Dependency Schemes
from Antiquity to Modernity

Tomáš Peitl

TU Wien, Austria

QBF Workshop
FLoC 2026, Lisbon

When does e depend on u?

$\forall u \exists e \quad (u \vee \overline{e}) \wedge (\overline{u} \vee e)$

$\forall u \exists e \quad (u \vee e) \wedge (\overline{u} \vee e)$

$\forall u \exists e \quad (u \vee \overline{e}) \wedge (u \vee e)$

When does e depend on u?

$\forall u \exists e \quad (u \vee \overline{e}) \wedge (\overline{u} \vee e)$ true, only via $e := u$ — e must **react** to u

$\forall u \exists e \quad (u \vee e) \wedge (\overline{u} \vee e)$

$\forall u \exists e \quad (u \vee \overline{e}) \wedge (u \vee e)$

When does e depend on u?

$\forall u \exists e \quad (u \vee \overline{e}) \wedge (\overline{u} \vee e)$ true, only via $e := u$ — e must **react** to u

$\forall u \exists e \quad (u \vee e) \wedge (\overline{u} \vee e)$ true, via $e := 1$ — e can **ignore** u

$\forall u \exists e \quad (u \vee \overline{e}) \wedge (u \vee e)$

When does e depend on u?

$\forall u \exists e$	$(u \vee \overline{e}) \wedge (\overline{u} \vee e)$	true, only via $e := u$ — e must **react** to u
$\forall u \exists e$	$(u \vee e) \wedge (\overline{u} \vee e)$	true, via $e := 1$ — e can **ignore** u
$\forall u \exists e$	$(u \vee \overline{e}) \wedge (u \vee e)$	false — $u := 0$ defeats both moves of e

When does e depend on u?

$\forall u \exists e$	$(u \vee \overline{e}) \wedge (\overline{u} \vee e)$	true, only via $e := u$ — e must **react** to u
$\forall u \exists e$	$(u \vee e) \wedge (\overline{u} \vee e)$	true, via $e := 1$ — e can **ignore** u
$\forall u \exists e$	$(u \vee \overline{e}) \wedge (u \vee e)$	false — $u := 0$ defeats both moves of e

the prefix only says e *may* look at u — whether it *must* is written in the matrix

$$\overbrace{Q_1 x_1 \cdots Q_n x_n}^{\text{prefix}} \; \overbrace{C_1 \wedge \cdots \wedge C_m}^{\text{matrix}} \qquad Q_i \in \{\forall, \exists\}$$

- the prefix is a **linear order**: every variable may depend on *all* outer variables
- **game semantics**: the $\forall$- and the $\exists$-player assign variables in prefix order, $\exists$ trying to satisfy the matrix — true $\iff$ the $\exists$-player has a **winning strategy**
- running example: $\forall u \exists e \exists f \exists g \quad \underbrace{(u \vee \overline{e}) \wedge (\overline{u} \vee e)}_{\Phi\text{: mentions } u} \wedge \underbrace{(f \vee g) \wedge (\overline{f} \vee \overline{g})}_{\Psi\text{: never meets } u}$

Expansion and the Variable-Disjoint Case

$$\forall u\, \Phi \equiv \Phi[u \mapsto 0] \wedge \Phi[u \mapsto 1]$$

- $\forall$ = "compressed" conjunction: so decompress = expand
- existentials inner to u may answer differently in each copy: **duplicated** — the formula (nearly) doubles per expanded universal: **exponential** blow-up
- but the two copies of a variable disjoint subformula Ψ will be **identical** up to renaming — keep just one!
- nothing connects u to Ψ, so u's value has *nowhere to travel* — Ψ is **independent** of u

two threads for the whole talk: which *connections* count as real,
and *how* independence is used — **standalone** or **integrated**

Before Dependency Schemes

Resolve and Expand (Biere, 2004)

