Interprocedural analysis: Sharir-Pnueli’s functional approach

Deepak D’Souza

Department of Computer Science and Automation
Indian Institute of Science, Bangalore.

18 September 2013
Outline

1. Functional Approach
2. Example
3. Iterative Approach
4. Exercises
Equations to capture JOP: why it works

- We want JOP at $N$.
- If transfer functions are distributive, then we can take join over paths at any intermediate point $M$, and then join over paths from $M$ to $N$. 
Equation solving: Problems with naive approach

- In non-procedural case, we setup equations to capture JOP assuming distributivity. Least solution to these equations gave us exact/over-approx JOP depending on distributive/monotonic framework.
- Try to set up similar equations for $x_N$ (JVP at program point $N$).
- How do we describe $x_N$ in terms of $x_J$?
Instead try to capture join over **complete** paths first

- Set up equations to capture join over **complete** paths.
- Now set up equations to capture JVP using join over complete path values.
- Root of procedure $p$ is denoted $r_p$.
- Exit (return) of procedure $p$ is denoted $e_p$.
- Sometimes use $r_1$ for $r_{main}$.
- Assume WLOG that main is not called.
Example paths

An example valid path in $IVP(r_1, I)$.

An example valid and complete path in $IVP_0(r_1, D)$.

Path “FGHLFKJMIJ” is valid and complete and is in $IVP_0(r_p, J)$.
Basic idea: Why join over complete paths help

An IVP path $\rho$ from $r_1$ to $N$ in procedure $p$ can be written as $\delta \cdot \eta$ where $\delta$ is in $\text{IVP}(r_1, r_p)$, and $\eta$ is in $\text{IVP}_0(r_p, N)$. 

Path $\eta$ is suffix after last pending call to procedure $p$ was made.
For a procedure $p$ and node $N$ in $p$, define:

$$\phi_{r_p,N} : D \rightarrow D$$

given by

$$\phi_{r_p,N}(d) = \bigcup_{\rho \in \text{IVP}_0(r_p,N)} f_{\rho}(d).$$

$\phi_{r_p,N}$ is thus the join of all functions $f_{\rho}$ where $\rho$ is an interprocedurally valid and complete path from $r_p$ to $N$. 
Visualizing $\phi_{r_p,N}$
Using $\phi_{r_p,N}$'s to get JVP values

Assuming distributivity of underlying transfer functions, JVP value at $N$ equals $\phi_{r_p,N}$ applied to JVP value at $r_p$. 
Equations (1) to capture $\phi_{r_p,N}$

\[
\begin{align*}
y_{r_p,r_p} &= id_D & \text{(root)} \\
y_{r_p,N} &= f_{MN} \circ y_{r_p,M} & \text{(stmt)} \\
y_{r_p,N} &= y_{r_q,e_q} \circ y_{r_p,M} & \text{(call)} \\
y_{r_p,N} &= y_{r_p,L \sqcup y_{r_p,M}} & \text{(join)}
\end{align*}
\]
Example: Available expressions analysis

Lattice for Av-Exp analysis.

- Is \( a*b \) available at program point \( N \)?

0 (not available)
1 (available)
\( \bot \)
Example: Available expressions analysis

- 0 (not available)
- 1 (available)
- \(\perp\)

Lattice for Av-Exp analysis.

- Is \(a*b\) available at program point \(N\)?
- No if we consider all paths.

```plaintext
read a,b
\[t := a*b\]
call p
\[t := a*b\]
print t
```

- \(a := a - 1\)
- \(a \neq 0\)
- call p
- call p
- \(t := a*b\)
- ret
Example: Available expressions analysis

- 0 (not available)
- 1 (available)
- ⊥

Lattice for Av-Exp analysis.

