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

Concrete Interpretation Rules

Statement	Update formula
x =NULL	$x'(v) = 0$
x= malloc()	$x'(v) = \mathrm{IsNew}(v)$
x=y	$x'(v) = y(v)$
x=y $\rightarrow$next	$x'(v) = \exists w: y(w) \wedge n(w, v)$
x $\rightarrow$next=y	$n'(v, w) =$ $(\neg x(v) \wedge n(v, w)) \vee$ $(x(v) \wedge y(w))$

Plan

- ✓ Concrete interpretation using logic
- Canonical abstraction
- Abstract interpretation using canonical abstraction (next lesson)

3-Valued Logical Structures

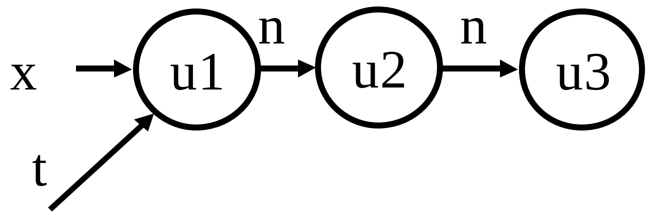
- A set of individuals (nodes) U
- Predicate meaning
  - $p^S: (U^S)^k$ ® $\{0, 1, 1/2\}$

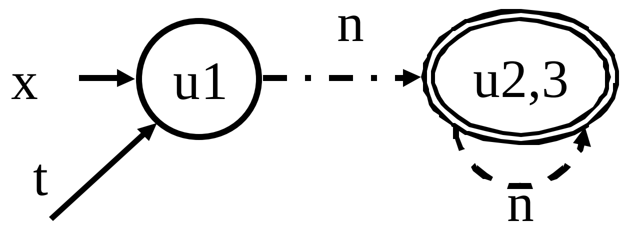
Canonical Abstraction

- Partition the individuals into equivalence classes based on the values of their unary predicates
  - Every individual is mapped into its equivalence class
- Collapse predicates via $\sqcup$
  - $p^S(u'_1, \ldots, u'_k) = \sqcup \{p^B(u_1, \ldots, u_k) \mid f(u_1)=\mathrm{u}'_1, \ldots, f(u'_k)=u'_k\}$
- At most 2^A abstract individuals

Canonical Abstraction

```
x = NULL;
while (…) do {
    t = malloc();
    t →next=x;
    x = t
}
```

