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

Computational Logic Seminar
TU Wien
July 16, 2025

Introduction

Formula Equations (FEQ)

Given $\exists \overline{X}\, \varphi$, where φ is first-order, find first-order predicates $\overline{\alpha}$ such that $\models \varphi[\overline{X} \leftarrow \overline{\alpha}]$, if they exist. We call $\overline{\alpha}$ *FEQ-witnesses*.

- Generalizes problems of software verification, inductive theorem proving, Boolean unification and others
- Undecidable (contains first-order validity problem), but recursively enumerable
- Not studied much in this general setting

Introduction

Second-order quantifier elimination (SOQE)

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

- Applications in modal correspondence theory, forgetting in ontologies and more
- Not recursively enumerable (not even arithmetical[1])
- Prominent algorithms are the saturation-based approach SCAN[2] and the Ackermann[3]-based approach DLS[4]

[1] VD01.
[2] GO92.
[3] Ack35.
[4] 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. $\exists\overline{X}\,\varphi \equiv \varphi[\overline{X} \leftarrow \overline{\alpha}]$, if they exist.
We call the $\overline{\alpha}$ *WSOQE-witnesses*, or simply *witnesses*.

- 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.

Examples

- $\exists X\, X(a)$
  - is valid (equivalent to $\top$)
  - one witness is $\lambda u.\top$, another one is $\lambda u.u \approx a$
- $\exists X\, (X(a) \wedge \forall u\, (X(u) \rightarrow B(u)))$
  - is equivalent to $B(a)$
  - some WSOQE-witnesses are
    - $\lambda u.u \approx a$
    - $\lambda u.B(u)$
    - $\lambda u.u \approx a \vee B(u)$
    - $\lambda u.u \approx a \wedge B(u)$
  - can be solved using Ackermann's lemma

Ackermann's lemma

Lemma

Let φ, ψ be first-order formulas where X only occurs positively in φ and X does not occur in ψ. Then

$$\exists X\,(\varphi \wedge \forall \overline{u}\,(X(\overline{u}) \rightarrow \psi(\overline{u}, \overline{v}))) \equiv \varphi[X \leftarrow \lambda \overline{u}.\psi(\overline{u}, \overline{v})]$$

Let φ, ψ be first-order formulas where X only occurs negatively in φ and X does not occur in ψ. Then

$$\exists X\,(\varphi \wedge \forall \overline{u}\,(\psi(\overline{u}, \overline{v}) \rightarrow X(\overline{u}))) \equiv \varphi[X \leftarrow \lambda \overline{u}.\psi(\overline{u}, \overline{v})]$$

- This is a first method for solving WSOQE!
- However, there are examples it cannot solve, even though witnesses exist

Example where Ackermann's lemma fails

Consider the formula

$$\exists X \,\forall u \forall v \begin{pmatrix} \quad B(a, v) \\ \wedge\ X(a) \\ \wedge\ (B(u, v) \vee \neg X(u) \vee X(v)) \\ \wedge\ \neg X(c) \end{pmatrix}$$

No version of Ackermann's lemma is applicable, but we will show how to construct a witness for this formula.

Outline

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.
  - This means 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*).

- Example:

$$\frac{X(a) \qquad \neg X(u) \vee B(u)}{a \not\approx u \vee B(u)} \text{ Res}$$

Constraint factoring:

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

SCAN Derivation Steps

Constraint elimination

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 $\overline{s}$.

- Standard resolution calculus combines resolution and constraint elimination.
- But we want to derive, e.g., $a \not\approx c$ from $X(a)$ and $\neg X(c)$.
- We often tacitly perform constraint elimination after any inference.

