Strong (D)QBF Dependency Schemes via Tautology-free Resolution Paths

Olaf Beyersdorff Joshua Blinkhorn **Tomáš Peitl**

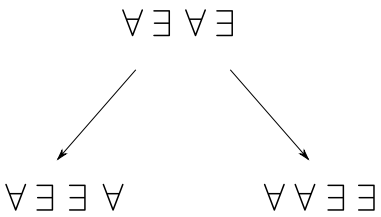
Friedrich-Schiller-Universität Jena, Germany

June 25, 2020

Dependencies

$$\forall \exists \forall \exists$$

Dependencies



Dependencies

DQBF

We consider (closed, prenex) *dependency quantified Boolean formulas* of the following form (a.k.a. *S-form DQBF*):

$$\Psi = \underbrace{\overbrace{\forall u_1}^{\text{universal variable}} \cdots \forall u_m \quad \overbrace{\exists x_1}^{\text{existential variable}}(S_{x_1}) \cdots \exists x_n \overbrace{(S_{x_n})}^{\text{support set}}}_{\text{prefix}} \cdot \underbrace{C_1 \wedge \cdots \wedge C_r}_{\text{matrix}}$$

$$\underbrace{u_1 \vee \overbrace{\neg x_2}^{\text{literal}} \vee \neg u_3 \vee x_4}_{\text{clause}}$$

A DQBF is *true* if there exist functions $f_{x_i} : \{0,1\}^{S_{x_i}} \to \{0,1\}$ whose substitution for x_i yields a propositional tautology.

QBF

- DQBF extends QBF:

$$\Phi = \overbrace{\forall U_1}^{\text{block}} \exists X_1 \forall U_2 \exists X_2 \cdots \forall U_k \exists X_k \cdot C_1 \wedge \cdots \wedge C_r$$

- If $x_i \in X_i$, then $S_{x_i} = \bigcup_{j<i} U_j$.
- A DQBF is a QBF if and only if the support sets are linearly ordered under inclusion.

Applications

- Deciding whether a given QBF is true is PSPACE-complete.

Applications

- Deciding whether a given QBF is true is PSPACE-complete.
- Deciding whether a given DQBF is true is NEXP-complete.

Applications

- Deciding whether a given QBF is true is PSPACE-complete.
- Deciding whether a given DQBF is true is NEXP-complete.
- DQBFs can be used to model various real-world problems arising in areas such as formal verification, synthesis, automated design of circuits, or games such as chess.

Applications

- Deciding whether a given QBF is true is PSPACE-complete.
- Deciding whether a given DQBF is true is NEXP-complete.
- DQBFs can be used to model various real-world problems arising in areas such as formal verification, synthesis, automated design of circuits, or games such as chess.
- We are interested in solving DQBFs as efficiently as possible.

Spurious Dependencies

- Consider the formula $\forall u \exists x(\{u\}) \cdot (x \vee u) \wedge (x \vee \neg u)$.

Spurious Dependencies

- Consider the formula $\forall u \exists x(\{u\}) \cdot (x \vee u) \wedge (x \vee \neg u)$.
- It is obviously true by setting $x := 1$.

Spurious Dependencies

- Consider the formula $\forall u \exists x(\{u\}) \cdot (x \vee u) \wedge (x \vee \neg u)$.
- It is obviously true by setting $x := 1$.
- But that does not need the dependency on u.

Spurious Dependencies

- Consider the formula $\forall u \exists x(\{u\}) \cdot (x \vee u) \wedge (x \vee \neg u)$.
- It is obviously true by setting $x := 1$.
- But that does not need the dependency on u.
- Hence, the dependency of x on u is spurious.