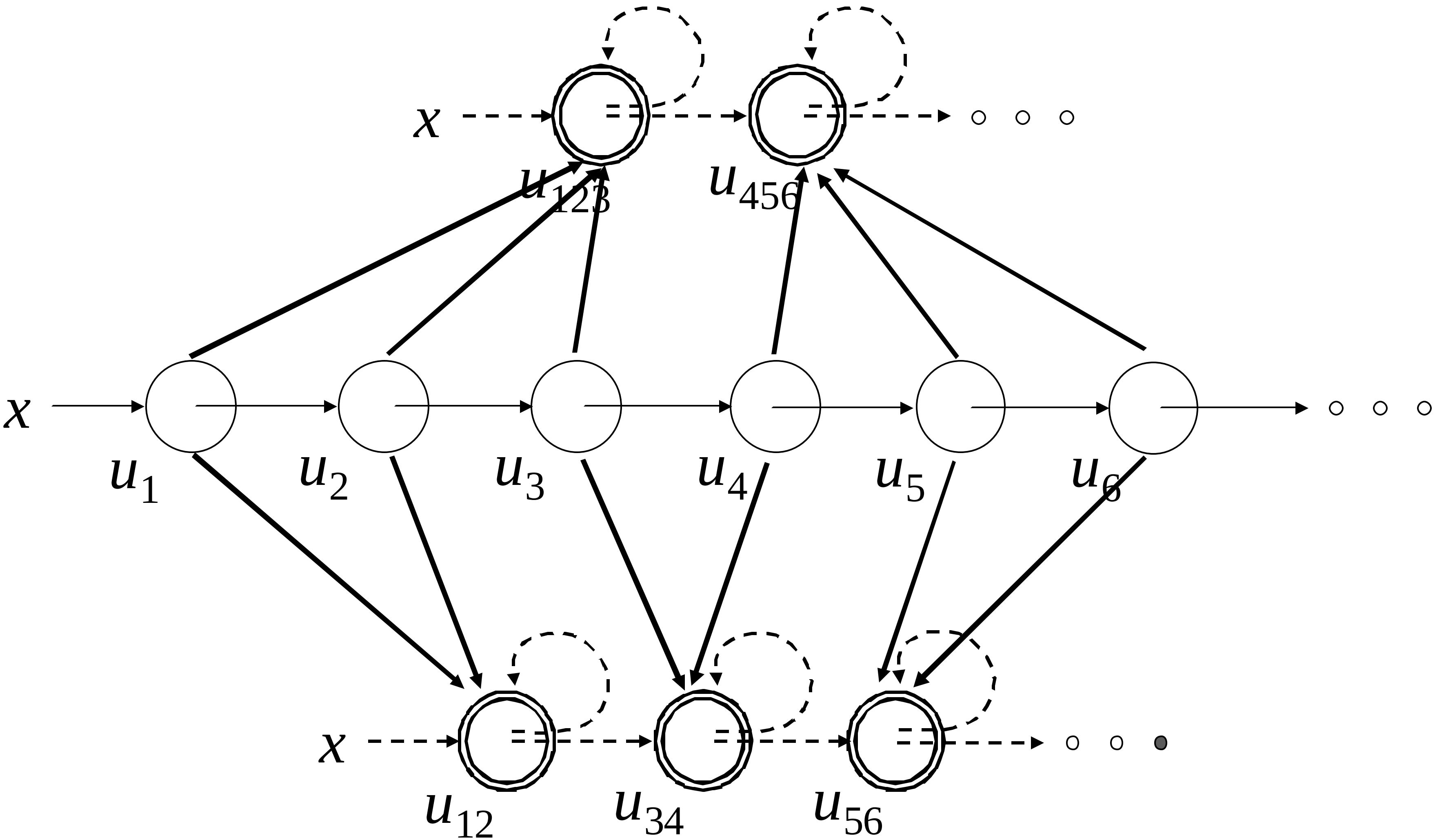




Summary

- Canonical abstraction guarantees finite number of structures
- The concrete location of an object plays no significance
- But what is the significance of 3-valued logic?

Embedding



Embedding

- $B \sqsubseteq^{f} S$
  - onto function f
  - $p^B(u_1, .., u_k) \sqsubseteq p^S(f(u_1), ..., f(u_k))$
- S is a tight embedding of B with respect to f if:
  - $p^S(u^\#_1, .., u^\#_k) = \sqcup\{p^B(u_1 ..., u_k) \mid f(u_1)=\mathrm{u}^\#_1, ..., f(u_k)=\mathrm{u}^\#_k\}$
- Canonical Abstraction is a tight embedding

Embedding (cont)

- $S_1 \sqsubseteq_f S_2 \Leftrightarrow$ every concrete state represented by S_1 is also represented by S_2
- The set of nodes in S_1 and S_2 may be different
  - No meaning for node names (abstract locations)
- $\gamma(S^\#)= \{S : \text{2-valued structure } S, S \sqsubseteq_f S^\#\}$

Embedding Theorem

- Assume B $\sqsubseteq^f$ S,

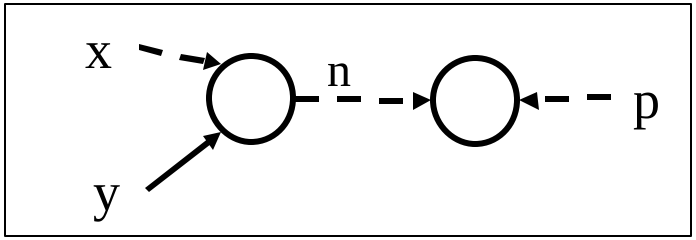
$$p^B(u_1, .., u_k) \sqsubseteq p^S(f(u_1), ..., f(u_k))$$

- Then every formula φ is preserved:
  - If $[\![\varphi]\!] = 1$ in S, then $[\![\varphi]\!] = 1$ in B
  - If $[\![\varphi]\!] = 0$ in S, then $[\![\varphi]\!] = 0$ in B
  - If $[\![\varphi]\!] = 1/2$ in S, then $[\![\varphi]\!]$ could be 0 or 1 in B

Embedding Theorem

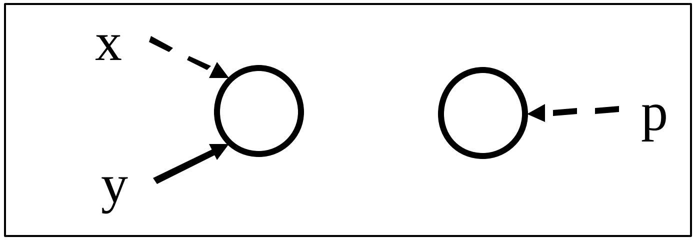
- For every formula φ is preserved:
  - If $\llbracket \varphi \rrbracket = 1$ in *S*, then $\llbracket \varphi \rrbracket = 1$ for all *B Î g(S)*
  - If $\llbracket \varphi \rrbracket = 0$ in *S*, then $\llbracket \varphi \rrbracket = 0$ for all *B Î g(S)*
  - If $\llbracket \varphi \rrbracket = 1/2$ in *S*, then $\llbracket \varphi \rrbracket$ could be 0 or 1 in *g(S)*

