Computing Witnesses Using the SCAN Algorithm

Fabian Achammer[1], Stefan Hetzl[1], Renate Schmidt[2]

[1]Institute of Discrete Mathematics and Geometry
TU Wien
and
[2]Department of Computer Science
University of Manchester

CADE-30
DHBW Stuttgart
July 30th, 2025

Introduction

Formula Equations (FEQ)

Given $\exists \overline{X}\, \varphi$, where φ is first-order, find first-order predicates $\overline{\alpha}$ such that

$$\models \varphi[\overline{X} \leftarrow \overline{\alpha}].$$

We call such $\overline{\alpha}$ *FEQ-witnesses*.

- Similarity to solving equations
  - Finding first-order X such that $\beta(X) \equiv \gamma(X)$ is equivalent to finding fist-order X such that $\models \beta(X) \leftrightarrow \gamma(X)$

Example

$\exists X\, X(a)$ has witness $\lambda u.u \simeq a$

- Generalizes problems of software verification, inductive theorem proving, Boolean unification and others
- Undecidable (contains first-order validity problem), but recursively enumerable

Introduction

Second-order quantifier elimination (SOQE)

Given $\exists\overline{X}\,\varphi$, where φ is first-order, find a first-order formula ψ such that

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

Example

$\exists X\,(X(a) \land \forall u\,(X(u) \rightarrow B(u))) \equiv B(a)$

- Applications in modal correspondence theory, forgetting in ontologies and more
- Not recursively enumerable (not even arithmetical)
- Prominent algorithms are the saturation-based approach SCAN[1] and the Ackermann[2]-based approach DLS[3]

---

[1] GO92.
[2] Ack35.
[3] DLS97.

Introduction

Bridging the gap: Witnessed Second-order quantifier elimination (WSOQE)

Given $\exists\overline{X}\,\varphi$, where φ is first-order, find first-order predicates $\overline{\alpha}$ s.t.

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

We call such $\overline{\alpha}$ *WSOQE-witnesses*, or simply *witnesses*.

Example

$\exists X\,(X(a) \wedge \forall u\,(X(u) \rightarrow B(u))) \equiv B(a)$
Some witnesses are $\lambda u.B(u)$ and $\lambda u.u \simeq a$

- witnesses yield solutions to SOQE
- witnesses reduce corresponding FEQ-problem to first-order validity checking

Contribution of this talk:

- If φ is a clause set and SCAN terminates on $\exists\overline{X}\,\varphi$, we can construct a (potentially infinite) WSOQE-witness.

Outline

Introduction

SCAN Algorithm

Computing Witnesses

Discussion

SCAN Algorithm

For this talk we assume that we operate on clause sets N and the only second-order quantifier is $\exists X$

- Apply $\exists X$ -equivalence-preserving inference and deletion steps to N...
  - i.e., if N/N' is a derivation step, then $\exists X\, N \equiv \exists X\, N'$
- ...until the clause set does not contain X anymore. Then we found a first-order formula equivalent to $\exists X\, N$
- We capture the sequence of derivation steps in a derivation D
- If SCAN terminates we use D to compute a witness in a post-processing step

SCAN Derivation Steps

Inference steps

Constraint resolution:

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

where L is an X-literal ($L^{\perp}$ denotes the *dual literal*).
Constraint factoring:

$$\frac{L(\overline{t}) \vee L(\overline{s}) \vee C}{\overline{t} \not\approx \overline{s} \vee L(\overline{t}) \vee C} \text{ Fac}$$

Constraint elimination:

$$\frac{\overline{t} \not\approx \overline{s} \vee C}{C\sigma} \text{ ConstrElim}$$

where σ is a most general unifier of $\overline{t}$ and $\overline{s}$.

- Separate constraint resolution and constraint elimination so we can derive, e.g., $a \not\approx c$ from $X(a)$ and $\neg X(c)$.

