MOLECULAR COMPUTING

William Bricken
October 1993

My latest is molecular computing: imagine programs as chemical soup. You throw in fragments (molecules of the "factorial" function, for eg), the fragments bounce around, interacting at the chemical level, the answer precipitates out of the molecular interactions. Turns out that if the silicon lets you assume this model, then programming gets very easy cause you have a pure declarative parallel computational model.

OK, so programs are actually cake recipes. You throw in ingredients and mix, the ingredients bounce around, interacting in the oven, the answer cooks out of the ingredients. The proof is in the pudding.

Recipe for factorial n:

Toss in the ingredients, the members of the set {1,...,n}

Ingredient interaction, X and Y are any fragments:

```
(X) (Y)  ==>  (X*Y)
```

This is nice cause it removes all linear and recursive structure from the program and computes in O(log n) time. Note that X and Y are not ordered, not even simple since Y could be the result of a previous molecular interaction. Program ends when there are no longer two fragments/molecules to interact, the last remaining molecule is the answer. You can also stop the process at any time and have a partial solution available (for real-time).

OK, so programs are actually like intercourse...

I've got molecular/cooking/fucking models for numbers, logic, sets, and algebra, and a good design for the silicon itself. (Yes, it is boundary math in disguise. Programs are actually bubble baths.)