

CS:4350 Logic in Computer Science

Quantified Boolean Formulas

Cesare Tinelli

Spring 2021



Credits

These slides are largely based on slides originally developed by **Andrei Voronkov** at the University of Manchester. Adapted by permission.

Outline

Quantified Boolean Formulas

- Syntax and Semantics

- Free and Bound Variables

- Prenex Form

- Satisfiability Checking

 - Splitting

 - Conjunctive Normal Form

 - DPLL

- QBF and BDDs

Two-Player Games



Does she have a winning strategy?

Two-Player Games

Given: a propositional formula G with variables $p_1, q_1, \dots, p_n, q_n$

Two-Player Games

Given: a propositional formula G with variables $p_1, q_1, \dots, p_n, q_n$

There are two players: P and Q

Two-Player Games

Given: a propositional formula G with variables $p_1, q_1, \dots, p_n, q_n$

There are two players: P and Q

At step k each player makes a move:

Two-Player Games

Given: a propositional formula G with variables $p_1, q_1, \dots, p_n, q_n$

There are two players: P and Q

At step k each player makes a move:

1. the player P can choose a Boolean value for the variable p_k

Two-Player Games

Given: a propositional formula G with variables $p_1, q_1, \dots, p_n, q_n$

There are two players: P and Q

At step k each player makes a move:

1. the player P can choose a Boolean value for the variable p_k
2. the player Q can choose a Boolean value for the variable q_k

Two-Player Games

Given: a propositional formula G with variables $p_1, q_1, \dots, p_n, q_n$

There are two players: P and Q

At step k each player makes a move:

1. the player P can choose a Boolean value for the variable p_k
2. the player Q can choose a Boolean value for the variable q_k

Player P wins if after n steps the chosen values satisfy formula G

Suppose Both Players Make no Errors: Who Wins?

Consider several special cases:

G

Outcome

Suppose Both Players Make no Errors: Who Wins?

Consider several special cases:

G	Outcome
1. p_1	

Suppose Both Players Make no Errors: Who Wins?

Consider several special cases:

G	Outcome
1. p_1	P wins with $\{p_1 \mapsto 1\}$

Suppose Both Players Make no Errors: Who Wins?

Consider several special cases:

G	Outcome
1. p_1	P wins with $\{p_1 \mapsto 1\}$
2. $p_1 \rightarrow q_1$	

Suppose Both Players Make no Errors: Who Wins?

Consider several special cases:

G	Outcome
1. p_1	P wins with $\{p_1 \mapsto 1\}$
2. $p_1 \rightarrow q_1$	P wins with $\{p_1 \mapsto 0\}$

Suppose Both Players Make no Errors: Who Wins?

Consider several special cases:

G	Outcome
1. p_1	P wins with $\{p_1 \mapsto 1\}$
2. $p_1 \rightarrow q_1$	P wins with $\{p_1 \mapsto 0\}$
3. $q_1 \rightarrow q_2$	

Suppose Both Players Make no Errors: Who Wins?

Consider several special cases:

G	Outcome
1. p_1	P wins with $\{p_1 \mapsto 1\}$
2. $p_1 \rightarrow q_1$	P wins with $\{p_1 \mapsto 0\}$
3. $q_1 \rightarrow q_2$	G has no p_i vars, P 's choices are immaterial

Suppose Both Players Make no Errors: Who Wins?

Consider several special cases:

G	Outcome
1. p_1	P wins with $\{p_1 \mapsto 1\}$
2. $p_1 \rightarrow q_1$	P wins with $\{p_1 \mapsto 0\}$
3. $q_1 \rightarrow q_2$	G has no p_i vars, P 's choices are immaterial
4. $q_1 \rightarrow q_1$	

Suppose Both Players Make no Errors: Who Wins?

Consider several special cases:

G	Outcome
1. p_1	P wins with $\{p_1 \mapsto 1\}$
2. $p_1 \rightarrow q_1$	P wins with $\{p_1 \mapsto 0\}$
3. $q_1 \rightarrow q_2$	G has no p_i vars, P 's choices are immaterial
4. $q_1 \rightarrow q_1$	G is valid, P always wins!

Suppose Both Players Make no Errors: Who Wins?

Consider several special cases:

G	Outcome
1. p_1	P wins with $\{p_1 \mapsto 1\}$
2. $p_1 \rightarrow q_1$	P wins with $\{p_1 \mapsto 0\}$
3. $q_1 \rightarrow q_2$	G has no p_i vars, P 's choices are immaterial
4. $q_1 \rightarrow q_1$	G is valid, P always wins!
5. $p_1 \wedge \neg p_1$	

Suppose Both Players Make no Errors: Who Wins?

Consider several special cases:

G	Outcome
1. p_1	P wins with $\{p_1 \mapsto 1\}$
2. $p_1 \rightarrow q_1$	P wins with $\{p_1 \mapsto 0\}$
3. $q_1 \rightarrow q_2$	G has no p_i vars, P 's choices are immaterial
4. $q_1 \rightarrow q_1$	G is valid, P always wins!
5. $p_1 \wedge \neg p_1$	G is unsatisfiable, Q always wins!

Suppose Both Players Make no Errors: Who Wins?

Consider several special cases:

G	Outcome
1. p_1	P wins with $\{p_1 \mapsto 1\}$
2. $p_1 \rightarrow q_1$	P wins with $\{p_1 \mapsto 0\}$
3. $q_1 \rightarrow q_2$	G has no p_i vars, P 's choices are immaterial
4. $q_1 \rightarrow q_1$	G is valid, P always wins!
5. $p_1 \wedge \neg p_1$	G is unsatisfiable, Q always wins!
6. $p_1 \leftrightarrow q_1$	

Suppose Both Players Make no Errors: Who Wins?

Consider several special cases:

G	Outcome
1. p_1	P wins with $\{p_1 \mapsto 1\}$
2. $p_1 \rightarrow q_1$	P wins with $\{p_1 \mapsto 0\}$
3. $q_1 \rightarrow q_2$	G has no p_i vars, P 's choices are immaterial
4. $q_1 \rightarrow q_1$	G is valid, P always wins!
5. $p_1 \wedge \neg p_1$	G is unsatisfiable, Q always wins!
6. $p_1 \leftrightarrow q_1$	each move by P can be beaten by Q

Winning Strategy

Problem: does P have a winning strategy?

Winning Strategy

Problem: does P have a winning strategy?

P has a winning strategy

iff

there exists a move for P (a value for p_1) such that

Winning Strategy

Problem: does P have a winning strategy?

P has a winning strategy

iff

there exists a move for P (a value for p_1) such that
for all moves of Q (values for q_1)

Winning Strategy

Problem: does P have a winning strategy?

P has a winning strategy

iff

there exists a move for P (a value for p_1) such that

for all moves of Q (values for q_1)

there exists a move for P (a value for p_2) such that

Winning Strategy

Problem: does P have a winning strategy?

P has a winning strategy

iff

there exists a move for P (a value for p_1) such that

for all moves of Q (values for q_1)

there exists a move for P (a value for p_2) such that

for all moves of Q (values for q_2)

Winning Strategy

Problem: does P have a winning strategy?

P has a winning strategy

iff

there exists a move for P (a value for p_1) such that

for all moves of Q (values for q_1)

there exists a move for P (a value for p_2) such that

for all moves of Q (values for q_2)

...

Winning Strategy

Problem: does P have a winning strategy?

P has a winning strategy

iff

there exists a move for P (a value for p_1) such that

for all moves of Q (values for q_1)

there exists a move for P (a value for p_2) such that

for all moves of Q (values for q_2)

...

there exists a move for P (a value for p_n) such that

Winning Strategy

Problem: does P have a winning strategy?

P has a winning strategy

iff

there exists a move for P (a value for p_1) such that

for all moves of Q (values for q_1)

there exists a move for P (a value for p_2) such that

for all moves of Q (values for q_2)

...

there exists a move for P (a value for p_n) such that

for all moves of Q (values for q_n)

Winning Strategy

Problem: does P have a winning strategy?

P has a winning strategy

iff

there exists a move for P (a value for p_1) such that

for all moves of Q (values for q_1)

there exists a move for P (a value for p_2) such that

for all moves of Q (values for q_2)

...

there exists a move for P (a value for p_n) such that

for all moves of Q (values for q_n)

the formula G is satisfiable

Winning Strategy

Problem: does P have a winning strategy?

P has a winning strategy

iff

there exists a move for P (a value for p_1) such that

for all moves of Q (values for q_1)

there exists a move for P (a value for p_2) such that

for all moves of Q (values for q_2)

...

there exists a move for P (a value for p_n) such that

for all moves of Q (values for q_n)

the formula G is satisfiable

The existence of a winning strategy can be expressed by the *quantified Boolean formula*

$$\exists p_1 \forall q_1 \exists p_2 \forall q_2 \dots \exists p_n \forall q_n G$$

Quantified Boolean Formulas

Propositional Formula:

- Every Boolean variable is a (propositional) formula
- \top and \perp are formulas
- If F is a PF, then $\neg F$ is a formula
- If F_1, \dots, F_n are formulas, where $n \geq 2$, then $(F_1 \wedge \dots \wedge F_n)$ and $(F_1 \vee \dots \vee F_n)$ are formulas
- If F and G are formulas, then $(F \rightarrow G)$ and $(F \leftrightarrow G)$ are formulas

Quantified Boolean Formulas

Propositional Formula:

- Every Boolean variable is a (propositional) formula
- \top and \perp are formulas
- If F is a PF, then $\neg F$ is a formula
- If F_1, \dots, F_n are formulas, where $n \geq 2$, then $(F_1 \wedge \dots \wedge F_n)$ and $(F_1 \vee \dots \vee F_n)$ are formulas
- If F and G are formulas, then $(F \rightarrow G)$ and $(F \leftrightarrow G)$ are formulas

Quantified Boolean Formulas (QBFs):

- Every propositional formula is a QBF
- If p is a Boolean variable and F is a QBF, then $\forall p F$ and $\exists p F$ are QBFs

Quantifiers

- \forall is called the *universal quantifier* (symbol)
- \exists is called the *existential quantifier* (symbol)
- $\forall pF$ is read as “for all p , F ”
- $\exists pF$ is read as “there exists p such that F ” or “for some p , F ”

Changing interpretations pointwise

Let \mathcal{I} be an interpretation

Notation:

$$\mathcal{I}[p \mapsto b](q) \stackrel{\text{def}}{=} \begin{cases} \mathcal{I}(q), & \text{if } p \neq q \\ b, & \text{if } p = q \end{cases}$$

Example: $\mathcal{I} = \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}$

$$\mathcal{I}[q \mapsto 1] = \{p \mapsto 1, q \mapsto 1, r \mapsto 1\}$$

$$\mathcal{I}[q \mapsto 0] = \{p \mapsto 1, q \mapsto 0, r \mapsto 1\} = \mathcal{I}$$

$$\mathcal{I}[p \mapsto 0] = \{p \mapsto 0, q \mapsto 0, r \mapsto 1\}$$

Changing interpretations pointwise

Let \mathcal{I} be an interpretation

Notation:

$$\mathcal{I}[p \mapsto b](q) \stackrel{\text{def}}{=} \begin{cases} \mathcal{I}(q), & \text{if } p \neq q \\ b, & \text{if } p = q \end{cases}$$

Example: $\mathcal{I} = \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}$

$$\mathcal{I}[q \mapsto 1] = \{p \mapsto 1, q \mapsto 1, r \mapsto 1\}$$

$$\mathcal{I}[q \mapsto 0] = \{p \mapsto 1, q \mapsto 0, r \mapsto 1\} = \mathcal{I}$$

$$\mathcal{I}[p \mapsto 0] = \{p \mapsto 0, q \mapsto 0, r \mapsto 1\}$$

QBF Semantics

1. $\mathcal{I}(\top) = 1$ and $\mathcal{I}(\perp) = 0$
2. $\mathcal{I}(F_1 \wedge \cdots \wedge F_n) = 1$ iff $\mathcal{I}(F_i) = 1$ for all i
3. $\mathcal{I}(F_1 \vee \cdots \vee F_n) = 1$ iff $\mathcal{I}(F_i) = 1$ for some i
4. $\mathcal{I}(\neg F) = 1$ iff $\mathcal{I}(F) = 0$
5. $\mathcal{I}(F \rightarrow G) = 1$ iff $\mathcal{I}(F) = 0$ or $\mathcal{I}(G) = 1$
6. $\mathcal{I}(F \leftrightarrow G) = 1$ iff $\mathcal{I}(F) = \mathcal{I}(G)$
7. $\mathcal{I}(\forall p F) = 1$ iff $\mathcal{I}[p \mapsto 0](F) = 1$ and $\mathcal{I}[p \mapsto 1](F) = 1$
8. $\mathcal{I}(\exists p F) = 1$ iff $\mathcal{I}[p \mapsto 0](F) = 1$ or $\mathcal{I}[p \mapsto 1](F) = 1$

QBF Semantics

1. $\mathcal{I}(\top) = 1$ and $\mathcal{I}(\perp) = 0$
2. $\mathcal{I}(F_1 \wedge \cdots \wedge F_n) = 1$ iff $\mathcal{I}(F_i) = 1$ for all i
3. $\mathcal{I}(F_1 \vee \cdots \vee F_n) = 1$ iff $\mathcal{I}(F_i) = 1$ for some i
4. $\mathcal{I}(\neg F) = 1$ iff $\mathcal{I}(F) = 0$
5. $\mathcal{I}(F \rightarrow G) = 1$ iff $\mathcal{I}(F) = 0$ or $\mathcal{I}(G) = 1$
6. $\mathcal{I}(F \leftrightarrow G) = 1$ iff $\mathcal{I}(F) = \mathcal{I}(G)$
7. $\mathcal{I}(\forall p F) = 1$ iff $\mathcal{I}[p \mapsto 0](F) = 1$ and $\mathcal{I}[p \mapsto 1](F) = 1$
8. $\mathcal{I}(\exists p F) = 1$ iff $\mathcal{I}[p \mapsto 0](F) = 1$ or $\mathcal{I}[p \mapsto 1](F) = 1$

Evaluating a formula: and-or trees

How to evaluate $\forall p \exists q (p \leftrightarrow q)$ on the interpretation $\{p \mapsto 1, q \mapsto 0\}$

Evaluating a formula: and-or trees

How to evaluate $\forall p \exists q (p \leftrightarrow q)$ on the interpretation $\{ p \mapsto 1, q \mapsto 0 \}$

Notation: Denote any interpretation $\{ p \mapsto b_1, q \mapsto b_2 \}$ by $\mathcal{I}_{b_1 b_2}$

Evaluating a formula: and-or trees

How to evaluate $\forall p \exists q (p \leftrightarrow q)$ on the interpretation $\{ p \mapsto 1, q \mapsto 0 \}$

Notation: Denote any interpretation $\{ p \mapsto b_1, q \mapsto b_2 \}$ by $\mathcal{I}_{b_1 b_2}$

$$\mathcal{I}_{10} \models \forall p \exists q (p \leftrightarrow q)$$

Evaluating a formula: and-or trees

How to evaluate $\forall p \exists q (p \leftrightarrow q)$ on the interpretation $\{p \mapsto 1, q \mapsto 0\}$

