THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

An Example of TVLA

Three Valued Logic Analyzer

Presenter: **Chengpeng Wang**
Date: Oct 30th , 2019
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology

Background: What is shape analysis?

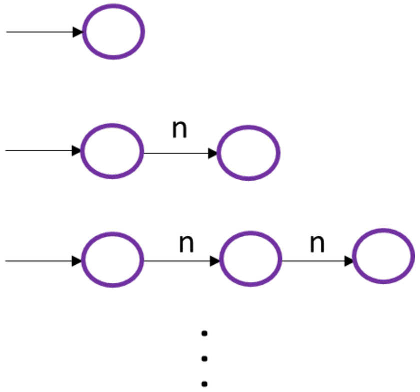
- Definition in [Jones and Muchnick 1981]
    - Determine the possible shapes of a dynamically allocated data structure at a given program point
- Reason the geometry structures of dynamically allocated heap data and their relations
    - Geometry structures
        - Is it a Tree, a DAG, or a Cyclic Graph?
        - Self-defined properties: sorted linked lists …
    - Structural relationship
        - Overlapping or disjoint?
- In general, shape analysis aims to property reasoning on heap structures

TVLA

- Motivations
  - The expressivity is limited
    - Generate shape invariants by predicates
  - The approach is not general
    - Propose a parametric approach/framework to synthesize different kinds of shape invariants

TVLA

- Challenges
  - Dynamically allocated data structures are unbounded



  - The trade-off between precision and efficiency
    - More predicates are used, more precise shape information extracted while more overhead in shape analysis

TVLA I: Intraprocedural

- Problem 1: How to abstract
- Problem 2: How to use predicates

Shape Abstraction

- Problem 3: How to embed

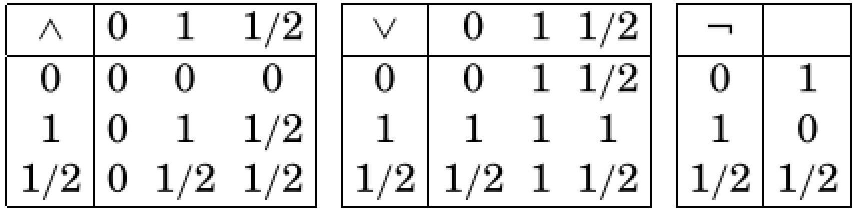
Canonical Embedding

- Problem 4: How to update formula according to the statement
  - Focus and Coerce

Formula Update

Shape Abstraction

- Model shape info in 3-valued logic
  - S: logical structure, denoted by $< U^S,\ l^S >$
    - U^S: A universe of individuals
    - l^S maps arity-k predicate and k-tuple of individuals to 0(false), 1(true) or 1/2(unknown)



$\wedge$	0	1	1/2
0	0	0	0
1	0	1	1/2
1/2	0	1/2	1/2

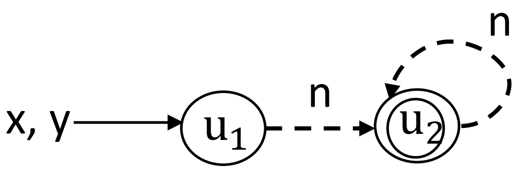
$\vee$	0	1	1/2
0	0	1	1/2
1	1	1	1
1/2	1/2	1	1/2

$\neg$	
0	1
1	0
1/2	1/2

- Example

Encode shape graph in a logical way

    - $sm(v)$: Does v represent more than one concrete individuals?
    - $q(n)$: Does pointer variable q point to element n?
    - $n(v_1, v_2)$: Does the n field of v_1 point to v_2?



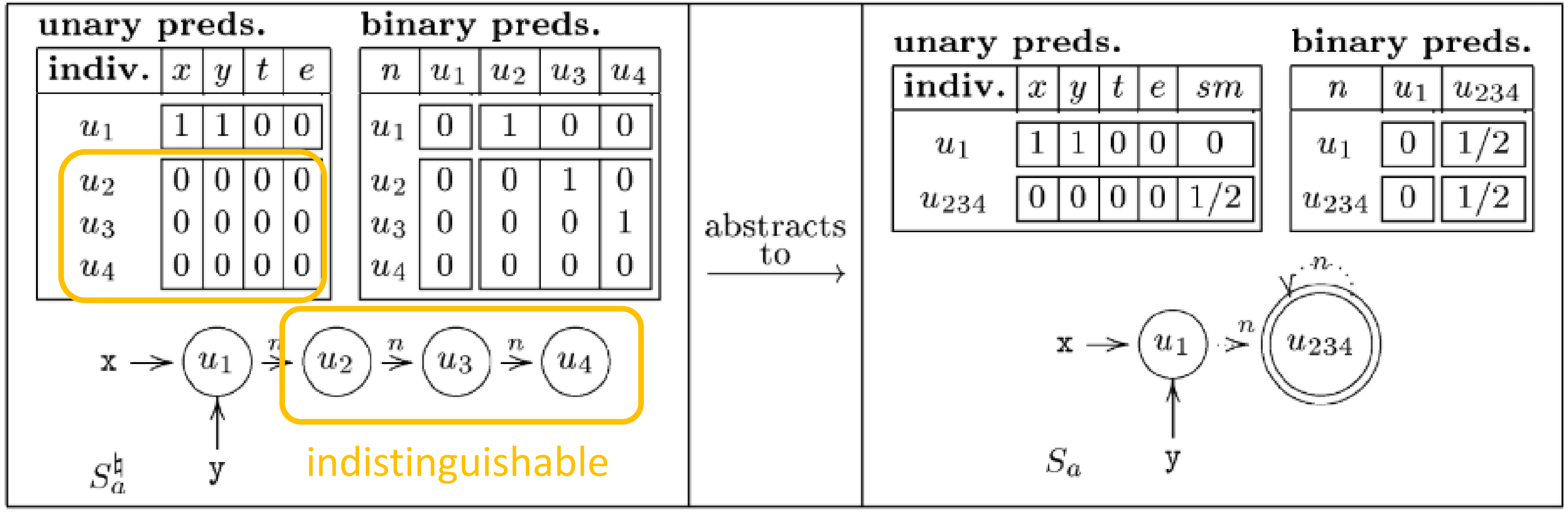
Indiv.	x	y	sm
u_1	1	1	0
u_2	0	0	1/2

n	u_1	u_2
u_1	0	**1/2**
u_2	0	**1/2**

Sagiv M, Reps T, Wilhelm R. **Parametric shape analysis via 3-valued logic**[J]. TOPLAS 2002

Canonical Embedding

- Most precise embedding $f_{embed_c}(v_1) = f_{embed_c}(v_2) \;\Leftrightarrow p(v_1) = p(v_2) \;\; \forall p \in A$
  - Abstraction predicates: $A = \{x, y, t, e\}$



unary preds.

indiv.	x	y	t	e
u_1	1	1	0	0
u_2	0	0	0	0
u_3	0	0	0	0
u_4	0	0	0	0

binary preds.

n	u_1	u_2	u_3	u_4
u_1	0	1	0	0
u_2	0	0	1	0
u_3	0	0	0	1
u_4	0	0	0	0

unary preds.

indiv.	x	y	t	e	sm
u_1	1	1	0	0	0
u_{234}	0	0	0	0	1/2

binary preds.

n	u_1	u_{234}
u_1	0	1/2
u_{234}	0	1/2

Sagiv M, Reps T, Wilhelm R. **Parametric shape analysis via 3-valued logic**[J]. TOPLAS 2002

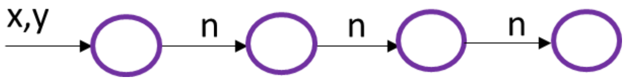
Formula Update

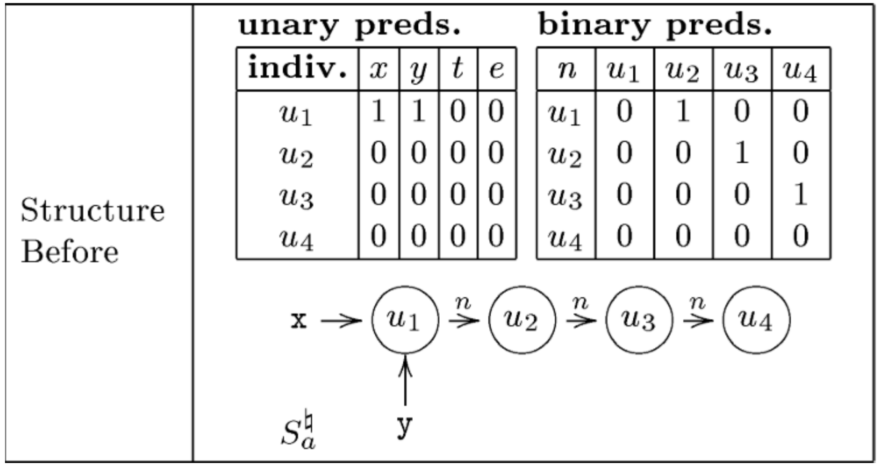
- An example
  - Statement y = y->n

```
void insert(List x, int d) {
    List y, t, e;
    assert(acyclic_list(x) && x != NULL);
    y = x;
    while (y->n != NULL && ...) {
        y = y->n;
    }
    ...
}
```

Predicate Update Formulae	$x'(v) = x(v)$ $y'(v) = \exists v_1 : y(v_1) \wedge n(v_1, v)$ $t'(v) = t(v)$ $e'(v) = e(v)$ $n'(v_1, v_2) = n(v_1, v_2)$

  - 2-valued Logic structure





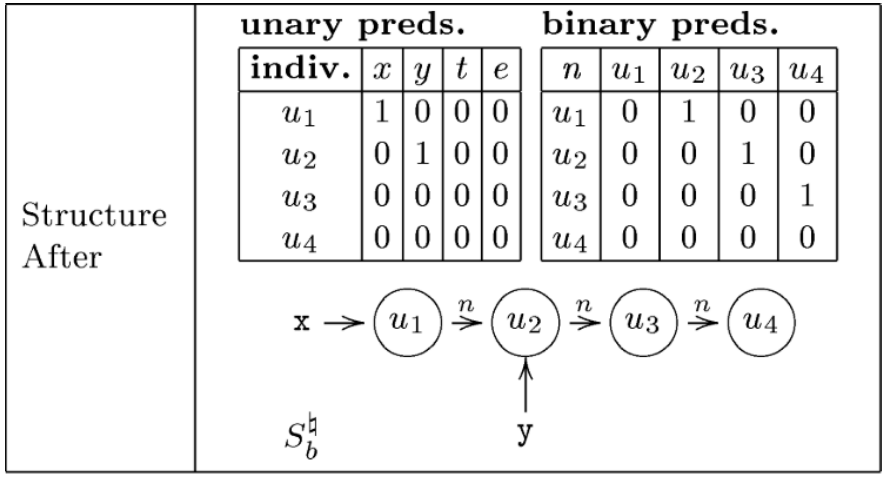
Structure Before

unary preds.

indiv.	x	y	t	e
u_1	1	1	0	0
u_2	0	0	0	0
u_3	0	0	0	0
u_4	0	0	0	0

binary preds.

n	u_1	u_2	u_3	u_4
u_1	0	1	0	0
u_2	0	0	1	0
u_3	0	0	0	1
u_4	0	0	0	0



Structure After

unary preds.

