What's next

Algorithms for satisfiability, validity of QBF:

- Splitting
- DPLL

Reminder:
(i) $F(p_1, \ldots, p_n)$ is satisfiable iff $\exists p_1 \ldots \exists p_n F(p_1, \ldots, p_n)$ is true.
(ii) $F(p_1, \ldots, p_n)$ is valid iff $\forall p_1 \ldots \forall p_n F(p_1, \ldots, p_n)$ is true.

Algorithms will check whether a closed formula is true or false.

Splitting: foundations

Lemma

- *A closed formula $\forall p F$ is true if and only if the formulas $F_p^\perp$ and $F_p^\top$ are true.*
- *A closed formula $\exists p F$ is true if and only if at least one of the formulas $F_p^\perp$ or $F_p^\top$ is true.*

Splitting

Simplification rules for $\top$:

$$\neg\top \;\Rightarrow\; \bot$$

$$\top \wedge F_1 \wedge \ldots \wedge F_n \;\Rightarrow\; F_1 \wedge \ldots \wedge F_n$$

$$\top \vee F_1 \vee \ldots \vee F_n \;\Rightarrow\; \top$$

$$F \rightarrow \top \;\Rightarrow\; \top \qquad \top \rightarrow F \;\Rightarrow\; F$$

$$F \leftrightarrow \top \;\Rightarrow\; F \qquad \top \leftrightarrow F \;\Rightarrow\; F$$

Simplification rules for $\bot$:

$$\neg\bot \;\Rightarrow\; \top$$

$$\bot \wedge F_1 \wedge \ldots \wedge F_n \;\Rightarrow\; \bot$$

$$\bot \vee F_1 \vee \ldots \vee F_n \;\Rightarrow\; F_1 \vee \ldots \vee F_n$$

$$F \rightarrow \bot \;\Rightarrow\; \neg F \qquad \bot \rightarrow F \;\Rightarrow\; \top$$

$$F \leftrightarrow \bot \;\Rightarrow\; \neg F \qquad \bot \leftrightarrow F \;\Rightarrow\; \neg F$$

Splitting

Simplification rules for $\top$:

$$
\begin{aligned}
\neg\top &\Rightarrow \bot \\
\top \wedge F_1 \wedge \ldots \wedge F_n &\Rightarrow F_1 \wedge \ldots \wedge F_n \\
\top \vee F_1 \vee \ldots \vee F_n &\Rightarrow \top \\
F \rightarrow \top \Rightarrow \top &\qquad \top \rightarrow F \Rightarrow F \\
F \leftrightarrow \top \Rightarrow F &\qquad \top \leftrightarrow F \Rightarrow F \\
\forall p \top &\Rightarrow \top \\
\exists p \top &\Rightarrow \top
\end{aligned}
$$

Simplification rules for $\bot$:

$$
\begin{aligned}
\neg\bot &\Rightarrow \top \\
\bot \wedge F_1 \wedge \ldots \wedge F_n &\Rightarrow \bot \\
\bot \vee F_1 \vee \ldots \vee F_n &\Rightarrow F_1 \vee \ldots \vee F_n \\
F \rightarrow \bot \Rightarrow \neg F &\qquad \bot \rightarrow F \Rightarrow \top \\
F \leftrightarrow \bot \Rightarrow \neg F &\qquad \bot \leftrightarrow F \Rightarrow \neg F \\
\forall p \bot &\Rightarrow \bot \\
\exists p \bot &\Rightarrow \bot
\end{aligned}
$$

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 and a boolean value for it)
begin
$F \; := \; simplify(F)$
if $F = \bot$ **then** **return** 0
if $F = \top$ **then** **return** 1
Let F have the form $\exists\!\!\!\forall p_1 \ldots \exists\!\!\!\forall p_k F_1$
$(p, b) \; := \; select\_variable\_value(F)$
Let F' be obtained from F by deleting $\exists\!\!\!\forall p$ from its outermost prefix
if $b = 0$ **then** //$p \mapsto \bot$ branch first
case $(splitting((F')^{\bot}_{p}), \; \exists\!\!\!\forall)$ **of**
$(0, \forall) \Rightarrow$ **return** 0 $\quad$ $(1, \exists) \Rightarrow$ **return** 1
$(1, \forall) \Rightarrow$ **return** $splitting((F')^{\top}_{p})$ $\quad$ $(0, \exists) \Rightarrow$ **return** $splitting((F')^{\top}_{p})$
end

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 and a boolean value for it)
begin
$F \ := \ \mathit{simplify}(F)$
if $F = \bot$ **then return** 0
if $F = \top$ **then return** 1
Let F have the form $\exists\!\!\forall p_1 \ldots \exists\!\!\forall p_k F_1$
$(p, b) \ := \ \mathit{select\_variable\_value}(F)$
Let F' be obtained from F by deleting $\exists\!\!\forall p$ from its outermost prefix
if $b = 0$ **then** //$p \mapsto \bot$ branch first
case $(\mathit{splitting}((F')^{\bot}_{p}), \ \exists\!\!\forall)$ **of**
$(0, \forall) \Rightarrow$ **return** 0 $(1, \exists) \Rightarrow$ **return** 1
$(1, \forall) \Rightarrow$ **return** $\mathit{splitting}((F')^{\top}_{p})$ $(0, \exists) \Rightarrow$ **return** $\mathit{splitting}((F')^{\top}_{p})$
end
else //$b = 1$
case $(\mathit{splitting}((F')^{\top}_{p}), \ \exists\!\!\forall)$ **of**
$(0, \forall) \Rightarrow$ **return** 0 $(1, \exists) \Rightarrow$ **return** 1
$(1, \forall) \Rightarrow$ **return** $\mathit{splitting}((F')^{\bot}_{p})$ $(0, \exists) \Rightarrow$ **return** $\mathit{splitting}((F')^{\bot}_{p})$
end

Splitting: examples

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

Splitting: examples

$$\begin{tikzcd}[column sep=small]
 & \forall p \exists q (p \leftrightarrow q) \arrow[dl, "p = 0"'] \\
\exists q (\neg q) & & \wedge
\end{tikzcd}$$

Splitting: examples

$$\begin{tikzcd}[column sep=small]
 & & \forall p \exists q (p \leftrightarrow q) \arrow[dl, "p = 0"'] \\
 & \exists q(\neg q) \arrow[dl, "q = 0"'] & \wedge \\
\boxed{1}\ \top & \vee &
\end{tikzcd}$$

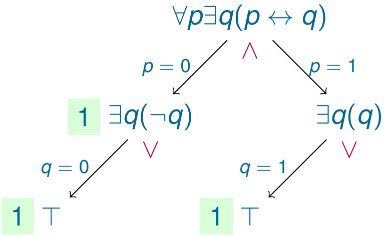
Splitting: examples

$$
\begin{tikzcd}
 & & \forall p \exists q (p \leftrightarrow q) \quad \wedge \arrow[dl, "p = 0"'] \\
 & \boxed{1}\ \exists q (\neg q) \quad \vee \arrow[dl, "q = 0"'] & \\
\boxed{1}\ \top & &
\end{tikzcd}
$$

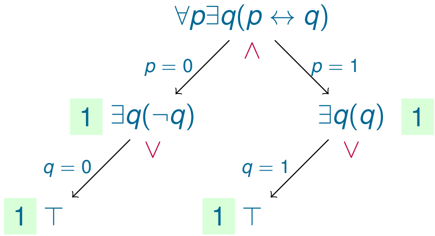
Splitting: examples

$$
\begin{tikzcd}
 & & \forall p \exists q (p \leftrightarrow q) \arrow[dl, "p = 0"'] \arrow[dr, "p = 1"] \arrow[d, phantom, "\wedge"] & \\
 & \boxed{1}\ \exists q(\neg q) \arrow[dl, "q = 0"'] \arrow[d, phantom, "\vee"] & {} & \exists q(q) \\
 \boxed{1}\ \top & {} & &
\end{tikzcd}
$$

Splitting: examples



Splitting: examples



Splitting: examples

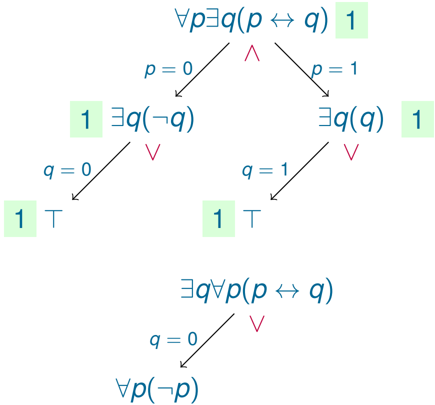
$$
\begin{tikzcd}[column sep=small]
 & & \underset{\wedge}{\forall p \exists q (p \leftrightarrow q)}\ \boxed{1} \arrow[dl, "p=0"'] \arrow[dr, "p=1"] & & \\
 & \boxed{1}\ \underset{\vee}{\exists q (\neg q)} \arrow[dl, "q=0"'] & & \underset{\vee}{\exists q (q)}\ \boxed{1} \arrow[dl, "q=1"'] & \\
 \boxed{1}\ \top & & \boxed{1}\ \top & &
\end{tikzcd}
$$

Splitting: examples

$$
\begin{tikzcd}
 & & \underset{\wedge}{\forall p \exists q (p \leftrightarrow q)}\ \boxed{1} \arrow[dl, "p=0"'] \arrow[dr, "p=1"] & & \\
 & \boxed{1}\ \underset{\vee}{\exists q(\neg q)} \arrow[dl, "q=0"'] & & \underset{\vee}{\exists q(q)}\ \boxed{1} \arrow[dl, "q=1"'] & \\
\boxed{1}\ \top & & \boxed{1}\ \top & &
\end{tikzcd}
$$

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

Splitting: examples



Splitting: examples

$$
\begin{tikzcd}[column sep=small]
 & & \forall p \exists q (p \leftrightarrow q)\ \boxed{1} \arrow[dl, "p = 0"'] \arrow[dr, "p = 1"] \arrow[d, phantom, "\wedge"] & & \\
 & \boxed{1}\ \exists q (\neg q) \arrow[dl, "q = 0"'] \arrow[d, phantom, "\vee"] & {} & \exists q (q)\ \boxed{1} \arrow[dl, "q = 1"'] \arrow[d, phantom, "\vee"] & \\
 \boxed{1}\ \top & {} & \boxed{1}\ \top & {} &
\end{tikzcd}
$$

$$
\begin{tikzcd}[column sep=small]
 & & \exists q \forall p (p \leftrightarrow q) \arrow[dl, "q = 0"'] \arrow[d, phantom, "\vee"] \\
 & \forall p (\neg p) \arrow[dl, "p = 1"'] \arrow[d, phantom, "\wedge"] & {} \\
 \boxed{0}\ \bot & {} &
\end{tikzcd}
$$

Splitting: examples

$$
\begin{tikzcd}
 & & \forall p \exists q (p \leftrightarrow q)\ \boxed{1} \arrow[dl, "p=0"'] \arrow[dr, "p=1"] \arrow[d, phantom, "\wedge"] & \\
 & \boxed{1}\ \exists q(\neg q) \arrow[dl, "q=0"'] \arrow[d, phantom, "\vee"] & & \exists q(q)\ \boxed{1} \arrow[dl, "q=1"'] \arrow[d, phantom, "\vee"] \\
\boxed{1}\ \top & & \boxed{1}\ \top &
\end{tikzcd}
$$

$$
\begin{tikzcd}
 & & \exists q \forall p (p \leftrightarrow q) \arrow[dl, "q=0"'] \arrow[d, phantom, "\vee"] \\
 & \boxed{0}\ \forall p(\neg p) \arrow[dl, "p=1"'] \arrow[d, phantom, "\wedge"] & \\
\boxed{0}\ \bot & &
\end{tikzcd}
$$

Splitting: examples

$$\begin{tikzcd}
 & & \forall p \exists q (p \leftrightarrow q)\ \boxed{1} \arrow[dl, "p = 0"'] \arrow[dr, "p = 1"] \arrow[d, phantom, "\wedge"] & \\
 & \boxed{1}\ \exists q(\neg q) \arrow[dl, "q = 0"'] \arrow[d, phantom, "\vee"] & & \exists q(q)\ \boxed{1} \arrow[dl, "q = 1"'] \arrow[d, phantom, "\vee"] \\
\boxed{1}\ \top & & \boxed{1}\ \top &
\end{tikzcd}$$

$$\begin{tikzcd}
 & & \exists q \forall p (p \leftrightarrow q) \arrow[dl, "q = 0"'] \arrow[dr, "q = 1"] \arrow[d, phantom, "\vee"] & \\
 & \boxed{0}\ \forall p(\neg p) \arrow[dl, "p = 1"'] \arrow[d, phantom, "\wedge"] & & \forall p(p) \\
\boxed{0}\ \bot & & &
\end{tikzcd}$$

Splitting: examples

$$\begin{tikzcd}
 & & \underset{\land}{\forall p \exists q (p \leftrightarrow q)}\ \boxed{1} \arrow[dl, "p = 0"'] \arrow[dr, "p = 1"] & \\
 & \boxed{1}\ \underset{\lor}{\exists q (\neg q)} \arrow[dl, "q = 0"'] & & \underset{\lor}{\exists q (q)}\ \boxed{1} \arrow[dl, "q = 1"'] \\
\boxed{1}\ \top & & \boxed{1}\ \top &
\end{tikzcd}$$

$$\begin{tikzcd}
 & & \underset{\lor}{\exists q \forall p (p \leftrightarrow q)} \arrow[dl, "q = 0"'] \arrow[dr, "q = 1"] & \\
 & \boxed{0}\ \underset{\land}{\forall p (\neg p)} \arrow[dl, "p = 1"'] & & \underset{\land}{\forall p (p)} \arrow[dl, "p = 0"'] \\
\boxed{0}\ \bot & & \boxed{0}\ \bot &
\end{tikzcd}$$

Splitting: examples

$$\begin{tikzcd}
 & & \forall p \exists q (p \leftrightarrow q)\ \boxed{1} \arrow[dl, "p = 0"'] \arrow[dr, "p = 1"] \arrow[d, phantom, "\wedge"] & & \\
 & \boxed{1}\ \exists q (\neg q) \arrow[dl, "q = 0"'] \arrow[d, phantom, "\vee"] & {} & \exists q (q)\ \boxed{1} \arrow[dl, "q = 1"'] \arrow[d, phantom, "\vee"] & \\
\boxed{1}\ \top & {} & \boxed{1}\ \top & {} &
\end{tikzcd}$$

$$\begin{tikzcd}
 & & \exists q \forall p (p \leftrightarrow q) \arrow[dl, "q = 0"'] \arrow[dr, "q = 1"] \arrow[d, phantom, "\vee"] & & \\
 & \boxed{0}\ \forall p (\neg p) \arrow[dl, "p = 1"'] \arrow[d, phantom, "\wedge"] & {} & \forall p (p)\ \boxed{0} \arrow[dl, "p = 0"'] \arrow[d, phantom, "\wedge"] & \\
\boxed{0}\ \bot & {} & \boxed{0}\ \bot & {} &
\end{tikzcd}$$

Splitting: examples

$$
\begin{tikzcd}
 & & \forall p \exists q (p \leftrightarrow q)\ \boxed{1} \arrow[dl, "p = 0"'] \arrow[dr, "p = 1"] \arrow[d, phantom, "\wedge"] & \\
 & \boxed{1}\ \exists q (\neg q) \arrow[dl, "q = 0"'] \arrow[d, phantom, "\vee"] & {} & \exists q (q)\ \boxed{1} \arrow[dl, "q = 1"'] \arrow[d, phantom, "\vee"] \\
 \boxed{1}\ \top & {} & \boxed{1}\ \top & {}
\end{tikzcd}
$$

$$
\begin{tikzcd}
 & & \exists q \forall p (p \leftrightarrow q)\ \boxed{0} \arrow[dl, "q = 0"'] \arrow[dr, "q = 1"] \arrow[d, phantom, "\vee"] & \\
 & \boxed{0}\ \forall p (\neg p) \arrow[dl, "p = 1"'] \arrow[d, phantom, "\wedge"] & {} & \forall p (p)\ \boxed{0} \arrow[dl, "p = 0"'] \arrow[d, phantom, "\wedge"] \\
 \boxed{0}\ \bot & {} & \boxed{0}\ \bot & {}
\end{tikzcd}
$$

