neg q$

Resulting formula: $f(p,q) \equiv (p \vee \neg q) \wedge (\neg p \vee \neg q)$

Check $f(p,q) \equiv (p \vee \neg q) \wedge (\neg p \vee \neg q)$:

Consider a row with **0** value – then the added disjunct is false.

Consider a row with **1** value – then all disjuncts are true.

CNF: Truth Tables

Theorem. For every Boolean function there is an equivalent CNF.

Algorithm (Truth Tables). If all rows have value **1** then $f(p, q) \equiv \top$.

p	q	$f(p, q)$
0	**0**	**1**
0	**1**	**0**
1	**0**	**1**
1	**1**	**0**

Goal: find a set of disjunctions equiv. to $f(p, q)$

Consider a row with **0** value: **0** **1** **0**

Add for such row: $p \vee \neg q$

Next row: **1** **1** **0**

$\neg p \vee \neg q$

Resulting formula: $f(p, q) \equiv (p \vee \neg q) \wedge (\neg p \vee \neg q)$

Check $f(p, q) \equiv (p \vee \neg q) \wedge (\neg p \vee \neg q)$:

Consider a row with **0** value – then the added disjunct is false.

Consider a row with **1** value – then all disjuncts are true.

CNF: Truth Tables

Theorem. For every Boolean function there is an equivalent CNF.

Algorithm (Truth Tables). If all rows have value **1** then $f(p,q) \equiv \top$.

p	q	$f(p,q)$
0	**0**	**1**
0	**1**	**0**
1	**0**	**1**
1	**1**	**0**

Goal: find a set of disjunctions equiv. to $f(p,q)$

Consider a row with **0** value: **0** **1** **0**

Add for such row: $p \vee \neg q$

Next row: **1** **1** **0**

$\neg p \vee \neg q$

Resulting formula: $f(p,q) \equiv (p \vee \neg q) \wedge (\neg p \vee \neg q)$

Check $f(p,q) \equiv (p \vee \neg q) \wedge (\neg p \vee \neg q)$:

Consider a row with **0** value – then the added disjunct is false.

Consider a row with **1** value – then all disjuncts are true.

CNF: Truth Tables

Theorem. For every Boolean function there is an equivalent CNF.

Algorithm (Truth Tables). If all rows have value **1** then $f(p,q) \equiv \top$.

p	q	$f(p,q)$
0	**0**	**1**
0	**1**	**0**
1	**0**	**1**
1	**1**	**0**

Goal: find a set of disjunctions equiv. to $f(p,q)$

Consider a row with **0** value: **0** **1** **0**

Add for such row: $p \vee \neg q$

Next row: **1** **1** **0**

$\neg p \vee \neg q$

Resulting formula: $f(p,q) \equiv (p \vee \neg q) \wedge (\neg p \vee \neg q)$

Check $f(p,q) \equiv (p \vee \neg q) \wedge (\neg p \vee \neg q)$:

Consider a row with **0** value – then the added disjunct is false.

Consider a row with **1** value – then all disjuncts are true.

CNF: Truth Tables

Theorem. For every Boolean function there is an equivalent CNF.

Algorithm (Truth Tables). If all rows have value $\mathbf{1}$ then $f(p,q) \equiv \top$.

p	q	$f(p,q)$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{1}$
$\mathbf{0}$	$\mathbf{1}$	$\mathbf{0}$
$\mathbf{1}$	$\mathbf{0}$	$\mathbf{1}$
$\mathbf{1}$	$\mathbf{1}$	$\mathbf{0}$

Goal: find a set of disjunctions equiv. to $f(p,q)$

Consider a row with $\mathbf{0}$ value: $\mathbf{0}$ $\mathbf{1}$ $\mathbf{0}$

Add for such row: $p \vee \neg q$

Next row: $\mathbf{1}$ $\mathbf{1}$ $\mathbf{0}$

$\neg p \vee \neg q$

Resulting formula: $f(p,q) \equiv (p \vee \neg q) \wedge (\neg p \vee \neg q)$

Check $f(p,q) \equiv (p \vee \neg q) \wedge (\neg p \vee \neg q)$:

Consider a row with $\mathbf{0}$ value – then the added disjunct is false.