SAT'23 Test-of-Time Award

- generalizes variable-disjointness to **connectedness**:
  - *we call a variable **locally connected** to another variable if both occur in the same clause. The relation **connected** is defined as the **transitive closure of locally connected**, **ignoring variables smaller** than the expanded variable and all other universal variables in the same scope.*
- first formulation of independence in terms of **paths**:
  - no path $u \rightsquigarrow e \Rightarrow e$ independent $\Rightarrow$ **not duplicated** when expanding u
- **integrated** use of independence
  - standalone = preprocess $\Phi \mapsto \Phi'$, then solve Φ'
  - integrated = solve Φ and keep checking what is independent
- almost every dependency scheme since is a refinement of **which paths count**

Why Ignore Outer Variables?

$$\exists x \forall u \exists e \exists f \exists g \quad (x \vee u \vee \overline{e}) \wedge (x \vee \overline{u} \vee e) \wedge (\overline{x} \vee f \vee g) \wedge (\overline{x} \vee \overline{f} \vee \overline{g})$$

- now *every* pair of clauses is connected — through x
- but u **can wait for** x: x is assigned before u moves, and once x has a value, the connection is gone:

$$x = 0: \quad (u \vee \overline{e}) \wedge (\overline{u} \vee e) \qquad\qquad x = 1: \quad (f \vee g) \wedge (\overline{f} \vee \overline{g})$$

- paths through **outer variables do not count**: they are cut before u even moves
- effectively disjointness in a mid-game state

The Early Days

The standard dependency scheme D^{std}

Samer and Szeider (2007, 2009)

1. formalizes the notion of a **dependency scheme**
2. crystallizes Biere (2004)'s connectedness into the **standard** dependency scheme
   - includes idea of Bubeck and Kleine Büning (2007): **ignore** hops via **universal** variables

Definition (D^{std})

$(x, y) \in D^{\mathrm{std}}(\Phi)$ iff

1. $x <_\Phi y$ and
2. some clause containing x is connected to some clause containing y via clauses sharing **existential** variables **inner to** x

$$\exists x \forall u \exists e \exists f \exists g \quad (x \vee u \vee \overline{e}) \wedge (x \vee \overline{u} \vee e) \wedge (\overline{x} \vee f \vee g) \wedge (\overline{x} \vee \overline{f} \vee \overline{g})$$

f, g reach u only through x (outer, ignored): only e depends on u — f, g are **independent**

Soundness via prefix shifting

Definition (Samer and Szeider (2007, 2009))

A **dependency scheme** assigns to each PCNF formula a (polytime computable) **relation** that is a subset of the prefix order and such that shifting a variable and all its dependent variables to the very end of the prefix preserves truth value.

- **cumulative** = may shift a *set* of variables with their dependencies
- soundness = a **prefix rewrite** that preserves truth: **prefix** $\rightarrow$ **prefix**

Theorem

The standard dependency scheme is a cumulative dependency scheme.

standalone soundness concept

From Variables to Literals

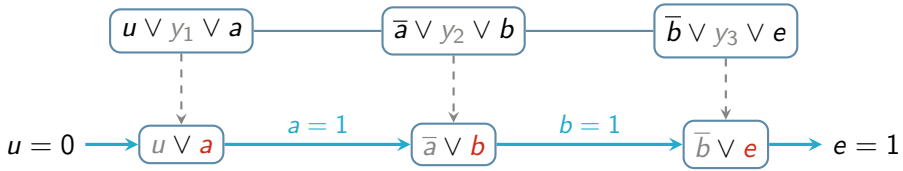
- standard dependencies are fine, but they ignore polarities completely

$$\forall u \exists e \quad (u \vee e)$$

- **triangle** dependency scheme: u should connect to both e and $\neg e$
  - also Samer and Szeider (2009)
  - connectivity is variable-based, we target a clause that contains both literals
- the beginning of a chain of generalizations, to understand which we need the concept of paths **carrying value**

Carrying value — my intuition for schemes

- a path that **carries**: assign the other outer variables ($y_1 = y_2 = y_3 = 0$), the clauses collapse, and u's value travels all the way to e



- a path that **cannot carry**: a tautology on an outer variable y — *no* assignment activates the whole chain

$y = 0$: middle clause satisfied
$y = 1$: first clause satisfied

Triangle, quadrangle, . . .

