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:

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

Ad (2):

We only consider the assertion for monotonic f_i .

Let D_0 denote the least solution. We show:

- $D_0[x_i] \sqsupseteq D[x_i]$ (all the time)
- $D[x_i] \not\sqsupseteq f_i\,\mathsf{eval} \implies x_i \in W$ (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:

$$\boxed{D[x_i] = D[x_i] \sqcup t;} \quad \Longrightarrow \quad \boxed{D[x_i] = \qquad\quad t;}$$

- In presence of widening, we replace:

$$\boxed{D[x_i] = D[x_i] \sqcup t;} \quad \Longrightarrow \quad \boxed{D[x_i] = D[x_i] \sqcup\!\!\!\!\_ \; t;}$$

- In presence of Narrowing, we replace:

$$\boxed{D[x_i] = D[x_i] \sqcup t;} \quad \Longrightarrow \quad \boxed{D[x_i] = D[x_i] \sqcap\!\!\!\!\!\!\overline{} \; t;}$$

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 ... :-)

$\Longrightarrow$ 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 $I[x_j]$:-)

$$\begin{aligned} \text{eval } x_i \; x_j \quad = \quad & \text{solve } x_j; \\ & I[x_j] = I[x_j] \cup \{x_i\}; \\ & D[x_j]; \end{aligned}$$

→ 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 :

$$
\begin{array}{lll}
\text{solve } x_i & = & \text{if } (x_i \notin \mathit{Stable}) \ \{ \\
 & & \quad \mathit{Stable} = \mathit{Stable} \cup \{x_i\}; \\
 & & \quad t = f_i \, (\text{eval } \ x_i); \\
 & & \quad \text{if } (t \not\sqsubseteq D[x_i]) \ \{ \\
 & & \quad\quad W = I[x_i]; \quad I[x_i] = \emptyset; \\
 & & \quad\quad D[x_i] = D[x_i] \sqcup t; \\
 & & \quad\quad \mathit{Stable} = \mathit{Stable} \backslash W; \\
 & & \quad\quad \text{app } \ \text{solve } \ W; \\
 & & \quad \} \\
 & & \}
\end{array}
$$

Helmut Seidl, TU München ;-)

Example:

Consider our standard example:

$$
\begin{array}{lll}
x_1 & \supseteq & \{a\} \cup x_3 \\
x_2 & \supseteq & x_3 \cap \{a, b\} \\
x_3 & \supseteq & x_1 \cup \{c\}
\end{array}
$$

A trace of the fixpoint algorithm then looks as follows: