

IS593: Language-based Security

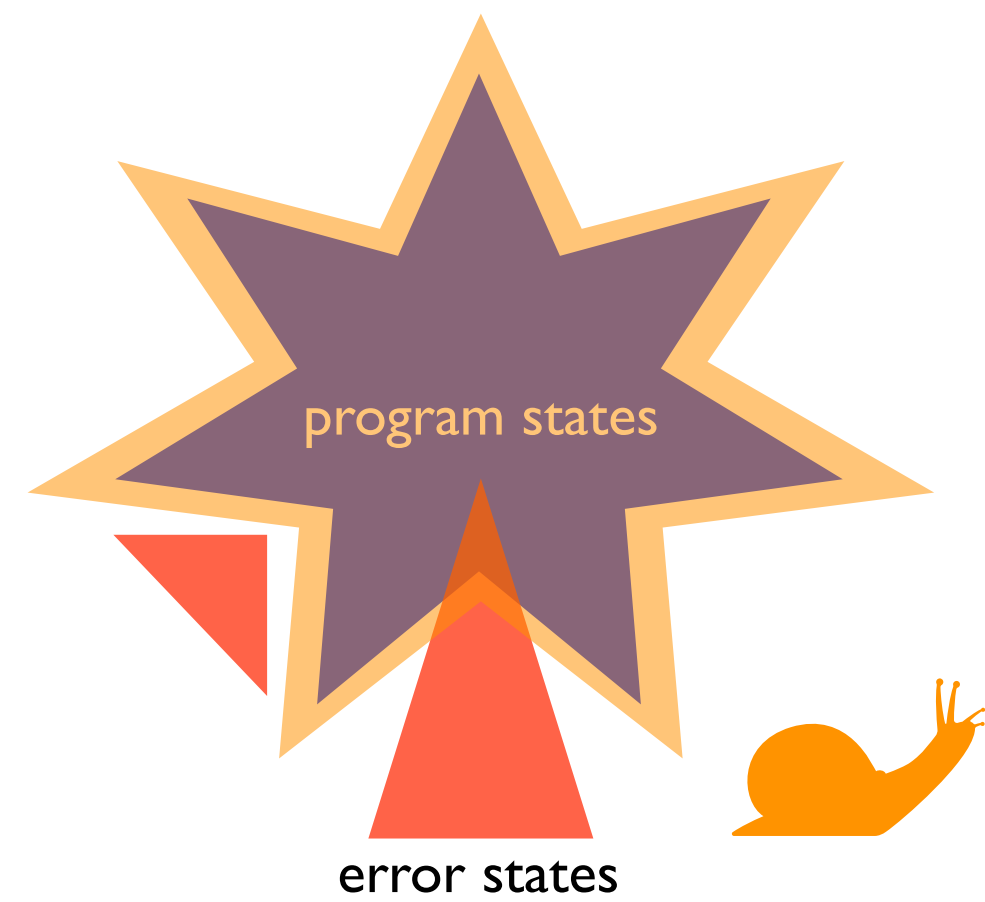
9. Sparse Analysis

Kihong Heo



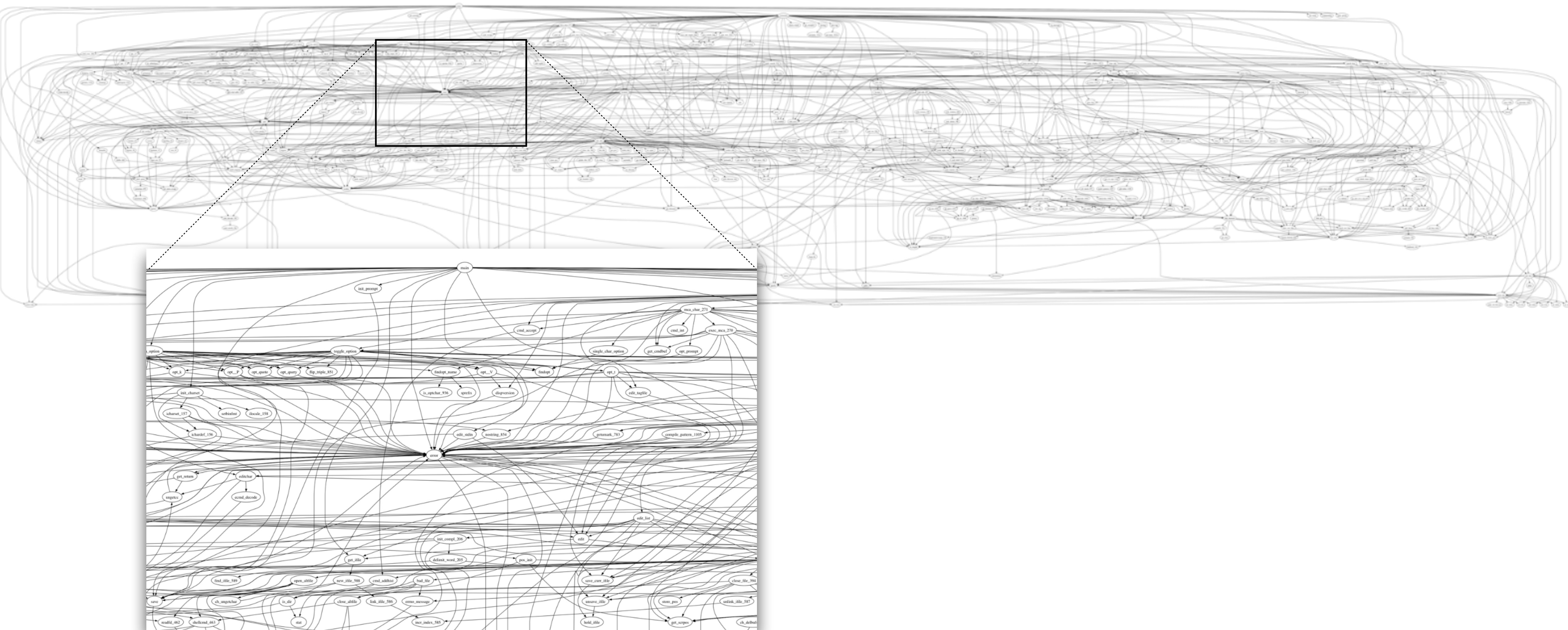
Cost Reduction Techniques

- How to **reduce** the analysis cost **without sacrificing** the analysis precision?
- In terms of memory and time consumption



Software Complexity

less-382 (23,822 LOC)



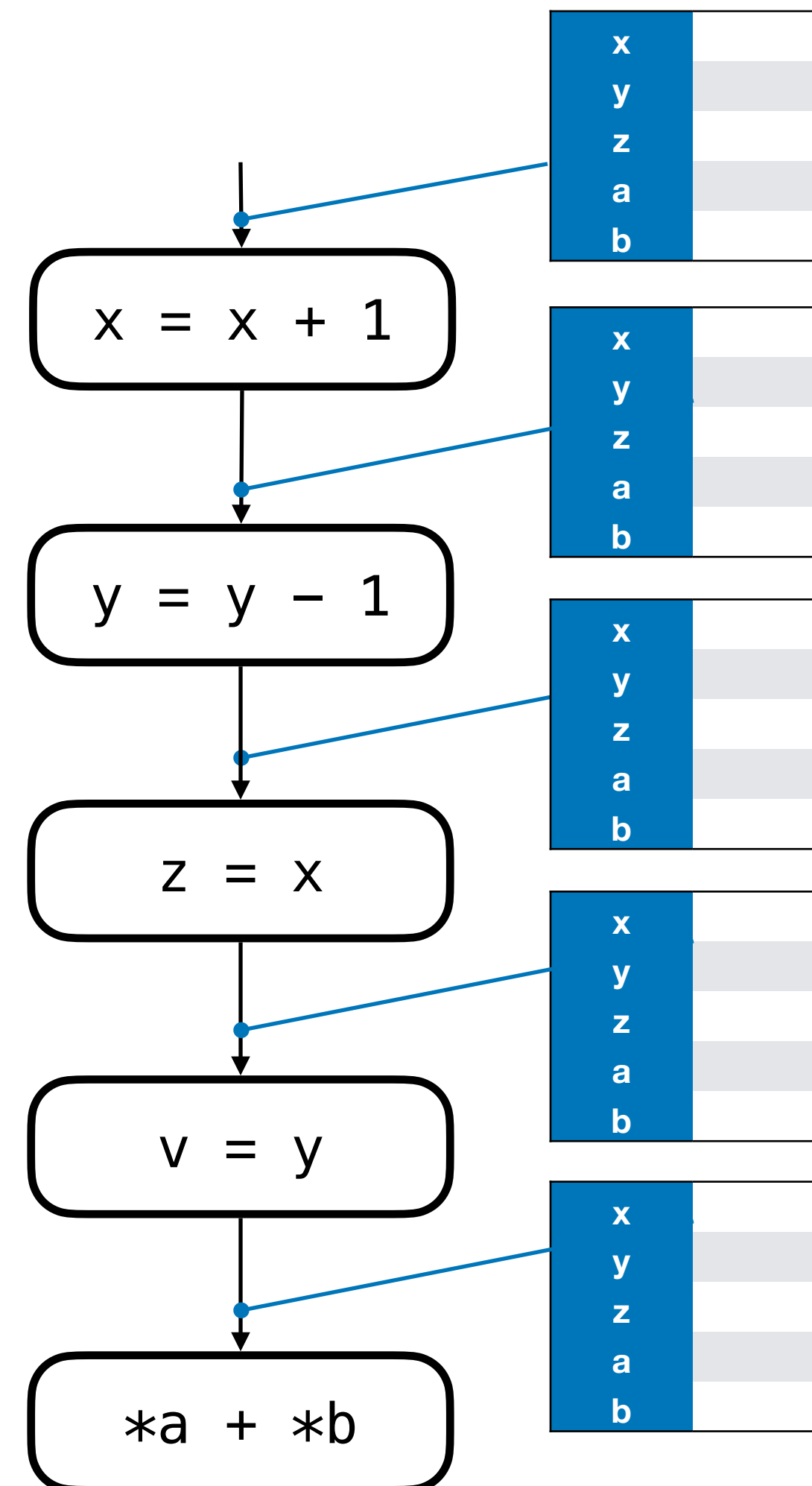
Key Idea: Sparsity

- “*Meaningful semantic effect is so sparse*”
- **Spatial sparsity**: each program portion (an expression, a statement, a block, etc) usually accesses only a small part of the whole memory
- **Temporal sparsity**: after the definition of a memory location, its use is not immediate but a while later
- Generally applicable to flow-sensitive analyses when a memory is a map from abstract locations to abstract values, e.g.,

$$\mathbb{D}^\# = \mathbb{L} \rightarrow (\mathbb{X} \rightarrow \mathbb{V}^\#)$$

Example

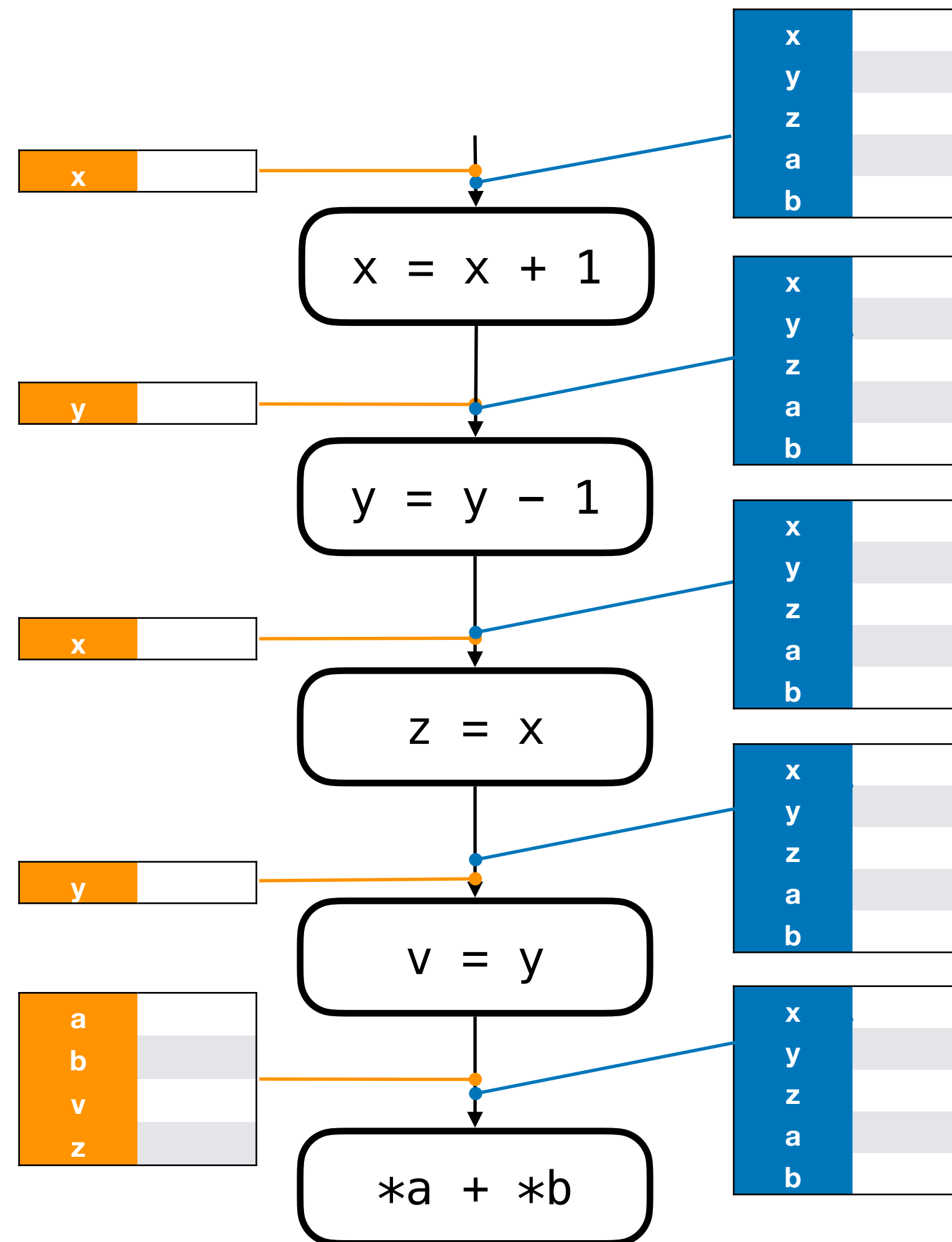
- Vanilla (non-sparse) analysis



*Assume a and b point to z and v

Example

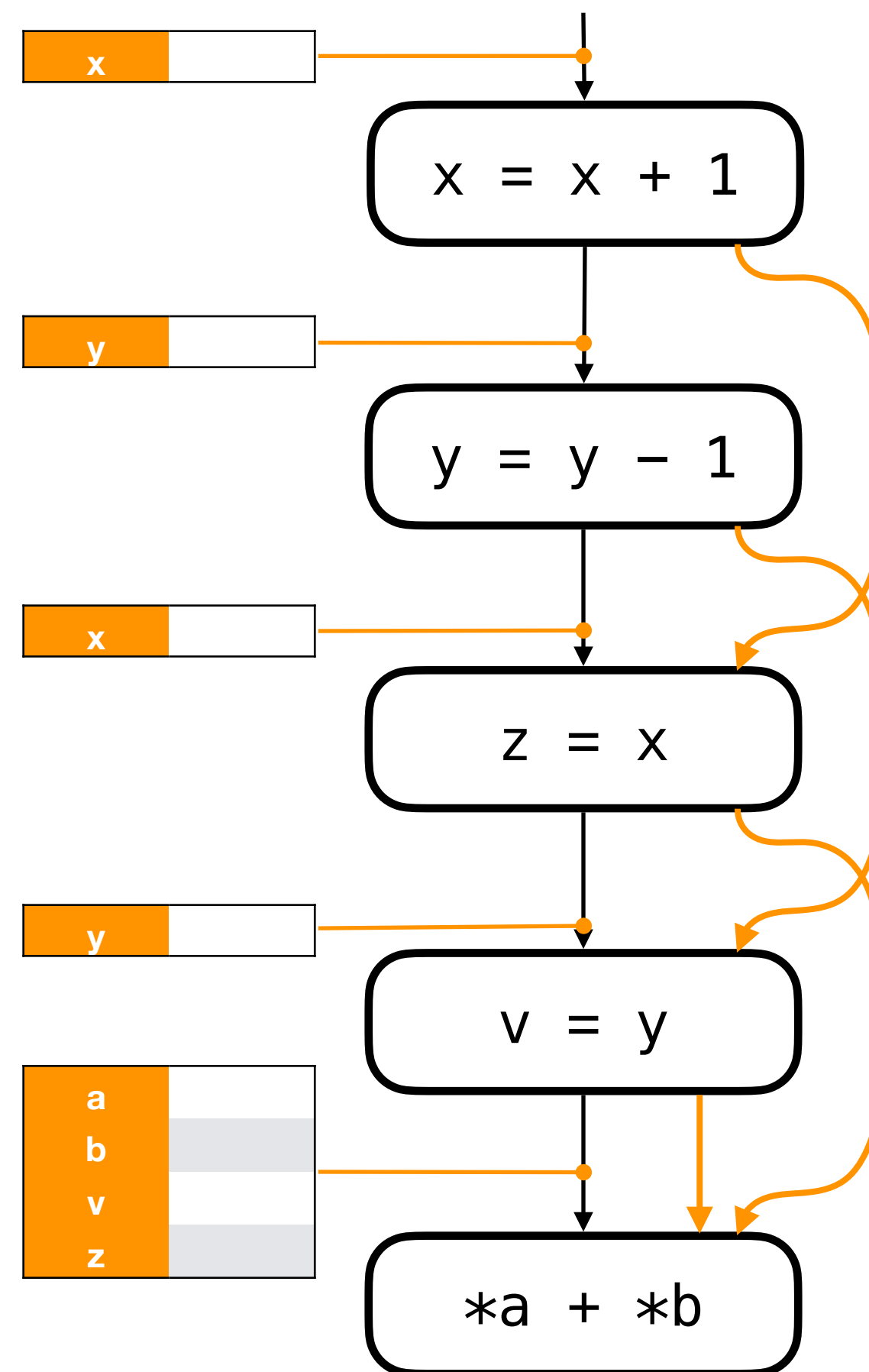
- Spatial sparsity



*Assume a and b point to z and v

Example

- Spatial + temporal sparsity



*Assume a and b point to z and v

Spatial Sparsity

- Only need the part of the memory **used** in that program portion
 - Otherwise, discard (so-called abstract garbage collection)
- The original abstract semantic function:

$$F^\# : (\mathbb{L} \rightarrow \mathbb{M}^\#) \rightarrow (\mathbb{L} \rightarrow \mathbb{M}^\#)$$

- The sparse version:

$$F_{sparse}^\# : (\mathbb{L} \rightarrow \mathbb{M}_{sparse}^\#) \rightarrow (\mathbb{L} \rightarrow \mathbb{M}_{sparse}^\#)$$

$$\mathbb{M}_{sparse}^\# = \{m^\# \in \mathbb{M}^\# \mid \text{dom}(m^\#) = U^\#(l), l \in \mathbb{L}\} \cup \{\perp\}$$