Van Gelder (2011)

- all schemes ask the same thing: is there a path $u \rightsquigarrow e$? They differ only in **which paths count** — each row tightens "channel", detecting *more* independence:

relation	a dependency needs channels reaching. . .	literals
trivial	nothing: every inner e depends	—
standard	e (some existential path $u \rightsquigarrow e$)	1
triangle (SS'09)	one of $u, \overline{u}$ and *both* $e, \overline{e}$	3
quadrangle	*both* $u, \overline{u}$ and *both* $e, \overline{e}$	4
resolution-path	all four, via real *resolutions*	4

- resolution-path: every clause transition must **change polarity**, not just share a literal

$$C \vee x \not\to D \vee x \qquad C \vee x \to D \vee \neg x$$

- Van Gelder calls these *relations* (not yet dependency *schemes*)

Chain of dependency relations

Van Gelder (2011)

- **smaller** relation = more detected independence = **stronger**
- arrow $D_1 \to D_2$ means $D_1 \subseteq D_2$: every D_1-dependency is also a D_2-dependency

- soundness travels **right**: supersets of a sound relation are sound
- further **left** means stronger — upcoming contributions are extensions leftward

Integrated Use in Solvers and Proof Systems

Proof Systems in QBF

- proof system = rules to derive clauses from existing clauses and the prefix
  - from A and B it follows (under quantification) that C
- models a learning solver: learning should be explainable by rules
- **refutation** = apply rules long enough to obtain the **empty clause**
- **proof** = derive the **empty cube** (conjunction)
- for PCNF, refutation is more natural and better studied

Q-resolution

Kleine Büning et al. (1995)

- sound and refutationally complete proof system for false PCNFs
- two rules:
  1. **resolution** on existential pivots (as in SAT)
  2. **universal reduction**: drop a universal u from a clause if no existential **right of** u (i.e. depending on u) remains
- universal reduction uses the **prefix order**

example: $\exists e \forall u \quad (u \vee \overline{e}) \wedge (\overline{u} \vee e)$ ($e \leftrightarrow u$, but e moves first: **false**)

$$(u \vee \overline{e}) \xrightarrow{\text{reduce } u} \overline{e} \qquad (\overline{u} \vee e) \xrightarrow{\text{reduce } \overline{u}} e \qquad \overline{e},\ e \xrightarrow{\text{resolve}} \square$$

- reduction is legal because e is *left* of u: **reduce first, then resolve**

Long-distance Q-resolution

Zhang and Malik (2002); Balabanov and Jiang (2012)

- allows resolvents to contain **merged** universal literals u^* (u and $\overline{u}$ together)
  - normally forbidden as tautological; permitted under side conditions on the prefix
- exponentially more powerful than plain Q-resolution on some families
- underpins QCDCL solvers (clause learning naturally produces merges)

same formula: resolving first on e creates the tautology $u \vee \overline{u}$ — kept as the merged literal u^*

$$(u \vee \overline{e}),\ (\overline{u} \vee e) \xrightarrow{\text{resolve } e} u^* \qquad\qquad u^* \xrightarrow{\text{reduce}} \square$$

- merging is legal because u is *right* of the pivot e: **merge first, then reduce**
- its dual, **long-distance Q-consensus** (for true formulas), tells a longer story with dependency schemes — we come back to it later

DepQBF

Biere and Lonsing (2010)

- first **search-based** (QCDCL) solver to integrate a dependency scheme
- efficient representation + computation of the standard scheme (Biere and Lonsing, 2009)
- independence relaxes the **decision** order and **reduction / propagation**
  - turns out decision order can be relaxed even without schemes: see afternoon invited talk

QCDCL on $\forall u \exists e \quad (u \vee \overline{e}) \wedge (u \vee e)$; scheme D such that $(u, e) \notin D$

- prefix order: **decide** u first, then propagate, conflict, learn, ...
- with D: u is reducible, both clauses are unit: **conflict before any decision**
- **but!** reduction steps like these cannot be adequately explained by a single quantifier shift
  - different steps may require different shifts, but a solver should only solve one formula

Soundness via proof systems

Slivovsky and Szeider (2014, 2016b)

- idea: replace the **prefix order** in universal reduction by a dependency scheme D
- gives the parametrised system **Q(D)-resolution**
  - reduce u from a clause once no existential that D-*depends* on u remains
  - changes (strengthens) the rule of the proof system
- D is **sound** now means: the generalized proof system stays sound
  - soundness is **integrated** — it depends on which calculus we plug D into
  - can also express dependencies of **universals on existentials**

Q(D)-resolution at work

example: $\forall u \exists e \quad (u \vee \overline{e}) \wedge (u \vee e)$ (the **false** formula from the first slide)

- plain Q-resolution cannot reduce u: e is right of u — it must resolve first
- but $\overline{u}$ occurs **nowhere**: every dependency pair needs paths from *both* u and $\overline{u}$, so $(u, e) \notin D^{\mathrm{rrs}}$

$$(u \vee \overline{e}) \xrightarrow{\text{reduce } u} \overline{e} \qquad (u \vee e) \xrightarrow{\text{reduce } u} e \qquad \overline{e},\ e \xrightarrow{\text{resolve}} \square$$

- the reductions are exactly the steps the **prefix** forbids and the **scheme** licenses

Soundness and complexity

- Q(D^{res})-resolution with plain resolution-path is **unsound** (Slivovsky and Szeider, 2014) — yet D^{res} is provably sound **standalone** (Slivovsky and Szeider, 2012, 2016a): **sound standalone, unsound integrated**
- repair: the **reflexive** scheme D^{rrs} — paths may run *through the target variable* — is sound for Q-resolution (Slivovsky and Szeider, 2016b)
- **Complexity**: resolution-path dependencies computable in **linear time** (Slivovsky and Szeider, 2012)
- long-distance Q(D^{rrs})-resolution also sound (P. et al., 2019)
- so far all for false PCNF. For true:
  - Q(D^{rrs})-consensus is sound (Slivovsky and Szeider, 2016b)
  - long-distance Q(D^{rrs})-consensus' soundness is unknown
  - this gap is neatly explained by the DQBF-centric view (coming)

Digression: QRAT

Heule et al. (2014)

- **QRAT**: a proof system to certify all of QBF **preprocessing**
- **extended universal reduction** (EUR): a resolution-path reachability check locally justifying a *single* reduction step
  - with auxiliary variables, EUR lets QRAT simulate **expansion**
- recent incarnation of this idea: **independent extension** (Chew and P., 2025)
  - extension variables with *shrunken dependency sets*
  - simulates (not only) expansion in (D)QBF
  - is simulated by $D^{\forall\text{pure}}$ (Chew and P., 2026)

Wanted: plain Q-resolution again

- Q(D)-resolution works, but it is **cumbersome**:
  - every property must be re-established per *pair* calculus $\times$ scheme: soundness, strategy extraction, certification — and again for long-distance, consensus, expansion, ...
- standalone soundness is simpler: transform **formula** once, run *unmodified* machinery
- wish: recast Q(D)-resolution as **plain Q-resolution on a simplified formula**
- but *which* formula? a scheme outputs a **relation**; a prefix is a **linear order**

$$\forall u_1 \forall u_2 \exists e \exists f \qquad e \text{ needs only } u_1, \quad f \text{ needs only } u_2$$

- needed: a language where dependencies are **first-class**

The DQBF-Centric View

Soundness via DQBF

- **Dependency QBF**: each existential carries an explicit **dependency set**

$$\forall u_1 \forall u_2 \; \exists e(\{u_1\}) \; \exists f(\{u_2\}) \;\; \varphi$$

