Computing Witnesses Using the SCAN Algorithm (Extended Abstract)

Fabian Achammer[1,*], Stefan Hetzl[1] and Renate A. Schmidt[2]

[1]TU Wien, Austria
[2]The University of Manchester, United Kingdom

Abstract

We show show how to extend the SCAN algorithm for second-order quantifier elimination to solve a more general problem: Finding a witness of the second-order quantifiers that results in a logically equivalent first-order formula.

Keywords

Second-order quantifier elimination, SCAN algorithm, Saturation theorem proving

1. Introduction

Second-order quantifier elimination (SOQE) [1, 2] is the problem of, given a formula with second-order quantifiers $\exists\overline{X}\,\varphi$ where φ is first-order, finding a first-order formula ψ that is logically equivalent to $\exists\overline{X}\,\varphi$, i.e.,

$$\exists\overline{X}\,\varphi \equiv \psi.$$

An algorithm for eliminating existential quantifiers can be used to eliminate universal quantifiers by writing $\forall\overline{X}$ as $\neg\exists\overline{X}\neg$.

A prominent approach to solving SOQE is the SCAN algorithm introduced in [1]. SCAN is based on the idea of computing all logical consequences of $\exists\overline{X}\,\varphi$ and omitting those formulas that contain predicate variables from $\overline{X}$. It transforms the first-order part φ into clausal normal form and computes the closure under a constraint resolution calculus. During this process, clauses with $\overline{X}$-literals for which sufficiently many inferences have been performed are deleted [3, 4]. If the algorithm terminates, it returns a set of first-order consequences which is logically equivalent to $\exists\overline{X}\,\varphi$. Reverse Skolemization might be applied, if during the clause form transformation Skolemization was performed.

In this abstract we are interested in the more general problem of *witnessed second-order quantifier elimination (WSOQE)*: Given a formula $\exists\overline{X}\,\varphi$ with first-order φ, find first-order predicates (formulas) $\overline{\alpha}$ satisfying the *WSOQE-condition*

$$\exists\overline{X}\,\varphi \equiv \varphi[\overline{X} \leftarrow \overline{\alpha}]. \tag{$*$}$$

That is, the goal is to find a first-order instantiation $\overline{\alpha}$ (in the same language as φ) of variables $\overline{X}$ such that an equivalent first-order formula is obtained. We call the substitution $[\overline{X} \leftarrow \overline{\alpha}]$ a *WSOQE-witness for* $\exists\overline{X}\,\varphi$ (for the purposes of this abstract we simply call $[\overline{X} \leftarrow \overline{\alpha}]$ a *witness*). Consider the second-order formula $\Phi = \exists X\,(X(a) \wedge \forall u\,(X(u) \rightarrow B(u)))$. One can verify that the substitutions that $\sigma_1 = [X \leftarrow \lambda u.u \simeq a]$ and $\sigma_2 = [X \leftarrow \lambda u.B(u)]$ are both witnesses for Φ.

CI-BD-SOQE 2026: Workshop on Craig Interpolation, Beth Definability, and Second-Order Quantifier Elimination at FLoC 2026, July 24–25, 2026, Lisbon, Portugal

*Corresponding author.

fabian.achammer@tuwien.ac.at (F. Achammer); stefan.hetzl@tuwien.ac.at (S. Hetzl); renate.schmidt@manchester.ac.uk (R. A. Schmidt)

https://fabian.achammer.dev/ (F. Achammer); https://dmg.tuwien.ac.at/hetzl/ (S. Hetzl); https://www.cs.man.ac.uk/~schmidt/ (R. A. Schmidt)

0009-0002-3799-6393 (F. Achammer); 0000-0002-6461-5982 (S. Hetzl); 0000-0002-6673-3333 (R. A. Schmidt)

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

WSOQE bridges the gap between SOQE and another important problem, namely, *solving formula equations (FEQ)*: Given an input formula $\exists \overline{X}\, \varphi$, where φ is first-order, find a tuple $\overline{\alpha}$ of first-order predicates such that $\models \varphi[\overline{X} \leftarrow \overline{\alpha}]$.

FEQ is the central problem underlying several areas of computational logic, even though this is often not made explicit. For example, solving constrained Horn clauses [5], a popular formalism in verification, is a restriction of FEQ, with the existentially quantified variables representing unknown loop invariants in applications of software verification. More generally, the problem of inductive theorem proving, e.g., in Peano arithmetic, can be formulated as a restriction of FEQ, where the predicate variables stand for the unknown induction formulas.

An algorithm for WSOQE can be used to solve both SOQE and FEQ: A WSOQE-witness $\overline{\alpha}$ satisfies $(*)$ and therefore $\varphi[\overline{X} \leftarrow \overline{\alpha}]$ solves SOQE for $\exists \overline{X}\, \varphi$. Apart from solving SOQE, if one has a WSOQE-witness $\overline{\alpha}$ for $\exists \overline{X}\, \varphi$ one can determine whether it is an FEQ-solution for $\exists \overline{X}\, \varphi$ by checking whether $\varphi[\overline{X} \leftarrow \overline{\alpha}]$ is valid which can be done with a first-order theorem prover.

In this abstract we describe a novel algorithm, called *WSCAN*, to solve the WSOQE problem for clause sets. It extends SCAN with a post-processing step which extracts a witness from the derivation of a terminating SCAN run. We implemented a prototype in GAPT 2.19.0 [6].

We extend [7] by developing a method that produces first-order witnesses for a larger class of derivations. This class is characterized by properties of certain graphs that capture the redundancy information at each deletion step of the derivation. A detailed treatment of the results with full proofs can be found in [8].

2. SCAN Algorithm

To derive new clauses in the saturation process, the SCAN algorithm uses the following inference rules.

Constraint resolution

$$\frac{C \vee L(\bar{t}) \qquad C' \vee L(\overline{t'})^{\perp}}{\bar{t} \not\approx \overline{t'} \vee C \vee C'}\ \mathrm{Res}$$

where L is a literal and $L^{\perp}$ its dual

Constraint factoring

$$\frac{C \vee L(\bar{t}) \vee L(\overline{t'})}{\bar{t} \not\approx \overline{t'} \vee C \vee L(\bar{t})}\ \mathrm{Fac}$$

where L is a literal

Constraint elimination

$$\frac{\bar{t} \not\approx \overline{t'} \vee C}{C\sigma}\ \mathrm{ConstrElim}$$

where σ is a most general unifier of $\bar{t}$ and $\overline{t'}$

We denote by $\mathcal{C}$ the calculus operating on clause sets with the following derivation steps:

Inference

$$\frac{N}{N \cup \{C\}}\ I$$

if $\dfrac{C_1 \cdots C_n}{C}\ I$ is an inference rule from above and $C_1, \ldots, C_n \in N$.

Redundancy deletion

$$\frac{N \cup \{C\}}{N}\ \mathrm{RedDel}$$

if C is redundant in N (e.g., C is tautological or subsumed)

Variable elimination

$$\frac{N \cup \{v \not\approx t \vee C\}}{N \cup \{C[v \leftarrow t]\}}\ \mathrm{VarElim}$$

where v is a variable and is not a proper subterm of t

Extended purity deletion

$$\frac{N}{N \setminus \{C \in N \mid C \text{ contains } X\}}\ \mathrm{ExtPurDel}^{p}_{X}$$

for $p \in \{+, -\}$, where X is a predicate variable if every clause in N containing X contains X with polarity p

An additional derivation step is deleting a clause when sufficiently many inferences have been performed with the clause *on a specific literal* and the rest of the clause set. This leads to the notion of a *pointed clause*, i.e., a clause P with a designated literal $L \in P$. Resolution with P then means resolving P with

a clause on the designated literal of P. We denote the resolution closure of a clause set N with respect to a pointed clause P by $\mathrm{Res}_P^{<\omega}(N)$. Having performed sufficiently many inferences is captured in the notion of P *being purified in* N which can be any decidable criterion ensuring that N logically implies $\mathrm{Res}_P^{<\omega}(N)$.

