Quantifier Elimination

Paul Feautrier

Paul.Feautrier@ens-lyon.fr.

ENS Lyon

Plan

- Definition
- References
- Basic Techniques
- Test Points
- Virtual Substitution
- Complexity
- Perspectives

Definition

Consider a logical formula $\exists x \in \mathbb{R} : \Phi(x)$,

(1) $$\Phi(x) = \mathcal{B}(P_1(x)\,\rho_1 0, \ldots, P_n(x)\,\rho_n 0)$$

where :

- $\mathcal{B}$ is a boolean combinator,
- The P_i are polynomials in x and other variables (the parameters).
- The ρ_i are one of the comparators $<, \leq, =, \neq, \geq, >$.

Find a quantifier-free formula Φ' such that:

(2) $$\exists x \in \mathbb{R} : \Phi(x) \equiv \Phi'.$$

- Generalization:
  - The logical formula may be arbitrary, with arbitrarily nested existential and universal quantifi ers.
  - The only restriction is that the literals are of the form $P(x)\,\rho\,0$ where P is a polynomial.

Normalization

- Universal quantifiers can be eliminated by the rule $\forall x : \phi \equiv \neg \exists x : \neg \phi$.
- Others boolean connectives can be replaced by their definitions in terms of $\wedge, \vee, \neg$.
- The negation can be "innermosted" with de Morgan's laws: $\neg (p \wedge q) \equiv (\neg p) \vee (\neg q)$, etc.
- Innermost $\neg$ can be eliminated by rules such as $\neg (P > 0) \equiv P \leq 0$.
- We can eliminate quantifiers one by one starting from an innermost one.
- We can suppose that the combinator $\mathcal{B}$ uses only the $\vee$ and $\wedge$ connectives.

References

- Alfred Tarski. A decision method for elementary algebra and geometry. Technical Report, RAND, Santa Monica, CA.
- Volker Weisspfenning. The complexity of linear problems in fields. J. of Symb. Computation, 5(1-2):3–27, 1988.
- G. E. Collins. Quantifier Elimination for the Elementary Theory of Real Closed Field by Cylindrical Algebraic Elimination. In H. Brackage (ed) LNCS 33.
- Jeanne Ferrante and Charles W. Rackoff. The Computational Complexity of Logical Theories. LNCS 718.
- Andreas Dolzmann. Algorithmic Strategies for Applicable Real Quantifier Elimination. PhD. Thesis, University of Passau, March 2000.

Why is Quantifier Elimination Possible?

- The sign of a litteral changes only when x cross a root of the associated polynomial.
- Since a polynomial has only a finite number of roots, one can find a finite set of intervals in which the value of Φ is constant.
- If one is able to find a test point in each interval, then one can take $\Phi' = \vee_{i=1}^{p} \Phi(t_i)$ where the t_i are the test points.
- Since in general one has to reinsert Φ' in the original formula and eliminate another quantifier, minimizing the number of test points is crucial for the complexity of the method.

The linear case

- Here I will consider only the linear case, in which all the polynomials are first degree. The reason is that finding the roots of a first degree polynomial is easy, while the task becomes more and more difficult as the degree increases.
- However, the coefficients of the linear forms in Φ may involve arbitrary parameters. In such a case, the formula one obtain after eliminating all quantifiers is a predicate on the value of parameters.
- Solving the resulting constraints for appropriate values of the parameters may be difficult, but is not the subject of this talk.

Why is Quantifier Elimination Useful?

- Quantifier Elimination is very similar to the Fourier-Motzkin algorithm, but the class of tractable problems is much wider.
- While Fourier-Motzkin works only if the litterals are linear forms, QE accepts forms which are linear only in the bound variables.
- While F-M accepts only a conjunction of litterals, QE accepts an arbitrary boolean formula.
- While F-M accepts only existential quantifiers, QE accepts any mixture of existential and universal quantifiers.