Abstract locations to be
used for each label

Temporal Sparsity

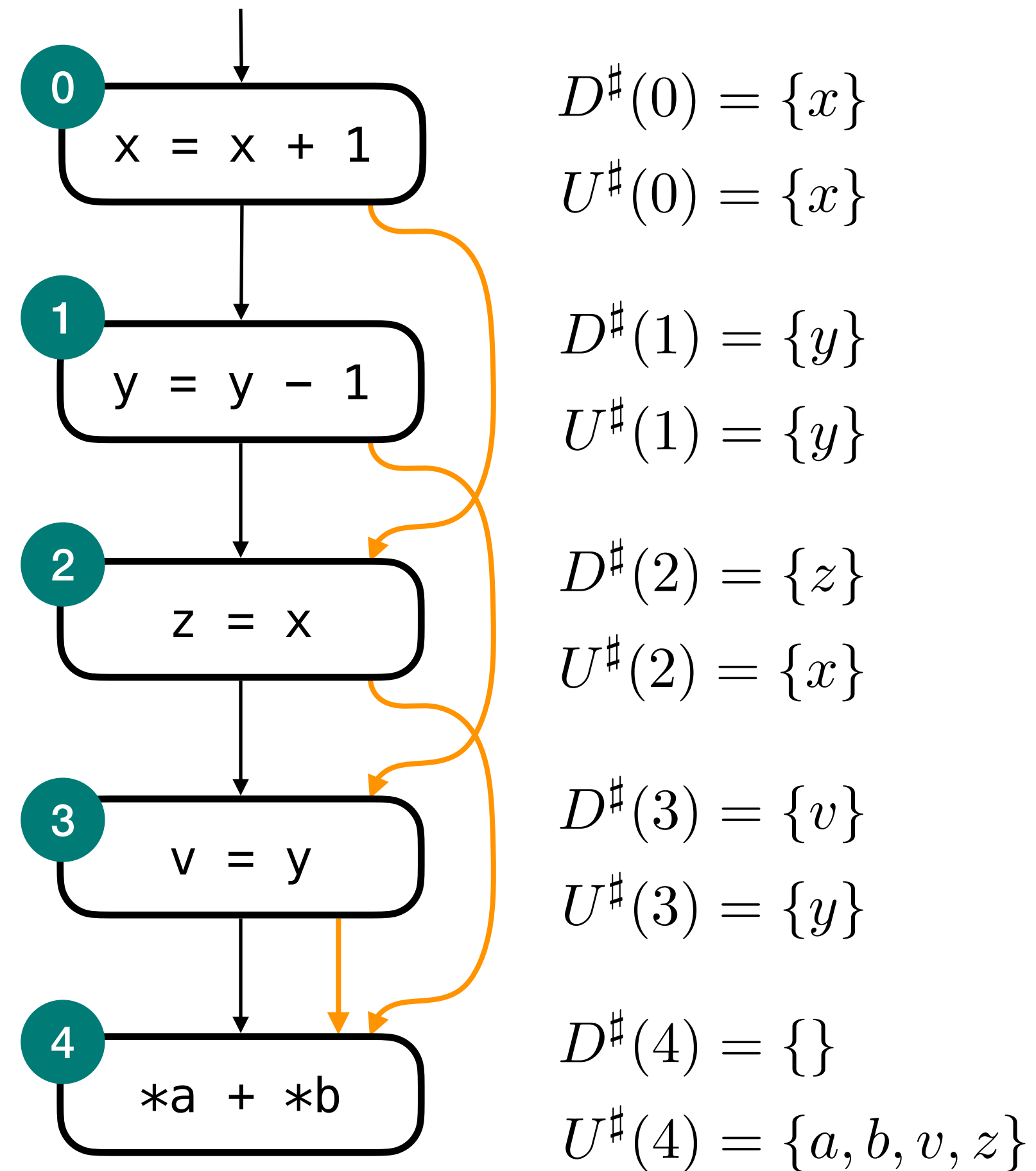
- Follow the **semantic dependency** to directly deliver the memory effect
 - Not blindly following the syntactic control-flow
- **Def-use chain**: for each label, defined locations are directly passed to its use labels

$$\langle l, m^\# \rangle \hookrightarrow_{sparse}^\# \langle l', m^{\#'} \rangle$$

Directly propagate defined locations
to the use points

Example

- Def-use chain



*Assume a and b point to z and v

Def-Use Chain



- How to formally define **semantic def and use sets**?
- Def and Use:

$$D^\#(l) = \{x \mid \exists m^\# \sqsubseteq \text{lf}p F^\#(l). m^\#(x) \neq m^{\#'}(x), \langle l, m^\# \rangle \hookrightarrow^\# \langle -, m^{\#'} \rangle\}$$

$$U^\#(l) = \{x \mid \exists m^\# \sqsubseteq \text{lf}p F^\#(l). m_1^\#|_{D^\#(l)} \neq m_2^\#|_{D^\#(l)}, \langle l, m^\# \rangle \hookrightarrow^\# \langle -, m_1^\# \rangle, \langle l, m^\# \setminus x \rangle \hookrightarrow^\# \langle -, m_2^\# \rangle\}$$

- Def-use chain:

$$l_0 \overset{x}{\rightsquigarrow} l_n \iff \langle l_0, \text{lf}p F^\#(l_0) \rangle \hookrightarrow^\# \dots \hookrightarrow^\# \langle l_n, \text{lf}p F^\#(l_n) \rangle \\ \wedge x \in D^\#(l_0) \wedge x \in U^\#(l_n) \wedge \forall i \in (0, n). x \notin D^\#(l_i)$$

- Theorem:

$$\text{lf}p F^\# = \text{lf}p F_{sparse}^\# \text{ modulo } D^\#$$

Computing Def-Use Chain

- The ideal def-use chain is available **only after** the main analysis
 - It requires the full abstract semantics of a given program
- But, we need def-use chain **before** the analysis to speed up
- Solution: compute **approximated def-use chain** by yet another analysis