Challenge 2 - Destructive Update



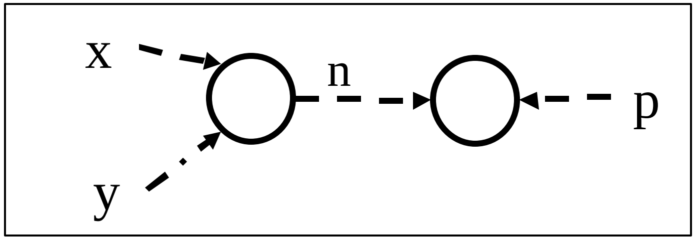
$y \rightarrow next = NULL$

$n'(v, w) = \neg y(v) \wedge n(v, w)$



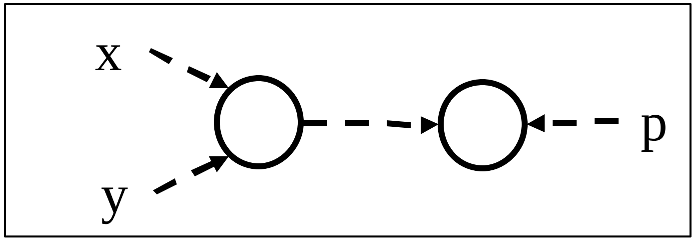
Sound ☺

Challenge 2 - Destructive Update



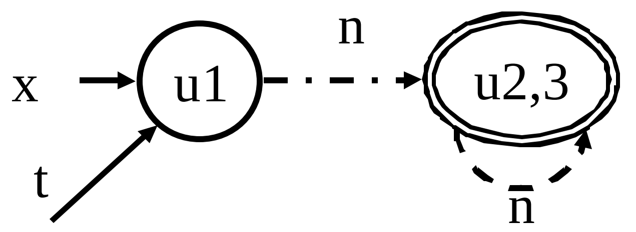
y→next = NULL

$$n'(v, w) = \neg\, y(v) \wedge n(v, w)$$



Sound ☺

Embedding Theorem



$\exists v: x(v)$	1=Yes
$\exists v: x(v) \wedge t(v)$	1=Yes
$\exists v: x(v) \wedge y(v)$	0=No
$\exists v,w: x(v) \wedge n(v, w)$	½=Maybe
$\exists v, w: x(v) \wedge n(v, w) \wedge n^{+}(w, v)$	0=No
$\exists \mathit{v},w: x(\mathit{v}) \wedge \mathit{n}^{*}(\mathit{v},w) \wedge n^{+}(w, w)$	1/2=Maybe

Summary

- The embedding theorem eliminates the need for proving near commutavity
- Guarantees soundness
- Applied to arbitrary logics
- But can be imprecise

Limitations

- Information on summary nodes is lost
- Leads to useless verification

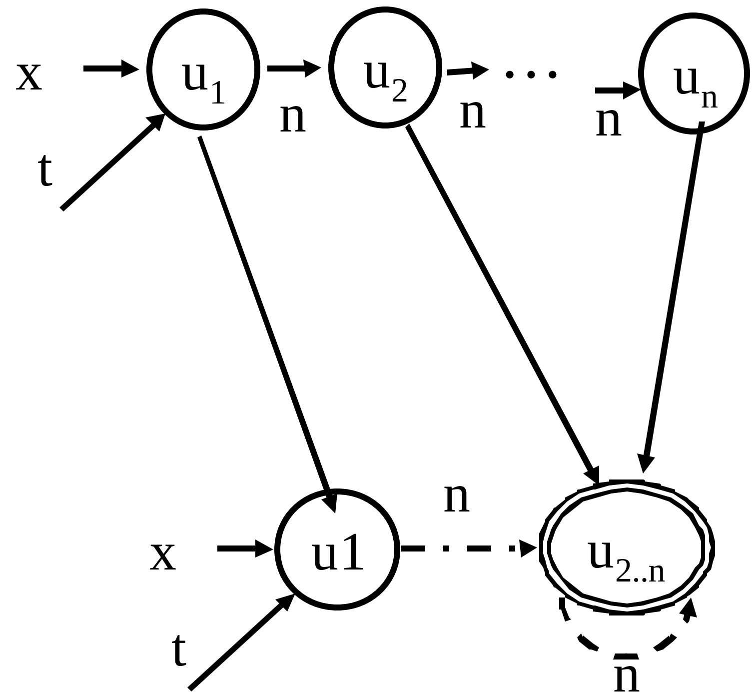
Increasing Precision

- User (Programming Language) supplied global invariants
  - Naturally expressed in FO^{TC}
- Record extra information in the concrete interpretation
  - Tune the abstraction
  - Refine concretization