Consider a row with $\mathbf{1}$ value – then all disjuncts are true.

Systems of linear inequalities

Fundamental problem used in many applications.

Given a system of linear inequalities over natural numbers find whether it has a solution.

Example:

$$\begin{aligned} x + y - 2z &\geq 1 \\ y - x &\geq 2 \\ z &\geq 1 \end{aligned}$$

Related to linear discrete optimization.

Applications:

- scheduling
- planning
- finanice

Systems of linear inequalities

Fundamental problem used in many applications.

Given a system of linear inequalities over natural numbers find whether it has a solution.

Example:

$$\begin{aligned} x + y - 2z &\geq 1 \\ y - x &\geq 2 \\ z &\geq 1 \end{aligned}$$

Has a solution: $x = 1, y = 3, z = 1$.

Related to linear discrete optimization.

Applications:

- scheduling
- planning
- finanice

Systems of linear inequalities

Fundamental problem used in many applications.

Given a system of linear inequalities over natural numbers find whether it has a solution.

Example:

$$\begin{aligned} x + y - 2z &\geq 1 \\ y - x &\geq 2 \\ z &\geq 1 \end{aligned}$$

Has a solution: $x = 1, y = 3, z = 1$.

Related to linear discrete optimization.

Applications:

- scheduling
- planning
- finanice

Methods of solving based on the simplex algorithm+ cutting planes.

Systems of linear inequalities

Fundamental problem used in many applications.

Given a system of linear inequalities over natural numbers find whether it has a solution.

Example:

$$\begin{array}{rcl} x + y - 2z & \geq & 1 \\ y - x & \geq & 2 \\ z & \geq & 1 \end{array}$$

Has a solution: $x = 1, y = 3, z = 1$.

Related to linear discrete optimization.

Applications:

- scheduling
- planning
- finanice

Methods of solving based on the simplex algorithm+ cutting planes.

Can we apply propositional SAT solvers to this problem?

Encoding fixed bit-width arithmetic

Binary notation.

2	1	0
1	1	0

Bit-vector of length 3.

Represents the number: $0 \times 2^0 + 1 \times 2^1 + 1 \times 2^2 = 6$

Numbers can be represented as sequences of bits in other words sequences of Boolean values.

We restrict our considerations to numbers of a fixed bit width n.

Encoding fixed bit-width arithmetic

Binary notation.

2	1	0
1	1	0

Bit-vector of length 3.

Represents the number: $0 \times 2^0 + 1 \times 2^1 + 1 \times 2^2 = 6$

Numbers can be represented as sequences of bits in other words sequences of Boolean values.

We restrict our considerations to numbers of a fixed bit width n.

Encoding fixed bit-width arithmetic

Binary notation.

2	1	0
1	1	0

Bit-vector of length 3.

Represents the number: $0 \times 2^0 + 1 \times 2^1 + 1 \times 2^2 = 6$

Numbers can be represented as sequences of bits in other words sequences of Boolean values.

We restrict our considerations to numbers of a fixed bit width n.

Representing arithmetic variables.

How to represent variables over binary numbers (of bit-width n) using propositional logic ?

Consider a variable x.

Introduce Boolean variables for each bit of x: $b_0^x, b_1^x, \ldots, b_{n-1}^x$.

The sequence $\langle b_0^x, \ldots, b_{n-1}^x \rangle$ represents x in binary notation.

Representing arithmetic operations.

Examples:

arithmetic relation	representation
$x = 0$	
$x = 5$	
$x \geq 1$	
$x = y$	

Representing arithmetic operations.

Examples:

arithmetic relation	representation
$x = 0$	$\neg b_0^x \wedge \ldots \wedge \neg b_{n-1}^x$
$x = 5$	
$x \geq 1$	
$x = y$	

Representing arithmetic operations.

Examples:

arithmetic relation	representation
$x = 0$	$\neg b_0^x \wedge \ldots \wedge \neg b_{n-1}^x$
$x = 5$	$b_0^x \wedge \neg b_1^x \wedge b_2^x \wedge \neg b_3^x \ldots \wedge \neg b_{n-1}^x$
$x \geq 1$	
$x = y$	

Representing arithmetic operations.

Examples:

arithmetic relation	representation
$x = 0$	$\neg b_0^x \wedge \ldots \wedge \neg b_{n-1}^x$
$x = 5$	$b_0^x \wedge \neg b_1^x \wedge b_2^x \wedge \neg b_3^x \ldots \wedge \neg b_{n-1}^x$
$x \geq 1$	$b_0^x \vee \ldots \vee b_{n-1}^x$
$x = y$	

Representing arithmetic operations.

Examples:

arithmetic relation	representation
$x = 0$	$\neg b_0^x \wedge \ldots \wedge \neg b_{n-1}^x$
$x = 5$	$b_0^x \wedge \neg b_1^x \wedge b_2^x \wedge \neg b_3^x \ldots \wedge \neg b_{n-1}^x$
$x \geq 1$	$b_0^x \vee \ldots \vee b_{n-1}^x$
$x = y$	$(b_0^x \leftrightarrow b_0^y) \wedge \ldots \wedge (b_{n-1}^x \leftrightarrow b_{n-1}^y)$

$x + y = z$

0	0	1	1	x
0	1	0	1	y
				z
				c (*carry*)

$x + y = z$

0	0	1	1	x
0	1	0	1	y
			0	z
				c ($carry$)

$x + y = z$

0	0	1	1	x
0	1	0	1	y
			0	z
			1	c $(carry)$

$x + y = z$

0	0	1	1	x
0	1	0	1	y
		0	0	z
			1	c (*carry*)

$x + y = z$

0	0	1	1	x
0	1	0	1	y
		0	0	z
		1	1	c (*carry*)

$x + y = z$

0	0	1	1	x
0	1	0	1	y
	0	0	0	z
		1	1	c (*carry*)

$x + y = z$

0	0	1	1	x
0	1	0	1	y
	0	0	0	z
	1	1	1	c (*carry*)

$x + y = z$

0	0	1	1	x
0	1	0	1	y
1	0	0	0	z
0	1	1	1	c (*carry*)

$x + y = z$

0	0	1	1	x
0	1	0	1	y
1	0	0	0	z
0	1	1	1	c $(carry)$

Input			Output	
b_i^x	b_i^y	b_{i-1}^c	b_i^z	b_i^c
0	0	0	0	0
1	0	0	1	0
0	1	0	1	0
1	1	0	0	1
0	0	1	1	0
1	0	1	0	1
0	1	1	0	1
1	1	1	1	1

$x + y = z$

0	0	1	1	x
0	1	0	1	y
1	0	0	0	z
0	1	1	1	c (*carry*)

Input			Output	
b_i^x	b_i^y	b_{i-1}^c	b_i^z	b_i^c
0	0	0	0	0
1	0	0	1	0
0	1	0	1	0
1	1	0	0	1
0	0	1	1	0
1	0	1	0	1
0	1	1	0	1
1	1	1	1	1

Define propositional formula $add(\bar{b}^x, \bar{b}^y, \bar{b}^z)$ representing $x + y = z$.

(Recall CNF from truth tables)

$b_i^z \leftrightarrow$

$[(b_i^x \vee b_i^y \vee b_{i-1}^c) \wedge (\neg b_i^x \vee \neg b_i^y \vee b_{i-1}^c) \wedge (\neg b_i^x \vee b_i^y \vee \neg b_{i-1}^c) \wedge (b_i^x \vee \neg b_i^y \vee \neg b_{i-1}^c)]$

$x + y = z$ *cont.*

Define propositional formulas:

$F_i = b_i^z \leftrightarrow$

$[(b_i^x \vee b_i^y \vee b_{i-1}^c) \wedge (\neg b_i^x \vee \neg b_i^y \vee b_{i-1}^c) \wedge (\neg b_i^x \vee b_i^y \vee \neg b_{i-1}^c) \wedge (b_i^x \vee \neg b_i^y \vee \neg b_{i-1}^c)]$

$G_i = b_i^c \leftrightarrow$

$[(b_i^x \vee b_i^y \vee b_{i-1}^c) \wedge (\neg b_i^x \vee b_i^y \vee b_{i-1}^c) \wedge (b_i^x \vee \neg b_i^y \vee b_{i-1}^c) \wedge (b_i^x \vee b_i^y \vee \neg b_{i-1}^c)]$

For $0 \leq i \leq n-1$, where in the case when $i = 0$, b_{i-1}^c is replaced by $\perp$.

Finally

$$add(\bar{b}^x, \bar{b}^y, \bar{b}^z) = \left[\bigwedge_{0 \leq i \leq n-1} F_i \wedge G_i \right] \wedge \neg c_{n-1}$$

Question: why $\neg c_{n-1}$ was added to the formula?

$x + y = z$ *cont.*

Define propositional formulas:

$F_i = b_i^z \leftrightarrow$
$[(b_i^x \vee b_i^y \vee b_{i-1}^c) \wedge (\neg b_i^x \vee \neg b_i^y \vee b_{i-1}^c) \wedge (\neg b_i^x \vee b_i^y \vee \neg b_{i-1}^c) \wedge (b_i^x \vee \neg b_i^y \vee \neg b_{i-1}^c)]$

$G_i = b_i^c \leftrightarrow$
$[(b_i^x \vee b_i^y \vee b_{i-1}^c) \wedge (\neg b_i^x \vee b_i^y \vee b_{i-1}^c) \wedge (b_i^x \vee \neg b_i^y \vee b_{i-1}^c) \wedge (b_i^x \vee b_i^y \vee \neg b_{i-1}^c)]$

For $0 \leq i \leq n-1$, where in the case when $i = 0$, b_{i-1}^c is replaced by $\perp$.

Finally

$$add(\bar{b}^x, \bar{b}^y, \bar{b}^z) = \left[\bigwedge_{0 \leq i \leq n-1} F_i \wedge G_i \right] \wedge \neg c_{n-1}$$

Question: why $\neg c_{n-1}$ was added to the formula?

$x + y = z$ *cont.*

Define propositional formulas:

$F_i = b_i^z \leftrightarrow$
$[(b_i^x \vee b_i^y \vee b_{i-1}^c) \wedge (\neg b_i^x \vee \neg b_i^y \vee b_{i-1}^c) \wedge (\neg b_i^x \vee b_i^y \vee \neg b_{i-1}^c) \wedge (b_i^x \vee \neg b_i^y \vee \neg b_{i-1}^c)]$

$G_i = b_i^c \leftrightarrow$
$[(b_i^x \vee b_i^y \vee b_{i-1}^c) \wedge (\neg b_i^x \vee b_i^y \vee b_{i-1}^c) \wedge (b_i^x \vee \neg b_i^y \vee b_{i-1}^c) \wedge (b_i^x \vee b_i^y \vee \neg b_{i-1}^c)]$

For $0 \leq i \leq n-1$, where in the case when $i = 0$, b_{i-1}^c is replaced by $\perp$.

Finally

$$add(\bar{b}^x, \bar{b}^y, \bar{b}^z) = \left[\bigwedge_{0 \leq i \leq n-1} F_i \wedge G_i \right] \wedge \neg c_{n-1}$$

Question: why $\neg c_{n-1}$ was added to the formula?

$x + y = z$ *cont.*

Define propositional formulas:

$F_i = b_i^z \leftrightarrow$

$[(b_i^x \vee b_i^y \vee b_{i-1}^c) \wedge (\neg b_i^x \vee \neg b_i^y \vee b_{i-1}^c) \wedge (\neg b_i^x \vee b_i^y \vee \neg b_{i-1}^c) \wedge (b_i^x \vee \neg b_i^y \vee \neg b_{i-1}^c)]$

$G_i = b_i^c \leftrightarrow$

$[(b_i^x \vee b_i^y \vee b_{i-1}^c) \wedge (\neg b_i^x \vee b_i^y \vee b_{i-1}^c) \wedge (b_i^x \vee \neg b_i^y \vee b_{i-1}^c) \wedge (b_i^x \vee b_i^y \vee \neg b_{i-1}^c)]$

For $0 \leq i \leq n-1$, where in the case when $i = 0$, b_{i-1}^c is replaced by $\perp$.

Finally

$$add(\bar{b}^x, \bar{b}^y, \bar{b}^z) = \left[\bigwedge_{0 \leq i \leq n-1} F_i \wedge G_i \right] \wedge \neg c_{n-1}$$

Question: why $\neg c_{n-1}$ was added to the formula?