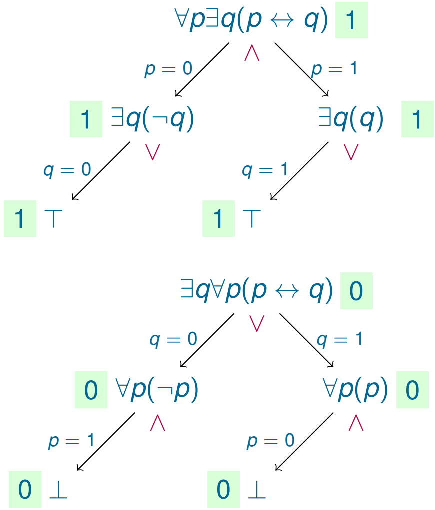
Splitting: examples

$$
\begin{tikzcd}[column sep=small]
 & & \forall p \exists q (p \leftrightarrow q)\ \boxed{1} \arrow[dl, "p = 0"'] \arrow[dr, "p = 1"] \arrow[d, phantom, "\wedge"] & \\
 & \boxed{1}\ \exists q (\neg q) \arrow[dl, "q = 0"'] \arrow[d, phantom, "\vee"] & & \exists q (q)\ \boxed{1} \arrow[dl, "q = 1"'] \arrow[d, phantom, "\vee"] \\
\boxed{1}\ \top & & \boxed{1}\ \top &
\end{tikzcd}
$$

$$
\begin{tikzcd}[column sep=small]
 & & \exists q \forall p (p \leftrightarrow q)\ \boxed{0} \arrow[dl, "q = 0"'] \arrow[dr, "q = 1"] \arrow[d, phantom, "\vee"] & \\
 & \boxed{0}\ \forall p (\neg p) \arrow[dl, "p = 1"'] \arrow[d, phantom, "\wedge"] & & \forall p (p)\ \boxed{0} \arrow[dl, "p = 0"'] \arrow[d, phantom, "\wedge"] \\
\boxed{0}\ \bot & & \boxed{0}\ \bot &
\end{tikzcd}
$$

Note: the order of variables is important!

Splitting: examples



Note: the order of variables is important!

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 it false.

CNF

For more efficient algorithms we need formulas to be in a convenient normal form.

CNF

For more efficient algorithms we need formulas to be in a convenient normal form.

Our next aim is to modify CNF and DPLL to deal with quantified boolean formulas.

CNF

For more efficient algorithms we need formulas 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 CNF, if it is either $\bot$, or $\top$, or has the form $\exists\!\!\!\forall_1 p_1 \ldots \exists\!\!\!\forall_n p_n(C_1 \land \ldots \land C_m)$, where $C_1, \ldots, C_m$ are clauses.

CNF

For more efficient algorithms we need formulas 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 CNF, if it is either $\bot$, or $\top$, or has the form $\exists\!\!\!\forall_1 p_1 \ldots \exists\!\!\!\forall_n p_n (C_1 \wedge \ldots \wedge C_m)$, where $C_1, \ldots, C_m$ 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:

$$
\begin{array}{rcl}
F \leftrightarrow G & \Rightarrow & (\neg F \vee G) \wedge (\neg G \vee F), \\
F \rightarrow G & \Rightarrow & \neg F \vee G, \\
\neg(F \wedge G) & \Rightarrow & \neg F \vee \neg G, \\
\neg(F \vee G) & \Rightarrow & \neg F \wedge \neg G, \\
\neg\neg F & \Rightarrow & F, \\
(F_1 \wedge \ldots \wedge F_m) \vee G_1 \vee \ldots \vee G_n & \Rightarrow & \begin{array}{cc} (F_1 \vee G_1 \vee \ldots \vee G_n) & \wedge \\ \ldots & \wedge \\ (F_m \vee G_1 \vee \ldots \vee G_n). & \end{array}
\end{array}
$$

DPLL

Input of DPLL:

- Q: quantifier sequence $\exists\!\!\forall_1 p_1 \ldots\ldots \exists\!\!\forall_n p_n$
- S: a set of clauses

DPLL:

- splitting
- unit propagation

DPLL

Input of DPLL:

- Q: quantifier sequence $\exists\!\!\!\forall_1 p_1 \ldots\ldots \exists\!\!\!\forall_n p_n$
- S: a set of clauses

DPLL:

- splitting
- unit propagation

Unit propagation with respect to Q, S:

if S contains a unit clause, i.e. a clause consisting of one literal L of the form p or $\neg p$ then

DPLL

Input of DPLL:

- Q: quantifier sequence $\exists\!\!\!\forall_1 p_1 \ldots\ldots \exists\!\!\!\forall_n p_n$
- S: a set of clauses

DPLL:

- splitting
- unit propagation

Unit propagation with respect to Q, S:

if S contains a unit clause, i.e. a clause consisting of one literal L of the form p or $\neg p$ then

- if Q contains $\exists p$ or p does not occur in Q
  1. remove from S every clause of the form $L \vee C'$;
  2. replace in S every clause of the form $\overline{L} \vee C'$ by the clause C'.

DPLL

Input of DPLL:

- Q: quantifier sequence $\exists\!\!\!\forall_1 p_1 \ldots\ldots \exists\!\!\!\forall_n p_n$
- S: a set of clauses

DPLL:

- splitting
- unit propagation

Unit propagation with respect to Q, S:
if S contains a unit clause, i.e. a clause consisting of one literal L of the form p or $\neg p$ then

- if Q contains $\exists p$ or p does not occur in Q
  1. remove from S every clause of the form $L \vee C'$;
  2. replace in S every clause of the form $\overline{L} \vee C'$ by the clause C'.
- if Q contains $\forall p$, then replace S by the set $\{\square\}$;

DPLL

Input of DPLL:

- Q: quantifier sequence $\exists\!\!\!\forall_1 p_1 \ldots\ldots \exists\!\!\!\forall_n p_n$
- S: a set of clauses

DPLL:

- splitting
- unit propagation

Unit propagation with respect to Q, S:
if S contains a unit clause, i.e. a clause consisting of one literal L of the form p or $\neg p$ then

- if Q contains $\exists p$ or p does not occur in Q
  1. remove from S every clause of the form $L \vee C'$;
  2. replace in S every clause of the form $\overline{L} \vee C'$ by the clause C'.
- if Q contains $\forall p$, then replace S by the set $\{\square\}$;

Why different for universal quantifiers? Use intuition from games!

DPLL

Input of DPLL:

- Q: quantifier sequence $\exists\!\!\forall_1 p_1 \ldots\ldots \exists\!\!\forall_n p_n$
- S: a set of clauses

DPLL:

- splitting
- unit propagation

Unit propagation with respect to Q, S:

if S contains a unit clause, i.e. a clause consisting of one literal L of the form p or $\neg p$ then

- if Q contains $\exists p$ or p does not occur in Q
  1. remove from S every clause of the form $L \lor C'$;
  2. replace in S every clause of the form $\overline{L} \lor C'$ by the clause C'.
- if Q contains $\forall p$, then replace S by the set $\{\square\}$;

Why different for universal quantifiers? Use intuition from games!

The player playing $\forall$ wants to make the formula false.

DPLL

Input of DPLL:

- Q: quantifier sequence $\exists\!\!\!\forall_1 p_1 \ldots\ldots \exists\!\!\!\forall_n p_n$
- S: a set of clauses

DPLL:

- splitting
- unit propagation

Unit propagation with respect to Q, S:
if S contains a unit clause, i.e. a clause consisting of one literal L of the form p or $\neg p$ then

- if Q contains $\exists p$ or p does not occur in Q
  1. remove from S every clause of the form $L \vee C'$;
  2. replace in S every clause of the form $\overline{L} \vee C'$ by the clause C'.
- if Q contains $\forall p$, then replace S by the set $\{\square\}$;

Why different for universal quantifiers? Use intuition from games!

The player playing $\forall$ wants to make the formula false.
When it is his turn to make a move $\forall p$, he has a winning move: select the value for p which makes the unit clause false (and hence the conjunction of clauses false too).

DPLL algorithm

```
procedure DPLL(Q, S)
input: quantifier sequence Q = ∃∀₁p₁ ... ∃∀ₙpₙ, set of clauses S
output: 0 or 1
parameters: function select_variable_value (selects a variable
              from the outermost prefix of F and a boolean value for it)
begin
  S := unit_propagate(Q, S)
  if S is empty then return 1
  if S contains □ then return 0
  (p, b) := select_variable_value(Q, S)
  Let Q′ be obtained from Q by deleting ∃∀p from its outermost prefix
  if b = 0 then L := ¬p
           else L := p
  case (DPLL(Q′, S ∪ {L}), ∃∀) of
    (0, ∀) ⇒ return 0
    (1, ∀) ⇒ return DPLL(Q′, S ∪ {L̄})
    (1, ∃) ⇒ return 1
    (0, ∃) ⇒ return DPLL(Q′, S ∪ {L̄})
end
```

