Math 267a - Propositional Proof Complexity

Lecture #4: 28 January 2002

Lecturer: Sam Buss

Scribe Notes by: Alan Nash

1 The Unification Problem

1.1 The Problem

Last time we looked at how to solve the system: $x = \phi$, $\phi = \psi \to x$. In general, given:

- Variables: $x, y, z, \ldots$
- Function Symbols: $f, g, h, \ldots$, each with specified arity (including constants as 0-ary functions)
- Terms: built from variables and function symbols
- Finite sets of equations: $s_i \doteq t_i$ (where s_i, t_i are terms)

the *unification problem* is to find a substitution σ such that $s_i\sigma = t_i\sigma$ where equality here is equality of symbols.

Example Take the system with a single binary function symbol f and equations:

$$\begin{aligned} x_1 &\doteq f(x_2, x_2) \\ x_2 &\doteq f(x_3, x_3) \\ x_3 &\doteq f(x_4, x_4) \end{aligned}$$

a solution is given by σ such that:

$$\begin{aligned} \sigma(x_3) &= f(x_4, x_4) \\ \sigma(x_2) &= f(f(x_4, x_4), f(x_4, x_4)) \\ \sigma(x_1) &= f(f(f(x_4, x_4), f(x_4, x_4)), f(f(x_4, x_4), f(x_4, x_4))) \end{aligned}$$

As this example shows, in the worst case the solution's size is exponential in the size of the problem (total number of symbols to represent it).

Example The system with the single equation $f(x_1, x_2) = g(x_1, x_2)$ is unsolvable: the function symbols clash.

Example The system with equations $x \doteq g(y)$ and $y \doteq h(x)$ is unsolvable (the solution must be finite): it yields the derived equation $x \doteq g(h(x))$ ("occurs check").

We will see that these are the only two kinds of things that can go wrong.

1.2 An Algorithm for the Unification Problem

First, notice that terms can be represented by ordered directed acyclic diagrams (ODAGs). By 'ordered' we mean that there is a total ordering on each set of edges coming out of a vertex. When represented as ODAGs, the solutions are always polynomial in size.

To solve an unification problem, we first define an equivalence relation $\approx$ on terms, as follows:

1. $s \doteq t \rightarrow s \approx t$ (equation in system)
2. $s = t \rightarrow s \approx t$ (equal as strings)
3. $f(s_1, s_2, \ldots, s_k) \approx f(t_1, t_2, \ldots, t_k) \rightarrow \forall i(s_i \approx t_i)$
4. $s \approx t \rightarrow t \approx s$
5. $r \approx s \approx t \rightarrow r \approx t$

Example Given the system $f(x, g(y)) \doteq f(h(y), z)$ we have $x \approx h(y)$ and $z \approx g(y)$

Claim A unification problem is solvable iff (a) There are no $r \approx s$ so that r and s have different principal (outermost) function symbols and (b) The $\approx$-equivalence classes are well-ordered (i.e. no cycles) under the extension of the proper subterm relation to the equivalence classes (i.e., $[r] \prec [s]$ iff r is a proper subterm of s)

Proof: [$\Rightarrow$] suppose that σ is a solution. Then it is easy to show that $r \approx s$ implies $r\sigma = s\sigma$ holds, by induction on the cases defining $\approx$. Similarly, the relation "$r\sigma$ is a proper subterm of $s\sigma$" is a well-ordering that refines $r \prec s$.

[$\Leftarrow$] Define σ by induction along $\prec$ as follows. The base elements are of the form $[x]$, containing only variables and at most a single constant c. If $c \in [x]$ then define $\sigma(x) = c$. Otherwise, define $\sigma(x) = y$ where y is a new variable that depends only on $[x]$.

For the rest, i.e. $x \in [f(s_1, \ldots, s_k)]$ define $\sigma(x) = (f(s_1, \ldots, s_k))\sigma = f(s_1\sigma, \ldots, s_k\sigma)$.