Dependency Schemes

- A *dependency scheme* as defined for QBF is a mapping:

$$D : \Phi \mapsto D(\Phi) \subseteq \mathcal{D}^{\text{trv}}(\Phi) = \{(x, y) \mid x < y\}$$

Dependency Schemes

- A *dependency scheme* as defined for QBF is a mapping:

$$D : \Phi \mapsto D(\Phi) \subseteq \mathcal{D}^{\text{trv}}(\Phi) = \{(x, y) \mid x < y\}$$

- Prominent dependency schemes are the *standard* $\mathcal{D}^{\text{std}}$ and the *reflexive resolution-path* $\mathcal{D}^{\text{rrs}}$;

Dependency Schemes

- A *dependency scheme* as defined for QBF is a mapping:

$$D : \Phi \mapsto D(\Phi) \subseteq \mathcal{D}^{\text{trv}}(\Phi) = \{(x, y) \mid x < y\}$$

- Prominent dependency schemes are the *standard* $\mathcal{D}^{\text{std}}$ and the *reflexive resolution-path* $\mathcal{D}^{\text{rrs}}$;
- First proposed by Samer and Szeider for backdoor sets, the definition has since evolved to accommodate different use cases; each of the following tools supports a dependency scheme in some form: DepQBF, Qute, HQSpre, CaQE, Qesto;

Dependency Schemes

- A *dependency scheme* as defined for QBF is a mapping:

$$D : \Phi \mapsto D(\Phi) \subseteq \mathcal{D}^{\text{trv}}(\Phi) = \{(x, y) \mid x < y\}$$

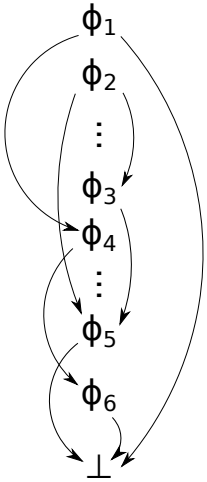
- Prominent dependency schemes are the *standard* $\mathcal{D}^{\text{std}}$ and the *reflexive resolution-path* $\mathcal{D}^{\text{rrs}}$;
- First proposed by Samer and Szeider for backdoor sets, the definition has since evolved to accommodate different use cases; each of the following tools supports a dependency scheme in some form: DepQBF, Qute, HQSpre, CaQE, Qesto;
- Because dependency schemes were created for QBF, dependencies are defined both ways. This turned out unnecessary in the analysis of refutational proof systems, and becomes meaningless in DQBF.

Proof Systems

- A *proof system* is a set of rules that prescribe how to derive new clauses from existing ones

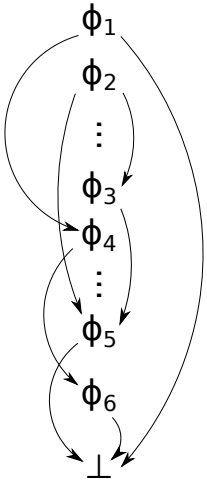
Proof Systems

- A *proof system* is a set of rules that prescribe how to derive new clauses from existing ones
- A *derivation* in a proof system is a sequence of clauses each of which can be derived from previous clauses using the rules



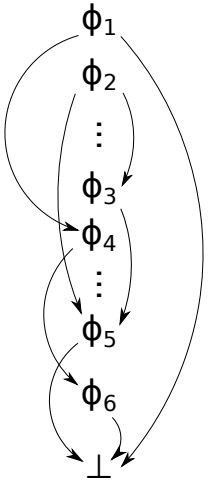
Proof Systems

- A *proof system* is a set of rules that prescribe how to derive new clauses from existing ones
- A *derivation* in a proof system is a sequence of clauses each of which can be derived from previous clauses using the rules
- A *refutation* is a derivation of the empty clause



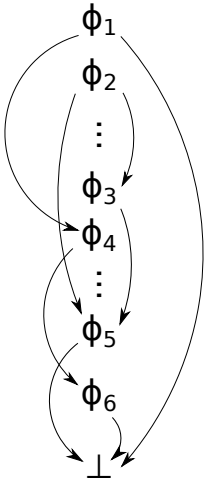
Proof Systems

- A *proof system* is a set of rules that prescribe how to derive new clauses from existing ones
- A *derivation* in a proof system is a sequence of clauses each of which can be derived from previous clauses using the rules
- A *refutation* is a derivation of the empty clause
- In particular, we are interested in $\forall$Exp+Res and Q-Res



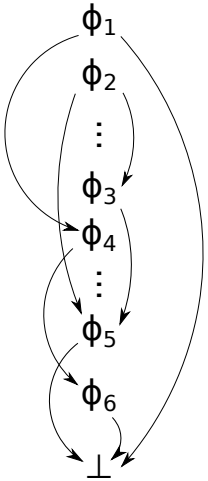
Proof Systems

- A *proof system* is a set of rules that prescribe how to derive new clauses from existing ones
- A *derivation* in a proof system is a sequence of clauses each of which can be derived from previous clauses using the rules
- A *refutation* is a derivation of the empty clause
- In particular, we are interested in $\forall$Exp+Res and Q-Res
- A Q-Res refutation is a sequence of clauses that are either existential resolvents or universal reducts;



