Learning Reductions with Descriptive Complexity and SAT Solvers

Charles Jordan and Łukasz Kaiser

JST ERATO and LIAFA, CNRS

July 26, 2013

The Magic of SAT

$$\textbf{find } x : \text{ ``}x \textbf{ is good''}$$

Reduction Finding

$$\textbf{find } r : \ \forall x \, (x \in P \leftrightarrow r(x) \in Q)$$

Beyond SAT

$$\textbf{find } f : \ \forall x \text{ ``}f(x) \textbf{ is good''}$$

Questions

- How to represent r, P, Q and x?
- How to approach the problem? **(CEGAR vs QBF vs ASP)**
- How do current tools perform?

Motivation

Finding Simple Objects

Often, we look for simple objects

- Gadgets, simpler formulas, reductions, counter-examples.

Computers can help!

Related Work

- J., K. Experiments with Reduction Finding. SAT 2013.
- J., K. Benchmarks from Reduction Finding. QBF 2013.
- J., K. Learning Programs as Logical Queries. LTC 2013.
- Carmosino, Immerman, J. Experimental Descriptive Complexity. Kozen Festschrift, 2012.
- Itzhaky, Gulwani, Immerman, Sagiv. A simple inductive synthesis methodology and its applications. OOPSLA 2010.
- Crouch, Immerman, Moss. Finding Reductions Automatically. Gurevich Festschrift, 2010.

Overview

What is new?

- Unify work on synthesis and descriptive complexity
- Free, open reduction-finding tools
- Comparison of many approaches
- Benchmark instances

Implementations

1. http://toss.sf.net/reduct.html
2. http://toss.sf.net/reductGen.html
3. http://www-erato.ist.hokudai.ac.jp/~skip/de
4. http://toss.sf.net/gameGen.html

Focus on *performance* of tools.

How do we represent reductions?

Representing Reductions

$$\textbf{reduction } r: \ \forall x \, (x \in P \leftrightarrow r(x) \in Q)$$

Standard reductions

- r is a (ptime, logspace, ...) Turing machine
- x is a word
- P,Q are sets of words given by Turing machines

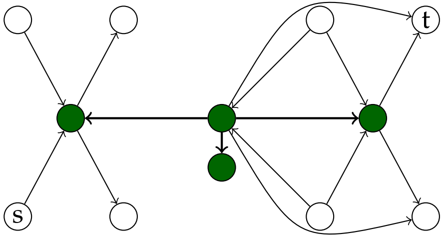
Reductions in logic

- r is a (quantifier-free, first-order, ...) query
- x is a relational structure
- P,Q are sets of structures given by formulas

Question: is there a useful correspondence?

Relational Structures and Logics

Relational Structures $A = (U, R_1^A, R_2^A, \ldots, R_r^A, c_1^A, \ldots, c_d^A)$



First-Order and Second-Order Logic over $\sigma = \{E\}$

Clique (FO): $\forall x, y\colon (x = y \vee E(x,y))$

3-colorable ($\exists$SO):

$$\exists R, G, B\ \forall x, y : \ (R(x) \vee G(x) \vee B(x)) \quad \wedge$$
$$\big(E(x,y) \to \neg\,(\ (R(x) \wedge R(y))\ \vee\ (G(x) \wedge G(y))\ \vee\ (B(x) \wedge B(y))\ )$$

Descriptive Complexity

Decision problem P

Computational complexity: resources needed to check it

Descriptive complexity: **expressive power** needed to define it

The two notions are **isomorphic**!
"Hard to check" $\equiv$ "Requires expressive language to define"

Characterizations of complexity classes

- PSPACE = SO(TC)
- NP = SO$\exists$, coNP = SO$\forall$, PH = SO
- P = FO(LFP)
- NL = FO(TC) ...

