Exponential Separations in a Hierarchy of Clause Learning Proof Systems

Jan Johannsen

Institut für Informatik
LMU München

SAT 2013, Helsinki

Resolution

Clause: disjunction $a_1 \vee \ldots \vee a_k$ of literals $a_i = x$ or $a_i = \bar{x}$.

Resolution

Clause: disjunction $a_1 \vee \ldots \vee a_k$ of literals $a_i = x$ or $a_i = \bar{x}$.

The width of C is $w(C) := k$.

Resolution

Clause: disjunction $a_1 \vee \ldots \vee a_k$ of literals $a_i = x$ or $a_i = \bar{x}$.

The width of C is $w(C) := k$.

Formula (in CNF): conjunction $C_1 \wedge \ldots \wedge C_m$ of clauses.

Resolution

Clause: disjunction $a_1 \vee \ldots \vee a_k$ of literals $a_i = x$ or $a_i = \bar{x}$.

The width of C is $w(C) := k$.

Formula (in CNF): conjunction $C_1 \wedge \ldots \wedge C_m$ of clauses.

Resolution rule

If C, D are clauses with $x \in C$ and $\bar{x} \in D$, then

$$Res_x(C, D) := (C \setminus x) \vee (D \setminus \bar{x})$$

Resolution proofs

Definition

A Resolution derivation R of clause C from formula F is a dag labelled with clauses s.t.

Resolution proofs

Definition

A Resolution derivation R of clause C from formula F is a dag labelled with clauses s.t.

- there is exactly one sink labelled C

Resolution proofs

Definition

A Resolution derivation R of clause C from formula F is a dag labelled with clauses s.t.

- there is exactly one sink labelled C
- If v has predecessors u and u', then

  $$C_v = Res_x(C_u, C_{u'})$$

  for some variable x

Resolution proofs

Definition

A Resolution derivation R of clause C from formula F is a dag labelled with clauses s.t.

- there is exactly one sink labelled C
- If v has predecessors u and u', then

$$C_v = Res_x(C_u, C_{u'})$$

  for some variable x
- if v is a source, then $C_v \in F$

Resolution proofs

Definition

A Resolution derivation R of clause C from formula F is a dag labelled with clauses s.t.

- there is exactly one sink labelled C
- If v has predecessors u and u', then

  $$C_v = Res_x(C_u, C_{u'})$$

  for some variable x
- if v is a source, then $C_v \in F$

If the dag is a tree, we call R tree-like

Resolution proofs

Definition

A Resolution derivation R of clause C from formula F is a dag labelled with clauses s.t.

- there is exactly one sink labelled C
- If v has predecessors u and u', then

$$C_v = Res_x(C_u, C_{u'})$$

  for some variable x
- if v is a source, then $C_v \in F$

If the dag is a tree, we call R tree-like

A Resolution refutation of F is a derivation of the empty clause $\Box$ from F.

DPLL and Tree Resolution

Theorem

If unsatisfiable formula F is refuted by DPLL in s steps, then F has a tree-like resolution refutation R of size s.

DPLL and Tree Resolution

Theorem

If unsatisfiable formula F is refuted by DPLL in s steps, then F has a tree-like resolution refutation R of size s.

The converse also holds.

DPLL and Tree Resolution

Theorem

If unsatisfiable formula F is refuted by DPLL in s steps, then F has a tree-like resolution refutation R of size s.

The converse also holds.

Wanted: Similar correspondence for DPLL with clause learning.

Resolution Trees with Lemmas

A Resolution tree with lemmas (RTL) for formula F is an ordered binary tree labelled with clauses s.t.

Resolution Trees with Lemmas

A Resolution tree with lemmas (*RTL*) for formula F is an ordered binary tree labelled with clauses s.t.

- $C_{\text{root}} = \Box$

Resolution Trees with Lemmas

A Resolution tree with lemmas (RTL) for formula F is an ordered binary tree labelled with clauses s.t.

- $C_{\mathrm{root}} = \Box$
- if v has children u and u', then

$$C_v = Res_x(C_u, C_{u'}) \quad \text{for some variable } x$$

Resolution Trees with Lemmas

A Resolution tree with lemmas (RTL) for formula F is an ordered binary tree labelled with clauses s.t.

- $C_{\mathrm{root}} = \Box$
- if v has children u and u', then

  $C_v = Res_x(C_u, C_{u'})$ for some variable x
- if v is a leaf, then

  $C_v \in F$

Resolution Trees with Lemmas

A Resolution tree with lemmas (RTL) for formula F is an ordered binary tree labelled with clauses s.t.

- $C_{\mathrm{root}} = \square$
- if v has children u and u', then

  $C_v = Res_x(C_u, C_{u'})$ for some variable x
- if v is a leaf, then

  $C_v \in F$ or $C_v = C_u$ for some $u \prec v$ (lemma)

$\prec$ is the post-order on trees.

Clause learning and *RTL*

Theorem (Buss, Hoffmann, JJ 08)

If unsatisfiable formula F is refuted by DPLL+CL in s steps, then F has an RTL-refutation R of size $s \cdot n^{O(1)}$.

Clause learning and *RTL*

Theorem (Buss, Hoffmann, JJ 08)

If unsatisfiable formula F is refuted by DPLL+CL in s steps, then F has an RTL-refutation R of size $s \cdot n^{O(1)}$.

Moreover, the lemmas used in R are among the clauses learned by the algorithm.

Clause learning and *RTL*

Theorem (Buss, Hoffmann, JJ 08)

If unsatisfiable formula F is refuted by DPLL+CL in s steps, then F has an RTL-refutation R of size $s \cdot n^{O(1)}$.

Moreover, the lemmas used in R are among the clauses learned by the algorithm.

In fact, the paper defines a subsystem $WRTI < RTL$ for which also the converse holds.

Clause learning and RTL

Theorem (Buss, Hoffmann, JJ 08)

If unsatisfiable formula F is refuted by DPLL+CL in s steps, then F has an RTL-refutation R of size $s \cdot n^{O(1)}$.

Moreover, the lemmas used in R are among the clauses learned by the algorithm.

In fact, the paper defines a subsystem $WRTI < RTL$ for which also the converse holds.

A refutation R in RTL is in $RTL(k)$, if every lemma C used in R is of width $w(C) \leq k$.

Previous lower bounds

Theorem (BHJ 08)

Every $RTL(n/2)$-refutation of PHP_n is of size $2^{\Omega(n \log n)}$.

Previous lower bounds

Theorem (BHJ 08)

Every $RTL(n/2)$*-refutation of* PHP_n *is of size* $2^{\Omega(n \log n)}$.

Theorem (JJ 09)

Every $RTL(n/4)$*-refutation of* Ord_n *is of size* $2^{\Omega(n)}$.

Previous lower bounds

Theorem (BHJ 08)

Every $RTL(n/2)$*-refutation of* PHP_n *is of size* $2^{\Omega(n \log n)}$.

Theorem (JJ 09)

Every $RTL(n/4)$*-refutation of* Ord_n *is of size* $2^{\Omega(n)}$.

Theorem (Ben-Sasson, JJ 10)

If resolution refutations of F *require width* w,
then every $RTL(k)$*-refutation of* F *is of size* 2^{w-2k}.

Previous lower bounds

Theorem (BHJ 08)

Every $RTL(n/2)$-refutation of PHP_n is of size $2^{\Omega(n \log n)}$.

Theorem (JJ 09)

Every $RTL(n/4)$-refutation of Ord_n is of size $2^{\Omega(n)}$.

Theorem (Ben-Sasson, JJ 10)

If resolution refutations of F require width w,
then every $RTL(k)$-refutation of F is of size 2^{w-2k}.

Here we show:

Theorem

For every k, there are formulas $F_n^{(k)}$ such that

Previous lower bounds

Theorem (BHJ 08)

Every $RTL(n/2)$-refutation of PHP_n is of size $2^{\Omega(n \log n)}$.

Theorem (JJ 09)

Every $RTL(n/4)$-refutation of Ord_n is of size $2^{\Omega(n)}$.

Theorem (Ben-Sasson, JJ 10)

If resolution refutations of F require width w,
then every $RTL(k)$-refutation of F is of size 2^{w-2k}.

Here we show:

Theorem

For every k, there are formulas $F_n^{(k)}$ such that

- $F_n^{(k)}$ *have $RTL(k+1)$-refutations of size $n^{O(1)}$.*

Previous lower bounds

Theorem (BHJ 08)

Every $RTL(n/2)$*-refutation of* PHP_n *is of size* $2^{\Omega(n \log n)}$.

Theorem (JJ 09)

Every $RTL(n/4)$*-refutation of* Ord_n *is of size* $2^{\Omega(n)}$.

Theorem (Ben-Sasson, JJ 10)

If resolution refutations of F *require width* w,
then every $RTL(k)$*-refutation of* F *is of size* 2^{w-2k}.

Here we show:

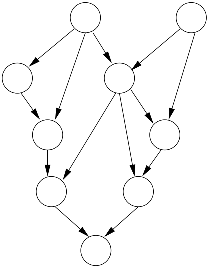
Theorem

For every k*, there are formulas* $F_n^{(k)}$ *such that*

- $F_n^{(k)}$ *have* $RTL(k+1)$*-refutations of size* $n^{O(1)}$.
- $F_n^{(k)}$ *requires* $RTL(k)$*-refutations of size* $2^{\Omega(n/\log n)}$.

Graph Pebbling

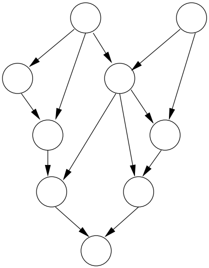
Pointed DAG: G with in-degree 2 and one sink t.



Graph Pebbling

Pointed DAG: G with in-degree 2 and one sink t.

Pebble game on G:

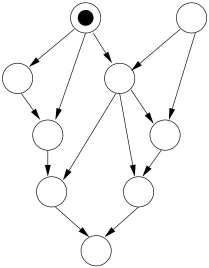


Graph Pebbling

Pointed DAG: G with in-degree 2 and one sink t.

Pebble game on G:

- put pebble on any source

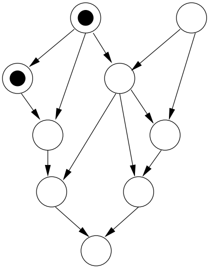


Graph Pebbling

Pointed DAG: G with in-degree 2 and one sink t.

Pebble game on G:

- put pebble on any source
- put pebble on any vertex where both predecessors have a pebble

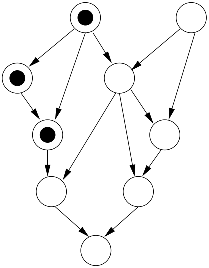


Graph Pebbling

Pointed DAG: G with in-degree 2 and one sink t.

Pebble game on G:

- put pebble on any source
- put pebble on any vertex where both predecessors have a pebble

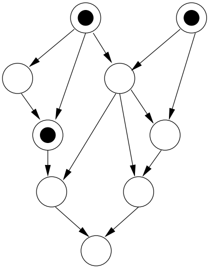


Graph Pebbling

Pointed DAG: G with in-degree 2 and one sink t.

Pebble game on G:

- put pebble on any source
- put pebble on any vertex where both predecessors have a pebble
- remove any pebble

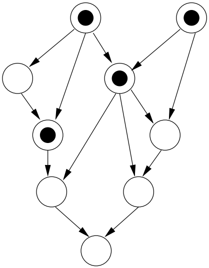


Graph Pebbling

Pointed DAG: G with in-degree 2 and one sink t.

Pebble game on G:

- put pebble on any source
- put pebble on any vertex where both predecessors have a pebble
- remove any pebble

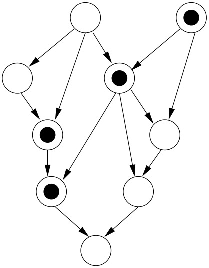


Graph Pebbling

Pointed DAG: G with in-degree 2 and one sink t.