Pre-Analysis

- A coarser (hence quicker) analysis than the main analysis
- Any sound approximation of the main analysis is eligible
- For example, the flow-insensitive version of the main analysis:

$$\mathbb{D}^\# = \mathbb{L} \rightarrow \mathbb{M}^\# \xrightleftharpoons[\alpha]{\gamma} \mathbb{D}_{pre}^\# = \mathbb{M}^\#$$

- Relationship between the pre and main analysis
 - The pre-analysis **controls the sparsity**, not the final precision

Precision-Preserving Def-Use Chain

- Safe def / use sets from pre-analysis must satisfy the following conditions:

1. The def and use sets from the pre-analysis over-approximate those of the original analysis:

$$\forall l \in \mathbb{L}. D^\#(l) \subseteq D_{pre}^\#(l) \quad \text{and} \quad U^\#(l) \subseteq U_{pre}^\#(l)$$

2. All spurious definitions from the pre-analysis are included in the use set from the pre-analysis:

$$\forall l \in \mathbb{L}. D_{pre}^\#(l) \setminus D^\#(l) \subseteq U_{pre}^\#(l)$$

- Approximated def-use chain by the pre-analysis

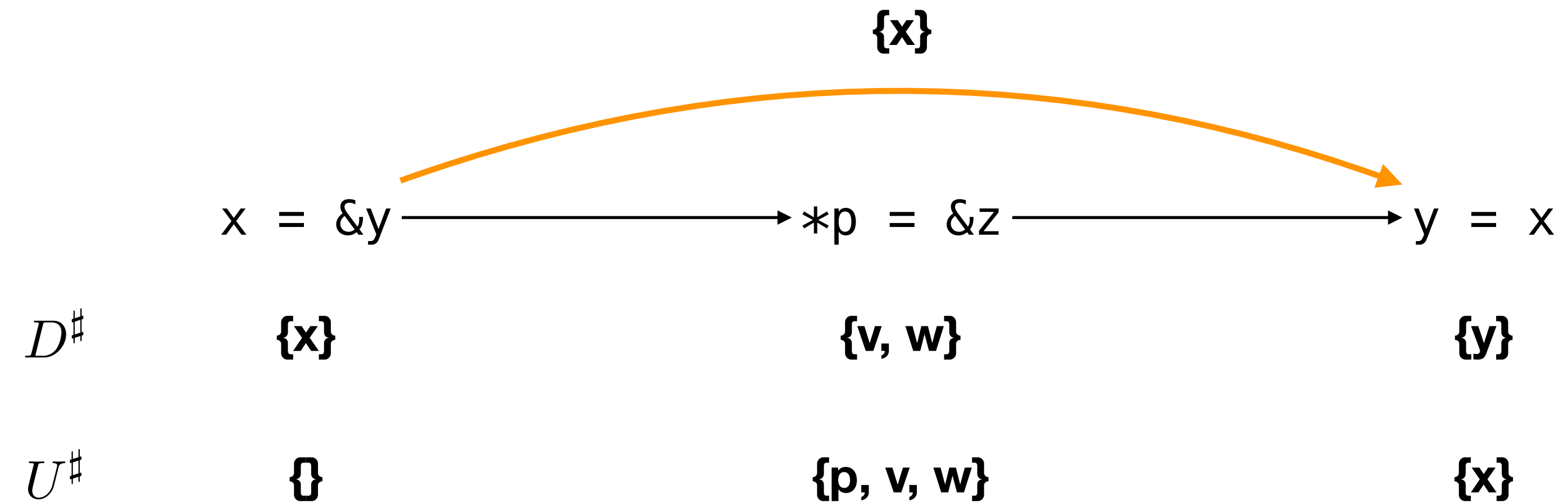
$$l_0 \overset{x}{\rightsquigarrow}_{pre} l_n \iff$$

$$\langle l_0, \mathbf{lfp} F_{pre}^\#(l_0) \rangle \hookrightarrow^\# \dots \hookrightarrow^\# \langle l_n, \mathbf{lfp} F_{pre}^\#(l_n) \rangle$$

$$\wedge x \in D_{pre}^\#(l_0) \wedge x \in U_{pre}^\#(l_n) \wedge \forall i \in (0, n). x \notin D_{pre}^\#(l_i)$$

