PROPAGATION = SATISFACTION

William Bricken
May 1989

(This is from unbound memory...)

The difference between constraint propagation and clause satisfaction is historical. Folks (Steele in particular) could do propagation first (early 70's) because it is a subcase of term rewriting (presented as graph rewriting) in which the lhs is a single variable. The rhs was constrained to be a linear relation. That is: x = y + 3.

Algorithms attacking a more general form (constraints) weren't smart enough to know which particular variables to solve for. Algorithms attacking non-linear forms were too costly (this was before folks worried about falling into non-linear chaotic cycles).

Successive subsets of the general pattern-matching and substitution problem were solved with good algorithms. Most all of the theory had been long available in matrix algebra, but AI researchers are particularly fond of making up disciplines from scratch. (I'll volunteer myself as a prime example.) We can throw Prolog, Mathematica, Bertrand, and just about every other implementation of formal systems into the same general Find-and-Replace bag.

The unifying perspective comes from viewing "variables" as purely symbolic marks, independent of semantics. Yes, the set of variables {x, y, z, ...} is identified with a domain, but for the purposes of computation, they are only marks. From here, what the logician calls unification is Match-and-Substitute, what the algebraist calls solving equations is also M&S. The differences are

1) my domain is truth, and yours is only silly numbers. My domain is the continuum, and I have infinite variables and yours are only finite. My domain is recursive functions, and I call it composition but you call proof or solution... and

2) the prioritization of what is done when. Which forms are allowed to be matched, which substitutions are made in what order, ...

Use equational logic and forget the difference between numerical and logical variables (other than they are different marks). Use matrix logic and forget both numbers and truth values. Optimize the hell out of Find-and-Replace, and save all interpretation of what is going on until the result pops out. Explore the Unknown, loosen up that ole Interpretative Apparatus, and Free the Variables!

William the Underbound