- Is \(a \times b\) available at program point \(N\)?
  - No if we consider all paths.
  - Yes if we consider interprocedurally valid paths only.
Functions we will use for example analysis

- \( D = \{ \bot, 1, 0 \} \).
- \( 0 : D \rightarrow D \) given by
  
  \begin{align*}
  \bot & \mapsto \bot \\
  0 & \mapsto 0 \\
  1 & \mapsto 0
  \end{align*}

- \( 1 : D \rightarrow D \) given by
  
  \begin{align*}
  \bot & \mapsto \bot \\
  0 & \mapsto 1 \\
  1 & \mapsto 1
  \end{align*}

- \( \text{id} : D \rightarrow D \) given by
  
  \begin{align*}
  \bot & \mapsto \bot \\
  0 & \mapsto 0 \\
  1 & \mapsto 1
  \end{align*}

- Ordering: \( 1 \leq \text{id} \leq 0 \).
Example: Equations for $\phi$'s

\[
\begin{align*}
 y_{A,A} & = id \\
 y_{A,B} & = 0 \circ y_{A,A} \\
 y_{A,C} & = 1 \circ y_{A,B} \\
 y_{A,P} & = y_F, J \circ y_{A,C} \\
 y_{A,D} & = 1 \circ y_{A,P} \\
 y_{A,E} & = id \circ y_{A,D} \\
 y_{F,F} & = id \\
 y_{F,G} & = id \circ y_{F,F} \\
 y_{F,K} & = id \circ y_{F,F} \\
 y_{F,H} & = 0 \circ y_{F,G} \\
 y_{F,Q} & = y_F, J \circ y_{F,H} \\
 y_{F,I} & = 1 \circ y_{F,Q} \\
 y_{F,J} & = y_F, I \sqcup y_{F,K} \\
\end{align*}
\]
Using $\phi_{r_p,N}$'s to get JVP values

Assuming distributivity of underlying transfer functions, JVP value at $N$ equals $\phi_{r_p,N}$ applied to JVP value at $r_p$. 
Equations (2) to capture JVP

\[ x_1 = d_0 \]
\[ x_{r_p} = \bigcup_{\text{calls } C \to p} x_C \]
\[ x_N = \phi_{r_p, N}(x_{r_p}) \quad \text{for } N \in \text{ProgPts}(p) - \{r_p\}. \]
Example: Equations for $x_N$'s (JVP)

\[
\begin{align*}
    x_A &= 0 \\
    x_B &= 0(x_A) \\
    x_C &= 1(x_A) \\
    x_P &= 1(x_A) \\
    x_D &= 1(x_A) \\
    x_E &= 1(x_A) \\
    x_F &= x_C \sqcup x_H \\
    x_G &= \text{id}(x_F) \\
    x_K &= \text{id}(x_F) \\
    x_H &= 0(x_F) \\
    x_Q &= 0(x_F) \\
    x_I &= 1(x_F) \\
    x_J &= \text{id}(x_F).
\end{align*}
\]

Fig. shows values of $\phi_{r_p,N}$'s in bold.
Correctness claims

- Consider lattice \((F, \leq)\) of functions from \(D\) to \(D\), obtained by closing the transfer functions, identity, and \(f_{\perp} : d \mapsto \perp\) under composition and join. (Alternatively we can take \(F\) to be all monotone functions on \(D\).)
- Ordering is \(f \leq g\) iff \(f(d) \leq g(d)\) for each \(d \in D\).
- \((F, \leq)\) is also a complete lattice.
- \(\bar{f}\) induced by Eq (1) is monotone on complete lattice \((\overline{F}, \leq)\).
  - Sufficient to argue that function composition \(\circ\) is monotone when applied to monotone functions.
  - Join operation \(\bigvee\) is monotone.
- LFP / least solution (say \(y_{r_p,N}^*\)'s) exists by Knaster-Tarski.
- Each \(y_{r_p,N}^*\) is necessarily monotonic.

Claim

\(\phi_{r_p,N}\)'s are the least solution to Eq (1) (i.e. \(\phi_{r_p,N} = y_{r_p,N}^*\)) when \(f_{MN}\)'s are distributive. Otherwise each \(\phi_{r_p,N} \leq y_{r_p,N}^*\).
Using Kildall to compute LFP

- We can use Kildall’s algo to compute the LFP of these equations as follows.
  - Initialize the value at program points with RHS of the constant equations (in this case \( id \) at entry of procedures), and the bottom value (in this case \( f_\perp \)) everywhere else.
  - Mark all values
  - Pick a marked value at point say \( N \), and “propagate” it (i.e. for any node \( M \) in the LHS of an equation in which \( N \) occurs in the RHS, evaluate \( M \) and join it with the existing value at \( M \)). Mark as before in Kildall’s algo.
  - Stop when no more marked values to propagate.