indiv.	x	y	t	e
u_1	1	0	0	0
u_2	0	1	0	0
u_3	0	0	0	0
u_4	0	0	0	0

binary preds.

n	u_1	u_2	u_3	u_4
u_1	0	1	0	0
u_2	0	0	1	0
u_3	0	0	0	1
u_4	0	0	0	0

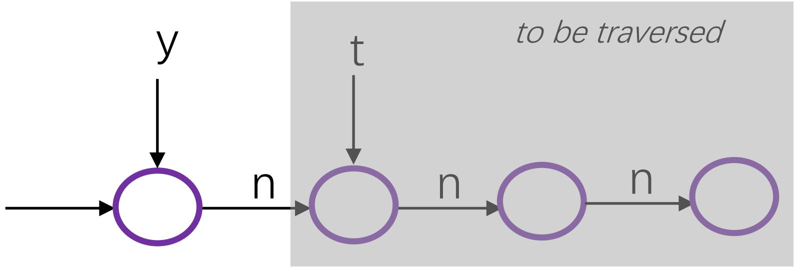
Sagiv M, Reps T, Wilhelm R. **Parametric shape analysis via 3-valued logic**[J]. TOPLAS 2002

An Application of TVLA: Free Analysis

Free analysis

It is safe to free the node pointed by y after line 10

```
         public static void main(String args[])
         {
             L x, y, t;
             x = null;
             while (...) {
                 y = new L();
                 y.n = x;
                 x = y;
             }
             y = x;
             while (y != null) {
Line 10          t = y.n;
                 y = t;
             }
         }
```

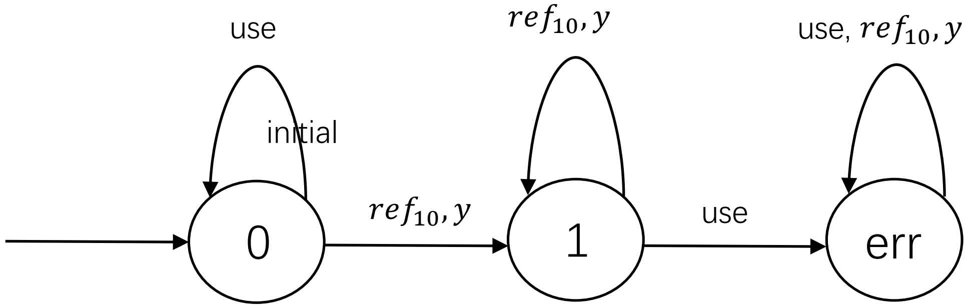


Use Heap Safety Automata to track *Use* state

```
public static void main(String args[])
{
    L x, y, t;
    x = null;
    while (...) {
        y = new L();
        y.n = x;
        x = y;
    }
    y = x;
    while (y != null) {
        t = y.n;
        y = t;
    }
}
```

Line 10

statement	use events are triggered for an object referenced by
x = y	y
x = y.f	$y, y.f$
x.f = null	x
x.f = y	x, y
x binop y	x, y



Example: the HSA of y at line 10

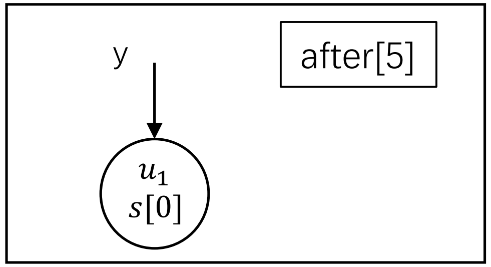
Accepting state: {0,1}

Loop 1

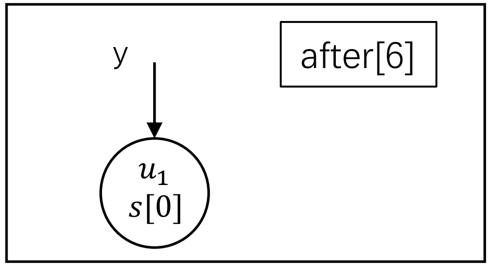
```
public static void main(String args[])
{
    L x, y, t;
    x = null;
    while (...) {
        y = new L();
        y.n = x;
        x = y;
    }
    y = x;
    while (y != null) {
        t = y.n;
        y = t;
    }
}
```

Loop 1, iteration 1

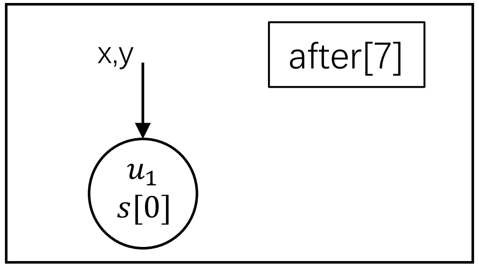
```
y = new L();
y.n = x;
x = y;
```

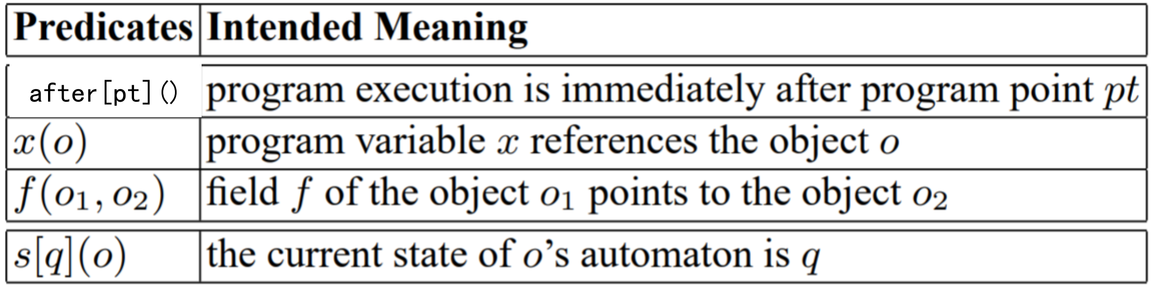


$$n'(u,v) = (y(u) \wedge x(v)) \vee n(u,v)$$



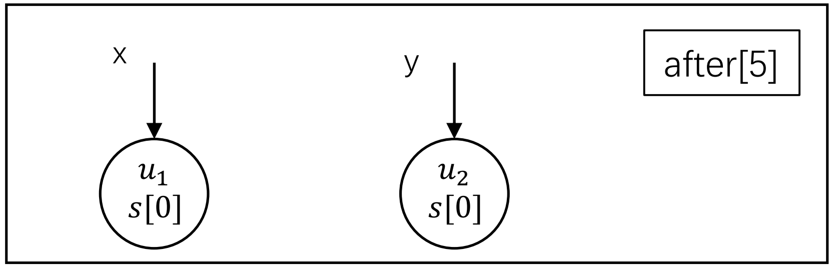
$$x'(u) = y(u)$$



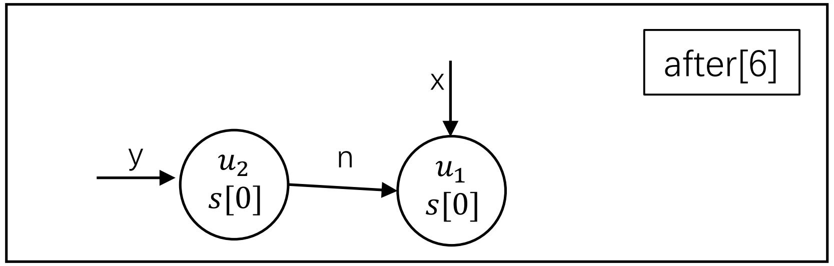


Predicates	**Intended Meaning**
after[pt]()	program execution is immediately after program point *pt*
$x(o)$	program variable x references the object o
$f(o_1, o_2)$	field f of the object o_1 points to the object o_2
$s[q](o)$	the current state of o's automaton is q

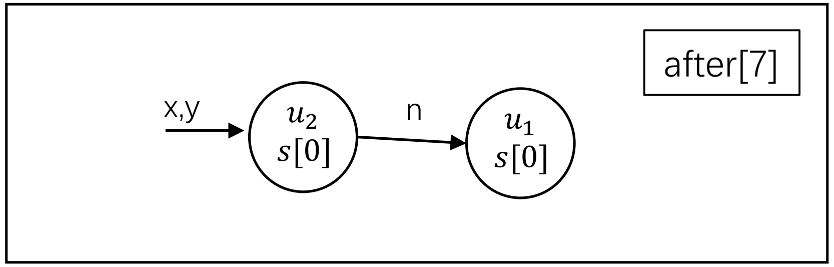
Loop 1, iteration 2



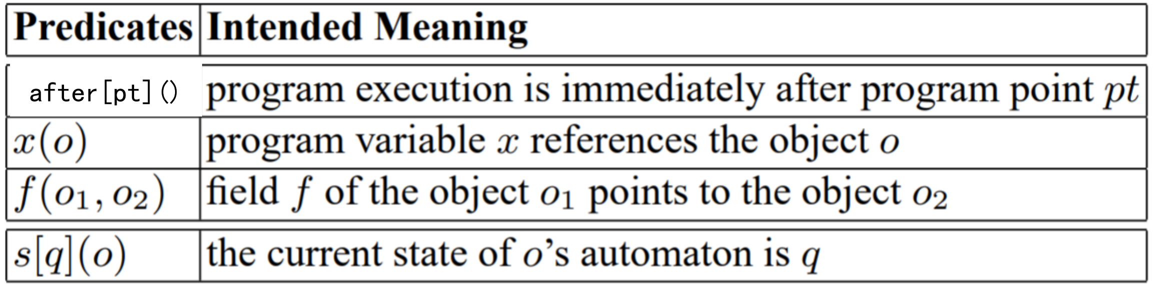
$$n'(u,v) = (y(u) \land x(v)) \lor n(u,v)$$



$$x'(u) = y(u)$$

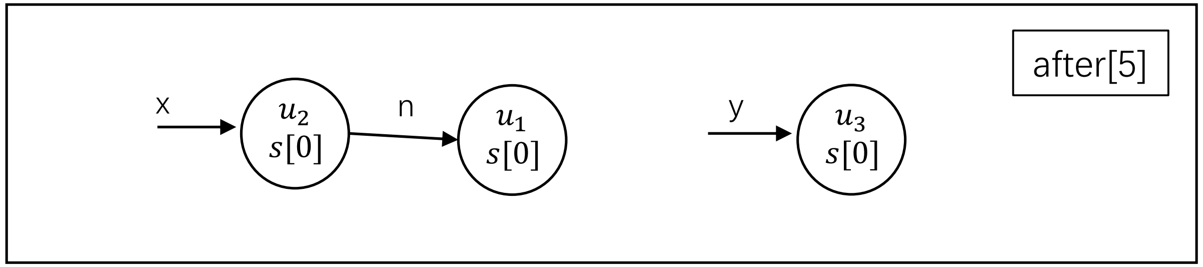


```
y = new L();
y.n = x;
x = y;
```



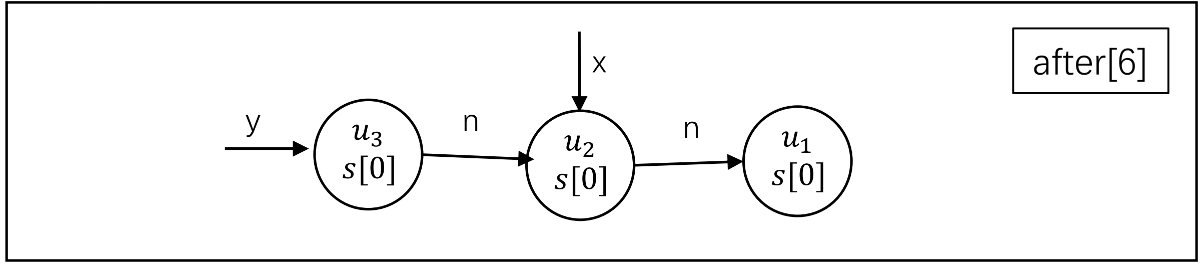
Predicates	**Intended Meaning**
`after[pt]()`	program execution is immediately after program point pt
$x(o)$	program variable x references the object o
$f(o_1, o_2)$	field f of the object o_1 points to the object o_2
$s[q](o)$	the current state of o's automaton is q