Cyclicity predicate

$$c[x]() = \exists v_1,v_2\colon x(v_1) \wedge n^*(v_1,v_2) \wedge n^+(v_2, v_2)$$

$\boxed{c[x]()=0}$

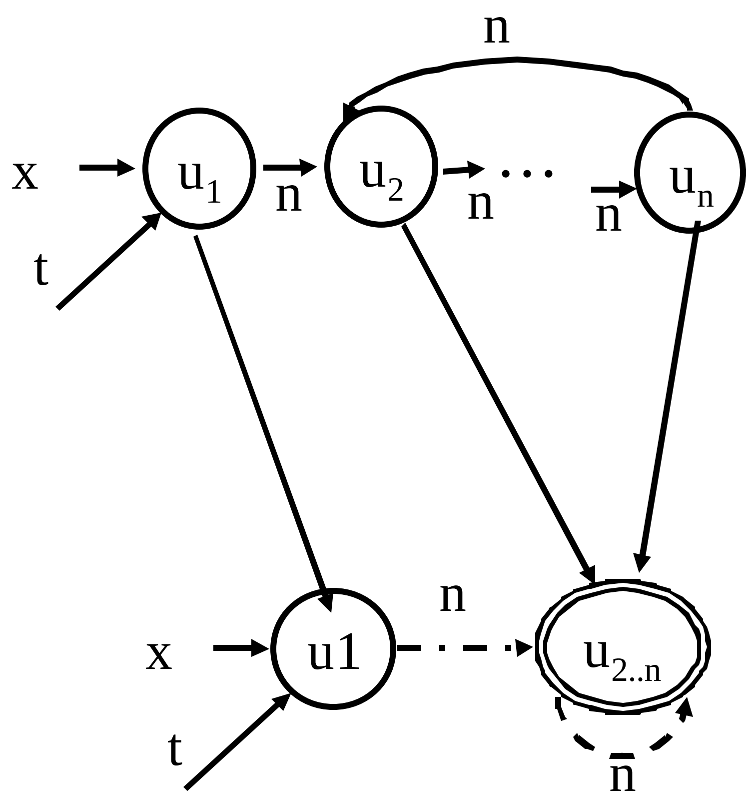


$\boxed{c[x]()=0}$

Cyclicity predicate

$$c[x]() = \exists v_1,v_2 \colon x(v_1) \wedge n^*(v_1,v_2) \wedge n^+(v_2, v_2)$$

