

Interference Graphs for Programs in Static Single Information Form are Interval Graphs

Philip Brisk

Processor Architecture
Laboratory (LAP)

EPFL

Lausanne, Switzerland



Majid Sarrafzadeh

Embedded and Reconfigurable
Systems (ER) Laboratory

UCLA

Los Angeles, California, USA



Outline

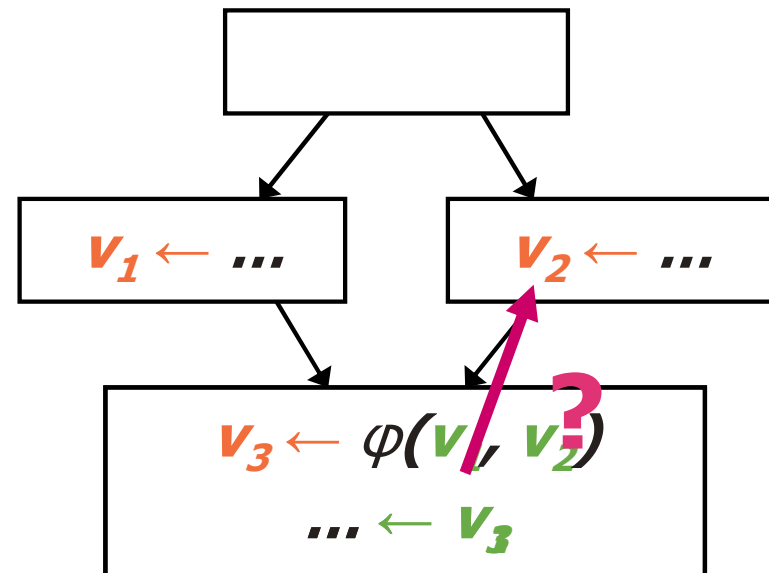
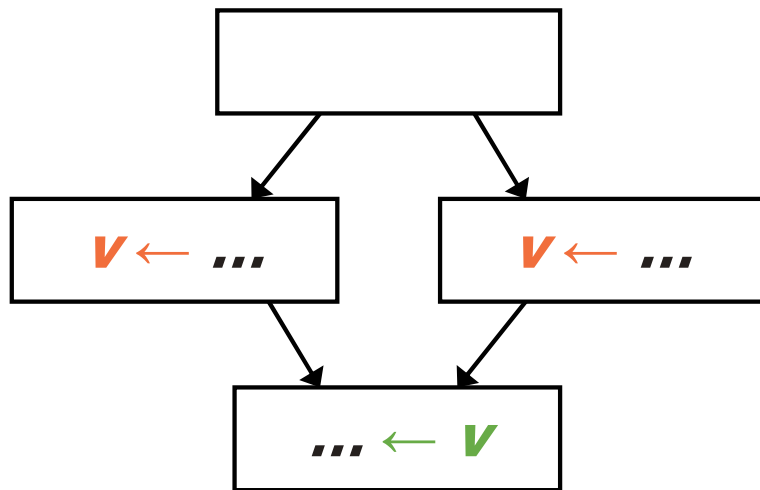
- Contributions
- Introduction to SSA/SSI Form
- Register Allocation Overview
- Interval Graphs
 - Interference Graphs in SSI Form are Interval Graphs
- Liveness Analysis
 - Converges in 1-Iteration for SSI Form
- Conclusion

Contributions

- Theorem 1:
 - Interference Graphs in SSI Form Are Interval Graphs
 - Weighted k-Colorable Subgraph Problem
 - Poly. Time Solution
 - Linear Scan Register Allocation in SSI
 - No Lifetime Holes
- Theorem 2:
 - Liveness Analysis in SSI Converges in 1 Iteration

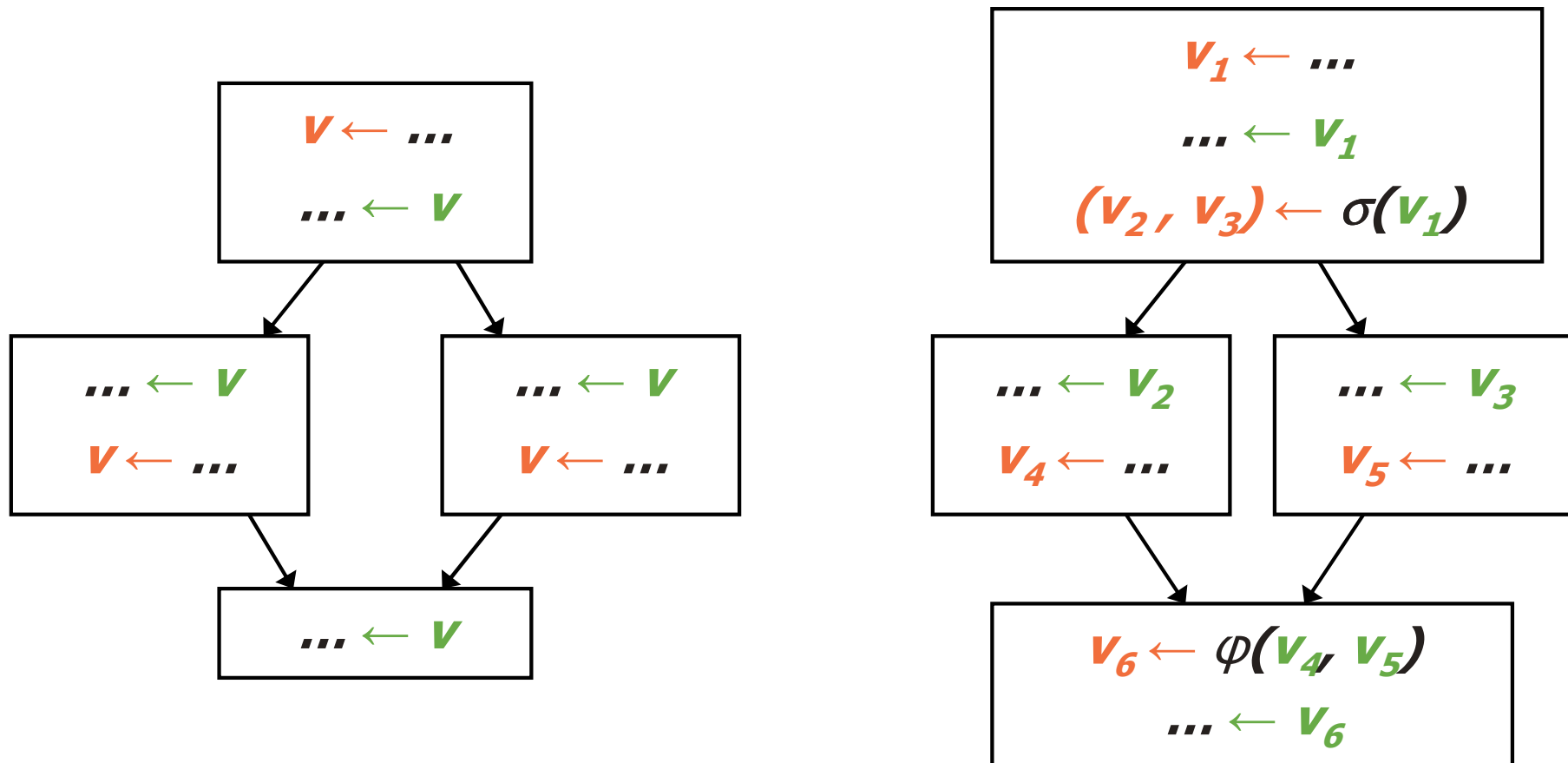
Static Single Assignment Form

- In SSA Form:
 1. Each Variable is Defined **Once**
 - Rename v at each Definition ($v_1 \leftarrow \dots, v_2 \leftarrow \dots$, etc.)
 2. Each Use of a Variable Corresponds to One Definition
 - Insert ϕ -functions at Joint Points to Merge Definitions



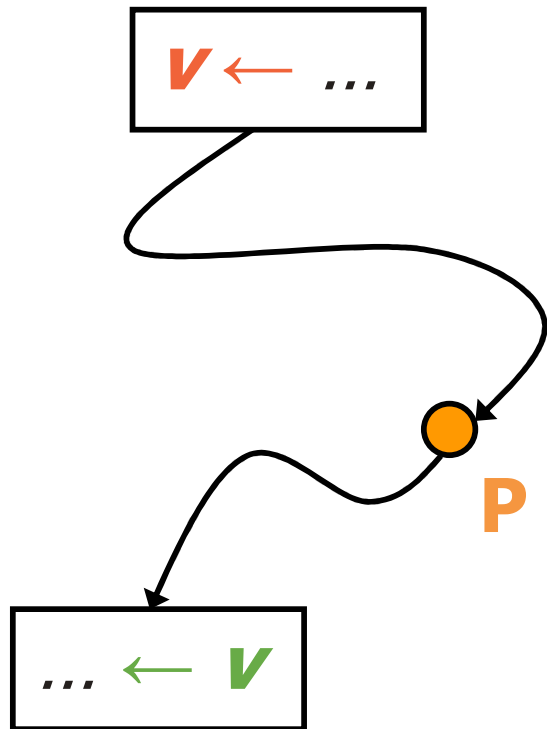
Static Single Information Form

- Extension of SSA Form:
 - Insert σ -Functions at Split Points



Liveness

Definition of Variable v



Use of Variable v

v is **LIVE** at **P** if there is:

1. A Path from a Definition of v to **P**
2. A Path from **P** to a Use of v

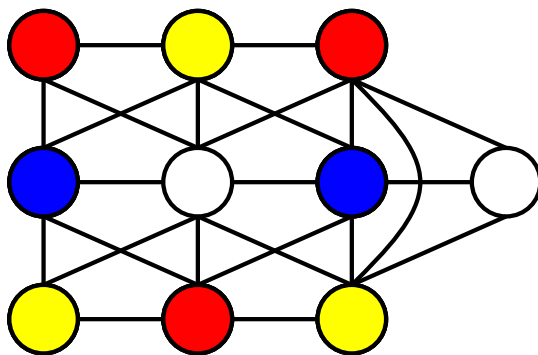
Two Variables **Interfere** if their Live Ranges Overlap