Pebble game on G:

- put pebble on any source
- put pebble on any vertex where both predecessors have a pebble
- remove any pebble

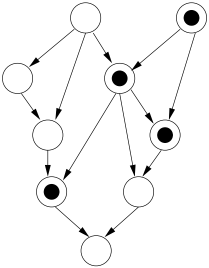


Graph Pebbling

Pointed DAG: G with in-degree 2 and one sink t.

Pebble game on G:

- put pebble on any source
- put pebble on any vertex where both predecessors have a pebble
- remove any pebble

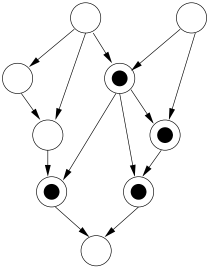


Graph Pebbling

Pointed DAG: G with in-degree 2 and one sink t.

Pebble game on G:

- put pebble on any source
- put pebble on any vertex where both predecessors have a pebble
- remove any pebble



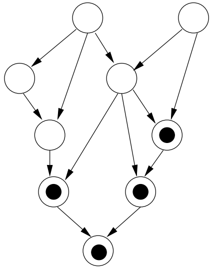
Graph Pebbling

Pointed DAG: G with in-degree 2 and one sink t.

Pebble game on G:

- put pebble on any source
- put pebble on any vertex where both predecessors have a pebble
- remove any pebble

until a pebble is put on t.



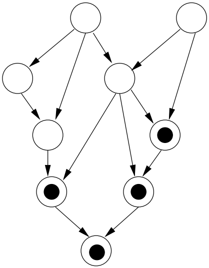
Graph Pebbling

Pointed DAG: G with in-degree 2 and one sink t.

Pebble game on G:

- put pebble on any source
- put pebble on any vertex where both predecessors have a pebble
- remove any pebble

until a pebble is put on t.



Pebbling number $\mathrm{Peb}(G)$: min. # of pebbles in game on G.

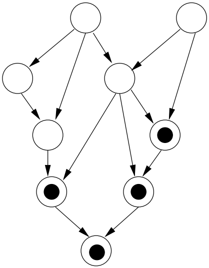
Graph Pebbling

Pointed DAG: G with in-degree 2 and one sink t.

Pebble game on G:

- put pebble on any source
- put pebble on any vertex where both predecessors have a pebble
- remove any pebble

until a pebble is put on t.



Pebbling number $\text{Peb}(G)$: min. # of pebbles in game on G.

Theorem (Celoni, Paul, Tarjan 1977)

There are dags G_n of size n with $\text{Peb}(G_n) \geq \Omega(n/\log n)$.

Pebbling formulas

Clauses $Imp(G)$:

x_v	for every source v
$x_u \wedge x_{u'} \to x_v$	for $(u, v), (u', v) \in E$
$\bar{x}_t$	for the sink t

Pebbling formulas

Clauses $Imp(G)$:

x_v	for every source v
$x_u \wedge x_{u'} \rightarrow x_v$	for $(u, v), (u', v) \in E$
$\bar{x}_t$	for the sink t

$Imp^2(G)$: replace every x_v in $Imp(G)$ by $x_{v,1} \vee x_{v,2}$

Pebbling formulas

Clauses $\mathit{Imp}(G)$:

x_v	for every source v
$x_u \wedge x_{u'} \rightarrow x_v$	for $(u, v), (u', v) \in E$
$\bar{x}_t$	for the sink t

$\mathit{Imp}^2(G)$: replace every x_v in $\mathit{Imp}(G)$ by $x_{v,1} \vee x_{v,2}$

Theorem (Ben-Sasson et al. 2004)

Every tree resolution refutation of $\mathit{Imp}^2(G)$ is of size $2^{\Omega(\mathrm{Peb}(G))}$.

XORification

- $X(x, k) = x_1 \oplus \ldots \oplus x_k$ for a variable x.