Subclaim: $v \approx s$ implies $r\sigma = s\sigma$ (almost immediate).

Partial proof: Suppose $r = f(r_1, \ldots, r_k)$ and $s = f(s_1, \ldots, s_k)$. Then by the induction hypothesis, we have $r_i\sigma = s_i\sigma$.

What we have just described is a polynomial time algorithm. We can obtain the transitive closure using, for example, a breadth-first search or some other similar means (linear in size of graph). In fact, it is quadratic time. Notice that we have considered so far the decision problem; to provide an output in polynomial time we must output ODAGs.

supply reference: Paterson and Wegman provide a linear-time algorithm.

To unify a set of terms $\{r, s, t, \ldots\}$ means to unify the set $\{r \doteq s, s \doteq t, \ldots\}$. Conversely, to unify $\{r_i \doteq s_i : 1 \leq i \leq k\}$ is the same as to unify $\{f(r_1, r_2, \ldots, r_k) = f(s_1, s_2, \ldots, s_k)\}$

2 Extended Frege Systems (Again)

Now we look at an alternative definition of extended Frege systems ($e\mathcal{F}$-systems). An $e\mathcal{F}$-system is a Frege system ($\mathcal{F}$-system) augmented by *the extension rule*. That is, an $e\mathcal{F}$-proof consists of formulas $\phi_1, \ldots, \phi_n$ where each ϕ_i is:

- an axiom
- inferred by a rule from previous formulas
- is $z \leftrightarrow \psi$ where
  - z does not occur in ψ,
  - z does not occur in any previous line of the proof, and
  - z does not occur in ϕ_n.

To convert an $e\mathcal{F}$-proof into a $\mathcal{F}$-proof, proceed as follows:

- Replace z by ϕ wherever z occurs
- Replace $\psi \leftrightarrow \psi$ with a $\mathcal{F}$-proof of it ($O(1)$ lines).

Theorem 1 *(Statman [1]) An n-line $\mathcal{F}$-proof of ϕ can be converted into an $e\mathcal{F}$proof of ϕ with $O(n + |\phi|)$ lines and $O(n + |\phi|^2)$ symbols.*

Theorem 2 *Any two $e\mathcal{F}$-systems p-simulate each other*

Conjecture 1 *$\mathcal{F}$-systems do not simulate $e\mathcal{F}$-systems*

Notice that in polynomial-size $\mathcal{F}$-proofs, each line is a polynomial-size formula, while in polynomial-size $e\mathcal{F}$-proofs, each line is equivalent to a polynomial-size circuit. This converts to circuits; every use of resolution defines a circuit that is then used as input to subsequent circuits: $\psi(x_1, \ldots, x_k, z_1, \ldots, z_\ell)$ where $z_i \leftrightarrow \chi_i(x_1, \ldots, x_k, z_i, \ldots, z_{i-1})$.

Open Problem Polynomial size formulas have the same expressive power as polynomial circuits

Not only we do not know whether $\mathcal{F}$ systems simulate $e\mathcal{F}$ systems and whether p-size formulas simulate p-size circuits, we also do not know how to prove either one of this implications if the other one holds.

Proof System	Nonuniform Complexity	Uniform Complexity	Bounded Arithmetic
$\mathcal{F}$	p-size formulas	ALOGTIME	TNC_1
$e\mathcal{F}$	p-size circuits	P	PV/S_2^1

For references on this table, see Steve Cook's slides from the Edinburgh Complexity Workshop, October 2001, available at

`http://www.cs.toronto.edu/ sacook/edinburgh.ps`

References

[1] R. Statman, *Complexity of derivations from quantifier-free Horn formulae, mechanical introduction of explicit definitions, and refinement of completeness theorems*, in Logic Colloquium '76, R. Gandy and M. Hyland, eds., Amsterdam, 1977, North-Holland, pp. 505–517.