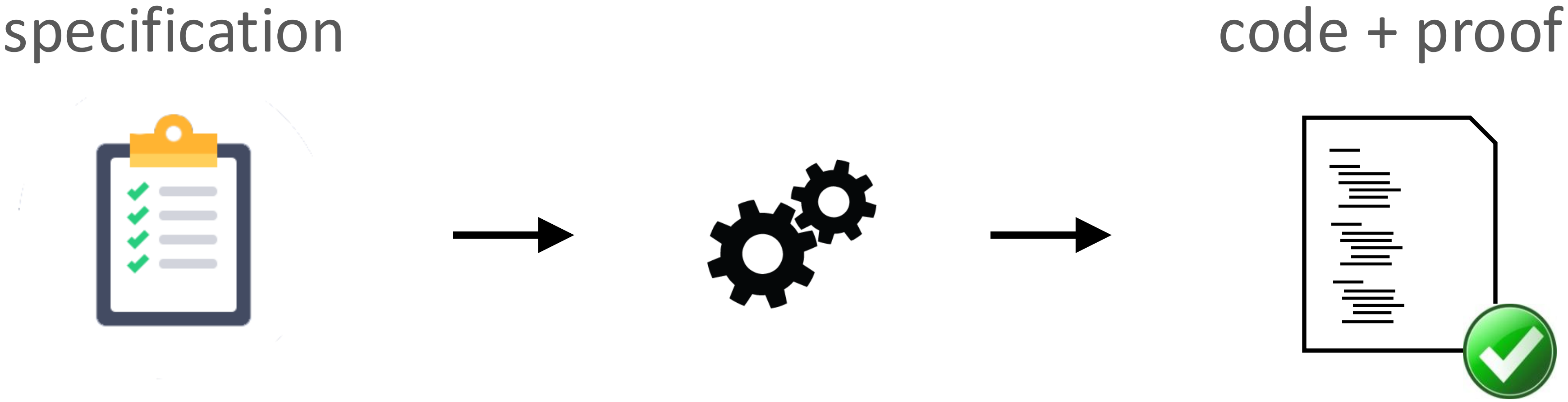
#27: Separation Logic and Deductive Synthesis

Sankha Narayan Guria

EECS 700: Introduction to Program Synthesis

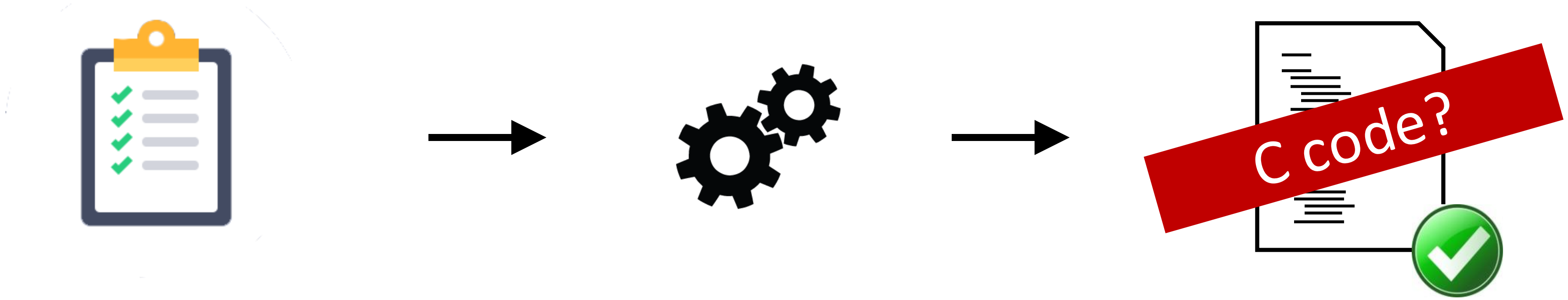
Program synthesis with guarantees



Program synthesis with guarantees

specification

code + proof



☹ verbose
☹ unstructured
☹ pointers

The trouble with pointers

- Can we naively apply Hoare logic to programs with pointers?

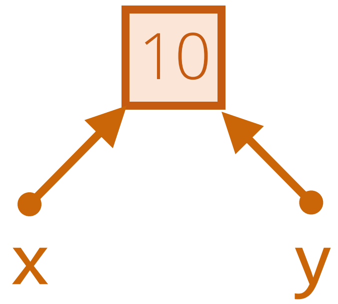
$$\{*\,x = 10 \wedge *\,y = 10\}$$
$$\Rightarrow$$
$$\{(*\,x) + 5 = 15 \wedge (*\,y) - 5 = 5\}$$

```
*x = *x + 5;
```

$$\{*\,x = 15 \wedge (*\,y) - 5 = 5\}$$

```
*y = *y - 5;
```

$$\{*\,x = 15 \wedge *\,y = 5\}$$

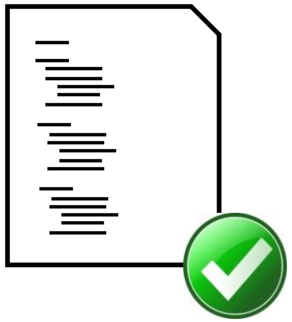


SuSLik

Synthesis Using Separation Logik

The SuSLik approach

separation logic → deductive synthesis → code



☺ reasoning about pointers & aliasing

☺ uses specs to guide synthesis

Outline

1. example: swap

   a taste of SuSLik

2. separation logic

   specifying pointer-manipulating programs

3. deductive synthesis

   from SL specifications to programs

Outline

1. example: swap
2. separation logic
3. deductive synthesis

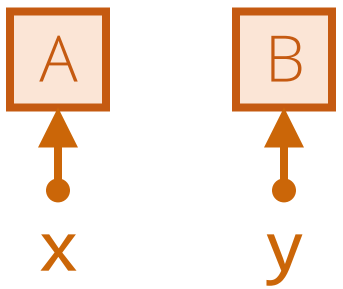
Example: swap

Swap values of two *distinct* pointers

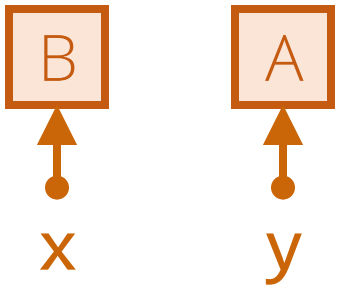
```
void swap(loc x, loc y)
```

Example: swap

start state:



end state:



in separation logic:

precondition

$\{ x \mapsto A * \boxed{y} \mapsto B \}$

separately

```
void swap(loc x, loc y)
```

$\{ x \mapsto B * \boxed{y} \mapsto A \}$

postcondition

logical variables

Demo: swap

Swap values of two *distinct* pointers

```
void swap(loc x, loc y)
```

$$\{ x \boxed{\mapsto A * y} \mapsto B \}$$

??

$$\{ x \mapsto B * y \mapsto A \}$$

```
let a1 = *x;
```

$\{ x \mapsto a1 * y \mapsto B\ \}$

??

$\{ x \mapsto B * y \mapsto a1 \}$

```
let a1 = *x;
let b1 = *y;
```

$\{\ x \mapsto a1 * y \mapsto b1\ \}$

??

$\{\ x \boxed{\mapsto b1 * y} \mapsto a1\ \}$

```
let a1 = *x;
let b1 = *y;
*x = b1;
```

$\{ x \mapsto b1 * y \mapsto b1 \}$

??

$\{ x \mapsto b1 * y \mapsto a1 \}$

```
let a1 = *x;
let b1 = *y;
*x = b1;
*y = a1;
```

$\{\, x \mapsto b1 * y \mapsto a1 \,\}$

??

$\{\, x \mapsto b1 * y \mapsto a1 \,\}$

same

```
let a1 = *x;
let b1 = *y;
*x = b1;
*y = a1;
```

```
void swap(loc x, loc y) {
        let a1 = *x;
        let b1 = *y;
        *x = b1;
        *y = a1;
}
```

Outline

1. example: swap
2. separation logic
3. deductive synthesis

Separation logic (SL)

Hoare logic
“about the heap”

Separation logic (SL)

SL assertions

$$\{P\}\ c\ \{Q\}$$

program

starting in a state that satisfies P
program c will execute without memory errors,
and upon its termination the state will satisfy Q