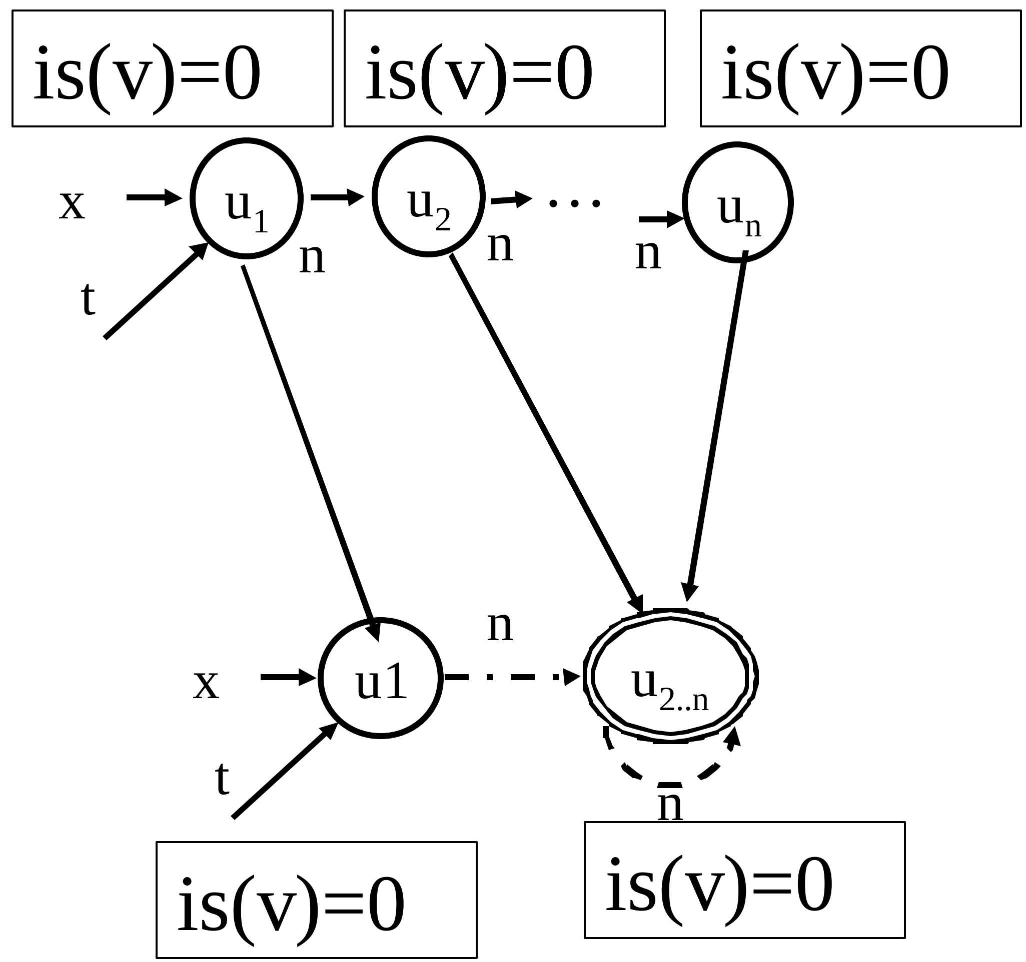
$\boxed{c[x]()=1}$



$\boxed{c[x]()=1}$

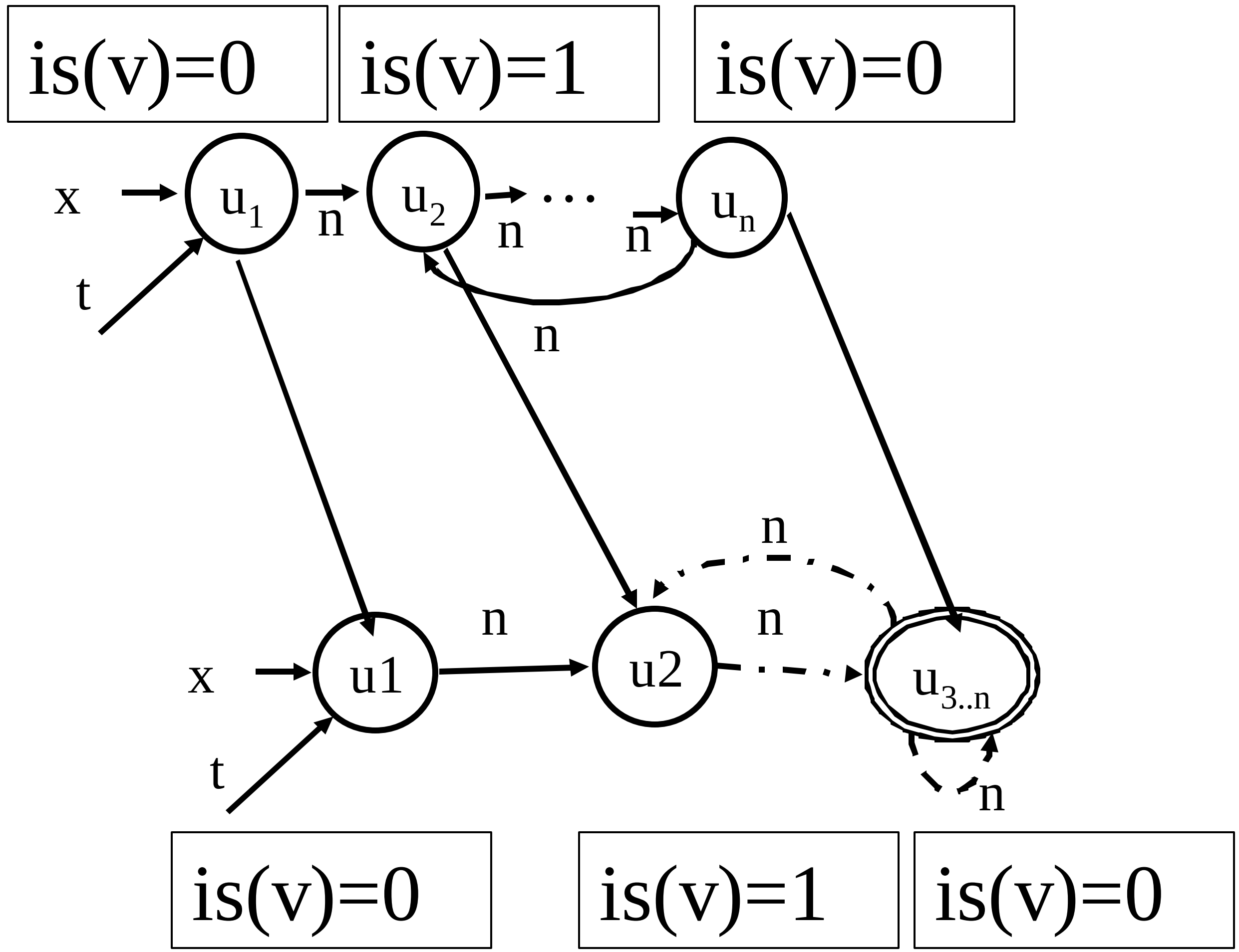
Heap Sharing predicate

$$is(v) = \exists v_1,v_2\colon n(v_1,v) \wedge n(v_2,v) \wedge v_1 \neq v_2$$



Heap Sharing predicate

$$is(v) = \exists v_1,v_2 \colon n(v_1,v) \wedge n(v_2,v) \wedge v_1 \neq v_2$$

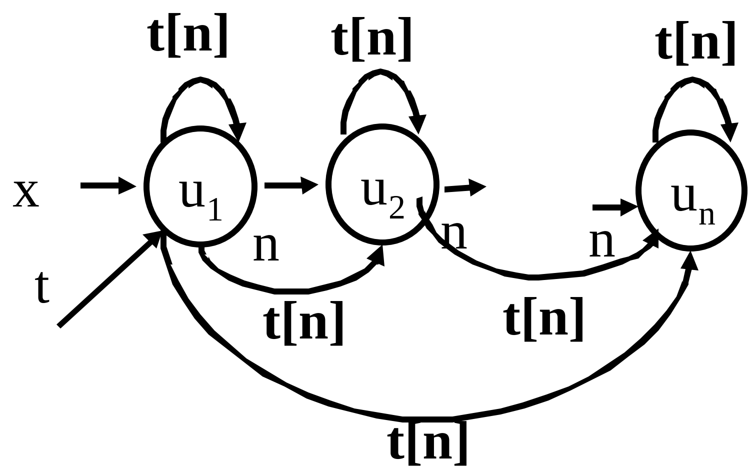


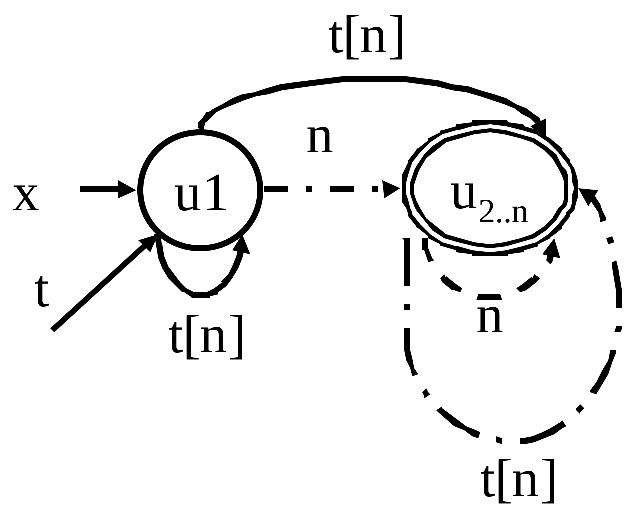
Concrete Interpretation Rules

Statement	Update formula
x =NULL	$x'(v) = 0$
x= malloc()	$x'(v) = IsNew(v)$ $is'(v) = is(v) \land \neg IsNew(v)$
x=y	$x'(v) = y(v)$
x=y $\rightarrow$next	$x'(v) = \exists w: y(w) \land n(w, v)$
x $\rightarrow$next=NULL	$n'(v, w) = \neg x(v) \land n(v, w)$ $is'(v) = is(v) \land$ $\exists v1, v2: n(v1, v) \land n(v2, v) \land$ $\neg x(v1) \land \neg x(v2) \land \neg eq(v1, v2)$

Reachability predicate

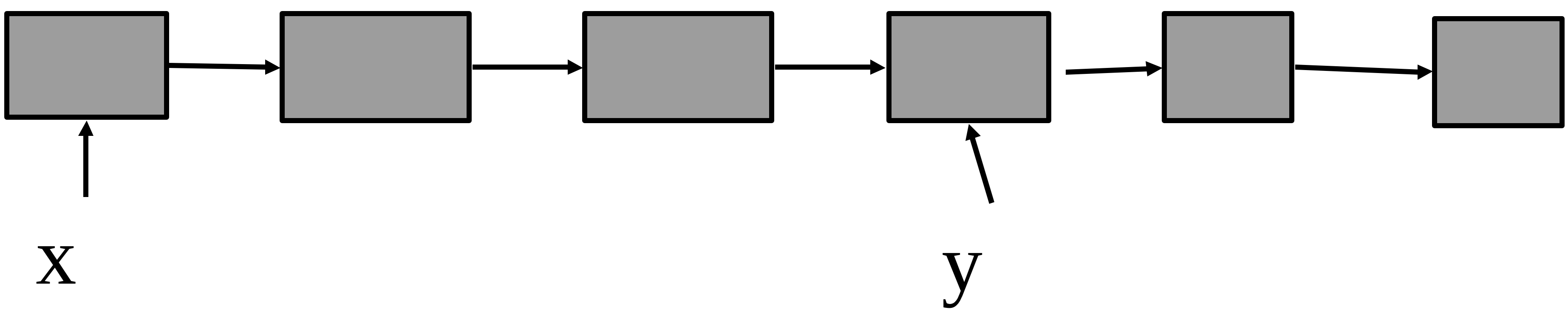
$$t[n](v1, v2) = n^*(v1,v2)$$





Reachability from a variable

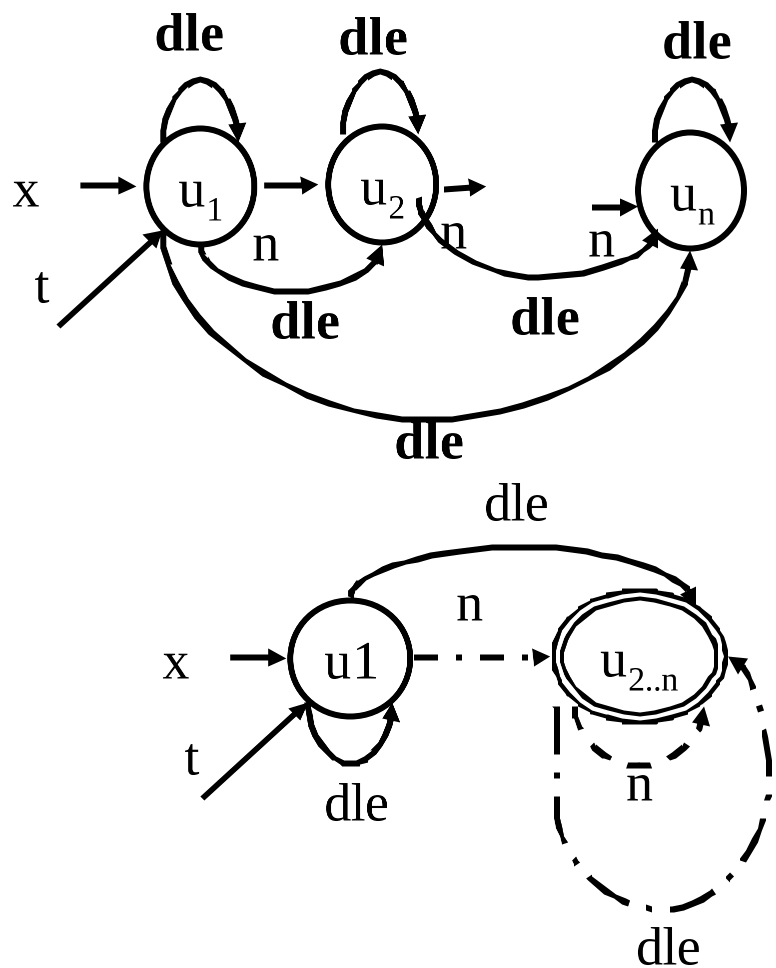
- $r[n,x](v) = \exists w: x(w) \wedge n^*(w, v)$



Proving Correctness of Sorting Implementations (Lev-Ami, Reps, S, Wilhelm ISSTA 2000)

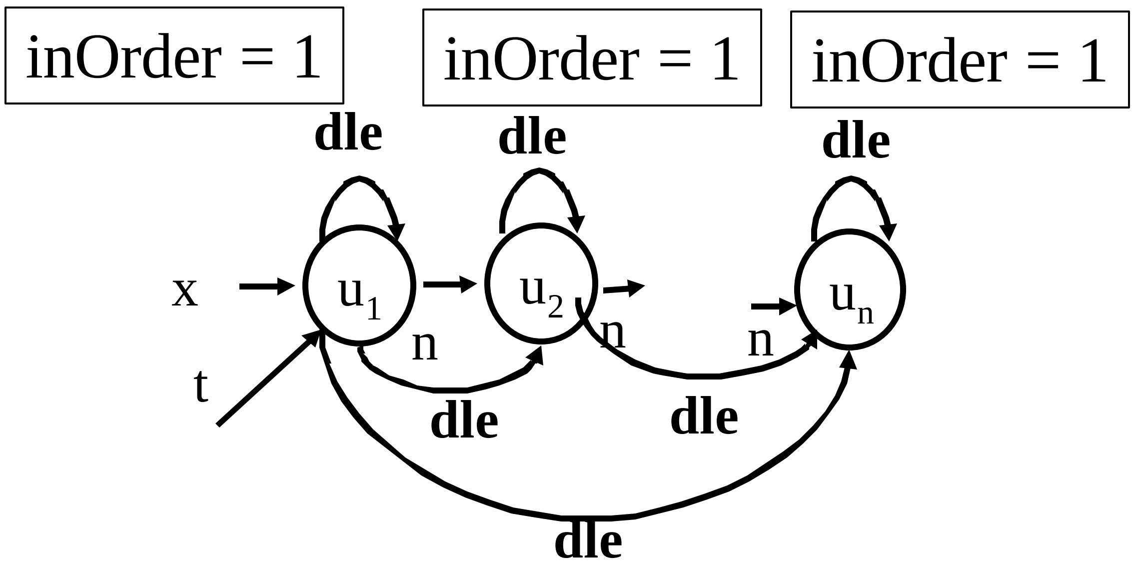
- Partial correctness
  - The elements are sorted
  - The list is a permutation of the original list
- Termination
  - At every loop iterations the set of elements reachable from the head is decreased