Proof Systems

- A *proof system* is a set of rules that prescribe how to derive new clauses from existing ones
- A *derivation* in a proof system is a sequence of clauses each of which can be derived from previous clauses using the rules
- A *refutation* is a derivation of the empty clause
- In particular, we are interested in ∀Exp+Res and Q-Res
- A Q-Res refutation is a sequence of clauses that are either existential resolvents or universal reducts;
- A ∀Exp+Res refutation is a resolution refutation of the universally expanded formula (a.k.a. Shannon expansion);



Sound Use of Dependency Schemes

- Dependency analysis using a dependency scheme $\mathcal{D}$ in reasoning based on a proof system P is captured by adding $\mathcal{D}$ to P, resulting in a proof system P($\mathcal{D}$);

Sound Use of Dependency Schemes

- Dependency analysis using a dependency scheme $\mathcal{D}$ in reasoning based on a proof system P is captured by adding $\mathcal{D}$ to P, resulting in a proof system P($\mathcal{D}$);
- The goal is to show that P($\mathcal{D}$) is sound and stronger than just P;

Sound Use of Dependency Schemes

- Dependency analysis using a dependency scheme $\mathcal{D}$ in reasoning based on a proof system P is captured by adding $\mathcal{D}$ to P, resulting in a proof system P($\mathcal{D}$);
- The goal is to show that P($\mathcal{D}$) is sound and stronger than just P;
- Defining P($\mathcal{D}$) and proving its soundness can be highly non-trivial.

Sound Use of Dependency Schemes

- Dependency analysis using a dependency scheme $\mathcal{D}$ in reasoning based on a proof system P is captured by adding $\mathcal{D}$ to P, resulting in a proof system P($\mathcal{D}$);
- The goal is to show that P($\mathcal{D}$) is sound and stronger than just P;
- Defining P($\mathcal{D}$) and proving its soundness can be highly non-trivial.

Theorem ([SS16])

A QBF is false if, and only if, it has a Q($\mathcal{D}^{\text{rrs}}$,$\mathcal{D}^{\text{std}}$)-Res refutation.

Recap

We are

Recap

We are

- trying to solve a DQBF;

Recap

We are

- trying to solve a DQBF;
- identify as many spurious dependencies as possible;

Recap

We are

- trying to solve a DQBF;
- identify as many spurious dependencies as possible;
- while maintaining soundness of the proof system.

Overview of Contributions

Overview of Contributions

1. A clean DQBF-centric definition of dependency schemes along with a *characterisation* of when a dependency scheme can be used in *any* DQBF proof system;

Overview of Contributions

1. A clean DQBF-centric definition of dependency schemes along with a *characterisation* of when a dependency scheme can be used in *any* DQBF proof system;
2. A new, *tautology-free* dependency scheme $\mathcal{D}^{\mathrm{tf}}$ that generalizes the to-date strongest known *resolution-path* dependency scheme;

Overview of Contributions

1. A clean DQBF-centric definition of dependency schemes along with a *characterisation* of when a dependency scheme can be used in *any* DQBF proof system;
2. A new, *tautology-free* dependency scheme $\mathcal{D}^{\mathrm{tf}}$ that generalizes the to-date strongest known *resolution-path* dependency scheme;
3. *DQBF-genuine* exponential separations of $\forall$Exp+Res with and without $\mathcal{D}^{\mathrm{rrs}}$ and $\mathcal{D}^{\mathrm{tf}}$;

Overview of Contributions

1. A clean DQBF-centric definition of dependency schemes along with a *characterisation* of when a dependency scheme can be used in *any* DQBF proof system;
2. A new, *tautology-free* dependency scheme $\mathcal{D}^{\mathrm{tf}}$ that generalizes the to-date strongest known *resolution-path* dependency scheme;
3. *DQBF-genuine* exponential separations of $\forall$Exp+Res with and without $\mathcal{D}^{\mathrm{rrs}}$ and $\mathcal{D}^{\mathrm{tf}}$;
4. *QBF-genuine* exponential separations of Q-Res with $\mathcal{D}^{\mathrm{rrs}}$ and with $\mathcal{D}^{\mathrm{tf}}$.