Loop 1, iteration 3

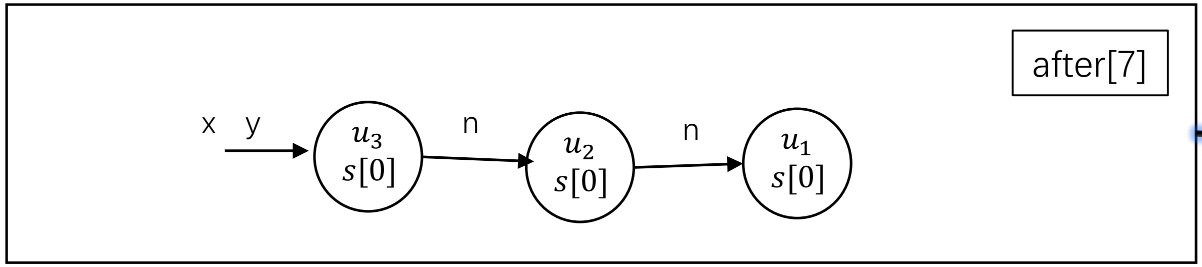


```
y = new L();
y.n = x;
x = y;
```

$$n'(u,v) = (y(u) \land x(v)) \lor n(u,v)$$



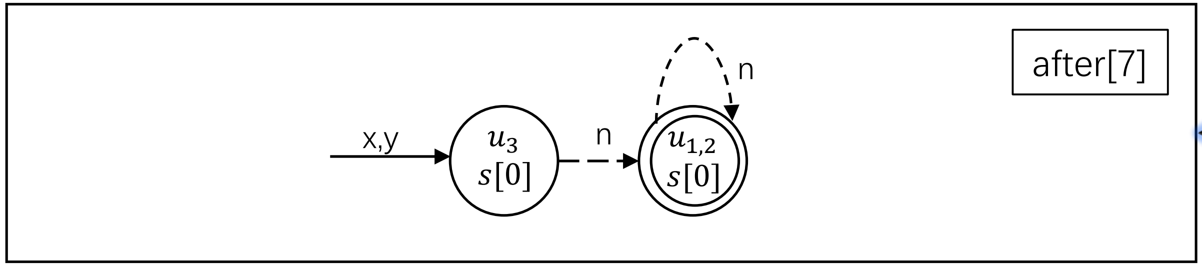
$$x'(u) = y(u)$$



Abstract Predicates

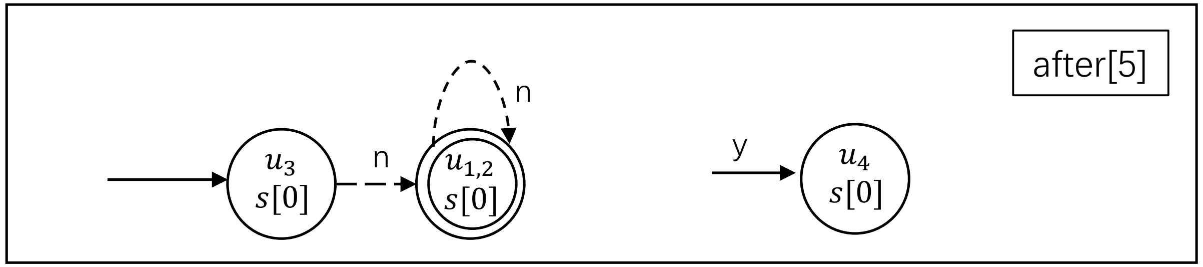
$x(u) \quad y(u) \quad r_{x\,n}(u) \quad r_{y\,n}(u)$

Canonical abstraction

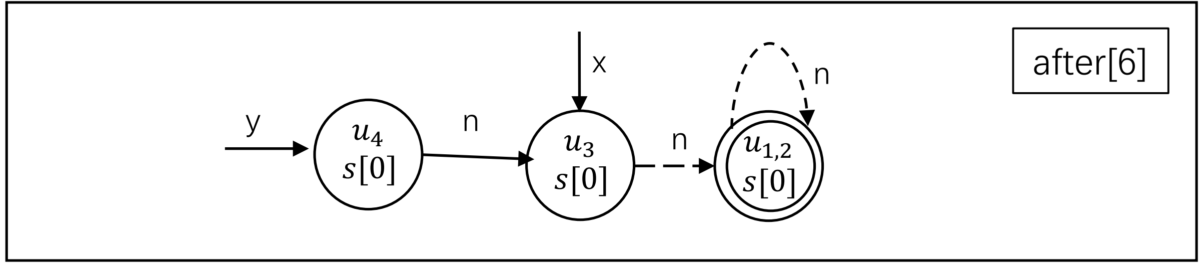


Loop 1, iteration 4

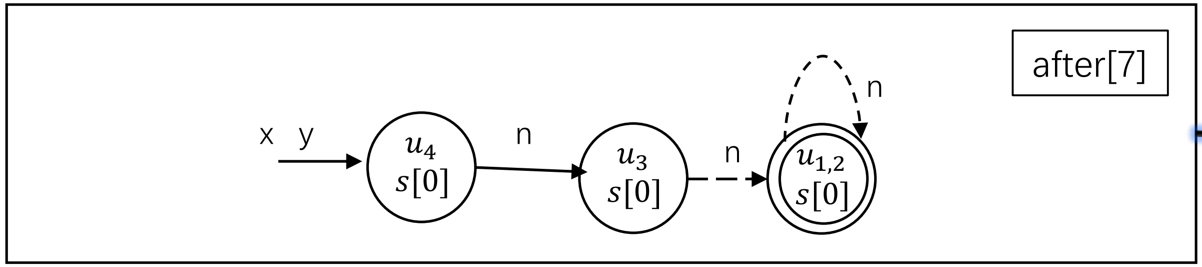
```
y = new L();
y.n = x;
x = y;
```



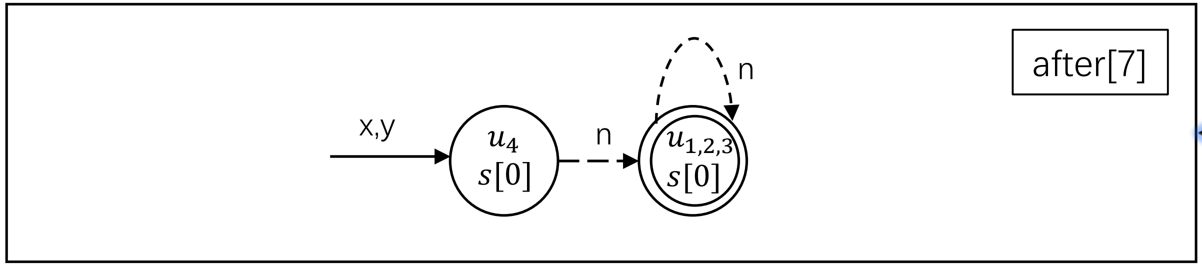
$$n'(u,v) = (y(u) \land x(v)) \lor n(u,v)$$



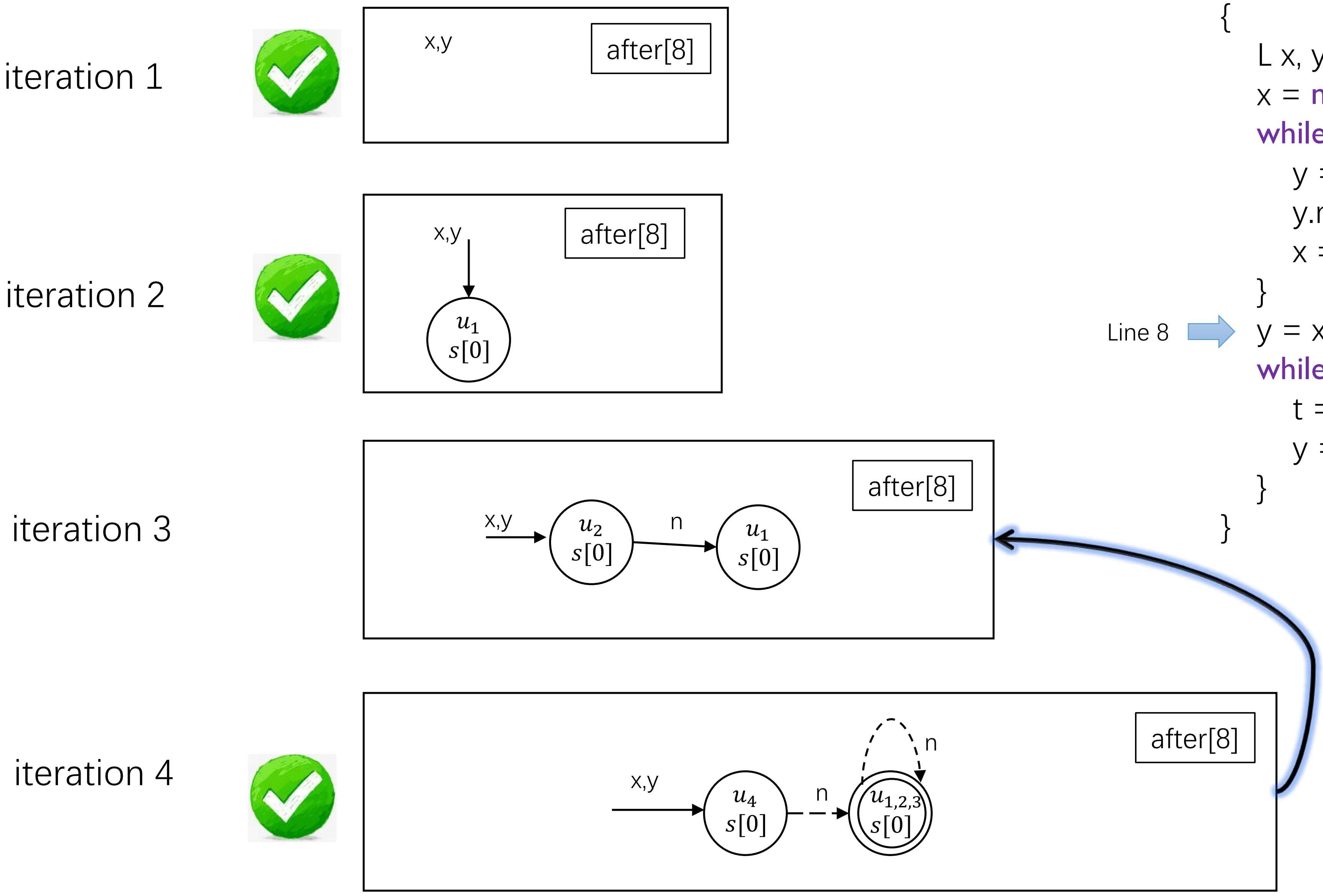
$$x'(u) = y(u)$$



Canonical abstraction



Jump out of Loop 1, after lc 8



Line 8 → `y = x;`

```
public static void main(String args[])
{
    L x, y, t;
    x = null;
    while (...) {
        y = new L();
        y.n = x;
        x = y;
    }
    y = x;
    while (y != null) {
        t = y.n;
        y = t;
    }
}
```

Loop 2

```
public static void main(String args[])
{
    L x, y, t;
    x = null;
    while (...) {
        y = new L();
        y.n = x;
        x = y;
    }
    y = x;
    while (y != null) {
        t = y.n;
        y = t;
    }
}
```

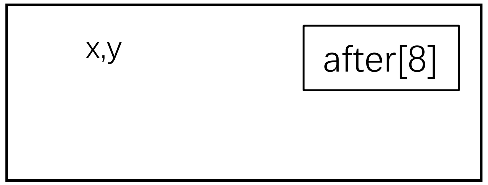
Loop 2, iteration 1, program location 9

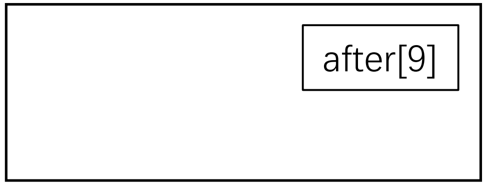
y != null

Constraint:

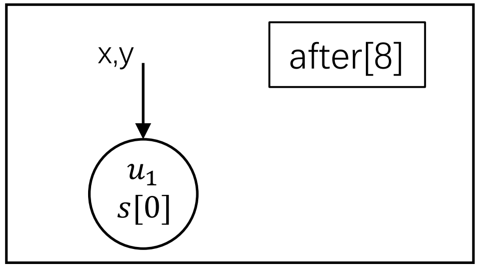
$\exists u\ y(u)$

✓

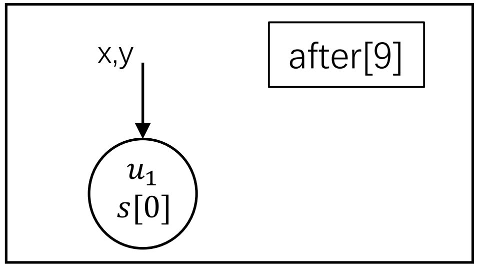




✓

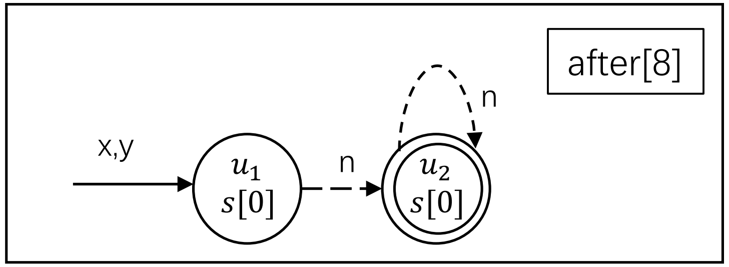


✓

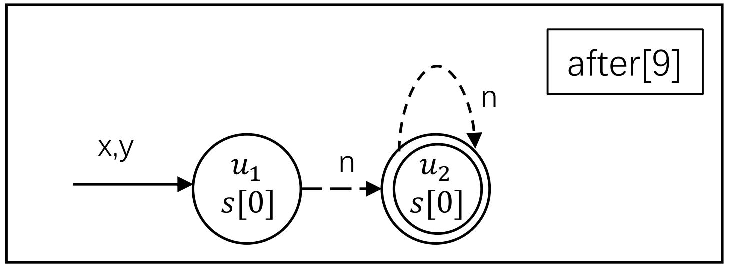


Case 1

✓

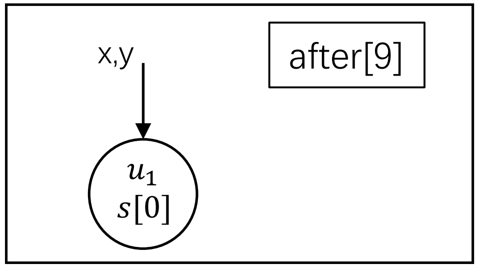


✓

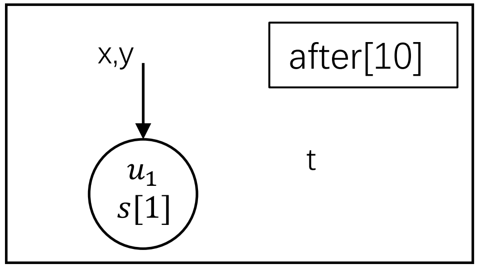


Case 2

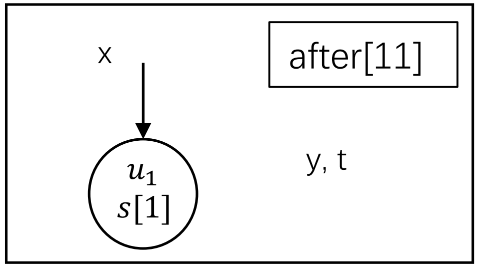
Loop 2, iteration 1, case 1



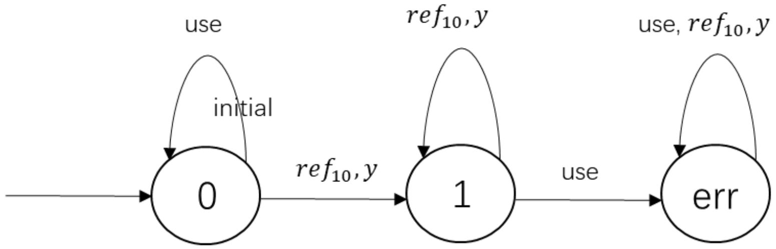
$$t'(u) = \exists y(v) \wedge n(v,u)$$



$$y'(u) = t(u)$$



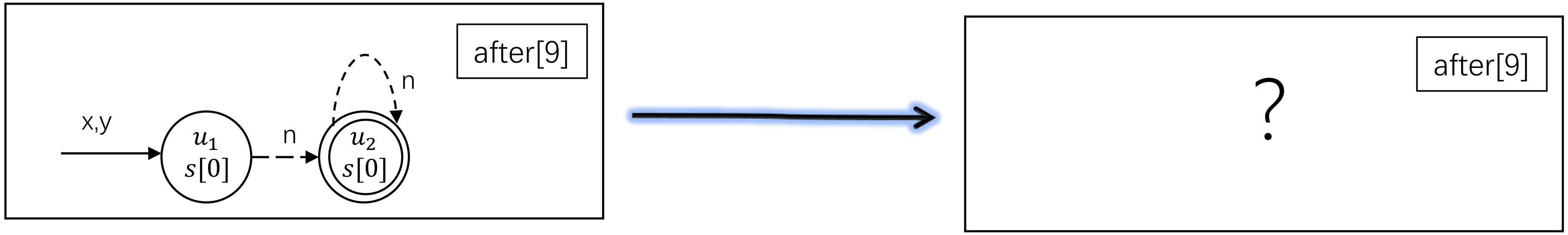
```
while (y != null) {
    t = y.n;
    y = t;
}
```



Example: the HSA of y at line 10

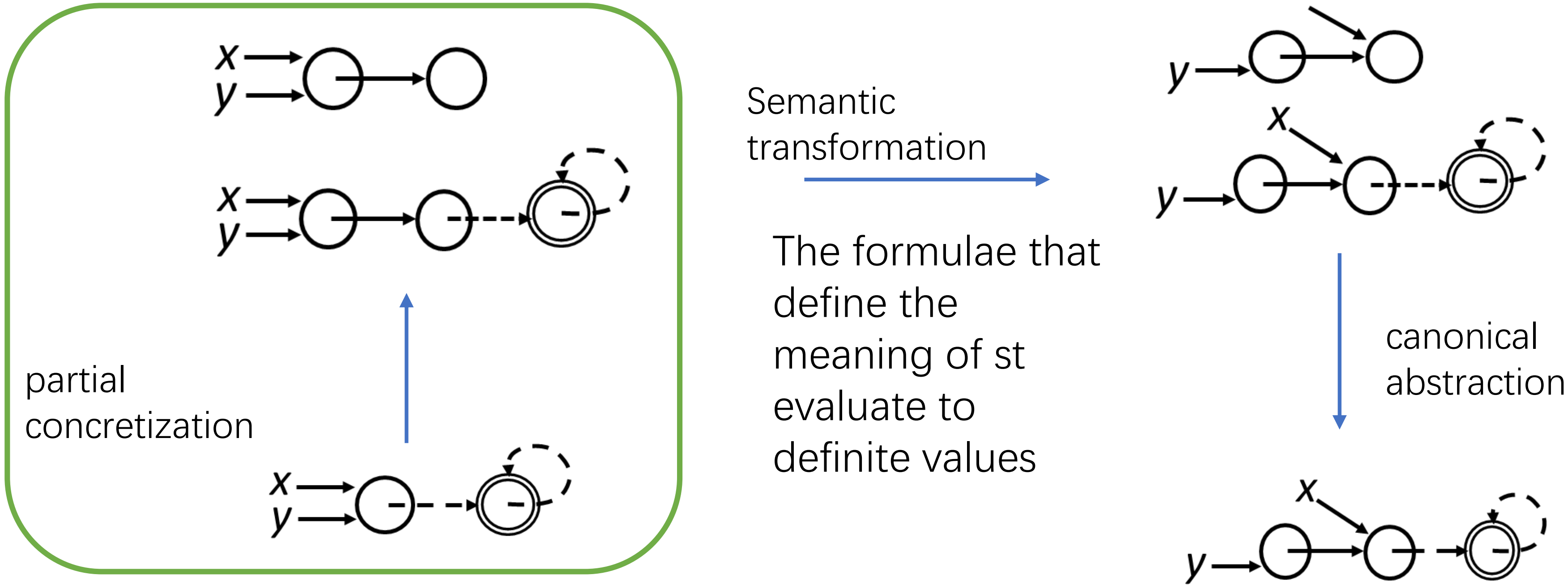
Accepting state: {0,1}

Loop 2, iteration 1, case 2 t = y.n



Formula Update

- Recover the truth: Focus



partial concretization

Semantic transformation

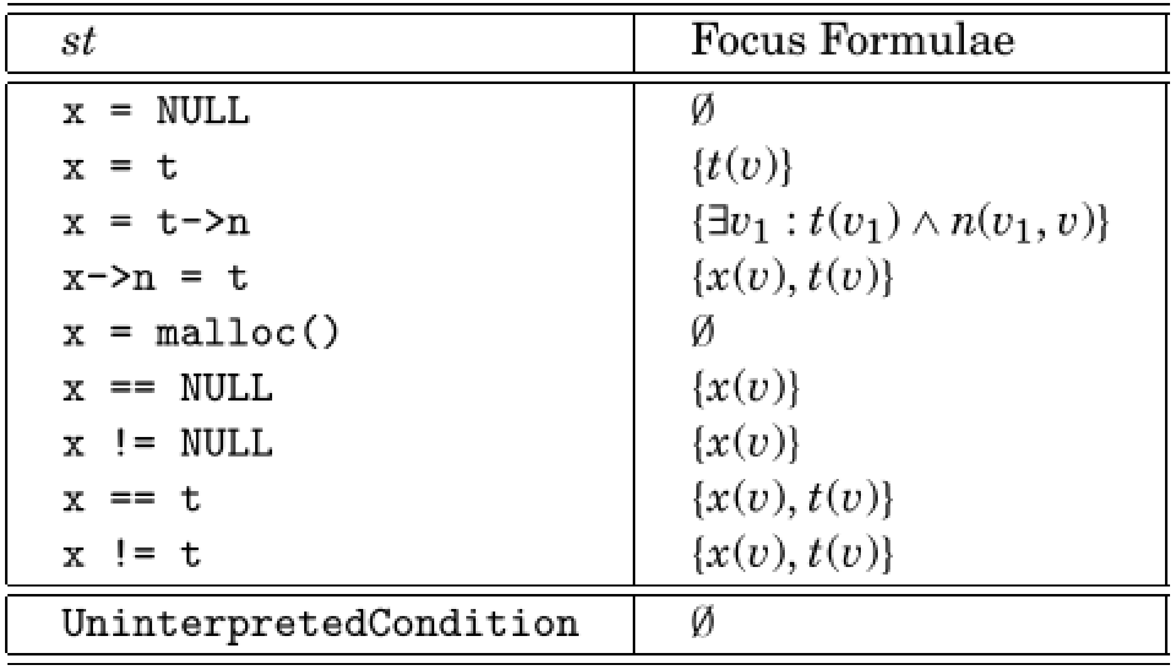
The formulae that define the meaning of st evaluate to definite values

canonical abstraction

Sagiv M, Reps T, Wilhelm R. **Parametric shape analysis via 3-valued logic**[J]. TOPLAS 2002

Formula Update

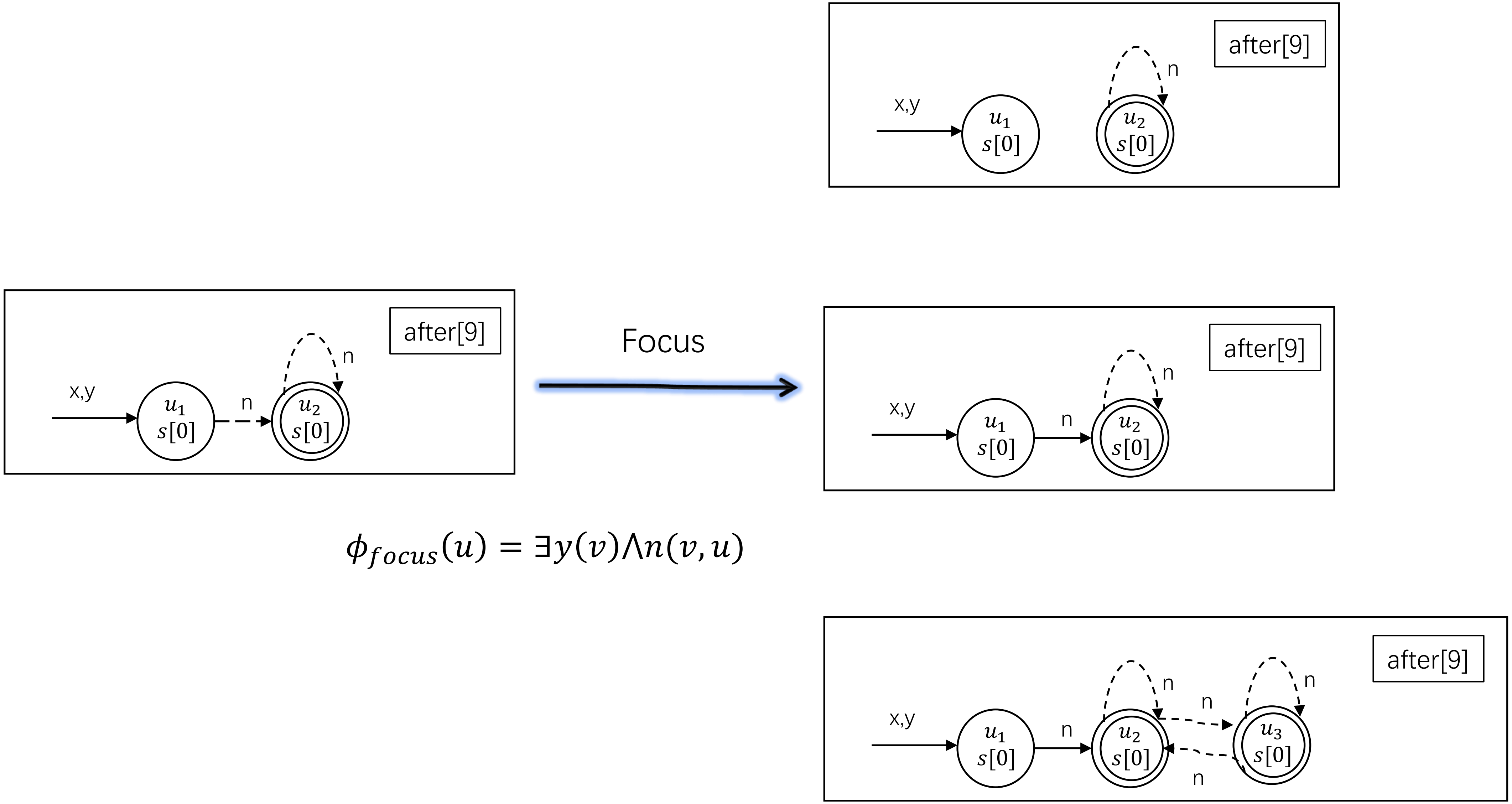
- Focus formulae



st	Focus Formulae
`x = NULL`	$\emptyset$
`x = t`	$\{t(v)\}$
`x = t->n`	$\{\exists v_1 : t(v_1) \wedge n(v_1, v)\}$
`x->n = t`	$\{x(v), t(v)\}$
`x = malloc()`	$\emptyset$
`x == NULL`	$\{x(v)\}$
`x != NULL`	$\{x(v)\}$
`x == t`	$\{x(v), t(v)\}$
`x != t`	$\{x(v), t(v)\}$
`UninterpretedCondition`	$\emptyset$

Sagiv M, Reps T, Wilhelm R. **Parametric shape analysis via 3-valued logic**[J]. TOPLAS 2002

Loop 2, iteration 1, case 2 t = y.n



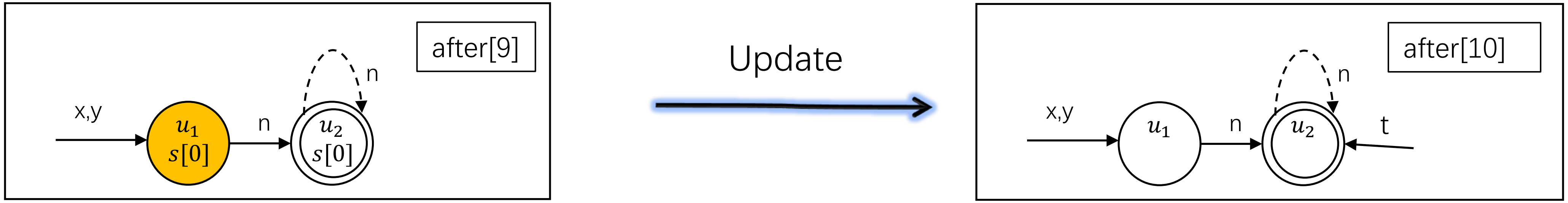
$$\phi_{focus}(u) = \exists y(v) \wedge n(v,u)$$

Loop 2, iteration 1, case 2

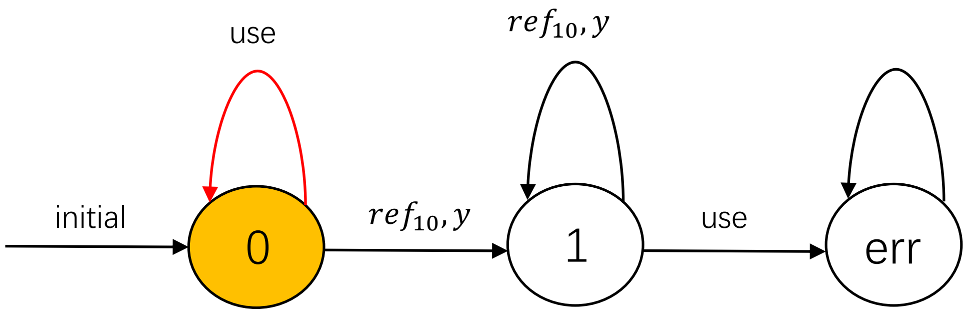
t = y.n

Use objects pointed by y, y.n

$$t'(u) = \exists y(v) \land n(v,u)$$



u_1 is pointed by y. Use action triggered

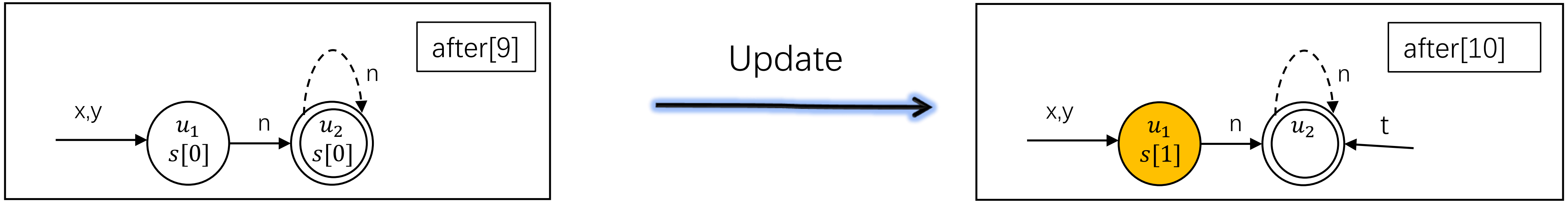


Loop 2, iteration 1, case 2

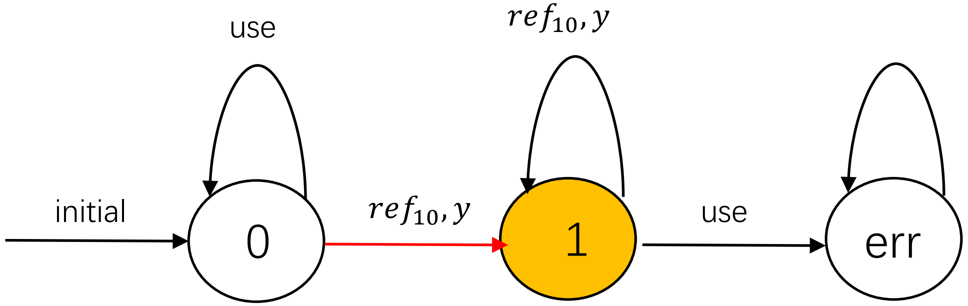
t = y.n

Use objects pointed by y, y.n

$$t'(u) = \exists y(v) \land n(v,u)$$



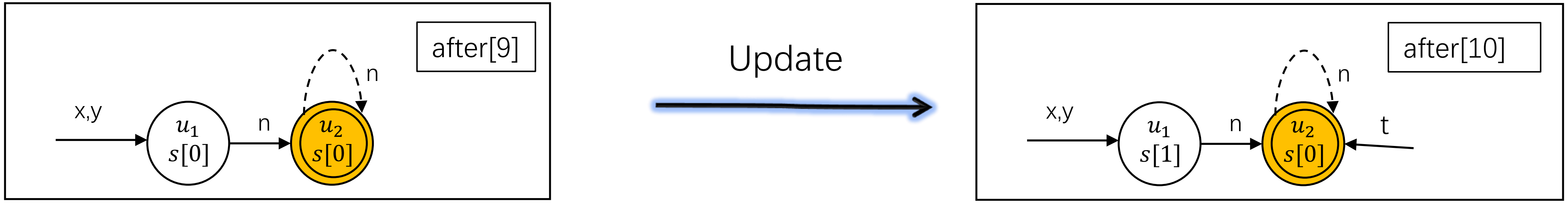
u_1 is pointed by y. Use action triggered
Use is triggered at program location 10, hence $ref_{10,y}$ is triggered.