SCAN Derivation Steps

Extended purity deletion

Positive extended purity deletion:

$$\frac{N}{\{C \in N \mid X \text{ does not occur in } C\}}\ \mathrm{ExtPurDel}^{+}_{X}$$

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

- Example:

$$\frac{\{X(a)\}}{\emptyset}\ \mathrm{ExtPurDel}^{+}_{X}$$

- Note that $\lambda u.\top$ is a witness for premise:
  - $\exists X\, X(a) \Rightarrow \exists X\, \emptyset \Rightarrow \top \Rightarrow X(a)[X \leftarrow \lambda u.\top] \Rightarrow \exists X\, X(a)$

SCAN Derivation Steps

Extended purity deletion

Negative extended purity deletion:

$$\frac{N}{\{C \in N \mid X \text{ does not occur in } C\}}\ \text{ExtPurDel}_X^-$$

if for every clause $C \in N$ that contains X, we have that X occurs negatively 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:
  - $\exists X\, N \Rightarrow \exists X\ \{B(a,v)\} \Rightarrow N[X \leftarrow \lambda u.\bot] \Rightarrow \exists X\, N$

SCAN Derivation Steps

Redundancy elimination

- Tautology deletion:

$$\frac{N \cup \{C\}}{N}\ \text{TautDel}$$

  if C is a tautology

- Subsumption deletion:

$$\frac{N \cup \{C\}}{N}\ \text{SubsDel}$$

  if there is a clause $C' \in N$ and a first-order substitution σ such that $C'\sigma \subseteq C$

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 apply 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 Derivations

Example

$$
\begin{array}{r|l}
\{X(a), \neg X(u) \vee B(u)\} & \\
\hline
\{X(a), \neg X(u) \vee B(u), \mathbf{B(a)}\} & \text{Res, ConstrElim} \\
\hline
\{\cancel{X(a)}, \neg X(u) \vee B(u), B(a)\} & \text{PurDel}_{\underline{X(a)}} \\
\hline
\{\cancel{X(a)}, \cancel{\neg X(u) \vee B(u)}, B(a)\} & \text{ExtPurDel}^{-}_{X}
\end{array}
$$

$$
\begin{array}{r|l}
\{X(a), \neg X(u) \vee B(u)\} & \\
\hline
\{X(a), \neg X(u) \vee B(u), \mathbf{B(a)}\} & \text{Res, ConstrElim} \\
\hline
\{X(a), \cancel{\neg X(u) \vee B(u)}, B(a)\} & \text{PurDel}_{\underline{\neg X(u)} \vee B(u)} \\
\hline
\{\cancel{X(a)}, \cancel{\neg X(u) \vee B(u)}, B(a)\} & \text{ExtPurDel}^{+}_{X}
\end{array}
$$

SCAN Algorithm

Derivations

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

- A finite sequence of derivation steps $D = (D_k)_{1 \leq k \leq m}$ is a *derivation* from N if all derivations steps D_k are applicable to N_{k-1}
- D is X-*eliminating* if N_m does not contain X

Outline

Introduction

SCAN Algorithm

Computing Witnesses

Discussion

Computing Witnesses

Approach

- Compute witness iteratively from an X-eliminating derivation $D = (D_k)_{1 \leq k \leq m}$
- For all N_k we want a witness α_k
  - i.e., $\exists X\, N_k \equiv N_k[X \leftarrow \alpha_k]$ for all $0 \leq k \leq m$
- Last clause set N_m contains no X, thus any first-order predicate is a witness
- Transform witness α_k of N_k to a witness α_{k-1} of N_{k-1}

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

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

- α_0 is a witness for $N_0 = N$

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 $T_S(\alpha)$ via

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

Computing Witnesses

Resolution closure of a purified clause

- 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 a purified clause $P = \underline{L(\overline{t})} \vee C$ define $\mathrm{ResU}_P(\overline{c})$ to be the closure of $\left\{L(\overline{c})^{\perp}\right\}$ under (constraint) resolution on P, e.g.,
  - if $P = \underline{\neg X(a)}$, then $\mathrm{ResU}_P(c) = \{X(c), a \not\approx c\}$
  - if $P = B(u, v) \vee \underline{\neg X(u)} \vee X(v)$, then $\mathrm{ResU}_P(c) =$

$$\begin{aligned}\{&X(c),\\ &B(c,v) \vee X(v),\\ &B(c,v) \vee B(v,v') \vee X(v'),\\ &B(c,v) \vee B(v,v') \vee B(v',v'') \vee X(v''),\\ &\ldots\}\end{aligned}$$

Computing Witnesses

Extending Witnesses across purified clause deletion

Define pResU_P via

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

- 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) $B(a, v) \vee X(v)$ (2 with 3)

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

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

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) $B(a, v) \vee X(v)$ (2 with 3)

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

k	D_k	N_k	α_k
0		1, 2, 3, 4	$\text{pResU}_{3.2}[X \leftarrow \alpha_1]$ is infinite!
1	$\text{PurDel}_{3.2}$	1, 2, 4	$\lambda u.u \approx a$
2	$\text{Res}_{2,4}$	1, 2, 4, 6	$\text{pResU}_2[X \leftarrow \lambda u.\bot] \equiv \lambda u.u \approx a$
3	PurDel_2	1, 4, 6	$\lambda u.\bot$
4	ExtPurDel^-_X	1, 6	$\lambda u.W(u)$

Witness Preservation Lemma for PurDel_P

Lemma (Witness Preservation Lemma for PurDel_P)

Consider a purified clause deletion step

$$\frac{N \cup \{P\} := N_P}{N}\ \mathrm{PurDel}_P.$$

where P is purified in N and N is closed under factoring and constraint elimination. Then:

If $\exists X\, N \equiv N[X \leftarrow \alpha]$ *then* $\exists X\, N_P \equiv N_P[X \leftarrow \mathrm{pResU}_P[X \leftarrow \alpha]]$.

If $\exists X\, N \equiv N[X \leftarrow \alpha]$ then $\exists X\, N_P \equiv N_P[X \leftarrow \underbrace{\mathrm{pResU}_P[X \leftarrow \alpha]}_{:=\alpha_P}]$.

Proof sketch.

- Suffices to show $\exists X\, N_P \Rightarrow N_P[X \leftarrow \alpha_P]$.
- Since $N \subseteq N_P$ we have $\exists X\, N_P \Rightarrow \exists X\, N$.
- α is witness for N, therefore $\exists X\, N \Rightarrow N[X \leftarrow \alpha]$.
- Remains to show $N[X \leftarrow \alpha] \Rightarrow N_P[X \leftarrow \alpha_P]$.
- This reduces to $N[X \leftarrow \alpha] \Rightarrow N[X \leftarrow \alpha_P]$ and $N[X \leftarrow \alpha] \Rightarrow P[X \leftarrow \alpha_P]$.

□

Lemma

Let P be a pointed clause and let C be a clause. Then $\models \mathrm{Res}_P(C) \rightarrow C[X \leftarrow \mathrm{pResU}_P]$ *and* $\models P[X \leftarrow \mathrm{pResU}_P]$.

Outline

Further results

- Witnesses are finite if no redundancy is employed
- Witnesses are finite for *one-sided* derivations
  - pointed clause P is *one-sided* if X occurs in P only positively or only negatively
  - derivation D is *one-sided* if all purified clause deletions are performed on one-sided pointed clauses
- 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
- Prototype implementation in GAPT[5]

[5] `https://logic.at/gapt/`

Limitations

- Currently open how to always ensure *finite* witnesses when SCAN terminates in the presence of redundancy criteria
- There are formulas where SCAN terminates, but no witnesses exist, e.g. $\exists X \, \exists u \, \exists v \, (X(u) \land \neg X(v))$ is equivalent to $\exists u \, \exists v \; u \neq v$, but it can be shown that no witness exists
  - Could skolemize, but then all witnesses contain Skolem symbols which can be undesirable
- Quantifier alternations: Consider the dual WSOQE-problem: given $\forall \overline{X} \, \varphi$, where φ is first-order, find first-order predicates $\overline{\alpha}$ such that $\forall \overline{X} \, \varphi \equiv \varphi[\overline{X} \leftarrow \overline{\alpha}]$.
  - Note that $\overline{\alpha}$ is a witness for the dual problem iff it is a witness for $\exists \overline{X} \, \neg\varphi$.
  - Introduces a negation on the input formula.
  - If input is a clause set, the negation would in general not be a clause set anymore

Conclusion

We showed how to extend SCAN to solve the more general 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
- 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.

References II

[VD01] Johan Van Benthem and Kees Doets. „Higher-Order Logic“. In: *Handbook of Philosophical Logic*. Ed. by D. M. Gabbay and F. Guenthner. Dordrecht: Springer Netherlands, 2001, pp. 189–243. ISBN: 978-94-015-9833-0. DOI: 10.1007/978-94-015-9833-0_3. URL: https://doi.org/10.1007/978-94-015-9833-0_3.