XORification

- $X(x, k) = x_1 \oplus \ldots \oplus x_k$ for a variable x.
- $X(\bar{x}, k) = X(x, k) \oplus 1$ for a negated variable $\bar{x}$.

XORification

- $X(x,k) = x_1 \oplus \ldots \oplus x_k$ for a variable x.
- $X(\bar{x},k) = X(x,k) \oplus 1$ for a negated variable $\bar{x}$.
- $X(C,k) = \bigvee_{a \in C} X(a,k)$ expanded into CNF, for a clause C.

XORification

- $X(x,k) = x_1 \oplus \ldots \oplus x_k$ for a variable x.
- $X(\bar{x},k) = X(x,k) \oplus 1$ for a negated variable $\bar{x}$.
- $X(C,k) = \bigvee_{a \in C} X(a,k)$ expanded into CNF, for a clause C.
- $X(F,k) = \bigwedge_{C \in F} X(C,k)$ for a CNF formula F.

XORification

- $X(x, k) = x_1 \oplus \ldots \oplus x_k$ for a variable x.
- $X(\bar{x}, k) = X(x, k) \oplus 1$ for a negated variable $\bar{x}$.
- $X(C, k) = \bigvee_{a \in C} X(a, k)$ expanded into CNF, for a clause C.
- $X(F, k) = \bigwedge_{C \in F} X(C, k)$ for a CNF formula F.

We write $Imp^{\oplus k}(G)$ for $X(Imp(G), k)$.

XORification

- $X(x,k) = x_1 \oplus \ldots \oplus x_k$ for a variable x.
- $X(\bar{x},k) = X(x,k) \oplus 1$ for a negated variable $\bar{x}$.
- $X(C,k) = \bigvee_{a \in C} X(a,k)$ expanded into CNF, for a clause C.
- $X(F,k) = \bigwedge_{C \in F} X(C,k)$ for a CNF formula F.

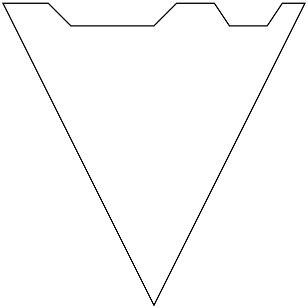
We write $Imp^{\oplus k}(G)$ for $X(Imp(G),k)$.

Theorem

For every G of size n,
$Imp^{\oplus k}(G)$ has $RTL(k)$-refutations of size $O(2^{3k}n)$.

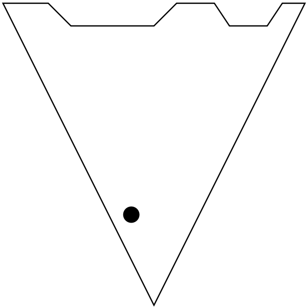
Proof of the lower bound

- Let R be a refutation of $Imp^{\oplus(k+1)}(G)$



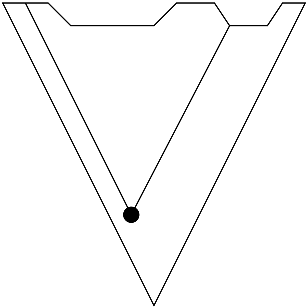
Proof of the lower bound

- Let R be a refutation of $Imp^{\oplus(k+1)}(G)$
- Find first C with $w(C) \leq k$



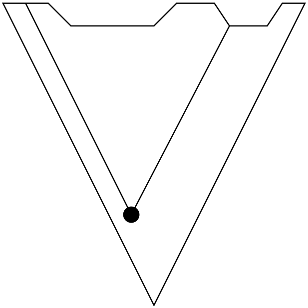
Proof of the lower bound

- Let R be a refutation of $Imp^{\oplus(k+1)}(G)$
- Find first C with $w(C) \leq k$
- Subtree R_C is tree-like derivation of C



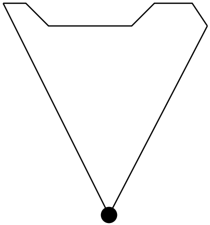
Proof of the lower bound

- Let R be a refutation of $Imp^{\oplus(k+1)}(G)$
- Find first C with $w(C) \leq k$
- Subtree R_C is tree-like derivation of C
- Pick ρ with $C\lceil\rho = 0$



