Shape Analysis via 3-Valued Logic

Mooly Sagiv
Tel Aviv University

http://www.cs.tau.ac.il/~msagiv/toplas02.pdf

www.cs.tau.ac.il/~tvla

Tentative Schedule

- ✓ Concrete interpretation
- ✓ Canonical abstraction
- Abstract interpretation using canonical abstraction
- TVLA &Applications
- Advanced Topics
  - Coarser abstractions [SAS’04]
  - Handling procedures [POPL’05, SAS’05]
  - Deriving update formulas [ESOP’05]
  - Employing theorem provers [VMCAI’04, TACAS’04, CSL’04, CAV’04, CADE’05]
  - Learning instrumentation predicates [CAV’05]

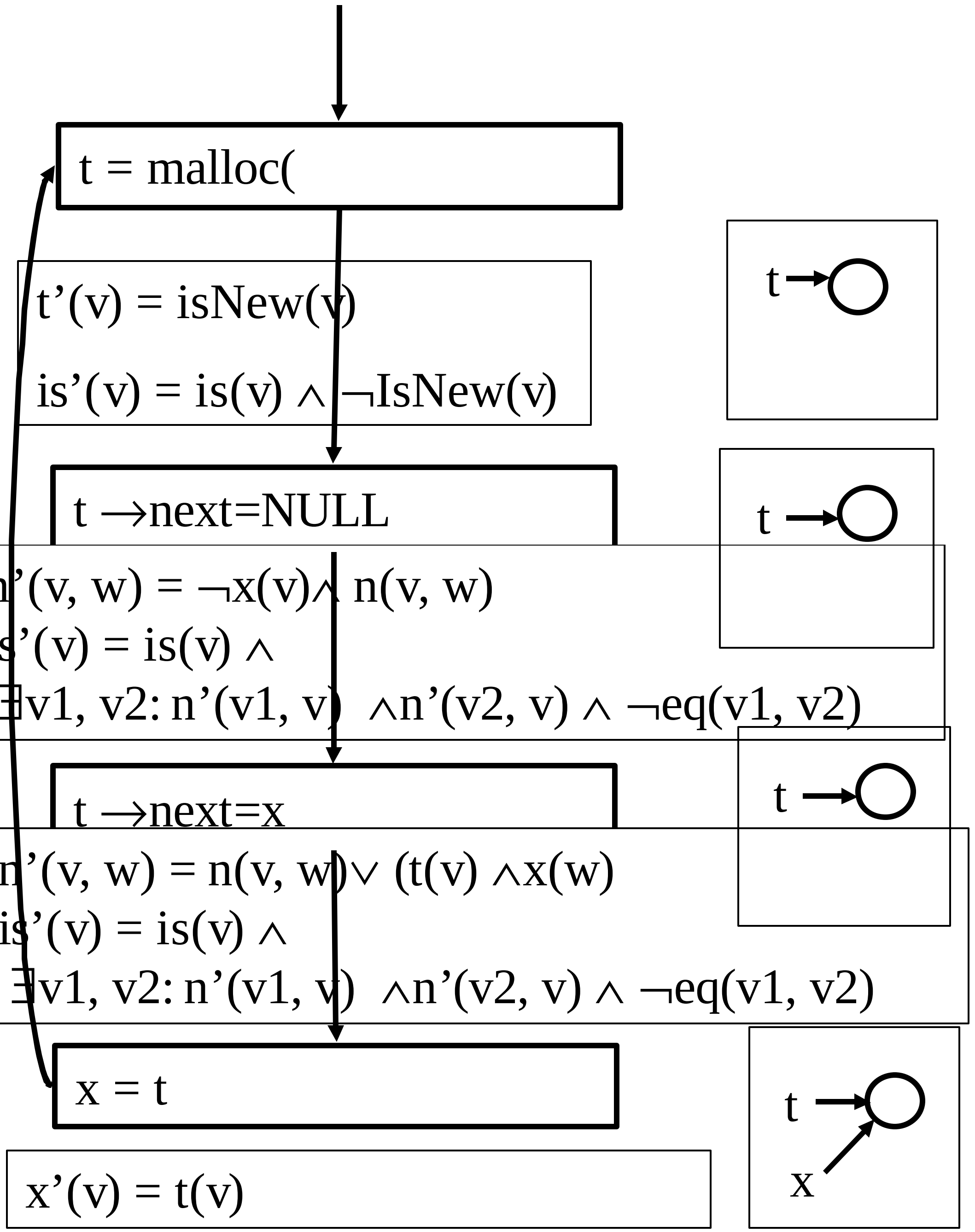
Plan

- A simple and efficient solution (Kleene)
- Best transformer
- The TVLA solution

Abstract Interpretation via Kleene Evaluation

- Reinterpret concrete semantic formulas using 3-valued logic
- After every statement apply canonical abstraction (blur)
- Simple and efficient
- Handle instrumentation
- But imprecise

Kleene Evaluation (create)



t'(v) = isNew(v)

is'(v) = is(v) ∧ ¬IsNew(v)

n'(v, w) = ¬x(v)∧ n(v, w)

is'(v) = is(v) ∧

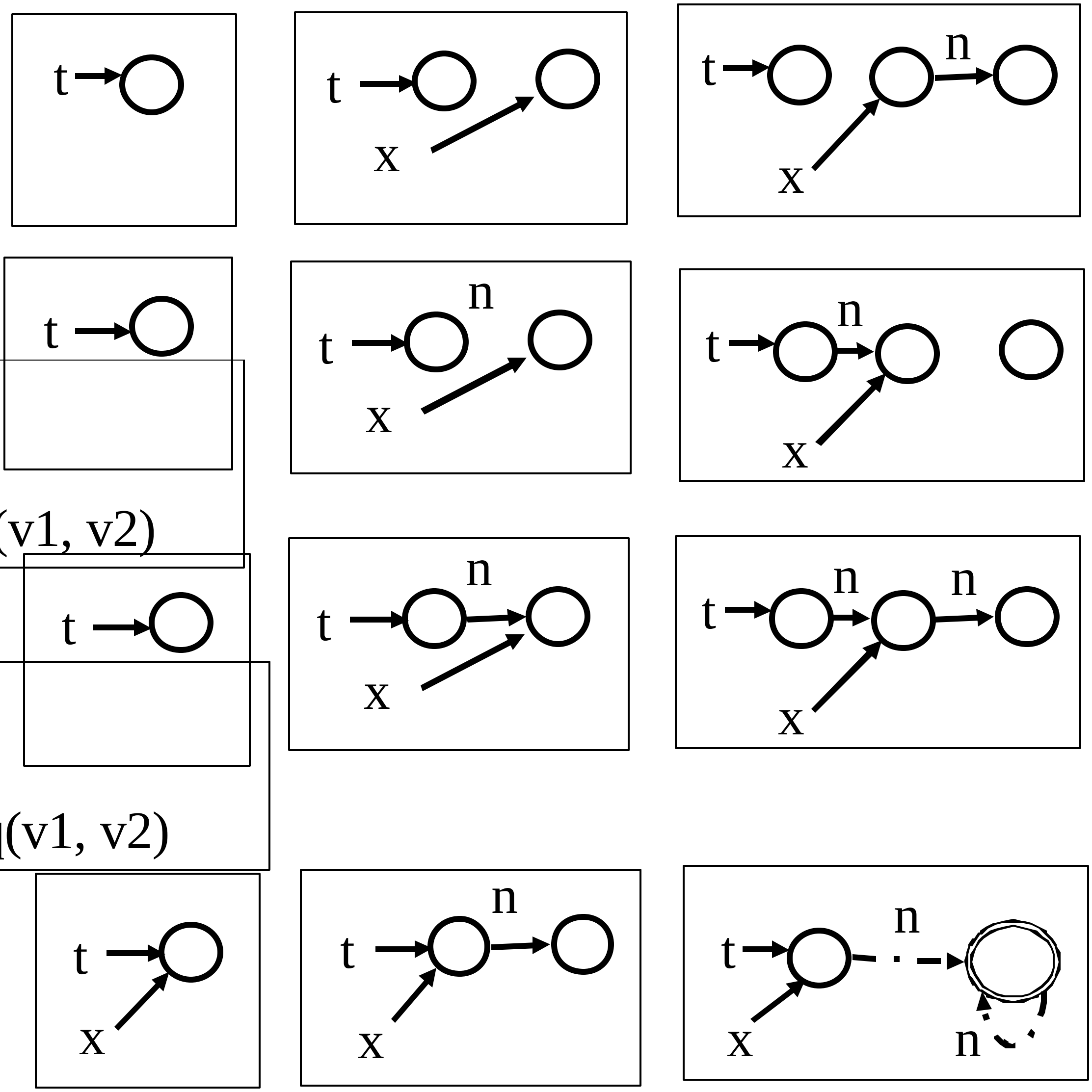
∃v1, v2: n'(v1, v) ∧n'(v2, v) ∧ ¬eq(v1, v2)

n'(v, w) = n(v, w)∨ (t(v) ∧x(w)

is'(v) = is(v) ∧

∃v1, v2: n'(v1, v) ∧n'(v2, v) ∧ ¬eq(v1, v2)

x'(v) = t(v)



Kleene Evaluation (create)

t = malloc(

t’(v) = isNew(v)

is’(v) = is(v) ∧ ¬IsNew(v)

t →next=NULL

n’(v, w) = ¬x(v)∧ n(v, w)

is’(v) = is(v) ∧

∃v1, v2: n’(v1, v) ∧n’(v2, v) ∧ ¬eq(v1, v2)

t →next=x

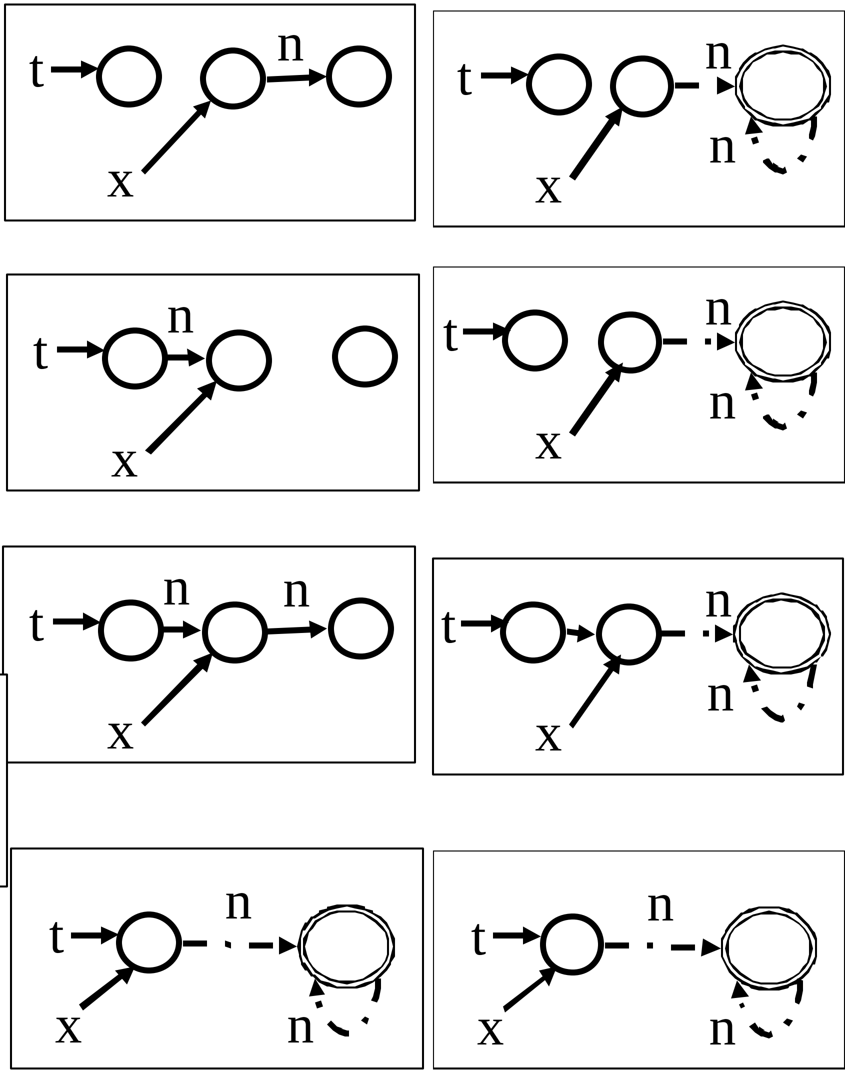
n’(v, w) = n(v, w)∨ (t(v) ∧x(w)

is’(v) = is(v) ∧

∃v1, v2: n’(v1, v) ∧n’(v2, v) ∧ ¬eq(v1, v2)

x = t

x’(v) = t(v)