- Kildall’s algo will compute \( y_{r_p,N}^* \) if \( D \) is finite. Note that finite height of \( (D, \leq) \) is not sufficient for termination.
Correctness and algo - II

Consider Eq (2)’:

\[
\begin{align*}
x_1 &= d_0 \\
x_{r_p} &= \bigsqcup \text{calls } C \text{ to } p \times C \\
x_N &= y_{r_p,N}^*(x_{r_p}) \quad \text{for } N \in N_p - \{r_p\}.
\end{align*}
\]

(Recall that \(y_{r_p,N}^*\)’s are the least solution of Eq (1).)

- \(\overline{f}\) induced by Eq (2)’ is a monotone function on the complete lattice \((\overline{D}, \leq)\).
- LFP / least solution (say \(x_N^*\)’s) exists by Knaster-Tarski.

**Claim**

JVP values are the least solution to Eq (2)’ (i.e. \(\text{JVP}_N = x_N^*\)) when \(f_{MN}\)’s are distributive. Otherwise \(\text{JVP}_N \leq x_N^*\) for each \(N\).

Kleene/Kildall’s algo will compute \(x_N^*\)’s (assuming \(D\) finite).
Example: Computing $\phi_{r_p,N}$’s ($y_{r_p,N}^*$ to be precise) using Kildall’s algo

$y_{A,A} = id$
$y_{A,B} = 0 \circ y_{A,A}$
$y_{A,C} = 1 \circ y_{A,B}$
$y_{A,P} = y_{F,J} \circ y_{A,C}$
$y_{A,D} = 1 \circ y_{A,P}$
$y_{A,E} = id \circ y_{A,D}$

$y_{F,F} = id$
$y_{F,G} = id \circ y_{F,F}$
$y_{F,K} = id \circ y_{F,F}$
$y_{F,H} = 0 \circ y_{F,G}$
$y_{F,Q} = y_{F,J} \circ y_{F,H}$
$y_{F,I} = 1 \circ y_{F,Q}$
$y_{F,J} = y_{F,I} \sqcup y_{F,K}$
Example: Computing $\phi_{r_p,N}$’s ($y_{r_p,N}^*$ to be precise) using Kildall’s algo

\[
\begin{align*}
\gamma_{A,A} &= id \\
\gamma_{A,B} &= 0 \circ \gamma_{A,A} \\
\gamma_{A,C} &= 1 \circ \gamma_{A,B} \\
\gamma_{A,P} &= \gamma_F,J \circ \gamma_{A,C} \\
\gamma_{A,D} &= 1 \circ \gamma_{A,P} \\
\gamma_{A,E} &= id \circ \gamma_{A,D} \\
\gamma_{F,F} &= id \\
\gamma_{F,G} &= id \circ \gamma_{F,F} \\
\gamma_{F,K} &= id \circ \gamma_{F,F} \\
\gamma_{F,H} &= 0 \circ \gamma_{F,G} \\
\gamma_{F,Q} &= \gamma_F,J \circ \gamma_{F,H} \\
\gamma_{F,I} &= 1 \circ \gamma_{F,Q} \\
\gamma_{F,J} &= \gamma_{F,I} \sqcup \gamma_{F,K}
\end{align*}
\]
Example: Computing $\phi_{r_p,N}$’s ($y_{r_p,N}^*$ to be precise) using Kildall’s algo

$$y_{A,A} = id$$

$$y_{A,B} = 0 \circ y_{A,A}$$

$$y_{A,C} = 1 \circ y_{A,B}$$

$$y_{A,P} = y_F, J \circ y_{A,C}$$

$$y_{A,D} = 1 \circ y_{A,P}$$

$$y_{A,E} = id \circ y_{A,D}$$

$$y_{F,F} = id$$

$$y_{F,G} = id \circ y_{F,F}$$

$$y_{F,K} = id \circ y_{F,F}$$

$$y_{F,H} = 0 \circ y_{F,G}$$

$$y_{F,Q} = y_F, J \circ y_{F,H}$$

$$y_{F,I} = 1 \circ y_{F,Q}$$

$$y_{F,J} = y_F, I \uplus y_F, K$$
Example: Computing $\phi_{r_p,N}$’s ($y_{r_p,N}^*$ to be precise) using Kildall’s algo.