Notation: Denote any interpretation $\{p \mapsto b_1, q \mapsto b_2\}$ by $\mathcal{I}_{b_1 b_2}$

$$\mathcal{I}_{10} \models \forall p \exists q (p \leftrightarrow q) \quad \Leftrightarrow \quad \begin{array}{l} \mathcal{I}_{00} \models \exists q (p \leftrightarrow q) \\ \mathcal{I}_{10} \models \exists q (p \leftrightarrow q) \end{array} \text{ and}$$

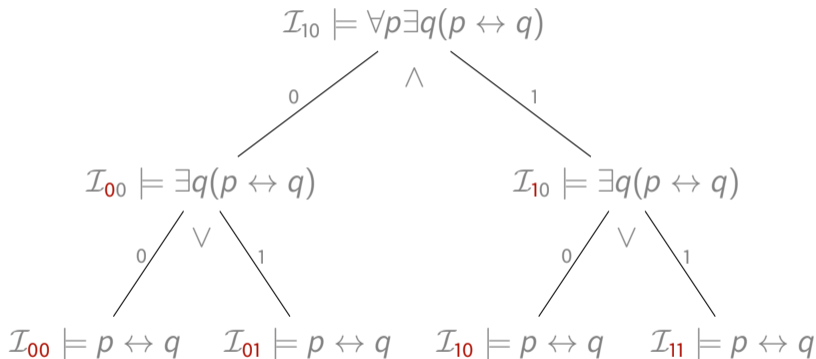
Evaluating a formula: and-or trees

How to evaluate $\forall p \exists q (p \leftrightarrow q)$ on the interpretation $\{p \mapsto 1, q \mapsto 0\}$

Notation: Denote any interpretation $\{p \mapsto b_1, q \mapsto b_2\}$ by $\mathcal{I}_{b_1 b_2}$

$$\begin{aligned} \mathcal{I}_{10} \models \forall p \exists q (p \leftrightarrow q) &\Leftrightarrow \boxed{\begin{array}{l} \mathcal{I}_{00} \models \exists q (p \leftrightarrow q) \\ \mathcal{I}_{10} \models \exists q (p \leftrightarrow q) \end{array} \text{ and}} \\ &\Leftrightarrow \boxed{\begin{array}{l} \boxed{\begin{array}{l} \mathcal{I}_{00} \models p \leftrightarrow q \\ \mathcal{I}_{01} \models p \leftrightarrow q \end{array} \text{ or}} \\ \boxed{\begin{array}{l} \mathcal{I}_{10} \models p \leftrightarrow q \\ \mathcal{I}_{11} \models p \leftrightarrow q \end{array} \text{ or}} \end{array} \text{ and}} \end{aligned}$$

Evaluating a formula: and-or trees



Evaluating a formula

Notation: Denote any interpretation $\{ p \mapsto b_1, q \mapsto b_2 \}$ by $\mathcal{I}_{b_1 b_2}$
Use wildcards $*$ to denote *any* Boolean value

$$\mathcal{I}_{**} \models \forall p \exists q (p \leftrightarrow q)$$

Evaluating a formula

Notation: Denote any interpretation $\{ p \mapsto b_1, q \mapsto b_2 \}$ by $\mathcal{I}_{b_1 b_2}$
Use wildcards $*$ to denote *any* Boolean value

$$\mathcal{I}_{**} \models \forall p \exists q (p \leftrightarrow q) \quad \Leftrightarrow \quad \boxed{\begin{array}{l} \mathcal{I}_{0*} \models \exists q (p \leftrightarrow q) \\ \mathcal{I}_{1*} \models \exists q (p \leftrightarrow q) \end{array} \text{ and}}$$

Evaluating a formula

Notation: Denote any interpretation $\{ p \mapsto b_1, q \mapsto b_2 \}$ by $\mathcal{I}_{b_1 b_2}$
Use wildcards $*$ to denote *any* Boolean value

$$\begin{aligned} \mathcal{I}_{**} \models \forall p \exists q (p \leftrightarrow q) &\Leftrightarrow \boxed{\begin{array}{l} \mathcal{I}_{0*} \models \exists q (p \leftrightarrow q) \\ \mathcal{I}_{1*} \models \exists q (p \leftrightarrow q) \end{array}} \text{ and} \\ &\Leftrightarrow \boxed{\begin{array}{l} \boxed{\begin{array}{l} \mathcal{I}_{00} \models p \leftrightarrow q \\ \mathcal{I}_{01} \models p \leftrightarrow q \end{array}} \text{ or} \\ \boxed{\begin{array}{l} \mathcal{I}_{10} \models p \leftrightarrow q \\ \mathcal{I}_{11} \models p \leftrightarrow q \end{array}} \text{ or} \end{array}} \text{ and} \end{aligned}$$

Evaluating a formula

Notation: Denote any interpretation $\{ p \mapsto b_1, q \mapsto b_2 \}$ by $\mathcal{I}_{b_1 b_2}$
Use wildcards $*$ to denote *any* Boolean value

$$\mathcal{I}_{**} \models \forall p \exists q (p \leftrightarrow q) \quad \Leftrightarrow \quad \begin{array}{l} \mathcal{I}_{0*} \models \exists q (p \leftrightarrow q) \\ \mathcal{I}_{1*} \models \exists q (p \leftrightarrow q) \end{array} \text{ and}$$
$$\Leftrightarrow \quad \begin{array}{l} \begin{array}{l} \mathcal{I}_{00} \models p \leftrightarrow q \\ \mathcal{I}_{01} \models p \leftrightarrow q \end{array} \text{ or} \\ \begin{array}{l} \mathcal{I}_{10} \models p \leftrightarrow q \\ \mathcal{I}_{11} \models p \leftrightarrow q \end{array} \text{ or} \end{array} \text{ and}$$

The variables p and q are *bound* by the quantifiers $\forall p$ and $\exists q$, so
the value of the formula does not depend on the values p and q

Subformula

Propositional formulas:

- F is the immediate subformula of $\neg F$
- F_1, \dots, F_n are the immediate subformulas of $F_1 \wedge \dots \wedge F_n$
- F_1, \dots, F_n are the immediate subformulas of $F_1 \vee \dots \vee F_n$
- F_1 and F_2 are the immediate subformulas of $F_1 \rightarrow F_2$
- F_1 and F_2 are the immediate subformulas of $F_1 \leftrightarrow F_2$
- ...

Subformula

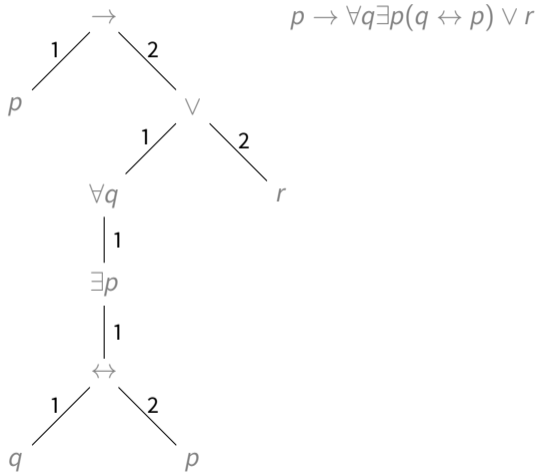
Propositional formulas:

- F is the immediate subformula of $\neg F$
- F_1, \dots, F_n are the immediate subformulas of $F_1 \wedge \dots \wedge F_n$
- F_1, \dots, F_n are the immediate subformulas of $F_1 \vee \dots \vee F_n$
- F_1 and F_2 are the immediate subformulas of $F_1 \rightarrow F_2$
- F_1 and F_2 are the immediate subformulas of $F_1 \leftrightarrow F_2$
- ...

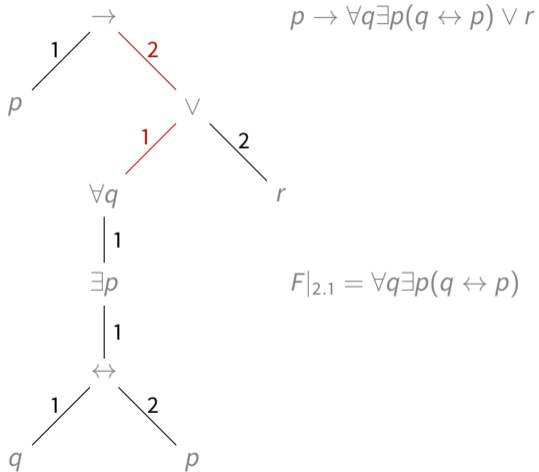
Quantified Boolean formulas:

- F is the immediate subformula of $\forall pF$ and of $\exists pF$

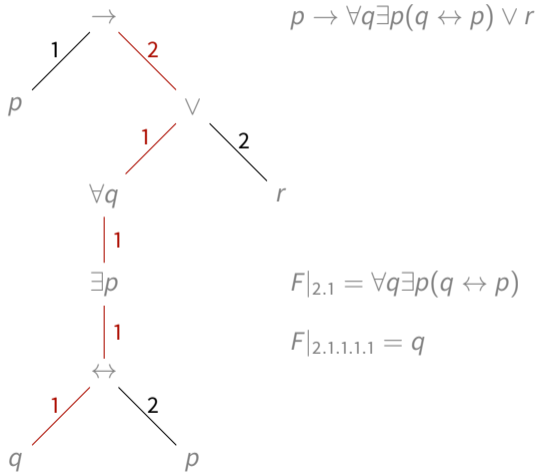
Positions and polarity by example



Positions and polarity by example



Positions and polarity by example



Positions and Polarity

Let $F|_{\pi} = A$

Propositional formulas:

- If A has the form $\neg A_1$,
then $\pi.1$ is a position in F , $F|_{\pi.1} \stackrel{\text{def}}{=} A_1$ and $\text{pol}(F, \pi.1) \stackrel{\text{def}}{=} -\text{pol}(F, \pi)$
- If A has the form $A_1 \wedge \dots \wedge A_n$ or $A_1 \vee \dots \vee A_n$ and $i \in \{1, \dots, n\}$,
then $\pi.i$ is a position in F and $\text{pol}(F, \pi.i) \stackrel{\text{def}}{=} \text{pol}(F, \pi)$
- ...

Positions and Polarity

Let $F|_{\pi} = A$

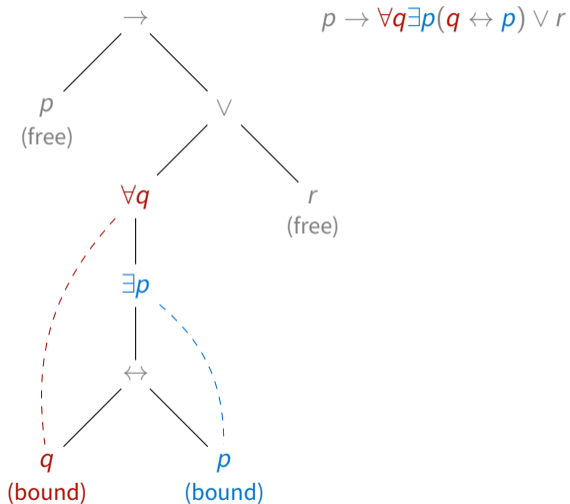
Propositional formulas:

- If A has the form $\neg A_1$,
then $\pi.1$ is a position in F , $F|_{\pi.1} \stackrel{\text{def}}{=} A_1$ and $\text{pol}(F, \pi.1) \stackrel{\text{def}}{=} \neg \text{pol}(F, \pi)$
- If A has the form $A_1 \wedge \dots \wedge A_n$ or $A_1 \vee \dots \vee A_n$ and $i \in \{1, \dots, n\}$,
then $\pi.i$ is a position in F and $\text{pol}(F, \pi.i) \stackrel{\text{def}}{=} \text{pol}(F, \pi)$
- ...

Quantified Boolean formulas:

- If A has the form $\forall p B$ or $\exists p B$,
then $\pi.1$ is a position in F , $F|_{\pi.1} \stackrel{\text{def}}{=} B$ and $\text{pol}(F, \pi.1) \stackrel{\text{def}}{=} \text{pol}(F, \pi)$

Free and bound variables by example



Free and bound occurrences in programs

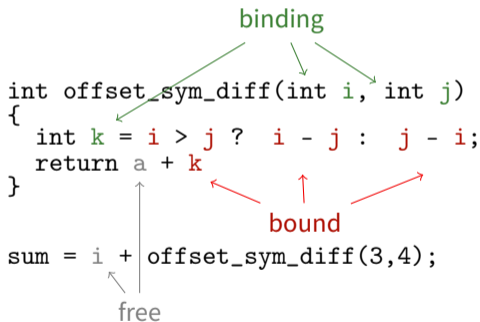
- Free variables in formulas are analogous to global variables in programs
- Bound variables in formulas are analogous to local variables in programs

```
int offset_sym_diff(int i, int j)
{
    int k = i > j ? i - j : j - i;
    return a + k
}
```

```
sum = i + offset_sym_diff(3,4);
```

Free and bound occurrences in programs

- Free variables in formulas are analogous to global variables in programs
- **Bound** variables in formulas are analogous to **local** variables in programs



Free and bound occurrences of variables

Let F be a QBF and p be atom of at position π

The occurrence of p at position π in F is *bound* if π can be represented as a concatenation of two strings $\pi_1\pi_2$ such that $F|_{\pi_1}$ has the form $\forall pG$ or $\exists pG$

Free and bound occurrences of variables

Let F be a QBF and p be atom of at position π

The occurrence of p at position π in F is *bound* if π can be represented as a concatenation of two strings $\pi_1\pi_2$ such that $F|_{\pi_1}$ has the form $\forall pG$ or $\exists pG$

A bound occurrence of p is an occurrence *in the scope of* $\forall p$ or $\exists p$

Free and bound occurrences of variables

Let F be a QBF and p be atom of at position π

The occurrence of p at position π in F is *bound* if π can be represented as a concatenation of two strings $\pi_1\pi_2$ such that $F|_{\pi_1}$ has the form $\forall pG$ or $\exists pG$

A bound occurrence of p is an occurrence *in the scope of* $\forall p$ or $\exists p$

Free occurrence: not bound

Free and bound occurrences of variables

Let F be a QBF and p be atom of at position π

The occurrence of p at position π in F is *bound* if π can be represented as a concatenation of two strings $\pi_1\pi_2$ such that $F|_{\pi_1}$ has the form $\forall pG$ or $\exists pG$

A bound occurrence of p is an occurrence *in the scope of* $\forall p$ or $\exists p$

Free occurrence: not bound

Free (bound) variable of a formula: a variable with at least one free (bound) occurrence

Free and bound occurrences of variables

Let F be a QBF and p be atom of at position π

The occurrence of p at position π in F is *bound* if π can be represented as a concatenation of two strings $\pi_1\pi_2$ such that $F|_{\pi_1}$ has the form $\forall pG$ or $\exists pG$

A bound occurrence of p is an occurrence *in the scope of* $\forall p$ or $\exists p$

Free occurrence: not bound

Free (bound) variable of a formula: a variable with at least one free (bound) occurrence

Closed formula: formula with no free variables

Only free variables matter for truth

The truth value of a QBF formula F depends only on the values of its free variables:

Lemma 1

Suppose $\mathcal{I}_1(p) = \mathcal{I}_2(p)$ for all free variables p of F . Then

$$\mathcal{I}_1 \models F \text{ iff } \mathcal{I}_2 \models F$$

Only free variables matter for truth

The truth value of a QBF formula F depends only on the values of its free variables:

Lemma 1

Suppose $\mathcal{I}_1(p) = \mathcal{I}_2(p)$ for all free variables p of F . Then

$$\mathcal{I}_1 \models F \text{ iff } \mathcal{I}_2 \models F$$

Theorem 2

Let F be a closed formula and let $\mathcal{I}_1, \mathcal{I}_2$ be two interpretations. Then

$$\mathcal{I}_1 \models F \text{ iff } \mathcal{I}_2 \models F$$

Truth, Validity and Satisfiability

Validity and satisfiability are defined as for propositional formulas

Truth, Validity and Satisfiability

Validity and satisfiability are defined as for propositional formulas

There is no difference between these notions for closed formulas:

Lemma 3

For every interpretation \mathcal{I} and closed formula F the following statements are equivalent: (i) $\mathcal{I} \models F$; (ii) F is satisfiable; and (iii) F is valid.