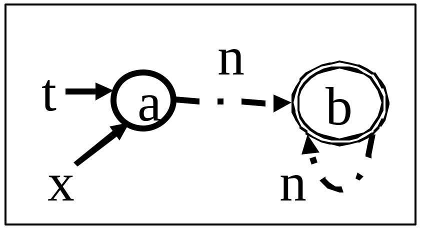


Why not use Kleene Evaluation

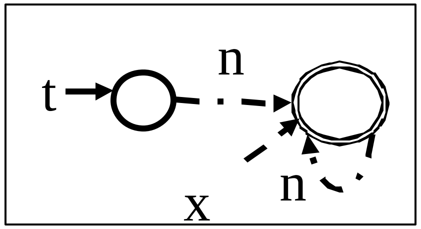
Predicate-Update Formulae for “x = x $\rightarrow$ n”

$$x'(v) = \exists\, w{:}\; x(\mathrm{w}) \wedge n(\mathrm{w}, v)$$

Old:

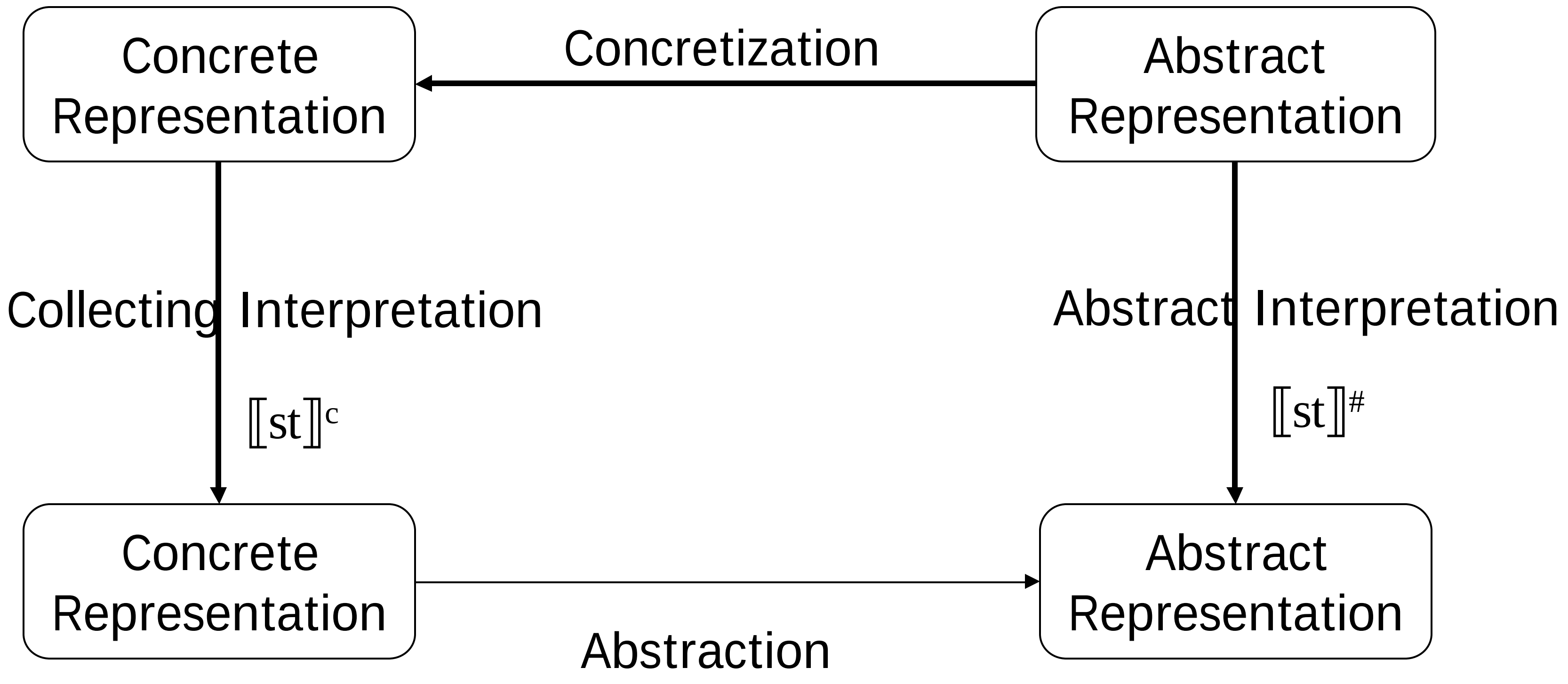


New:

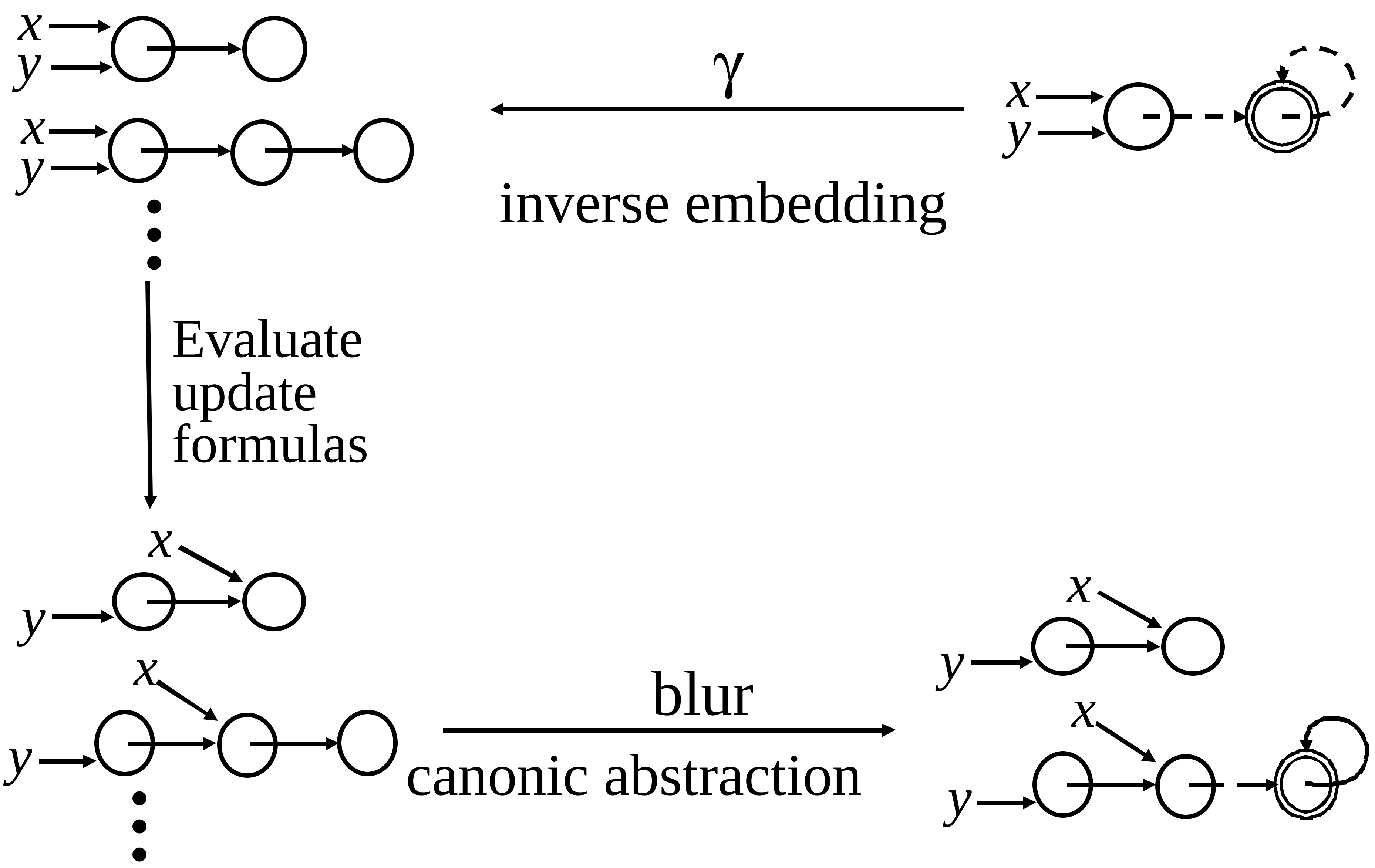


1/2

Best Conservative Interpretation (CC79)

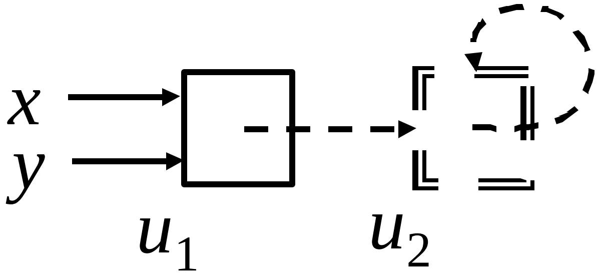


Best Transformer ($x = x \to n$)

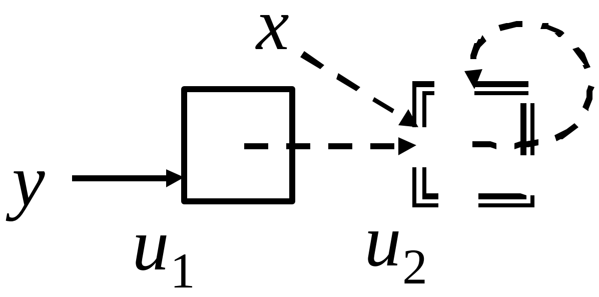


Materialization

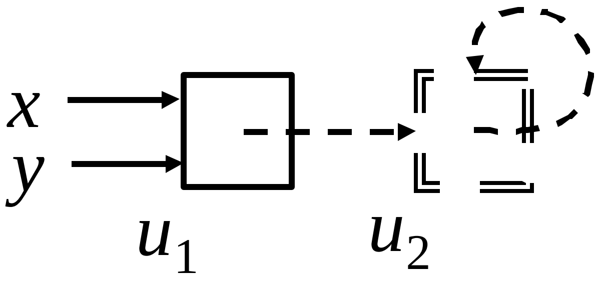
[Chase, Wegman, & Zadeck 90]



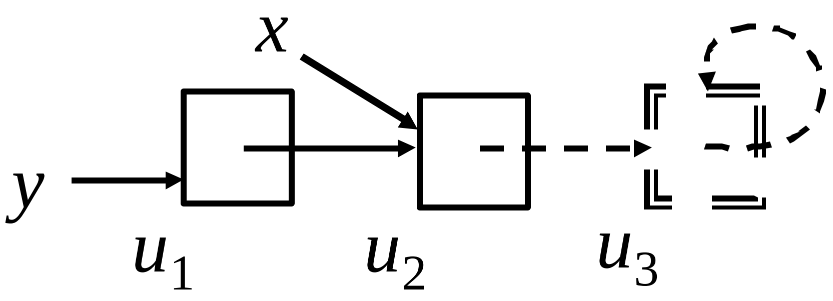
$x = x \rightarrow n$



[Sagiv, Reps, & Wilhelm 96, 98]



