Quantifier Elimination via Functional Composition

Jie-Hong Roland Jiang

Dept. of Electrical Eng. / Grad. Inst. of Electronics Eng.
National Taiwan University
Taipei 10617, Taiwan

Outline

- Motivations
- Prior work
- Quantifier elimination by functional composition
  - Propositional logic
  - Predicate logic
- Experimental results
- Conclusions

Introduction

- Quantifier elimination transforms a *quantified formula*, e.g., $\exists x_1 \forall x_2 \exists x_3 \cdots \forall x_n\, \varphi$, into an equivalent *quantifier-free formula* ψ
  - ψ can be preferable to $\exists x_1 \forall x_2 \exists x_3 \cdots \forall x_n\, \varphi$

    E.g.,
    - Properties of ψ can be reasoned more easily
    - ψ can be treated as a synthesis result for implementation

Introduction

- QE examples
  - Gauss elimination for systems of linear equalities
  - Fourier-Motzkin elimination for systems of linear inequalities
  - Cylindrical algebraic decomposition for systems of polynomial inequalities

Motivations

- QE arises in many contexts, including computation theory, mathematical logic, optimization, …
  - Constraint reduction
  - Quantified Boolean Formula (QBF) solving

Main focus

- Propositional logic
  - Quantifier elimination for QBFs

Prior work

- Formula expansion
  - $\exists y\ \varphi(\mathbf{x},y) = \varphi(\mathbf{x},0) \vee \varphi(\mathbf{x},1)$
  - BDD, AIG based image-computation [Coudert90][Pigorsch06]

- Normal-form conversion
  - Existential (universal) quantification is computationally trivial for disjunctive (conjunctive) normal form formulas
    - Simply remove from the formula the literals of variables to be quantified
      E.g., $\forall x_1[(x_1 \vee x_2 \vee x_3)(\neg x_1 \vee x_3)(x_2 \vee x_4)] = (x_2 \vee x_3)(x_3)(x_2 \vee x_4)$
  - Formula conversion between CNF and DNF [McMillan02]

- Solution enumeration
  - Compute $\psi(\mathbf{x}) = \exists \mathbf{y}\ \varphi(\mathbf{x},\mathbf{y})$ by enumerating all satisfiable assignments on $\mathbf{x}$
  - SAT-based image computation, e.g., [Ganai04]

- Yet another way?

Question

- Given a quantified formula $\exists y\ \varphi(\mathbf{x},y)$, what should a function *f* be such that $\varphi(\mathbf{x},f(\mathbf{x})) = \exists y\ \varphi(\mathbf{x},y)$?
- I.e., QE by **functional composition**

Answer

- $\varphi(\mathbf{x},f(\mathbf{x})) = \exists y\ \varphi(\mathbf{x},y)$ if and only if
  - f has

    care onset $\varphi(\mathbf{x},1) \wedge \neg\varphi(\mathbf{x},0)$

    care offset $\varphi(\mathbf{x},0) \wedge \neg\varphi(\mathbf{x},1)$

    don't care set $\varphi(\mathbf{x},1) \equiv \varphi(\mathbf{x},0)$

  - In other words,

    $(\varphi(\mathbf{x},1) \wedge \neg\varphi(\mathbf{x},0)) \leq f \leq \neg(\varphi(\mathbf{x},0) \wedge \neg\varphi(\mathbf{x},1))$

  - Such f always exists

Problem formulation

- For universal quantification
  $\forall y\ \varphi(\mathbf{x},y) = \neg\exists y\ \neg\varphi(\mathbf{x},y) = \neg\neg\varphi(\mathbf{x},f(\mathbf{x})) = \varphi(\mathbf{x},f(\mathbf{x}))$
  - *f* has
    care onset $\neg\varphi(\mathbf{x},1) \wedge \varphi(\mathbf{x},0)$
    care offset $\neg\varphi(\mathbf{x},0) \wedge \varphi(\mathbf{x},1)$
    don't care set $\varphi(\mathbf{x},1) \equiv \varphi(\mathbf{x},0)$
- So by computing composite functions *f*, one can iteratively eliminate the quantifiers of any QBF

Computation

- *f* can be computed by
  - Binary decision diagrams (BDDs)
    - Not scalable for large φ
  - Craig interpolation

Craig interpolation

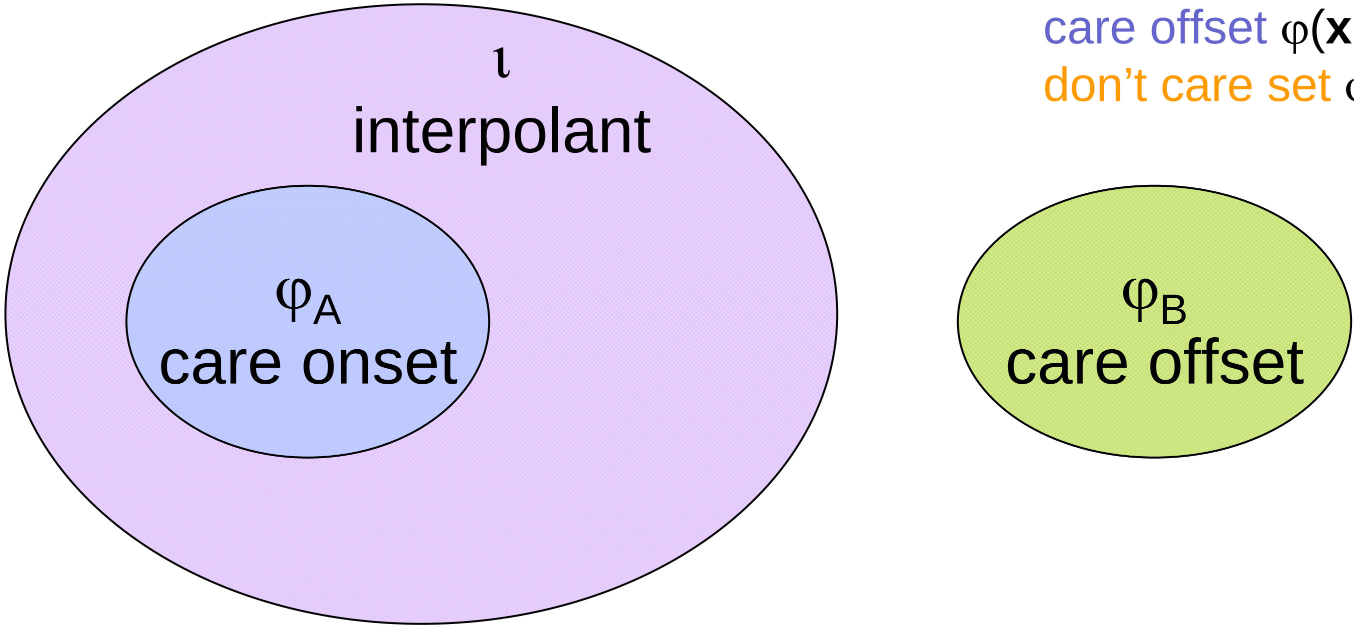
- (Propositional logic)

For $\varphi_A \wedge \varphi_B$ unsatisfiable, there exists an interpolant ι of φ_A w.r.t. φ_B such that

1. $\varphi_A \Rightarrow \iota$
2. $\iota \wedge \varphi_B$ is unsatisfiable
3. ι refers only to the common variables of φ_A and φ_B

Computation

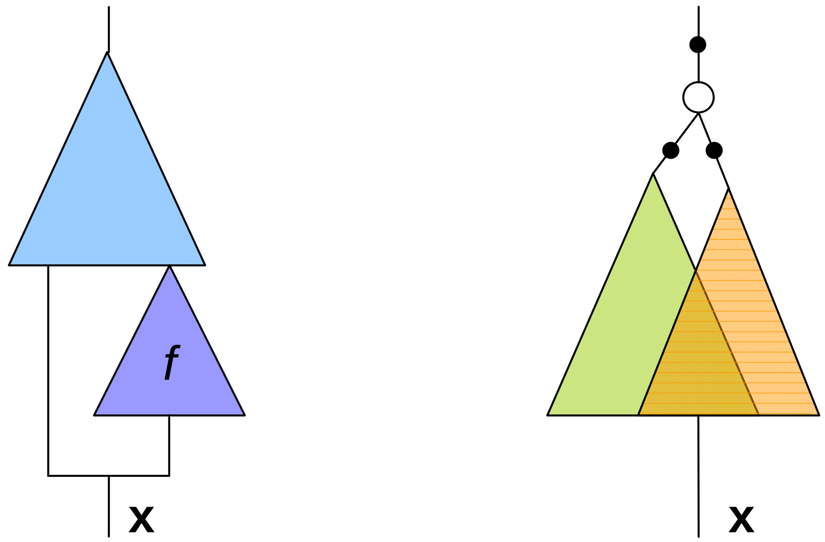
care onset $\varphi(\mathbf{x},1) \wedge \neg\varphi(\mathbf{x},0)$
care offset $\varphi(\mathbf{x},0) \wedge \neg\varphi(\mathbf{x},1)$
don't care set $\varphi(\mathbf{x},1) \equiv \varphi(\mathbf{x},0)$



The interpolant is a valid implementation of *f*, which can be obtained from the refutation of $\varphi_A \wedge \varphi_B$ in SAT solving and can be naturally represented in **And-Inverter Graphs (AIGs)**

Composition vs. expansion

- Is $\varphi(\mathbf{x},f(\mathbf{x}))$ better than $\varphi(\mathbf{x},0) \vee \varphi(\mathbf{x},1)$?



in terms of AIGs, where structurally identical nodes are merged

Composition vs. expansion

- [∃] Consider simplifying $\varphi(\mathbf{x},1)$ in $\varphi(\mathbf{x},0) \vee \varphi(\mathbf{x},1)$ using $\varphi(\mathbf{x},0)$ as don't care

  care onset $\varphi(\mathbf{x},1) \wedge \neg\varphi(\mathbf{x},0)$

  care offset $\neg\varphi(\mathbf{x},1) \wedge \neg\varphi(\mathbf{x},0)$

  In contrast to *f* with

  care onset $\varphi(\mathbf{x},1) \wedge \neg\varphi(\mathbf{x},0)$

  care offset $\varphi(\mathbf{x},0) \wedge \neg\varphi(\mathbf{x},1)$

  For existential quantification, composition can be much better than expansion for sparse φ (due to simple interpolants)

Composition vs. expansion

- [$\forall$] Consider simplifying $\varphi(\mathbf{x},1)$ in $\varphi(\mathbf{x},0) \wedge \varphi(\mathbf{x},1)$ using $\neg\varphi(\mathbf{x},0)$ as don't care

  care onset $\varphi(\mathbf{x},1) \wedge \varphi(\mathbf{x},0)$

  care offset $\neg\varphi(\mathbf{x},1) \wedge \varphi(\mathbf{x},0)$

  In contrast to *f* with

  care onset $\neg\varphi(\mathbf{x},1) \wedge \varphi(\mathbf{x},0)$

  care offset $\neg\varphi(\mathbf{x},0) \wedge \varphi(\mathbf{x},1)$

  For universal quantification, composition can be much better than expansion for dense φ (due to simple interpolants)

Generalization to predicate logic

- For a language $\mathcal{L}$ in predicate logic under structure (interpretation) $\mathcal{I}$,

  $$\models_{\mathcal{I}} \forall \mathbf{x}(\exists y\ \varphi(\mathbf{x},y) = \exists F\ \varphi(\mathbf{x},F\mathbf{x}))$$

  - QE is possible if such function F is finitely expressible in the language
    - If $\exists y\ \varphi(\mathbf{x},y) = \varphi(\mathbf{x},f\mathbf{x})$, then $\varphi(a,b) \vee \neg\exists y\ \varphi(a,y)$ is satisfied for any a, b with $f(a)=b$
    - If for any a, b with $f(a)=b$ satisfies $\varphi(a,b) \vee \neg\exists y\ \varphi(a,y)$, then

      $\exists y\ \varphi(\mathbf{x},y) = \bigvee (\gamma_i \wedge \varphi(\mathbf{x},f_i\mathbf{x}))$, where $f = f_i$ if γ_i holds
    - $\{(a,b) \mid \varphi(a,b) \vee \neg\exists y\ \varphi(a,y)\}$ characterizes the flexibility of f, which can be exploited to simplify QE