Truth, Validity and Satisfiability

Validity and satisfiability are defined as for propositional formulas

There is no difference between these notions for closed formulas:

Lemma 3

For every interpretation \mathcal{I} and closed formula F the following statements are equivalent: (i) $\mathcal{I} \models F$; (ii) F is satisfiable; and (iii) F is valid.

Satisfiability can be expressed through satisfiability/validity of closed formulas:

Lemma 4

Let F be a formula with free variables p_1, \dots, p_n .

- F is satisfiable iff $\exists p_1 \dots \exists p_n F$ is satisfiable/valid
- F is valid iff the formula $\forall p_1 \dots \forall p_n F$ is satisfiable/valid

Substitutions for propositional formulas

Substitution: F_p^G : denotes the formula obtained from F by replacing all occurrences of the variable p by G

Example:

$$((p \vee s) \wedge (q \rightarrow p))_p^{(l \wedge s)} = ((l \wedge s) \vee s) \wedge (q \rightarrow (l \wedge s))$$

Property: Applying any substitution to a valid formula results in a valid formula

Substitutions for propositional formulas

Substitution: F_p^G : denotes the formula obtained from F by replacing all occurrences of the variable p by G

Example:

$$((p \vee s) \wedge (q \rightarrow p))_p^{(l \wedge s)} = ((l \wedge s) \vee s) \wedge (q \rightarrow (l \wedge s))$$

Property: Applying any substitution to a valid formula results in a valid formula

Substitutions for propositional formulas

Substitution: F_p^G : denotes the formula obtained from F by replacing all occurrences of the variable p by G

Example:

$$((p \vee s) \wedge (q \rightarrow p))_p^{(l \wedge s)} = ((l \wedge s) \vee s) \wedge (q \rightarrow (l \wedge s))$$

Property: Applying **any substitution** to a **valid** formula results in a **valid** formula

Substitution for quantified formulas

Some problems ...

Substitution for quantified formulas

Some problems ...

Consider $\exists q(\neg p \leftrightarrow q)$

Substitution for quantified formulas

Some problems ...

Consider $\exists q(\neg p \leftrightarrow q)$

We cannot simply replace variables by formulas any more:

$\exists(r \rightarrow r)(\neg p \leftrightarrow r \rightarrow r)$???

Substitution for quantified formulas

Some problems ...

Consider $\exists q(\neg p \leftrightarrow q)$

We cannot simply replace variables by formulas any more:

$\exists(r \rightarrow r)(\neg p \leftrightarrow r \rightarrow r)$???

Free variables are **parameters**: we can only substitute for parameters.
But a variable can have both **free** and **bound** occurrences in a formula,
e.g.,

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

Renaming bound variables

Notation: $\exists\forall$: any of \exists, \forall

Renaming bound variables

Notation: $\exists\forall$: any of \exists, \forall

Renaming bound variables in $F[\exists p G]$:

1. Take a *fresh* variable q (i.e., a variable **not occurring** in F)
2. Replace all free occurrences of p in G (not in F !) by q , obtaining G'
3. Consider $F[\exists q G']$

Renaming bound variables

Notation: $\exists\forall$: any of \exists, \forall

Renaming bound variables in $F[\exists p G]$:

1. Take a *fresh* variable q (i.e., a variable **not occurring** in F)
2. Replace all free occurrences of p in G (not in F !) by q , obtaining G'
3. Consider $F[\exists q G']$

Example:

$\exists r(\forall p ((p \rightarrow r) \wedge p)) \vee p$ rename p to q obtaining

$\exists r(\forall q ((q \rightarrow r) \wedge q)) \vee p$

Renaming bound variables

Notation: $\exists\forall$: any of \exists, \forall

Renaming bound variables in $F[\exists p G]$:

1. Take a *fresh* variable q (i.e., a variable **not occurring** in F)
2. Replace all free occurrences of p in G (not in F !) by q , obtaining G'
3. Consider $F[\exists q G']$

Example:

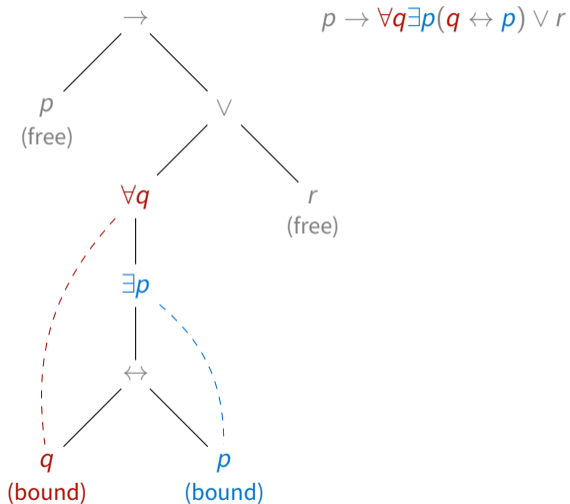
$\exists r(\forall p ((p \rightarrow r) \wedge p)) \vee p$ rename p to q obtaining

$\exists r(\forall q ((q \rightarrow r) \wedge q)) \vee p$

Lemma 5

$$F[\exists p G] \equiv F[\exists q G']$$

Free and bound variables by example



Rectified formulas

Rectified formula F :

1. **no variable** appears **both free and bound** in F
2. for every variable p , there is **at most one** occurrence of quantifier $\exists p$ in F

Any formula can be rectified by renaming its bound variables

We can use the usual notation F_p^G for substitutions into a rectified formula F , assuming p occurs only free in F .

Rectified formulas

Rectified formula F :

1. **no variable** appears **both free and bound** in F
2. for every variable p , there is **at most one** occurrence of quantifier \exists/p in F

Any formula can be rectified by renaming its bound variables

We can use the usual notation F_p^G for substitutions into a rectified formula F , assuming p occurs only free in F .

Rectified formulas

Rectified formula F :

1. **no variable** appears **both free and bound** in F
2. for every variable p , there is **at most one** occurrence of quantifier \exists/p in F

Any formula can be rectified by renaming its bound variables

We can use the usual notation F_p^G for substitutions into a rectified formula F , assuming p occurs **only free** in F

Rectification: Example

$$p \rightarrow \exists p(p \wedge \forall p(p \vee r \rightarrow \neg p)) \Rightarrow$$

$$p \rightarrow \exists p(p \wedge \forall p_1(p_1 \vee r \rightarrow \neg p_1)) \Rightarrow$$

$$p \rightarrow \exists p_2(p_2 \wedge \forall p_1(p_1 \vee r \rightarrow \neg p_1))$$

Rectification: Example

$$p \rightarrow \exists p(p \wedge \forall p(p \vee r \rightarrow \neg p)) \Rightarrow$$

$$p \rightarrow \exists p(p \wedge \forall p_1(p_1 \vee r \rightarrow \neg p_1)) \Rightarrow$$

$$p \rightarrow \exists p_2(p_2 \wedge \forall p_1(p_1 \vee r \rightarrow \neg p_1))$$

Rectification: Example

$$p \rightarrow \exists p(p \wedge \forall p(p \vee r \rightarrow \neg p)) \Rightarrow$$

$$p \rightarrow \exists p(p \wedge \forall p_1(p_1 \vee r \rightarrow \neg p_1)) \Rightarrow$$

$$p \rightarrow \exists p_2(p_2 \wedge \forall p_1(p_1 \vee r \rightarrow \neg p_1))$$

Rectification: Example

$$p \rightarrow \exists p(p \wedge \forall p(p \vee r \rightarrow \neg p)) \Rightarrow$$

$$p \rightarrow \exists p(p \wedge \forall p_1(p_1 \vee r \rightarrow \neg p_1)) \Rightarrow$$

$$p \rightarrow \exists p_2(p_2 \wedge \forall p_1(p_1 \vee r \rightarrow \neg p_1))$$

Another problem

$\exists q(\neg p \leftrightarrow q)$ This formula is **valid** (whatever p is, choose the opposite for q)

substitute p by q

$\exists q(\neg q \leftrightarrow q)$ This formula is **unsatisfiable!**

Substitutions below a quantifier should not lead to **variable capturing**

Another problem

$\exists q(\neg p \leftrightarrow q)$ This formula is **valid** (whatever p is, choose the opposite for q)

substitute p by q

$\exists q(\neg q \leftrightarrow q)$ This formula is **unsatisfiable!**

Substitutions below a quantifier should not lead to **variable capturing**

Another problem

$\exists q(\neg p \leftrightarrow q)$ This formula is **valid** (whatever p is, choose the opposite for q)

substitute p by q

$\exists q(\neg q \leftrightarrow q)$ This formula is **unsatisfiable!**

Substitutions below a quantifier should not lead to variable capturing

Another problem

$\exists q(\neg p \leftrightarrow q)$ This formula is **valid** (whatever p is, choose the opposite for q)

substitute p by q

$\exists q(\neg q \leftrightarrow q)$ This formula is **unsatisfiable!**

Substitutions below a quantifier should not lead to **variable capturing**

Another restriction

Suppose we want to substitute G for p in $F[p]$

Requirement: no free variables in G become bound in F_p^G

(In previous example, $(\exists q(\neg p \leftrightarrow q))_p^G$ does not satisfy this requirement)

Uniform solution: renaming of bound variables

Example:

Since $\exists q(\neg p \leftrightarrow q) \equiv \exists r(\neg p \leftrightarrow r)$

we can use $(\exists r(\neg p \leftrightarrow r))_p^G$ instead of $(\exists q(\neg p \leftrightarrow q))_p^G$

Another restriction

Suppose we want to substitute G for p in $F[p]$

Requirement: no free variables in G become bound in F_p^G

(In previous example, $(\exists q(\neg p \leftrightarrow q))_p^q$ does not satisfy this requirement)

Uniform solution: renaming of bound variables

Example:

Since $\exists q(\neg p \leftrightarrow q) \equiv \exists r(\neg p \leftrightarrow r)$

we can use $(\exists r(\neg p \leftrightarrow r))_p^q$ instead of $(\exists q(\neg p \leftrightarrow q))_p^q$

Another restriction

Suppose we want to substitute G for p in $F[p]$

Requirement: no free variables in G become bound in F_p^G

(In previous example, $(\exists q(\neg p \leftrightarrow q))_p^q$ does not satisfy this requirement)

Uniform solution: renaming of bound variables

Example:

Since $\exists q(\neg p \leftrightarrow q) \equiv \exists r(\neg p \leftrightarrow r)$

we can use $(\exists r(\neg p \leftrightarrow r))_p^q$ instead of $(\exists q(\neg p \leftrightarrow q))_p^q$

Another restriction

Suppose we want to substitute G for p in $F[p]$

Requirement: no free variables in G become bound in F_p^G

(In previous example, $(\exists q(\neg p \leftrightarrow q))_p^q$ does not satisfy this requirement)

Uniform

From now on, we always assume that:

Example

1. formulas are **rectified**
2. all substitutions **satisfy the requirement** above

Since \exists
we can

Equivalent replacement

Lemma 6

Let \mathcal{I} be an interpretation and $\mathcal{I} \models F_1 \leftrightarrow F_2$. Then $\mathcal{I} \models G[F_1] \leftrightarrow G[F_2]$.

Theorem 7 (Equivalent Replacement)

Let $F_1 \equiv F_2$. Then $G[F_1] \equiv G[F_2]$.

Equivalent replacement

Lemma 6

Let \mathcal{I} be an interpretation and $\mathcal{I} \models F_1 \leftrightarrow F_2$. Then $\mathcal{I} \models G[F_1] \leftrightarrow G[F_2]$.

Theorem 7 (Equivalent Replacement)

Let $F_1 \equiv F_2$. Then $G[F_1] \equiv G[F_2]$.

More equivalences

Theorem 8

The following holds for all QBFs F :

1. $\forall p_1 \forall p_2 F \equiv \forall p_2 \forall p_1 F$
2. $\exists p_1 \exists p_2 F \equiv \exists p_2 \exists p_1 F$
3. $\exists \forall p F \equiv F$ if p does not occur free in F

Note: In general, $\exists p_1 \forall p_2 F \not\equiv \forall p_2 \exists p_1 F$

Example:

- $\forall p \exists q (p \leftrightarrow q) \equiv \top$

More equivalences

Theorem 8

The following holds for all QBFs F :

1. $\forall p_1 \forall p_2 F \equiv \forall p_2 \forall p_1 F$
2. $\exists p_1 \exists p_2 F \equiv \exists p_2 \exists p_1 F$
3. $\exists \forall p F \equiv F$ if p does not occur free in F

Note: In general, $\exists p_1 \forall p_2 F \not\equiv \forall p_2 \exists p_1 F$

Example:

- $\forall p \exists q (p \leftrightarrow q) \equiv \top$

More equivalences

Theorem 8

The following holds for all QBFs F :

1. $\forall p_1 \forall p_2 F \equiv \forall p_2 \forall p_1 F$
2. $\exists p_1 \exists p_2 F \equiv \exists p_2 \exists p_1 F$
3. $\exists \forall p F \equiv F$ if p does not occur free in F

Note: In general, $\exists p_1 \forall p_2 F \not\equiv \forall p_2 \exists p_1 F$

Example:

- $\forall p \exists q (p \leftrightarrow q) \equiv \top$
- $\exists q \forall p (p \leftrightarrow q) \equiv \perp$

More equivalences

Theorem 8

The following holds for all QBFs F :

1. $\forall p_1 \forall p_2 F \equiv \forall p_2 \forall p_1 F$
2. $\exists p_1 \exists p_2 F \equiv \exists p_2 \exists p_1 F$
3. $\exists \forall p F \equiv F$ if p does not occur free in F

Note: In general, $\exists p_1 \forall p_2 F \not\equiv \forall p_2 \exists p_1 F$

Example:

- $\forall p \exists q (p \leftrightarrow q) \equiv \top$
- $\exists q \forall p (p \leftrightarrow q) \equiv \perp$

Prenex form

Quantifier-free formula: no quantifiers (that is, propositional)