- the prefix is no longer linear — dependencies are **first-class**, not derived from order
  - a scheme's output *is* a DQBF dependency set; D sound = the DQBF stays equivalent (Beyersdorff et al., 2019)
  - soundness decoupled from proof system
- Samer (2008) already had this DQBF-centric viewpoint:
  - *"Two variables are considered independent if their relative order in the quantifier prefix does not affect the truth value [...] easy to formalize if we consider only adjacent variables [...] However, if x and y are not adjacent, things become more complicated"*
  - his solution was DQBF-style semantic, but only considered each pair **individually**
  - the correct way is to consider all at once

Full exhibition

Beyersdorff and Blinkhorn (2016)

- D is **fully exhibited** if it maps every **true** QBF to a **true** DQBF (Slivovsky, 2015; Beyersdorff and Blinkhorn, 2016) — all independences hold **simultaneously**: **one** model exhibits them all
- the right currency: Q-resolution itself is **sound for DQBF** (Balabanov et al., 2014), so full exhibition $\Rightarrow$ Q(D)-resolution sound (Slivovsky, 2015); for **expansion** calculi it characterizes soundness *exactly* (Beyersdorff et al., 2019)
- D^{rrs} *is* fully exhibited (Beyersdorff and Blinkhorn, 2016)
  - as a consequence of soundness of Q-resolution and Q-consensus on DQBF, soundness of $Q(D^{\mathrm{rrs}})$-res/con follows
  - not for long-distance variants!

Schemes for DQBF

- first lift of a dependency scheme to DQBF: **preprocessing** with the standard scheme (Wimmer et al., 2015); industrial strength in **HQSpre** (Wimmer et al., 2017)
- which QBF schemes stay sound for DQBF? (Wimmer et al., 2016) the **reflexive** ones do; triangle and quadrangle do **not** — sound for QBF, **unsound for DQBF**
- here a scheme *shrinks* dependency sets — smaller sets $\Rightarrow$ easier expansion / solving
  - quantifier shifting $\rightarrow$ dependency relation $\rightarrow$ native DQBF shrinkage
- understanding has crystallized neatly, let us return to defining more schemes

Tautology-/implication-free schemes

Beyersdorff et al. (2020, 2024)

$y = 0$: middle clause satisfied
$y = 1$: first clause satisfied

- **tautology-free**: forbid exactly those paths, strengthening reflexive resolution-path
- **implication-free**: a finer, more permissive condition — an **infinite family** generalising D^{rrs}
- yields sound schemes that are more general than D^{rrs}
- all sound for *any* DQBF proof system — orthogonal soundness by construction
- complexity: polynomial-time as long as
  1. the blocking condition is polytime
  2. the segment has bounded length (otherwise NP-complete)
- intuition: reachability with memory

The $\forall$-pure dependency scheme $D^{\forall\text{pure}}$

Chew and P. (2026)

- the same kill-condition, now with u *itself*: a resolution path for the pair (u, e) that passes through $\overline{u}$ is **void**
- yields a sound scheme that is more general than D^{rrs}
- can do for DQRAT what D^{rrs} in EUR did for QRAT + more elegantly (see SAT talk Wed 14:00)

the path matters only when $u = 0$
— but then $\overline{u}$ satisfies the middle clause

Definition-based dependency scheme (Kattermann et al., 2026)

- a **completely different** idea — not connectivity at all
- uses variables that are **propositionally defined** in terms of a set of others
  - a defined variable carries no independent choice $\Rightarrow$ no genuine dependency
- only poly-time computable under **strong restrictions** (definition detection is hard in general)
- but: for a **PSPACE-complete** problem, needing **SAT/NP queries** to compute a scheme may be a perfectly acceptable price
- the beginning of something new?

Shortening proofs

- starting from D^{rrs}, every dependency scheme provides exponential speedup compared to the previous scheme
  - for various proof systems including Q-resolution and expansion
- first result in this line is Blinkhorn and Beyersdorff (2017)
  - new schemes come with "separation included"
- unknown whether D^{std} also provides an exponential speedup
- there is also work on complexity of dependency schemes within QCDCL-modelling proof systems (Choudhury and Mahajan, 2024, 2025)

Soundness of long-distance Q(D)-consensus

- **consensus** = the dual calculus for *true* formulas (terms/cubes instead of clauses)
- long-distance Q(D^{std})-resolution sound by P. et al. (2016), but its dual has been **open** since at least DepQBF; long-distance Q-consensus had *not* been shown sound with **any** dependency scheme
  - clause-based paths hard to connect to cubes in Q-consensus
  - long-distance prevents DQBF arguments
- resolved this year: **long-distance Q(D^{std})-consensus is sound** (Choudhury et al., 2026)
- still open for stronger schemes (e.g. D^{rrs})

Summary

Paths carrying values

- the whole story is about tightening connectivity (except definition-based)
- intuition: paths enable unit propagation
  - also resolution paths are related to proofs, hence difficulty with consensus proofs
- complexity will explode if path condition is
  - complex
  - long
- most generalizations lead to proof-complexity separation

Soundness, three ways

era	soundness of D means...	mode	$\forall$-on-$\exists$?
SS'09	shifting the prefix preserves truth	standalone	**yes**
Q(D)-res	the generalized proof system stays sound	integrated	**yes**
DQBF	the DQBF dependency sets stay equivalent	standalone	**no**

- a scheme can be sound in one mode and not the other
- each lens trades something: DQBF buys orthogonality but loses $\forall$-on-$\exists$ and every DQBF-unsound system (long-distance!)

Open Problems and Outlook

Open problems

- soundness of long-distance Q-consensus with stronger schemes
- separation between D^{std} and D^{rrs}
- generalizations of $\forall$-pure and implication-free

Outlook

- **outer variables in DQBF**: dependency sets only speak of universals — what about dependencies on outer existentials?
- **non-P computation**: natural for the definition-based schemes, potentially makes sense for others as well
- **practical effectiveness**: how much do the strong schemes actually help solvers and preprocessors?

Backup: unsoundness of Q(D^{res})-resolution (Slivovsky and Szeider, 2014)

Counterexample (formula from Samer, 2008):

$$\forall x \exists z \forall u \exists y \quad (x \vee u \vee \overline{y}) \wedge (\overline{x} \vee \overline{u} \vee \overline{y}) \wedge (z \vee u \vee y) \wedge (\overline{z} \vee u \vee \overline{y}) \wedge (\overline{z} \vee \overline{u} \vee y) \wedge (z \vee \overline{u} \vee \overline{y})$$

- **true**: $z := \overline{x}$, $y := x \oplus u$ — so y *must react* to x
- but every resolution path from x or $\overline{x}$ to y runs **through** y, so $(x, y) \notin D^{\mathrm{res}}$
- Q(D^{res})-resolution may thus reduce x:

$$(x \vee u \vee \overline{y}) \mapsto (u \vee \overline{y}) \qquad (\overline{x} \vee \overline{u} \vee \overline{y}) \mapsto (\overline{u} \vee \overline{y})$$

- resolve away y with the z-clauses, reduce u, resolve on z: the **empty clause** — a refutation of a true formula
- D^{rrs} counts the through-y paths: the dependency (x, y) is restored, and so is soundness

References I

Balabanov, V., Chiang, H.-J. K., and Jiang, J. R. (2014). Henkin quantifiers and Boolean formulae: A certification perspective of DQBF. *Theoretical Computer Science*, 523:86–100.

Balabanov, V. and Jiang, J.-H. R. (2012). Unified QBF certification and its applications. *Formal Methods in System Design*, 41(1):45–65.

Beyersdorff, O. and Blinkhorn, J. (2016). Dependency schemes in QBF calculi: Semantics and soundness. In Rueher, M., editor, *Principles and Practice of Constraint Programming*, pages 96–112, Cham. Springer International Publishing.