Generalization to predicate logic

Example

$\exists x(a \cdot x^2+c=0)$ over the real number

$$f(a,c) = \begin{cases} (-c/a)^{1/2} & \text{if } c/a \leq 0 \\ - & \text{if } c/a > 0 \end{cases}$$

Taking $f(a,c) = (((-c/a)^2)^{1/2})^{1/2}$, this quantified formula is equivalent to

$a \cdot ((((-c/a)^2)^{1/2})^{1/2})^2+c=0$

Experiments

- Given a sequential circuit, we compute its transition relation with input variables being quantified out, i.e.,

$$\exists \mathbf{x} \, [\bigwedge_i \, (s_i' \equiv \delta_i(\mathbf{x},\mathbf{s}))]$$

  - Simple quantification scheduling applied
  - AIG minimization applied

Experimental results

circuit	(#in, #reg, #n, #l)	rel before QE		QE-Exp				QE-Cmp			
		#n	#l	#n	#l	time	mem	#n	#l	time	mem
prolog	(36, 136, 1656, 26)	1474	29	—	—	—	—	**1088**	**31**	6.27	38.0
s1196	(14, 18, 529, 24)	548	22	**3473**	**21**	5.15	37.3	21881	2532	123.15	37.3
s1269	(18, 37, 569, 35)	622	37	31005	**39**	59.24	37.5	**1694**	116	41.05	37.5
s13207.1	(62, 638, 8027, 59)	5272	45	—	—	—	—	**4741**	**44**	50.60	40.6
s1423	(17, 74, 657, 59)	757	63	17619	**59**	25.45	38.1	**3142**	452	6.19	38.1
s1488	(8, 6, 653, 17)	686	19	1269	**21**	2.90	38.1	**515**	48	3.82	38.1
s1494	(8, 6, 647, 17)	696	20	1261	**21**	2.98	38.1	**607**	42	2.54	38.1
s1512	(29, 57, 780, 30)	697	28	1187	**24**	2.64	37.7	**823**	53	3.78	37.7
s15850.1	(77, 534, 9786, 82)	5679	57	—	—	—	—	**180597**	**14247**	49409.27	427.4
s208.1	(10, 8, 104, 11)	103	14	65	**11**	0.08	37.4	**49**	12	0.06	37.4
s298	(3, 14, 119, 9)	157	15	**117**	**12**	0.08	37.4	122	**12**	0.23	37.4
s3271	(26, 116, 1573, 28)	1565	32	**1549**	**29**	3.08	38.0	1604	62	7.11	38.0
s3330	(40, 132, 1789, 29)	1434	29	—	—	—	—	**1029**	**28**	6.37	38.0
s3384	(43, 183, 1702, 60)	1801	63	1307	**58**	6.94	38.3	**1276**	**58**	17.29	38.3
s344	(9, 15, 160, 20)	164	19	**140**	**19**	0.33	37.1	155	**19**	0.81	37.1
s349	(9, 15, 161, 20)	168	19	**140**	**19**	0.26	37.5	155	**19**	0.82	37.5
s382	(3, 21, 158, 9)	220	19	**179**	**16**	0.10	37.7	189	**16**	0.27	37.7
s38417	(28, 1636, 22397, 47)	15762	44	**15705**	**40**	44.79	48.7	18865	106	149.13	46.8
s38584.1	(38, 1426, 19407, 56)	18094	48	57105	**45**	1382.97	71.4	**38089**	1362	268.94	46.0
b12	(5, 121, 952, 19)	1485	26	**1740**	**24**	0.65	38.3	1908	41	2.21	38.3
b13	(10, 53, 299, 20)	472	20	435	**16**	0.49	37.6	**423**	**16**	1.15	37.6
ratio				1.000	1.000	1.000	1.000	0.036	8.064	0.013	0.952

Discussion

- Expansion vs. composition based QE
  - Analogy with two-level vs. multi-level circuit minimization
    - Relaxing level constraints admits more compact circuit representation
- Sparsity may play an essential role in the effectiveness of composition-based QE

Conclusions

- Quantifier elimination with functional composition can be effective at least for some applications (where the sparsity condition holds)
- Future work
  - Find more applications
  - QE in predicate logic

Thanks for your attention!

Questions?