Prenex formula: formula of the form

$$\underbrace{\exists v_1 p_1 \cdots \exists v_n p_n}_{\text{quantifier prefix}} \underbrace{G}_{\text{matrix}}$$

with G quantifier-free

Outermost prefix of $\exists v_1 p_1 \cdots \exists v_n p_n G$: the longest subsequence $\exists v_1 p_1 \cdots \exists v_k p_k$ of $\exists v_1 p_1 \cdots \exists v_n p_n$ such that $\exists v_1 = \cdots = \exists v_k$

Prenex form

Quantifier-free formula: no quantifiers (that is, propositional)

Prenex formula: formula of the form

$$\overbrace{\exists \forall_1 p_1 \cdots \exists \forall_n p_n}^{\text{quantifier prefix}} \underbrace{G}_{\text{matrix}}$$

with G quantifier-free

Outermost prefix of $\exists \forall_1 p_1 \cdots \exists \forall_n p_n G$: the longest subsequence $\exists \forall_1 p_1 \cdots \exists \forall_k p_k$ of $\exists \forall_1 p_1 \cdots \exists \forall_n p_n$ such that $\exists \forall_1 = \cdots = \exists \forall_k$

Prenex form

Quantifier-free formula: no quantifiers (that is, propositional)

Prenex formula: formula of the form

$$\overbrace{\exists \forall_1 p_1 \cdots \exists \forall_n p_n}^{\text{quantifier prefix}} \underbrace{G}_{\text{matrix}}$$

with G quantifier-free

Outermost prefix of $\exists \forall_1 p_1 \cdots \exists \forall_n p_n G$: the longest subsequence $\exists \forall_1 p_1 \cdots \exists \forall_k p_k$ of $\exists \forall_1 p_1 \cdots \exists \forall_n p_n$ such that $\exists \forall_1 = \cdots = \exists \forall_k$

Prenex form

Quantifier-free formula: no quantifiers (that is, propositional)

Prenex formula: formula of the form

$$\overbrace{\exists v_1 p_1 \cdots \exists v_n p_n}^{\text{quantifier prefix}} \underbrace{G}_{\text{matrix}}$$

with G quantifier-free

Outermost prefix of $\exists v_1 p_1 \cdots \exists v_n p_n G$: the longest subsequence $\exists v_1 p_1 \cdots \exists v_k p_k$ of $\exists v_1 p_1 \cdots \exists v_n p_n$ such that $\exists v_1 = \cdots = \exists v_k$

Example

- outermost prefix of $\forall p \forall q \exists r (r \wedge p \rightarrow q)$: $\forall p \forall q$
- outermost prefix of $\exists p \forall q \exists r (r \wedge p \rightarrow q)$: $\exists p$

Prenex form

Quantifier-free formula: no quantifiers (that is, propositional)

Prenex formula: formula of the form

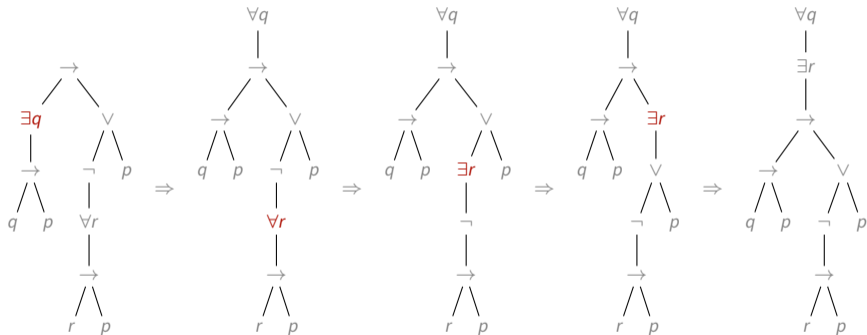
$$\overbrace{\exists v_1 p_1 \cdots \exists v_n p_n}^{\text{quantifier prefix}} \underbrace{G}_{\text{matrix}}$$

with G quantifier-free

Outermost prefix of $\exists v_1 p_1 \cdots \exists v_n p_n G$: the longest subsequence $\exists v_1 p_1 \cdots \exists v_k p_k$ of $\exists v_1 p_1 \cdots \exists v_n p_n$ such that $\exists v_1 = \cdots = \exists v_k$

A formula F is a *prenex form* of a formula G if F is prenex and $F \equiv G$

Conversion to prenex form, Example I



Conversion to prenex form, Example I

$$\exists q(q \rightarrow p) \rightarrow \neg \forall r(r \rightarrow p) \vee p \quad \Rightarrow$$

$$\forall q((q \rightarrow p) \rightarrow \neg \forall r(r \rightarrow p) \vee p) \quad \Rightarrow$$

$$\forall q((q \rightarrow p) \rightarrow \exists r \neg(r \rightarrow p) \vee p) \quad \Rightarrow$$

$$\forall q((q \rightarrow p) \rightarrow \exists r(\neg(r \rightarrow p) \vee p)) \quad \Rightarrow$$

$$\forall q \exists r((q \rightarrow p) \rightarrow \neg(r \rightarrow p) \vee p)$$

Prenexing rules

$$(\exists p F_1) \wedge \cdots \wedge F_n \Rightarrow \exists p (F_1 \wedge \cdots \wedge F_n)$$

$$(\exists p F_1) \vee \cdots \vee F_n \Rightarrow \exists p (F_1 \vee \cdots \vee F_n)$$

$$(\forall p F_1) \rightarrow F_2 \Rightarrow \exists p (F_1 \rightarrow F_2) \quad F_1 \rightarrow (\exists p F_2) \Rightarrow \exists p (F_1 \rightarrow F_2)$$

$$(\exists p F_1) \rightarrow F_2 \Rightarrow \forall p (F_1 \rightarrow F_2) \quad F_1 \rightarrow (\forall p F_2) \Rightarrow \forall p (F_1 \rightarrow F_2)$$

$$\neg \forall p F \Rightarrow \exists p \neg F$$

$$\neg \exists p F \Rightarrow \forall p \neg F$$

Conversion to prenex form, Example II

$$\exists q(q \rightarrow p) \rightarrow \neg \forall r(r \rightarrow p) \vee p \quad \Rightarrow$$

$$\exists q(q \rightarrow p) \rightarrow \exists r \neg(r \rightarrow p) \vee p \quad \Rightarrow$$

$$\exists q(q \rightarrow p) \rightarrow \exists r(\neg(r \rightarrow p) \vee p) \quad \Rightarrow$$

$$\exists r(\exists q(q \rightarrow p) \rightarrow \neg(r \rightarrow p) \vee p) \quad \Rightarrow$$

$$\exists r \forall q((q \rightarrow p) \rightarrow \neg(r \rightarrow p) \vee p)$$

Checking the satisfiability of QBFs

The algorithms for propositional satisfiability or validity can be adapted to QBF

We will see:

- Splitting
- DPLL

Recall:

1. $F(p_1, \dots, p_n)$ is satisfiable iff $\exists p_1 \dots \exists p_n F(p_1, \dots, p_n)$ is satisfiable
2. $F(p_1, \dots, p_n)$ is valid iff $\forall p_1 \dots \forall p_n F(p_1, \dots, p_n)$ is satisfiable
3. A closed QBF is either always true (valid) or false (unsatisfiable)

The algorithms will check whether a closed formula is valid or unsatisfiable

Checking the satisfiability of QBFs

The algorithms for propositional satisfiability or validity can be adapted to QBF
We will see:

- Splitting
- DPLL

Recall:

1. $F(p_1, \dots, p_n)$ is **satisfiable** iff $\exists p_1 \dots \exists p_n F(p_1, \dots, p_n)$ is **satisfiable**
2. $F(p_1, \dots, p_n)$ is **valid** iff $\forall p_1 \dots \forall p_n F(p_1, \dots, p_n)$ is **satisfiable**
3. A **closed QBF** is either **always true** (valid) or **false** (unsatisfiable)

The algorithms will check whether a closed formula is valid or unsatisfiable

Checking the satisfiability of QBFs

The algorithms for propositional satisfiability or validity can be adapted to QBF
We will see:

- Splitting
- DPLL

Recall:

1. $F(p_1, \dots, p_n)$ is **satisfiable** iff $\exists p_1 \dots \exists p_n F(p_1, \dots, p_n)$ is **satisfiable**
2. $F(p_1, \dots, p_n)$ is **valid** iff $\forall p_1 \dots \forall p_n F(p_1, \dots, p_n)$ is **satisfiable**
3. A **closed QBF** is either **always true** (valid) or **false** (unsatisfiable)

The algorithms will check whether a **closed formula is valid or unsatisfiable**

Splitting: foundations

Lemma 9

- A closed formula $\forall p F$ evaluates to *true* iff *both* F_p^\perp *and* F_p^\top evaluate to true.
- A closed formula $\exists p F$ evaluates to *true* iff *either* F_p^\perp *or* F_p^\top evaluates to true.

Splitting

Simplification rules for \top :

$$\neg\top \Rightarrow \perp$$

$$\top \wedge F_1 \wedge \dots \wedge F_n \Rightarrow F_1 \wedge \dots \wedge F_n$$

$$\top \vee F_1 \vee \dots \vee F_n \Rightarrow \top$$

$$F \rightarrow \top \Rightarrow \top \quad \top \rightarrow F \Rightarrow F$$

$$F \leftrightarrow \top \Rightarrow F \quad \top \leftrightarrow F \Rightarrow F$$

Simplification rules for \perp :

$$\neg\perp \Rightarrow \top$$

$$\perp \wedge F_1 \wedge \dots \wedge F_n \Rightarrow \perp$$

$$\perp \vee F_1 \vee \dots \vee F_n \Rightarrow F_1 \vee \dots \vee F_n$$

$$F \rightarrow \perp \Rightarrow \neg F \quad \perp \rightarrow F \Rightarrow \top$$

$$F \leftrightarrow \perp \Rightarrow \neg F \quad \perp \leftrightarrow F \Rightarrow \neg F$$

Splitting

Simplification rules for \top :

$$\neg\top \Rightarrow \perp$$

$$\top \wedge F_1 \wedge \dots \wedge F_n \Rightarrow F_1 \wedge \dots \wedge F_n$$

$$\top \vee F_1 \vee \dots \vee F_n \Rightarrow \top$$

$$F \rightarrow \top \Rightarrow \top \quad \top \rightarrow F \Rightarrow F$$

$$F \leftrightarrow \top \Rightarrow F \quad \top \leftrightarrow F \Rightarrow F$$

$$\forall p \top \Rightarrow \top$$

$$\exists p \top \Rightarrow \top$$

Simplification rules for \perp :

$$\neg\perp \Rightarrow \top$$

$$\perp \wedge F_1 \wedge \dots \wedge F_n \Rightarrow \perp$$

$$\perp \vee F_1 \vee \dots \vee F_n \Rightarrow F_1 \vee \dots \vee F_n$$

$$F \rightarrow \perp \Rightarrow \neg F \quad \perp \rightarrow F \Rightarrow \top$$

$$F \leftrightarrow \perp \Rightarrow \neg F \quad \perp \leftrightarrow F \Rightarrow \neg F$$

$$\forall p \perp \Rightarrow \perp$$

$$\exists p \perp \Rightarrow \perp$$

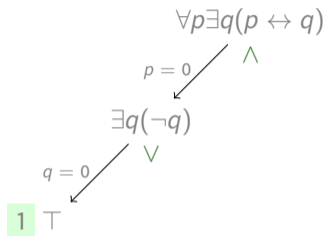
Splitting, Example

$$\forall p \exists q (p \leftrightarrow q)$$

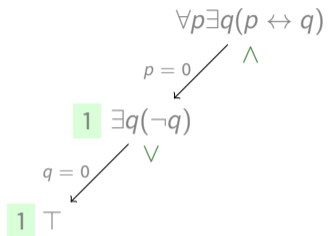
Splitting, Example

$$\begin{array}{l} \forall p \exists q (p \leftrightarrow q) \\ \swarrow \quad \wedge \\ p = 0 \\ \searrow \\ \exists q (\neg q) \end{array}$$

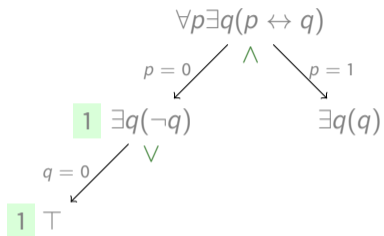
Splitting, Example



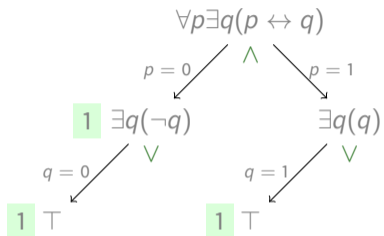
Splitting, Example



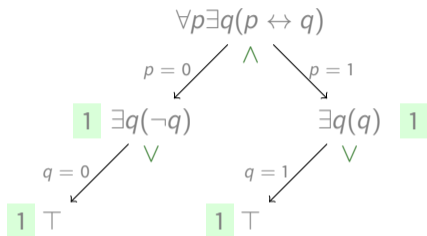
Splitting, Example



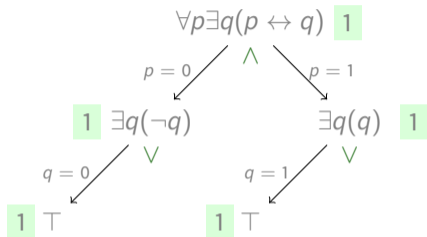
Splitting, Example



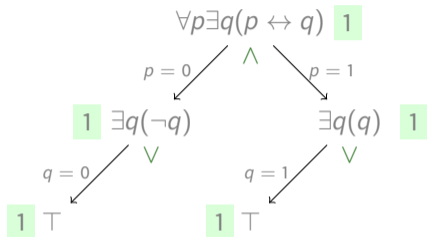
Splitting, Example



Splitting, Example

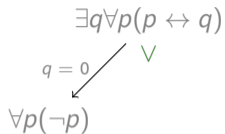
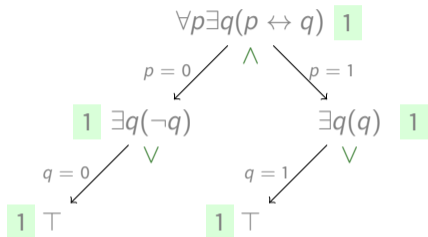


Splitting, Example

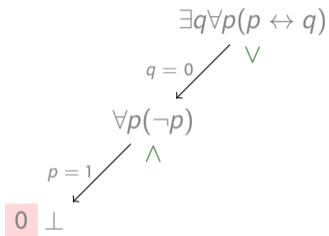
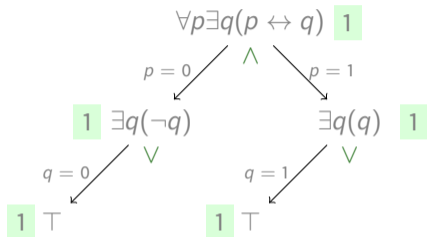


$\exists q \forall p (p \leftrightarrow q)$

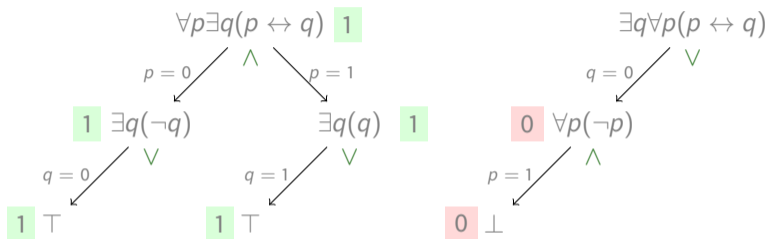
Splitting, Example



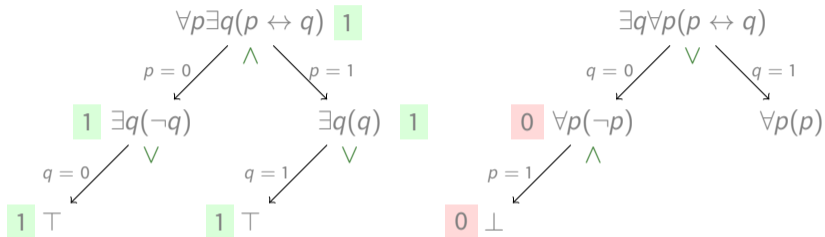
Splitting, Example



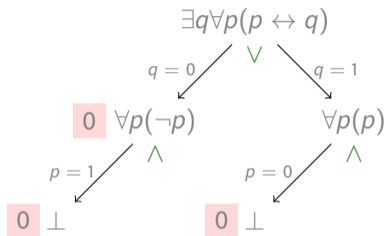
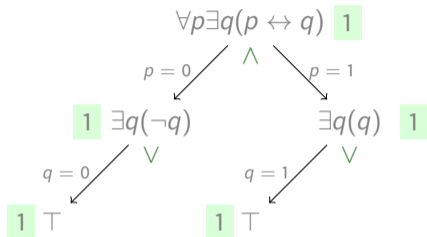
Splitting, Example



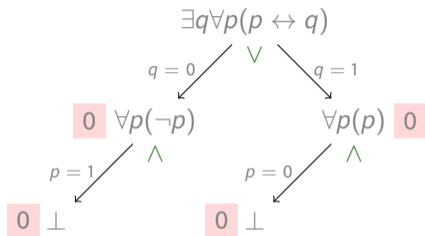
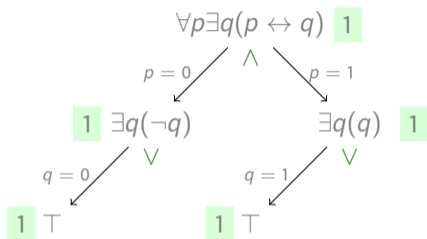
Splitting, Example



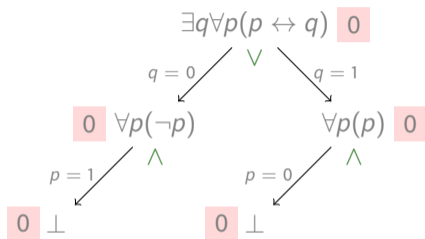
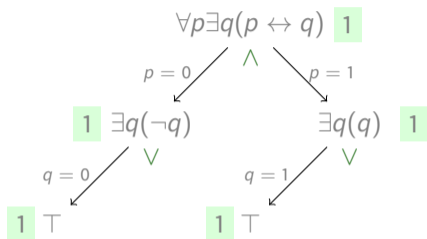
Splitting, Example



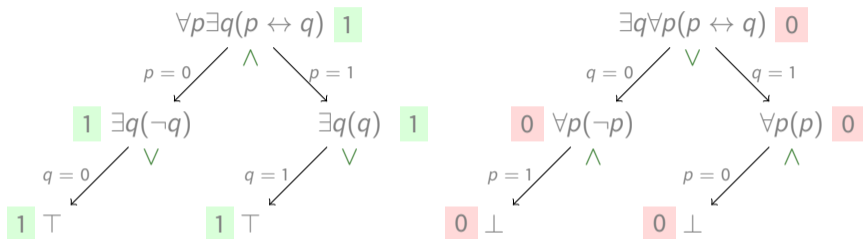
Splitting, Example



Splitting, Example



Splitting, Example



To minimize search the selection of variable values is best seen as a two-player game:

by selecting a value for $\exists q$ one is trying to make the formula true,
by selecting a value for $\forall p$ one is trying to make the formula false

Splitting algorithm

Notation: if $\mathbf{p} = (p_1, \dots, p_k)$ then $\exists \mathbf{p} F$ denotes $\exists p_1 \dots \exists p_k F$

Splitting algorithm

procedure *splitting*(F)

input: closed rectified prenex formula F

output: 0 or 1

parameters: function *select_variable_value* // selects a variable from the outermost prefix
// of F as well as a Boolean value for it

$F := \text{simplify}(F)$ // apply extended simplification rules to completion

if $F = \perp$ then return 0

if $F = \top$ then return 1

// else F has the form $\exists \forall p F'$ where p is F 's outermost prefix

$(p, b) := \text{select_variable_value}(F)$

Let G be obtained from F by deleting p from p

if $b = 0$ then $A := \perp$; $B := \top$ else $A := \top$; $B := \perp$

$b := \text{splitting}(G_p^A)$

case $(b, \exists \forall)$ of

$(0, \forall) \Rightarrow$ return 0

$(0, \exists) \Rightarrow$ return $\text{splitting}(G_p^B)$

$(1, \forall) \Rightarrow$ return $\text{splitting}(G_p^B)$

$(1, \exists) \Rightarrow$ return 1

end

Conjunctive Normal Form

For [more efficient algorithms](#) we need QBFs to be in a convenient normal form

Conjunctive Normal Form

For **more efficient algorithms** we need QBFs to be in a convenient normal form

Our next aim is to modify **CNF** and **DPLL** to deal with quantified Boolean formulas

Conjunctive Normal Form

For [more efficient algorithms](#) we need QBFs to be in a convenient normal form

Our next aim is to modify **CNF** and **DPLL** to deal with quantified Boolean formulas

A quantified Boolean formula F is in *Conjunctive Normal Form (CNF)*, if

- it is either \perp , or \top , or
- it has the form

$$\exists \forall_1 p_1 \cdots \exists \forall_n p_n (C_1 \wedge \cdots \wedge C_m)$$

where C_1, \dots, C_m are clauses

Conjunctive Normal Form

For **more efficient algorithms** we need QBFs to be in a convenient normal form

Our next aim is to modify **CNF** and **DPLL** to deal with quantified Boolean formulas

A quantified Boolean formula F is in **Conjunctive Normal Form (CNF)**, if

- it is either \perp , or \top , or
- it has the form

$$\exists v_1 p_1 \cdots \exists v_n p_n (C_1 \wedge \cdots \wedge C_m)$$

where C_1, \dots, C_m are clauses

Example:

$$\forall p \exists q \exists s ((\neg p \vee s \vee q) \wedge (s \vee \neg q) \wedge \neg s)$$

CNF rules

Prenexing rules

+

propositional CNF rules:

$$\begin{aligned} F \leftrightarrow G &\Rightarrow (\neg F \vee G) \wedge (\neg G \vee F) \\ F \rightarrow G &\Rightarrow \neg F \vee G \\ \neg(F \wedge G) &\Rightarrow \neg F \vee \neg G \\ \neg(F \vee G) &\Rightarrow \neg F \wedge \neg G \\ \neg\neg F &\Rightarrow F \\ (F_1 \wedge \dots \wedge F_m) \vee G_1 \vee \dots \vee G_n &\Rightarrow \begin{array}{l} (F_1 \vee G_1 \vee \dots \vee G_n) \quad \wedge \\ \dots \quad \wedge \\ (F_m \vee G_1 \vee \dots \vee G_n) \end{array} \end{aligned}$$

DPLL for quantified boolean formulas

Input:

Q : quantifier sequence $\exists \forall_1 \mathbf{p}_1 \cdots \exists \forall_n \mathbf{p}_n$

S : set of clauses with variables from $\mathbf{p}_1, \dots, \mathbf{p}_n$

Main components:

Unit propagation

Splitting on literals

Unit Propagation

Q : quantifier sequence

S : current clause set

Propositional formulas:

For each **unit clause** L in S

1. remove all clauses containing literal L from S
2. remove every literal \bar{L} from remaining clauses

Quantified Boolean formulas:

For each **unit clause** L in S of the form p or $\neg p$

- If Q does not contain p or contains $\exists p$,
 1. remove all clauses containing literal L from S
 2. remove every literal \bar{L} from remaining clauses
- otherwise (Q contains $\forall p$), add \square to S

Unit Propagation

Q : quantifier sequence

S : current clause set

Propositional formulas:

For each **unit clause** L in S

1. remove all clauses containing literal L from S
2. remove every literal \bar{L} from remaining clauses

Quantified Boolean formulas:

For each **unit clause** L in S of the form p or $\neg p$

- If Q does not contain p or contains $\exists p$,
 1. remove all clauses containing literal L from S
 2. remove every literal \bar{L} from remaining clauses
- otherwise (Q contains $\forall p$), add \square to S

Unit Propagation

Q : quantifier sequence

S : current clause set

Propositional formulas:

For each **unit clause** L in S

1. remove all clauses containing literal L from S
2. remove every literal \bar{L} from remaining clauses

Quantified Boolean formulas:

For each **unit clause** L in S of the form p or $\neg p$

- If Q does not contain p or contains $\exists p$,
 1. remove all clauses containing literal L from S
 2. remove every literal \bar{L} from remaining clauses
- otherwise (Q contains $\forall p$), add \square to S

Unit Propagation

Q : quantifier sequence

S : current clause set

Propositional formulas:

For each **unit clause** L in S

1. remove all clauses containing literal L from S
2. remove every literal \bar{L} from remaining clauses

Quantified Boolean formulas:

For each **unit clause** L in S of the form p or $\neg p$

- If Q does not contain p or contains $\exists p$,
 1. remove all clauses containing literal L from S
 2. remove every literal \bar{L} from remaining clauses
- otherwise (Q contains $\forall p$), add \square to S

DPLL algorithm

Why do we add \square to S when Q is $\forall p \exists_1 p_1 \cdots \exists_m p_m$ and
 S is $\{p, C_1, \dots, C_n\}$ (or $\{\neg p, C_1, \dots, C_n\}$)?

DPLL algorithm

Why do we add \square to S when Q is $\forall p \exists_1 p_1 \cdots \exists_m p_m$ and
 S is $\{p, C_1, \dots, C_n\}$ (or $\{\neg p, C_1, \dots, C_n\}$)?

Because

1. The intended input formula is

$$G = \forall p \exists_1 q_1 \cdots \exists_m q_m (p \wedge C_1 \wedge \cdots \wedge C_m)$$

2. $G \equiv \exists_1 q_1 \cdots \exists_m q_m ((p \wedge C_1 \wedge \cdots \wedge C_m)_p^\perp \wedge (p \wedge C_1 \wedge \cdots \wedge C_m)_p^T)$

DPLL algorithm

Why do we add \square to S when Q is $\forall p \exists_1 p_1 \cdots \exists_m p_m$ and
 S is $\{p, C_1, \dots, C_n\}$ (or $\{\neg p, C_1, \dots, C_n\}$)?

Because

1. The intended input formula is

$$G = \forall p \exists_1 q_1 \cdots \exists_m q_m (p \wedge C_1 \wedge \cdots \wedge C_m)$$

2. $G \equiv \exists_1 q_1 \cdots \exists_m q_m ((p \wedge C_1 \wedge \cdots \wedge C_m) \perp_p \wedge (p \wedge C_1 \wedge \cdots \wedge C_m) \top_p)$

DPLL algorithm

Why do we add \square to S when Q is $\forall p \exists_1 p_1 \cdots \exists_m p_m$ and
 S is $\{p, C_1, \dots, C_n\}$ (or $\{\neg p, C_1, \dots, C_n\}$)?

Because

1. The intended input formula is

$$G = \forall p \exists_1 q_1 \cdots \exists_m q_m (p \wedge C_1 \wedge \cdots \wedge C_m)$$

2. $G \equiv \exists_1 q_1 \cdots \exists_m q_m ((p \wedge C_1 \wedge \cdots \wedge C_m)_p^\perp \wedge (p \wedge C_1 \wedge \cdots \wedge C_m)_p^\top)$
 $= \exists_1 q_1 \cdots \exists_m q_m (\perp \wedge (C_1 \wedge \cdots \wedge C_m)_p^\perp \wedge (p \wedge C_1 \wedge \cdots \wedge C_m)_p^\top)$

DPLL algorithm

Why do we add \square to S when Q is $\forall p \exists_1 p_1 \cdots \exists_m p_m$ and
 S is $\{p, C_1, \dots, C_n\}$ (or $\{\neg p, C_1, \dots, C_n\}$)?

Because

1. The intended input formula is

$$G = \forall p \exists_1 q_1 \cdots \exists_m q_m (p \wedge C_1 \wedge \cdots \wedge C_m)$$

2. $G \equiv \exists_1 q_1 \cdots \exists_m q_m ((p \wedge C_1 \wedge \cdots \wedge C_m)_p^\perp \wedge (p \wedge C_1 \wedge \cdots \wedge C_m)_p^\top)$
 $\equiv \exists_1 q_1 \cdots \exists_m q_m (\perp \wedge (C_1 \wedge \cdots \wedge C_m)_p^\perp \wedge (p \wedge C_1 \wedge \cdots \wedge C_m)_p^\top)$
 $\equiv \exists_1 q_1 \cdots \exists_m q_m \perp$
 $\equiv \perp$

DPLL algorithm

Why do we add \square to S when Q is $\forall p \exists_1 p_1 \cdots \exists_m p_m$ and
 S is $\{p, C_1, \dots, C_n\}$ (or $\{\neg p, C_1, \dots, C_n\}$)?

Because

1. The intended input formula is

$$G = \forall p \exists_1 q_1 \cdots \exists_m q_m (p \wedge C_1 \wedge \cdots \wedge C_m)$$

2. $G \equiv \exists_1 q_1 \cdots \exists_m q_m ((p \wedge C_1 \wedge \cdots \wedge C_m)_p^\perp \wedge (p \wedge C_1 \wedge \cdots \wedge C_m)_p^\top)$
 $\equiv \exists_1 q_1 \cdots \exists_m q_m (\perp \wedge (C_1 \wedge \cdots \wedge C_m)_p^\perp \wedge (p \wedge C_1 \wedge \cdots \wedge C_m)_p^\top)$
 $\equiv \exists_1 q_1 \cdots \exists_m q_m \perp$
 $\equiv \perp$

DPLL algorithm

Why do we add \square to S when Q is $\forall p \exists_1 p_1 \cdots \exists_m p_m$ and
 S is $\{p, C_1, \dots, C_n\}$ (or $\{\neg p, C_1, \dots, C_n\}$)?

Alternatively, using the **game** metaphor, because

the \forall -**player** wants to falsify the formula

DPLL algorithm

Why do we add \square to S when Q is $\forall p \exists_1 p_1 \cdots \exists_m p_m$ and
 S is $\{p, C_1, \dots, C_n\}$ (or $\{\neg p, C_1, \dots, C_n\}$)?