Linear inequalities

Exercise: define propositional formula $greater(\bar{b}^x, \bar{b}^y)$ representing $x \geq y$.

Then an inequality

$$x + x + y \geq z$$

is represented by propositional formula:

$$add(\bar{b}^x, \bar{b}^x, \bar{b}^u) \wedge add(\bar{b}^u, \bar{b}^y, \bar{b}^v) \wedge greater(\bar{b}^v, \bar{b}^z)$$

Where $\bar{b}^u$ and $\bar{b}^v$ represent intermediate results of summations.

Systems of linear inequalities are represented by conjunction of propositional formulas representing inequalities.

Lemma. A system of linear inequalities has a solution of bit width n if and only if its propositional representation is satisfiable.

Linear inequalities

Exercise: define propositional formula $greater(\bar{b}^x, \bar{b}^y)$ representing $x \geq y$. Then an inequality

$$x + x + y \geq z$$

is represented by propositional formula:

$$add(\bar{b}^x, \bar{b}^x, \bar{b}^u) \wedge add(\bar{b}^u, \bar{b}^y, \bar{b}^v) \wedge greater(\bar{b}^v, \bar{b}^z)$$

Where $\bar{b}^u$ and $\bar{b}^v$ represent intermediate results of summations.

Systems of linear inequalities are represented by conjunction of propositional formulas representing inequalities.

Lemma. A system of linear inequalities has a solution of bit width n if and only if its propositional representation is satisfiable.

Linear inequalities

Exercise: define propositional formula $greater(\bar{b}^x, \bar{b}^y)$ representing $x \geq y$. Then an inequality

$$x + x + y \geq z$$

is represented by propositional formula:

$$add(\bar{b}^x, \bar{b}^x, \bar{b}^u) \land add(\bar{b}^u, \bar{b}^y, \bar{b}^v) \land greater(\bar{b}^v, \bar{b}^z)$$

Where $\bar{b}^u$ and $\bar{b}^v$ represent intermediate results of summations.

Systems of linear inequalities are represented by conjunction of propositional formulas representing inequalities.

Lemma. A system of linear inequalities has a solution of bit width n if and only if its propositional representation is satisfiable.

Linear inequalities

Exercise: define propositional formula $greater(\bar{b}^x, \bar{b}^y)$ representing $x \geq y$. Then an inequality

$$x + x + y \geq z$$

is represented by propositional formula:

$$add(\bar{b}^x, \bar{b}^x, \bar{b}^u) \land add(\bar{b}^u, \bar{b}^y, \bar{b}^v) \land greater(\bar{b}^v, \bar{b}^z)$$

Where $\bar{b}^u$ and $\bar{b}^v$ represent intermediate results of summations.

Systems of linear inequalities are represented by conjunction of propositional formulas representing inequalities.

Lemma. A system of linear inequalities has a solution of bit width n if and only if its propositional representation is satisfiable.

Non-linear (in)equalities

Systems of non-linear equalities:

$$
\begin{aligned}
3x^3 - 2y^2 + z &\geq 2 \\
y \times z^2 - x^6 &= 10 \\
x \times y \times z &\geq 23
\end{aligned}
$$

Problem: find a solution for such a system or show that no solution exists.

A bit of history

Diophantine equations (Diophantus of Alexandria 3d century AD).

Given a non-linear equation

$p(x_1, \ldots, x_n) = 0$.

Does it have an integer solution ?

Example: $x^5 - xy + z^5 - 13 = 0$

Fundamental problem in mathematics: Euler, Gauss, Abel, Galois ...

Hilbert's 10th problem (1900)

Is there an algorithm for solving

Diophantine equations?

A bit of history

Diophantine equations (Diophantus of Alexandria 3d century AD).

Given a non-linear equation

$p(x_1, \ldots, x_n) = 0$.

Does it have an integer solution ?

Example: $x^5 - xy + z^5 - 13 = 0$

Fundamental problem in mathematics: Euler, Gauss, Abel, Galois ...

Hilbert's 10th problem (1900)

Is there an algorithm for solving

Diophantine equations?

A bit of history

Diophantine equations (Diophantus of Alexandria 3d century AD).

Given a non-linear equation

$p(x_1, \ldots, x_n) = 0$.

Does it have an integer solution ?

Example: $x^5 - xy + z^5 - 13 = 0$


DIOPHANTI
ALEXANDRINI
ARITHMETICORVM
LIBRI SEX.
ET DE NVMERIS MVLTANGVLIS
LIBER VNVS.
LVTETIAE PARISIORVM,
Sumptibus SEBASTIANI CRAMOISY, via
Iacobæa, sub Ciconiis.
M. DC. XXI.
CVM PRIVILEGIO REGIS.

Fundamental problem in mathematics: Euler, Gauss, Abel, Galois ...

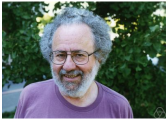
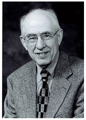
Hilbert's 10th problem (1900):

Is there an algorithm for solving

Diophantine equations?

A bit of history cont.

10th Hilbert's problem solved in 1970 by Yuri Matiyasevich based on work of Martin Davis, Hilary Putnam, Julia Robinson.

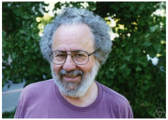
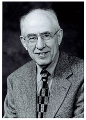

Theorem (DPRM). There is no algorithm which given a Diophantine equation outputs whether it has a solution.

DPRM theorem holds even when we restrict the number of variables to 9.

Already equations over 3 variables are problematic for analytic methods.

A bit of history cont.

10th Hilbert's problem solved in 1970 by Yuri Matiyasevich based on work of Martin Davis, Hilary Putnam, Julia Robinson.

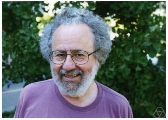
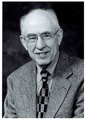
Theorem (DPRM). There is no algorithm which given a Diophantine equation outputs whether it has a solution.

DPRM theorem holds even when we restrict the number of variables to 9.

Already equations over 3 variables are problematic for analytic methods.

A bit of history cont.

10th Hilbert's problem solved in 1970 by Yuri Matiyasevich based on work of Martin Davis, Hilary Putnam, Julia Robinson.

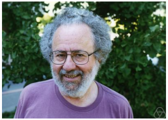
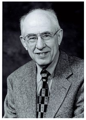
Theorem (DPRM). There is no algorithm which given a Diophantine equation outputs whether it has a solution.

DPRM theorem holds even when we restrict the number of variables to 9.

Already equations over 3 variables are problematic for analytic methods.

A bit of history cont.

10th Hilbert's problem solved in 1970 by Yuri Matiyasevich based on work of Martin Davis, Hilary Putnam, Julia Robinson.

Theorem (DPRM). There is no algorithm which given a Diophantine equation outputs whether it has a solution.

DPRM theorem holds even when we restrict the number of variables to 9.

Already equations over 3 variables are problematic for analytic methods.

Encoding non-linear (in)-equalities into SAT

Can we apply propositional methods for solving non-linear equations/inequalities ?

Consider numbers of a fixed bit-width.

Exercise: Define $mult(\bar{b}^x, \bar{b}^y, \bar{b}^z)$ representing $x \times y = z$.

Lemma. A system of non-linear (in)equalities has a solution of bit-width n if and only if its propositional representation is satisfiable.

Encoding non-linear (in)-equalities into SAT

Can we apply propositional methods for solving non-linear equations/inequalities ?

Consider numbers of a fixed bit-width.

Exercise: Define $mult(\bar{b}^x, \bar{b}^y, \bar{b}^z)$ representing $x \times y = z$.

Lemma. A system of non-linear (in)equalities has a solution of bit-width n if and only if its propositional representation is satisfiable.

Encoding non-linear (in)-equalities into SAT

Can we apply propositional methods for solving non-linear equations/inequalities ?

Consider numbers of a fixed bit-width.

Exercise: Define $mult(\bar{b}^x, \bar{b}^y, \bar{b}^z)$ representing $x \times y = z$.

Lemma. A system of non-linear (in)equalities has a solution of bit-width n if and only if its propositional representation is satisfiable.

Summary

Propositional logic can be used to encode many combinatorial problems.

- N-Queens problem
- Solving systems of linear and non-linear constraints
- optimization problems
- planning
- scheduling
- verification
- ...