Computing Witnesses Using the SCAN Algorithm

Fabian Achammer, Stefan Hetzl, Renate Schmidt

Institute of Discrete Mathematics and Geometry
TU Wien
and
Department of Computer Science
University of Manchester

Formal Methods Seminar
University of Manchester
November 20, 2024

Introduction

Formula Equations (FEQ)

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

- Generalizes problems of software verification, inductive theorem proving, Boolean unification and others
- Undecidable in general (contains first-order validity problem)
- 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 $\models \exists \overline{X}\, \varphi \leftrightarrow \psi$, if it exists.

- Applications in modal correspondence theory, forgetting in ontologies and more
- Undecidable in general
- 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 between FEQ and SOQE: WSOQE

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

- If we can solve WSOQE, we reduce FEQ to first-order validity checking

Contribution of this talk:

- If φ is given as a clause set and SCAN terminates on $\exists\overline{X}\,\varphi$, we can construct corresponding WSOQE-witnesses, but they are potentially infinite.

Outline

Examples

SCAN Algorithm

Computing Witnesses

Dicussion & Conclusion

Examples

Use lambda notation to denote first-order predicates that can be substituted for second-order variables X.
For this talk we assume that we operate on clause sets N and the only existential quantifier is over X

- $\exists X\, X(a)$
  - 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

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

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

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

- 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

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

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

Outline

SCAN Algorithm

- Approach of SCAN is to saturate input clause set N according to $\exists X$-equivalence-preserving derivation steps
- 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 Algorithm

Extended purity deletion

Positive extended purity deletion:

$$\frac{N}{\{C \in N \mid X \text{ does not occur in } C\}} \; \mathsf{ExtPur}_X^+$$

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

- Example:

$$\frac{\{X(a)\}}{\emptyset} \; \mathsf{ExtPur}_X^+$$

- Note that $\lambda u.\top$ is a witness for premise

SCAN Algorithm

Extended purity deletion

Negative extended purity deletion:

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

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

- Example:

$$\frac{\{\neg B(a,v),\ \neg B(u,v) \vee \neg X(u) \vee X(v),\ \neg X(c),\ a \not\approx c\}}{\{\neg B(a,v),\ a \not\approx c\}}\ \mathrm{ExtPur}_X^-$$

- Note that $\lambda u.\bot$ is a witness for the premise and substitution gives clause set equivalent to conclusion

SCAN Algorithm

Inference steps

Constraint resolution:

$$\frac{C \vee X(\overline{t}) \qquad C' \vee \neg X(\overline{s})}{C \vee C' \vee \overline{t} \not\approx \overline{s}}\ \text{Res}$$

- Example:

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

- Side note: We often tacitly perform necessary renaming of variables so both clauses have disjoint variables

Constraint factoring:

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

Possible to add negative factoring as well

SCAN Algorithm

Purified clause deletion

- Clause $K = X(\bar{t}) \vee C$ (or $K = \neg X(\bar{t}) \vee C$) is *purified* with respect to the literal $(\neg)X(\bar{t})$ in a clause set N, if all possible derivation steps between this literal and N are redundant in N
- Use underlining to denote the literal with respect to which the clause is purified, e.g. $\underline{X(\bar{t})} \vee C$ (or $\underline{\neg X(\bar{t})} \vee C$)
- Purified clause deletion:

$$\frac{N \cup \{K\}}{N} \text{ Pur}_K$$

if $K = \underline{(\neg)X(\bar{t})} \vee C$ is purified with respect to $(\neg)X(\bar{t})$ in N

SCAN Algorithm

Example

$$\cfrac{\cfrac{\cfrac{\{X(a), \neg X(u) \vee B(u)\}}{\{X(a), \neg X(u) \vee B(u), \mathbf{a} \not\approx \mathbf{u} \vee \mathbf{B}(\mathbf{u})\}}\ \text{Res}}{\{\neg X(u) \vee B(u), a \not\approx u \vee B(u)\}}\ \text{Pur}_{\underline{X(a)}}}{\{a \not\approx u \vee B(u)\}}\ \text{ExtPur}^{-}_{X}$$

$$\cfrac{\cfrac{\cfrac{\{X(a), \neg X(u) \vee B(u)\}}{\{X(a), \neg X(u) \vee B(u), \mathbf{a} \not\approx \mathbf{u} \vee \mathbf{B}(\mathbf{u})\}}\ \text{Res}}{\{X(a), a \not\approx u \vee B(u)\}}\ \text{Pur}_{\underline{\neg X(u)} \vee B(u)}}{\{a \not\approx u \vee B(u)\}}\ \text{ExtPur}^{+}_{X}$$

SCAN Algorithm

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 $C'\sigma \subseteq C$

- Constraint elimination:

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

  where σ is a most general unifier of t and s.
  - Side note: We often tacitly perform constraint elimination after any inference step
- Possibly other $\forall X$-equivalence-preserving simplifications

SCAN Algorithm

Derivations

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

- A sequence of derivation steps $D = (D_k)_{1 \leq k \leq m}$ is a *derivation* from N if all D_k are applicable to N_{k-1}
- D is *saturated* if it cannot be extended to a longer sequence

Outline

Computing Witnesses

Approach

- Compute witness iteratively from a saturated derivation $D = (D_k)_{1 \leq k \leq m}$
- For all N_k we want a witness χ_k, i.e. it holds $\models (\exists X\, N_k) \leftrightarrow N_k[X \leftarrow \chi_k]$ for all $0 \leq k \leq m$
- Last clause set N_m contains no X (D is saturated!), thus any first-order predicate is a witness (we choose $\lambda \overline{u}.\bot$)
- Then extend witness χ_k of N_k to a witness χ_{k-1} of N_{k-1} across the derivation step D_k for all $1 \leq k \leq m$ using an operation that takes D_k as input
- χ_0 is a witness for $N_0 = N$

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

$$\chi_0 \xleftarrow{\text{ext}} \chi_1 \quad \xleftarrow{\text{ext}} \quad \dots \quad \xleftarrow{\text{ext}} \chi_{m-1} \quad \xleftarrow{\text{ext}} \chi_m = \lambda \overline{u}.\bot$$

Computing Witnesses

Extending Witnesses across derivation steps

We define a first-order predicate $\mathrm{ext}(D_k, \chi)$ via

$$\begin{aligned}
\mathrm{ext}(\mathsf{Res}, \chi) &= \chi \\
\mathrm{ext}(\mathsf{Fac}, \chi) &= \chi \\
\mathrm{ext}(\mathsf{TautDel}, \chi) &= \chi \\
\mathrm{ext}(\mathsf{SubsDel}, \chi) &= \chi \\
\mathrm{ext}(\mathsf{ConstrElim}, \chi) &= \chi \\
\mathrm{ext}(\mathsf{ExtPur}^{+}_{\mathsf{X}}, \chi) &= \lambda \overline{u}.\top \\
\mathrm{ext}(\mathsf{ExtPur}^{-}_{\mathsf{X}}, \chi) &= \lambda \overline{u}.\bot \\
\mathrm{ext}(\mathsf{Pur}_{\mathsf{K}}, \chi) &= \mathrm{res}_K[X \leftarrow \chi]
\end{aligned}$$

Computing Witnesses

Resolution closure of a purified clause

- Recall purified clause deletion:
$$\frac{N \cup \{K\}}{N}\ \mathrm{Pur}_K$$
if $K = \underline{(\neg)X(\bar{t})} \vee C$ is purified with respect to $(\neg)X(\bar{t})$ in N
- For a purified clause $K = \underline{\neg X(\bar{t})} \vee C$ define Res^*_K to be the closure of $\{X(\overline{u})\}$ using constraint resolution on K, e.g.
  - if $K = \underline{\neg X(a)}$, then $\mathrm{Res}^*_K = \{X(u), a \not\approx u\}$
  - if $K = \neg B(u, v) \vee \underline{\neg X(u)} \vee X(v)$, then $\mathrm{Res}^*_K =$
$$\begin{aligned}\{&X(u),\\ &\neg B(u,v) \vee X(v),\\ &\neg B(u,v) \vee \neg B(v,v') \vee X(v'),\\ &\neg B(u,v) \vee \neg B(v,v') \vee \neg B(v',v'') \vee X(v''),\\ &\ldots\}\end{aligned}$$
- Analogous definition, if $K = \underline{X(\bar{t})} \vee C$, but perform closure of $\{\neg X(\overline{u})\}$ under constraint resolution with K

Computing Witnesses

Extending Witnesses across purified clause deletion

Define res_K via

$$\mathrm{res}_K = \begin{cases} \lambda\overline{u}.\, \bigwedge_{R(\overline{u},\overline{v})\in \mathrm{Res}^*_K} \forall \overline{v}\, R(\overline{u},\overline{v}) & \text{if } K = \underline{\neg X(\overline{t})} \vee C \\ \lambda\overline{u}.\, \bigvee_{R(\overline{u},\overline{v})\in \mathrm{Res}^*_K} \exists \overline{v}\, \neg R(\overline{u},\overline{v}) & \text{if } K = \underline{X(\overline{t})} \vee C \end{cases}$$

- res_K is potentially infinite!

Computing Witnesses

Example

(1) $\neg B(a, v)$

(2) $X(a)$

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

(4) $\neg X(c)$

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

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

k	D_k	N_k	χ_k
0		$1, 2, 3, 4$	$\lambda u.u \approx a \longleftarrow$ **obtained witness**
1	$\mathrm{Res}_{2,4}$	$1, 2, 3, 4, 6$	$\mathrm{res}_2[X \leftarrow \lambda u.\bot] = \lambda u.u \approx a$
2	Pur_2	$1, 3, 4, 6$	$\lambda u.\bot$
3	ExtPur_X^-	$1, 6$	$\lambda u.\bot$

Computing Witnesses

Example

(1) $\neg B(a, v)$

(2) $X(a)$

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

(4) $\neg X(c)$

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

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

k	D_k	N_k	χ_k
0		$1, 2, 3, 4$	$\mathrm{res}_{3.2}[X \leftarrow \chi_1]$ is infinite!
1	$\mathrm{Pur}_{3.2}$	$1, 2, 4$	$\lambda u. u \approx a$
2	$\mathrm{Res}_{2,4}$	$1, 2, 4, 6$	$\mathrm{res}_2[X \leftarrow \lambda u. \bot] = \lambda u. u \approx a$
3	Pur_2	$1, 4, 6$	$\lambda u. \bot$
4	ExtPur^{-}_{X}	$1, 6$	$\lambda u. \bot$

Outline

Discussion

- Currently still open whether the termination of SCAN on clause sets ensures a *finite* witness in the presence of redundancy:
  - True, if we omit redundancy criteria
  - Also true, if the derivation has the property that purified clause deletion only occurs on clauses where X occurs with a single polarity
  - We conjecture that we can construct such a derivation from any given saturated derivation
- 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

Conclusion

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

What we're currently looking at:

- If SCAN terminates, is there always a *finite* witness?
- Investigate classes where SCAN terminates, including the modal logic Sahlqvist class
- Finding and characterizing extended classes with SCAN-based finite witnesses
- How can Skolemization be handled in witness generation?

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.