Alternatively, using the **game** metaphor, because

the \forall -**player wants to falsify** the formula

Winning move for the \forall -player:

select the value for p that falsifies the unit clause p , and hence the whole CNF

DPLL algorithm

Why do we add \square to S when Q is $\forall p \exists_1 p_1 \cdots \exists_m p_m$ and
 S is $\{ p, C_1, \dots, C_n \}$ (or $\{ \neg p, C_1, \dots, C_n \}$)?

Alternatively, using the **game** metaphor, because

the \forall -**player wants to falsify** the formula

Winning move for the \forall -player:

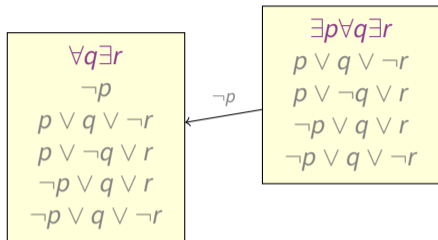
select the value for p that falsifies the unit clause p , and hence the whole CNF

(argument is similar for $\{ \neg p, C_1, \dots, C_n \}$)

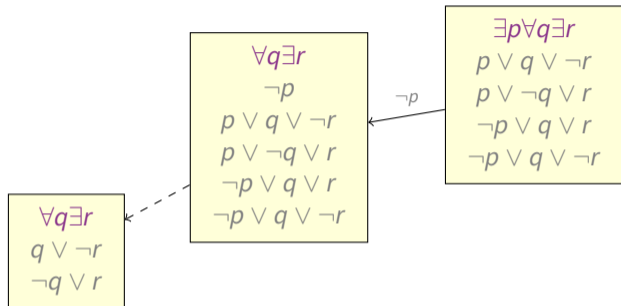
DPLL, Example

$$\exists p \forall q \exists r$$
$$p \vee q \vee \neg r$$
$$p \vee \neg q \vee r$$
$$\neg p \vee q \vee r$$
$$\neg p \vee q \vee \neg r$$

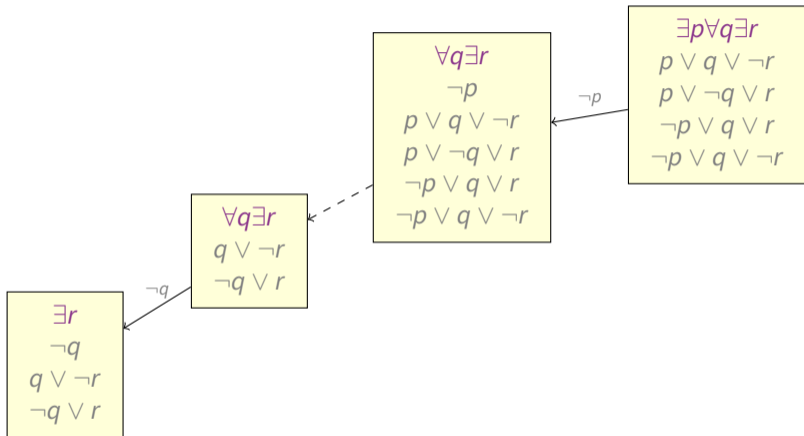
DPLL, Example



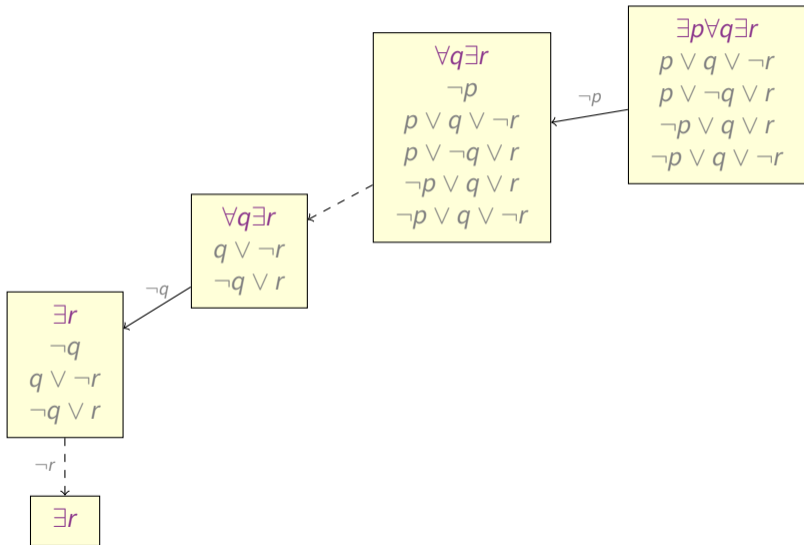
DPLL, Example



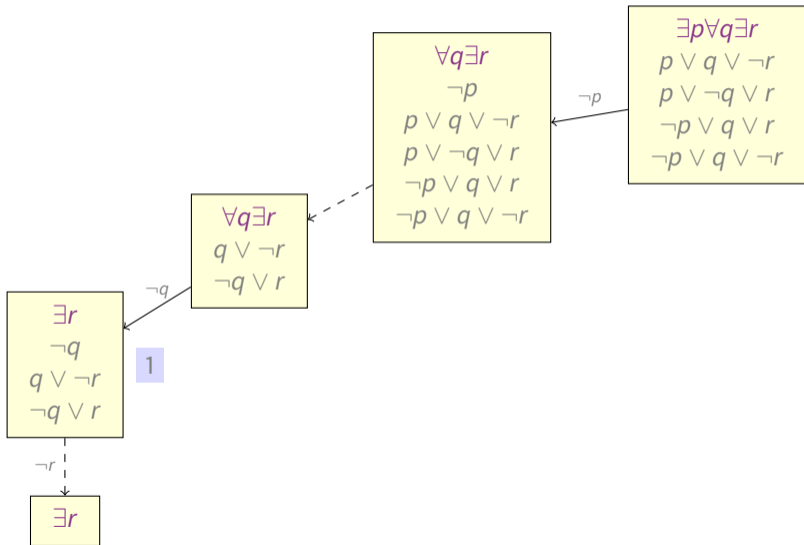
DPLL, Example



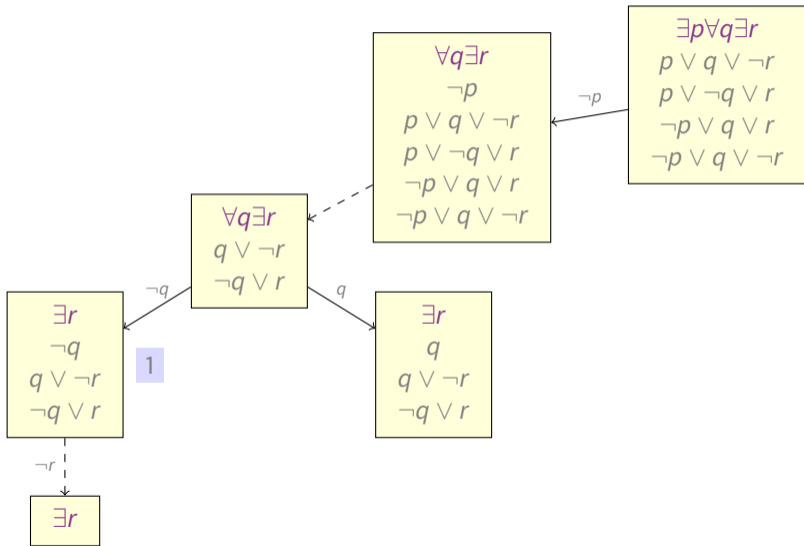
DPLL, Example



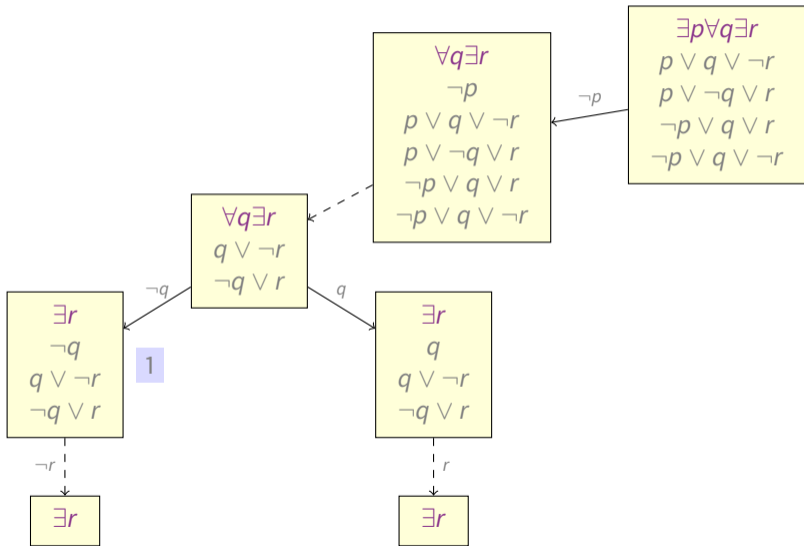
DPLL, Example



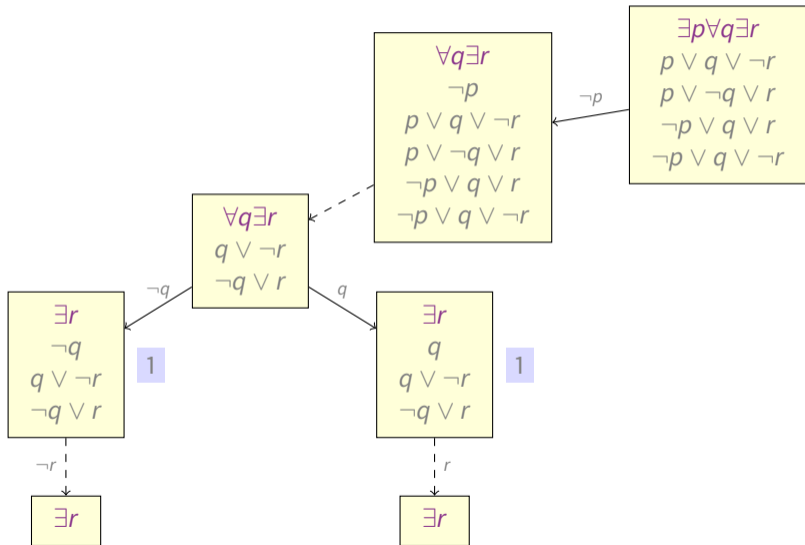
DPLL, Example



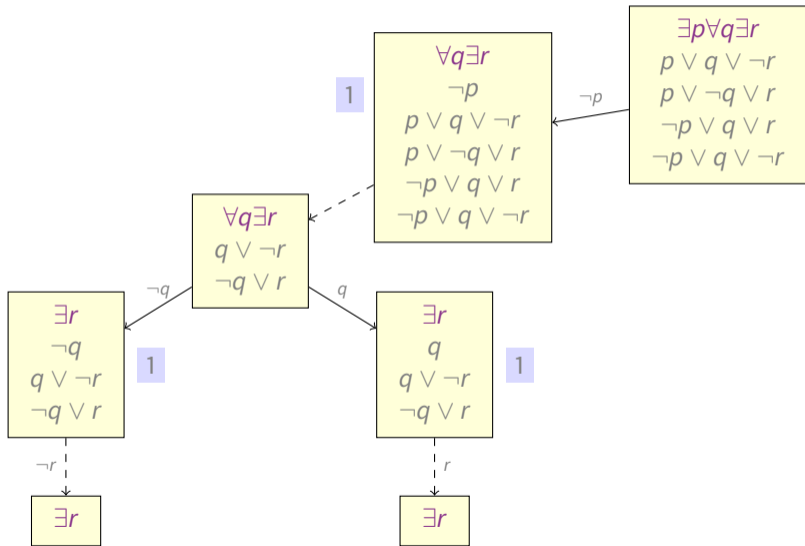
DPLL, Example



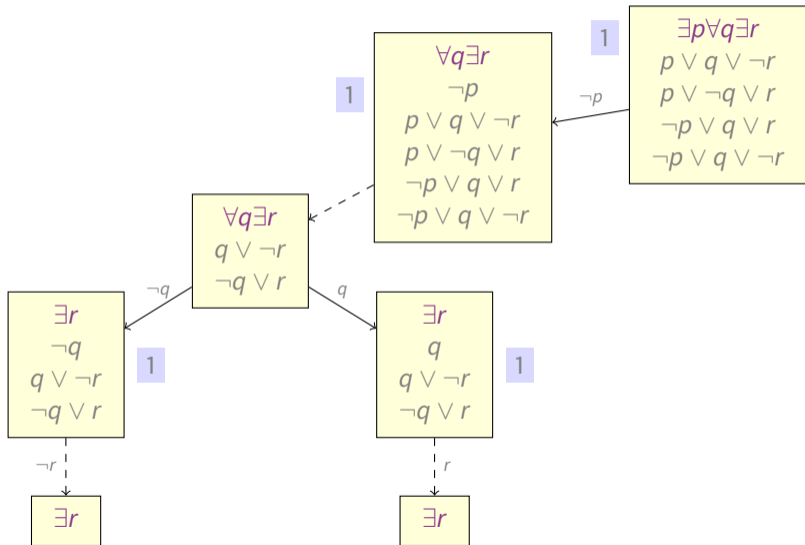
DPLL, Example



DPLL, Example



DPLL, Example



DPLL algorithm

procedure $DPLL(Q, S)$

input: quantifier sequence $Q = \exists \forall_1 p_1 \cdots \exists \forall_n p_n$,
clause set S with vars from Q

output: 0 or 1

parameters: function $select_variable_value$

begin

$S := unit_propagate(Q, S)$

if S is empty **then return** 1

if S contains \square **then return** 0

$(p, b) := select_variable_value(p_1, S)$

Let Q' be obtained from Q by deleting $\exists \forall_1 p$ from $\exists \forall_1 p_1$

if $b = 0$ **then** $L := \neg p$

else $L := p$

case $(DPLL(Q', S \cup \{L\}), \exists \forall)$ **of**

$(0, \forall) \Rightarrow$ **return** 0

$(0, \exists) \Rightarrow$ **return** $DPLL(Q', S \cup \{\bar{L}\})$

$(1, \forall) \Rightarrow$ **return** $DPLL(Q', S \cup \{\bar{L}\})$

$(1, \exists) \Rightarrow$ **return** 1

end

Improving DPLL with further simplifications

$$\exists p \exists q \forall r \exists s ((p \vee \neg r) \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s))$$

- We can treat $\neg r$ in $p \vee \neg r$ as 0 without loss of generality
- We can apply unit propagation
- We can treat r as 0 everywhere without loss of generality
- We can apply unit propagation to $\neg q$
- We can apply unit propagation to s

Improving DPLL with further simplifications

$$\exists p \exists q \forall r \exists s ((p \vee \neg r) \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s))$$

- We can treat $\neg r$ in $p \vee \neg r$ as 0 without loss of generality
- We can apply unit propagation
- We can treat r as 0 everywhere without loss of generality
- We can apply unit propagation to $\neg q$
- We can apply unit propagation to s

Improving DPLL with further simplifications

$$\begin{aligned} \exists p \exists q \forall r \exists s ((p \vee \neg r) \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) &\Rightarrow \\ \exists p \exists q \forall r \exists s (p \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) & \end{aligned}$$