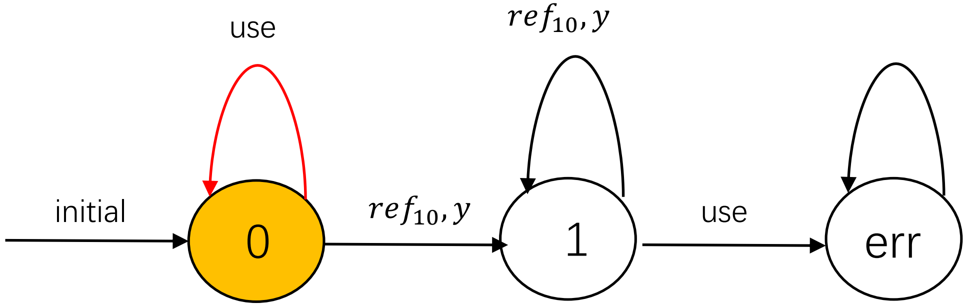
Loop 2, iteration 1, case 2 t = y.n

Use objects pointed by y, y.n

$$t'(u) = \exists y(v) \land n(v,u)$$



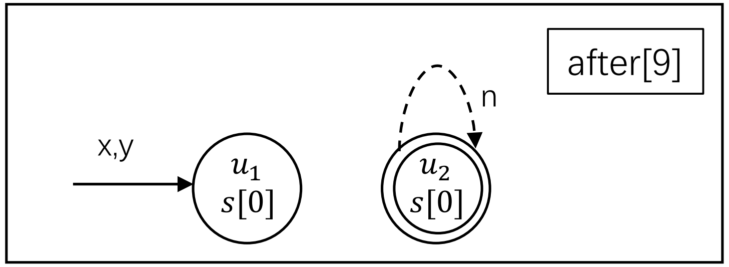
u_2 is pointed by y.n. Use action triggered. No $ref_{10,y}$ is triggered.

