Several Interesting Problems from Applied Logic

Grigori Mints
Stanford University/SRI
Stanford Logic Seminar
January 29, 2013

Problems discussed here arise in the context of "safe programming" when a program is required to be verified by a complete proof that it satisfies specifications.

Contents

Saving Propositional Formulas

Assume that a resolution proof P of a formula F is obtained but is too big to be saved.
Problem. What kind of information can be saved so that a complete proof can be reconstructed sufficiently fast?

Saving Propositional Formulas

Assume that a resolution proof P of a formula F is obtained but is too big to be saved.
Problem. What kind of information can be saved so that a complete proof can be reconstructed sufficiently fast? Resolution rule:

$$\frac{D \vee l \quad E \vee \neg L}{D \vee E}$$

Completeness. $C_1, \ldots, C_n \models L \vee M \vee N$ iff
$C_1, \ldots, C_n, \neg L, \neg M, \neg N \vdash \emptyset$

Possible solutions.
1. Save all unary clauses from given proof P *in the order of their generation*:

$$L_1, L_2, \ldots, L_m, \qquad m \leq \text{ the number of variables.}$$

It seems natural to expect that repeating the proof-search process with the goal L_1 would terminate faster than unguided search due to unit clause strategies, and the same for $L_2, \ldots, L_m$.
It may be interesting to check this expectation experimentally and if it is fulfilled often enough, try to find for which strategies of resolution this can be proved.

2. The same for binary clauses

$$L_1 \vee M_1, L_2 \vee M_2, \ldots, L_m \vee M_m. \tag{1}$$

Note that a goal $L \vee M$ corresponds to adding two unary clauses $\neg L, \neg M$ to original set of clauses, which may accellerate proof search process even more than in the first case.
Similar problems arise with Davis-Putnam proof procedure (splitting truth-assignment process), proofs with(out) cut, etc.

Exponential Lower Bounds for Proofs in QBF with Cut

If provability problem for some class of formulas in some system is not in NP, then "there are no short proofs".
Relatively recent results of this kind by P.Hrubeŝ
A lower bound fir intuitionistic logic, Ann. Pure Appl. Logic, 146, no. 1, 72-90, 2007
are established for many non-classical (modal and intuitionistic) propositional PSPACE-complete systems including S4 and intuitionistic propositional logic.
However for the standard PSPACE-complete system, namely QBF (Quantified Boolean Formulae) it is not known whether there exists a sequence

$$F_1, \ldots, F_n, \ldots$$

of QBF, polynomial p and a constant c such that for every deririvation d_n of F_n (even with cut)

$$|F_n| < p(n), \ |d_n| > exp(cn)$$

My guess is that this should be true even when comprehension rule in QBF is unrestricted.

Solving Logical Equations in Arithmetic

A lot of efforts in the work related to verification of programs is devoted to finding predicates satisfying certain conditions, that is solving logical equations.

Example 1. (from a paper by N.Bjorner, M. McMillan and A. Rybalchenko). Finding intermediate conditions for sequential composition of programs.

Corresponding proof rule in Hoare logic:

$$\frac{\{P\}s_1\{X\} \quad \{X\}s_2\{Q\}}{\{P\}\ s_1; s_2\ \{Q\}}$$

Suppose that our program has one variable x. Then verification condition can be written in this form:

$$\forall xx'(P(x)\&[s_1](x,x') \Rightarrow X(x')\ ),$$

$$\forall xx'(X(x)\&[s_2](x,x') \Rightarrow Q(x')\ )$$

The authors give no details in that paper, but in this case general theory (known in propositional case to Schröder and Löwenheim) provides a *general* solution.

$$\forall xx'(P(x)\&[s_1](x,x') \Rightarrow X(x')\ ),$$
$$\forall xx'(X(x)\&[s_2](x,x') \Rightarrow Q(x')\ )$$

After an equivalent transformation:

$$\exists y(P(y)\&[s_1](y,x)) \Rightarrow X(x), \qquad P'(x) \to X(x) \qquad (2)$$
$$X(x) \Rightarrow\ \forall y([s_2](x,y) \Rightarrow Q(y)) \qquad X(x) \to Q'(x) \qquad (3)$$

The general solution: $X = \ P' \vee (Q'\&Z)$, more precisely

$$Xx = \ P'(x) \vee (Q'(x)\&Z(x)) \qquad (4)$$

where Z is an arbitrary predicate. In other words, for arbitrary predicate Z the r.h.s. of (4) satisfies (2),(3),
provided $\forall x(P'(x) \to Q'(x))$, and
for every predicate X satisfying (2),(3) there is a Z such that (4), namely $Z = X$.

Loop Invariants

Wikipedia:

... in Floyd-Hoare logic,[1][2] the partial correctness of a **while** loop is governed by the following rule of inference:

$$\frac{\{C \quad X\} \text{ body } \{X\}}{\{X\} \text{ while } C \text{ body } \{\neg C \ \& X\}}$$

The literature on verification contains devices for transforming programs containing loops into systems of (logical) equations for loop invariants.

Often each equation is a Horn clause, like in PROLOG. So in general all computable functions can be encoded, even if there is just one function symbol S, or even without function symbols.

Attempts of a general approach to solving such systems should be based on finding tractable classes.

We are looking for *modular* approaches when a system is naturally partitioned by dependence graph of predicates into blocks and it is possible to find a general solution for each block.