Outline

1. example: swap
2. separation logic
   2.1. programs
   2.2. assertions
   2.3. specifying data transformations
3. deductive synthesis

Separation logic (SL)

$$\{P\} \;\; c \;\; \{Q\}$$

program

Programs

do nothing	**skip**

Programs

offset (natural number)

do nothing	**skip**
read from heap	**let** y = *(x + n)

variables

Programs

do nothing	**skip**	
read from heap	**let** y = *(x + n)	
write to heap	*(x + n) = e	← expression (arithmetic, boolean)

Programs

do nothing	**skip**
read from heap	**let** y = *(x + n)
write to heap	*(x + n) = e
allocate block	**let** y = **malloc**(n)

Programs

do nothing	**skip**
read from heap	**let** y = *(x + n)
write to heap	*(x + n) = e
allocate block	**let** y = **malloc**(n)
free block	**free**(x)

Programs

do nothing	**skip**
read from heap	**let** y = *(x + n)
write to heap	*(x + n) = e
allocate block	**let** y = **malloc**(n)
free block	**free**(x)
procedure call	$p(e_1, \ldots, e_n)$

Programs

do nothing	**skip**
read from heap	**let** y = *(x + n)
write to heap	*(x + n) = e
allocate block	**let** y = **malloc**(n)
free block	**free**(x)
procedure call	$p(e_1, \ldots, e_n)$
~~assignment~~	only heap is mutable, not stack variables!

Programs

do nothing	**skip**
read from heap	**let** y = *(x + n)
write to heap	*(x + n) = e
allocate block	**let** y = **malloc**(n)
free block	**free**(x)
procedure call	$p(e_1, \ldots, e_n)$
sequential composition	$c_1 ; c_2$
conditional	**if** (e) $\{c_1\}$ else $\{c_2\}$

Outline

1. example: swap
2. separation logic
   2.1. programs
   2.2. assertions
   2.3. specifying data transformations
3. deductive synthesis

Separation logic (SL)

SL assertions

$$\{P\}\ \ c\ \ \{Q\}$$

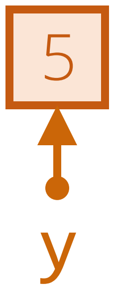
SL assertions

empty heap { emp }

SL assertions

empty heap $\{ \mathrm{emp} \}$

singleton heap $\{ y \mapsto 5 \}$



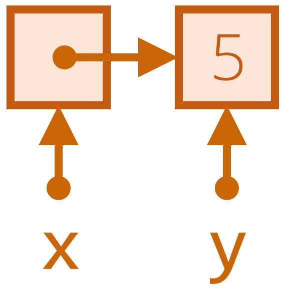
SL assertions

empty heap $\{ \text{emp} \}$

singleton heap $\{ y \mapsto 5 \}$

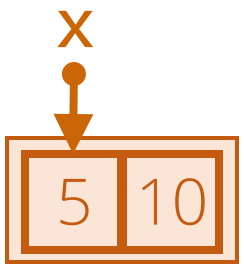
separating conjunction $\{ x \mapsto y * y \mapsto 5 \}$

heaplets



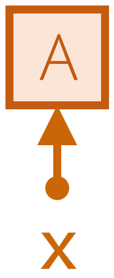
SL assertions

empty heap	$\{ \mathrm{emp} \}$
singleton heap	$\{ y \mapsto 5 \}$
separating conjunction	$\{ x \mapsto y * y \mapsto 5 \}$
memory block	$\{ [x, 2] * x \mapsto 5 * (x + 1) \mapsto 10 \}$