- $y_{A,A} = id$
- $y_{A,B} = 0 \circ y_{A,A}$
- $y_{A,C} = 1 \circ y_{A,B}$
- $y_{A,P} = y_{F,J} \circ y_{A,C}$
- $y_{A,D} = 1 \circ y_{A,P}$
- $y_{A,E} = id \circ y_{A,D}$

- $y_{F,F} = id$
- $y_{F,G} = id \circ y_{F,F}$
- $y_{F,K} = id \circ y_{F,F}$
- $y_{F,H} = 0 \circ y_{F,G}$
- $y_{F,Q} = y_{F,J} \circ y_{F,H}$
- $y_{F,I} = 1 \circ y_{F,Q}$
- $y_{F,J} = y_{F,I} \uplus y_{F,K}$
Example: Computing $\phi_{r_p,N}$’s ($y^*_{r_p,N}$ to be precise) using Kildall’s algo

$$
\begin{align*}
\gamma_A, A &= id \\
\gamma_A, B &= 0 \circ \gamma_A, A \\
\gamma_A, C &= 1 \circ \gamma_A, B \\
\gamma_A, P &= y_F, J \circ \gamma_A, C \\
\gamma_A, D &= 1 \circ \gamma_A, P \\
\gamma_A, E &= id \circ \gamma_A, D \\
\gamma_F, F &= id \\
\gamma_F, G &= id \circ \gamma_F, F \\
\gamma_F, K &= id \circ \gamma_F, F \\
\gamma_F, H &= 0 \circ \gamma_F, G \\
\gamma_F, Q &= y_F, J \circ \gamma_F, H \\
\gamma_F, I &= 1 \circ \gamma_F, Q \\
\gamma_F, J &= y_F, I \sqcup y_F, K
\end{align*}
$$
Example: Computing $\phi_{r_p,N}$’s ($y_{r_p,N}^*$ to be precise) using Kildall’s algo

\begin{align*}
y_{A,A} &= id \\
y_{A,B} &= 0 \circ y_{A,A} \\
y_{A,C} &= 1 \circ y_{A,B} \\
y_{A,P} &= y_{F,J} \circ y_{A,C} \\
y_{A,D} &= 1 \circ y_{A,P} \\
y_{A,E} &= id \circ y_{A,D} \\
y_{F,F} &= id \\
y_{F,G} &= id \circ y_{F,F} \\
y_{F,K} &= id \circ y_{F,F} \\
y_{F,H} &= 0 \circ y_{F,G} \\
y_{F,Q} &= y_{F,J} \circ y_{F,H} \\
y_{F,I} &= 1 \circ y_{F,Q} \\
y_{F,J} &= y_{F,I} \sqcup y_{F,K}
\end{align*}
Example: Computing $\phi_{r_p,N}$'s ($y^*_{r_p,N}$ to be precise) using Kildall's algo

$y_{A,A} = \text{id}$
$y_{A,B} = 0 \circ y_{A,A}$
$y_{A,C} = 1 \circ y_{A,B}$
$y_{A,P} = y_F,J \circ y_{A,C}$
$y_{A,D} = 1 \circ y_{A,P}$
$y_{A,E} = \text{id} \circ y_{A,D}$

$y_{F,F} = \text{id}$
$y_{F,G} = \text{id} \circ y_{F,F}$
$y_{F,K} = \text{id} \circ y_{F,F}$
$y_{F,H} = 0 \circ y_{F,G}$
$y_{F,Q} = y_{F,J} \circ y_{F,H}$
$y_{F,I} = 1 \circ y_{F,Q}$
$y_{F,J} = y_{F,I} \sqcup y_{F,K}$
Example: Computing $\phi_{r_p,N}'s \ (y_{r_p,N}^* \ to \ be \ precise)$ using Kildall’s algo

\[
\begin{align*}
    y_{A,A} &= id \\
    y_{A,B} &= 0 \circ y_{A,A} \\
    y_{A,C} &= 1 \circ y_{A,B} \\
    y_{A,P} &= y_F, J \circ y_{A,C} \\
    y_{A,D} &= 1 \circ y_{A,P} \\
    y_{A,E} &= id \circ y_{A,D} \\
    y_{F,F} &= id \\
    y_{F,G} &= id \circ y_{F,F} \\
    y_{F,K} &= id \circ y_{F,F} \\
    y_{F,H} &= 0 \circ y_{F,G} \\
    y_{F,Q} &= y_F, J \circ y_{F,H} \\
    y_{F,I} &= 1 \circ y_{F,Q} \\
    y_{F,J} &= y_F, I \sqcup y_F, K
\end{align*}
\]
Example: Computing $\phi_{r_p,N}$’s ($y^*_{r_p,N}$ to be precise) using Kildall’s algo

\[
\begin{align*}
y_{A,A} &= id \\
y_{A,B} &= 0 \circ y_{A,A} \\
y_{A,C} &= 1 \circ y_{A,B} \\
y_{A,P} &= y_{F,J} \circ y_{A,C} \\
y_{A,D} &= 1 \circ y_{A,P} \\
y_{A,E} &= id \circ y_{A,D} \\
y_{F,F} &= id \\
y_{F,G} &= id \circ y_{F,F} \\
y_{F,K} &= id \circ y_{F,F} \\
y_{F,H} &= 0 \circ y_{F,G} \\
y_{F,Q} &= y_{F,J} \circ y_{F,H} \\
y_{F,I} &= 1 \circ y_{F,Q} \\
y_{F,J} &= y_{F,I} \sqcup y_{F,K}
\end{align*}
\]
Example: Computing $\phi_{r_p,N}$'s ($y^*_{r_p,N}$ to be precise) using Kildall's algo

\[
\begin{align*}
y_{A,A} &= id \\
y_{A,B} &= 0 \circ y_{A,A} \\
y_{A,C} &= 1 \circ y_{A,B} \\
y_{A,P} &= y_{F,J} \circ y_{A,C} \\
y_{A,D} &= 1 \circ y_{A,P} \\
y_{A,E} &= id \circ y_{A,D} \\
y_{F,F} &= id \\
y_{F,G} &= id \circ y_{F,F} \\
y_{F,K} &= id \circ y_{F,F} \\
y_{F,H} &= 0 \circ y_{F,G} \\
y_{F,Q} &= y_{F,J} \circ y_{F,H} \\
y_{F,I} &= 1 \circ y_{F,Q} \\
y_{F,J} &= y_{F,I} \sqcup y_{F,K}
\end{align*}
\]
Example: Computing $\phi_{r_p,N}$’s ($y_{r_p,N}^*$ to be precise) using Kildall’s algo

\[
\begin{align*}
y_{A,A} &= id \\
y_{A,B} &= 0 \circ y_{A,A} \\
y_{A,C} &= 1 \circ y_{A,B} \\
y_{A,P} &= y_{F,J} \circ y_{A,C} \\
y_{A,D} &= 1 \circ y_{A,P} \\
y_{A,E} &= id \circ y_{A,D} \\
y_{F,F} &= id \\
y_{F,G} &= id \circ y_{F,F} \\
y_{F,K} &= id \circ y_{F,F} \\
y_{F,H} &= 0 \circ y_{F,G} \\
y_{F,Q} &= y_{F,J} \circ y_{F,H} \\
y_{F,I} &= 1 \circ y_{F,Q} \\
y_{F,J} &= y_{F,I} \sqcup y_{F,K} \\
\end{align*}
\]
Example: Computing $\phi_{r_p,N}$'s ($y_{r_p,N}^*$ to be precise) using Kildall's algo

$y_{A,A} = id$
$y_{A,B} = 0 \circ y_{A,A}$
$y_{A,C} = 1 \circ y_{A,B}$
$y_{A,P} = y_{F,J} \circ y_{A,C}$
$y_{A,D} = 1 \circ y_{A,P}$
$y_{A,E} = id \circ y_{A,D}$

$y_{F,F} = id$
$y_{F,G} = id \circ y_{F,F}$
$y_{F,K} = id \circ y_{F,F}$
$y_{F,H} = 0 \circ y_{F,G}$
$y_{F,Q} = y_{F,J} \circ y_{F,H}$
$y_{F,I} = 1 \circ y_{F,Q}$
$y_{F,J} = y_{F,I} \sqcup y_{F,K}$
Example: Computing $\phi_{r_p,N}$’s ($y^*_{r_p,N}$ to be precise) using Kildall’s algo

\[
\begin{align*}
y_{A,A} &= id \\
y_{A,B} &= 0 \circ y_{A,A} \\
y_{A,C} &= 1 \circ y_{A,B} \\
y_{A,P} &= y_{F,J} \circ y_{A,C} \\
y_{A,D} &= 1 \circ y_{A,P} \\
y_{A,E} &= id \circ y_{A,D} \\
y_{F,F} &= id \\
y_{F,G} &= id \circ y_{F,F} \\
y_{F,K} &= id \circ y_{F,F} \\
y_{F,H} &= 0 \circ y_{F,G} \\
y_{F,Q} &= y_{F,J} \circ y_{F,H} \\
y_{F,I} &= 1 \circ y_{F,Q} \\
y_{F,J} &= y_{F,I} \sqcup y_{F,K}
\end{align*}
\]
Example: Computing $\phi_{r^*_p,N}$’s ($y^*_r_{r^*_p,N}$ to be precise) using Kildall’s algo

$$y_{A,A} = id$$
$$y_{A,B} = 0 \circ y_{A,A}$$
$$y_{A,C} = 1 \circ y_{A,B}$$
$$y_{A,P} = y_F,J \circ y_{A,C}$$
$$y_{A,D} = 1 \circ y_{A,P}$$
$$y_{A,E} = id \circ y_{A,D}$$

$$y_{F,F} = id$$
$$y_{F,G} = id \circ y_{F,F}$$
$$y_{F,K} = id \circ y_{F,F}$$
$$y_{F,H} = 0 \circ y_{F,G}$$
$$y_{F,Q} = y_F,J \circ y_{F,H}$$
$$y_{F,I} = 1 \circ y_{F,Q}$$
$$y_{F,J} = y_{F,I} \sqcup y_{F,K}$$
Example: Computing $\phi_{r_p,N}$’s ($y_{r_p,N}^*$, to be precise) using Kildall’s algo

\[
\begin{align*}
  y_{A,A} &= id \\
  y_{A,B} &= 0 \circ y_{A,A} \\
  y_{A,C} &= 1 \circ y_{A,B} \\
  y_{A,P} &= y_F,J \circ y_{A,C} \\
  y_{A,D} &= 1 \circ y_{A,P} \\
  y_{A,E} &= id \circ y_{A,D} \\
  y_{F,F} &= id \\
  y_{F,G} &= id \circ y_{F,F} \\
  y_{F,K} &= id \circ y_{F,F} \\
  y_{F,H} &= 0 \circ y_{F,G} \\
  y_{F,Q} &= y_F,J \circ y_{F,H} \\
  y_{F,I} &= 1 \circ y_{F,Q} \\
  y_{F,J} &= y_F,I \sqcup y_{F,K} \\
\end{align*}
\]
Example: Computing JVP values (x^*_N’s to be precise)

\[
\begin{align*}
\times_A &= 0 \\
\times_B &= 0(x_A) \\
\times_C &= 1(x_A) \\
\times_P &= 1(x_A) \\
\times_D &= 1(x_A) \\
\times_E &= 1(x_A) \\
\times_F &= x_C \sqcup x_H \\
\times_G &= id(x_F) \\
\times_K &= id(x_F) \\
\times_H &= 0(x_F) \\
\times_Q &= 0(x_F) \\
\times_I &= 1(x_F) \\
\times_J &= id(x_F).
\end{align*}
\]
Example: Computing JVP values ($x^*_N$’s to be precise)

\[
\begin{align*}
    x_A &= 0, \\
    x_B &= 0(x_A), \\
    x_C &= 1(x_A), \\
    x_P &= 1(x_A), \\
    x_D &= 1(x_A), \\
    x_E &= 1(x_A) ,
\end{align*}
\]

\[
\begin{align*}
    x_F &= x_C \sqcup x_H, \\
    x_G &= id(x_F), \\
    x_K &= id(x_F), \\
    x_H &= 0(x_F), \\
    x_Q &= 0(x_F), \\
    x_I &= 1(x_F), \\
    x_J &= id(x_F).
\end{align*}
\]

Fig shows initial (red) and final (blue) values.
Example: Computing JVP values ($x^*_N$’s to be precise)

\[
\begin{align*}
x_A &= 0 \\
x_B &= 0(x_A) \\
x_C &= 1(x_A) \\
x_P &= 1(x_A) \\
x_D &= 1(x_A) \\
x_E &= 1(x_A) \\
x_F &= x_C \sqcup x_H \\
x_G &= id(x_F) \\
x_K &= id(x_F) \\
x_H &= 0(x_F) \\
x_Q &= 0(x_F) \\
x_I &= 1(x_F) \\
x_J &= id(x_F).
\end{align*}
\]

Fig shows initial (red) and final (blue) values.
Summary of functional approach

- Uses a two step approach
  1. Compute $\phi_{r_p, N}$’s.
  2. Compute $x_n$’s (JVP’s) at each point.