Beyersdorff, O., Blinkhorn, J., Chew, L., Schmidt, R., and Suda, M. (2019). Reinterpreting dependency schemes: Soundness meets incompleteness in DQBF. *Journal of Automated Reasoning*, 63(3):597–623.

Beyersdorff, O., Blinkhorn, J., and P., T. (2020). Strong (D)QBF dependency schemes via tautology-free resolution paths. In Pulina, L. and Seidl, M., editors, *Theory and Applications of Satisfiability Testing - SAT 2020 - 23rd International Conference*, volume 12178 of *Lecture Notes in Computer Science*, pages 394–411. Springer.

Beyersdorff, O., Blinkhorn, J., and P., T. (2024). Strong (D)QBF dependency schemes via implication-free resolution paths. *ACM Trans. Comput. Theory*, 16(4):22:1–22:25.

References II

Biere, A. (2004). Resolve and expand. In *Proceedings of SAT 2004 (Seventh International Conference on Theory and Applications of Satisfiability Testing, 10–13 May, 2004, Vancouver, BC, Canada)*, pages 59–70.

Biere, A. and Lonsing, F. (2009). A compact representation for syntactic dependencies in QBFs. In Kullmann, O., editor, *Theory and Applications of Satisfiability Testing - SAT 2009*, volume 5584 of *Lecture Notes in Computer Science*, pages 398–411. Springer Verlag.

Biere, A. and Lonsing, F. (2010). Integrating dependency schemes in search-based QBF solvers. In Strichman, O. and Szeider, S., editors, *Theory and Applications of Satisfiability Testing - SAT 2010*, volume 6175 of *Lecture Notes in Computer Science*, pages 158–171. Springer Verlag.

Blinkhorn, J. and Beyersdorff, O. (2017). Shortening QBF proofs with dependency schemes. In *Theory and Applications of Satisfiability Testing - SAT 2017 - 20th International Conference, Melbourne, VIC, Australia, August 28 - September 1, 2017, Proceedings*, pages 263–280.

Bubeck, U. and Kleine Büning, H. (2007). Bounded universal expansion for preprocessing QBF. In *Theory and Applications of Satisfiability Testing – SAT 2007*, volume 4501 of *Lecture Notes in Computer Science*, pages 244–257. Springer Verlag.

References III

Chew, L. and P., T. (2025). Better extension variables in DQBF via independence. In Berg, J. and Nordström, J., editors, *28th International Conference on Theory and Applications of Satisfiability Testing, SAT 2025*, volume 341 of *LIPIcs*, pages 12:1–12:24.

Chew, L. and P., T. (2026). Strong (D)QBF dependency schemes via pure paths with applications to proof checking. In *Theory and Applications of Satisfiability Testing – SAT 2026*. to appear.

Choudhury, A. and Mahajan, M. (2024). Dependency schemes in CDCL-based QBF solving: A proof-theoretic study. *Journal of Automated Reasoning*, 68(3):16.

Choudhury, A. and Mahajan, M. (2025). On the interplay of cube learning and dependency schemes in {QCDCL} proof systems. In Aiswarya, C., Mehta, R., and Roy, S., editors, *45th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2025, BITS Pilani, K K Birla Goa Campus, India, December 17-19, 2025*, volume 360 of *LIPIcs*, pages 25:1–25:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Choudhury, A., Mahajan, M., and Slivovsky, F. (2026). Long-distance Q(D^{std})-consensus is sound. In *Theory and Applications of Satisfiability Testing – SAT 2026*. to appear.

References IV

Heule, M., Seidl, M., and Biere, A. (2014). A unified proof system for QBF preprocessing. In Demri, S., Kapur, D., and Weidenbach, C., editors, *Automated Reasoning - 7th International Joint Conference, IJCAR 2014*, volume 8562 of *Lecture Notes in Computer Science*, pages 91–106. Springer Verlag.

Kattermann, D., Hofstadler, C., and Seidl, M. (2026). Definition-based dependency schemes. In *Theory and Applications of Satisfiability Testing – SAT 2026*. to appear.