DQBF Dependency Schemes

Dependency Schemes for DQBF

We define a DQBF dependency scheme $\mathcal{D}$ as a mapping between DQBFs that

Dependency Schemes for DQBF

We define a DQBF dependency scheme $\mathcal{D}$ as a mapping between DQBFs that

1. preserves the matrix and does not enlarge support sets:

$$\mathcal{D}(\Psi) \leq \Psi$$

Dependency Schemes for DQBF

We define a DQBF dependency scheme $\mathcal{D}$ as a mapping between DQBFs that

1. preserves the matrix and does not enlarge support sets:

$$\mathcal{D}(\Psi) \leq \Psi$$

2. is polynomial-time computable.

Dependency Schemes for DQBF

We define a DQBF dependency scheme $\mathcal{D}$ as a mapping between DQBFs that

1. preserves the matrix and does not enlarge support sets:

$$\mathcal{D}(\Psi) \leq \Psi$$

2. is polynomial-time computable.

We say that a dependency scheme $\mathcal{D}$ is *fully exhibited* if $\mathcal{D}(\Psi) \overset{\text{tr}}{\equiv} \Psi$ for every DQBF Ψ.

Parameterising Proof Systems

Definition (P($\mathcal{D}$))

Let P be a DQBF proof system and let $\mathcal{D}$ be a dependency scheme. A P($\mathcal{D}$) refutation of a DQBF Ψ is a P refutation of $\mathcal{D}(\Psi)$.

Proposition

Given a DQBF proof system P *and a dependency scheme* $\mathcal{D}$, P($\mathcal{D}$) *is sound and complete if, and only if,* $\mathcal{D}$ *is fully exhibited.*

The Tautology-free Dependency Scheme

Definition ($\mathcal{D}^{\mathrm{rrs}}$ [SS16] and $\mathcal{D}^{\mathrm{tf}}$)

The *reflexive resolution path dependency scheme* ($\mathcal{D}^{\mathrm{rrs}}$) is defined as the mapping $\Psi \mapsto \Psi'$, where

$$\begin{aligned} \Psi &:= \forall u_1 \cdots \forall u_m \exists x_1(S_{x_1}) \cdots \exists x_n(S_{x_n}) \cdot \psi\,, \\ \Psi' &:= \forall u_1 \cdots \forall u_m \exists x_1(S'_{x_1}) \cdots \exists x_n(S'_{x_n}) \cdot \psi\,, \end{aligned}$$

and S'_{x_i} is the set of universal variables $u \in S_{x_i}$ for which there exists a sequence $C_1, \ldots, C_k$ of clauses in ψ and a sequence $p_1, \ldots, p_{k-1}$ of existential literals satisfying the following conditions:

(a) $u \in C_1$ and $\overline{u} \in C_k$;

(b) for some $j \in [k-1]$, $x_i = \mathrm{var}(p_j)$;

(c) for each $j \in [k-1]$, $p_j \in C_j$, $\overline{p}_j \in C_{j+1}$, and $u \in S_{\mathrm{var}(p_j)}$;

(d) for each $j \in [k-2]$, $\mathrm{var}(p_j) \neq \mathrm{var}(p_{j+1})$.

The *tautology-free dependency scheme* ($\mathcal{D}^{\mathrm{tf}}$) adds to $\mathcal{D}^{\mathrm{rrs}}$ the condition

(e) for each $j \in [k-1]$, $(C_j \cup C_{j+1})\restriction_{I_\exists(\Psi)}$ is non-tautological.