Example

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

$$\lor$$

Example

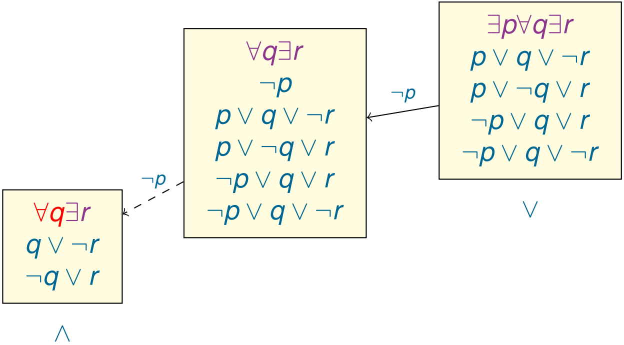
$$\forall q \exists r$$
$$\neg p$$
$$p \vee q \vee \neg r$$
$$p \vee \neg q \vee r$$
$$\neg p \vee q \vee r$$
$$\neg p \vee q \vee \neg r$$

$$\xleftarrow{\neg p}$$

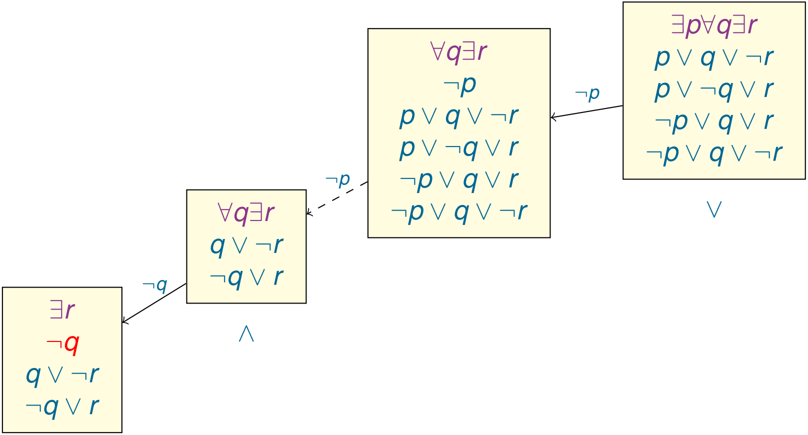
$$\exists p \forall q \exists r$$
$$p \vee q \vee \neg r$$
$$p \vee \neg q \vee r$$
$$\neg p \vee q \vee r$$
$$\neg p \vee q \vee \neg r$$

$$\vee$$

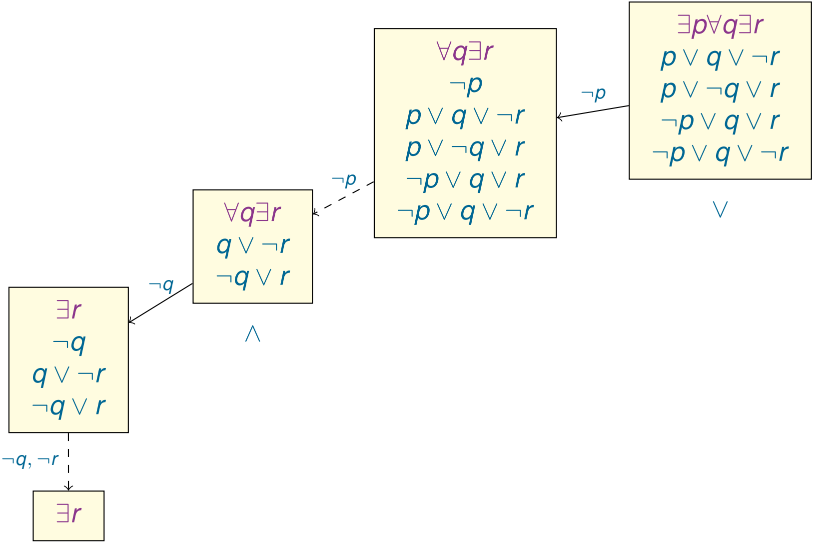
Example


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

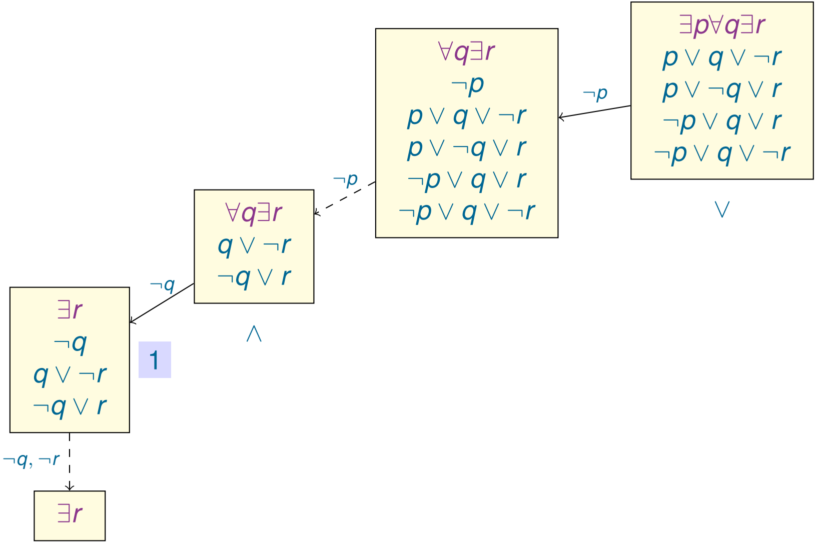
Example


$\exists p \forall q \exists r$
$p \lor q \lor \neg r$
$p \lor \neg q \lor r$
$\neg p \lor q \lor r$
$\neg p \lor q \lor \neg r$
$\lor$
$\neg p$
$\forall q \exists r$
$\neg p$
$p \lor q \lor \neg r$
$p \lor \neg q \lor r$
$\neg p \lor q \lor r$
$\neg p \lor q \lor \neg r$
$\neg p$
$\forall q \exists r$
$q \lor \neg r$
$\neg q \lor r$
$\land$
$\neg q$
$\exists r$
$\neg q$
$q \lor \neg r$
$\neg q \lor r$

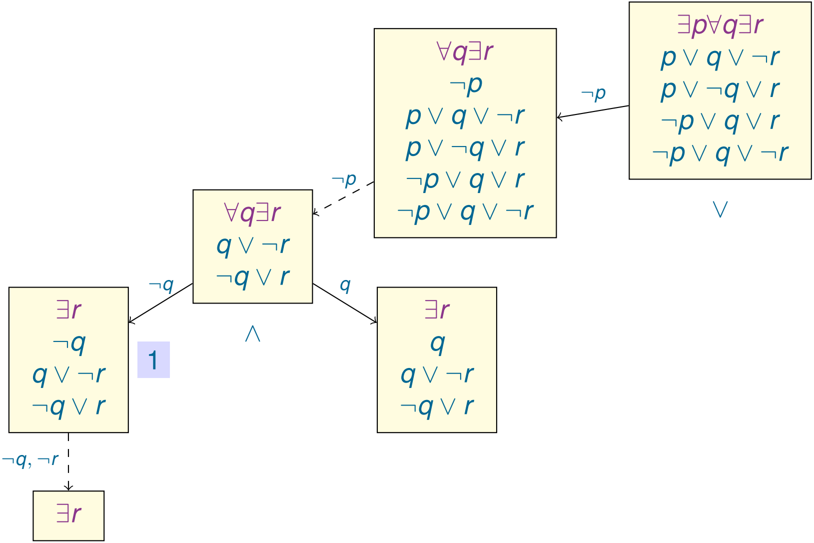
Example


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

Example

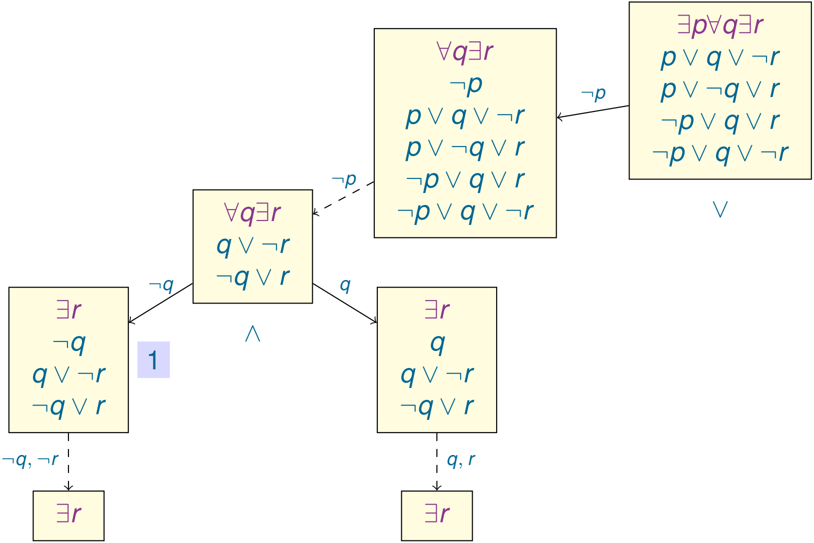

$\exists p \forall q \exists r$
$p \vee q \vee \neg r$
$p \vee \neg q \vee r$
$\neg p \vee q \vee r$
$\neg p \vee q \vee \neg r$
$\vee$
$\neg p$
$\forall q \exists r$
$\neg p$
$p \vee q \vee \neg r$
$p \vee \neg q \vee r$
$\neg p \vee q \vee r$
$\neg p \vee q \vee \neg r$
$\neg p$
$\forall q \exists r$
$q \vee \neg r$
$\neg q \vee r$
$\wedge$
$\neg q$
$\exists r$
$\neg q$
$q \vee \neg r$
$\neg q \vee r$
1
$\neg q, \neg r$
$\exists r$

