Combining Algebraic Reasoning with Propositional Solvers

Thomas Sturm

Departamento de Matemáticas, Estadística y Computación
Facultad de Ciencias, Universidad de Cantabria, Santander, Spain

Aachen, Germany, May 28, 2009

Overview

- Reduce and Redlog
- Real Quantifier Elimination and Variants
- Non-Real Domains
- Parametric Quantified SAT-Checking
- Connection Between Reduce/Redlog and Other Solvers

From Computer Algebra to Computer Logic

Computer Algebra ($\sim$ 1965)

- extend the idea of algebra to mechanized computations
- use computers to formally manipulate algebraic expressions
- in contrast to numerical calculations
- entirely new idea of *computation*

Computer Logic

- extend the idea of computer algebra to first-order logic
- use computers to formally manipulate formulas (containing algebraic expressions)
- benefit from everything that is already there
- natural framework for **parametric** computer algebra

Reduce and Redlog

- Redlog = Reduce logic system
- component of the computer algebra system Reduce

History

1960	Lisp (McCarthy)
1963	contact and cooperation McCarthy–Hearn
1967	first descriptions and announcements
1968	first distribution “Reduce 1.0”
1970	Reduce 2.0 featuring Rlisp
1983	Reduce 3.0, developer community
1992	First Redlog prototypes
1996	Redlog 1.0 published on the Web
1997	Redlog integrated into Reduce 3.6
1999	Reduce 3.7/Redlog 2.0
2004	Reduce 3.8/Redlog 3.0 last commercial Release
2008	Open Source Reduce at Sourceforge

Online Resources

- Reduce-Homepage at Sourceforge

  `http://reduce-algebra.sourceforge.net/`

- SourceForge Project reduce-algebra:

  `http://sourceforge.net/projects/reduce-algebra/`

- Reduce Wiki

  `http://apps.sourceforge.net/mediawiki/reduce-algebra/`

- Redlog Homepage
  - Documentation as both HTML and for download.
  - Remis = Redlog Example Management and Information System
  - References (generated from the Remis database)

  `http://www.redlog.eu`

Quantifier Elimination—Two Real Examples

Quantifier Elimination

Fix **syntax**: a set of function symbols and relation symbols.

Fix **semantics**: a domain and an interpretation for these symbols.

Given a first-order formula φ find quantifier-free φ' such that $\varphi' \longleftrightarrow \varphi$.

Very Easy Example (syntax $(0, 1, +, -, \cdot, =, \leq, \neq, <)$ / semantics $\mathbb{R}$)

$$\varphi \equiv \exists x(ax^2 + bx + c = 0)$$

$$\updownarrow$$

$$\varphi' \equiv (a \neq 0 \wedge b^2 - 4ac \geq 0) \vee (a = 0 \wedge b \neq 0) \vee (a = 0 \wedge b = 0 \wedge c = 0)$$

Easy Example by Hong (syntax / semantics as above)

$$\varphi \equiv \forall x \exists y(x^2 + xy + b > 0 \wedge x + ay^2 + b \leq 0)$$

$$\updownarrow$$

$$\varphi' \equiv \ldots$$

What is Research in Quantifier Elimination About?

Devise regular quantifier elimination procedures

- Pick an interesting Domain (syntax + semantics)
- Find an efficient quantifier elimination procedure
- Alternatively, prove that this is not possible
- Complexity analysis
- The methods are typically algebraic!

Specify variants of quantifier elimination

- More efficiency at the price of reduced information
- Provide additional information
- Weaken the notion of quantifier elimination to provide something at all

Provide software

- Provide robust implementations and support
- Identify possible fields of applications
- Demonstrate practical applicability

Real Elimination Methods in REDLOG

Partial CAD (Collins 1973, Collins and Hong 1991)

- Doubly exponential in the number of all variables
- Generally applicable
- Reasonably simple results

Virtual Substitution (Weispfenning 1988)

- Doubly exponential in the number of quantifier changes
- Restricted to formulas with low-degree polynomials
- Produces a large number of atomic formulas

Hermitian Quantifier Elimination (Weispfenning 1993)

- Not elementary recursive
- Aims at formulas with many equations
- Produces huge polynomials with huge coefficients

Automatic Combination of Methods

Currently Fallback Quantifier Elimination

- Virtual Substitution as long as possible
- Then partial CAD

Hong's Example

$$\varphi \equiv \forall x \exists y (x^2 + xy + b > 0 \wedge x + ay^2 + b \leq 0) \rightsquigarrow \varphi' \equiv a < 0 \wedge b > 0$$

- First eliminate $\exists y$ by virtual substitution,
  then eliminate $\forall x$ by partial CAD.