$\mathcal{D}^{\text{tf}}$ Intuition

- $\mathcal{D}^{\text{rrs}}$ identifies potential information flows between variables as resolution paths;

$\mathcal{D}^{\mathrm{tf}}$ Intuition

- $\mathcal{D}^{\mathrm{rrs}}$ identifies potential information flows between variables as resolution paths;
- A resolution path is a sequence of clauses which can trigger unit propagation under a suitable assignment;

$\mathcal{D}^{\mathrm{tf}}$ Intuition

- $\mathcal{D}^{\mathrm{rrs}}$ identifies potential information flows between variables as resolution paths;
- A resolution path is a sequence of clauses which can trigger unit propagation under a suitable assignment;
- If a resolution path connects u and x, then assigning u may affect the choices for x;

$\mathcal{D}^{\mathrm{tf}}$ Intuition

- $\mathcal{D}^{\mathrm{rrs}}$ identifies potential information flows between variables as resolution paths;
- A resolution path is a sequence of clauses which can trigger unit propagation under a suitable assignment;
- If a resolution path connects u and x, then assigning u may affect the choices for x;
- However, certain resolution paths are blocked: they contain tautologies on variables that are "already assigned at the time" u is assigned, such as the *independent* existential variables $I_{\exists}(\Psi)$.

Example ($\mathcal{D}^{\text{rrs}}$ vs. $\mathcal{D}^{\text{tf}}$)

$$\forall u \exists x(\emptyset) \exists z(\{u\}) \cdot (x \vee u \vee z) \wedge (\neg x \vee \neg u \vee \neg z)$$

Example ($\mathcal{D}^{\text{rrs}}$ vs. $\mathcal{D}^{\text{tf}}$)

$$\forall u \exists x(\emptyset) \exists z(\{u\}) \cdot (x \vee u \vee z) \wedge (\neg x \vee \neg u \vee \neg z)$$

- The two clauses $(x \vee u \vee z)$ and $(\neg x \vee \neg u \vee \neg z)$ constitute a resolution path that connects u and z. Indeed, if x is set to false, the first clause simplifies to the implication $\neg u \implies z$, and if x is set to true, the second clause simplifies to $u \implies \neg z$. The value of u may potentially force either value of z.

Example ($\mathcal{D}^{\text{rrs}}$ vs. $\mathcal{D}^{\text{tf}}$)

$$\forall u \exists x(\emptyset) \exists z(\{u\}) \cdot (x \vee u \vee z) \wedge (\neg x \vee \neg u \vee \neg z)$$

- The two clauses $(x \vee u \vee z)$ and $(\neg x \vee \neg u \vee \neg z)$ constitute a resolution path that connects u and z. Indeed, if x is set to false, the first clause simplifies to the implication $\neg u \implies z$, and if x is set to true, the second clause simplifies to $u \implies \neg z$. The value of u may potentially force either value of z.
- Accordingly, $\mathcal{D}^{\text{rrs}}$ identifies z as truly dependent on u.

Example ($\mathcal{D}^{\text{rrs}}$ vs. $\mathcal{D}^{\text{tf}}$)

$$\forall u \exists x(\emptyset) \exists z(\{u\}) \cdot (x \vee u \vee z) \wedge (\neg x \vee \neg u \vee \neg z)$$

- The two clauses $(x \vee u \vee z)$ and $(\neg x \vee \neg u \vee \neg z)$ constitute a resolution path that connects u and z. Indeed, if x is set to false, the first clause simplifies to the implication $\neg u \implies z$, and if x is set to true, the second clause simplifies to $u \implies \neg z$. The value of u may potentially force either value of z.
- Accordingly, $\mathcal{D}^{\text{rrs}}$ identifies z as truly dependent on u.
- But x has to be set "before" z, because it does not depend on anything. Hence one of the implications is always killed. In other words, the union of the clauses, restricted to independent existential variables, is a tautology.

Example ($\mathcal{D}^{\text{rrs}}$ vs. $\mathcal{D}^{\text{tf}}$)

$$\forall u \exists x(\emptyset) \exists z(\{u\}) \cdot (x \vee u \vee z) \wedge (\neg x \vee \neg u \vee \neg z)$$

- The two clauses $(x \vee u \vee z)$ and $(\neg x \vee \neg u \vee \neg z)$ constitute a resolution path that connects u and z. Indeed, if x is set to false, the first clause simplifies to the implication $\neg u \implies z$, and if x is set to true, the second clause simplifies to $u \implies \neg z$. The value of u may potentially force either value of z.
- Accordingly, $\mathcal{D}^{\text{rrs}}$ identifies z as truly dependent on u.
- But x has to be set "before" z, because it does not depend on anything. Hence one of the implications is always killed. In other words, the union of the clauses, restricted to independent existential variables, is a tautology.
- $\mathcal{D}^{\text{tf}}$ detects the tautology and concludes that z is independent of u. Indeed $x \mapsto 0$ and $z \mapsto 1$ is a model that exhibits this.