Example

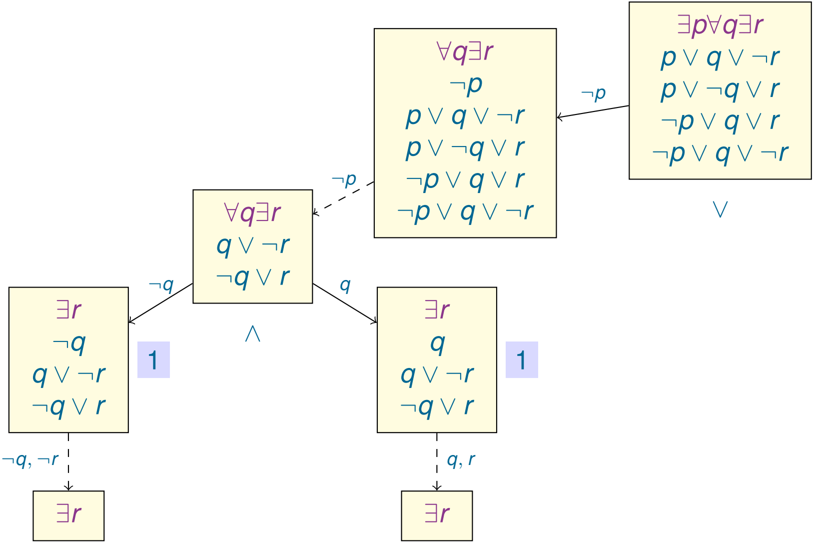


∃p∀q∃r
p ∨ q ∨ ¬r
p ∨ ¬q ∨ r
¬p ∨ q ∨ r
¬p ∨ q ∨ ¬r
∨
¬p
∀q∃r
¬p
p ∨ q ∨ ¬r
p ∨ ¬q ∨ r
¬p ∨ q ∨ r
¬p ∨ q ∨ ¬r
¬p
∀q∃r
q ∨ ¬r
¬q ∨ r
∧
¬q
q
∃r
¬q
q ∨ ¬r
¬q ∨ r
1
∃r
q
q ∨ ¬r
¬q ∨ r
¬q, ¬r
∃r

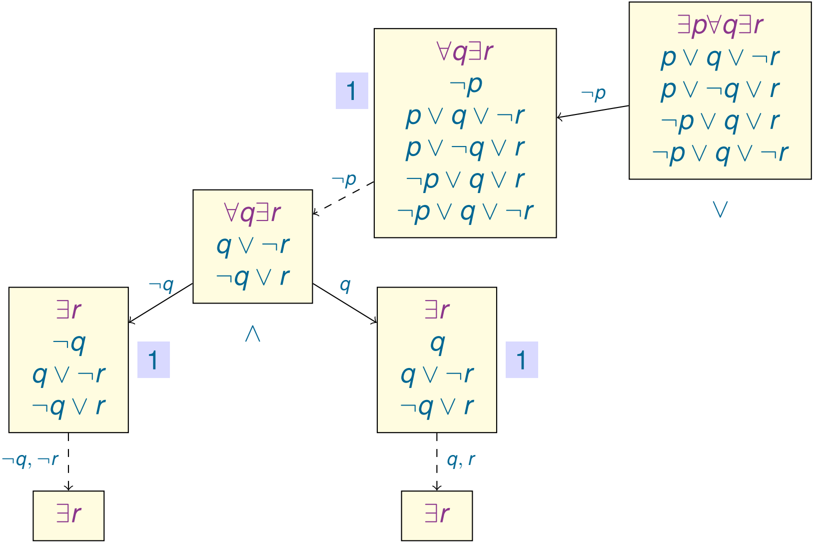
Example


$\exists p \forall q \exists r$
$p \vee q \vee \neg r$
$p \vee \neg q \vee r$
$\neg p \vee q \vee r$
$\neg p \vee q \vee \neg r$
$\vee$
$\neg p$
$\forall q \exists r$
$\neg p$
$p \vee q \vee \neg r$
$p \vee \neg q \vee r$
$\neg p \vee q \vee r$
$\neg p \vee q \vee \neg r$
$\neg p$
$\forall q \exists r$
$q \vee \neg r$
$\neg q \vee r$
$\wedge$
$\neg q$
$\exists r$
$\neg q$
$q \vee \neg r$
$\neg q \vee r$
1
q
$\exists r$
q
$q \vee \neg r$
$\neg q \vee r$
$\neg q, \neg r$
$\exists r$
q, r
$\exists r$

Example

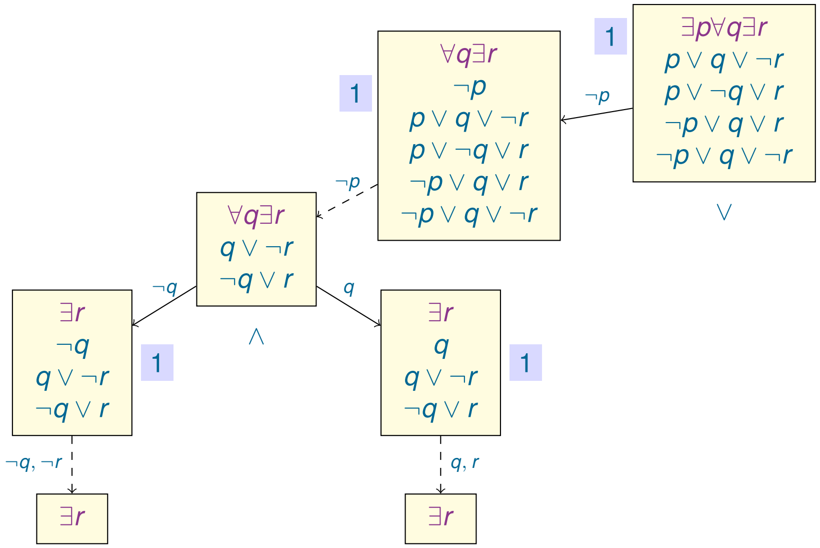

∃p∀q∃r
p ∨ q ∨ ¬r
p ∨ ¬q ∨ r
¬p ∨ q ∨ r
¬p ∨ q ∨ ¬r
∨
¬p
∀q∃r
¬p
p ∨ q ∨ ¬r
p ∨ ¬q ∨ r
¬p ∨ q ∨ r
¬p ∨ q ∨ ¬r
¬p
∀q∃r
q ∨ ¬r
¬q ∨ r
∧
¬q
q
∃r
¬q
q ∨ ¬r
¬q ∨ r
1
∃r
q
q ∨ ¬r
¬q ∨ r
1
¬q, ¬r
q, r
∃r
∃r

Example



∃p∀q∃r
p ∨ q ∨ ¬r
p ∨ ¬q ∨ r
¬p ∨ q ∨ r
¬p ∨ q ∨ ¬r
∨
¬p
∀q∃r
¬p
p ∨ q ∨ ¬r
p ∨ ¬q ∨ r
¬p ∨ q ∨ r
¬p ∨ q ∨ ¬r
1
¬p
∀q∃r
q ∨ ¬r
¬q ∨ r
∧
¬q
q
∃r
¬q
q ∨ ¬r
¬q ∨ r
1
∃r
q
q ∨ ¬r
¬q ∨ r
1
¬q, ¬r
q, r
∃r
∃r

Example



1
∃p∀q∃r
p ∨ q ∨ ¬r
p ∨ ¬q ∨ r
¬p ∨ q ∨ r
¬p ∨ q ∨ ¬r
∨
¬p
1
∀q∃r
¬p
p ∨ q ∨ ¬r
p ∨ ¬q ∨ r
¬p ∨ q ∨ r
¬p ∨ q ∨ ¬r
¬p
∀q∃r
q ∨ ¬r
¬q ∨ r
∧
¬q
q
∃r
¬q
q ∨ ¬r
¬q ∨ r
1
∃r
q
q ∨ ¬r
¬q ∨ r
1
¬q, ¬r
q, r
∃r
∃r

Pure literal rule

Let Q be quantifier prefix and S set of clauses.
Let literal L be pure in S (i.e. $\overline{L}$ does not occur in S) then:

- If the variable of L is existentially quantified in Q then we can remove all clauses in which L occurs.
- If the variable of L is universally quantified then we can remove L from all clauses where L occurs.