SCAN Derivation Steps

Extended purity deletion

Negative (positive) extended purity deletion:

$$\frac{N}{N \setminus \{C \in N \mid C \text{ contains } X\}} \text{ExtPurDel}_X^{-(+)}$$

if for every clause $C \in N$ that contains X, we have that X occurs negatively (positively) in C.

Example

$$\frac{\{B(a,v),\ B(u,v) \vee \neg X(u) \vee X(v),\ \neg X(c)\}}{\{B(a,v)\}} \text{ExtPurDel}_X^{-}$$

Note that $\lambda u.\bot$ is a witness for the premise N

SCAN Derivation Steps

Redundancy elimination

- Tautology deletion
- Subsumption deletion
- Potentially other equivalence-preserving simplification steps

SCAN Derivation Steps

Purified clause deletion

- Pointed clause $P = \underline{L(\bar{t})} \vee C$: Underlining designates a literal in P with respect to which we perform resolution
- P is *purified* in a clause set N, if all resolvents between P and N are redundant in N
- Purified clause deletion:

$$\frac{N \cup \{P\}}{N}\ \mathrm{PurDel}_P$$

if P is purified in N and N is closed under constraint factoring and constraint elimination

SCAN Derivation Steps

Example

(1) $B(a, v)$

(2) $X(a)$

(3) $B(u, v) \vee \neg X(u) \vee X(v)$

(4) $\neg X(c)$

(5) $B(a, v) \vee X(v)$ (resolve 2 with 3, subsumed by 1)

(6) $a \not\approx c$ (resolve 2 with 4)

k	D_k	N_k
0		1, 2, 3, 4
1	$\text{Res}_{2,4}$	1, 2, 3, 4, 6
2	PurDel_2	1, 3, 4, 6
3	ExtPurDel_X^-	1, 6

Computing Witnesses

Approach

Let $D = (D_k)_{1 \leq k \leq m}$ be an X-eliminating derivation from N.

$$N = N_0 \xrightarrow{D_1} N_1 \quad \xrightarrow{D_2} \quad \ldots \quad \xrightarrow{D_{m-1}} N_{m-1} \quad \xrightarrow{D_m} N_m$$

$$\alpha_0 \xleftarrow{T_{D_1}} \alpha_1 \quad \xleftarrow{T_{D_2}} \quad \ldots \quad \xleftarrow{T_{D_{m-1}}} \alpha_{m-1} \quad \xleftarrow{T_{D_m}} \alpha_m = \lambda\overline{u}.W(\overline{u})$$

Computing Witnesses

Extending Witnesses across derivation steps

Lemma (Witness Preservation Lemma)

If S is a derivation step from N to N' and $\exists X\, N' \equiv N'[X \leftarrow \alpha]$, then $\exists X\, N \equiv N[X \leftarrow T_S(\alpha)]$.

We define $T_S(\alpha)$ by

$$
\begin{aligned}
T_{\mathsf{Res}}(\alpha) &= \alpha \\
T_{\mathsf{Fac}}(\alpha) &= \alpha \\
T_{\mathsf{ConstrElim}}(\alpha) &= \alpha \\
T_{\mathsf{TautDel}}(\alpha) &= \alpha \\
T_{\mathsf{SubsDel}}(\alpha) &= \alpha \\
T_{\mathsf{ExtPurDel}_X^+}(\alpha) &= \lambda\overline{u}.\top \\
T_{\mathsf{ExtPurDel}_X^-}(\alpha) &= \lambda\overline{u}.\bot \\
T_{\mathsf{PurDel}_P}(\alpha) &= \mathsf{pResU}_P[X \leftarrow \alpha]
\end{aligned}
$$

Computing Witnesses

P-resolution closure with a unit

- Recall purified clause deletion:

$$\frac{N \cup \{P\}}{N}\ \mathrm{PurDel}_P$$

if P is purified in N and closed under constraint factoring and constraint elimination.