Kleine Büning, H., Karpinski, M., and Flögel, A. (1995). Resolution for quantified Boolean formulas. *Information and Computation*, 117(1):12–18.

P., T., Slivovsky, F., and Szeider, S. (2016). Long distance Q-resolution with dependency schemes. In Creignou, N. and Berre, D. L., editors, *Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5-8, 2016, Proceedings*, volume 9710 of *Lecture Notes in Computer Science*, pages 500–518. Springer Verlag.

P., T., Slivovsky, F., and Szeider, S. (2019). Long-distance Q-resolution with dependency schemes. *Journal of Automated Reasoning*, 63(1):127–155.

References V

Samer, M. (2008). Variable dependencies of quantified CSPs. In Cervesato, I., Veith, H., and Voronkov, A., editors, *Logic for Programming, Artificial Intelligence, and Reasoning, 15th International Conference, LPAR 2008, Doha, Qatar, November 22-27, 2008. Proceedings*, volume 5330 of *Lecture Notes in Computer Science*, pages 512–527. Springer Verlag.

Samer, M. and Szeider, S. (2007). Backdoor sets of quantified boolean formulas. In Marques-Silva, J. and Sakallah, K. A., editors, *Proceedings of SAT 2007, Tenth International Conference on Theory and Applications of Satisfiability Testing, May 28-31, 2007, Lisbon, Portugal,*, volume 4501 of *Lecture Notes in Computer Science*, pages 230–243.

Samer, M. and Szeider, S. (2009). Backdoor sets of quantified Boolean formulas. *Journal of Automated Reasoning*, 42(1):77–97.

Slivovsky, F. (2015). *Structure in #SAT and QBF*. PhD thesis, TU Wien.

Slivovsky, F. and Szeider, S. (2012). Computing resolution-path dependencies in linear time. In Cimatti, A. and Sebastiani, R., editors, *Theory and Applications of Satisfiability Testing - SAT 2012*, volume 7317 of *Lecture Notes in Computer Science*, pages 58–71. Springer Verlag.

References VI

Slivovsky, F. and Szeider, S. (2014). Variable dependencies and Q-resolution. In Sinz, C. and Egly, U., editors, *Theory and Applications of Satisfiability Testing - SAT 2014*, volume 8561 of *Lecture Notes in Computer Science*, pages 269–284. Springer Verlag.

Slivovsky, F. and Szeider, S. (2016a). Quantifier reordering for QBF. *Journal of Automated Reasoning*, 56(4):459–477.

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

Van Gelder, A. (2011). Variable independence and resolution paths for quantified Boolean formulas. In Lee, J., editor, *Principles and Practice of Constraint Programming - CP 2011*, volume 6876 of *Lecture Notes in Computer Science*, pages 789–803. Springer Verlag.

Wimmer, R., Gitina, K., Nist, J., Scholl, C., and Becker, B. (2015). Preprocessing for DQBF. In *Theory and Applications of Satisfiability Testing – SAT 2015*, volume 9340 of *Lecture Notes in Computer Science*, pages 173–190. Springer Verlag.

References VII

Wimmer, R., Reimer, S., Marin, P., and Becker, B. (2017). Hqspre - an effective preprocessor for QBF and DQBF. In Legay, A. and Margaria, T., editors, *Tools and Algorithms for the Construction and Analysis of Systems - 23rd International Conference, TACAS 2017*, volume 10205 of *Lecture Notes in Computer Science*, pages 373–390.

Wimmer, R., Scholl, C., Wimmer, K., and Becker, B. (2016). Dependency schemes for DQBF. In *Theory and Applications of Satisfiability Testing – SAT 2016*, volume 9710 of *Lecture Notes in Computer Science*, pages 473–489. Springer Verlag.

Zhang, L. and Malik, S. (2002). Conflict driven learning in a quantified Boolean satisfiability solver. In *Proceedings of the 2002 IEEE/ACM International Conference on Computer-Aided Design (ICCAD 2002)*, pages 442–449.