  Takes 0.4 seconds altogether.
- Virtual substitution for $\forall x$ fails (degree 4).
- Partial CAD for the entire problem takes 86 seconds.

  Factor > 100.

Long-term goal: Meta Quantifier Elimination.

Virtual Substitution

- Given $\exists x \varphi$, where $\varphi \equiv ax + b = 0$.
- For fixed a, b every such φ describes a finite union of intervals.
- Collect all endpoints of intervals **guarded** by conditions for their existence:

$$E = \{(a \neq 0, -b/a)\}.$$

- Add to the **elimination set** one point with "true" as its guard:

$$E = \{(a \neq 0, -b/a), (\text{true}, 0)\}.$$

- Use modified substitution for the pseudo-terms:

$$\exists x \varphi \longleftrightarrow \bigvee_{(\gamma,t)\in E} \gamma \wedge \varphi[x /\!\!/ t].$$

- The formal result:

$$\left(a \neq 0 \wedge \left(a \cdot \frac{-b}{a} + b\right) \cdot a = 0 \cdot a\right) \vee (\text{true} \wedge a \cdot 0 + b = 0).$$

- Simplify the result: $\varphi' \equiv a \neq 0 \vee b = 0$.

Extended Quantifier Elimination

Generalize $\exists x \varphi \longleftrightarrow \bigvee_{(\gamma,t)\in E} \gamma \wedge \varphi[x /\!\!/ t]$

to the **extended quantifier elimination scheme**

$$\exists x \varphi \rightsquigarrow \begin{bmatrix} \vdots & \vdots \\ \gamma \wedge \varphi[x /\!\!/ t] & \{x = t\} \\ \vdots & \vdots \end{bmatrix}.$$

Semantics

Fix all parameters.
If some left hand side condition holds, then $\exists x \varphi$ holds
and the corresponding right hand side term is **one** sample solution.

In Our Example

$$\begin{bmatrix} a \neq 0 & \{x = -\frac{b}{a}\} \\ b = 0 & \{x = 0\} \end{bmatrix}.$$

Generic Quantifier Elimination

Collect negated equation guards like $a \neq 0$ in a theory Θ:

$$\Theta = \{a \neq 0\}, \quad E = \left\{\left(\text{true}, -b/a\right)\right\}.$$

Elimination result after simplification

$$\varphi' \equiv \text{true}.$$

Output $(\Theta, \varphi') = (\{a \neq 0\}, \text{true})$.

Semantics

$$\bigwedge \Theta \longrightarrow (\varphi' \longleftrightarrow \varphi).$$

- φ' is correct for choices of parameters satisfying Θ.
- Exception set has a lower dimension than the parameter space.

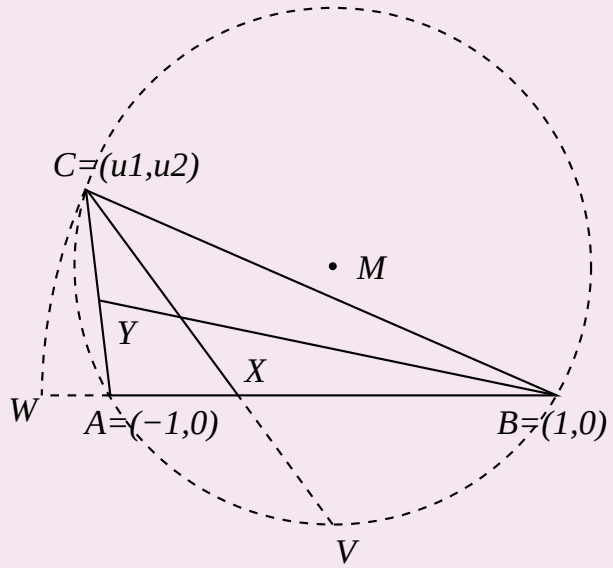
Variant of the Idea: Local Quantifier Elimination

- Add also ordering inequalities to Θ.
- Consistency of Θ by maintaining local consistency at a prescribed point.

Example: Variant of the Steiner–Lehmus-Theorem

The longer bisector goes to the shorter side

$$
\begin{aligned}
h_1 &\equiv u_2 \geq 0 \wedge x_1 \geq 0 \\
h_2 &\equiv r^2 = 1 + x_1^2 = u_1^2 + (u_2 - x_1)^2 \\
h_3 &\equiv x_2 \leq 0 \wedge r^2 = (x_2 - x_1)^2 \\
h_4 &\equiv u_1 x_2 + u_2 x_3 - x_2 x_3 = 0 \\
h_5 &\equiv x_4 \leq 1 \wedge (x_4 - 1)^2 = (u_1 - 1)^2 + u_2^2 \\
h_7 &\equiv (-1 - u_1)^2 + u_2^2 < 2^2 \\
h_6 &\equiv (x_4 - x_5)^2 + x_6^2 = (u_1 - x_5)^2 + (u_2 - x_6)^2 \wedge u_1 x_6 - u_2 x_5 - u_2 + x_6 = 0 \\
g &\equiv (u_1 - x_3)^2 + u_2^2 < (x_5 - 1)^2 + x_6^2
\end{aligned}
$$



- $\varphi \equiv \forall x_6 \forall x_5 \forall x_4 \forall x_3 \forall x_2 \forall x_1 \forall r \left(\bigwedge_{i=1}^{7} h_i \longrightarrow g \right)$
- GQE (1.3 s): φ' 231 atomic formulas, $\Theta = \{\underbrace{u_1^2 - 2u_1 + u_2^2 - 3 \neq 0}_{(u_1-1)^2+u_2^2 \neq 4}, u_1 \neq 0, u_2 \neq 0\}$.
- QEPCAD (< 2 min): $\forall u_1 \forall u_2 (\bigwedge \Theta \longrightarrow \varphi')$ ✓

Hopf Bifurcations in Algebraic Biology (1)

Starting Point

Boulier, Lefranc, Lemaire, Morant, Ürgüplü:
On proving the absence of oscillations in models of genetic circuits.
Algebraic Biology 2007. Springer LNCS 4545.

- The working group "models the gene regulatory network controlling the circadian clock of a unicellular green alga."
- One main problem (theoretically very difficult):
  - Given a system of parametric ODE built using mass action law kinetics.
  - Are there biologically meaningful ranges of values for the parameters and variables, where oscillating trajectories (limit cycles) can be found?
- Related but easier Problem:
  - Are there biologically meaningful ranges of values for the parameters and variables that give rise to a **Hopf bifurcation**?

A Picture Given by Boullier at al.

