1994

Recursive types and the subject reduction theorem

Richard Statman
Carnegie Mellon University

RECURSIVE TYPES AND THE SUBJECT REDUCTION THEOREM

by

Rick Statman
Department of Mathematics
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Research Report No. 94–164
March, 1994

RECURSIVE TYPES
and
THE SUBJECT REDUCTION THEOREM

by
Rick Statman
Carnegie-Mellon University
NSF CCR-9201893

INTRODUCTION

Dana Scott began talking about type algebras before 1975 ([9]) and his ideas have been persued by Albert Meyer and Val Breazu-Tannen ([3]) among others. It is clear that in this framework the notion of recursive type ([8]) can be developed and principal properties proved. The notion of type algebra provides an alternative enviornment to the mu calculus([4]). In this setting it is clear that the regular tree semantics ([4]) is just one possible semantics for recursive types. This semantics identifies

$$p=p\text{->}p$$

and

$$p=(p\text{->}p)\text{->}p$$

but the latter fails to syntactically type $\lambda x.\ xx$. In other words syntactic typing makes more distinctions. There is even a very interesting result concerning syntactic typing due to Mendler ([8]). Here we shall persue the syntactic approach.

In this note we shall prove that the type algebras which satisfy Curry's subject reduction theorem are precisely the recursive types. Here lambda calculus and type theory are in remarkable harmony. Along the way we shall solve the type inference problem for recursive types. This problem has not been adaquately considered in the literature.

PRELIMINARIES

Type expressions are built up from atoms p,q,r,... by ->. A,B,C, ... are type expressions. A type algebra T is a congruence on the type expressions. Instead of T(A,B) we write T |- A=B. T can be presented by sets of identities. If S is such a set we will often abuse notation and write S |- A=B instead of T |- A=B.

A simultaneous recursion R is a set of identities of the form

p=A

such that

(1) for each p there is at most one A such that p=A:R

(2) each A is a non-atom.

The set of atoms p such that there exists an A with p=A:R is denoted def(R) and the map p |-> A such that p=A:R with domain def(R) is denoted R(.). When R is finite we can write it as the simultaneous system

p1=A1(p1,...,pn)

.

.

.

pn=An(p1,...,pn).

This can be solved in the mu calculus ([ ]) by

q1 <- mux1 A1(x1,...,xn)

.

.

.

qn <- muxn An([qn-1/xn-1](...([q2/x2]q1)...),...,qn-1,xn)

pn <- qn

.

.

.

p1 <- [p2/x2,...,pn/xn]q1.

More generally it is easy to see that for finite recursions our formalization and the mu calculus are equivalent.

INVERTIBILITY

With each simultaneous recursion R we associate a rewrite system R+= { R(p) >-> p | p: def(R) }. Since the rules of R+ are length decreasing if there are no critical pairs then R+ is Church-Rosser. In particular, we can pass to what amounts to the Knuth-Bendix completeion to insure Church-Rosser as follows.

Suppose that s:Natural Numbers-> AtomsxAtoms such that

(a) s(t) has distinct coordinates

(b) if p and q are distinct then { t | s(t)=<p,q> } is infinite.

Here s is to be thought of as a fair schedule of pairs. Define two sequences Rn,Sn of sets of identities as follows. R0=R and S0= empty. If Rn and Sn are defined and s(n+1)=<p0,p1> then

(1) If p0=A0 and p1=A1 belong to Rn, A1 is a proper subexpression of A0 and A0* is the result of replacing each occurrence of A1 by p1 in A0 then Rn+1=Rn-{p0=A0}+{p0=A0*} and Sn+1= Sn

(2) If p0=A and p1=A belong to Rn and pi<p(1-i) , where we here assume that the atoms have somehow been encoded as natural numbers, then Rn+1=[pi/p(1-i)]Rn and Sn+1=Sn+{p(1-i)= pi}

and otherwise we set Rn+1=Rn and Sn+1=Sn. Now we let S^= U{Sn | n=0,1,...} and R^= {p=A | p=A: Rn for all but finitely many n}.

The principal fact about the above construction is that R is logically equivalent to R^ U S^. To see this we trace a given member of R through the construction of the Rn and Sn as a finite set Fn such that Fn+1 |= Fn. We begin with p=A:R and set F0={p=A}. To go from Fn to Fn+1 in case

(1) we replace p0=A0 by p1=A1 and p0=A0*

(2) we replace each identity E by [pi/p(1-i)]E and add p(1-i)=p1. Note that only finitely many changes can be made in the Fn by (1) since each such operation reduces the number of nested (non-outermost) ->'s. In addition,the atoms in the last such Fn resulting from operation (1) can be changed and augmented by (2) at most finitely often since the operation (2) decreases their numerical value. Thus there exists a t such that for all n>t Ft is contained in Fn. Thus Ft is contained in R^ U S^ and R^ U S^ |= p=A.

Now it is easily seen that R^ is a simultaneous recursion and R^+ has no critical pairs. The structure of S^ is similar. The members of S^ have the form p=q where p>q. For each p there is at most one q such that p=q:S^ and if such a q exists p does not occur in a member of R^. Let S^+ = { p>->q | p=q:S^ }; then R^+ U S^+ has no critical pairs and is terminating. Finally, if R |- A=B then there exist A+,B+ and C such that A>->>A+ and B>->> B+ in S^+ and A+>->>C<<-<B+ in R^+. Because S^+ is logically trivial we shall assume that all of our recursions are presented in the form of R^,i.e. no critical pairs.

A type algebra T is said to be invertible if whenever we have T |- A->B=C->D then T |- A=C and T |- B=D. We can now see that every simultaneous recursion R is invertible for if R |- A->B = C->D then by the Church-Rosser theorem there exists a type expression E such that A->B >->> E<<-< C->D. If E is an atom then the last step in each of the two reductions is the only step which uses the entire expression as a redex. Since R is a recursion it must be of the from R(E)>->E. In particular, R(E)= E0->E1 and A>->>E0<<-<C and B>->>E1<<-<D so R |- A=C and R |- B=D. If E is not an atom it has the form E0->E1 and no reductions use the entire expression as a redex. Thus A>->>E0<<-<C and B>->>E1

<<-<D and as before R |- A=C and R |- B=D.

Conversely every invertible type algebra can be presented as a simultaneous recursion together with a logically trivial set of identities between atoms. This will be seen below.

SOLVABILITY OF EQUATIONS

The type algebra T is said to solve the equation A=B if there is a homomorphism (substitution) h such that T |- h(A)=h(B). Here we wish to determine all solutions to a system S in a simultaneous recursion R. Note that for any homomorphism h, h(S) has the invertibility property. Let |-* refer to a congruence generated by using the invertibility rule in addition to the laws of logic. For each sub-congruence class S[p]={q | q an atom and S |-* q=p } pick a cannonical member p* and let A[p*] be any shortest type expression such that

(1) S |- *p*=A[p*]

(2) A[p*] contains only *'ed atoms.

Put S*= {p*=A[p*] | A[p*] a non-atom } and let T= { p=p* | p an atom }. We claim that S* U T logically implies S. To see this we prove by induction that if B is built up only from *'ed atoms and S |-* p*=B then S* |- p*=B. This is trivial if B is p* itself. Otherwise B=C->D and A[p*]=A0->A1, and S |-* C=A0 & D=A1. Now,by induction hypothesis S* |- C=A0 & D=A1, thus S* |- p*=B. The claim now follows easily.Since S |-* S* U T we now see that that S and S* U T have the same solutions in any simultaneous recursion R. Since T is logically trivial we now consider only the problem of solving simultaneous recursions S in simultaneous recursions R. This show the claim above that invertibility => recursiveness. Note also that if p* does not appear in A[p*] then p* can be e-liminaterd by substitution, so we shall assume that S is a simultaneous recursion such that p=A:S => p appears in A.

Given a homomorphism h we may think of h as a substitution of expressions for atoms.Since R+ is terminating&Church-Rosser we may assume that the expression substituted for an atom is in normal form. Thus we have $h(A) >\!\!->> h(p)$ for each p=A:S. Now any occurrence of h(p) in h(A) can have no residual in h(p) so, since h(p) is itself normal, h(p) must be a subexpression of an R+ redex. This bounds all solutions of S in R.

These remarks imply that solvability of S in R is an NP time problem. Indeed we have the

PROPOSITION The solvability of S in R is NP complete even for a fixed R. 2

PROOF: For each k define Rk as follows. Rk has the k atoms c1,c2,...,ck, and $d<i,j>$ for $0<i<j<k+1$. It consists of the identities

$$d<i,j> = ci->(cj->d<i,j>) \quad \text{for } 0<i<j<k+1.$$

The fixed R mentioned above will be R3. We shall encode graph colorability into solvability in the Rk above. Suppose the graph G=(V,E) is given. Define the simultaneous recursion S(G) as follows.

The atoms of S(G) are v1,...,vn for V={v1,...,vn},and e_{i,j} for each {i,j}:E. The identities of S(G) are the following

$$e\_\{i,j\}=vi->(vj->e\_\{i,j\}) \text{ for } \{i,j\}:E$$

Now suppose that F is a k coloring of G in the colors {c1,...,ck}. We define a homomorphism h by

h(vi)=F(vi)

$h(e\_\{i,j\})=d<F(i),F(j)>$ where we assume $F(i)<F(j)$.