Pure literal rule

Let Q be quantifier prefix and S set of clauses.
Let literal L be pure in S (i.e. $\overline{L}$ does not occur in S) then:

- If the variable of L is existentially quantified in Q then we can remove all clauses in which L occurs.
- If the variable of L is universally quantified then we can remove L from all clauses where L occurs.

Pure literal rule

Let Q be quantifier prefix and S set of clauses.
Let literal L be pure in S (i.e. $\overline{L}$ does not occur in S) then:

- If the variable of L is existentially quantified in Q then we can remove all clauses in which L occurs.
- If the variable of L is universally quantified then we can remove L from all clauses where L occurs.

Why?

Pure literal rule

Let Q be quantifier prefix and S set of clauses.
Let literal L be pure in S (i.e. $\overline{L}$ does not occur in S) then:

- If the variable of L is existentially quantified in Q then we can remove all clauses in which L occurs.
- If the variable of L is universally quantified then we can remove L from all clauses where L occurs.

Why?

- The $\exists$-player will make the literal true (so all clauses containing this literal will be satisfied).

Pure literal rule

Let Q be quantifier prefix and S set of clauses.
Let literal L be pure in S (i.e. $\overline{L}$ does not occur in S) then:

- If the variable of L is existentially quantified in Q then we can remove all clauses in which L occurs.
- If the variable of L is universally quantified then we can remove L from all clauses where L occurs.

Why?

- The $\exists$-player will make the literal true (so all clauses containing this literal will be satisfied).
- The $\forall$-player will make the literal false (so it can be removed from all clauses containing this literal).

Universal literal deletion

Consider a quantifier prefix Q and a conjunction of clauses S.

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

Example: If Q is $\forall q \exists p \forall r$ then q is quantified before both p and r; and p is quantified before r (in Q).

Theorem

Let Q be a quantifier prefix and S a conjunction of clauses. Suppose that

1. *C is a non-tautological clause in S;*
2. *a variable q in C is universal in Q;*
3. *all existential variables in C are quantified before q.*

Then the deletion of the literal containing q from C does not change the truth value of QS.

Universal literal deletion

Consider a quantifier prefix Q and a conjunction of clauses S.

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

Example: If Q is $\forall q \exists p \forall r$ then q is quantified before both p and r; and p is quantified before r (in Q).

Theorem

Let Q be a quantifier prefix and S a conjunction of clauses. Suppose that

1. *C is a non-tautological clause in S;*
2. *a variable q in C is universal in Q;*
3. *all existential variables in C are quantified before q.*

Then the deletion of the literal containing q from C does not change the truth value of QS.

Universal literal deletion

Consider a quantifier prefix Q and a conjunction of clauses S.

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

Example: If Q is $\forall q \exists p \forall r$ then q is quantified before both p and r; and p is quantified before r (in Q).

Theorem

Let Q be a quantifier prefix and S a conjunction of clauses. Suppose that

1. *C is a non-tautological clause in S;*
2. *a variable q in C is universal in Q;*
3. *all existential variables in C are quantified before q.*

Then the deletion of the literal containing q from C does not change the truth value of QS.

Universal literal deletion

Consider a quantifier prefix Q and a conjunction of clauses S.

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

Example: If Q is $\forall q \exists p \forall r$ then q is quantified before both p and r; and p is quantified before r (in Q).

Theorem

Let Q be a quantifier prefix and S a conjunction of clauses. Suppose that

1. *C is a non-tautological clause in S;*
2. *a variable q in C is universal in Q;*
3. *all existential variables in C are quantified before q.*

Then the deletion of the literal containing q from C does not change the truth value of QS.

Universal literal deletion

Let $q_1, \ldots, q_m$ be all universal variables of C such that all existential variables are quantified before them. Then C has the form:

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

Consider the position before the $q_1, \ldots, q_m$-moves of the $\forall$-player.

- If at least one of the literals $L_1, \ldots, L_n$ is true, deletion of $(\neg)q_1, \ldots, (\neg)q_m$ will not change the outcome of the game, since after any assignment to $q_1, \ldots, q_m$ the clause will be true.
- If all of the literals $L_1, \ldots, L_n$ are false, the $\forall$-player will make all $(\neg)q_1, \ldots, (\neg)q_m$ false and win the game, so deletion of these literals will not change the outcome of the game either.

Universal literal deletion

Let $q_1, \ldots, q_m$ be all universal variables of C such that all existential variables are quantified before them. Then C has the form:

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

Consider the position before the $q_1, \ldots, q_m$-moves of the $\forall$-player.

- If at least one of the literals $L_1, \ldots, L_n$ is true, deletion of $(\neg)q_1, \ldots, (\neg)q_m$ will not change the outcome of the game, since after any assignment to $q_1, \ldots, q_m$ the clause will be true.
- If all of the literals $L_1, \ldots, L_n$ are false, the $\forall$-player will make all $(\neg)q_1, \ldots, (\neg)q_m$ false and win the game, so deletion of these literals will not change the outcome of the game either.

Universal literal deletion

Let $q_1, \ldots, q_m$ be all universal variables of C such that all existential variables are quantified before them. Then C has the form:

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

Consider the position before the $q_1, \ldots, q_m$-moves of the $\forall$-player.

- If at least one of the literals $L_1, \ldots, L_n$ is true, deletion of $(\neg)q_1, \ldots, (\neg)q_m$ will not change the outcome of the game, since after any assignment to $q_1, \ldots, q_m$ the clause will be true.
- If all of the literals $L_1, \ldots, L_n$ are false, the $\forall$-player will make all $(\neg)q_1, \ldots, (\neg)q_m$ false and win the game, so deletion of these literals will not change the outcome of the game either.

Universal literal deletion

Let $q_1, \ldots, q_m$ be all universal variables of C such that all existential variables are quantified before them. Then C has the form:

$$L_1 \lor \ldots \lor L_n \lor (\neg)q_1 \lor \ldots \lor (\neg)q_m$$

Consider the position before the $q_1, \ldots, q_m$-moves of the $\forall$-player.

- If at least one of the literals $L_1, \ldots, L_n$ is true, deletion of $(\neg)q_1, \ldots, (\neg)q_m$ will not change the outcome of the game, since after any assignment to $q_1, \ldots, q_m$ the clause will be true.
- If all of the literals $L_1, \ldots, L_n$ are false, the $\forall$-player will make all $(\neg)q_1, \ldots, (\neg)q_m$ false and win the game, so deletion of these literals will not change the outcome of the game either.

Example

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

Example

$$\exists p \exists q \forall r \exists s ((p \lor \neg r) \land (\neg q \lor r) \land (\neg p \lor q \lor s) \land (\neg p \lor q \lor r \lor \neg s))$$

Example

$$\exists p \exists q \forall r \exists s ((p \lor \neg r) \land (\neg q \lor r) \land (\neg p \lor q \lor s) \land (\neg p \lor q \lor r \lor \neg s))$$

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

Example

$$\exists p \exists q \forall r \exists s ((p \lor \neg r) \land (\neg q \lor r) \land (\neg p \lor q \lor s) \land (\neg p \lor q \lor r \lor \neg s)) \Rightarrow$$
$$\exists p \exists q \forall r \exists s (p \land (\neg q \lor r) \land (\neg p \lor q \lor s) \land (\neg p \lor q \lor r \lor \neg s))$$

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

Example

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

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

Example

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

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

Example

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

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

Example

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

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

Example

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

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

Example

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

- Apply universal literal deletion to $p \vee \neg r$
- Unit propagation p
- Apply the pure literal rule to r
- Unit propagation $\neg q$, s

QBF and OBDD

Any QBF $\quad F(p_1, \ldots, p_n)$ represents a boolean function.

OBDDs can be used to canonically represent boolean functions.

We know how to apply boolean operations to OBDDs.
Can we also apply quantification to OBDDs in a straighforward way?

QBF and OBDD

Any QBF $F(p_1, \ldots, p_n)$ represents a boolean function.

OBDDs can be used to canonically represent boolean functions.

We know how to apply boolean operations to OBDDs.
Can we also apply quantification to OBDDs in a straighforward way?

Quantification: given an OBDD representing a formula F, find an OBDD representing $\exists\!\!\!\forall_1 p_1 \ldots \exists\!\!\!\forall_n p_n F$

Quantification for OBDDs

We can use the following properties of QBFs:

- $\exists p\,(\,\mathit{if}\ p\ \mathit{then}\ F\ \mathit{else}\ G) \equiv F \lor G$;
- $\forall p\,(\,\mathit{if}\ p\ \mathit{then}\ F\ \mathit{else}\ G) \equiv F \land G$;

Quantification for OBDDs

We can use the following properties of QBFs:

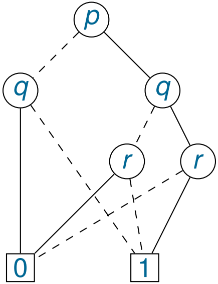
- $\exists p\,(\textit{ if } p \textit{ then } F \textit{ else } G) \equiv F \vee G$;
- $\forall p\,(\textit{ if } p \textit{ then } F \textit{ else } G) \equiv F \wedge G$;
- If $p \neq q$, then
  $\exists\!\forall p\,(\textit{ if } q \textit{ then } F \textit{ else } G) \equiv \textit{ if } q \textit{ then } \exists\!\forall pF \textit{ else } \exists\!\forall pG$

$\exists$-quantification algorithm for OBDDs

procedure $\exists quant(\{p_1, \ldots, p_k\}, \{n_1, \ldots, n_m\})$
parameters: global dag D
input: nodes $n_1, \ldots, n_m$ representing $F_1, \ldots, F_m$ in D
output: a node n representing $\exists p_1 \ldots \exists p_k (F_1 \vee \ldots \vee F_m)$ 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 $\exists quant(\{p_1, \ldots, p_k\}, \{n_1, \ldots, n_{i-1}, n_{i+1}, \ldots, n_m\})$
  $p := max\_atom(n_1, \ldots, n_m)$
  forall $i = 1 \ldots m$
    if n_i is labelled by p
      then $(l_i, r_i) := (neg(n_i), pos(n_i))$
      else $(l_i, r_i) := (n_i, n_i)$
  if $p \in \{p_1, \ldots, p_k\}$
    then **return** $\exists quant(\{p_1, \ldots, p_k\} - \{p\}, \{l_1, \ldots, l_m, r_1, \ldots, r_m\})$
    else
      $k_1 := \exists quant(\{p_1, \ldots, p_k\}, \{l_1, \ldots, l_m\})$
      $k_2 := \exists quant(\{p_1, \ldots, p_k\}, \{r_1, \ldots, r_m\})$
      return $integrate(k_1, p, k_2, D)$
end

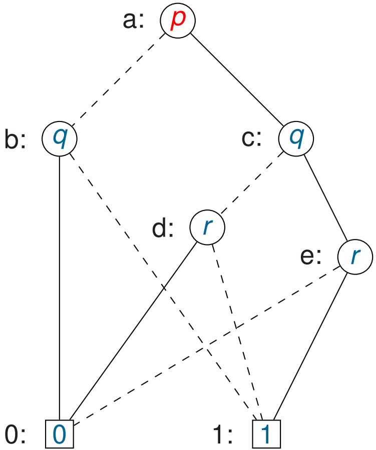
Example

Take the order $p > q > r$ and the formula $\exists r \exists p(p \leftrightarrow ((p \to r) \leftrightarrow q))$.



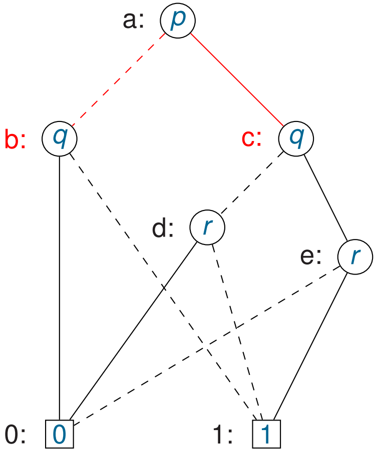
Example $\exists r \exists p (p \leftrightarrow ((p \rightarrow r) \leftrightarrow q))$

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

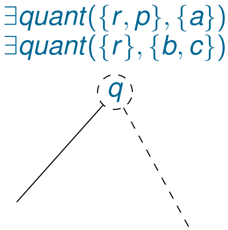


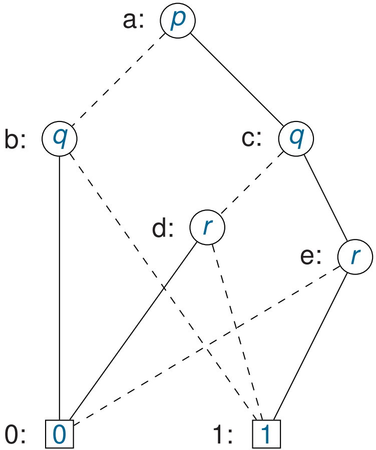
Example $\exists r \exists p (p \leftrightarrow ((p \rightarrow r) \leftrightarrow q))$

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

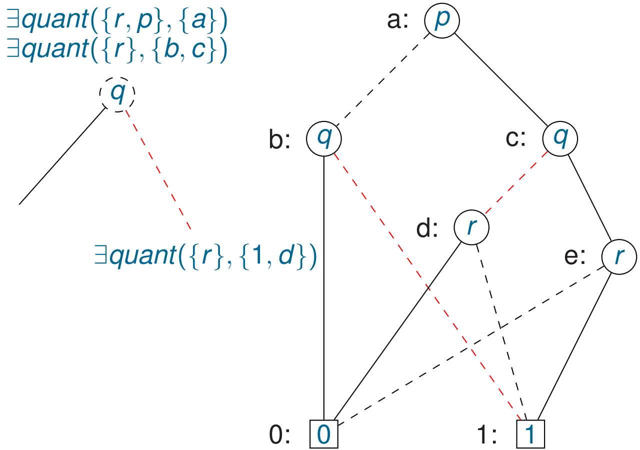


Example $\exists r \exists p(p \leftrightarrow ((p \rightarrow r) \leftrightarrow q))$

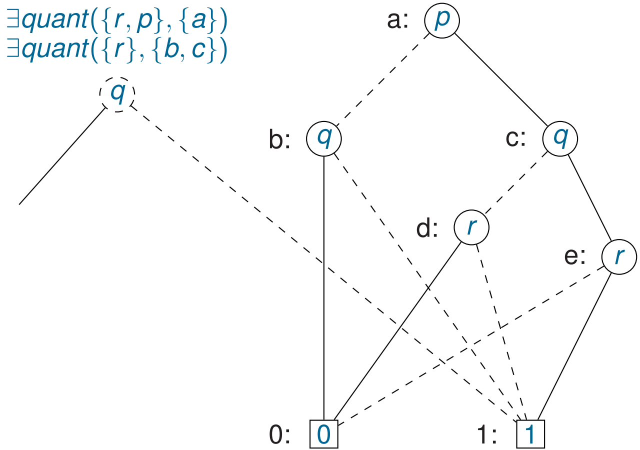




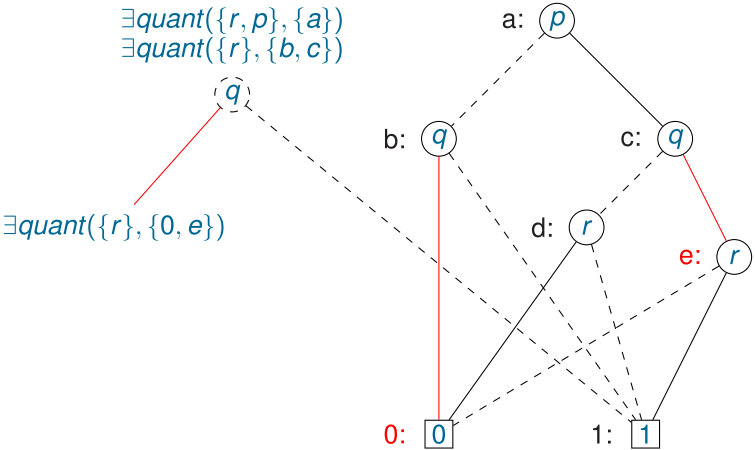
Example $\exists r \exists p (p \leftrightarrow ((p \rightarrow r) \leftrightarrow q))$



