Effective Set Membership in Computer Algebra

James H. Davenport
Department of Computer Science
University of Bath
Bath BA2 7AY England
J.H.Davenport@bath.ac.uk

July 30, 2008

Set Membership

$$S := \{x \in A \mid P(x)\}$$

where A is a set for which membership is "obvious", e.g. by construction, and P is some predicate, which will generally involve some existential quantifiers.

Effective Set Membership

Given some $x \in A$, produce

either an *effective* proof of $P(x)$

or a proof of $\neg P(x)$.

Effective Set Membership

Given some $x \in A$, produce

either an *effective* proof of $P(x)$

or a proof of $\neg P(x)$.

In general, it is the second part of the problem that is the hard one, at least for "natural" P.

Not solve the problem

(Discuss how it's expressed)

N **N** **P**

N

N

Not solve the problem
(Discuss how it's expressed)

$$S := \{k \in \mathbf{N}_{\text{odd}} \mid \exists n \in \mathbf{N}, p \in \mathbf{P} \quad k = 2^n - p\}\,.$$

We do not know if this is $\mathbf{N}_{\text{odd}}$ or not, merely that any element of $\mathbf{N}_{\text{odd}} \setminus S$ is greater than 219.

Not solve the problem
(Discuss how it's expressed)

$$S := \{k \in \mathbf{N}_{\text{odd}} \mid \exists n \in \mathbf{N}, p \in \mathbf{P} \quad k = 2^n - p\}\,.$$

We do not know if this is $\mathbf{N}_{\text{odd}}$ or not, merely that any element of $\mathbf{N}_{\text{odd}} \setminus S$ is greater than 219.

Note that even the humble 7 has, as its least representation in S,

$$7 = 2^{39} - 549755813881.$$

Ideal Membership

Ideal Membership

$$(p_1, \ldots, p_m) = \left\{ \sum_{i=1}^{m} f_i p_i : f_i \in k[x_1, \ldots, x_n] \right\}$$

"ideal membership" = "find the f_i"

Ideal Membership

$$(p_1, \ldots, p_m) = \left\{ \sum_{i=1}^{m} f_i p_i : f_i \in k[x_1, \ldots, x_n] \right\}$$

"ideal membership" = "find the f_i"

Either find the f_i, e.g. by repeated reduction.

Or ??

Ideal Membership

$$(p_1, \ldots, p_m) = \left\{ \sum_{i=1}^{m} f_i p_i : f_i \in k[x_1, \ldots, x_n] \right\}$$

"ideal membership" = "find the f_i"

Either find the f_i, e.g. by repeated reduction.

Or ??

Buchberger If the p_i are Gröbner, reduction to non-zero *implies* non-membership.

Constructivity?

- Testing for a Gröbner basis is constructive.

- Also, we can compute Gröbner bases.

  (Essentially a pre-conditioning)

Hence **or** consists of

Hence **or** consists of

1. A proof that $(p_1, \ldots, p_m)$ *is* Gröebner

Hence **or** consists of

1. A proof that $(p_1, \ldots, p_m)$ *is* Gröebner
2. A reduction of f to an irreducible non-zero.

Hence **or** consists of

1. A proof that $(p_1, \ldots, p_m)$ *is* Gröebner
2. A reduction of f to an irreducible non-zero.
3. (A proof that the p_i correspond to the original question q_i)

Hence **or** consists of

1. A proof that $(p_1, \ldots, p_m)$ *is* Gröebner
2. A reduction of f to an irreducible non-zero.
3. (A proof that the p_i correspond to the original question q_i)

1, 2 probably exist in the computation;

Hence **or** consists of

1. A proof that $(p_1, \ldots, p_m)$ *is* Gröebner
2. A reduction of f to an irreducible non-zero.
3. (A proof that the p_i correspond to the original question q_i)

1, 2 probably exist in the computation;

3 is implicit in "my GB algorithm is correct", and would probably need to be re-proved ($q_j \rightarrow^*_{(p_i)} 0$ suffices).

$\mathcal{I}$-Integration in 1964

Problem 1 *For given* $f \in \mathcal{I}$

either *exhibit* $g \in \mathcal{I}$ *such that* $f = g'$

or *return* `failed` *(*g *might exist, but hadn't been found),*

and a successful program was one which did not return `failed` when a freshman could see the answer.

$\mathcal{I}$-Integration in 1970 (Risch, Moses etc.)

Problem 2 *For given* f (normally $f \in \mathcal{I}$)

either *exhibit* $g \in \mathcal{I}$ *such that* $f = g'$

or *demonstrate that no such* g *exists.*

This is generally implemented for $\mathcal{I}$ elementary transcendental (modulo the constant problem), but the **or** generally has to be taken on trust.

Liouville's Theorem (1835)

Risch's Algorithm (1969)

Liouville's Theorem (1835)
Risch's Algorithm (1969)

If an elementary integral exists, then the original function must be of a certain form:

$$f = v_0' + \sum_{i=1}^{n} c_i \frac{v_i'}{v_i},$$

with $v_0 \in K$, $c_i \in C = \overline{\{g \in K \mid g' = 0\}}$, $v_i \in CK$

Hard to explain to the user

Hard to explain to the user

For example Maple 11 says merely

> If Maple cannot find a closed form expression for the integral, the function call is returned.

Hard to explain to the user

For example Maple 11 says merely

> If Maple cannot find a closed form expression for the integral, the function call is returned.

This is complicated by the "greedy salesman problem" — the salesman wants the most powerful integrator, not the best-defined integrator.

Implications for CA systems as oracles.

Implications for CA systems as oracles.

Either generally pretty good, but could do better.

Implications for CA systems as oracles.

Either generally pretty good, but could do better.

Or (trust me) Mixed: GB exposes the algorithmicity; integration etc. generally doesn't.

Implications for CA systems as oracles.

Either generally pretty good, but could do better.

Or (trust me) Mixed: GB exposes the algorithmicity; integration etc. generally doesn't.

Or (and here's the proof) Pretty poor.

Calculemus

Calculemus

et demonstrationes monstremus

Calculemus

et demonstrationes monstremus

(Let us calculate/prove

and show the proofs)