Clearly h solves S(G) in Rk. Now suppose that h is a solution of S(G) in Rk. By invertibility and Church-Rosser there exists a k coloring F in {c1,...,ck} such that h(vi)=F(vi). Thus S(G) is solvable in Rk $<=>$ Gis k colorable.

Let]-[be the equivalence relation on atoms generated by the relation p[q if VA (p=A:S & q appears in A). Let [- be the corresponding poset of equivalence classes. We have the following

COROLLLARY When [- is a flat ordering the problem of determining if S is solvable in R is P-time.

PROOF: We need only try each subexpression in R for any particular atom of S, for each equivalence class of]-[independently.

TYPING LAMBDA TERMS

Given a type algebra T we can type lambda terms a'la Church or a'la Curry ([2] but see [3]). If H is a basis then we write H |- M:A for the typing judgement that M:A is derivable from the basis H. The two notions of typing are equivalent for any type algebra T.

The subject reduction theorem for T is the statement H |- M:A & M->>N => H |- N:A for ->> taken as beta reduction.

A type algebra T0 is said to be an expansion of the type algebra T1 if the atoms of T1 are a subset of the atoms of T0 and the identities of T0 are logically equivalent to those of T1 plus a set of identities p=q for p an atom of T1 and q an atom of T0, and each q occurring at most once. We shall prove the

THEOREM The following are equivalent

(1)T is an expansion of a simultaneous recursion

(2)T is invertible

(3)T satisfies the subject reduction theorem

PROOF: We have in essence already proved that (1) and (2) are equivalent. For the proof that (2)=>(3) we need only copy the proof in [2] observing that all that is used about T is the invertibility property. Now suppose ~(2), we shall show ~(3). Suppose that T |-A->B=C->D but either ~ T |- A=C v ~ T |-

B=D for A->B shortest with this property and A=B if possible within these constraints. We distinguish two cases

(1) ~ T |- B=D

We have y:B, z:C |- (λx.y)z:D but not y:B,z:C |- y:D by the generation lemma for λ->-Curry ([2]). Thus subject reduction fails.

(2) T |- B=D & ~ T |- A=C

Let A=A1->(...(As->p)...). We have the following

u:p->B,v:A->p,y:C |- (λx.u(vx))y:D

Suppose that

u:p->B,v:A->p,y:C |- u(vy):D

By the generation lemma there exists a type expression E s.t. T |- p->B=E->D & A->p=C->E. We argue by cases

(i) s>0

In this case p->B is shorter than A->B so by choice of A->B we have T |- p=E. We have

x1:A1,...,xs:As,y:C |- (λx. xx1...xs)y:p

Suppose that

x1:A1,...,xs:As,y:C |- yx1...xs:p.

Then by the generation lemma

x1:A1,...,xs:As,y:C |- y:A

hence T |- A=C contradicting the choice of case.Thus the subject reduction theorem fails.

(ii) s=0

In this case we have T |- p->B=E->D & p->p=C->E.By the special choice of A->B, since p=A and ~ T|- A=C we have p=B. Thus we have

y:E |- (λx.x)y:p

and

z:C |- (λx.x)z:E.

If subject reduction holds in both cases we have

y:E |- y:p

z:C |- z:E

and by the generation lemma T |- C=E=p=A. This contradicts the choice of case and subject reduction must fail.

Thus is all cases subject reduction fails.This completes the proof.

REFERENCES

[1]Barendregt The Lambda Calculus
North Holland 1984

[2] Lambda calculi with types
in Handbook of Logic in Computer Science
Abramsky,Gabbay,&Maibaum eds.
North Holland 1992

[3]Breazu-Tannen & Meyer Lambda calculus with constrained types
in LNCS 193 : Logics of Programs
Parikh ed
Springer-Verlag 1985

[4]Coppo & Cardone Type inference with recursive types
Information and Computation
1992

[5]Kozen Finitely presented algebras and the polynomial time hierarchy
Dept of Comp. Sci. Tech. Report 77-303
Cornell Univ. 1977

[6] Complexity of finitely presented algebras
Dept. of Comp. Sci. Tech. Report 76-294
Cornell Univ. 1976

[7]Le Chenadec Cannonical Forms in Finitely Presented

Algebras
Pitman&Wiley 1986

[8]Mendler Recursive types and type constraints ...
in Proceedings LICS 1987
IEEE 1987

[9]Scott Some philosophicsl issues concerning theories of combinators
in LNCS 37 : Lambda Calculus and Computer Science Theory
Bohm ed
Springer-Verlag 1975