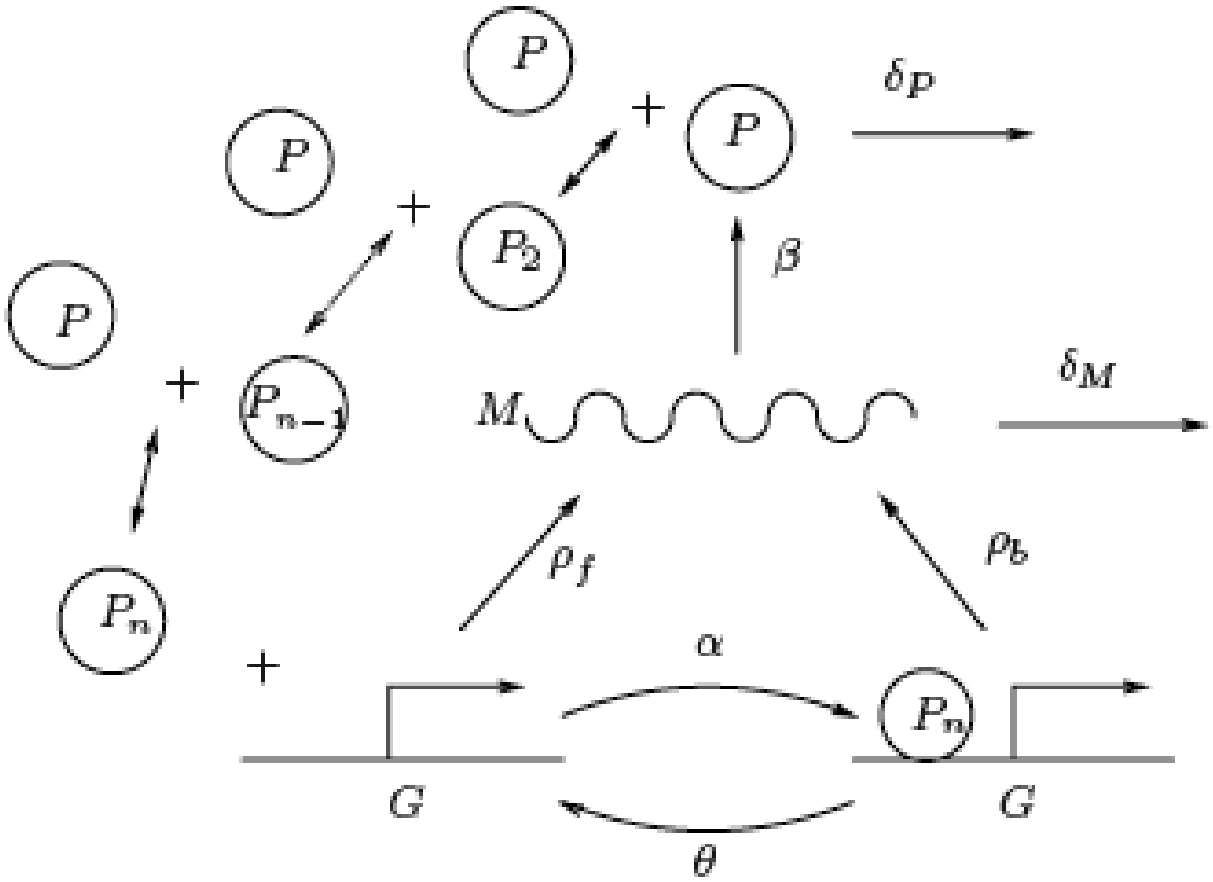


Fig. 1. A gene regulated by a polymer of its protein

Hopf Bifurcations in Algebraic Biology (2)

- Quasi-steady state approximation delivers a reduced system:

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}G(t) &= \vartheta \cdot \left(\gamma_0 - G(t) - G(t)P(t)^n\right) \\
\frac{\mathrm{d}}{\mathrm{d}t}P(t) &= n\alpha \cdot \left(\gamma_0 - G(t) - G(t)P(t)^n\right) + \delta \cdot \left(M(t) - P(t)\right) \\
\frac{\mathrm{d}}{\mathrm{d}t}M(t) &= \lambda_1 G(t) + \gamma_0\mu - M(t)
\end{aligned}$$

- Boulier et al. examine the existence of Hopf bifurcations for $n = 1, \ldots, 9$.
- Supported by Maple but considerable human intelligence and interaction.
- Fully automatic numerical methods can prove the presence but not the absence of Hopf bifurcations.

Our Fully Automatic Symbolic Approach

- Using results and an implementation by El Kahoui and Weber, generate for fixed n a first-order formula φ_n over the reals.

Example ($n = 9$)