Properties of $\mathcal{D}^{\mathrm{tf}}$

Proposition

$\mathcal{D}^{\mathrm{tf}}$ is a monotone dependency scheme, i.e. $\Psi \leq \Psi' \implies \mathcal{D}^{\mathrm{tf}}(\Psi) \leq \mathcal{D}^{\mathrm{tf}}(\Psi')$.

Properties of $\mathcal{D}^{\mathrm{tf}}$

Proposition

$\mathcal{D}^{\mathrm{tf}}$ is a monotone dependency scheme, i.e. $\Psi \leq \Psi' \implies \mathcal{D}^{\mathrm{tf}}(\Psi) \leq \mathcal{D}^{\mathrm{tf}}(\Psi')$.

Theorem

$\mathcal{D}^{\mathrm{tf}}$ is fully exhibited.

Properties of $\mathcal{D}^{\mathsf{tf}}$

Proposition

$\mathcal{D}^{\mathsf{tf}}$ is a monotone dependency scheme, i.e. $\Psi \leq \Psi' \implies \mathcal{D}^{\mathsf{tf}}(\Psi) \leq \mathcal{D}^{\mathsf{tf}}(\Psi')$.

Theorem

$\mathcal{D}^{\mathsf{tf}}$ is fully exhibited.

Proof.

By reduction to full exhibition of $\mathcal{D}^{\mathsf{rrs}}$ established for DQBF by Wimmer et al. [WSWB16]. If Ψ is true, pick a satisfying assignment α to $I_\exists(\Psi)$, and restrict with it. Because $\Psi[\alpha]$ has no independent existential variables, $\mathcal{D}^{\mathsf{tf}}$ reduces to $\mathcal{D}^{\mathsf{rrs}}$ and the theorem follows by full exhibition of $\mathcal{D}^{\mathsf{rrs}}$. □

Properties of $\mathcal{D}^{\mathrm{tf}}$

Proposition

$\mathcal{D}^{\mathrm{tf}}$ is a monotone dependency scheme, i.e. $\Psi \leq \Psi' \implies \mathcal{D}^{\mathrm{tf}}(\Psi) \leq \mathcal{D}^{\mathrm{tf}}(\Psi')$.

Theorem

$\mathcal{D}^{\mathrm{tf}}$ is fully exhibited.

Corollary

$\mathcal{D}^{\mathrm{tf}}$ can be plugged in into any proof system, in particular $\forall$Exp+Res.

Separations

Genuine DQBF Separations

- Extending the notion of genuine QBF hardness [Che17, BHP20], we define *genuine DQBF separations* as such separations, where the hardness is not witnessed by any *embedded QBF family*.

Genuine DQBF Separations

- Extending the notion of genuine QBF hardness [Che17, BHP20], we define *genuine DQBF separations* as such separations, where the hardness is not witnessed by any *embedded QBF family*.

Definition

Let P and Q be DQBF proof systems. We write $Q \not\leq_p^* P$ when there exists a DQBF family $\{\Psi_n\}_{n\in\mathbb{N}}$ such that:

(a) $\{\Psi_n\}_{n\in\mathbb{N}}$ has polynomial-size Q refutations;

(b) $\{\Psi_n\}_{n\in\mathbb{N}}$ requires superpolynomial-size P refutations;

(c) every QBF family $\{\Phi_n\}_{n\in\mathbb{N}}$ with $\Phi_n \leq \Psi_n$ has polynomial-size P refutations.

We write $P <_p^* Q$ when both $P \leq_p Q$ and $Q \not\leq_p^* P$ hold.

Main Theorem

Theorem

$\forall\text{Exp+Res} <_p^* \forall\text{Exp+Res}(\mathcal{D}^{\text{rrs}}) <_p^* \forall\text{Exp+Res}(\mathcal{D}^{\text{tf}})$.

Main Theorem

Theorem

$\forall\text{Exp+Res} <_p^* \forall\text{Exp+Res}(\mathcal{D}^{\text{rrs}}) <_p^* \forall\text{Exp+Res}(\mathcal{D}^{\text{tf}})$.