Register Allocation

- Spilling
 - Partition All Variables Between Registers and Memory at each Point in the Program
 - Goal: Minimize Dynamically Executed Loads and Stores
- Register Assignment
 - Assign non-Spilled Variables to Registers
 - Goal: Eliminate as Many Copies $y \leftarrow x$ by Assigning x and y to the same Register

Spilling

- k Colorable Subgraph Problem
 - Vertices Correspond to Variables
 - Edges are Placed Between Interfering Variables
- k : Number of Registers
 - k -Colorable Subgraph : Variables in Registers
 - All Others : Spilled to Memory



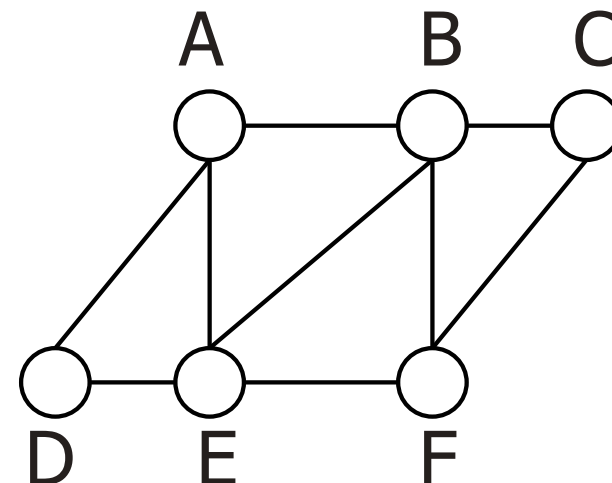
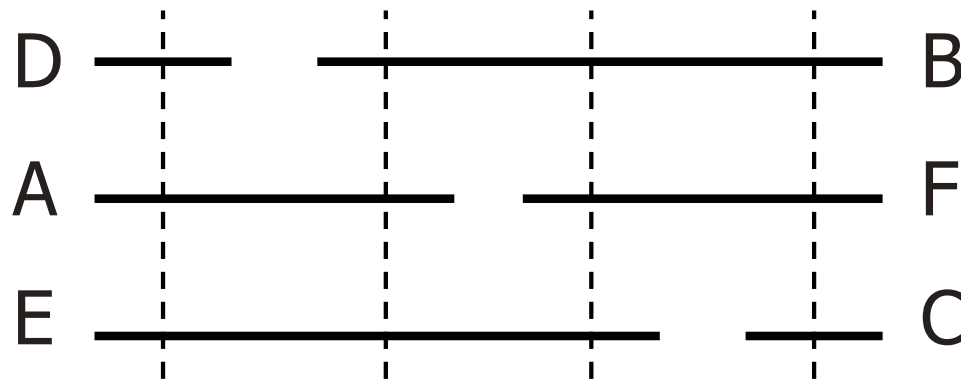
$k = 3$

● Register

○ Memory

Interval Graphs

- SSI Form – Interference Graphs are Interval Graphs
 - Intersection Graph of Intervals on a Line
 - All Max Cliques Containing Each Variable Can be Ordered Consecutively
 - (Weighted) k-Colorable Subgraph Problem Has Polynomial Solution [Yannakakis and Gavril, IPL, 1987]

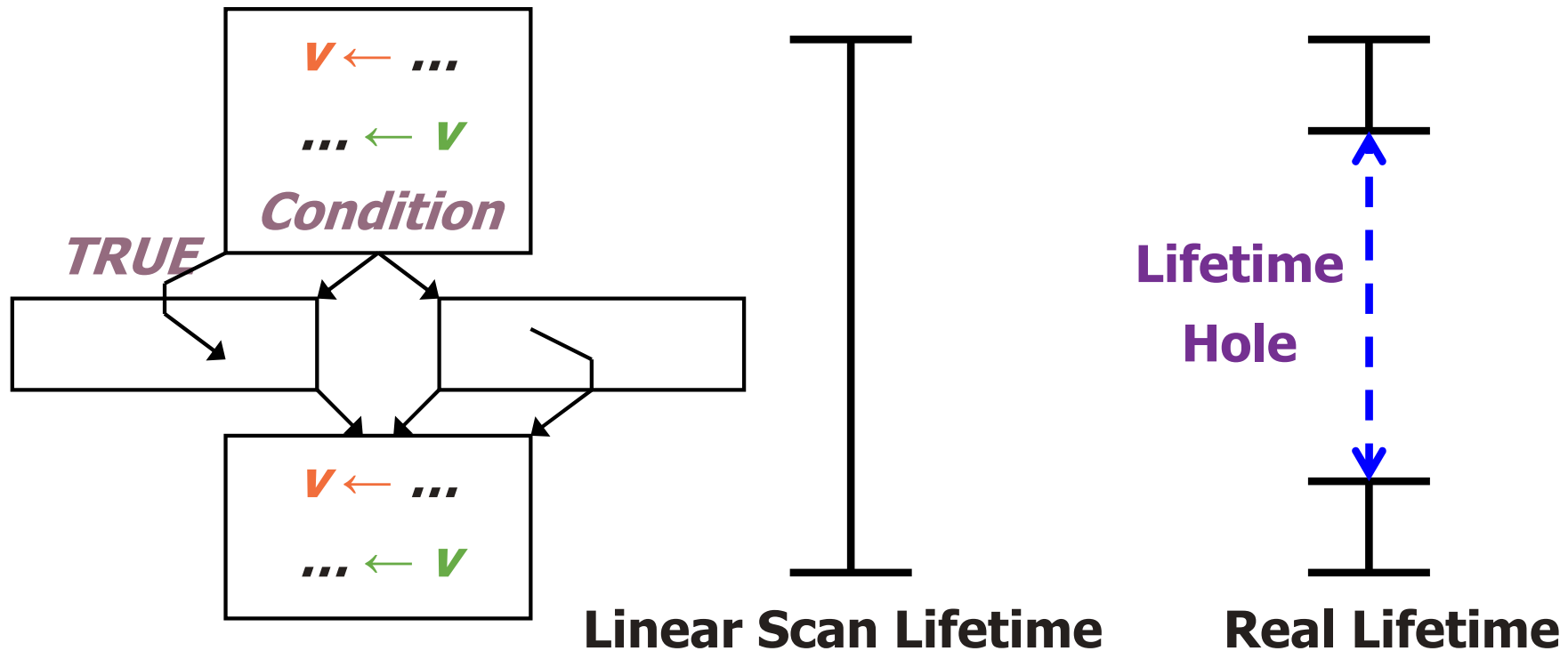


Related Work

- SSA Form: Interference Graphs are Chordal
 - $O(V+E)$ Time Algorithm for Min. Coloring
 - k -Colorable Subgraph Problem is:
 - Polynomial if k is Constant
 - $O(n^k)$ Algorithm
 - NP-Complete if k is Unbounded
 - All Interval Graphs are Chordal

Linear Scan Register Allocation

- Convert Program to a Linear List of Instructions
- Represent Each Variable v as an Interval
 - Lifetime Holes – v is not Live
 - Spill/Compute Interval Coloring w/o Building Graph



Dominance

- Let p, p' be Points in a Program
- Let r (t) be the Entry (Exit) Point of the Procedure
- p *Dominates* p' ($p \text{ dom } p'$) if Every Path from r to p' Goes Through p
- p *Strictly Dominates* p'

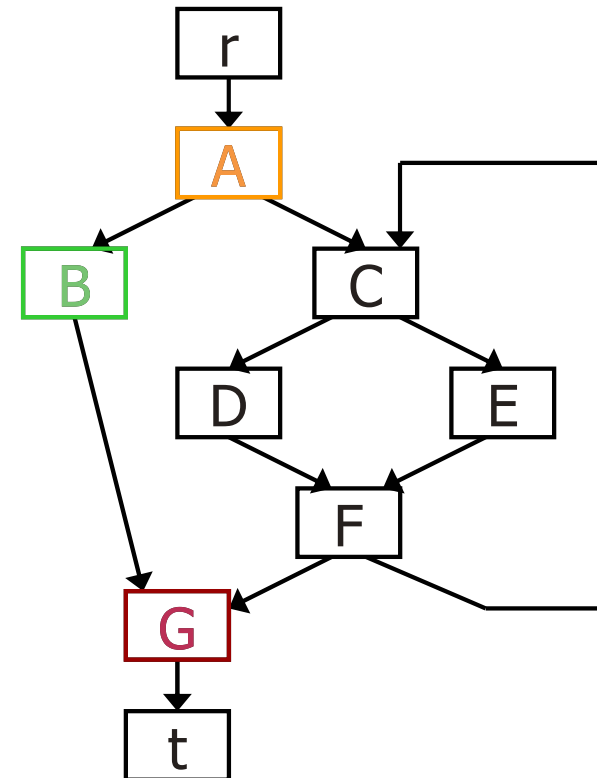
($p \text{ sdom } p'$) if:

1. $p \text{ sdom } p'$
2. $p \neq p'$

Example:

$A \text{ sdom } G$

$\neg B \text{ sdom } G$



Post-Dominance

- Let p, p' be Points in a Program
- Let r (t) be the Entry (Exit) Point of the Procedure
- p *Post-Dominates* p' ($p \text{ pdom } p'$) if Every Path from p' to t Goes Through p

- p *Strictly Post-Dominates* p'

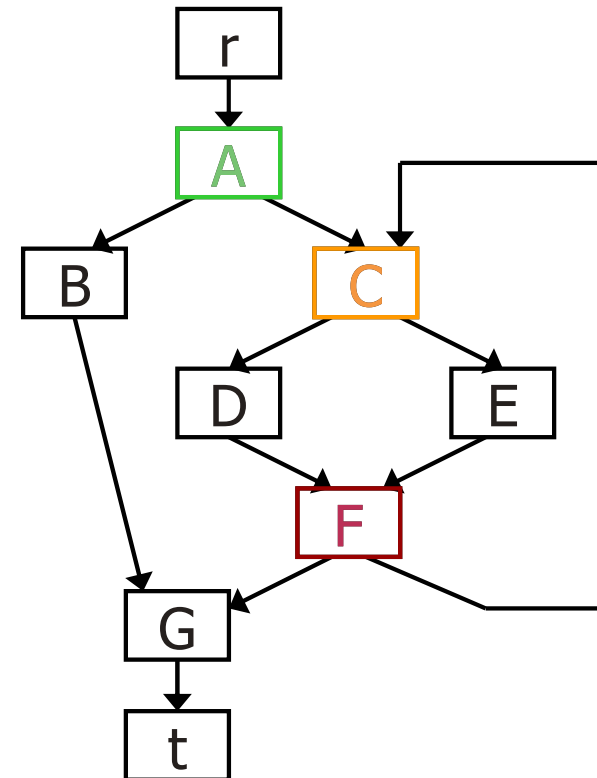
($p \text{ spdom } p'$) if

1. $p \text{ sdom } p'$
2. $p \neq p'$

Example:

$F \text{ spdom } C$

$\neg F \text{ spdom } A$



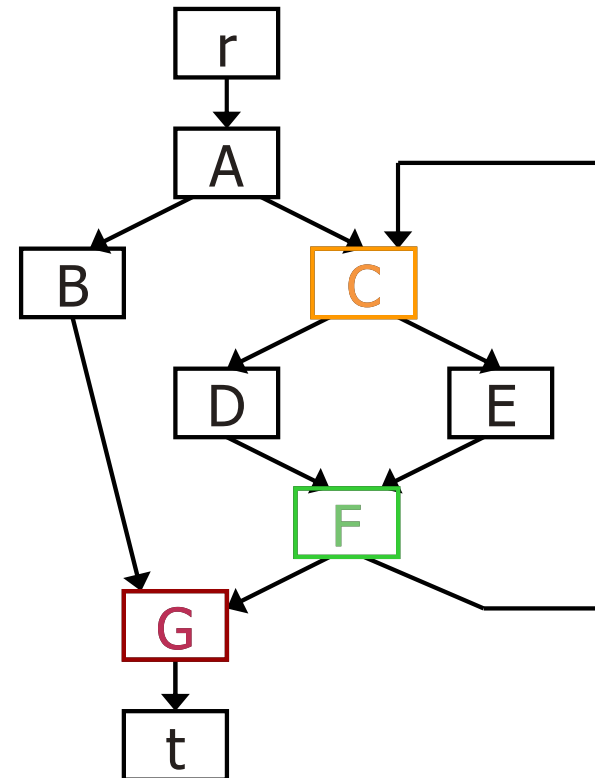
Immediate Post-Dominance

- The *Immediate Post-Dominator* of p' ($\text{ipdom } p'$) is a node p such that:
 1. $p \text{ spdom } p'$
 2. There is no node p'' s.t.
 $p \text{ spdom } p'' \text{ spdom } p'$

Example:

$G \text{ spdom } C$

$F = \text{ipdom } G$



SSI Form and (Post-)Dominance

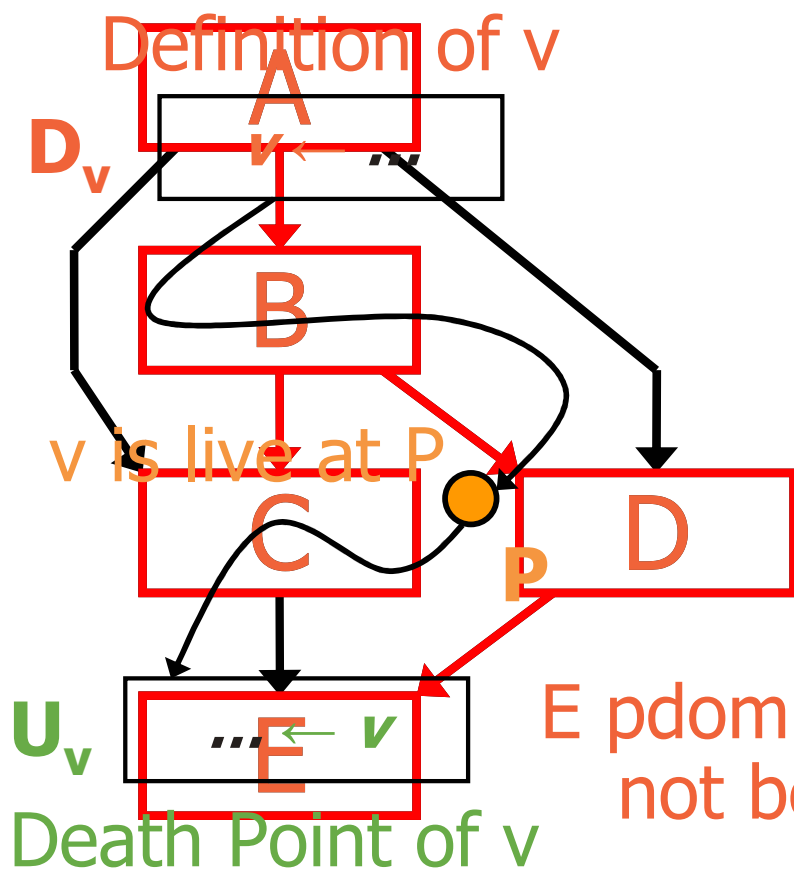
- SSA: The Def. of Each Var. Dominates Each Use