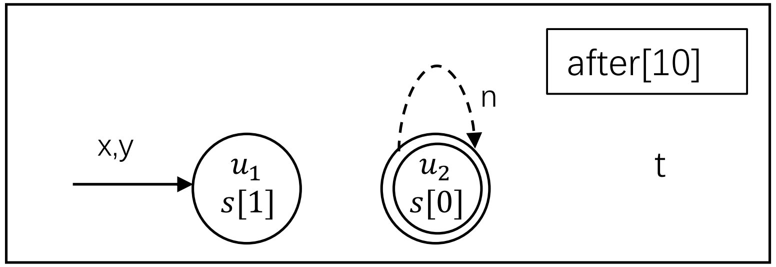


Loop 2, iteration 1, case 2

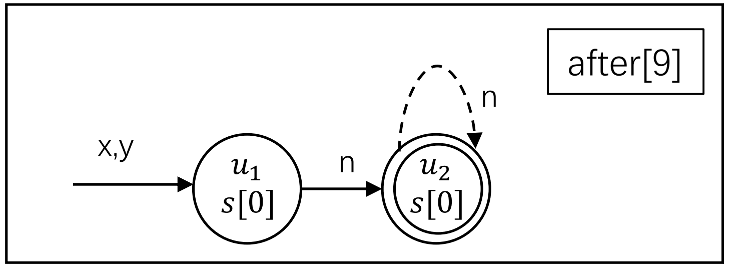
t = y.n

Use objects pointed by y, y.n

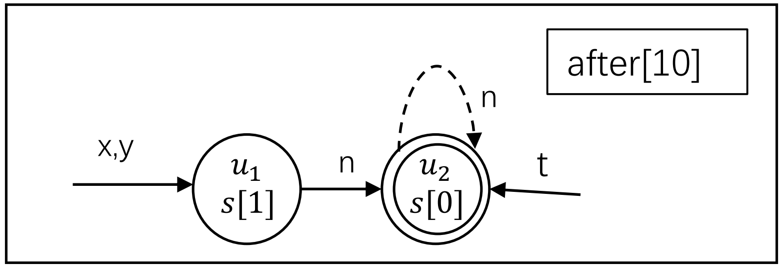


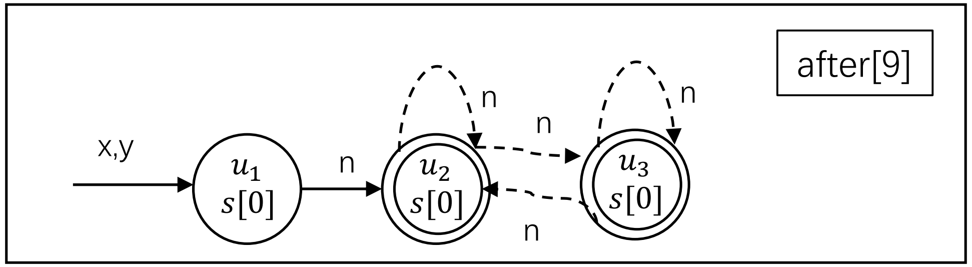


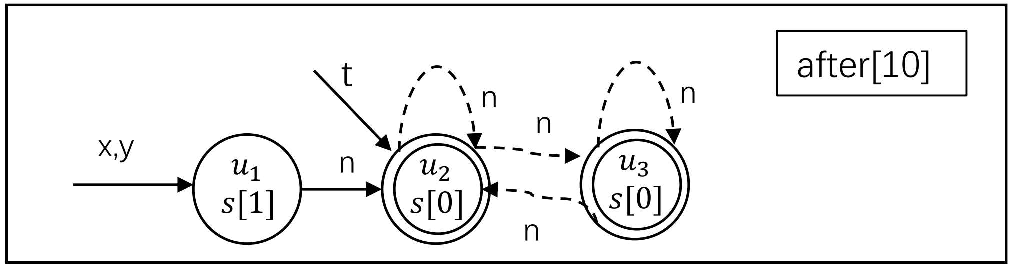
$$t'(u) = \exists y(v) \wedge n(v, u)$$



Update

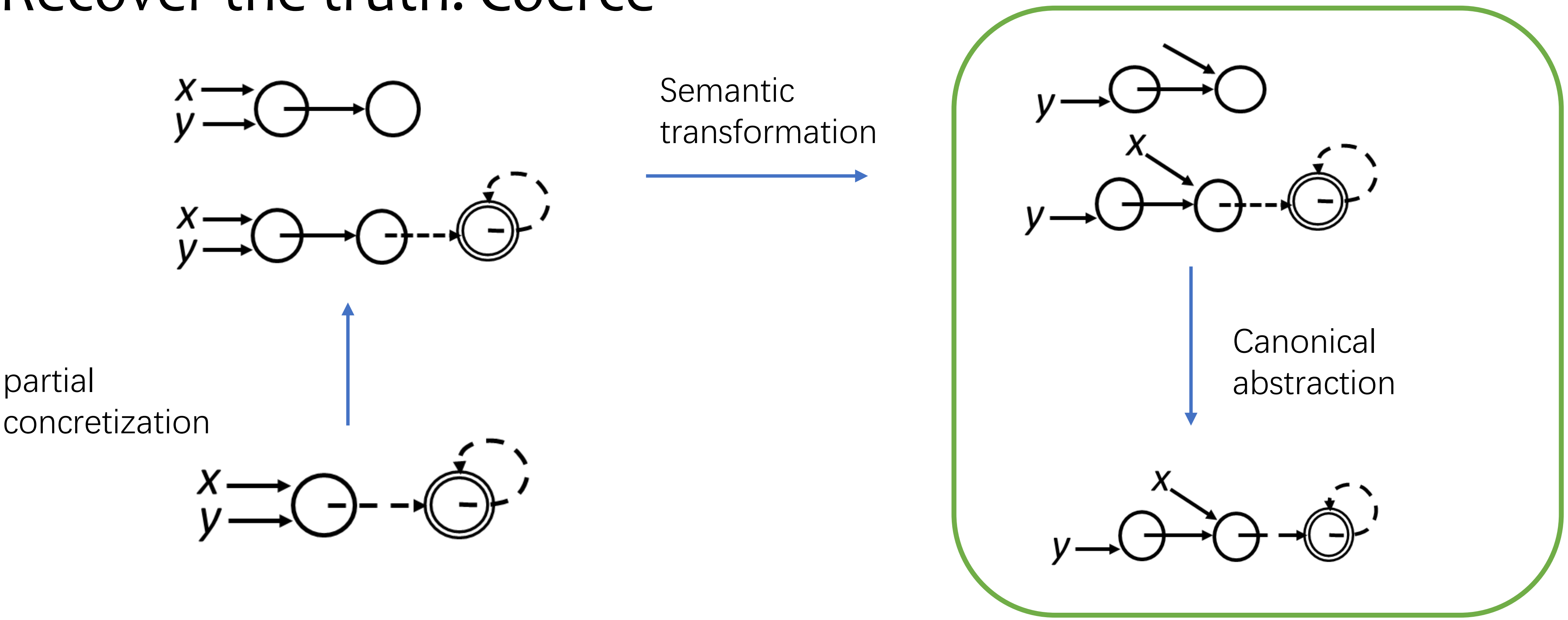






Formula Update

- Recover the truth: Coerce



Semantic transformation

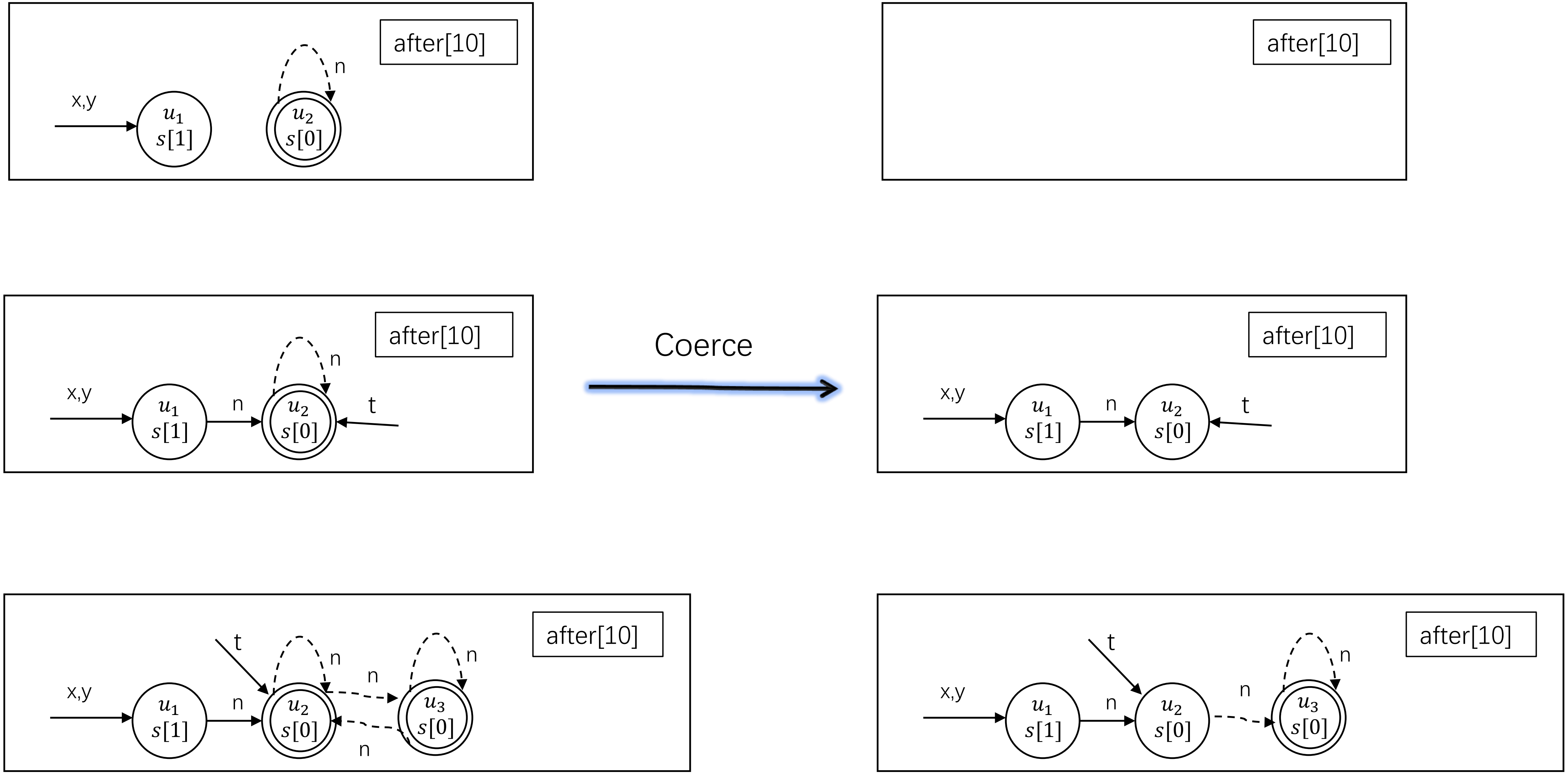
partial concretization

Canonical abstraction

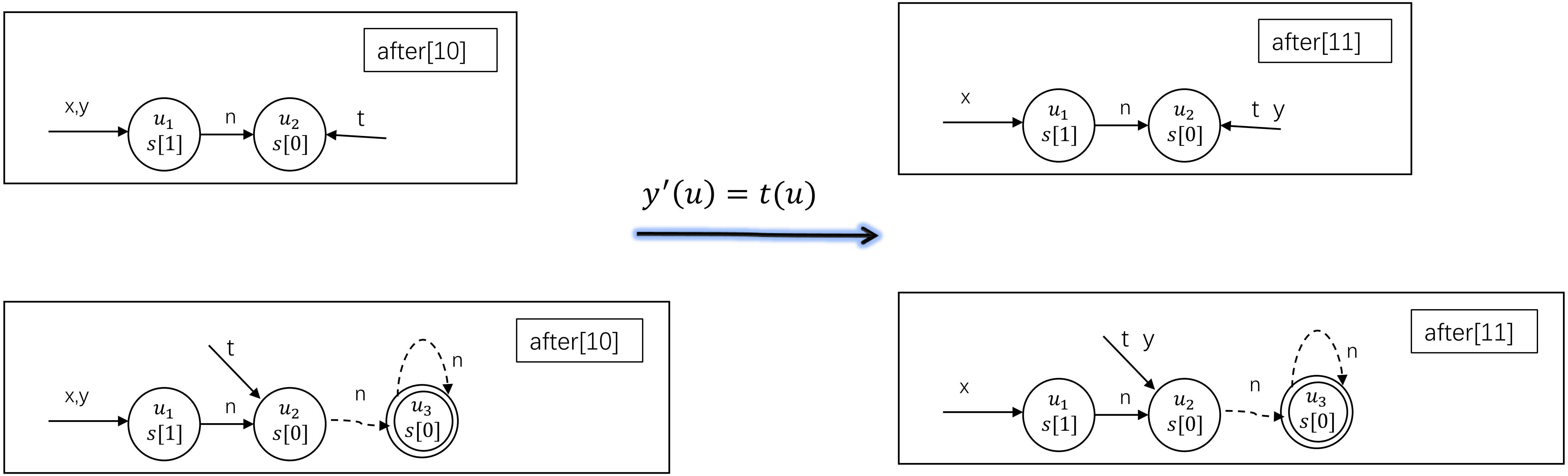
Sagiv M, Reps T, Wilhelm R. **Parametric shape analysis via 3-valued logic**[J]. TOPLAS 2002

Remove the inconsistency

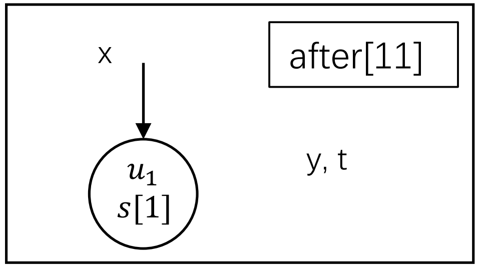
Loop 2, iteration 1, case 2 t = y.n

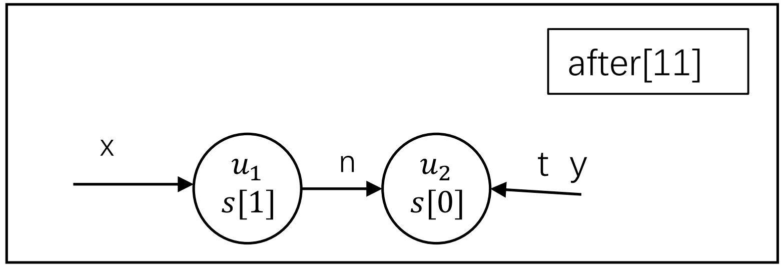


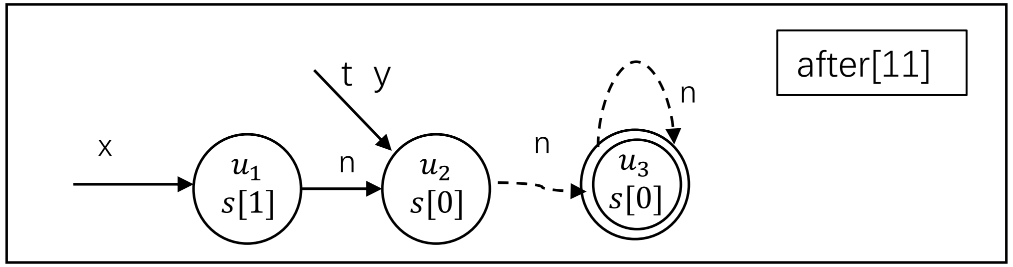
Loop 2, iteration 1, case 2　　　　y=t



Loop 2, iteration 1 finished

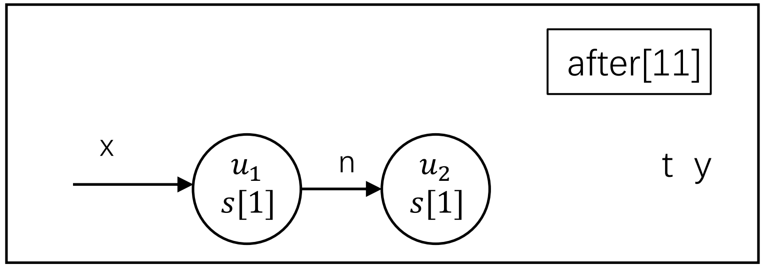


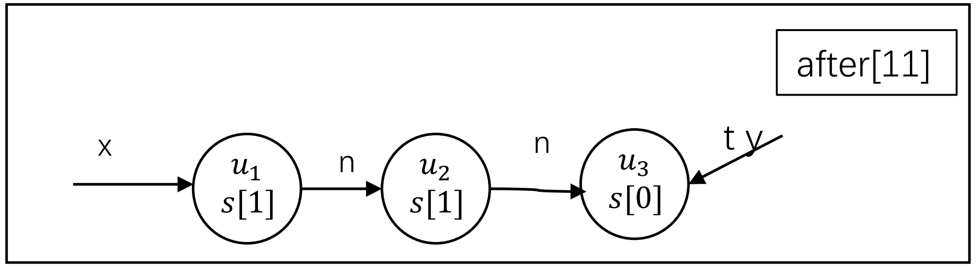


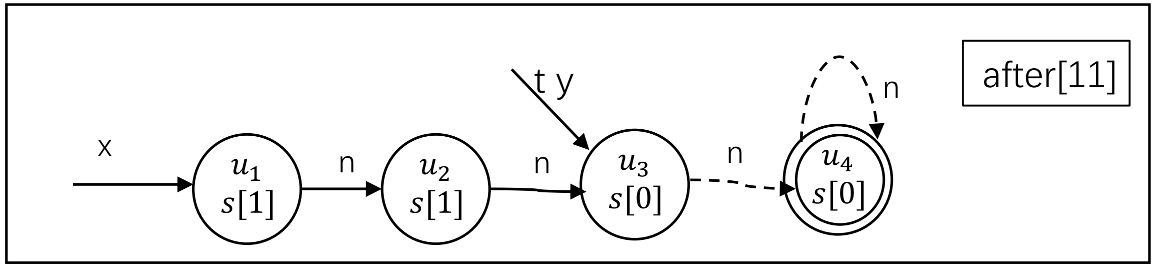


```
while (y != null) {
    t = y.n;
    y = t;
}
```