Proof of the lower bound

- Let R be a refutation of $Imp^{\oplus(k+1)}(G)$
- Find first C with $w(C) \leq k$
- Subtree R_C is tree-like derivation of C
- Pick ρ with $C\lceil\rho = 0$
- $R_C\lceil\rho$ is refutation of $Imp^{\oplus(k+1)}(G)\lceil\rho$



Generalized XORification

Let $\beta : V \to \{0,1\}^2$, with $\beta(x) = (\beta_0(x), \beta_1(x))$.

Generalized XORification

Let $\beta : V \to \{0,1\}^2$, with $\beta(x) = (\beta_0(x), \beta_1(x))$.

- $X(x, k, \beta) = x_1 \oplus \ldots \oplus x_k \oplus \beta_1(x)$ for a variable x with $\beta_0(x) = 0$.

Generalized XORification

Let $\beta : V \to \{0,1\}^2$, with $\beta(x) = (\beta_0(x), \beta_1(x))$.

- $X(x,k,\beta) = x_1 \oplus \ldots \oplus x_k \oplus \beta_1(x)$ for a variable x with $\beta_0(x) = 0$.
- $X(x,k,\beta) = x_1 \oplus \beta_1(x)$ for a variable x with $\beta_0(x) = 1$.

Generalized XORification

Let $\beta : V \to \{0,1\}^2$, with $\beta(x) = (\beta_0(x), \beta_1(x))$.

- $X(x,k,\beta) = x_1 \oplus \ldots \oplus x_k \oplus \beta_1(x)$ for a variable x with $\beta_0(x) = 0$.
- $X(x,k,\beta) = x_1 \oplus \beta_1(x)$ for a variable x with $\beta_0(x) = 1$.
- $X(\bar{x},k,\beta) = X(x,k,\beta) \oplus 1$ for a neg. variable $\bar{x}$.

Generalized XORification

Let $\beta : V \to \{0,1\}^2$, with $\beta(x) = (\beta_0(x), \beta_1(x))$.

- $X(x,k,\beta) = x_1 \oplus \ldots \oplus x_k \oplus \beta_1(x)$ for a variable x with $\beta_0(x) = 0$.
- $X(x,k,\beta) = x_1 \oplus \beta_1(x)$ for a variable x with $\beta_0(x) = 1$.
- $X(\bar{x},k,\beta) = X(x,k,\beta) \oplus 1$ for a neg. variable $\bar{x}$.
- $X(C,k,\beta) = \bigvee_{a \in C} X(a,k,\beta)$ expanded into CNF, for a clause C

Generalized XORification

Let $\beta : V \to \{0,1\}^2$, with $\beta(x) = (\beta_0(x), \beta_1(x))$.

- $X(x,k,\beta) = x_1 \oplus \ldots \oplus x_k \oplus \beta_1(x)$ for a variable x with $\beta_0(x) = 0$.
- $X(x,k,\beta) = x_1 \oplus \beta_1(x)$ for a variable x with $\beta_0(x) = 1$.
- $X(\bar{x},k,\beta) = X(x,k,\beta) \oplus 1$ for a neg. variable $\bar{x}$.
- $X(C,k,\beta) = \bigvee_{a \in C} X(a,k,\beta)$ expanded into CNF, for a clause C
- $X(F,k,\beta) = \bigwedge_{C \in F} X(C,k,\beta)$ for a CNF formula F

Generalized XORification

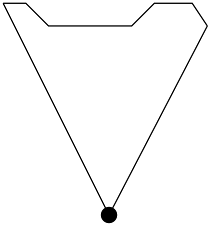
Let $\beta : V \to \{0,1\}^2$, with $\beta(x) = (\beta_0(x), \beta_1(x))$.