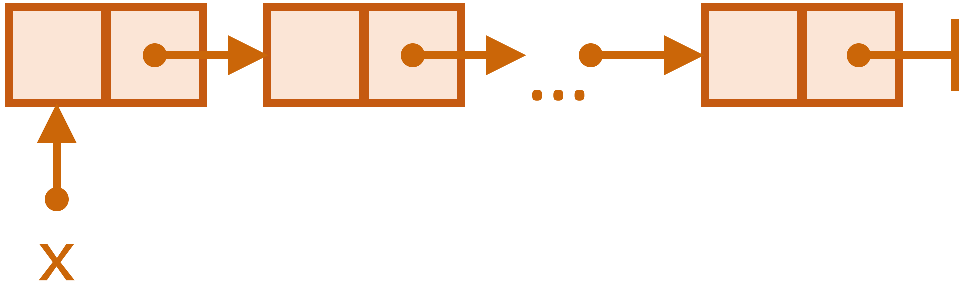


SL assertions

empty heap	$\{ \text{emp} \}$
singleton heap	$\{ y \mapsto 5 \}$
separating conjunction	$\{ x \mapsto y * y \mapsto 5 \}$
memory block	$\{ [x, 2] * x \mapsto 5 * (x + 1) \mapsto 10 \}$
+ pure formula	$\{ A > 5 \,;\, x \mapsto A \}$

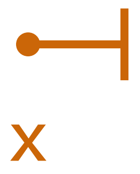


SL assertions: linked structures



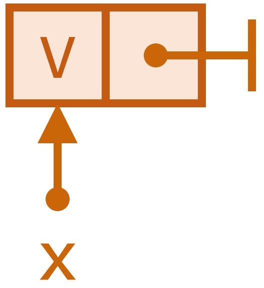
SL assertions: linked structures

linked list $\{ x = 0 \ ; \ emp \}$



SL assertions: linked structures

linked list $\{ [x, 2] * x \mapsto V * (x + 1) \mapsto 0 \}$

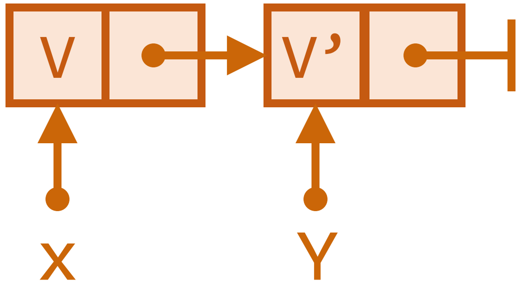


SL assertions: linked structures

linked list $\{ [x, 2] * x \mapsto V * (x + 1) \mapsto Y *$

$[Y, 2] * Y \mapsto V' * (Y + 1) \mapsto 0$

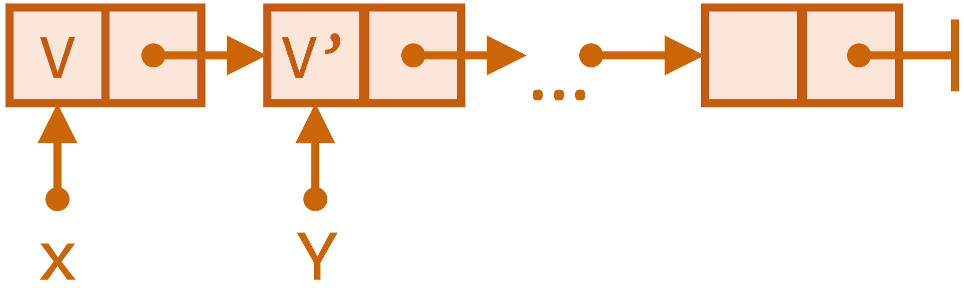
$\}$



SL assertions: linked structures

linked list

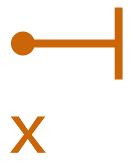
$$\{\ [x, 2] * x \mapsto V * (x + 1) \mapsto Y *$$
$$[Y, 2] * Y \mapsto V' * (Y + 1) \mapsto Y' *$$
$$\ldots$$
$$\}$$

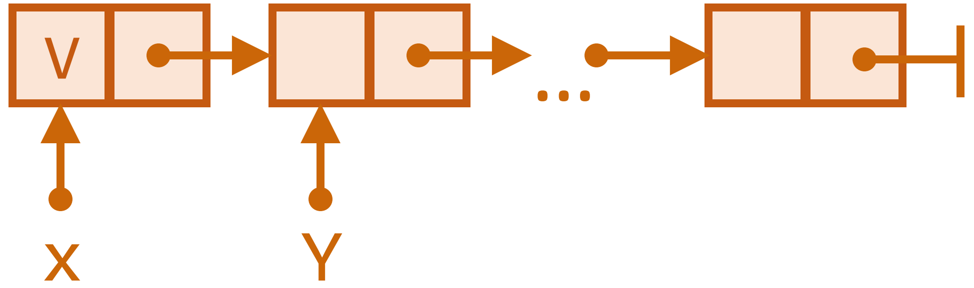


inductive predicates to the rescue!

The linked list predicate

```
predicate list (loc x) {
  | x = 0 => { emp }
  | x ≠ 0 => { [x, 2]
        ∗ x ↦ V
        ∗ (x + 1) ↦ Y
        ∗ list(Y)
      }
}
```





Outline

1. example: swap

2. separation logic
   2.1. programs
   2.2. assertions
   2.3. specifying data transformations

3. deductive synthesis

Example: dispose a list

```
void dispose(loc x)
```

{ list(x) }

{ emp }

Example: copy a list

```
void copy(loc x, loc ret)
```

{ list(x, S) * ret ↦ _ }

{ list(x, S) * ret ↦ Y * list(Y, S) }

return location

Outline

1. example: swap
2. the logic
3. deductive synthesis

Deductive synthesis

synthesis as proof search

Outline

1. example: swap
2. the logic
3. deductive synthesis
    3.1. proof system
    3.2. proof search

transforming entailment

$$P \rightsquigarrow Q \mid c$$

a state that satisfies P
can be transformed into a state that satisfies Q
using a program c

Synthetic separation logic (SSL)

proof system for
transforming entailment

$\{\mathrm{emp}\} \rightsquigarrow \{\mathrm{emp}\} \mid ??$

(Emp)

$$\{\mathsf{emp}\} \rightsquigarrow \{\mathsf{emp}\} \mid \mathbf{skip}$$

(Frame)

$$\frac{\{\, P \,\} \rightsquigarrow \{\, Q \,\} \mid c}{\{\, P * R \,\} \rightsquigarrow \{\, Q * R \,\} \mid c\,??}$$

(Write)

$$\frac{\{ x \mapsto e * P \} \rightsquigarrow \{ x \mapsto e * Q \} \mid c}{\{ x \mapsto \_ * P \} \rightsquigarrow \{ x \mapsto e * Q \} \mid {*x} \;??}$$

(Read)

$$\frac{[y/A]\{\, x \mapsto A * P \,\} \rightsquigarrow [y/A]\{\, Q \,\} \;\mid\; c}{\{\, x \mapsto A * P \,\} \rightsquigarrow \{\, Q \,\} \qquad \mid\;]\,?\,?}$$

SSL: basic rules

(Emp)

$$\{\mathrm{emp}\} \rightsquigarrow \{\mathrm{emp}\} \mid \texttt{skip}$$

(Read)

$$\frac{[y/A]\{\, x \mapsto A * P \,\} \rightsquigarrow [y/A]\{\, Q \,\} \mid \texttt{c}}{\{\, x \mapsto A * P \,\} \rightsquigarrow \{\, Q \,\} \mid \texttt{let y = *x; c}}$$

(Frame)

$$\frac{\{\, P \,\} \rightsquigarrow \{\, Q \,\} \mid \texttt{c}}{\{\, P * R \,\} \rightsquigarrow \{\, Q * R \,\} \mid \texttt{c}}$$

(Write)

$$\frac{\{\, x \mapsto e * P \,\} \rightsquigarrow \{\, x \mapsto e * Q \,\} \mid \texttt{c}}{\{\, x \mapsto \_ * P \,\} \rightsquigarrow \{\, x \mapsto e * Q \,\} \mid \texttt{*x = e; c}}$$

Example: swap

$$\{\, x \mapsto A * y \mapsto B \,\} \rightsquigarrow \{\, x \mapsto B * y \mapsto A \,\} \mid \quad ??$$

$$\{ x \mapsto A * y \mapsto B \} \rightsquigarrow \{ x \mapsto B * y \mapsto A \} \mid \;\; ??$$

$$\frac{\{\, x \mapsto a1 * y \mapsto B \,\} \rightsquigarrow \{\, x \mapsto B * y \mapsto a1 \,\} \mid \;\; \texttt{??}}{\{\, x \mapsto A * y \mapsto B \,\} \rightsquigarrow \{\, x \mapsto B * y \mapsto A \,\} \mid \;\; \texttt{let a1 = *x; ??}}\;(\text{Read})$$