- For $P = \underline{L(\overline{t})} \vee C$ define *the P-resolution closure with a unit* $\mathrm{ResU}_P(\overline{c})$ to be the closure of $\{L(\overline{c})^{\perp}\}$ under (constraint) resolution on P

Computing Witnesses

P-resolution closure with a unit

Example

If $P = \underline{X(a)}$, then $\mathrm{ResU}_P(c) = \{\neg X(c), a \not\approx c\}$

Example

If $P = B(u,v) \vee \underline{\neg X(u)} \vee X(v)$, then

$$\begin{aligned}\mathrm{ResU}_P(c) = \{&X(c),\\ &B(c,v) \vee X(v),\\ &B(c,v) \vee B(v,v') \vee X(v'),\\ &B(c,v) \vee B(v,v') \vee B(v',v'') \vee X(v''),\\ &\dots\}\end{aligned}$$

Computing Witnesses

Extending Witnesses across purified clause deletion

Define pResU_P by

$$\mathrm{pResU}_P = \begin{cases} \lambda\overline{u}.\bigwedge_{R(\overline{c},\overline{v})\in \mathrm{ResU}_P(\overline{c})} \forall\overline{v}\, R(\overline{u},\overline{v}) & \text{if } P = \underline{\neg X(\overline{t})} \vee C \\ \lambda\overline{u}.\bigvee_{R(\overline{c},\overline{v})\in \mathrm{ResU}_P(\overline{c})} \exists\overline{v}\,\neg R(\overline{u},\overline{v}) & \text{if } P = \underline{X(\overline{t})} \vee C \end{cases}$$

- pResU_P is potentially infinite!

Computing Witnesses

Example

(1) $B(a, v)$

(2) $X(a)$

(3) $B(u, v) \vee \neg X(u) \vee X(v)$

(4) $\neg X(c)$

(5) $B(a, v) \vee X(v)$ (resolve 2 with 3, subsumed by 1)

(6) $a \not\simeq c$ (resolve 2 with 4)

k	D_k	N_k	α_k
0		$1, 2, 3, 4$	$\lambda u.u \simeq a$
1	$\mathrm{Res}_{2,4}$	$1, 2, 3, 4, 6$	$\mathrm{pResU}_2[X \leftarrow \lambda u.\bot] \equiv \lambda u.u \simeq a$
2	PurDel_2	$1, 3, 4, 6$	$\lambda u.\bot$
3	$\mathrm{ExtPurDel}_X^-$	$1, 6$	$\lambda u.W(u)$

Computing Witnesses

Implementation

- Prototype implementation in GAPT[4]
- Tested on 26 examples created by us or picked from the literature
- Our implementation finds a witness for 21 of them
- For these the running times were between 0.03ms and 150.60ms with an average of 14.96ms.

[4] https://www.logic.at/gapt/

Further results

- Witnesses are finite if no redundancy is employed
- Witnesses are finite for *one-sided* derivations
- Exponential upper bound on size of witness (with respect to derivation length) for one-sided derivations
- Improvement over Ackermann's Lemma on clause sets
- New correctness proof of SCAN

Conclusion

We showed how to extend SCAN to solve the stronger WSOQE problem for the case of clause sets.

The three problems SOQE, WSOQE and FEQ provide a *common* logical framework for work done on all of these topics.

Future Work

- Construct *finite* witnesses
- Equality reasoning
- Handling Skolemization of input formula
- Quantifier alternations
- Computing witnesses using DLS(*)

References I

[Ack35] Wilhelm Ackermann. „Untersuchungen über das Eliminationsproblem der mathematischen Logik". In: *Mathematische Annalen* 110.1 (1935), pp. 390–413. DOI: 10.1007/BF01448035.

[DLS97] Patrick Doherty, Witold Lukaszewicz, and Andrzej Szalas. „Computing Circumscription Revisited: A Reduction Algorithm". In: *Journal of Automated Reasoning* 18.3 (1997), pp. 297–336. DOI: 10.1023/A:1005722130532.

[GO92] Dov Gabbay and Hans Jürgen Ohlbach. „Quantifier Elimination in Second Order Predicate Logic". In: *South African Computer Journal* 7 (1992), pp. 35–43.