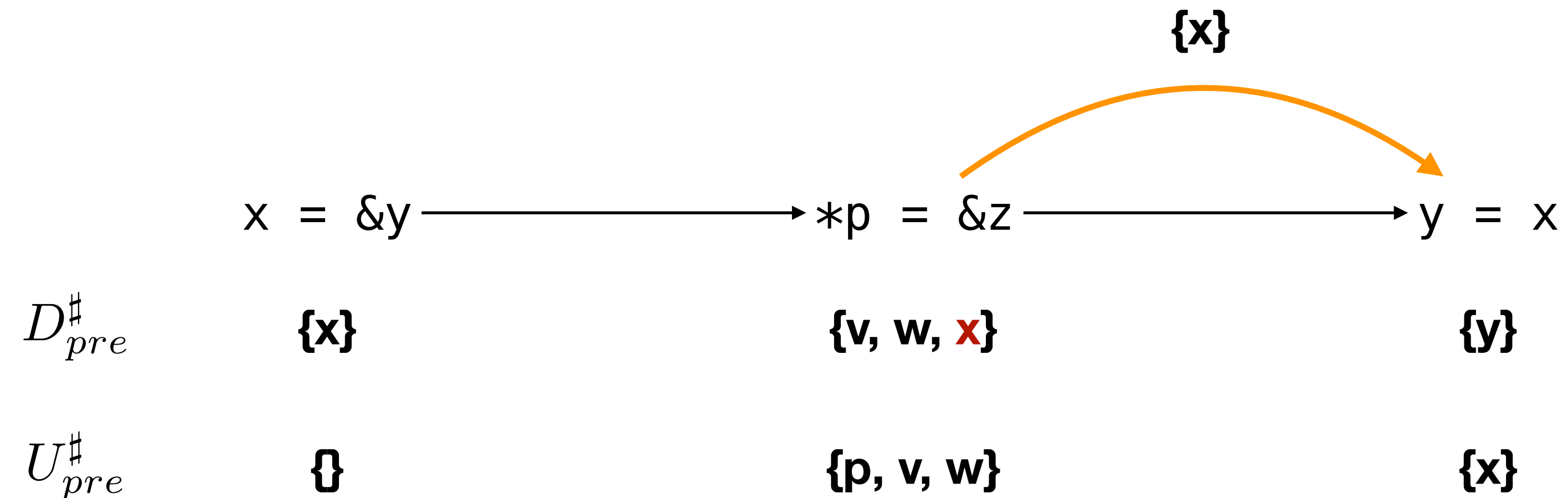
Example

- Def-use chain by the original analysis



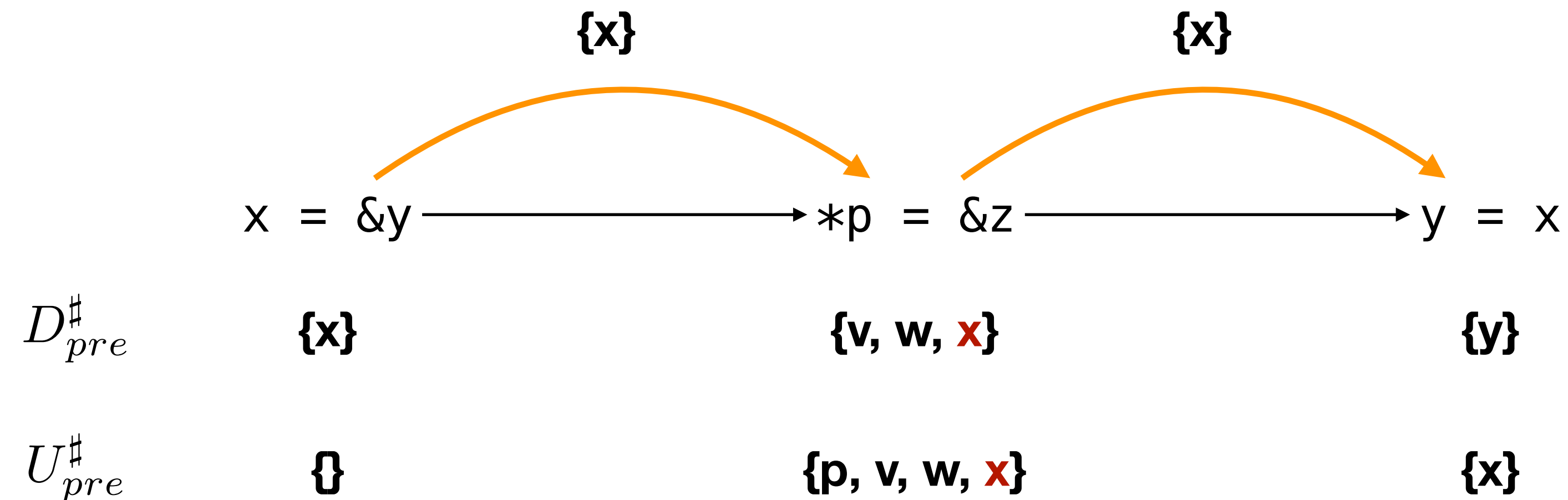
Example

- Unsafe def-use edge by the pre-analysis



Example

- Safe def-use edge by the pre-analysis



Realizable Sparse Analysis

- The final sparse abstract semantic function:

$$\mathbb{M}_{sparse\#}^\# = \{m^\# \in \mathbb{M}^\# \mid dom(m^\#) = U_{pre}^\#(l), l \in \mathbb{L}\} \cup \{\perp\}$$

$$F_{sparse\#}^\# : (\mathbb{L} \rightarrow \mathbb{M}_{sparse\#}^\#) \rightarrow (\mathbb{L} \rightarrow \mathbb{M}_{sparse\#}^\#)$$

- The sparse abstract state transitions (using approx. def-use chains):

$$\langle l, m^\# \rangle \hookrightarrow_{sparse\#}^\# \langle l', m^{\#'} \rangle$$