Purified clause deletion

$$\frac{N \uplus \{P\}}{N}\,\mathrm{PurDel}_P \quad \text{if } P \text{ is purified in } N.$$

A *derivation* in $\mathcal{C}$ is a sequence of $\mathcal{C}$-derivation steps $D = (S_1, \ldots, S_m)$ that starts from a given clause set N. It is *$\overline{X}$-eliminating* if, after applying all derivation steps $S_1, \ldots, S_m$ to N, we arrive at a clause set that contains no $\overline{X}$-variables.

3. Witness computation

Given a $\mathcal{C}$-derivation, the idea of WSCAN is to construct a witness back-to-front: Let $D = (S_1, \ldots, S_m)$ be an $\overline{X}$-eliminating $\mathcal{C}$-derivation starting from N. Denote by $N_1, \ldots, N_m$ the intermediate clause sets after applying $S_1, \ldots, S_m$ respectively and set $N_0 = N$. Then the substitution σ_m whose domain is in $\overline{X}$ is a witness for $\exists\overline{X}\, N_m$. For $1 < i \leq m$ and a witness σ_i for $\exists\overline{X}\, N_i$ we now construct a witness σ_{i-1} for $\exists\overline{X}\, N_{i-1}$ by prepending a substitution τ_{S_i} that depends on the derivation step S_i, i.e., $\sigma_{i-1} = \tau_{S_i}\sigma_i$. Thus $\sigma_0 = \tau_{S_1} \ldots \tau_{S_m}\sigma_m$ is a witness for $\exists\overline{X}\, N$. The following figure illustrates the algorithm.

$$\begin{array}{rccccccc} N = N_0 & \xrightarrow{S_1} & N_1 & \xrightarrow{S_2} & \ldots & N_{m-1} & \xrightarrow{S_m} & N_m \\ \sigma_0 & \xleftarrow{\tau_{S_1}} & \sigma_1 & \xleftarrow{\tau_{S_2}} & \ldots & \sigma_{m-1} & \xleftarrow{\tau_{S_m}} & \sigma_m \end{array}$$

For this to work the substitutions τ_{S_i} need to preserve the witness property ($*$), i.e., if σ is a witness for $\exists\overline{X}\, N_i$, then $\tau_{S_i}\sigma$ needs to be a witness for $\exists\overline{X}\, N_{i-1}$. We call such substitutions *witness-preserving across* S_i. For derivation steps besides purified clause deletion one can show that the following substitutions are witness-preserving across their respective derivation steps.

$$\begin{aligned} \tau_S &:= \mathrm{id} \quad \text{for } S \in \{\mathrm{Res}, \mathrm{ConstrElim}, \mathrm{Fac}, \mathrm{RedDel}, \mathrm{VarElim}\}, \\ \tau_{\mathrm{ExtPurDel}_X^-} &:= [X \leftarrow \lambda\overline{u}.\bot], \quad \tau_{\mathrm{ExtPurDel}_X^+} := [X \leftarrow \lambda\overline{u}.\top]. \end{aligned}$$

For purified clause deletion steps the definition of such a substitution τ_S is more involved and is the key to constructing witnesses with this back-to-front algorithm. As an important building block we define the *local resolution closure of* $P = \underline{L(\overline{t})} \vee C$ by the (potentially infinite) predicate expression

$$\ell\mathrm{Res}_P := \lambda\overline{c}. \bigwedge_{R(\overline{c},\overline{v}) \in \mathrm{Res}_P^{<\omega}(L(\overline{c})^\bot)} \forall\overline{v}\, R(\overline{c}, \overline{v}).$$

Furthermore we set

$$\tau_{\mathrm{PurDel}_P} := \begin{cases} [X \leftarrow \ell\mathrm{Res}_P] & \text{if } L \text{ is negative,} \\ [X \leftarrow \neg\ell\mathrm{Res}_P] & \text{if } L \text{ is positive.} \end{cases}$$

Showing that τ_{PurDel_P} is witness-preserving across purified clause deletion steps is not straightforward. For the proof it is essential to use the side condition that P is purified in N, i.e., that $\mathrm{Res}_P^{<\omega}(N)$ is logically implied by N. Having shown that all τ_{S_i} are witness-preserving we get the following theorem.