$$
\begin{aligned}
\varphi_9 \quad = \quad & \exists \mathbf{v}_2 \exists \mathbf{v}_1 \exists \mathbf{v}_3 (0 < \mathbf{v}_1 \wedge 0 < \mathbf{v}_3 \wedge 0 < \mathbf{v}_2 \wedge \vartheta\gamma_0 - \mathbf{v}_1 - \mathbf{v}_1 {\mathbf{v}_3}^9 = 0 \wedge \\
& \lambda_1 \mathbf{v}_1 + \gamma_0 \mu - \mathbf{v}_2 = 0 \wedge 9\alpha\gamma_0 - \mathbf{v}_1 - \mathbf{v}_1 {\mathbf{v}_3}^9 + \delta \mathbf{v}_2 - \mathbf{v}_3 = 0 \wedge \\
& 0 < \vartheta\delta + \vartheta {\mathbf{v}_3}^9 \delta + 9\lambda_1 \vartheta \mathbf{v}_1 {\mathbf{v}_3}^8 \delta \wedge \\
& 162\vartheta {\mathbf{v}_3}^{17} \alpha \mathbf{v}_1 + 162\vartheta\alpha \mathbf{v}_1 {\mathbf{v}_3}^8 + 162\alpha \mathbf{v}_1 {\mathbf{v}_3}^8 \delta + \vartheta + 2\vartheta {\mathbf{v}_3}^9 \delta + \vartheta^2 {\mathbf{v}_3}^{18} \delta \\
& \quad + \vartheta {\mathbf{v}_3}^9 \vartheta\delta + 81\alpha \mathbf{v}_1 {\mathbf{v}_3}^8 \vartheta\delta + 81\alpha \mathbf{v}_1 {\mathbf{v}_3}^{17} \vartheta\delta + \delta^2 + \vartheta\delta^2 + \vartheta^2\delta + \vartheta^2 \\
& \quad + 2\vartheta^2 {\mathbf{v}_3}^9 + \vartheta^2 {\mathbf{v}_3}^{18} + 6561\alpha^2 {\mathbf{v}_1}^2 {\mathbf{v}_3}^{16} + 2\vartheta^2 {\mathbf{v}_3}^9 \delta + \delta + 81\alpha \mathbf{v}_1 {\mathbf{v}_3}^8 \\
& \quad + \vartheta {\mathbf{v}_3}^9 \delta^2 - 9\lambda_1 \vartheta \mathbf{v}_1 {\mathbf{v}_3}^8 \delta = 0 \wedge \\
& 0 < \vartheta \wedge 0 < \gamma_0 \wedge 0 < \mu \wedge 0 < \delta \wedge 0 < \alpha)
\end{aligned}
$$

- φ_n holds for some choice of parameters iff a Hopf bifurcation exists.
- $\exists \varphi_n$ holds iff there is such a choice of parameters at all (easier!)

Positive Real Quantifier Elimination

Thomas's Recipe for Efficient QE

$$\begin{aligned}\text{good QE} \quad &= \quad \text{virtual substitution} \\ &+ \text{ sophisticated heuristic simplification} \\ &+ \text{ efficient fallback methods for degree violations}\end{aligned}$$

Observation and Idea

- In many natural problems, all variables may be assumed to be positive.
- In others, this can possibly be achieved by a linear translation.
- Globally making this assumption greatly supports quantifier elimination.

- One typical simplification for atomic formulas over the reals:

$$x^2y^4 + 4x^2 + y^2 \geq 0 \quad \rightsquigarrow \quad \text{true} \qquad (\text{“trivial squaresum”})$$

- Virtual substitution of fractions:

$$(x + y > 0)\left[x \leftarrow \frac{a}{b}\right] \quad \Longleftrightarrow \quad b \neq 0 \wedge abx + b^2y > 0$$

Some Results

- Regular quantifier elimination mostly fails so far.
- Positive QE is successful not on φ_n but on $\underline{\exists}\varphi_n$ ($n = 2, \ldots, 10$).

n	$\underline{\exists}\varphi_n$	$\underline{\exists}\varphi_n[\lambda_1 \leftarrow -\lambda_1]$	$\underline{\exists}\varphi_n[\lambda_1 \leftarrow 0]$	time (s)	$\underline{\exists}\varphi[\lambda_1 \leftarrow \lambda_1 - \lambda_1']$	time (s)
2	false	false	false	< 0.01	SIGXCPU	> 600
3	false	false	false	19.28	SIGXCPU	> 600
4	false	false	false	21.58	SIGXCPU	> 600
5	false	false	false	19.09	SIGXCPU	> 600
6	false	false	false	23.72	SIGXCPU	> 600
7	false	false	false	23.89	SIGXCPU	> 600
8	false	false	false	22.35	SIGXCPU	> 600
9	true	false	false	0.17	true	69.10
10	true	false	false	0.17	true	69.19

- **Extended QE** delivers in addition approximate sample solutions:

Example ($n = 9$)

$\alpha = 1$ $\quad \delta = 1$ $\quad \gamma_0 = 0.0100554964908$

$\lambda_1 = 17617230.5528$ $\quad \mu = 0$ $\quad \vartheta = 0.000021144360845 5$

$\mathbf{v}_1 = 0.0000001702878321 89$ $\quad \mathbf{v}_2 = 3$ $\quad \mathbf{v}_3 = 1.24573093962.$