Summary of conditions: For each property (column heading), the conjunction of the ticked conditions (row headings) are sufficient to ensure the property.

<table>
<thead>
<tr>
<th></th>
<th>Termination</th>
<th>Least Sol of Eq(2) $\geq$ JVP</th>
<th>Least Sol of Eq(2) = JVP</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_{MN}$’s monotonic</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Finite underlying lattice</td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f_{MN}$’s distributive</td>
<td></td>
<td></td>
<td>√</td>
</tr>
</tbody>
</table>
Viewing $\phi$ computation as a table

```
read a,b

\( t := a \times b \)

\( \text{call p} \)

\( \text{print } t \)
```

```
\( a == 0 \)

\( a := a - 1 \)

\( \text{call p} \)

\( t := a \times b \)

\( \text{ret} \)
```
Viewing $\phi$ computation as a table
Viewing $\phi$ computation as a table
Viewing $\phi$ computation as a table

```
read a, b

A: 0 1

B: 0 0

C: 1 1

call p

D

print t

E

t := a * b

F

G

H: 0 0

I

J: 0 1

K: 0 1

L

M

N

O

P

Q

a := a - 1

a == 0

ret
```
Viewing $\phi$ computation as a table
Iterative/Tabulation Approach

- Main idea: de-couple the propagation of function rows.
- Maintain a table of values representing the current value of $\phi_{r_p,N}$ for each program point $N$ in procedure $p$.
- Expand column for data value $d$ in procedure $p$ only if $d$ is reachable at $r_p$.
- Informally, at $N$ in procedure $p$, the table has an entry $d \mapsto d'$ if we have seen
  1. valid paths $\rho$ from $r_1$ to $r_p$ with $\bigcup_{\rho} f_\rho(d_0) = d$, and
  2. valid and complete paths $\delta$ from $r_p$ to $N$ with $\bigcup_{\delta} f_\delta(d) = d'$.
Iterative/Tabulation Approach

- Apply Kildall’s algo with initial value of $d_0 \mapsto d_0$ at $r_1$.
- Propagating across a call to procedure $p$: value $d$ is propagated to the column for $d$ at root of $p$.
- Propagating across return nodes from procedure $p$: value $d'$ in column for $d$ is propagated to each column at a return site of a call to procedure $p$ that has the value $d$ in the preceding entry.
Example: Computing $\phi$'s iteratively: 1
Example: Computing $\phi$’s iteratively: 2

1. $t := a \times b$
2. $t := a \times b$
3. $a := a - 1$
4. $a := a - 1$
5. $t := a \times b$
6. $t := a \times b$
7. $a == 0$
8. $a == 0$
Example: Computing $\phi$’s iteratively: 3

read $a, b$

$t := a \times b$

$a := a - 1$

call $p$

$t := a \times b$

print $t$

$a == 0$

call $p$

$t := a \times b$

ret
Example: Computing $\phi$'s iteratively: 4
Example: Computing $\phi$’s iteratively: 5

```
read a, b

\[ t := a \times b \]

\[ a := a - 1 \]

\[ \text{call p} \]

\[ t := a \times b \]

\[ \text{print t} \]
```

```
\[ a == 0 \]

\[ a := a - 1 \]

\[ \text{call p} \]

\[ t := a \times b \]

\[ \text{ret} \]
```
Example: Computing $\phi$'s iteratively: 6

```
read a, b

A := a-1
B := a-1

C := a*b
D := a*b

E := a*b
F := a*b

G := a*b
H := a*b
```

```
print t
J := a*b
K := a*b
```

```
L := a*b
M := a*b
N := a*b
```

```
O := a*b
P := a*b
```

```
ret
```