Queries

Queries (Interpretations) $q = (k, \varphi_0, \varphi_1, \ldots, \varphi_r, \psi_1, \ldots, \psi_d)$

- k is the dimension
- $\varphi_0(x_1, \ldots, x_k)$ defines the new universe
- $\varphi_i(x_1, \ldots, x_{ka_i})$ define the new relations
- $\psi_j(x_1, \ldots, x_k)$ define the new constants

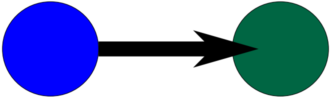
Example

$$\left(\; k = 2,\; \varphi_0 = \top,\; \varphi(x_1, x_2, x_3, x_4) = (x_1 = x_3) \wedge E(x_2, x_4) \;\right)$$

Choice of logic $\equiv$ Complexity of query

Reductions

reduction $r: \; \forall x \, (x \in P \leftrightarrow r(x) \in Q)$



Weak Reductions Suffice!
Quantifier-free projections suffice for natural problems.
No need to prove things can't be done in polytime[1].

Example: `http://toss.sf.net/reduct.html`

Same Example:

$$q := \langle k := 1, \;\; \varphi_0 := \top, \;\; \varphi_1(x_1, x_2) := x_1 = s \vee x_2 = t \vee E(x_2, x_1) \, \rangle$$

[1]E.g., Berman-Hartmanis conjecture *holds* for first-order projections!

Decidability and Parameters

$$\exists r : \forall x : (x \in P \leftrightarrow r(x) \in Q)$$

But this formula is infinite!

Decidability via Parameters:

- k dimension of the reduction
- n size of examples (x)
- c number of conjunctions in DNF

The formula is finite!
In spirit, a Σ_2^p problem.

How to approach the problem?

CEGAR vs QBF vs ASP

Approaches to Σ_2^p

Quantified Boolean Formula (QBF) Solvers

- PSPACE-complete, one call suffices
- CNF-conversion, prenexing: *problematic*
- 3QBF CNF, 2QBF CNF (negated), qpro (NNF), CQBF (experimental)

Disjunctive Answer Set Program (ASP) Solvers

- Disjunctive $\equiv \Sigma_2^p$, one call suffices
- Faber, Ricca. Solving hard ASP programs efficiently NMR 2005.
- Few solvers: claspD, cmodels, gnt(2)

Counter-example guided abstraction refinement (CEGAR)

- Crouch, Immerman, Moss. Finding reductions automatically. Gurevich Festschrift, 2010.
- Janota, Marques-Silva. Abstraction-based algorithm for 2QBF. SAT 2011.
- Janota, Klieber, Marques-Silva, Clarke. Solving QBF with counterexample guided refinement. SAT 2012.

CEGAR, SAT and Σ_2^p

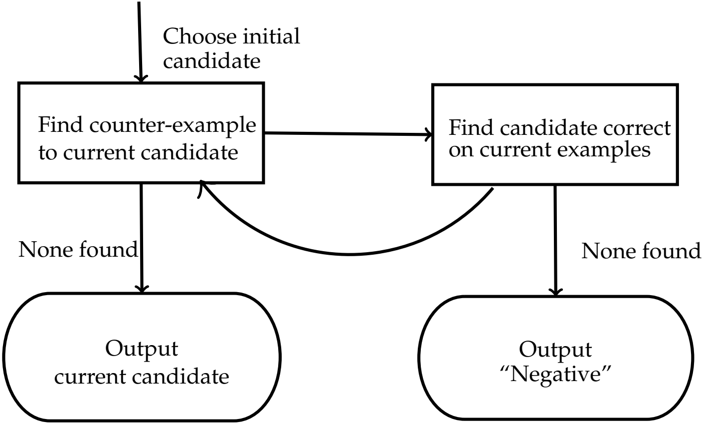


Figure: **C**ounter-**e**xample **g**uided **a**bstraction **r**efinement (CEGAR)

How do current tools perform?