- $X(x,k,\beta) = x_1 \oplus \ldots \oplus x_k \oplus \beta_1(x)$ for a variable x with $\beta_0(x) = 0$.
- $X(x,k,\beta) = x_1 \oplus \beta_1(x)$ for a variable x with $\beta_0(x) = 1$.
- $X(\bar{x},k,\beta) = X(x,k,\beta) \oplus 1$ for a neg. variable $\bar{x}$.
- $X(C,k,\beta) = \bigvee_{a \in C} X(a,k,\beta)$ expanded into CNF, for a clause C
- $X(F,k,\beta) = \bigwedge_{C \in F} X(C,k,\beta)$ for a CNF formula F

We write $Imp_\beta^{\oplus k}(G)$ for $X(Imp(G),k,\beta)$

Proof of the lower bound

- Let R be a refutation of $Imp^{\oplus(k+1)}(G)$
- Find first C with $w(C) \leq k$
- Subtree R_C is tree-like derivation of C
- Pick ρ with $C\lceil\rho = 0$
- $R_C\lceil\rho$ is refutation of $Imp^{\oplus(k+1)}(G)\lceil\rho$
- $Imp^{\oplus(k+1)}(G)\lceil\rho = Imp_{\beta}^{\oplus 2}(G)$



Lower bound

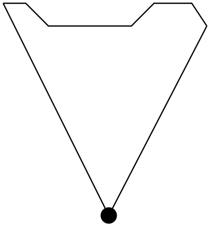
Generalization of the lower bound by Ben-Sasson et al.:

Theorem

Every tree resolution refutation of $Imp_{\beta}^{\oplus 2}(G)$ *is of size* $2^{\Omega(\mathrm{Peb}(G) - a(\beta))}$, *where* $a(\beta) := \#\{\, v \,;\, \beta_0(x_v) = 1 \,\}$

Proof of the lower bound

- Let R be a refutation of $Imp^{\oplus(k+1)}(G)$
- Find first C with $w(C) \leq k$
- Subtree R_C is tree-like derivation of C
- Pick ρ with $C \lceil \rho = 0$
- $R_C \lceil \rho$ is refutation of $Imp^{\oplus(k+1)}(G) \lceil \rho$
- $Imp^{\oplus(k+1)}(G) \lceil \rho = Imp_{\beta}^{\oplus 2}(G)$ for a β with $a(\beta) \leq 1$
- lower bound shows $|R| \geq |R_C| \geq 2^{\Omega(\mathrm{Peb}(G))}$



Lower bound

Generalization of the lower bound by Ben-Sasson et al.:

Theorem

Every tree resolution refutation of $Imp_\beta^{\oplus 2}(G)$ *is of size* $2^{\Omega(\mathrm{Peb}(G) - a(\beta))}$, *where* $a(\beta) := \#\{\, v\,;\, \beta_0(x_v) = 1 \,\}$

Theorem

For every k, *the formulas* $Imp^{\oplus(k+1)}(G_n)$

Lower bound

Generalization of the lower bound by Ben-Sasson et al.:

Theorem

Every tree resolution refutation of $Imp_{\beta}^{\oplus 2}(G)$ is of size $2^{\Omega(\mathrm{Peb}(G)-a(\beta))}$, where $a(\beta) := \#\{\, v \,;\, \beta_0(x_v) = 1 \,\}$

Theorem

For every k, the formulas $Imp^{\oplus(k+1)}(G_n)$

- *have $RTL(k+1)$-refutations of size $n^{O(1)}$.*

Lower bound

Generalization of the lower bound by Ben-Sasson et al.:

Theorem

Every tree resolution refutation of $Imp_{\beta}^{\oplus 2}(G)$ is of size $2^{\Omega(\mathrm{Peb}(G) - a(\beta))}$, where $a(\beta) := \#\{\, v\,;\, \beta_0(x_v) = 1 \,\}$

Theorem

For every k, the formulas $Imp^{\oplus(k+1)}(G_n)$

- *have $RTL(k+1)$-refutations of size $n^{O(1)}$.*
- *require $RTL(k)$-refutations of size $2^{\Omega(n/\log n)}$.*