Example: Computing $\phi$'s iteratively: 7

```
read a, b
read a, b

\begin{align*}
t &:= a \times b \\
a &:= a - 1 \\
t &:= a \times b \\
\end{align*}

print t
```

```
\begin{align*}
a &== 0 \\
a &:= a - 1 \\
call p \\
call p \\
t &:= a \times b \\
ret \\
\end{align*}
```
Example: Computing $\phi$’s iteratively: 8

read $a$, $b$

$t := a \times b$

$P \cdot 1$ -

call $p$

$P \cdot 1$ -

$t := a \times b$

$D \cdot 1$ -

print $t$

$E \cdot 1$ -

$a == 0$

$G \cdot -1$

$a := a - 1$

$H \cdot -0$

call $p$

$Q$

$t := a \times b$

$I \cdot -1$

ret

$J \cdot -1$

$K \cdot -1$

$F \cdot 0$

$0$

$1$

$L$

$O$

$M$

$N$
Example: Computing $\phi$’s iteratively: 9

```
read a, b

F
G
H

A 0
B 0
C 1
D 1
E 1

P 1
Q
R
S
T
U
V
W
X
Y
Z

a := a-1

a == 0

F 0 1
G 0 1
H 0 0
I 0 1
```
Example: Computing $\phi$’s iteratively: 10

```
read a, b
A

read a, b
B

read a, b
C

read a, b
D

 call p
P

 call p
Q

 call p
R

 call p
S

B == 0
T

0

6

ret

...}
```
Example: Computing $\phi$’s iteratively: 11

```
read a, b

\[ t := a \times b \]

\[ a := a - 1 \]

call p

\[ t := a \times b \]

print t
```

```
\textbf{Functional Approach}

\textbf{Example}

\textbf{Iterative Approach}

\textbf{Exercises}

\textbf{Diagram}

```
read a, b

\[ t := a \times b \]

\[ a := a - 1 \]

call p

\[ t := a \times b \]

print t
```

```
\textbf{Diagram}

```
read a, b

\[ t := a \times b \]

\[ a := a - 1 \]

call p

\[ t := a \times b \]

print t
```

```
\textbf{Diagram}

```
read a, b

\[ t := a \times b \]

\[ a := a - 1 \]

call p

\[ t := a \times b \]

print t
```
Example: Computing \( \phi \)'s iteratively: 12
Example: Computing $\phi$’s iteratively: 13

\[ a := a^{-1} \]
\[ t := a \cdot b \]
\[ \text{read } a, b \]
\[ \text{print } t \]
\[ \text{call } p \]
\[ a == 0 \]
\[ \text{ret} \]
Example: Finally compute $x_N$'s from $\phi$ values

At each point $N$ take join of reachable $\phi_{r_p,N}$ values.
Correctness of iterative algo

- Iterative algo terminates provided underlying lattice is finite.
- It computes the $y^*_{r_p,N}$’s (where $y^*_{r_p,N}$’s are the least solution to Eq (1)) “partially”: If it maps $d$ to $d' \neq \perp$ then $y^*_{r_p,N}(d) = d'$.
- The JVP values it gives (say $z_N$’s) are such that

$$\text{JVP}_N \leq z_N \leq x^*_N$$

(where $x^*_N$’s are the solution to Eq (2’)).
- If underlying transfer functions are distributive it computes $\phi_{r_p,N}$’s correctly (though partially), and the JVP values correctly.
- It thus computes an overapproximation of JVP for monotonic transfer functions, and exact JVP when transfer functions are distributive.
Exercise 1: Iterative algo

Run the iterative algo to do constant propagation analysis for the program below with initial value $\emptyset$. Assume here that “cond” is the condition “$a \leq 2$”.

```
a := a+1
a := 0
print a
call p
cond
a := a+1
call p
a := a-1
ret
```
Exercise 2: Functional vs Iterative algo

Run the functional and iterative algos to do constant propagation analysis for the program below with initial value ∅:

```
a := a+1
7
F
G
a := 0
D
call p
11
call p
cond
A
B
O
L
M
N
2
3
4
6
9
8
ret
a := a-1
10
I
J
K
C
P
H
```

```
a := 0
B

a := a+1
2
a := a+1
3

call p

print a
4

call p

cond
5

a := a+1
6

G

a := a+1
7

H

a := a+1
8

P

a := a-1
9

I

ret
10

J

11
```
Comparing functional vs iterative approach

- Functional algo can terminate even when underlying lattice is infinite, provided we can represent and compose/join functions “symbolically”.

- Iterative is typically more efficient than functional since it only computes $\phi_{r_p,N}$’s for values reachable at start of procedure.