Loop 2, iteration 2 finished

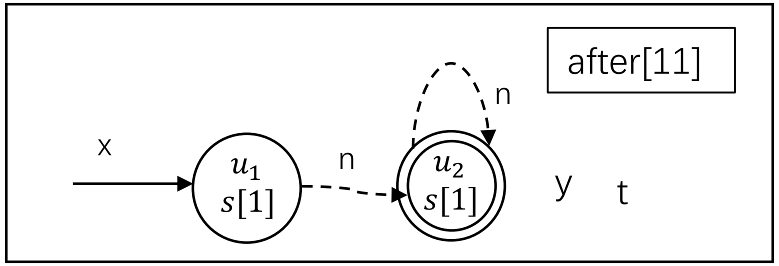


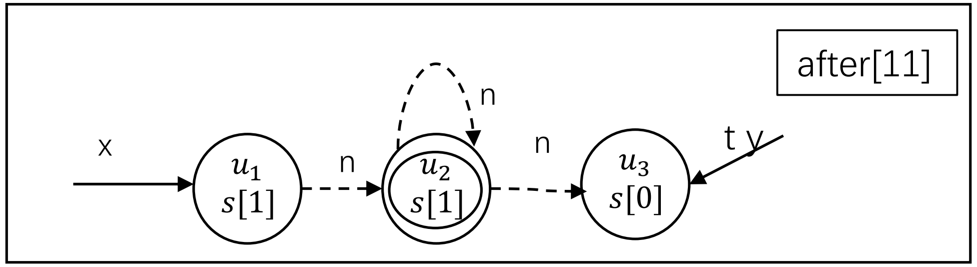


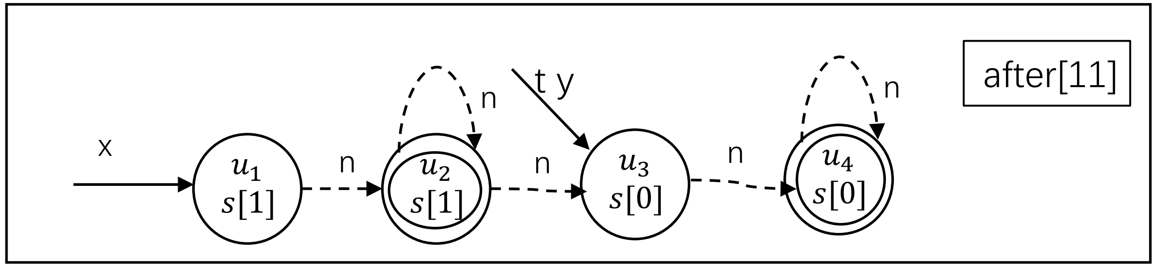


```
while (y != null) {
    t = y.n;
    y = t;
}
```

Loop 2, iteration 3 finished

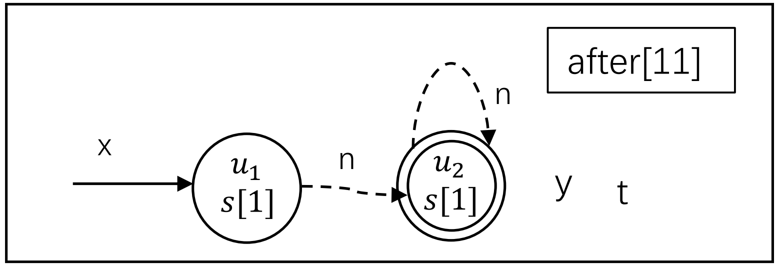




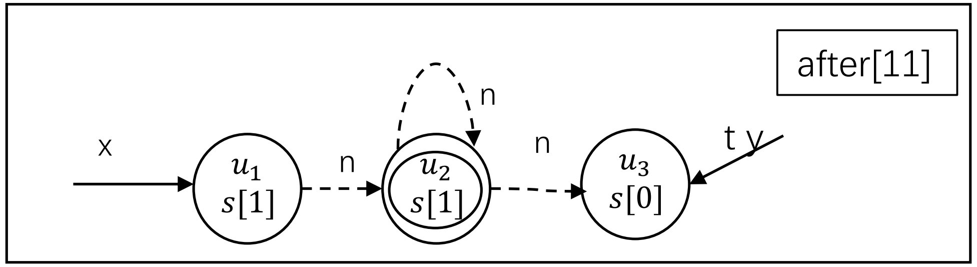


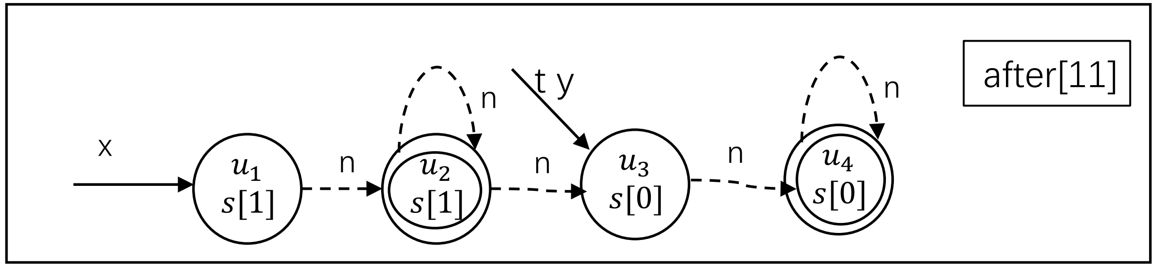
```
while (y != null) {
     t = y.n;
     y = t;
 }
```

Loop 2, iteration 4 finished,
reaching fixed point

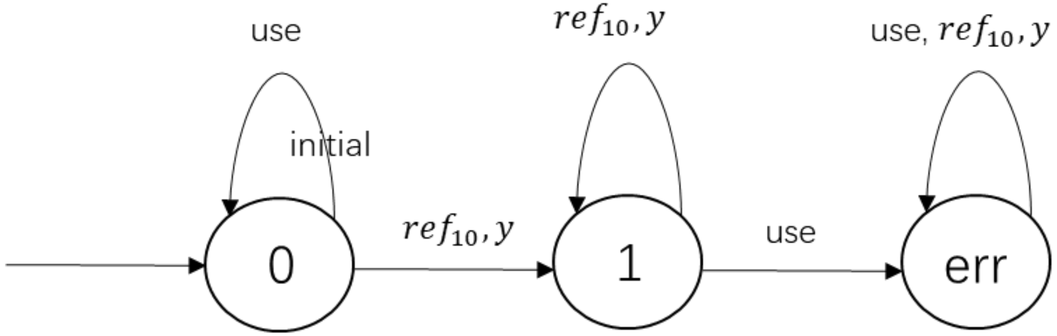


```
while (y != null) {
    t = y.n;
    y = t;
}
```



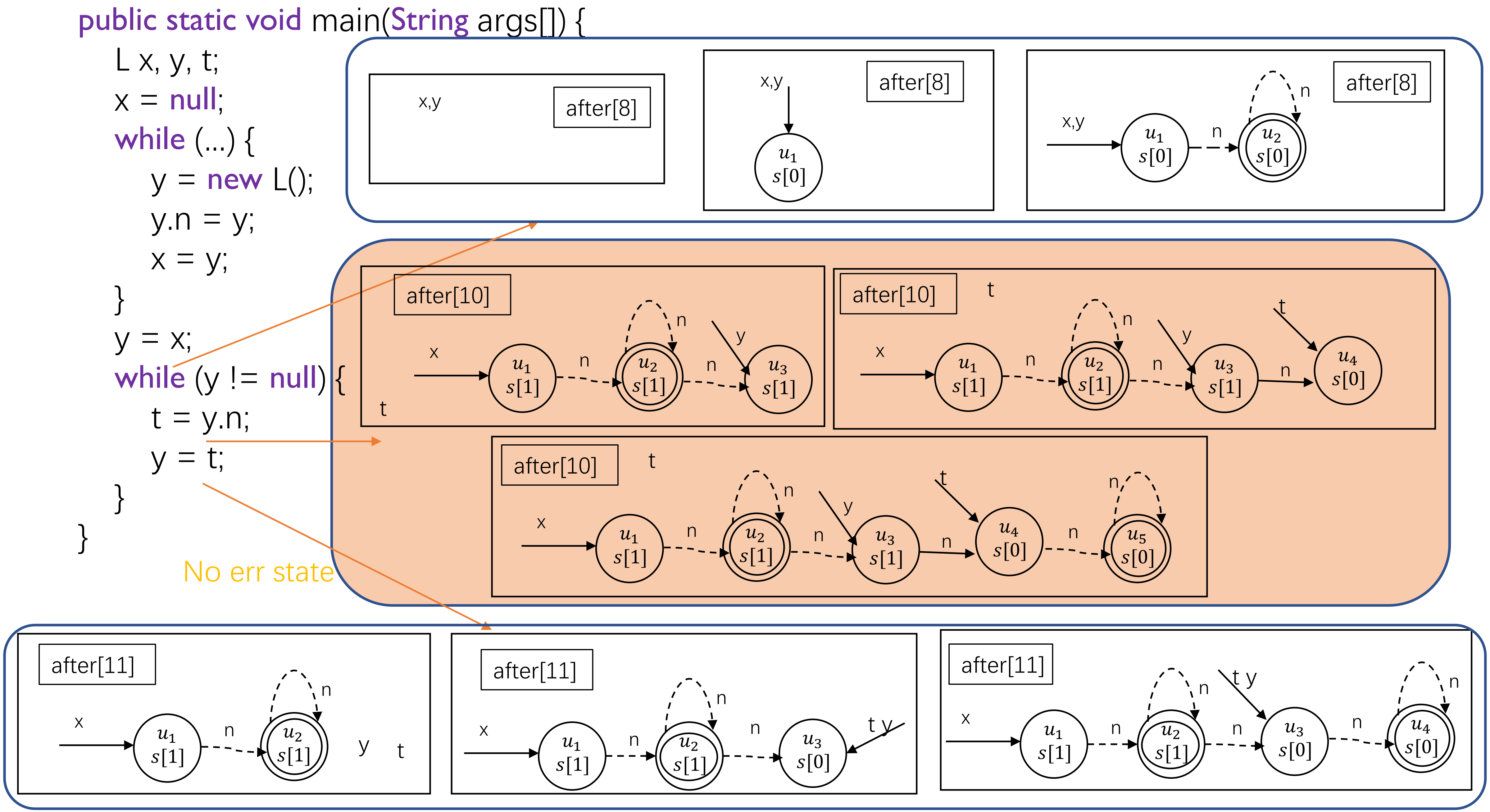


```
public static void main(String args[]) {
    L x, y, t;
    x = null;
    while (...) {
        y = new L();
        y.n = y;
        x = y;
    }
    y = x;
    while (y != null) {
        t = y.n;
        y = t;
    }
}
```



Example: the HSA of y at line 10

Accepting state: {0,1}



```
public static void main(String args[]) {
    L x, y, t;
    x = null;
    while (...) {
        y = new L();
        y.n = y;
        x = y;
    }
    y = x;
    while (y != null) {
        t = y.n;
        y = t;
    }
}
```

No err state

TVLA: Summary

- The limitations of analogue pointer analysis in stage 1
  - The expressivity is limited
  - The shape properties are different in previous works(lack of general approach)
- TVLA
  - Abstract memory configuration(shape graph) and transformer(function summary) **in logical structures**
  - Encode and abstract memory configuration(shape graph) **by predicates** canonical abstraction
  - Encode the semantics **by logic formula** predicate update formula
  - Statement guides the pointwise state transformation focus operation

Strong expressivity
General framework

TVLA: Summary

- Conclusion
  - TVLA is rigorous and elegant
    - Perform more strong updates by symbolic abstraction
    - Given abstract predicates, the logical structures are bounded and the number of them are finite. This guarantee the terminability.
  - BUT unscalable
    - First order logic constraint solving
    - It is non-trivial to choose proper instrumentation predicates and abstract predicates

Q & A