Theorem [Ananian, MS Thesis, MIT, 1999]:

- SSI: Each Use of Each Var. Post-Dominates its Def.
- Corollaries:
 - Def. and Uses of Each Var. Form a Total Order
 - If v is Live at Point p :
 1. Def. of v dom p
 2. Last Use (Death Point) of v pdom p

Post-Dominated Depth-First Search

- DFS Through the CFG of the Program
 - Process Node n Only After Each Node m , such that n pdom m is Processed



A, B, C, D, E

D_v dom P
 A, B, D, C, E
 D_v Processed Before P
 in any DFS D, E

A, C, D, B, E

U_v pdom P
 A, D, B, C, E
 E pdom D , but D has
 not been processed in any DFS
 Processed Before U_v
 in any DFS

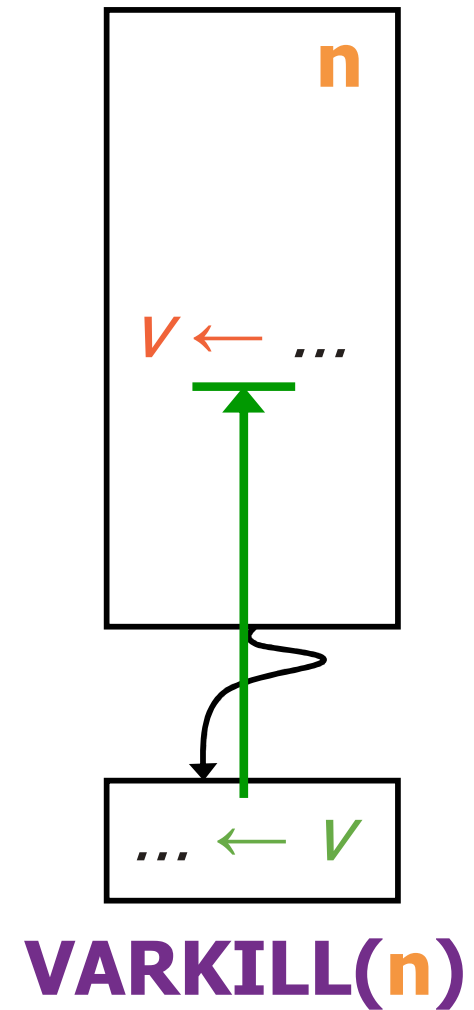
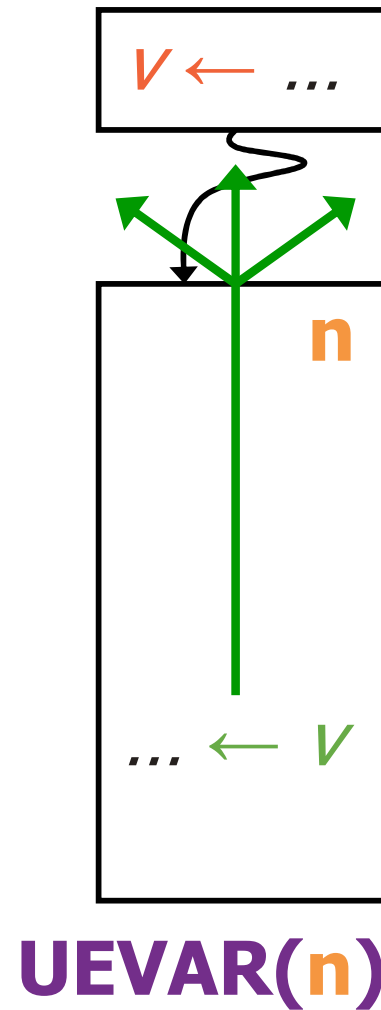
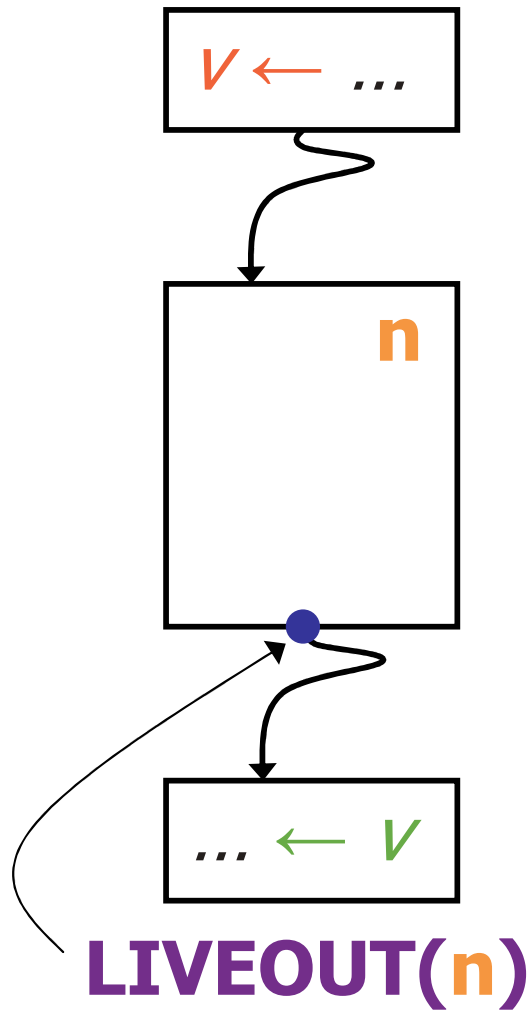
PD-DFS and SSI Form

