

# Splitting

---

Simplification rules for  $\top$ :

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

$$\top \wedge A_1 \wedge \dots \wedge A_n \Rightarrow A_1 \wedge \dots \wedge A_n$$

$$\top \vee A_1 \vee \dots \vee A_n \Rightarrow \top$$

$$A \rightarrow \top \Rightarrow \top \quad \top \rightarrow A \Rightarrow A$$

$$A \leftrightarrow \top \Rightarrow A \quad \top \leftrightarrow A \Rightarrow A$$

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

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

Simplification rules for  $\perp$ :

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

$$\perp \wedge A_1 \wedge \dots \wedge A_n \Rightarrow \perp$$

$$\perp \vee A_1 \vee \dots \vee A_n \Rightarrow A_1 \vee \dots \vee A_n$$

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

$$A \leftrightarrow \perp \Rightarrow \neg A \quad \perp \leftrightarrow A \Rightarrow \neg A$$

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

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

## Splitting algorithm

procedure *splitting*( $G$ )

input: closed rectified formula  $G$  ; output: 0 or 1

parameters: function *select\_literal*

begin

$G := \text{simplify}(G)$

if  $G = \perp$  then return 0 ; if  $G = \top$  then return 1

Let  $\exists\forall$  be a quantifier and  $P$  a set of variables such that  $G = \exists\forall P G_1$

$(p, b) := \text{select\_literal}(P, G)$

$G' := \exists\forall(P - \{p\})G_1$

if  $b = 0$  then  $(B_1, B_2) := (\perp, \top)$

else  $(B_1, B_2) := (\top, \perp)$

case  $(\text{splitting}((G')_p^{B_1}))$  of

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

$(0, \exists) \Rightarrow$  return  $\text{splitting}((G')_p^{B_2})$

$(1, \forall) \Rightarrow$  return  $\text{splitting}((G')_p^{B_2})$

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

end

## Pure literal rule

---

**Lemma** Let  $G$  be a prenex rectified closed formula

$$\exists \forall_1 P_1 \dots \exists \forall_n P_n \exists \forall p A.$$

1. If all occurrences of  $p$  in  $G$  are positive and  $\exists \forall = \exists$ , then  $G$  is equivalent to  $\exists \forall_1 P_1 \dots \exists \forall_n P_n A_p^\top$ .
2. If all occurrences of  $p$  in  $G$  are positive and  $\exists \forall = \forall$ , then  $G$  is equivalent to  $\exists \forall_1 P_1 \dots \exists \forall_n P_n A_p^\perp$ .
3. If all occurrences of  $p$  in  $G$  are negative and  $\exists \forall = \exists$ , then  $G$  is equivalent to  $\exists \forall_1 P_1 \dots \exists \forall_n P_n A_p^\perp$ .
4. If all occurrences of  $p$  in  $G$  are negative and  $\exists \forall = \forall$ , then  $G$  is equivalent to  $\exists \forall_1 P_1 \dots \exists \forall_n P_n A_p^\top$ .

## DLL algorithm

---

procedure  $DLL(Q, S)$

input: quantifier prefix  $Q = \exists \forall_1 P_1 \dots \exists \forall_n P_n$ , set of clauses  $S$

output: 0 or 1

parameters: function  $select\_literal$

begin

$S := propagate(S)$

if  $S$  is empty then return 1 ; if  $S$  contains  $\square$  then return 0

$L := select\_literal(\exists \forall_1 P_1, \exists \forall_2 P_2 \dots \exists \forall_n P_n, S)$

$Q_1 = \exists \forall_1 (P_1 - \{p\}) \exists \forall_2 P_2 \dots \exists \forall_n P_n$

case  $(DLL(Q_1, S \cup \{L\}), \exists \forall_1)$  of

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

$(0, \exists) \Rightarrow$  return  $DLL(Q_1, S \cup \{\tilde{L}\})$

$(1, \forall) \Rightarrow$  return  $DLL(Q_1, S \cup \{\tilde{L}\})$

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

end

## Quantifier elimination

---

procedure  $ex\_elim(P, \{n_1, \dots, n_m\})$

parameters: a dag  $D$  containing  $n_1, \dots, n_m$  and  $N_0, N_1$  as nodes

input: nodes  $n_1, \dots, n_m$  representing  $A_1, \dots, A_m$  in  $D$

output: a node  $n$  representing  $\exists P(A_1 \vee \dots \vee A_m)$  in (modified)  $D$

begin

if  $m = 0$  then return  $N_0$  ; if  $m = 1$  and  $P = \emptyset$  then return  $n_1$

if some  $n_i$  is a leaf labelled 0 then

return  $ex\_elim(P, \{n_1, \dots, n_{i-1}, n_{i+1}, \dots, n_m\})$

if some  $n_i$  is a leaf labelled 1 then return  $N_1$

$p := max\_atom(n_1, \dots, n_m)$

forall  $i = 1 \dots m$

if  $n_i$  is labelled by  $p$  then  $(l_i, r_i) := (left(n_i), right(n_i))$

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

if  $p \in P$  then return  $ex\_elim(P - \{p\}, \{l_1, \dots, l_m, r_1, \dots, r_m\})$

else  $(k_1, k_2) := (ex\_elim(P, \{l_1, \dots, l_m\}), ex\_elim(P, \{r_1, \dots, r_m\}))$

if  $k_1 = k_2$  then return  $k_1$

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

end