Successful Real Applications of REDLOG

- parametric and nonlinear optimization
- transportation problems
- circuit analysis, -design, -diagnosis
- generalized scheduling problems
- real implicitation
- automated theorem proving
- computational geometry
- solid modeling
- robot motion planning
- algebraic biology
- factorization of LPDOs
- automatic loop parallelization (Lengauer)
- bifurcation analysis (El Kahoui, Weber)
- theoretical mechanics (Ioakimidis)
- stability of differential equations (Hong, Liska, Steinberg)
- hybrid control theory (Yovine, Anai/FUJITSU)
- atmosphere chemistry (Lustfeld)
- hydraulic network diagnosis (ROSE)
- runtime properties of programs (Anderson et al.)
- reasoning in complex theories (Sofronie-Stokkermans)

Many More Domains and Applications

Reals (JSC 97, JAR 98, AAECC 99, CASC 00, ISSAC 97/00/03/04, . . .)

- discussed before

Complex

- language of rings only

Differential (CASC 2004)

- language of rings with unary differential operator
- computation in differentially closed field (A. Robinson, Blum)

Padics (JSC 00, ISSAC 99, CASC 01)

- linear formulas over p-adic fields for p prime
- optionally uniform in p
- used e.g. for solving parametric systems of congruences over the integers

Yet More Domains and Applications

Terms (CASC 2002)

- Malcev-type term algebras (with functions instead of relations)

Queues (C. Straßer at RWCA 2006)

- two-sided queues over the other domains (2-sorted)
- Implemented at present for queues of reals

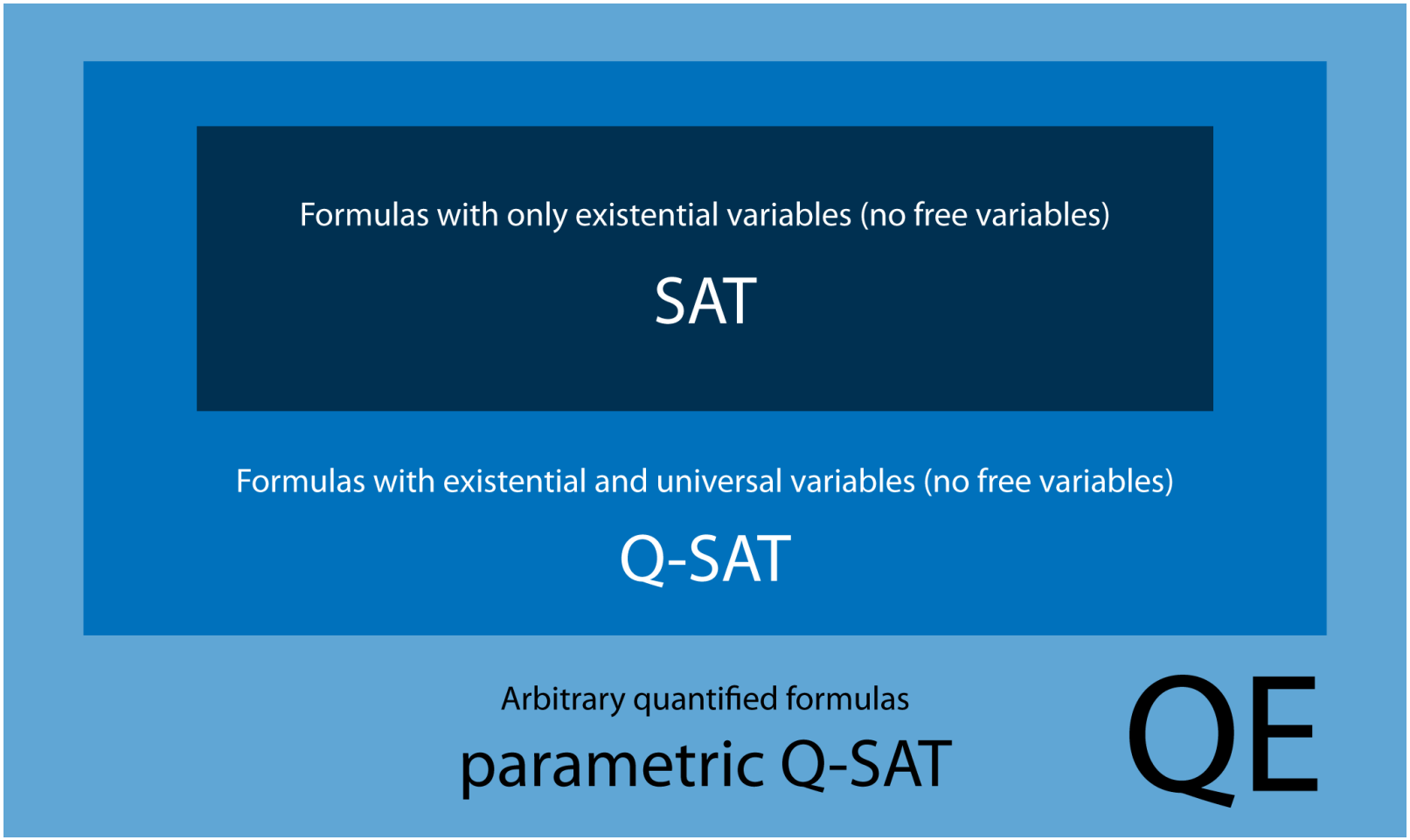
Integers (AAECC 2007, CASC 2007, Sturm/Lasaruk 2009)