For example, recursions are enclosed inside blocks, and only linear

Example 2. $h(d, e) \to h(d-1, c)$.
for unknown predicate h and variables c, d, e for natural numbers.
Since all variables are implicitly universally quantified the implication is equivalent to

$$h(d, e) \to \forall c h(d-1, c)$$

After some computations, the general solution of this equation for the unknowm predicate h is

$$h(d, e) = \ (\forall x \leq d) \forall c H(x, c) \ \& H(d, e)$$

where H is an arbitrary predicate.

Property-Directed Reachability

[Bradley,Manna,Bjorner]

Definition (Boolean Transition System)

1. A state description: boolean variables $\bar{x} = \{x_1, \ldots, x_n\}$,
2. An initial condition: a propositional fordmula $\theta(\bar{x})$,
3. A transition relation: a propositional formula $\rho(\bar{x}, \bar{x}')$.

A state s: an assignment of the variables $\bar{x}$

Inductive Strengthening

A formula is invariant if it is true in every reachable state.
A formula $\varphi(\bar{x})$ is inductive if
$\theta(\bar{x}) \rightarrow \varphi(\bar{x})$ and
$\varphi(\bar{x}) \wedge \rho(\bar{x}, \bar{x}') \rightarrow \varphi(\bar{x}')$. For a specification formula $\Pi(\bar{x})$, how to prove it is invariant?
Find a strengthening assertion χ such that $\Pi \wedge \chi$ is inductive.

A Solution

Let's consider only inductive step:

$$\Pi(\bar{x}) \wedge \chi(bx) \wedge \rho(\bar{x}, \bar{x}') \rightarrow \ \Pi(\bar{x}') \wedge \chi(\bar{x}')$$

Splitting the last $\wedge$ get a system:

$$\Pi(\bar{x}) \wedge \chi(bx) \wedge \rho(\bar{x}, \bar{x}') \rightarrow \ \Pi(\bar{x}'),$$

$$\Pi(\bar{x}) \wedge \chi(bx) \wedge \rho(\bar{x}, \bar{x}') \rightarrow \ \chi(\bar{x}')$$

It is easy to write a general solution for the first equation.

$$\Pi(\bar{x}) \wedge \chi(bx) \wedge \rho(\bar{x}, \bar{x}') \rightarrow \ \Pi(\bar{x}')$$

Denote $R(\bar{x}) := \ \forall \bar{x}'(\rho(\bar{x}, \bar{x}') \rightarrow \ \Pi(\bar{x}'))$.

$$\Pi(\bar{x}) \wedge \chi(bx) \rightarrow R(x)$$

$$\chi(\bar{x}) \rightarrow (\Pi(x) \rightarrow R(x))$$

The general solution:

$$\chi(\bar{x}) = \ (\Pi(x) \rightarrow R(x)) \wedge X(\bar{x}).$$

The second equation of the system:

$$\Pi(\bar{x}) \wedge \chi(bx) \wedge \rho(\bar{x}, \bar{x}') \rightarrow \ \chi(\bar{x}') \tag{5}$$

Substitute $\bar{x}' := \bar{x}$:

$$\Pi(\bar{x}) \wedge \chi(bx) \wedge \rho(\bar{x}, \bar{x}) \rightarrow \ \chi(\bar{x}) \tag{6}$$

The general solution of (6):

$$\chi(\bar{x}) = \ Xx \vee \Pi(\bar{x}) \wedge \rho(\bar{x}, \bar{x}) \wedge Y(\bar{x}). \tag{7}$$

But does it satisfy (3)?

Fresh Start

Consider again a finite state system

$$S = (I, T)$$

where $I(\bar{x})$ is a Boolean formula describing initial states of the system, $T(\bar{x}, \bar{x}')$ is a transition formula characterizing possible next states $\bar{x}'$ for a given state $\bar{x}$.
Given a property (Boolean formula) $P(\bar{x})$ we would like to test whether this property is *invariant*, that is,

$$s \models P(x) \text{ for each } \textit{reachable} \text{ state},$$

and moreover would like to construct a formula $G(x)$ such that $P(x) \wedge G(x)$ is inductive:

$$I \rightarrow P(x) \wedge G(x); \qquad P \wedge G \wedge T \rightarrow P' \wedge G'.$$

Let

$$\{c_1, \ldots, c_K\} \tag{8}$$

be the complete list of reachable states. We assume that $I(\bar{x}) \wedge P(\bar{x})$ is satisfiable, otherwise invariance of $P(\bar{x})$ is decided in one step: "yes", if $K = 0$, "no" otherwise.

A Simple Fact

Let $REACH = c_1 \vee \ldots \vee c_K$

Lemma (simple facts)

If P is invariant then $P \wedge REACH$ is inductive and

$$P \wedge REACH \leftrightarrow REACH. \tag{9}$$

Proof. Assume that P is invariant. Since it holds in each initial state $I \rightarrow P$ is valid.
To prove $P \wedge REACH \wedge T \rightarrow REACH' \wedge G'$ let's analyze the disjunction in $REACH$. Assuming one of c_i and $T(c_i, \bar{x}')$, we have that $\bar{x}'$ should also be one of s_i, and hence $REACH(\bar{x}')$ follows from the definition of $REACH$, and $P(\bar{x}')$ follows from the invariance of P.
(9) now follows from invariance: $REACH \rightarrow P$.