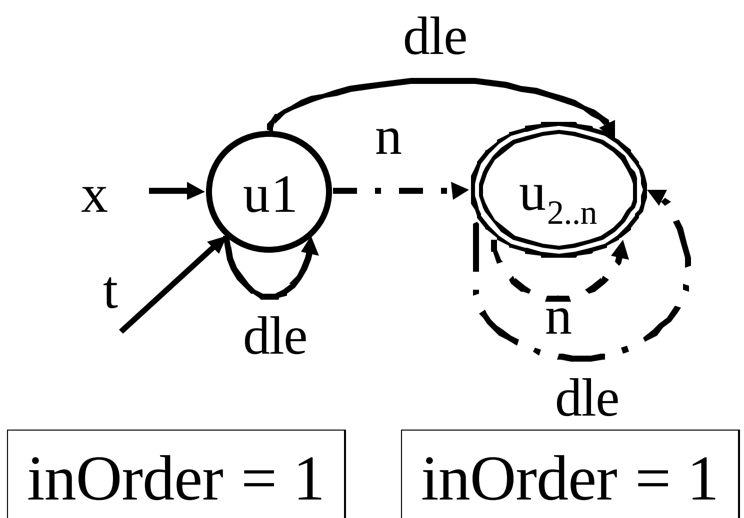
Sortedness


dle
dle
dle
x
u_1
u_2
u_n
n
n
n
t
dle
dle
dle
dle
n
x
u1
$u_{2..n}$
t
$\bar{n}$
dle
dle

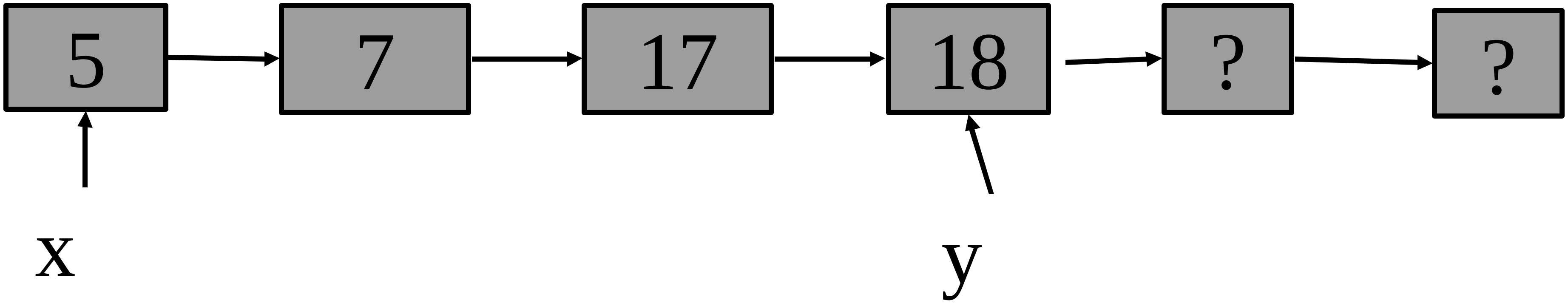
Example: Sortedness

$$\mathrm{inOrder}(v) = \forall v1\colon n(v,v1) \rightarrow \mathrm{dle}(v, v1)$$





Sortedness



Additional Instrumentation predicates

- $c_{fb}(v) = \forall v_1: f(v, v_1) \rightarrow b(v_1, v)$
- tree(v)
- avl(v)
- dag(v)
- Weakest Precondition [Ramalingam PLDI'02]
- Learned via Inductive Logic Programming [Loginov, CAV'05]

Instrumentation (Summary)

- Refines the abstraction

$$is(v) = \exists v_1,v_2: n(v_1,v) \wedge n(v_2,v) \wedge v_1 \neq v_2$$

- Adds global invariants

$$is(v) \leftrightarrow \exists v_1,v_2: n(v_1,v) \wedge n(v_2,v) \wedge v_1 \neq v_2$$

$$\gamma(S^\#)=\{S : S \models \Sigma, S \sqsubseteq^f S^\#\}$$

- But requires update-formulas (generated automatically in TVLA2)

Summary

- Canonical abstraction is powerful
  - Intuitive
  - Adapts to the property of interest
- Used to verify interesting program properties
  - Very few false alarms
- But scaling is an issue