- Some details soon . . .

Boolean (CASC 2003, Sturm/Zengler 2009)

- generalization of SAT-checking
- quantified propositional calculus (parametric QSAT-checking)

Parametric Quantified SAT-Checking



Boolean Domain in Redlog (CASC 2003)

- Consider theory of initial Boolean algebras
- Normal forms of first-order formulas with atoms $x = 1$:

$$\forall x \forall y (x = \sim y \longrightarrow \forall a (a \,\&\, y = a \,\&\, \sim x)$$
$$\rightsquigarrow \quad \forall x \forall y ((x = 1 \longleftrightarrow \neg y = 1) \longrightarrow \forall a (a = 1 \wedge y = 1 \longleftrightarrow a = 1 \wedge \neg x = 1)$$

- Propositional wrapper presents formulas as quantified propositional calculus:

$$\rightsquigarrow \quad \forall x \forall y ((X \longleftrightarrow \neg Y) \longrightarrow \forall a (A \wedge Y \longleftrightarrow A \wedge \neg X)$$

- Naive quantifier elimination by virtual substitution:

$$\exists x \varphi \longleftrightarrow \bigvee_{t \in \{0,1\}} \varphi[t/x]$$

- For the special case of (Q)SAT problems considerably less efficient than corresponding solvers.

Boolean Domain in Redlog (Zengler + S. 2009)

Implementation of a SAT-solver with state-of-the-art techniques:

- Preprocessing of the input formula
- Different selection heuristics (DLCS, DLIS, MOM, ZMOM, Activity)
- Conflict driven clause learning
- Unit propagation with watched literals
- Deletion of learned clauses with low activity
- Restart after a certain number of learned clauses
- Completely configurable

Key Idea for PQSAT Checking

- Apply QSAT solver for recursively assigned parameters
- The clauses learned during QSAT remain valid during recursion

The PQSAT Algorithm

Input: An arbitrary formula φ and the assignment α
Output: A formula in the free variables of φ

```
1  if fvars(φ) \ dom(α) = ∅ then
2      (φ, q) := QSAT(φ);
3      if q = true then
4          return form(α)
5      else
6          return "false"
7  else
8      x := choose a variable of fvars(φ) \ dom(α);
9      α' := α ∪ {x ← ⊤};
10     ψ1 = false;
11     if no conflict is reached after unit propagation then
12         (φ, ψ1) = PQSAT(φ, α');
13     α'' := α ∪ {x ← ⊥};
14     ψ2 = false;
15     if no conflict is reached after unit propagation then
16         (φ, ψ2) = PQSAT(φ, α'');
17     return ψ1 ∨ ψ2 after some straightforward simplification
```

From an Interactive Tool to a Library

- There are many successful applications of REDLOG in the literature
  (a) Interactive applications
  (b) Applications, where REDLOG performs as pre/postprocessor

Goal

- Use REDLOG as a library from other applications
- Frequently asked in particular by users from computer science

But

- Linking C-like code (or Java) with Lisp is a well-known problem.

Recent Project

- `libreduce.a`, which can be linked to C.
- Prototype exists (works with SPASS prover of MPI Saarbrücken).
- Extension to C++ etc. is straightforward.
- Extension to Java is possible.

Summary

- Redlog and Reduce are freely available on the Web
- Real quantifier elimination and variants, applications
- Many more domains
- Parametric quantified SAT-Checking
- Possible issues with and perspectives for connectivity