- Theorem: $\text{lfp} F^\# = \text{lfp} F_{sparse\#}^\#$ modulo $D_{pre}^\#$


Benchmarks

Program	LOC	Functions	Statements	Blocks	maxSCC	AbsLocs
gzip-1.2.4a	7K	132	6,446	4,152	2	1,784
bc-1.06	13K	132	10,368	4,731	1	1,619
tar-1.13	20K	221	12,199	8,586	13	3,245
less-382	23K	382	23,367	9,207	46	3,658
make-3.76.1	27K	190	14,010	9,094	57	4,527
wget-1.9	35K	433	28,958	14,537	13	6,675
screen-4.0.2	45K	588	39,693	29,498	65	12,566
a2ps-4.14	64K	980	86,867	27,565	6	17,684
sendmail-8.13.6	130K	756	76,630	52,505	60	19,135
nethack-3.3.0	211K	2,207	237,427	157,645	997	54,989
vim60	227K	2,770	150,950	107,629	1,668	40,979
emacs-22.1	399K	3,388	204,865	161,118	1,554	66,413
python-2.5.1	435K	2,996	241,511	99,014	723	51,859
linux-3.0	710K	13,856	345,407	300,203	493	139,667
gimp-2.6	959K	11,728	1,482,230	286,588	2	190,806
ghostscript-9.00	1,363K	12,993	2,891,500	342,293	39	201,161


*Oh et al., Design and Implementation of Sparse Global Analyses for C-like Languages, PLDI'12

Practical Impact

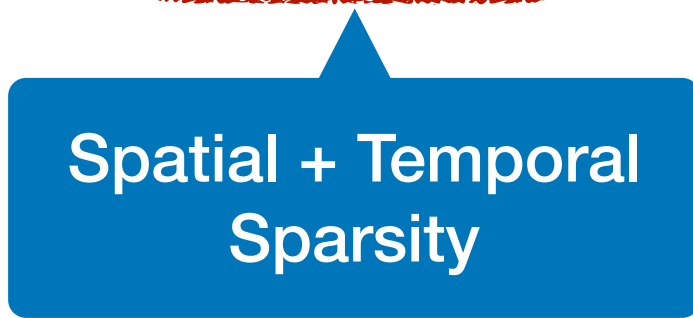
Programs	Interval _{vanilla}		Interval _{base}		Spd _{↑1}	Mem _{↓1}	Interval _{sparse}						Spd _{↑2}	Mem _{↓2}
	Time	Mem	Time	Mem			Dep	Fix	Total	Mem	$\hat{D}(c)$	$\hat{U}(c)$		
gzip-1.2.4a	772	240	14	65	55 x	73 %	2	1	3	63	2.4	2.5	5 x	3 %
bc-1.06	1,270	276	96	126	13 x	54 %	4	3	7	75	4.6	4.9	14 x	40 %
tar-1.13	12,947	881	338	177	38 x	80 %	6	2	8	93	2.9	2.9	42 x	47 %
less-382	9,561	1,113	1,211	378	8 x	66 %	27	6	33	127	11.9	11.9	37 x	66 %
make-3.76.1	24,240	1,391	1,893	443	13 x	68 %	16	5	21	114	5.8	5.8	90 x	74 %
wget-1.9	44,092	2,546	1,214	378	36 x	85 %	8	3	11	85	2.4	2.4	110 x	78 %
screen-4.0.2	∞	N/A	31,324	3,996	N/A	N/A	724	43	767	303	53.0	54.0	41 x	92 %
a2ps-4.14	∞	N/A	3,200	1,392	N/A	N/A	31	9	40	353	2.6	2.8	80 x	75 %
sendmail-8.13.6	∞	N/A	∞	N/A	N/A	N/A	517	227	744	678	20.7	20.7	N/A	N/A
nethack-3.3.0	∞	N/A	∞	N/A	N/A	N/A	14,126	2,247	16,373	5,298	72.4	72.4	N/A	N/A
vim60	∞	N/A	∞	N/A	N/A	N/A	17,518	6,280	23,798	5,190	180.2	180.3	N/A	N/A
emacs-22.1	∞	N/A	∞	N/A	N/A	N/A	29,552	8,278	37,830	7,795	285.3	285.5	N/A	N/A
python-2.5.1	∞	N/A	∞	N/A	N/A	N/A	9,677	1,362	11,039	5,535	108.1	108.1	N/A	N/A
linux-3.0	∞	N/A	∞	N/A	N/A	N/A	26,669	6,949	33,618	20,529	76.2	74.8	N/A	N/A
gimp-2.6	∞	N/A	∞	N/A	N/A	N/A	3,751	123	3,874	3,602	4.1	3.9	N/A	N/A
ghostscript-9.00	∞	N/A	∞	N/A	N/A	N/A	14,116	698	14,814	6,384	9.7	9.7	N/A	N/A



Non-sparse



Spatial Sparsity



Spatial + Temporal Sparsity

*Oh et al., Design and Implementation of Sparse Global Analyses for C-like Languages, PLDI'12

Summary

- Sparse analysis: a **general framework** for reducing the analysis cost while preserving the precision
 - Input: sound yet scalability-unattended static analysis
- Key idea: **right part at right moment**
- Based on semantic **def-use chain** rather than syntactic control-flow
- Approximated def-use chain by **pre-analysis**
 - **Safety conditions** on def and use