Precursors

- Linear programming consists in testing a formula:

  (3) $$\exists \boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} \geq \boldsymbol{0}.$$

  where $\boldsymbol{x}$ is the vector of existentially quantifi ed variable, $\boldsymbol{A}$ is the matrix of constraints, and $\boldsymbol{b}$ is the constant term. The Simplex algorithm is much faster than quantifi er elimination, but less general.

- The Fourier-Motzkin algorithm is clearly a special case of the test point method, see later.

- Suppose that the $\boldsymbol{b}$'s are considered as parameters. One obtains Parametric Linear Programming, which can be solved by a symbolic variant of the simplex.

- In scheduling, one fi nd formulas of the form:

  (4) $$\exists \boldsymbol{c}, \boldsymbol{d} : \forall \boldsymbol{x}(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} \geq \boldsymbol{0} \Rightarrow \boldsymbol{c}\boldsymbol{x} + \boldsymbol{d} \geq \boldsymbol{0}).$$

  While these are not linear in the above sense, they can be transformed into linear programming problems by the use of Farkas lemma or by the vertex method. The vertex method is clearly a test point method.

An important simplification

- If two consecutive quantifiers are of the same kind, as in:

$$\exists x, y : \Phi(x, y).$$

one first eliminate y:

(5) $$\exists x : \vee_{i=1}^{p} \Phi(x, t_i(x)).$$

- One can distribute the quantifier:

$$\vee_{i=1}^{p} \exists x : \phi(x, t_i(x)).$$

- One can then handle the remaining quantifiers independently. The algorithm becomes singly exponential instead of doubly exponential.

Selection of the tests points, I

- Let $a_k x + b_k \; \rho_k \; 0$ be one of the literals.
- Its truth value change only when $a_k x + b_k = 0$, i.e. for $x = -b_k / a_k = t_k$.
- This critical point delimits two segments in which the truth value of Φ does not change. However, depending on the sign of a_k and on the nature of the comparator ρ_k, t_k belongs either to the left segment or to the right segment.
- It may happen that in some cases there is a segment without test point.
- To decide, one needs to know the sign of a_k, which may not be possible if a_k depends on parameters.

Selection of the test points, II

- To be sure that there is a test point in each segment, one possibility is to use $\frac{t_k+t_{k+1}}{2}$. But this is correct only if one can sort the t_k in ascending order.
- Another possibility is to use $\frac{t_k+t_\ell}{2}$ for all pairs k, ℓ. The complexity becomes enormous.
- A last possibility is to use $t_k \pm \epsilon$, where ϵ is some infinitesimal. One is sure to have a test a point in each segment adjacent to t_k, and the complexity is acceptable.

The case of equations

- Suppose one of the literals is an equation $a_k x + b_k = 0$.
- The corresponding critical point is $t_k = -b_k/a_k$ and we may think to use $t_k \pm \epsilon$ as test points. But for these two values the equation is false!
- It seems that one has to use all three of $t_k, t_k \pm \epsilon$ as test points.
- Beside, one has to do the same for all critical points, because Φ may have hidden equations!

Gaussian Elimination

- There is a special case:

(6) $$\exists x : (ax + b = 0) \wedge \Phi(x) \quad \equiv \quad (a \neq 0 \wedge \Phi(-b/a))$$

(7) $$\vee\ (a = 0 \wedge b = 0 \wedge \exists x \Phi(x)).$$

Selection of the tests points, III

- The problem gets more complicated if a_k may be 0. In that case the test point disappears to infinity.
- It is nevertheless necessary to test the value of the literal, for instance by using some arbitrary test point, e.g. 0.
- One solution is to write the test point as a conditional:

(8) $$t_k = \text{if } a_k \neq 0 \text{ then } -b_k/a_k \pm \epsilon \text{ else } 0.$$

- Another possibility : guarded test points:

(9) $$\{(a_k \neq 0, -b_k/a_k \pm \epsilon), (\text{true}, 0)\}.$$

- One has to extend the concept of substitution to terms of this form.

Virtual Substitution

- What does $(ax + b \,\rho\, 0)[x \to t + \epsilon]$ means?