Compared

Instances
2304 instances per parameter set, 6+ sets, plus hard instances

DE (incremental)[2]
MiniSat-2[3], GlueMiniSat, CryptoMiniSat, BDD/CUDD

Toss (not incremental)
MiniSat-2, GlueMiniSat, Intel Decision Procedure Toolkit

QBF (qdimacs+nqdimacs)
rareqs, depqbf, QuBE, sKizzo, CirQit (qpro too)

ASP
(lparse, gringo) $\times$ (gnt2, cmodels, claspd)

Reduction Finder

---

[2]Preliminary runs with lingeling, treengeling, plingeling, PMSat, . . .
[3]No simplification – `https://github.com/niklasso/minisat/issues/3`

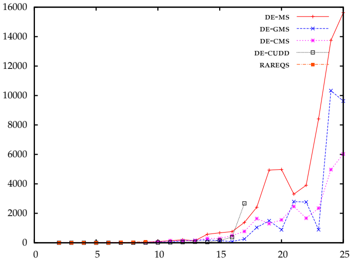
Reduction Finding Results

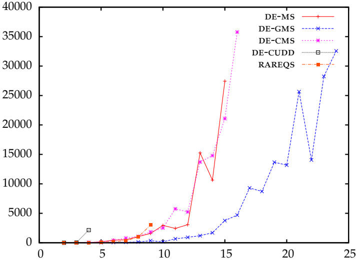
 # Unsolved cases of $48 \times 48 = 2304$: **CEGAR** vs **QBF** vs **claspD**

(c, n)	$(1,3)$	$(2,3)$	$(3,3)$	$(1,4)$	$(2,4)$	$(3,4)$
DE-GMS	0	0	10	0	5	103
DE-CUDD	0	116	537	0	186	722
RAREQS	0	0	16	19	65	204
DEPQBF	0	142	547	16	297	711
QUBE	10	536	949	82	760	1082
CIRQIT	58	673	1138	511	1092	1357
CIRQIT′	157	523	903	–	–	–
SKIZZO	522	1058	1156	975	1327	1434
GRINGO	40	393	590	72	593	836
LPARSE	51	396	605	75	635	850
RedFind	1	152	396	2	347	547

CEGAR Results

$\overline{\textbf{REACH}}$ to REACH, $k = 1$, scaling n with $c = 1, 2$





Dimension 2 ($c = 1$)

	DE-MS	DE-GMS	DE-CMS	DE-CUDD	RAREQS
$k = 1, n = 3$	0.05	0.06	0.08	0.07	0.03
$k = 2, n = 2$	0.06	0.11	0.28	6.30	0.06
$k = 2, n = 3$	3562.14	1696.26	1755.03	timeout	3267.10

QBF Gallery 2013 (Lonsing, Seidl, van Gelder)

14 QBF solvers on random sample of $k = 1, c = 3, n = 4$.

# of instances (of 150) solved in 900s.

http://www.kr.tuwien.ac.at/events/qbfgallery2013/

Outlook

What can we do?

- Simple evaluation and reduction finding
- `http://www-erato.ist.hokudai.ac.jp/~skip/de`
- `http://toss.sf.net/reduct.html`
- Useful as a *debugger*!
- Source of uniform instances. parameters $\rightarrow$ hardness

What is hard?

- high-dimensional reductions
- symmetry breaking in example finding problems
- using GPUs, massively parallel machines

Related Future Work

Finding Fast Programs

- Learn LFP equivalents to SO
- Examples: parity games, graph isomorphism, SAT
- **LFPTest.native**

Solving Games

- Does Player 1 win?
- `http://toss.sf.net/gameGen.html`
- **Note:** CEGAR loses to other solvers!

Learning Games

- Given set of example plays, learn rules
- Examples: Connect4, gomoku, chess
- J.,K. Learning Programs as Logical Queries, LTC 2013.

Much More!

Reason to Hope

Ranges
Start with size-2 examples, then move to 3. . .
Very big performance gain. Not enough for $k = 3$.

Encodings & More
QBF/SAT 2013 were inspiring – much to do!

Parallel & supercomputing
Cube and conquer? (march, treengeling, . . .)

Benchmarks
New QBF benchmarks, new QBF formats, ASP/etc. benchmarks in progress

New Ideas and Approaches?
We're new!

Thank you!