$$\dfrac{\dfrac{\{\, x \mapsto a1 * y \mapsto b1 \,\} \leadsto \{\, x \mapsto b1 * y \mapsto a1 \,\} \mid \ \texttt{??}}{\{\, x \mapsto a1 * y \mapsto B \,\} \leadsto \{\, x \mapsto B * y \mapsto a1 \,\} \mid \ \texttt{\textbf{let} b1 = *y; ??}}\ (\text{Read})}{\{\, x \mapsto A * y \mapsto B \,\} \leadsto \{\, x \mapsto B * y \mapsto A \,\} \mid \ \texttt{\textbf{let} a1 = *x; ??}}\ (\text{Read})$$

$$\dfrac{\dfrac{\dfrac{\{\, x \mapsto b1 * y \mapsto b1 \,\} \rightsquigarrow \{\, x \mapsto b1 * y \mapsto a1 \,\} \mid \texttt{??}}{\{\, x \mapsto a1 * y \mapsto b1 \,\} \rightsquigarrow \{\, x \mapsto b1 * y \mapsto a1 \,\} \mid \texttt{*x = b1; ??}}\;(\text{Write})}{\{\, x \mapsto a1 * y \mapsto B \,\} \rightsquigarrow \{\, x \mapsto B * y \mapsto a1 \,\} \mid \texttt{let b1 = *y; ??}}\;(\text{Read})}{\{\, x \mapsto A * y \mapsto B \,\} \rightsquigarrow \{\, x \mapsto B * y \mapsto A \,\} \mid \texttt{let a1 = *x; ??}}\;(\text{Read})$$

$$\{ y \mapsto b1 \} \rightsquigarrow \{ y \mapsto a1 \} \mid \texttt{??}$$

(Frame)

$$\{ x \mapsto b1 * y \mapsto b1 \} \rightsquigarrow \{ x \mapsto b1 * y \mapsto a1 \} \mid \texttt{??}$$

(Write)

$$\{ x \mapsto a1 * y \mapsto b1 \} \rightsquigarrow \{ x \mapsto b1 * y \mapsto a1 \} \mid \texttt{*x = b1; ??}$$

(Read)

$$\{ x \mapsto a1 * y \mapsto B \} \rightsquigarrow \{ x \mapsto B * y \mapsto a1 \} \mid \texttt{\textbf{let} b1 = *y; ??}$$

(Read)

$$\{ x \mapsto A * y \mapsto B \} \rightsquigarrow \{ x \mapsto B * y \mapsto A \} \mid \texttt{\textbf{let} a1 = *x; ??}$$

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\{\, y \mapsto a1 \,\} \rightsquigarrow \{\, y \mapsto a1 \,\} \mid \texttt{??}}{\{\, y \mapsto b1 \,\} \rightsquigarrow \{\, y \mapsto a1 \,\} \mid \texttt{*y = a1; ??}}\ (\text{Write})}{\{\, x \mapsto b1 * y \mapsto b1 \,\} \rightsquigarrow \{\, x \mapsto b1 * y \mapsto a1 \,\} \mid \texttt{??}}\ (\text{Frame})}{\{\, x \mapsto a1 * y \mapsto b1 \,\} \rightsquigarrow \{\, x \mapsto b1 * y \mapsto a1 \,\} \mid \texttt{*x = b1; ??}}\ (\text{Write})}{\{\, x \mapsto a1 * y \mapsto B \,\} \rightsquigarrow \{\, x \mapsto B * y \mapsto a1 \,\} \mid \textbf{let}\ \texttt{b1 = *y; ??}}\ (\text{Read})}{\{\, x \mapsto A * y \mapsto B \,\} \rightsquigarrow \{\, x \mapsto B * y \mapsto A \,\} \mid \textbf{let}\ \texttt{a1 = *x; ??}}\ (\text{Read})
$$

$$\{\ \mathrm{emp}\ \} \rightsquigarrow \{\ \mathrm{emp}\ \} \mid \texttt{??}$$

(Frame)

$$\{\ y \mapsto a1\ \} \rightsquigarrow \{\ y \mapsto a1\ \} \mid \texttt{??}$$

(Write)

$$\{\ y \mapsto b1\ \} \rightsquigarrow \{\ y \mapsto a1\ \} \mid \texttt{*y = a1; ??}$$