(10) $$(ax + b)[x \to t + \epsilon] \equiv$$

(11) $$\equiv a(t + \epsilon) + b \,\rho\, 0$$

(12) $$\equiv \textbf{if } at + b = 0 \textbf{ then } a \,\rho\, 0 \textbf{ else } at + b \,\rho\, 0.$$

- What does $\Phi(\textbf{if } p \textbf{ then } a \textbf{ else } b)$ means?

(13) $$\Phi(\textbf{if } p \textbf{ then } a \textbf{ else } b) \equiv \textbf{if } p \textbf{ then } \Phi(a) \textbf{ else } \Phi(b).$$

- This is “virtual substitution”.

An example

- Eliminate the quantifi er in $\exists x : ax + b \geq 0$. $\Phi(x) \equiv ax + b \geq 0$.
- Critical points: $-b/a$, provided $a \neq 0$ and 0.
- Test points, with guards:

$$(a \neq 0, t_1 = -b/a), \qquad (a \neq 0, t_2 = -b/a + \epsilon), \tag{14}$$

$$(a \neq 0, t_3 = -b/a - \epsilon), \qquad (\text{true}, t_4 = 0). \tag{15}$$

$$\Phi' \equiv (a \neq 0) \wedge (\Phi(t_1) \vee \Phi(t_2) \vee \Phi(t_3)) \vee (\text{true} \wedge \Phi(0)) \tag{16}$$

$$\equiv (a \neq 0) \wedge (\text{true} \vee a > 0 \vee a < 0) \vee (b \geq 0) \tag{17}$$

$$\equiv a \neq 0 \vee b \geq 0. \tag{18}$$

Many Quantifiers

- Quantifier Elimination proceeds from inside outward.
- Observe that the result of virtual substitution is linear in b but not in a.
- The original system must be affine in all the bound variables simultaneously.

QE and Optimization

- As soon as one has a parametric feasibility test, one can solve optimization problems.
- Suppose for instance that the problem is: $\exists x_1, \ldots, x_n : \Phi(x_1, \ldots, x_n)$ and that, among all solutions in $x_1, \ldots, x_n$, one wants the one which minimize some function $f(x)$.
- Consider the problem : $\exists x_1, \ldots, x_n : \Phi(x_1, \ldots, x_n) \wedge z \geq f(x)$, where z is a fresh parameter.
- If a minimum f_0 exists, then $z \geq f_0$ will appear in the feasibility conditions.

Extracting Solutions

- Almost always, one wants to know that the original problem is feasible, but also the coordinates of some feasible point.
- This information can be extracted from the lists of test points.
- There are problems for interpreting guards and infinitesimals.

Complexity

- Let $T(m,n)$ be the "complexity" of the elimination of m **existential** quantifiers before a formula with n literals.
- One builds about $3n$ test points which have to be substituted into n literals, for a work of order $3n^2$.
- Because of redundency, this gives about $2n$ independent subproblems with n literals and $m-1$ quantifiers.
- Hence the recurrence:

$$T(m,n) = 2nT(m-1,n) + 3n^2. \tag{19}$$

- Hence the complexity is $O(2^m n^{m+2})$.

Comparisons

- In the totaly linear case, the complexity is better than Fourier-Motzkin: $O(\frac{n}{2}^{2^m})$ ...
- ... but far worse that the Simplex ($O((m+n)n^2)$ in the mean).
- The range of application of QE is much wider than Fourier-Motzkin or the Simplex ...
- ... but much attention is needed in the implementation.

Perspectives

- There is an implementation from the University of Passau, which seems to have difficulties (probably due to too much attention given to quadratic problems). The implementation is based on the computer algebra system REDUCE.
- Reimplement?
  - Detect totally linear subproblems, which can be solved more effi ciently by Linear Programming.
  - Use outside information (context) to simplify guards and test points.
  - Use test point redundency to omit the more complex ones.
- Extension to integral problems? No implementation. Some theoretical results by Weisspfenning, indicating that the complexity increases enormously.