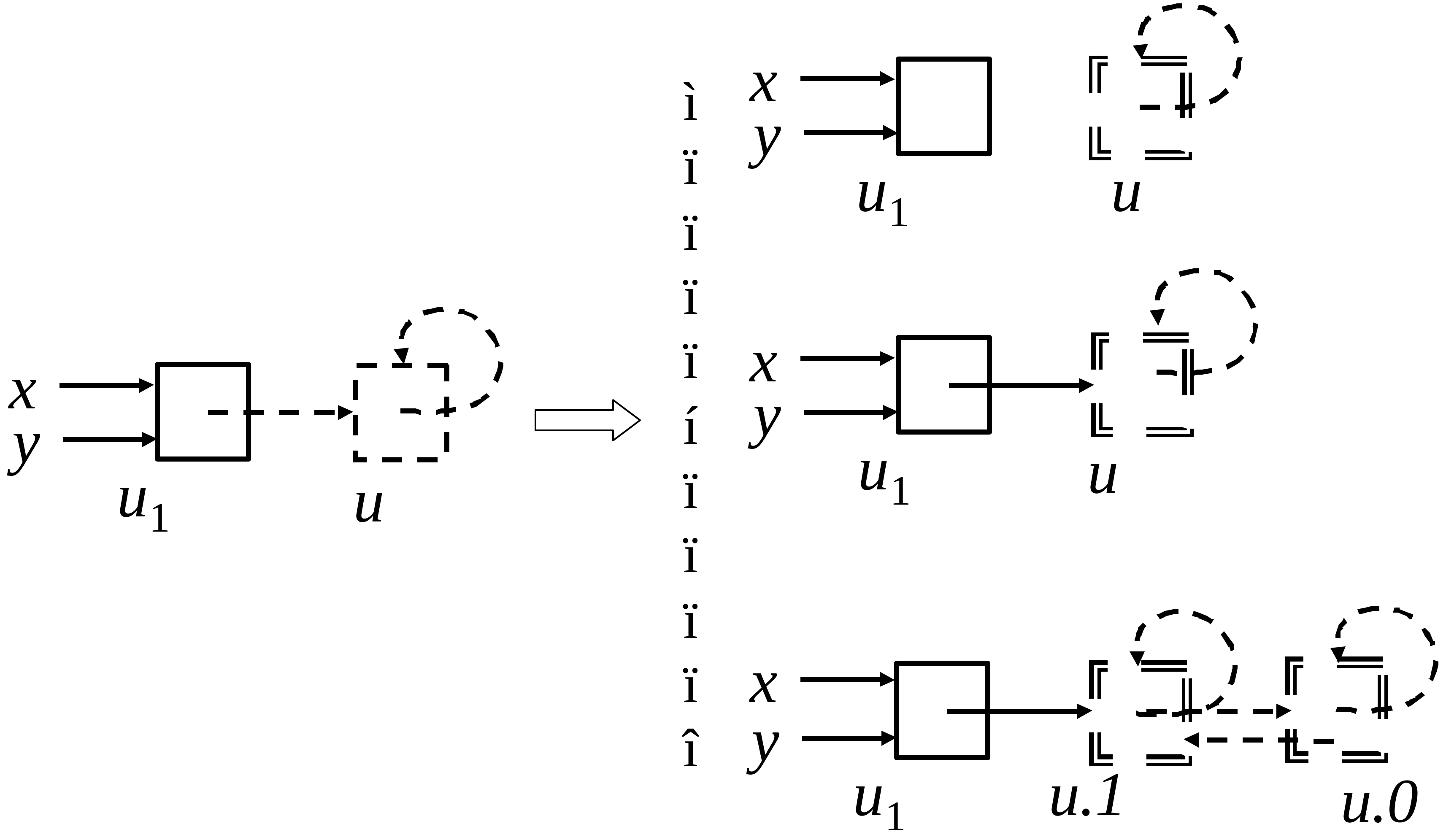
$x = x \rightarrow n$



The Focusing Principle

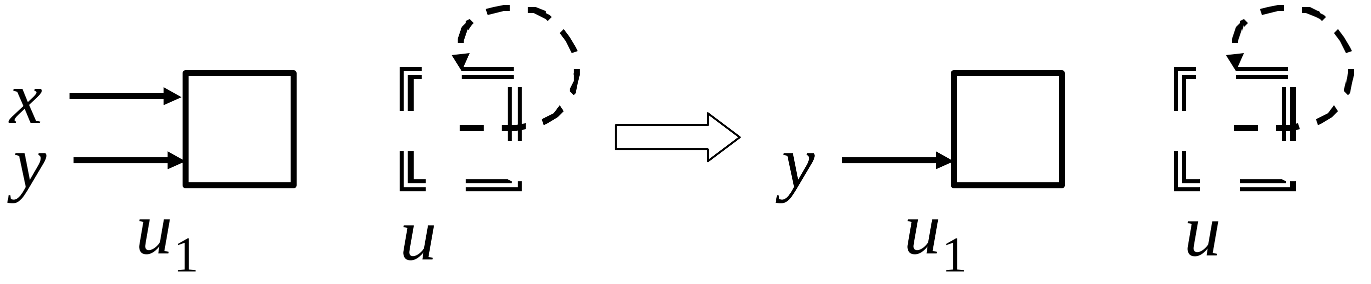
- To increase precision
  - “Bring the predicate-update formula into focus” (Force 1/2 to 0 or 1)
  - Then apply the predicate-update formulae
- Generalizes materialization
- Focus is partial concretization

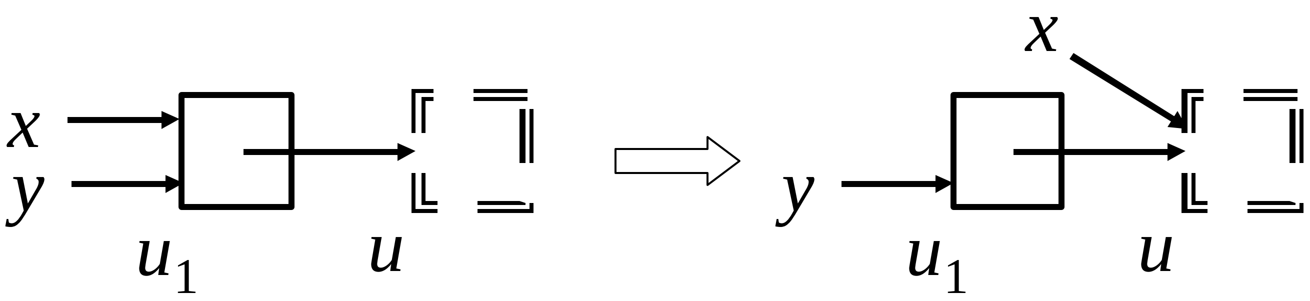
(1) Focus on $\exists\, v_1 \colon x(v_1) \wedge n(v_1, v)$

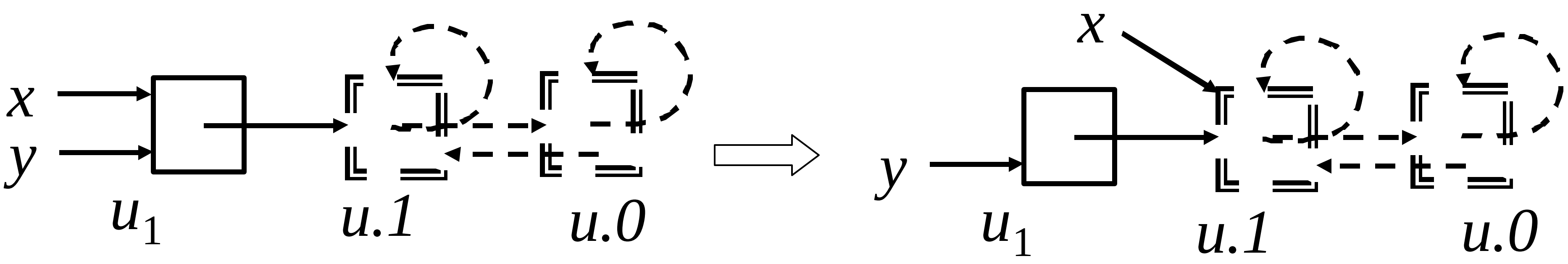


(2) Evaluate Predicate-Update Formulae

$$x'(v) = \exists v_1 : x(v_1) \wedge n(v_1, v)$$





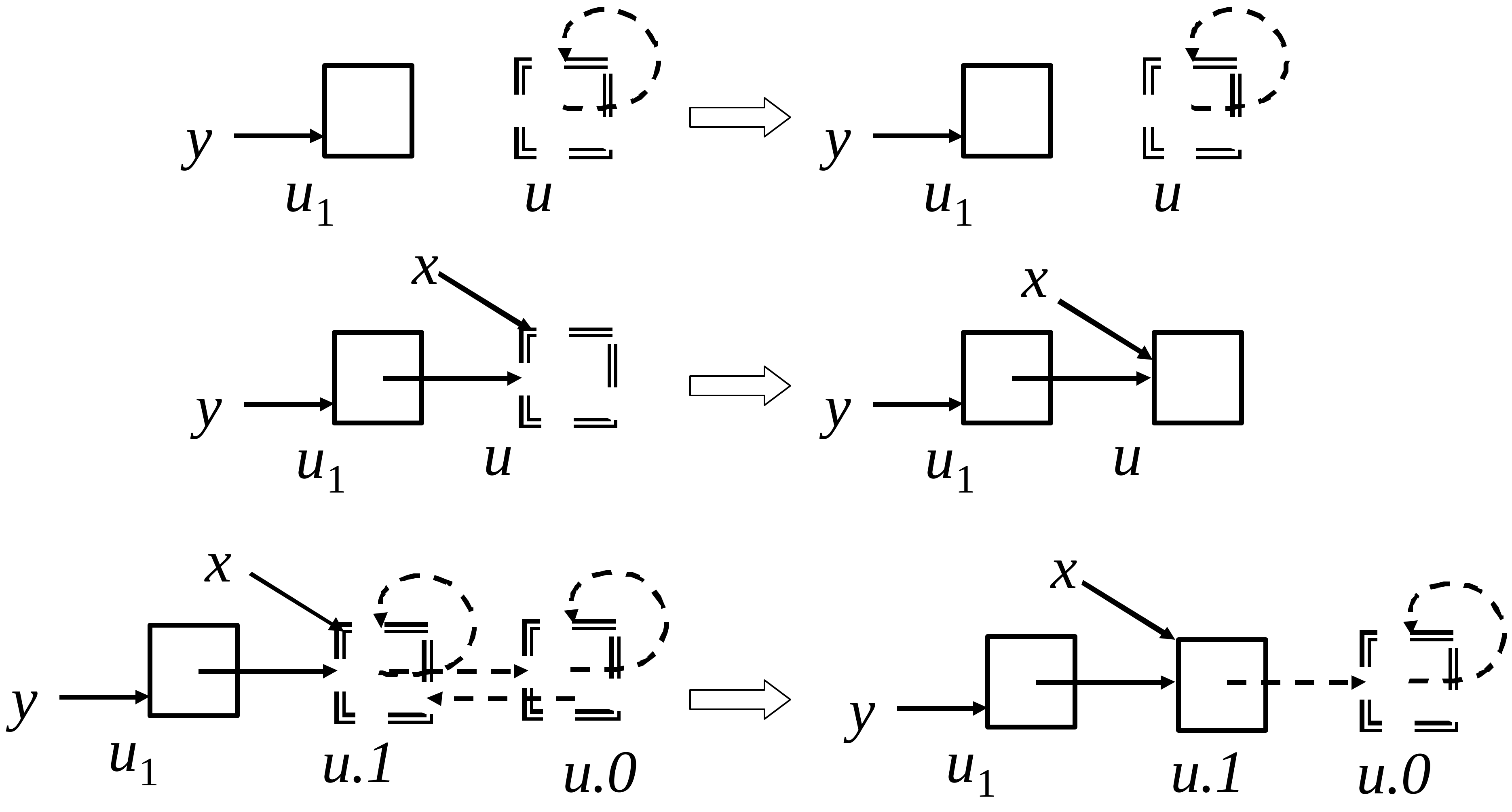


The Coercion Principle

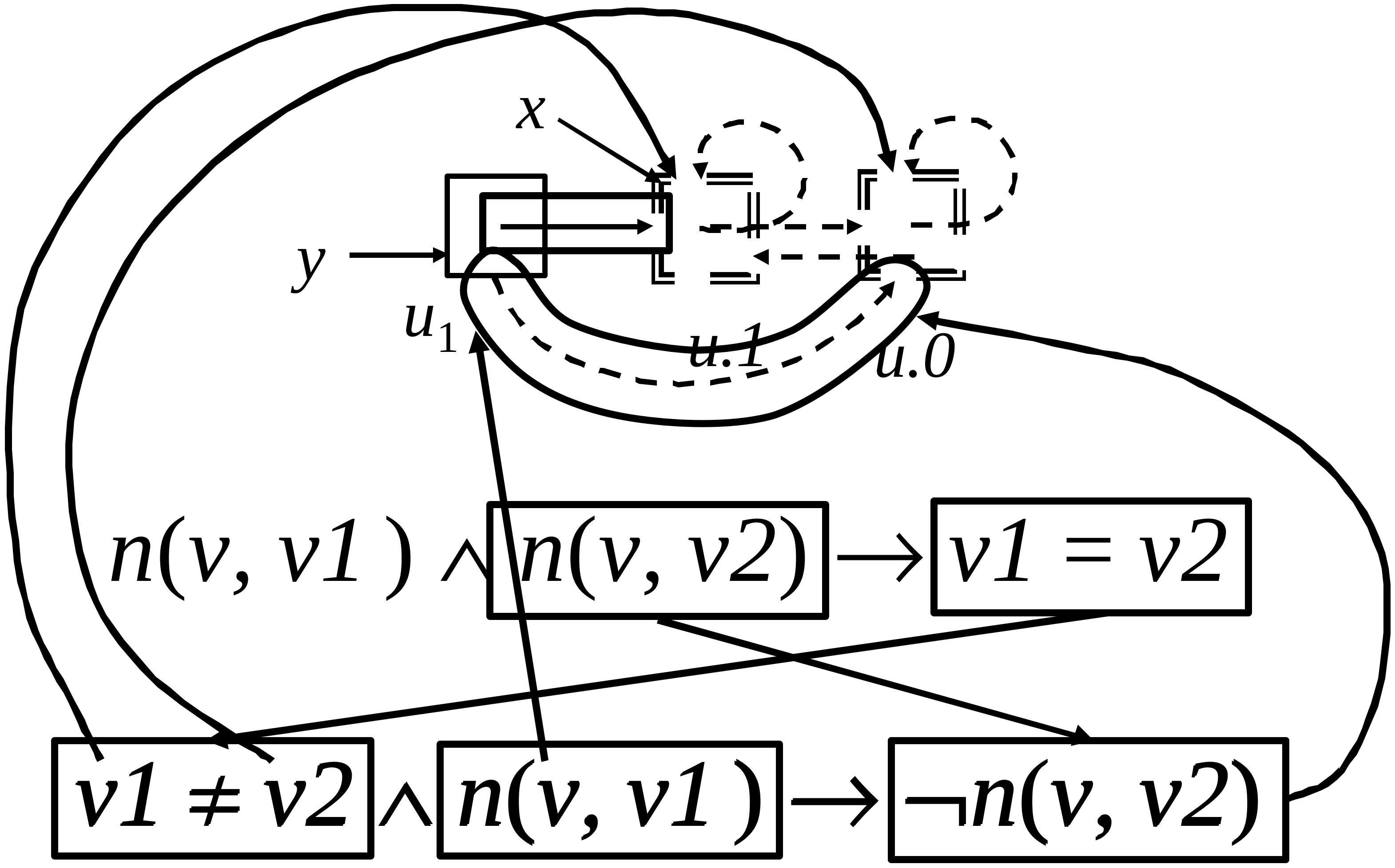
- Increase precision by exploiting some structural properties possessed by all stores (Global invariants)
- Structural properties captured by constraints

  (clauses)
- Apply a constraint solver

(3) Apply Constraint Solver



(3) Apply Constraint Solver

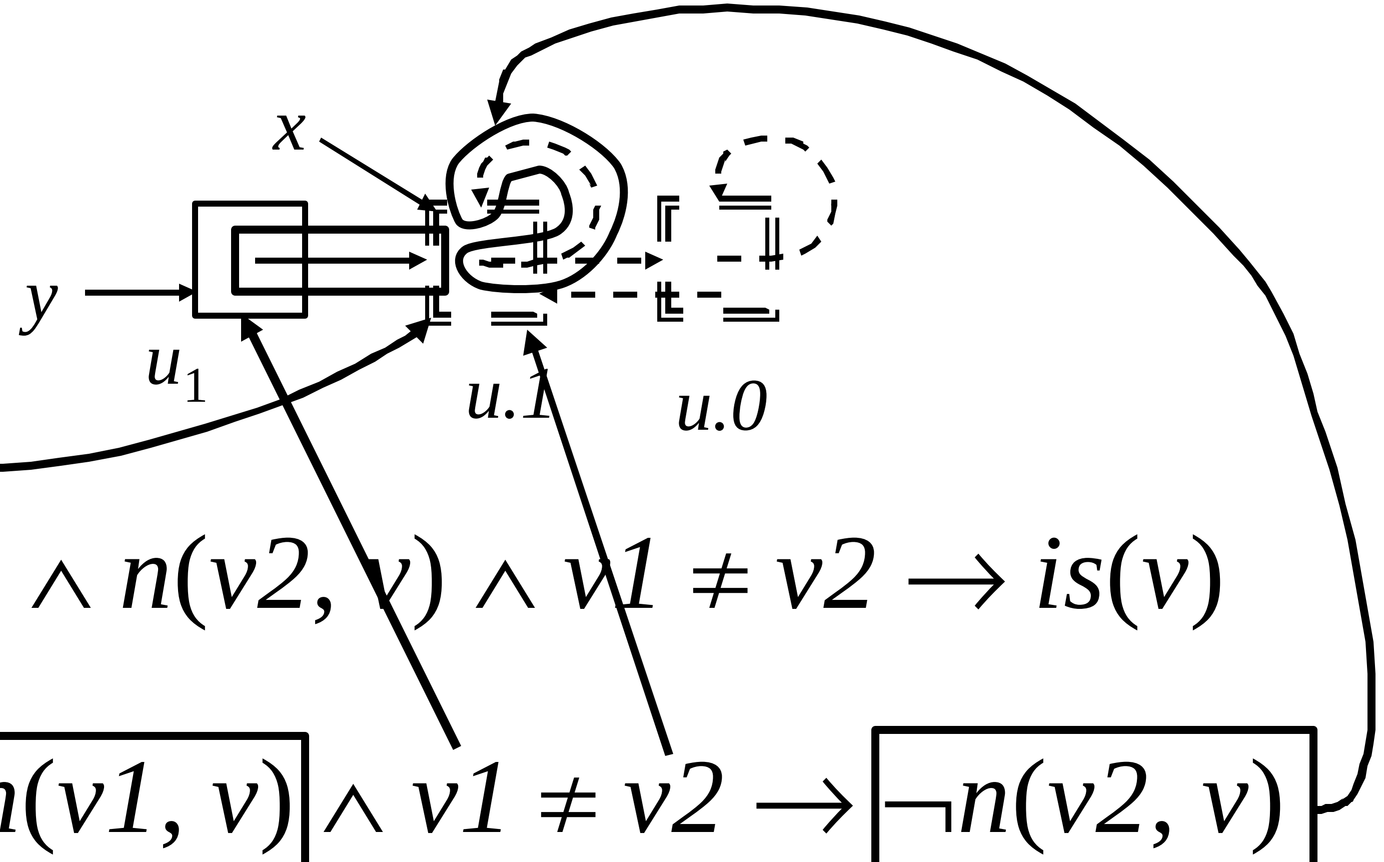


$$n(v, v1) \wedge \boxed{n(v, v2)} \rightarrow \boxed{v1 = v2}$$

$$\boxed{v1 \neq v2} \wedge \boxed{n(v, v1)} \rightarrow \boxed{\neg n(v, v2)}$$

$$\boxed{1} \wedge \boxed{1} \rightarrow \neg \boxed{0}$$

(3) Apply Constraint Solver

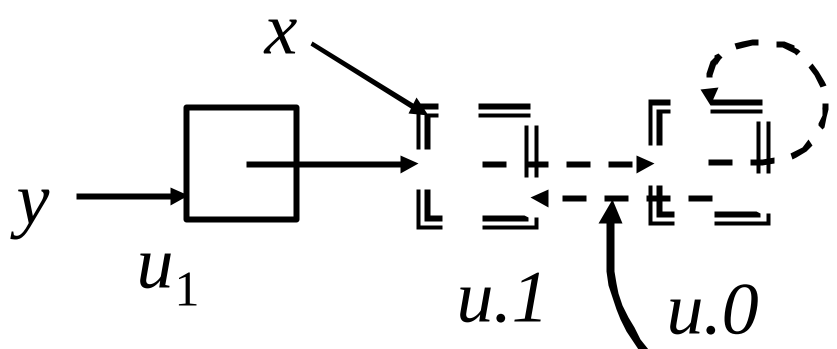


$$n(v1, v) \wedge n(v2, v) \wedge v1 \neq v2 \rightarrow is(v)$$

$$\neg is(v) \wedge \boxed{n(v1, v)} \wedge v1 \neq v2 \rightarrow \boxed{\neg n(v2, v)}$$

