Computer Algebra Systems: Patterns

Lecture 18

"Symbolic Computation" can be encoded as rewriting systems

- Primitive systems can be shown to be equivalent in computational power to Turing Machines
- (substantial literature on Post, Markov, Thue, ..systems showing they have same undecidability results)

Do we care?

- Are rewriting systems an appropriate model for computation?
  - Obviously for some people, for some computations.
  - Promoted by various groups.
    - Prolog,
    - OPS5
    - Equational logic fans (Knuth-Bendix completion)
      - Associative, Commutative, variations

Using matching to decompose expressions

- A reasonable implementation of a collection of (probably incompletely specified) transformations can be based on templates.
- A typical template:
  - If you see something like this .....A ....B.....C
  - Then do this: make a list: A matches x, B matches y, C matches z.
  - Then call a program f(A,B,C)
  - There may be many different assignments – backtracking is an important issue, e.g. try calling f(A',B',C') if the first call "fails".

Moses' pattern matcher : lisp with markers (SIN/Macsyma) improved on ELINST (Slagle)

- Intermix markers: freevar, coefftt, coeffpt, varp markers,
- (*

```
(^  (var varp) (r1 numberp))
(^
  (+
   (* (coefftt (c2 freevar))
      (^ (var varp) (q free1)))
   (coeffpp (c1 freevar)))
  (r2 numberp))
(coefftt (a freevar)))
```

looks for $x^{r1} *(c2*x^{q}+c1)^{r2} *a$, a "Chebyshev" substitution. If it matches it assigns r1, q, c1, c2, r2, a. Matching requires predicates/markers all succeed; special markers coeffpp coefftt coeffpt collect coefficients.

User level pattern matching- Maple

Maple / example: completing the square, note the "semantic" aspects. a,b,c are free vars. x is an anchor. (similar to Macsyma/ Fateman program)

```
match(5*x^2-3*x+z*x+y =
                a*(x+b)^2+c, x, 's');

s;

{b = - 3/10 + 1/10 z, a = 5, c = - 9/20 + 3/10 z - 1/20 z^2  + y}
```

This also illustrates Maple's terrible syntactic hack for returning multiple values (e.g. return value is true, s= { …})

Matching in Mathematica

- Syntactic, but with assoc/commutative/predicates
- _ anonymous
- x_ named
- x__ (two _) segment as in a+x__
- x___ (three _) segment, maybe empty
- x_Integer e.g. MatchQ[3,x_Integer] →true
  - but MatchQ[3,x_Rational] → false hmm.
- x_. default.. 0 in sum, 1 in product, …

Matching in Mathematica / Segment vars

- Definition Quadm[a_*x^2+b_*x+c_]:= {a,b,c}
- Definition Quadm2 [a_*x^2+b_*x+c___]:= {a,b,c}
- Quadm[r*x^2+s*x+t+u] → {r,s,t+u}
- Quadm2[r*x^2+s*x+t+u] → {r,s,t,u}
- Default[a]=1
- Quadm[a_.*x^2+b_*x+c_]:= {a,b,c}
- Quadm[x^2+s*x+t+u] → {1,s,t+u}
- I'm not sure how to pick out b=0 from a missing term; presumably one would also wish to have x be a variable and associate /.x==var with the pattern.
- Possible to set up exponential-time search e.g. m1[f[a_]+x___ + f[a_+1]]:={a,x}

Matching in Mathematica / Segment vars

- m1[x+y+z+f[r]+f[r+1]] → {r,x,y,z}
- m1[x+y+x+f[3]+f[4]] → nope.
- try this:
  - m2[x___+f[a_]+f[b_]/;a==b-1]:={a,x}

  now m2[x+y+x+f[3]+f[4]] →{3,x,y,z}

  anyway, we can express a predicate that requires examining all combinations of n items taken m at a time.

Patterns/Transformations/Evaluation/Simplification Tend to be intermixed

- a survey on evaluators (and much about systems) is on-line in readings directory.. readings/evalnew.pdf
  - a paper by RJF which appeared as a chapter in M. Wester's *Computer Algebra Systems*