Theorem 1. *If $D = (S_1, \ldots, S_m)$ is an $\overline{X}$-eliminating derivation from N, then $\tau_{S_1} \cdots \tau_{S_m}$ is a witness for $\exists\overline{X}\, N$.*

Note that the range of $\tau_{S_1} \cdots \tau_{S_m}$ might still contain the predicates to be eliminated since $L(\overline{c})^\bot \in \mathrm{Res}_P^{<\omega}$ and therefore τ_{PurDel} can contain predicates to be eliminated. In the context of Theorem 1 these are free variables so any instantiation of them yields a witness.

We can also express $\ell\mathrm{Res}_P$ as a greatest fixpoint which leads to the the following result.

Theorem 2. *If there exists a $\overline{X}$-eliminating derivation from N, then $\exists\overline{X}\, N$ has a fixpoint witness, i.e., a witness whose range consists of formulas in first-order logic with least and greatest fixpoints.*

To ensure first-order witnesses we need to adjust the witness-preserving substitution across purified clause deletion steps, since τ_{PurDel_P} is not first-order in general. Our method works for *acyclic* purified clause deletion steps which are those purified clause deletion steps $N \cup \{P\}\,/N$ where the following graph is acyclic.

Definition 3. *Let N be a clause set, P be a pointed clause with designated literal L. Furthermore, let s be a function that assigns to every pointed clause $P' \in N$ which is resolvable with P a clause $S \in N$ such that S subsumes the resolvent between P and P'. Define the* purification subsumption graph *$G(N, P, s) = (V, E)$ as the edge-labelled graph with $V = N$ and*

$$E = \left\{ P' \overset{L'}{\to} s(P') \mid P' \in N,\, P' \text{ resolvable with } P \text{ where } L' \text{ is the designated literal of } P' \right\}.$$

The substitution $\tau^k_{\mathrm{PurDel}_P}$ introduced below is witness-preserving across acyclic purified clause deletion if there exists a function s as above such that $G(N, P, s)$ has longest path length at most k.

Definition 4. *Let $P = \underline{L(\overline{t}(\overline{v}))} \vee C(\overline{v}) \vee \bigvee_{i=1}^{p} L(\overline{t_i}(\overline{v}))^\bot$ be a pointed clause where L is an X-literal and C does not contain $\overline{L^\bot}$-literals. Note that p is the number of literals in P which are dual to the designated literal $L(\overline{t}(\overline{v}))$. Let X have arity k. We inductively define the predicate expression β^k_P for $k \in \mathbb{N}$ by*

$$\beta^0_P := \lambda\overline{u}.\bot \quad \textit{and} \quad \beta^{k+1}_P := \lambda\overline{u}.L(\overline{u})^\bot \wedge \forall\overline{v}\,(\overline{u} \not\approx \overline{t}(\overline{v}) \vee C(\overline{v}) \vee \bigvee_{i=1}^{p} \beta^k_P(\overline{t_i}(\overline{v}), \overline{z_i}))$$

To get first-order witnesses we then set

$$\tau^k_{\mathrm{PurDel}_P} := \begin{cases} [X \leftarrow \beta^k_P] & \text{if the designated literal of } P \text{ is negative,} \\ [X \leftarrow \neg\beta^k_P] & \text{if the designated literal of } P \text{ is positive.} \end{cases}$$

Theorem 5. *If there exists an $\overline{X}$-eliminating $\mathcal{C}$-derivation D from N such that every purified clause deletion step in D is acyclic, then N has a first-order witness, i.e., a witness whose range consists of first-order formulas.*

We implemented a prototype of WSCAN in GAPT 2.19.0 and tested it on 44 examples which were created by us or selected from the literature. For 40 of the 44 examples our implementation of SCAN found a derivation taking an average of $\sim$ 375 milliseconds. For all those 40 derivations a first-order witness could be computed taking an average of $\sim$ 0.62 milliseconds. This shows that the witness computation takes significantly less time compared to the construction of the SCAN derivation itself.