$$\boxed{1} \wedge \boxed{1} \wedge \boxed{1} \rightarrow \neg \boxed{0}$$

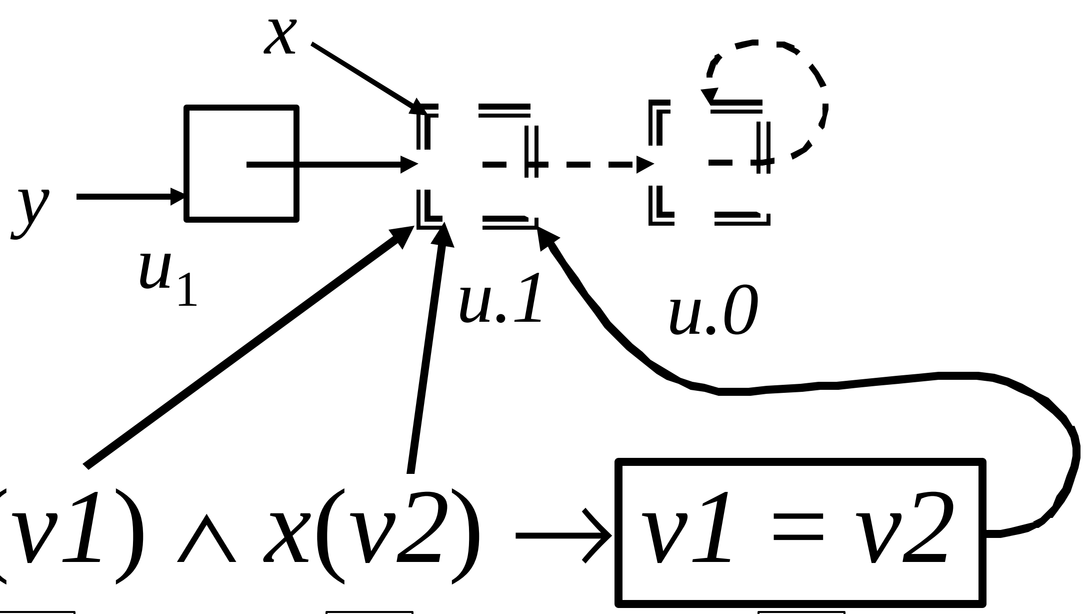
(3) Apply Constraint Solver



$$n(v1, v) \wedge n(v2, v) \wedge v1 \neq v2 \rightarrow is(v)$$

$$\neg is(v) \wedge n(v1, v) \wedge v1 \neq v2 \rightarrow \boxed{\neg n(v2, v)}$$

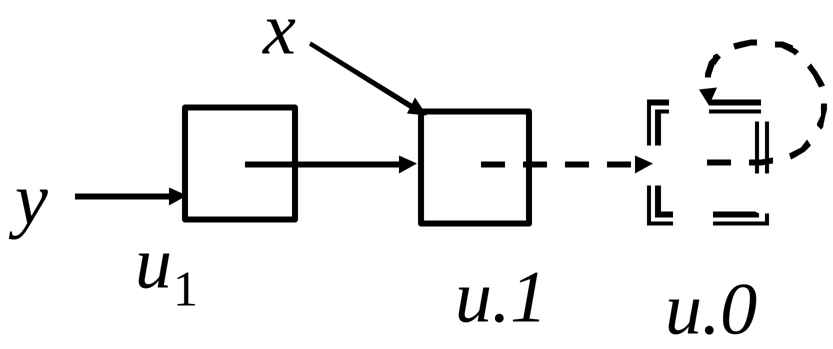
(3) Apply Constraint Solver



$$x(v1) \wedge x(v2) \rightarrow \boxed{v1 = v2}$$

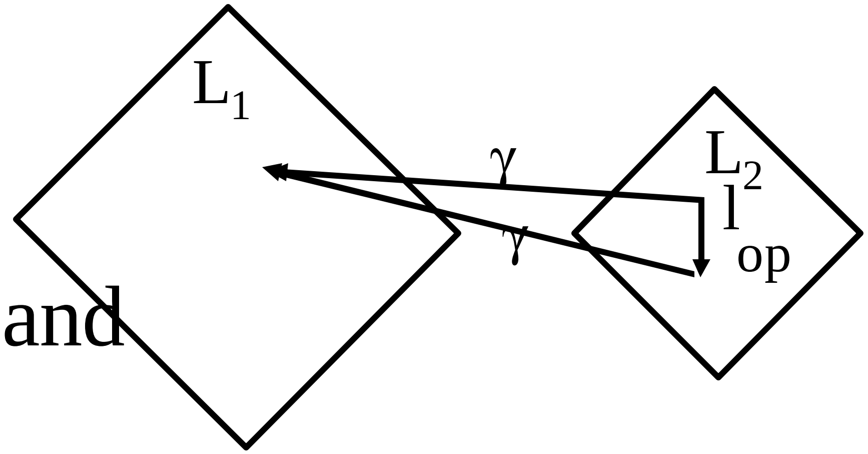
$$\boxed{1} \wedge \boxed{1} \rightarrow \boxed{1}$$

(3) Apply Constraint Solver



Semantic Reduction

- Improve the precision by recovering properties of the program semantics
- A Galois insertion $(L_1, \alpha, \gamma, L_2)$
- An operation $op: L_2 \rightarrow L_2$ is a semantic reduction
  - $\forall l \in L_2 \; op(l) \sqsubseteq l$
  - $\gamma(op(l)) = \gamma(l)$
- Can be applied before and after basic operations



Summary

- Proving near commutativity is hard
- Focus and Coerce eliminate the need for manual proofs
- Theorem provers (decision procedures) can be also useful here
- Necessary for parametric abstractions
- But slow