Definition (EQ_n^0 (adapted from [BBH19]))

$\text{EQ}_n^0 := \Pi_n^{\text{EQ}} \cdot \psi_n^{\text{EQ}}$, where

$$\Pi_n^{\text{EQ}} := \forall u_1 \cdots \forall u_n \exists x_1(\emptyset) \cdots \exists x_n(\emptyset)\, \exists z_1(u_1) \cdots \exists z_n(u_n)\,,$$

$$\psi_n^{\text{EQ}} := (\overline{z_1} \vee \cdots \vee \overline{z_n}) \wedge \bigwedge_{i=1}^{n} \Big((\overline{x_i} \vee \overline{u_i} \vee z_i) \wedge (x_i \vee u_i \vee z_i) \Big)\,.$$

Human readably:

- there are x_i and z_i depending on u_i such that for all values of the u_i
- if $u_i = x_i$, then z_i, but not all z_i.

First Separation

Theorem

$\{\mathsf{EQ}^0_n\}_{n\in\mathbb{N}}$ *requires exponential-size* $\forall$Exp+Res *refutations.*

First Separation

Theorem

$\{\mathsf{EQ}_n^0\}_{n\in\mathbb{N}}$ *requires exponential-size* $\forall$Exp+Res *refutations.*

Proposition ([BB19])

For all n, the dependency sets of $\mathcal{D}^{\mathsf{rrs}}(\mathsf{EQ}_n^0)$ *are empty.*

First Separation

Theorem

$\{\mathsf{EQ}^0_n\}_{n\in\mathbb{N}}$ *requires exponential-size* $\forall$Exp+Res *refutations.*

Proposition ([BB19])

For all n, the dependency sets of $\mathcal{D}^{\mathrm{rrs}}(\mathsf{EQ}^0_n)$ *are empty.*

Theorem ([BB19])

$\{\mathsf{EQ}^0_n\}_{n\in\mathbb{N}}$ *has linear-size* $\forall$Exp+Res$(\mathcal{D}^{\mathrm{rrs}})$ *refutations.*

Second Separation

Definition (EQ^1_n (adapted from [BB17]))

For each natural number n,

$$\mathsf{EQ}^1_n := \Pi^{\mathsf{EQ}}_n \exists r(\emptyset)\, \exists s(\{u_1, \ldots, u_n\}) \cdot \Big(\psi^{\mathsf{EQ}}_n \otimes (r \vee s)\Big) \wedge \Big(\psi^{\mathsf{EQ}}_n \otimes (\overline{r} \vee \overline{s})\Big) \wedge (r \vee \overline{s}) \wedge (\overline{r} \vee s)\,.$$

Second Separation

Definition (EQ^1_n (adapted from [BB17]))

For each natural number n,

$$\mathrm{EQ}^1_n := \Pi^{\mathrm{EQ}}_n \exists r(\emptyset) \exists s(\{u_1, \ldots, u_n\}) \cdot \left(\psi^{\mathrm{EQ}}_n \otimes (r \vee s)\right) \wedge \left(\psi^{\mathrm{EQ}}_n \otimes (\overline{r} \vee \overline{s})\right) \wedge (r \vee \overline{s}) \wedge (\overline{r} \vee s)\,.$$

Proposition

For each n, $\mathcal{D}^{\mathrm{rrs}}(\mathrm{EQ}^1_n) = \mathrm{EQ}^1_n$ and the dependency sets of $\mathcal{D}^{\mathrm{tf}}(\mathrm{EQ}^1_n)$ are all empty.

Second Separation

Definition (EQ^1_n (adapted from [BB17]))

For each natural number n,

$$\mathrm{EQ}^1_n := \Pi^{\mathrm{EQ}}_n \exists r(\emptyset)\, \exists s(\{u_1, \ldots, u_n\}) \cdot \left(\psi^{\mathrm{EQ}}_n \otimes (r \vee s)\right) \wedge \left(\psi^{\mathrm{EQ}}_n \otimes (\overline{r} \vee \overline{s})\right) \wedge (r \vee \overline{s}) \wedge (\overline{r} \vee s)\,.$$

Proposition

For each n, $\mathcal{D}^{\mathrm{rrs}}(\mathrm{EQ}^1_n) = \mathrm{EQ}^1_n$ and the dependency sets of $\mathcal{D}^{\mathrm{tf}}(\mathrm{EQ}^1_n)$ are all empty.