4. Conclusion

We have introduced WSCAN, an extension of the SCAN algorithm on clause sets, to compute witnesses for existential second-order quantifiers, thereby solving the more general WSOQE-problem. Given an $\overline{X}$-eliminating derivation D from a clause set N we can produce a witness for $\exists\overline{X}\, N$. To the best of our knowledge there are currently no other WSOQE-algorithms applicable to arbitrary clause sets.

This work paves the way to new applications of the SCAN algorithm in a variety of different areas, for example, modal correspondence theory, knowledge representation or verification.

On a more abstract level, we see this work as a contribution to bridging the gap between second-order quantifier elimination (SOQE) and solving formula equations (FEQ) which will be of interest to the respective communities investigating these problems. SOQE has been studied mostly in the context of modal logic, description logics, knowledge representation and answer set programming. On the other hand, FEQ is studied (usually in different formalisms and under different names) in the verification and automated (inductive) theorem proving communities. We believe that the complex of problems consisting of SOQE, WSOQE, and FEQ provides a suitable *common logical foundation* for work that is done on all of these topics, as has also been suggested in [9]. This perspective not only allows to relate these, at first sight different, problems to one another on a logical level, but also stimulates a cross-fertilization in terms of practical solution ideas, techniques, and algorithms.

Acknowledgments

This research was funded in part by the Austrian Science Fund (FWF) grant number 10.55776/P35787.

The first author is grateful for the support in facilitating two research visits to the University of Manchester, which greatly benefited this work.

We thank Luke Sanderson who dockerfied the original version of SCAN with help from the University of Manchester Open Source Software Club.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] D. Gabbay, H. J. Ohlbach, Quantifier elimination in second order predicate logic, South African Computer Journal 7 (1992) 35–43.

[2] D. M. Gabbay, R. A. Schmidt, A. Szałas, Second-Order Quantifier Elimination, College Publications, London, 2008.

[3] T. Engel, Quantifier Elimination in Second-Order Predicate Logic, Diplomarbeit, Fachbereich Informatik, Universität des Saarlandes, Saarbrücken, Germany, 1996.

[4] H. J. Ohlbach, SCAN: Elimination of predicate quantifiers, in: M. A. McRobbie, J. K. Slaney (Eds.), Automated Deduction: CADE-13, volume 1104 of *Lecture Notes in Artificial Intelligence*, Springer, Berlin, Heidelberg, 1996, pp. 161–165. doi:10.1007/3-540-61511-3_77.

[5] N. Bjørner, A. Gurfinkel, K. L. McMillan, A. Rybalchenko, Horn clause solvers for program verification, in: L. D. Beklemishev, A. Blass, N. Dershowitz, B. Finkbeiner, W. Schulte (Eds.), Fields of Logic and Computation II - Essays Dedicated to Yuri Gurevich on the Occasion of His 75th Birthday, volume 9300 of *Lecture Notes in Computer Science*, Springer, Cham, 2015, pp. 24–51. doi:10.1007/978-3-319-23534-9_2.

[6] G. Ebner, S. Hetzl, G. Reis, M. Riener, S. Wolfsteiner, S. Zivota, System description: Gapt 2.0, in: N. Olivetti, A. Tiwari (Eds.), Automated Reasoning, Springer International Publishing, Cham, 2016, pp. 293–301. doi:10.1007/978-3-319-40229-1_20.

[7] F. Achammer, S. Hetzl, R. A. Schmidt, Computing witnesses using the SCAN algorithm, in: Automated Deduction – CADE 30: 30th International Conference on Automated Deduction, Stuttgart, Germany, July 28-31, 2025, Proceedings, Springer-Verlag, Berlin, Heidelberg, 2025, p. 511–531. doi:10.1007/978-3-031-99984-0_27.

[8] F. Achammer, S. Hetzl, R. A. Schmidt, Computing witnesses using the SCAN algorithm, 2026. arXiv:2604.27939.

[9] C. Wernhard, The Boolean solution problem from the perspective of predicate logic, in: C. Dixon, M. Finger (Eds.), 11th International Symposium on Frontiers of Combining Systems (FroCoS), volume 10483 of *Lecture Notes in Computer Science*, Springer, Cham, 2017, pp. 333–350. doi:10.1007/978-3-319-66167-4_19.