(Frame)

$$\{\ x \mapsto b1 * y \mapsto b1\ \} \rightsquigarrow \{\ x \mapsto b1 * y \mapsto a1\ \} \mid \texttt{??}$$

(Write)

$$\{\ x \mapsto a1 * y \mapsto b1\ \} \rightsquigarrow \{\ x \mapsto b1 * y \mapsto a1\ \} \mid \texttt{*x = b1; ??}$$

(Read)

$$\{\ x \mapsto a1 * y \mapsto B\ \} \rightsquigarrow \{\ x \mapsto B * y \mapsto a1\ \} \mid \texttt{\textbf{let} b1 = *y; ??}$$

(Read)

$$\{\ x \mapsto A * y \mapsto B\ \} \rightsquigarrow \{\ x \mapsto B * y \mapsto A\ \} \mid \texttt{\textbf{let} a1 = *x; ??}$$

$$\overline{\phantom{\{\, \mathsf{emp} \,\} \rightsquigarrow \{\, \mathsf{emp} \,\} \mid \texttt{skip}}}\ \text{(Emp)}$$

$$\{\, \mathsf{emp} \,\} \rightsquigarrow \{\, \mathsf{emp} \,\} \mid \boxed{\texttt{skip}}$$

$$\overline{\phantom{\{\, y \mapsto a1 \,\} \rightsquigarrow \{\, y \mapsto a1 \,\} \mid ??}}\ \text{(Frame)}$$

$$\{\, y \mapsto a1 \,\} \rightsquigarrow \{\, y \mapsto a1 \,\} \mid \texttt{??}$$

$$\overline{\phantom{\{\, y \mapsto b1 \,\} \rightsquigarrow \{\, y \mapsto a1 \,\} \mid *y = a1; ??}}\ \text{(Write)}$$

$$\{\, y \mapsto b1 \,\} \rightsquigarrow \{\, y \mapsto a1 \,\} \mid \boxed{\texttt{*y = a1;}}\ \texttt{??}$$

$$\overline{\phantom{\{\, x \mapsto b1 * y \mapsto b1 \,\} \rightsquigarrow \{\, x \mapsto b1 * y \mapsto a1 \,\} \mid ??}}\ \text{(Frame)}$$

$$\{\, x \mapsto b1 * y \mapsto b1 \,\} \rightsquigarrow \{\, x \mapsto b1 * y \mapsto a1 \,\} \mid \texttt{??}$$

$$\overline{\phantom{\{\, x \mapsto a1 * y \mapsto b1 \,\} \rightsquigarrow \{\, x \mapsto b1 * y \mapsto a1 \,\} \mid *x = b1; ??}}\ \text{(Write)}$$

$$\{\, x \mapsto a1 * y \mapsto b1 \,\} \rightsquigarrow \{\, x \mapsto b1 * y \mapsto a1 \,\} \mid \boxed{\texttt{*x = b1;}}\ \texttt{??}$$

$$\overline{\phantom{\{\, x \mapsto a1 * y \mapsto B \,\} \rightsquigarrow \{\, x \mapsto B * y \mapsto a1 \,\} \mid let b1 = *y; ??}}\ \text{(Read)}$$

$$\{\, x \mapsto a1 * y \mapsto B \,\} \rightsquigarrow \{\, x \mapsto B * y \mapsto a1 \,\} \mid \boxed{\texttt{let b1 = *y;}}\ \texttt{??}$$

$$\overline{\phantom{\{\, x \mapsto A * y \mapsto B \,\} \rightsquigarrow \{\, x \mapsto B * y \mapsto A \,\} \mid let a1 = *x; ??}}\ \text{(Read)}$$

$$\{\, x \mapsto A * y \mapsto B \,\} \rightsquigarrow \{\, x \mapsto B * y \mapsto A \,\} \mid \boxed{\texttt{let a1 = *x;}}\ \texttt{??}$$

$\{ x \mapsto A * y \mapsto B \}$

```
let a1 = *x;  let b1 = *y;  *x = b1;  *y = a1;  skip
```

$\{ x \mapsto B * y \mapsto A \}$