- We can treat $\neg r$ in $p \vee \neg r$ as 0 without loss of generality
- We can apply unit propagation
- We can treat r as 0 everywhere without loss of generality
- We can apply unit propagation to $\neg q$
- We can apply unit propagation to s

Improving DPLL with further simplifications

$$\begin{aligned} & \exists p \exists q \forall r \exists s ((p \vee \neg r) \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) \Rightarrow \\ & \exists p \exists q \forall r \exists s (p \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) \end{aligned}$$

- We can treat $\neg r$ in $p \vee \neg r$ as 0 without loss of generality
- We can apply unit propagation
 - We can treat r as 0 everywhere without loss of generality
 - We can apply unit propagation to $\neg q$
 - We can apply unit propagation to s

Improving DPLL with further simplifications

$$\begin{aligned} & \exists p \exists q \forall r \exists s ((p \vee \neg r) \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) \Rightarrow \\ & \exists p \exists q \forall r \exists s (p \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) \Rightarrow \\ & \exists q \forall r \exists s ((\neg q \vee r) \wedge (q \vee s) \wedge (q \vee r \vee \neg s)) \end{aligned}$$

- We can treat $\neg r$ in $p \vee \neg r$ as 0 without loss of generality
- We can apply unit propagation
 - We can treat r as 0 everywhere without loss of generality
 - We can apply unit propagation to $\neg q$
 - We can apply unit propagation to s

Improving DPLL with further simplifications

$$\begin{aligned} & \exists p \exists q \forall r \exists s ((p \vee \neg r) \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) \Rightarrow \\ & \exists p \exists q \forall r \exists s (p \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) \Rightarrow \\ & \exists q \forall r \exists s ((\neg q \vee r) \wedge (q \vee s) \wedge (q \vee r \vee \neg s)) \end{aligned}$$

- We can treat $\neg r$ in $p \vee \neg r$ as 0 without loss of generality
- We can apply unit propagation
- We can treat r as 0 everywhere without loss of generality
 - We can apply unit propagation to $\neg q$
 - We can apply unit propagation to s

Improving DPLL with further simplifications

$$\begin{aligned} & \exists p \exists q \forall r \exists s ((p \vee \neg r) \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) \Rightarrow \\ & \exists p \exists q \forall r \exists s (p \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) \Rightarrow \\ & \exists q \forall r \exists s ((\neg q \vee r) \wedge (q \vee s) \wedge (q \vee r \vee \neg s)) \Rightarrow \\ & \exists q \exists s (\neg q \wedge (q \vee s) \wedge (q \vee \neg s)) \end{aligned}$$

- We can treat $\neg r$ in $p \vee \neg r$ as 0 without loss of generality
- We can apply unit propagation
- We can treat r as 0 everywhere without loss of generality
 - We can apply unit propagation to $\neg q$
 - We can apply unit propagation to s

Improving DPLL with further simplifications

$$\begin{aligned} & \exists p \exists q \forall r \exists s ((p \vee \neg r) \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) \Rightarrow \\ & \exists p \exists q \forall r \exists s (p \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) \Rightarrow \\ & \exists q \forall r \exists s ((\neg q \vee r) \wedge (q \vee s) \wedge (q \vee r \vee \neg s)) \Rightarrow \\ & \exists q \exists s (\neg q \wedge (q \vee s) \wedge (q \vee \neg s)) \end{aligned}$$

- We can treat $\neg r$ in $p \vee \neg r$ as 0 without loss of generality
- We can apply unit propagation
- We can treat r as 0 everywhere without loss of generality
- We can apply unit propagation to $\neg q$
- We can apply unit propagation to s

Improving DPLL with further simplifications

$$\begin{aligned} & \exists p \exists q \forall r \exists s ((p \vee \neg r) \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) \Rightarrow \\ & \exists p \exists q \forall r \exists s (p \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) \Rightarrow \\ & \exists q \forall r \exists s ((\neg q \vee r) \wedge (q \vee s) \wedge (q \vee r \vee \neg s)) \Rightarrow \\ & \exists q \exists s (\neg q \wedge (q \vee s) \wedge (q \vee \neg s)) \Rightarrow \\ & \exists s (s \wedge \neg s) \end{aligned}$$

- We can treat $\neg r$ in $p \vee \neg r$ as 0 without loss of generality
- We can apply unit propagation
- We can treat r as 0 everywhere without loss of generality
- We can apply unit propagation to $\neg q$
- We can apply unit propagation to s

Improving DPLL with further simplifications

$$\exists p \exists q \forall r \exists s ((p \vee \neg r) \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) \Rightarrow$$

$$\exists p \exists q \forall r \exists s (p \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) \Rightarrow$$

$$\exists q \forall r \exists s ((\neg q \vee r) \wedge (q \vee s) \wedge (q \vee r \vee \neg s)) \Rightarrow$$

$$\exists q \exists s (\neg q \wedge (q \vee s) \wedge (q \vee \neg s)) \Rightarrow$$

$$\exists s (s \wedge \neg s) \Rightarrow$$

□

- We can treat $\neg r$ in $p \vee \neg r$ as 0 without loss of generality
- We can apply unit propagation
- We can treat r as 0 everywhere without loss of generality
- We can apply unit propagation to $\neg q$
- We can apply unit propagation to s

Pure literal rule

Q : quantifier sequence

S : current clause set

L : literal of the form p or $\neg p$

Suppose L is *pure* in S (i.e., \bar{L} does not occur in S). Then:

- If p is *existentially* quantified in Q , we can *remove all clauses* containing L

Pure literal rule

Q : quantifier sequence

S : current clause set

L : literal of the form p or $\neg p$

Suppose L is *pure* in S (i.e., \bar{L} does not occur in S). Then:

- If p is **existentially** quantified in Q , we can **remove all clauses** containing L
- if p is **universally** quantified in Q , we can **remove L from all clauses**

Pure literal rule

Q : quantifier sequence

S : current clause set

L : literal of the form p or $\neg p$

Suppose L is *pure* in S (i.e., \bar{L} does not occur in S). Then:

- If p is **existentially** quantified in Q , we can **remove all clauses** containing L
- if p is **universally** quantified in Q , we can **remove L from all clauses**

Why?

Pure literal rule

Q : quantifier sequence

S : current clause set

L : literal of the form p or $\neg p$

Suppose L is *pure* in S (i.e., \bar{L} does not occur in S). Then:

- If p is **existentially** quantified in Q , we can **remove all clauses** containing L
- if p is **universally** quantified in Q , we can **remove L from all clauses**

Why?

- The \exists -player will make L true (satisfying all clauses containing L)

Pure literal rule

Q : quantifier sequence

S : current clause set

L : literal of the form p or $\neg p$

Suppose L is *pure* in S (i.e., \bar{L} does not occur in S). Then:

- If p is **existentially** quantified in Q , we can **remove all clauses** containing L
- if p is **universally** quantified in Q , we can **remove L from all clauses**

Why?

- The \exists -player will make L true (satisfying all clauses containing L)
- The \forall -player will make L false (so it can be removed from those clauses that contain L)

Universal literal deletion

Q : quantifier sequence

S : clause set

p, q : variables

- p is *existential in* Q if Q contains $\exists p$
- q is *universal in* Q if Q contains $\forall q$

Universal literal deletion

Q : quantifier sequence

S : clause set

p, q : variables

- p is *existential in* Q if Q contains $\exists p$
- q is *universal in* Q if Q contains $\forall q$
- p is *quantified before* a variable q if p occurs *before* q in Q

Universal literal deletion

Q : quantifier sequence

S : clause set

p, q : variables

- p is *existential in* Q if Q contains $\exists p$
- q is *universal in* Q if Q contains $\forall q$
- p is *quantified before* a variable q if p occurs *before* q in Q

Example: In $Q = \forall q \exists p \forall r$

q is quantified before both p and r ; and p is quantified before r

Universal literal deletion

Q : quantifier sequence

S : clause set

p, q : variables

- p is *existential in* Q if Q contains $\exists p$
- q is *universal in* Q if Q contains $\forall q$
- p is *quantified before* a variable q if p occurs *before* q in Q

Theorem 10

Suppose that

1. C is a clause in S ;
2. a variable q occurring in C is universal in Q ;
3. all existential variables of Q in C are quantified before q .

Then deleting the literal containing q from C does not change the truth value of $Q S$.

Universal literal deletion

Intuition behind Theorem 10

Consider a clause C from S of the form

$$L_1 \vee \cdots \vee L_n \vee (\neg)q_1 \vee \cdots \vee (\neg)q_m$$

where all existential variables of Q in C are quantified before q_1, \dots, q_m

- If at least one of L_1, \dots, L_n is true, then C is true regardless of the truth value of $(\neg)q_1, \dots, (\neg)q_m$
- If all of L_1, \dots, L_n are false, the \forall -player will make all $(\neg)q_1, \dots, (\neg)q_m$ false and win the game

Universal literal deletion

Intuition behind Theorem 10

Consider a clause C from S of the form

$$L_1 \vee \cdots \vee L_n \vee (\neg)q_1 \vee \cdots \vee (\neg)q_m$$

where all existential variables of Q in C are quantified before q_1, \dots, q_m

Consider the position before the q_1, \dots, q_m -moves of the \forall -player

- If at least one of L_1, \dots, L_n is true, then C is true regardless of the truth value of $(\neg)q_1, \dots, (\neg)q_m$
- If all of L_1, \dots, L_n are false, the \forall -player will make all $(\neg)q_1, \dots, (\neg)q_m$ false and win the game

Universal literal deletion

Intuition behind Theorem 10

Consider a clause C from S of the form

$$L_1 \vee \cdots \vee L_n \vee (\neg)q_1 \vee \cdots \vee (\neg)q_m$$

where all existential variables of Q in C are quantified before q_1, \dots, q_m

Consider the position before the q_1, \dots, q_m -moves of the \forall -player

- If at least one of L_1, \dots, L_n is true,
then C is true regardless of the truth value of $(\neg)q_1, \dots, (\neg)q_m$
- If all of L_1, \dots, L_n are false,
the \forall -player will make all $(\neg)q_1, \dots, (\neg)q_m$ false and win the game

Universal literal deletion

Intuition behind Theorem 10

Consider a clause C from S of the form

$$L_1 \vee \cdots \vee L_n \vee (\neg)q_1 \vee \cdots \vee (\neg)q_m$$

where all existential variables of Q in C are quantified before q_1, \dots, q_m

Consider the position before the q_1, \dots, q_m -moves of the \forall -player

- If at least one of L_1, \dots, L_n is true, then C is true regardless of the truth value of $(\neg)q_1, \dots, (\neg)q_m$
- If all of L_1, \dots, L_n are false, the \forall -player will make all $(\neg)q_1, \dots, (\neg)q_m$ false and win the game

Universal literal deletion

Intuition behind Theorem 10

Consider a clause C from S of the form

$$L_1 \vee \cdots \vee L_n \vee (\neg)q_1 \vee \cdots \vee (\neg)q_m$$

where all existential variables of Q in C are quantified before q_1, \dots, q_m

Consider the position before the q_1, \dots, q_m -moves of the \forall -player

- If at least one of L_1, \dots, L_n is true, then C is true regardless of the truth value of $(\neg)q_1, \dots, (\neg)q_m$
- If all of L_1, \dots, L_n are false, the \forall -player will make all $(\neg)q_1, \dots, (\neg)q_m$ false and win the game

In either case, the deletion of $(\neg)q_1, \dots, (\neg)q_m$ will not change the final outcome

Example revisited

$$\exists p \exists q \forall r \exists s ((p \vee \neg r) \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s))$$

Example revisited

$$\exists p \exists q \forall r \exists s ((p \vee \neg r) \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s))$$

- Apply universal literal deletion to $p \vee \neg r$

Example revisited

$$\begin{aligned} & \exists p \exists q \forall r \exists s ((p \vee \neg r) \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) \Rightarrow \\ & \exists p \exists q \forall r \exists s (p \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) \end{aligned}$$

- Apply universal literal deletion to $p \vee \neg r$

Example revisited

$$\begin{aligned} & \exists p \exists q \forall r \exists s ((p \vee \neg r) \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) \Rightarrow \\ & \exists p \exists q \forall r \exists s (p \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) \end{aligned}$$

- Apply universal literal deletion to $p \vee \neg r$
- Apply unit propagation

Example revisited

$$\begin{aligned} & \exists p \exists q \forall r \exists s ((p \vee \neg r) \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) \Rightarrow \\ & \exists p \exists q \forall r \exists s (p \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) \Rightarrow \\ & \exists q \forall r \exists s ((\neg q \vee r) \wedge (q \vee s) \wedge (q \vee r \vee \neg s)) \end{aligned}$$

- Apply universal literal deletion to $p \vee \neg r$
- Apply unit propagation

Example revisited

$$\begin{aligned} & \exists p \exists q \forall r \exists s ((p \vee \neg r) \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) \Rightarrow \\ & \exists p \exists q \forall r \exists s (p \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) \Rightarrow \\ & \exists q \forall r \exists s ((\neg q \vee r) \wedge (q \vee s) \wedge (q \vee r \vee \neg s)) \end{aligned}$$

- Apply universal literal deletion to $p \vee \neg r$
- Apply unit propagation
- Apply the pure literal rule to r

Example revisited

$$\begin{aligned} & \exists p \exists q \forall r \exists s ((p \vee \neg r) \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) \Rightarrow \\ & \exists p \exists q \forall r \exists s (p \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) \Rightarrow \\ & \exists q \forall r \exists s ((\neg q \vee r) \wedge (q \vee s) \wedge (q \vee r \vee \neg s)) \Rightarrow \\ & \exists q \exists s (\neg q \wedge (q \vee s) \wedge (q \vee \neg s)) \end{aligned}$$

- Apply universal literal deletion to $p \vee \neg r$
- Apply unit propagation
- Apply the pure literal rule to r

Example revisited

$$\begin{aligned} & \exists p \exists q \forall r \exists s ((p \vee \neg r) \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) \Rightarrow \\ & \exists p \exists q \forall r \exists s (p \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) \Rightarrow \\ & \exists q \forall r \exists s ((\neg q \vee r) \wedge (q \vee s) \wedge (q \vee r \vee \neg s)) \Rightarrow \\ & \exists q \exists s (\neg q \wedge (q \vee s) \wedge (q \vee \neg s)) \end{aligned}$$

- Apply universal literal deletion to $p \vee \neg r$
- Apply unit propagation
- Apply the pure literal rule to r
- Apply unit propagation

Example revisited

$$\begin{aligned} & \exists p \exists q \forall r \exists s ((p \vee \neg r) \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) \Rightarrow \\ & \exists p \exists q \forall r \exists s (p \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) \Rightarrow \\ & \exists q \forall r \exists s ((\neg q \vee r) \wedge (q \vee s) \wedge (q \vee r \vee \neg s)) \Rightarrow \\ & \exists q \exists s (\neg q \wedge (q \vee s) \wedge (q \vee \neg s)) \Rightarrow \\ & \exists s (s \wedge \neg s) \end{aligned}$$

- Apply universal literal deletion to $p \vee \neg r$
- Apply unit propagation
- Apply the pure literal rule to r
- Apply unit propagation

Example revisited

$$\begin{aligned} & \exists p \exists q \forall r \exists s ((p \vee \neg r) \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) \Rightarrow \\ & \exists p \exists q \forall r \exists s (p \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee s) \wedge (\neg p \vee q \vee r \vee \neg s)) \Rightarrow \\ & \exists q \forall r \exists s ((\neg q \vee r) \wedge (q \vee s) \wedge (q \vee r \vee \neg s)) \Rightarrow \\ & \exists q \exists s (\neg q \wedge (q \vee s) \wedge (q \vee \neg s)) \Rightarrow \\ & \exists s (s \wedge \neg s) \Rightarrow \\ & \square \end{aligned}$$

- Apply universal literal deletion to $p \vee \neg r$
- Apply unit propagation
- Apply the pure literal rule to r
- Apply unit propagation

Quantified Boolean Formulas and BDDs

OBDDs are efficient data structures to represent propositional formulas

Can we use them also to represent QBFs?

Quantified Boolean Formulas and BDDs

OBDDs are efficient data structures to represent propositional formulas

Can we use them also to represent QBFs?

Quantification: given an OBDD representing a formula F , find an OBDD representing $\exists x_1 p_1 \cdots \exists x_n p_n F$

Quantified Boolean Formulas and BDDs

OBDDs are efficient data structures to represent propositional formulas

Can we use them also to represent QBFs?

Quantification: given an OBDD representing a formula F , find an OBDD representing $\exists v_1 p_1 \cdots \exists v_n p_n F$

There is no simple algorithm for quantification over OBDDs in general, but there is one when $\exists v_1 \cdots \exists v_n$ are the same quantifier

Quantified Boolean Formulas and BDDs

OBDDs are efficient data structures to represent propositional formulas
Can we use them also to represent QBFs?

Quantification: given an OBDD representing a formula F , find an OBDD representing $\exists v_1 p_1 \cdots \exists v_n p_n F$

There is no simple algorithm for quantification over OBDDs in general,
but there is one when $\exists v_1 \cdots \exists v_n$ are the **same** quantifier

Quantification for OBDDs

We can rely on the following properties of QBFs:

- $\exists p$ (if p then F else G) $\equiv F \vee G$
- $\forall p$ (if p then F else G) $\equiv F \wedge G$
- \exists/p (if q then F else G) \equiv if q then \exists/pF else \exists/pG when $p \neq q$

Quantification for OBDDs

We can rely on the following properties of QBFs:

- $\exists p$ (if p then F else G) $\equiv F \vee G$
- $\forall p$ (if p then F else G) $\equiv F \wedge G$
- \exists/p (if q then F else G) \equiv if q then \exists/pF else \exists/pG when $p \neq q$

Quantification for OBDDs

We can rely on the following properties of QBFs:

- $\exists p (if\ p\ then\ F\ else\ G) \equiv F \vee G$
- $\forall p (if\ p\ then\ F\ else\ G) \equiv F \wedge G$
- $\exists \forall p (if\ q\ then\ F\ else\ G) \equiv if\ q\ then\ \exists \forall p F\ else\ \exists \forall p G$ **when** $p \neq q$

\exists -quantification algorithm for OBDDs

procedure $\exists quant(\{p_1, \dots, p_k\}, \{n_1, \dots, n_m\})$

parameters: global dag D

input: nodes n_1, \dots, n_m representing F_1, \dots, F_m in D

output: a node n representing $\exists p_1 \dots \exists p_k (F_1 \vee \dots \vee F_m)$ in (modified) D

begin

if $m = 0$ **then return** $\boxed{0}$

if some n_i is $\boxed{1}$ **then return** $\boxed{1}$

if some n_i is $\boxed{0}$ **then return** $\exists quant(\{p_1, \dots, p_k\}, \{n_1, \dots, n_{i-1}, n_{i+1}, \dots, n_m\})$

$p := max_atom(n_1, \dots, n_m)$

forall $i = 1 \dots m$

if n_i is labelled by p

then $(l_i, r_i) := (lo(n_i), hi(n_i))$

else $(l_i, r_i) := (n_i, n_i)$

if $p \in \{p_1, \dots, p_k\}$

then return $\exists quant(\{p_1, \dots, p_k\} - \{p\}, \{l_1, \dots, l_m, r_1, \dots, r_m\})$

else

$k_1 := \exists quant(\{p_1, \dots, p_k\}, \{l_1, \dots, l_m\})$

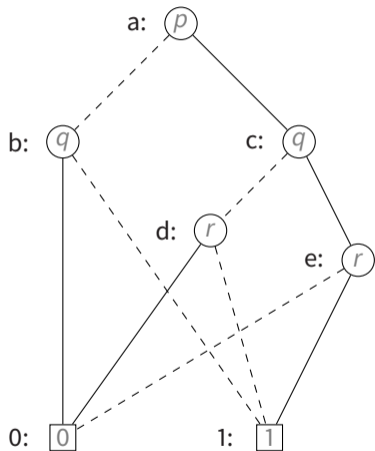
$k_2 := \exists quant(\{p_1, \dots, p_k\}, \{r_1, \dots, r_m\})$

return $integrate(k_1, p, k_2, D)$

end

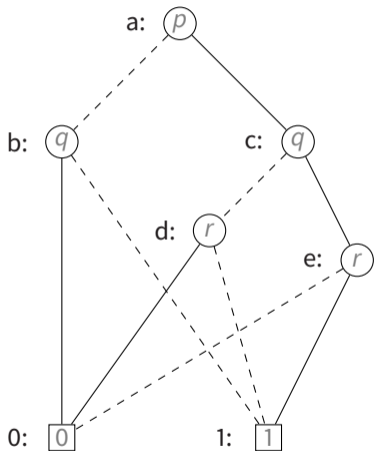
Example

$\exists \text{quant}(\{p, r\}, \{a\})$

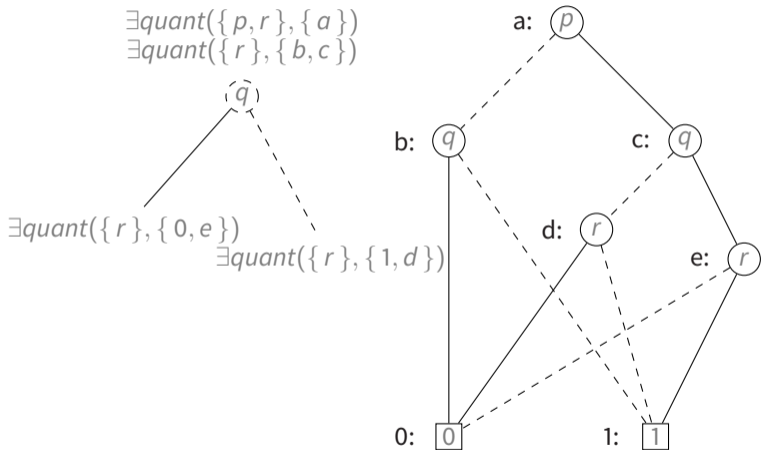


Example

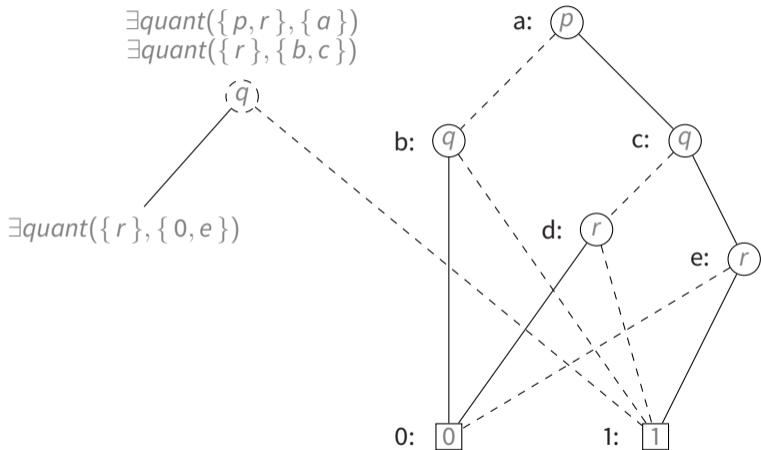
$\exists \text{quant}(\{p, r\}, \{a\})$
 $\exists \text{quant}(\{r\}, \{b, c\})$



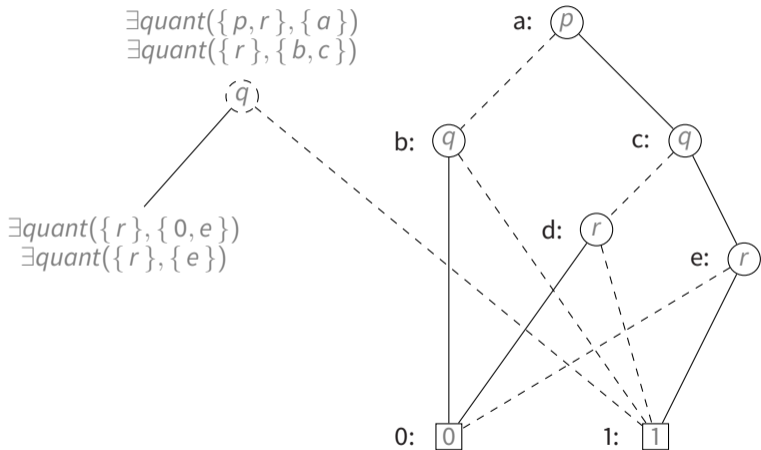
Example



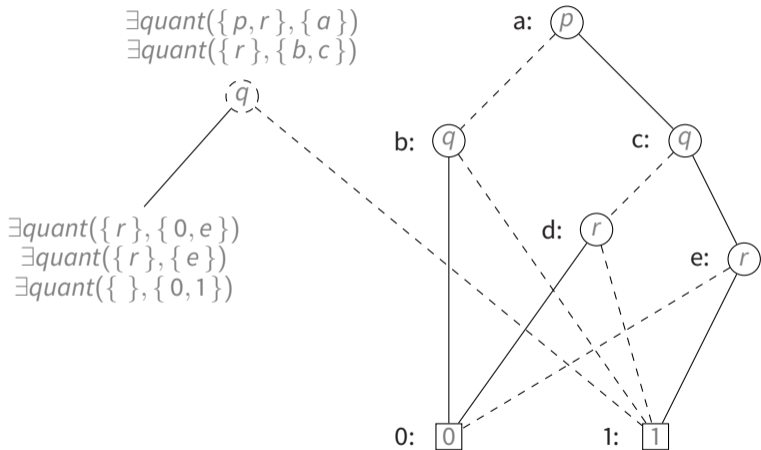
Example



Example

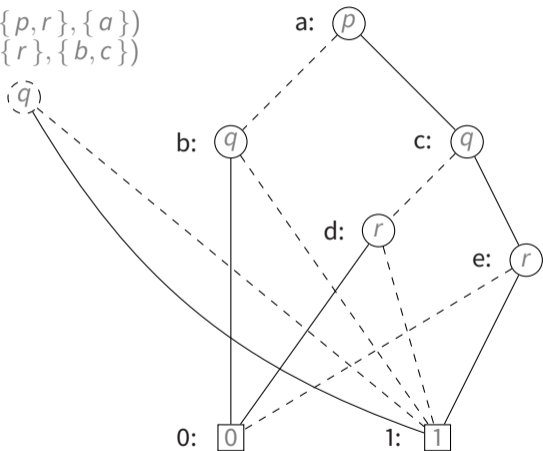


Example



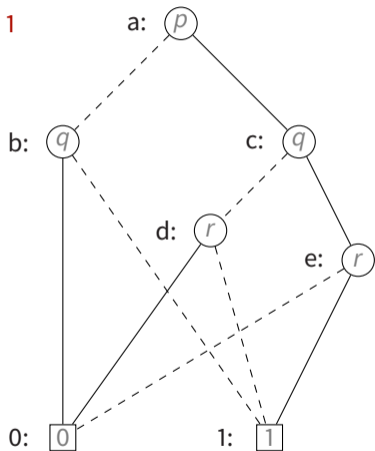
Example

$\exists \text{quant}(\{p, r\}, \{a\})$
 $\exists \text{quant}(\{r\}, \{b, c\})$



Example

$$\exists \text{quant}(\{p, r\}, \{a\}) = 1$$



\exists -quantification algorithm for OBDDs

procedure \exists quant($\{p_1, \dots, p_k\}, \{n_1, \dots, n_m\}$)

parameters: global dag D

input: nodes n_1, \dots, n_m representing F_1, \dots, F_m in D

output: a node n representing $\exists p_1 \dots \exists p_k (F_1 \vee \dots \vee F_m)$ in (modified) D

begin

if $m = 0$ then return $\boxed{0}$

if some n_i is $\boxed{1}$ then return $\boxed{1}$

if some n_i is $\boxed{0}$ then return \exists quant($\{p_1, \dots, p_k\}, \{n_1, \dots, n_{i-1}, n_{i+1}, \dots, n_m\}$)

$p := \text{max_atom}(n_1, \dots, n_m)$

forall $i = 1 \dots m$

if n_i is labelled by p

then $(l_i, r_i) := (\text{lo}(n_i), \text{hi}(n_i))$

else $(l_i, r_i) := (n_i, n_i)$

if $p \in \{p_1, \dots, p_k\}$

then return \exists quant($\{p_1, \dots, p_k\} - \{p\}, \{l_1, \dots, l_m, r_1, \dots, r_m\}$)

else

$k_1 := \exists$ quant($\{p_1, \dots, p_k\}, \{l_1, \dots, l_m\}$)

$k_2 := \exists$ quant($\{p_1, \dots, p_k\}, \{r_1, \dots, r_m\}$)

return integrate(k_1, p, k_2, D)

end

\forall -quantification algorithm for OBDDs

procedure $\forall quant(\{p_1, \dots, p_k\}, \{n_1, \dots, n_m\})$

parameters: global dag D

input: nodes n_1, \dots, n_m representing F_1, \dots, F_m in D

output: a node n representing $\forall p_1 \dots \forall p_k (F_1 \wedge \dots \wedge F_m)$ in (modified) D

begin

if $m = 0$ then return $\boxed{1}$

if some n_i is $\boxed{0}$ then return $\boxed{0}$

if some n_i is $\boxed{1}$ then return $\forall quant(\{p_1, \dots, p_k\}, \{n_1, \dots, n_{i-1}, n_{i+1}, \dots, n_m\})$

$p := max_atom(n_1, \dots, n_m)$

forall $i = 1 \dots m$

if n_i is labelled by p

then $(l_i, r_i) := (lo(n_i), hi(n_i))$

else $(l_i, r_i) := (n_i, n_i)$

if $p \in \{p_1, \dots, p_k\}$

then return $\forall quant(\{p_1, \dots, p_k\} - \{p\}, \{l_1, \dots, l_m, r_1, \dots, r_m\})$

else

$k_1 := \forall quant(\{p_1, \dots, p_k\}, \{l_1, \dots, l_m\})$

$k_2 := \forall quant(\{p_1, \dots, p_k\}, \{r_1, \dots, r_m\})$

return $integrate(k_1, p, k_2, D)$

end