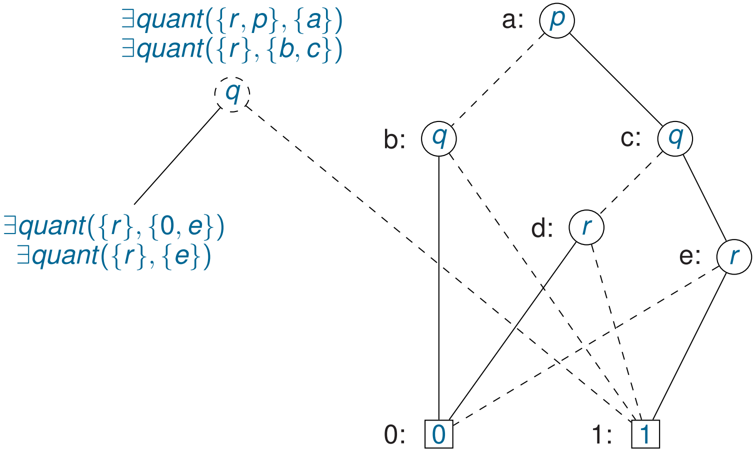
Example $\exists r \exists p (p \leftrightarrow ((p \to r) \leftrightarrow q))$



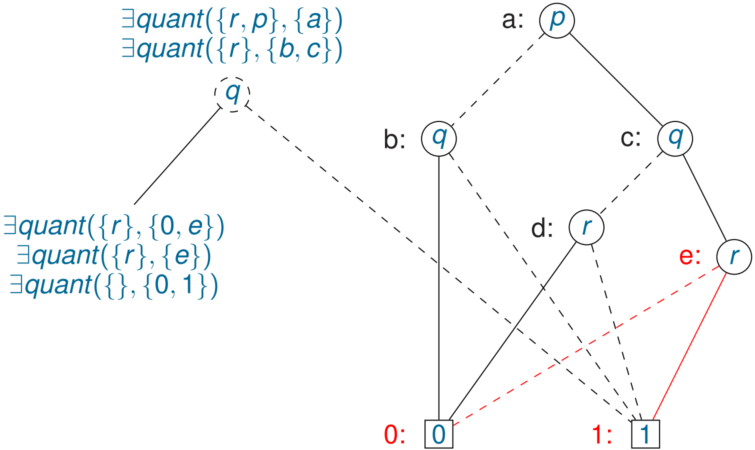
Example $\exists r \exists p (p \leftrightarrow ((p \rightarrow r) \leftrightarrow q))$



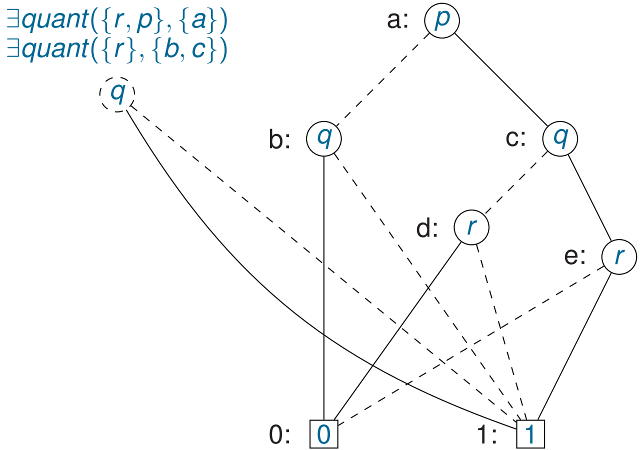
Example $\exists r \exists p(p \leftrightarrow ((p \rightarrow r) \leftrightarrow q))$



Example $\exists r \exists p (p \leftrightarrow ((p \rightarrow r) \leftrightarrow q))$

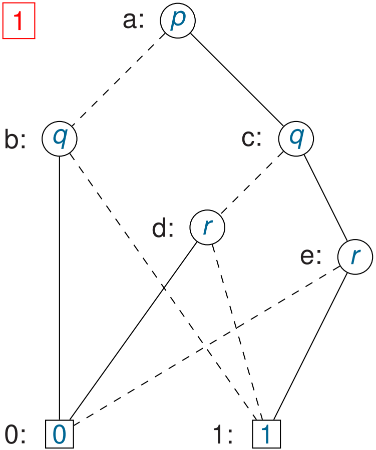


Example $\exists r \exists p(p \leftrightarrow ((p \rightarrow r) \leftrightarrow q))$



Example $\exists r \exists p(p \leftrightarrow ((p \rightarrow r) \leftrightarrow q))$

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



QBF $\exists r \exists p(p \leftrightarrow ((p \rightarrow r) \leftrightarrow q))$ is represented by the node $\boxed{1}$.
This formula is true.

$\exists$-quantification algorithm for OBDDs

procedure $\exists quant(\{p_1, \ldots, p_k\}, \{n_1, \ldots, n_m\})$
parameters: global dag D
input: nodes $n_1, \ldots, n_m$ representing $F_1, \ldots, F_m$ in D
output: a node n representing $\exists p_1 \ldots \exists p_k(F_1 \vee \ldots \vee F_m)$ 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 $\exists quant(\{p_1, \ldots, p_k\}, \{n_1, \ldots, n_{i-1}, n_{i+1}, \ldots, n_m\})$
  $p := max\_atom(n_1, \ldots, n_m)$
  forall $i = 1 \ldots m$
    if n_i is labelled by p
      then $(l_i, r_i) := (neg(n_i), pos(n_i))$
      else $(l_i, r_i) := (n_i, n_i)$
  if $p \in \{p_1, \ldots, p_k\}$
    then **return** $\exists quant(\{p_1, \ldots, p_k\} - \{p\}, \{l_1, \ldots, l_m, r_1, \ldots, r_m\})$
    else
      $k_1 := \exists quant(\{p_1, \ldots, p_k\}, \{l_1, \ldots, l_m\})$
      $k_2 := \exists quant(\{p_1, \ldots, p_k\}, \{r_1, \ldots, r_m\})$
      return $integrate(k_1, p, k_2, D)$
end

$\forall$-quantification algorithm for OBDDs

procedure $\forall quant(\{p_1, \ldots, p_k\}, \{n_1, \ldots, n_m\})$
parameters: global dag D
input: nodes $n_1, \ldots, n_m$ representing $F_1, \ldots, F_m$ in D
output: a node n representing $\forall p_1 \ldots \forall p_k (F_1 \wedge \ldots \wedge F_m)$ 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 $\forall quant(\{p_1, \ldots, p_k\}, \{n_1, \ldots, n_{i-1}, n_{i+1}, \ldots, n_m\})$
  $p := max\_atom(n_1, \ldots, n_m)$
  forall $i = 1 \ldots m$
    if n_i is labelled by p
      then $(l_i, r_i) := (neg(n_i), pos(n_i))$
      else $(l_i, r_i) := (n_i, n_i)$
  if $p \in \{p_1, \ldots, p_k\}$
    then **return** $\forall quant(\{p_1, \ldots, p_k\} - \{p\}, \{l_1, \ldots, l_m, r_1, \ldots, r_m\})$
    else
      $k_1 := \forall quant(\{p_1, \ldots, p_k\}, \{l_1, \ldots, l_m\})$
      $k_2 := \forall quant(\{p_1, \ldots, p_k\}, \{r_1, \ldots, r_m\})$
      return $integrate(k_1, p, k_2, D)$
end

Quantifier elimination

Lemma (quantifier elimination): For any quantified Boolean formula F there is an equivalent quantifier-free formula Q, ($F \equiv Q$).

Remarks:

- We can eliminate quantifiers from formulas one by one from innermost to outermost using OBDDs.
- In particular, we can evaluate/check satisfiability/validity of QBFs using quantification algorithms on OBDDs.
- Evaluation/satisfiability/validity of QBF is PSPACE-complete.

Summary

Quantified Boolean Formulas: boolean formulas + quantifiers $\exists, \forall$.

Any closed formula is either true or false (in all interpretations).

Satifiability/validity of formulas with free variables can be reduced to checking truth/falsity of closed formulas.

Prenex normal form: rectification + prenexing rules + CNF rules.

Alg. for checking truth/falsity of closed formulas in prenex form:

- Splitting: $\wedge, \vee$ nodes. Pure atom rule.
- DLL: splitting + unit popagation; $\wedge, \vee$ nodes.
  Pure literal rule, Universal literal rule.

QBF with free variables represent boolean functions.
Quantification algorithms for building OBDDs from QBFs.

Summary

Quantified Boolean Formulas: boolean formulas + quantifiers $\exists, \forall$.

Any closed formula is either true or false (in all interpretations).

Satifiability/validity of formulas with free variables can be reduced to checking truth/falsity of closed formulas.

Prenex normal form: rectification + prenexing rules + CNF rules.

Alg. for checking truth/falsity of closed formulas in prenex form:

- Splitting: $\wedge, \vee$ nodes. Pure atom rule.
- DLL: splitting + unit popagation; $\wedge, \vee$ nodes.
  Pure literal rule, Universal literal rule.

QBF with free variables represent boolean functions.
Quantification algorithms for building OBDDs from QBFs.

Next: Modelling using Propositional Logic of Finite Domains.