- Interference Graph is an Interval Graph
 - Intervals Defined by PD-DFS Order of CFG
 - Graph Coloring
 - Solve k-Colorable Subgraph Problem
 - Linear Scan
 - Place Basic Blocks in CFG in PD-DFS Order
 - Treat CFG as Linear List of Operations
 - No Lifetime Holes
 - No Need for 2nd Chance Binpacking
- [Traub, Holloway, and Smith, PLDI 1998]

Liveness Analysis

- Compute Live Variables at Each Program Point
- 72% of Runtime of Briggs' Register Allocator
[Cooper and Dasgupta, CGO 2006]
- Skip Liveness Analysis → Poor Quality Code
[Poletto & Sarkar, TOPLAS 1999]
- SSI Form – Converge in 1-Iteration

Liveness Analysis



Liveness Analysis

Initialize **LIVEOUT(n)** to Empty for All Blocks **n**

Repeat {

For each Block **n**

$$\mathbf{LIVEOUT(n)} = \mathbf{LIVEOUT(n)} \cup$$

$$\bigcup_{m \in succ(n)} \mathbf{UEVAR(m)} \cup (\mathbf{LIVEOUT(m)} \cap \overline{\mathbf{VARKILL(m)}})$$

}

While **LIVEOUT(n)** Changes for at Least One Block **n**

Liveness Analysis in SSI

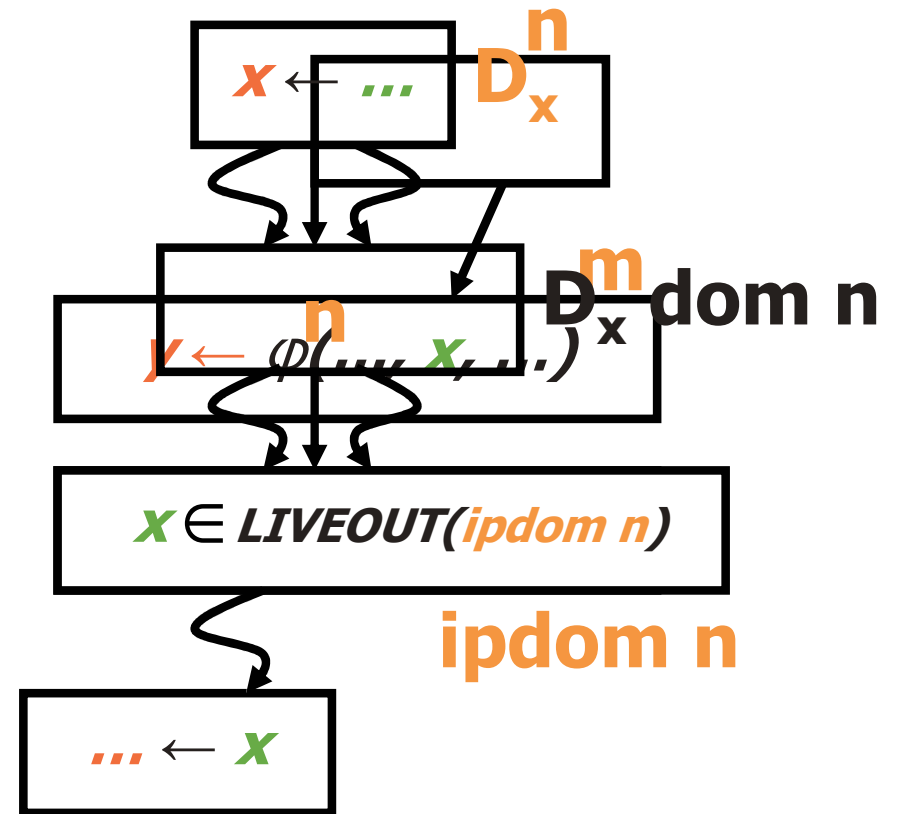
For each Basic Block n , taken in Reverse PD \Rightarrow ES Order:

LIVEOUT(n) =

$$\bigcup_{m \in \text{succ}(n)} \varphi\text{-Params}(n, m)$$

$$\cup \text{UEVAR}(\text{ipdom } n)$$

$$\cup (\text{LIVEOUT}(\text{ipdom } n) \cap \text{VARKILL}(\text{ipdom } n))$$



Liveness Analysis in SSI

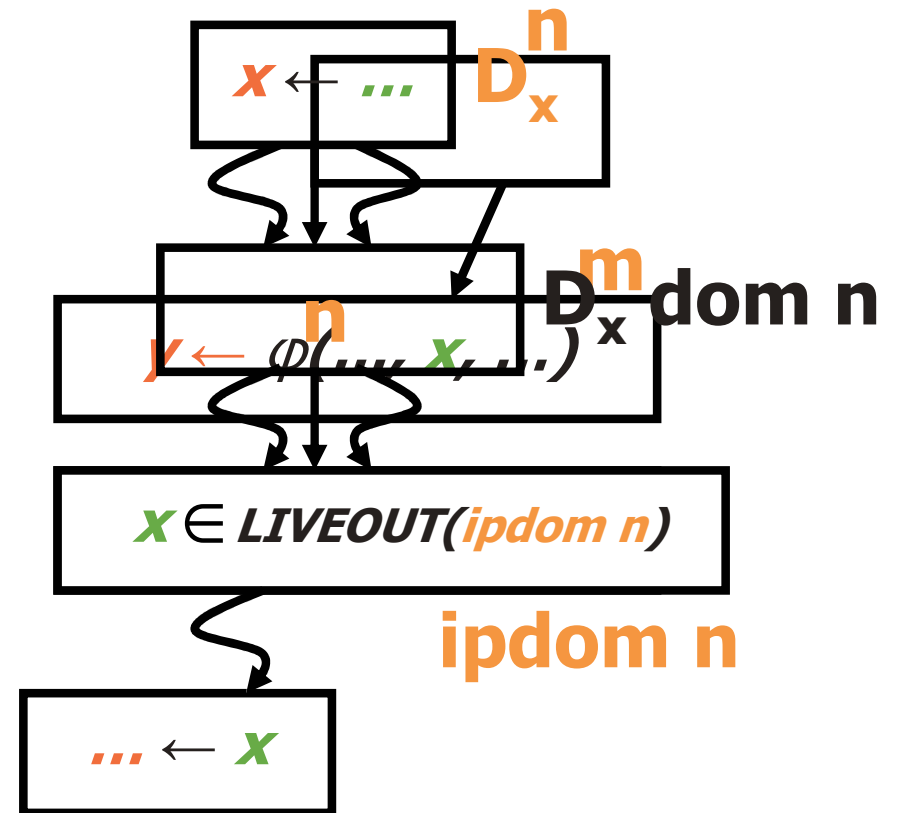
For each Basic Block n , taken in Reverse PD \rightarrow ES Order:

LIVEOUT(n) =

$$\bigcup_{m \in \text{succ}(n)} \varphi\text{-Params}(n, m)$$

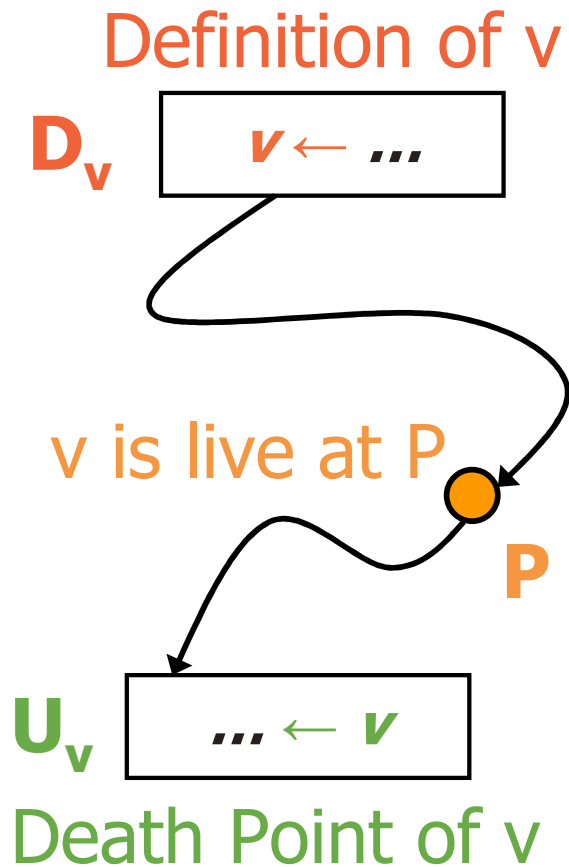
$$\cup \text{UEVAR}(\text{ipdom } n)$$

$$\cup (\text{LIVEOUT}(\text{ipdom } n) \cap \text{VARKILL}(\text{ipdom } n))$$



Correctness Argument

- Process CFG Nodes in Reverse PD DFS Order



$D_v \text{ dom } P$

P Processed Before D_v

Nodes that Dominate P Processed
in Reverse Order from P

$U_v \text{ pdom } P$

U_v Processed Before P

Nodes that Post Dominate P
Processed in Reverse Order
Toward P

Conclusion

- In SSI Form
 - Interference Graphs are Interval Graphs
 - Solve k-Colorable Subgraph Problem in Poly. Time
 - Linear Scan Without Lifetime Holes
 - Liveness Analysis Converges in 1 Iteration
 - Must Use Reverse PD-DFS Order for Basic Blocks
 - Propagate Liveness Info. from ipdom rather than Successor Nodes
- Future Work
 - Implement the Ideas Presented Here
 - Compare to Existing Register Allocation Methods

Questions, Comments, or...

ATTACK!

Thank You for Attending

Liveness Analysis

Complexity:

N – Number of Basic Blocks in the Program

V – Number of Variables

Each **LIVEOUT** Set is a Bit Vector of Size **V**

Space Complexity: $\Theta(NV)$

Time Complexity: $\Theta(NV)$ per Iteration

Altogether: $\Omega(NV)$ to $O(N^4V)$