Theorem

Hence, $\{\mathrm{EQ}^1_n\}_{n\in\mathbb{N}}$ requires exponential-size $\forall\mathrm{Exp}{+}\mathrm{Res}(\mathcal{D}^{\mathrm{rrs}})$ refutations, but has linear-size $\forall\mathrm{Exp}{+}\mathrm{Res}(\mathcal{D}^{\mathrm{tf}})$ refutations.

Summary

- We proposed a clean framework for DQBF dependency schemes and their proof complexity centered around the notion of full exhibition;

Summary

- We proposed a clean framework for DQBF dependency schemes and their proof complexity centered around the notion of full exhibition;
- We defined a novel dependency scheme $\mathcal{D}^{\mathrm{tf}}$ based on the intuition about how resolution paths do and do not transfer information between variables;

Summary

- We proposed a clean framework for DQBF dependency schemes and their proof complexity centered around the notion of full exhibition;
- We defined a novel dependency scheme $\mathcal{D}^{\mathrm{tf}}$ based on the intuition about how resolution paths do and do not transfer information between variables;
- We showed that $\mathcal{D}^{\mathrm{tf}}$ is fully exhibited;

Summary

- We proposed a clean framework for DQBF dependency schemes and their proof complexity centered around the notion of full exhibition;
- We defined a novel dependency scheme $\mathcal{D}^{\mathsf{tf}}$ based on the intuition about how resolution paths do and do not transfer information between variables;
- We showed that $\mathcal{D}^{\mathsf{tf}}$ is fully exhibited;
- We showed that the use of $\mathcal{D}^{\mathsf{tf}}$ in both $\forall$Exp+Res and Q-Res results in exponentially shorter proofs compared to $\mathcal{D}^{\mathsf{rrs}}$.

Summary

- We proposed a clean framework for DQBF dependency schemes and their proof complexity centered around the notion of full exhibition;
- We defined a novel dependency scheme $\mathcal{D}^{\mathsf{tf}}$ based on the intuition about how resolution paths do and do not transfer information between variables;
- We showed that $\mathcal{D}^{\mathsf{tf}}$ is fully exhibited;
- We showed that the use of $\mathcal{D}^{\mathsf{tf}}$ in both $\forall$Exp+Res and Q-Res results in exponentially shorter proofs compared to $\mathcal{D}^{\mathsf{rrs}}$.
- A short remark on the Equality formulas: the QBF template is hard for both proof system, but our DQBF version is only hard for $\forall$Exp+Res and becomes easy in Q-Res. Why?

References I

- Joshua Blinkhorn and Olaf Beyersdorff, *Shortening QBF proofs with dependency schemes*, International Conference on Theory and Practice of Satisfiability Testing (SAT) (Serge Gaspers and Toby Walsh, eds.), Lecture Notes in Computer Science, vol. 10491, Springer, 2017, pp. 263–280.
- Olaf Beyersdorff and Joshua Blinkhorn, *Dynamic QBF dependencies in reduction and expansion*, ACM Transactions on Computational Logic **21** (2019), no. 2, 1–27.
- Olaf Beyersdorff, Joshua Blinkhorn, and Luke Hinde, *Size, cost, and capacity: A semantic technique for hard random QBFs*, Logical Methods in Computer Science **15** (2019), no. 1.
- Olaf Beyersdorff, Luke Hinde, and Ján Pich, *Reasons for hardness in QBF proof systems*, ACM Transactions on Computation Theory **12** (2020), no. 2, 10:1–10:27.
- Hubie Chen, *Proof complexity modulo the polynomial hierarchy: Understanding alternation as a source of hardness*, ACM Transactions on Computation Theory **9** (2017), no. 3, 15:1–15:20.

References II

Friedrich Slivovsky and Stefan Szeider, *Soundness of Q-resolution with dependency schemes*, Theoretical Computer Science **612** (2016), 83–101.

Ralf Wimmer, Christoph Scholl, Karina Wimmer, and Bernd Becker, *Dependency schemes for DQBF*, International Conference on Theory and Practice of Satisfiability Testing (SAT) (Nadia Creignou and Daniel Le Berre, eds.), Lecture Notes in Computer Science, vol. 9710, Springer, 2016, pp. 473–489.