Proof:

Ad (1):

Every unknown x_i may change its value at most h times :-) Each time, the list $I[x_i]$ is added to W. Thus, the total number of evaluations is:

$$\leq n + \sum_{i=1}^{n} (h \cdot \# (I[x_i]))$$

= $n + h \cdot \sum_{i=1}^{n} \# (I[x_i])$
= $n + h \cdot \sum_{i=1}^{n} \# (Dep f_i)$
 $\leq h \cdot \sum_{i=1}^{n} (1 + \# (Dep f_i)))$
= $h \cdot N$

Ad (2):

We only consider the assertion for monotonic f_i . Let D_0 denote the least solution. We show:

- $D_0[x_i] \supseteq D[x_i]$ (all the time)
- $D[x_i] \not\supseteq f_i \text{ eval} \implies x_i \in W$ (at exit of the loop body)

On termination, the algo returns a solution :-))

Discussion:

- In the example, fewer evaluations of right-hand sides are required than for RR-iteration :-)
- The algo also works for non-monotonic f_i :-)
- For monotonic f_i , the algo can be simplified:

$$D[x_i] = D[x_i] \sqcup t; \implies D[x_i] = t;$$

• In presence of widening, we replace:

$$D[x_i] = D[x_i] \sqcup t; \implies D[x_i] = D[x_i] \sqcup t;$$

• In presence of Narrowing, we replace:

$$D[x_i] = D[x_i] \sqcup t; \implies D[x_i] = D[x_i] \sqcap t;$$

Warning:

• The algorithm relies on explicit dependencies among the unknowns.

So far in our applications, these were obvious. This need not always be the case :-(

- We need some strategy for extract which determines the next unknown to be evaluated.
- It would be ingenious if we always evaluated first and then accessed the result ... :-)

 \implies recursive evaluation ...

Idea:

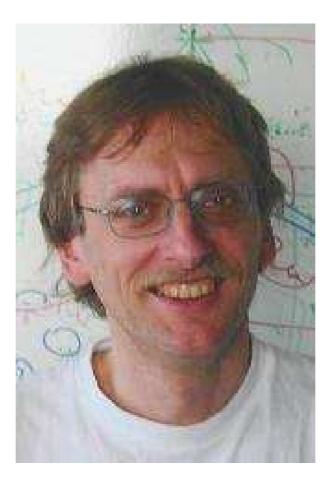
→ If during evaluation of f_i , an unknown x_j is accessed, x_j is first solved recursively. Then x_i is added to $I[x_j]$:-)

eval
$$x_i x_j$$
 = solve x_j ;
 $I[x_j] = I[x_j] \cup \{x_i\};$
 $D[x_j];$

→ In order to prevent recursion to descend infinitely, a set *Stable* of unknown is maintained for which **solve** just looks up their values :-) Initially, *Stable* = \emptyset ...

The Function solve :

solve x_i = if $(x_i \notin Stable)$ { Stable = Stable $\cup \{x_i\};$ $t = f_i (\text{eval } x_i);$ if $(t \not\sqsubseteq D[x_i])$ { $W = I[x_i]; \quad I[x_i] = \emptyset;$ $D[x_i] = D[x_i] \sqcup t;$ Stable = Stable \setminus W; app solve W; } }



Helmut Seidl, TU München ;-)

Example:

Consider our standard example:

 $x_1 \supseteq \{a\} \cup x_3$ $x_2 \supseteq x_3 \cap \{a, b\}$ $x_3 \supseteq x_1 \cup \{c\}$

A trace of the fixpoint algorithm then looks as follows: