Outline

⇒ 1. Transition Systems

2. Partial Correctness

3. Total Correctness

BinarySearch: Motivation

```
bool BinarySearch(int[] a, int e) {
  return BSearch(a, 0, |a| − 1, e);
}

bool BSearch(int[] a, int ℓ, int u, int e) {
  int m;
  if (ℓ > u) return false;
  else {
    m := (ℓ + u) div 2;
    if (a[m] = e) return true;
    else if (a[m] < e) return BSearch(a, m + 1, u, e);
    else return BSearch(a, ℓ, m − 1, e);
  }
}
```

Does BinarySearch return true iff a contains e?

BubbleSort: Motivation

```
int[] BubbleSort(int[] a) {
  int i, j, t;
  for (i := |a| − 1; i > 0; i := i − 1) {
    for (j := 0; j < i; j := j + 1) {
      if (a[j] > a[j + 1]) {
        t := a[j];
        a[j] := a[j + 1];
        a[j + 1] := t;
      }
    }
  }
  return a;
}
```

Does BubbleSort return a sorted array?

Motivation

Goal: Prove programs correct.

Strategy: Analyze programs as mathematical objects.

- Translate **program** and **specification** to set of **transition systems**.
- Analyze each transition system.

Specification: Statement of what program should do.

Transition System

$T : \langle \mathcal{V}, \Theta, \Omega, \mathcal{T} \rangle$

- variables $\mathcal{V} = \mathcal{F} \cup \mathcal{F}_0 \cup \mathcal{L} \cup \{\mathsf{rv}\}$
  - local variables $\mathcal{L}$
  - formal parameters $\mathcal{F}$
  - return value rv
  - variables to hold initial values of formals $\mathcal{F}_0$
- precondition $\Theta : \langle \ell_0, \theta \rangle$
  - initial location ℓ_0
  - assertion θ over $\mathcal{F}$
- postcondition $\Omega : \langle \ell_f, \omega \rangle$
  - final location ℓ_f
  - assertion ω over $\mathcal{F}_0 \cup \{\mathsf{rv}\}$

Transition System

$T : \langle \mathcal{V}, \Theta, \Omega, \mathcal{T} \rangle$

- variables $\mathcal{V}$
- precondition $\Theta : \langle \ell_0, \theta \rangle$
- postcondition $\Omega : \langle \ell_f, \omega \rangle$
- transitions $\mathcal{T}$
  transition $\tau : \langle \ell, \ell', \rho_\tau \rangle \in \mathcal{T}$
  - location ℓ
  - postlocation ℓ'
  - transition relation ρ_τ, an assertion over $\mathcal{V} \cup \mathcal{V}'$
    $\mathcal{V}'$ hold the values of $\mathcal{V}$ in the next state

BinarySearch: Transition System

```
@pre |a| ≥ 0 ∧ sorted(a, 0, |a| − 1)
@post rv ↔ (∃i ∈ [0, |a| − 1]) a[i] = e
bool BinarySearch(int[] a, int e) {
  return BSearch(a, 0, |a| − 1, e);
}

@pre 0 ≤ ℓ ∧ u < |a| ∧ sorted(a, 0, |a| − 1)
@post rv ↔ (∃i ∈ [ℓ, u]) a[i] = e
bool BSearch(int[] a, int ℓ, int u, int e) {
  int m;
  if (ℓ > u) return false;
  else {
    m := (ℓ + u) div 2;
    if (a[m] = e) return true;
    else if (a[m] < e) return BSearch(a, m + 1, u, e);
    else return BSearch(a, ℓ, m − 1, e);
  }
}
```

BinarySearch: Transition System

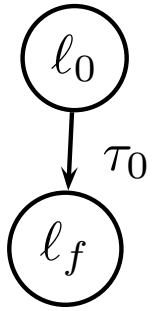
$\mathsf{@pre}\ |a| \geq 0\ \wedge\ \mathsf{sorted}(a, 0, |a| - 1)$

$\mathsf{@post}\ \mathsf{rv}\ \leftrightarrow\ (\exists i \in [0, |a| - 1])\ a[i] = e$

`bool` BinarySearch(`int[]` a, `int` e) {

  `return` BSearch$(a, 0, |a| - 1, e)$;

}



ρ_{τ_0} :

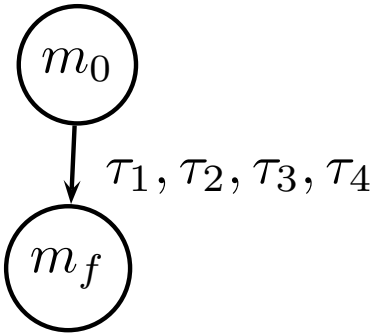
$$a'_0 = a\ \wedge\ e'_0 = e\ \wedge\ |a|'_0 = |a|$$
$$\wedge\ \mathsf{rv}' = \text{BSearch}(a, 0, |a| - 1, e)\ \wedge\ \mathsf{pres}(\ldots)$$

$\mathsf{pres}(V)$: preserve values of $V \subseteq \mathcal{V}$

$\mathsf{pres}(\ldots)$: preserve values of unmentioned variables of $\mathcal{V}$

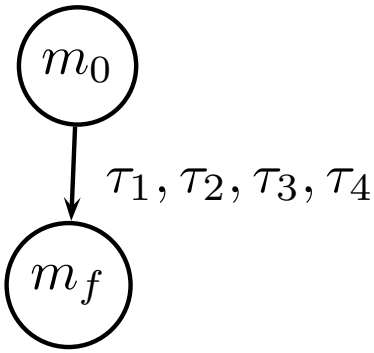
BinarySearch: Transition System

```
@pre 0 ≤ ℓ ∧ u < |a| ∧ sorted(a, 0, |a| − 1)
@post rv ↔ (∃i ∈ [ℓ, u]) a[i] = e
bool BSearch(int[] a, int ℓ, int u, int e) {
  int m;
  if (ℓ > u) return false;
  else {
    m := (ℓ + u) div 2;
    if (a[m] = e) return true;
    else if (a[m] < e) return BSearch(a, m + 1, u, e);
    else return BSearch(a, ℓ, m − 1, e);
  }
}
```



BinarySearch: Transition System

Let $\varphi_0 : \; a'_0 = a \;\wedge\; \ell'_0 = \ell \;\wedge\; u'_0 = u \;\wedge\; e'_0 = e.$



$$\rho_{\tau_1} : \; \varphi_0 \;\wedge\; \ell > u \;\wedge\; \mathsf{rv}' = \mathbf{false} \;\wedge\; \mathsf{pres}(\ldots)$$

$$\rho_{\tau_2} : \; \begin{array}{l} \varphi_0 \;\wedge\; \ell \leq u \;\wedge\; m' = \frac{\ell+u}{2} \;\wedge\; a[\frac{\ell+u}{2}] = e \\ \wedge\; \mathsf{rv}' = \mathbf{true} \;\wedge \mathsf{pres}(\ldots) \end{array}$$

$$\rho_{\tau_3} : \; \begin{array}{l} \varphi_0 \;\wedge\; \ell \leq u \;\wedge\; m' = \frac{\ell+u}{2} \;\wedge\; a[\frac{\ell+u}{2}] \neq e \;\wedge\; a[\frac{\ell+u}{2}] < e \\ \wedge\; \mathsf{rv}' = \textsc{BSearch}(a, \frac{\ell+u}{2} + 1, u, e) \;\wedge\; \mathsf{pres}(\ldots) \end{array}$$

$$\rho_{\tau_4} : \; \begin{array}{l} \varphi_0 \;\wedge\; \ell \leq u \;\wedge\; m' = \frac{\ell+u}{2} \;\wedge\; a[\frac{\ell+u}{2}] \neq e \;\wedge\; a[\frac{\ell+u}{2}] \geq e \\ \wedge\; \mathsf{rv}' = \textsc{BSearch}(a, \ell, \frac{\ell+u}{2} - 1, e) \;\wedge\; \mathsf{pres}(\ldots) \end{array}$$

BubbleSort: Transition System

```
@pre |a| ≥ 0
@post sorted(rv, 0, |a| − 1)
int[] BubbleSort(int[] a) {
  int i, j, t;
  for (i := |a| − 1; i > 0; i := i − 1) {
    for (j := 0; j < i; j := j + 1) {
      if (a[j] > a[j + 1]) {
        t := a[j];
        a[j] := a[j + 1];
        a[j + 1] := t;
      }
    }
  }
  return a;
}
```

BubbleSort: Transition System

$$\langle \{\underbrace{a, |a|}_{\mathcal{F}}, \underbrace{a_0, |a|_0}_{\mathcal{F}_0}, \underbrace{i, j, t}_{\mathcal{L}}, \mathsf{rv}\}, \langle \ell_0, \mathsf{@pre}\rangle, \langle \ell_f, \mathsf{@post}\rangle, \{\tau_0, \tau_1^T, \tau_1^F, \tau_2^1, \tau_2^2, \tau_2^3\}\rangle$$

$\tau_0 : \quad \langle \ell_0, \ell_1, \rho_{\tau_0} \rangle$

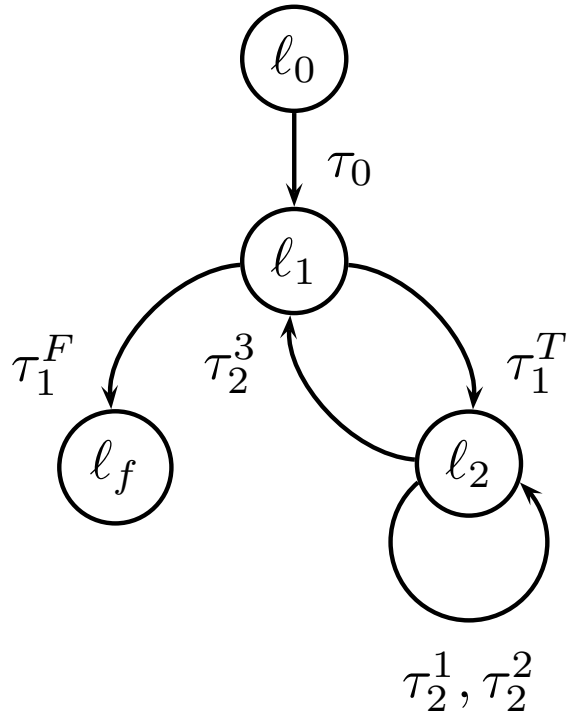
$\tau_1^T : \quad \langle \ell_1, \ell_2, \rho_{\tau_1^T} \rangle$

$\tau_1^F : \quad \langle \ell_1, \ell_f, \rho_{\tau_1^F} \rangle$

$\tau_2^1 : \quad \langle \ell_2, \ell_2, \rho_{\tau_2^1} \rangle$

$\tau_2^2 : \quad \langle \ell_2, \ell_2, \rho_{\tau_2^2} \rangle$

$\tau_2^3 : \quad \langle \ell_2, \ell_1, \rho_{\tau_2^3} \rangle$



BubbleSort: Transition System

$$\rho_{\tau_0}: \quad a'_0 = a \wedge |a|' = |a|_0 \wedge i' = |a| - 1 \wedge \mathsf{pres}(\ldots)$$

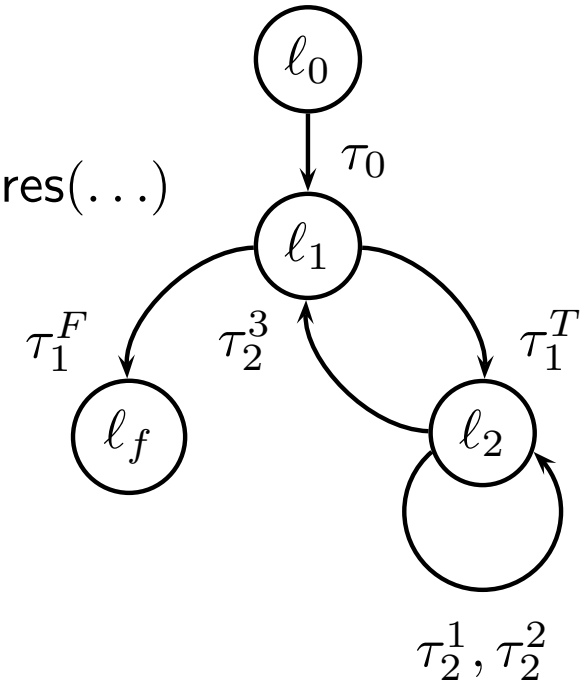
$$\rho_{\tau_1^T}: \quad i > 0 \wedge j' = 0 \wedge \mathsf{pres}(\ldots)$$

$$\rho_{\tau_1^F}: \quad i \leq 0 \wedge \mathsf{rv}' = a \wedge \mathsf{pres}(\ldots)$$

$$\rho_{\tau_2^1}: \quad \begin{array}{l} j < i \wedge a[j] > a[j+1] \wedge t' = a[j] \\ \wedge\ a' = a[j : a[j+1]][j+1 : a[j]] \wedge j' = j+1 \wedge \mathsf{pres}(\ldots) \end{array}$$

$$\rho_{\tau_2^2}: \quad j < i \wedge a[j] \leq a[j+1] \wedge j' = j+1 \wedge \mathsf{pres}(\ldots)$$

$$\rho_{\tau_2^3}: \quad j \geq i \wedge i' = i - 1 \wedge \mathsf{pres}(\ldots)$$



Translation: pi to a Transition System

One function at a time.

Variables:

- $\mathcal{V} = \mathcal{F} \cup \mathcal{F}_0 \cup \mathcal{L} \cup \{\mathsf{rv}\}$

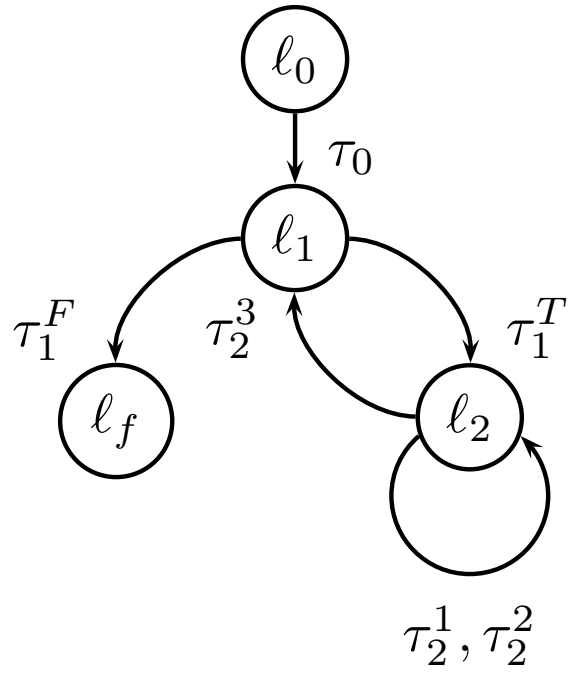
Locations:

- entry location ℓ_0
- final location ℓ_f
- one location per loop

BubbleSort: Locations

@pre $|a| \geq 0$
@post sorted(rv, $0, |a| - 1$)

```
int[] BubbleSort(int[] a) {
  int i, j, t;
  for (i := |a| − 1; i > 0; i := i − 1) {
    for (j := 0; j < i; j := j + 1) {
      if (a[j] > a[j + 1]) {
        t := a[j];
        a[j] := a[j + 1];
        a[j + 1] := t;
      }
    }
  }
  return a;
}
```



Translation: pi to a Transition System

Transitions:

- one per **straight** path of statements
- entry transition τ_0 : $\langle \ell_0, \cdot, \cdot \rangle$
  - assign $v'_0 = v$ for $v \in \mathcal{F}$
  - initialize other variables
  - straight path to first (if any) loop
- exit transitions (end in `return`): assign rv'

Example:

$$\langle \ell_0,\ \ell_1,\ a'_0 = a\ \wedge\ |a|' = |a|_0\ \wedge\ i' = |a| - 1\ \wedge\ \mathsf{pres}(\ldots) \rangle$$

$$\langle \ell_1,\ \ell_f,\ i \leq 0\ \wedge\ \mathsf{rv}' = a\ \wedge\ \mathsf{pres}(\ldots) \rangle$$

Translation: Assignments

Variable assignment, for $v \in \mathcal{V}$ and expression $e(\mathcal{V})$:

$$\ell_1 \colon v := e; \ell_2$$

$$\Downarrow$$

$$\langle \ell_1, \ell_2, v' = e \ \wedge \ \mathsf{pres}(\ldots) \rangle$$

Arrays:

$$\ell_1 \colon a[i] := e; \ell_2$$

$$\Downarrow$$

$$\langle \ell_1, \ell_2, a' = a[i : e] \ \wedge \ \mathsf{pres}(\ldots) \rangle$$

$a[i : e]$ is the array a with position i modified to e.

$\mathsf{pres}(V)$: preserve values of $V \subseteq \mathcal{V}$

$\mathsf{pres}(\ldots)$: preserve values of unmentioned variables of $\mathcal{V} - \mathcal{U}$

Examples: Assignments

$$m_3\colon\ m := (\ell + u)\ \texttt{div}\ 2;$$

$$\Downarrow$$

$$\tau_3 : \langle m_3, m_4,\ m' = \tfrac{\ell+u}{2}\ \wedge\ \mathsf{pres}(\ldots)\rangle$$

$$\ell_3\colon\ a[j+1] := t;$$

$$\Downarrow$$

$$\tau_3 :\ \langle \ell_3, \ell_4, a' = a[j+1 : t]]\ \wedge\ \mathsf{pres}(\ldots)\rangle$$

Translation: Branches

A branch produces two transitions.

$$\ell_1\texttt{: if } (e) \{\ell_2 \texttt{ : } \ldots\} \texttt{ else } \{\ell_3 \texttt{ : } \ldots\}$$

$$\Downarrow$$

$$\langle \ell_1, \ell_2, e \rangle$$

$$\langle \ell_1, \ell_3, \neg e \rangle$$

where $e(\mathcal{V})$ is a Boolean expression.

Example: Branches

$$m_1\texttt{: if } (\ell > u)$$

$$\Downarrow$$

$$\tau_1^T : \langle m_1, m_2,\ \ell > u\ \wedge\ \mathsf{pres}(\ldots)\rangle$$

$$\tau_1^F : \langle m_1, m_3,\ \ell \leq u\ \wedge\ \mathsf{pres}(\ldots)\rangle$$

Translation: Loops

`while` loops:

- Treat similarly to `if`: two branching transitions.
- Handle postlocations of `break` statements, `continue` statements, and final statement in block according to usual semantics.

`for` loops:

```
for (e1; e2; e3) {          ⇒      e1;
  ...                              while (e2) {
}                                    ...
                                     e3;
                                   }
```

Translation: Composing Transitions

Composition of transitions

$$\langle \ell_1, \ell_2, \rho_{\tau_1} \rangle;\ \langle \ell_2, \ell_3, \rho_{\tau_2} \rangle$$

is

$$\langle \ell_1, \ell_3, \rho_{\tau_2} \circ \rho_{\tau_1} \rangle$$

where

$$\rho_{\tau_2} \circ \rho_{\tau_1} \ = \ \underbrace{((\exists \mathcal{V}')[\rho_{\tau_1} \ \wedge \ \rho_{\tau_2}\{\mathcal{V} \leftarrow \mathcal{V}', \mathcal{V}' \leftarrow \mathcal{V}''\}])\,\{\mathcal{V}'' \leftarrow \mathcal{V}'\}}_{\text{assertion over } \mathcal{V} \cup \mathcal{V}'}$$

There has to be a better way.

Translation: Composing Transitions

Write ρ_τ as **guard** and **update**:

$$\rho_\tau : \; G(\mathcal{V}) \; \wedge \; \underbrace{U(\mathcal{V} \cup \mathcal{V}')}_{\bigwedge_i v_i' = e_i(\mathcal{V})}$$

Write update as a **substitution**:

$$U \; = \; \bigwedge_i v_i' = e_i$$

$$\mathsf{s}(U) \; = \; \{v_i \leftarrow e_i\}_i$$

Define inverse:

$$\mathsf{s}^{-1}(\{v_i \leftarrow e_i\}_i) \; = \; \bigwedge_i v_i' = e_i$$

Example:

$$t := a[j]; \quad \Rightarrow \quad t' = a[j] \quad \Rightarrow \quad \{t \leftarrow a[j]\}$$

Interlude: Substitutions

$$E \underbrace{\{v_i \leftarrow e_i\}_i}_{\text{substitution}}$$

In expression e, replace
each variable v_i with expression e_i
simultaneously.

Examples:

$$(x < 3 + y)\{x \leftarrow z\} = (z < 3 + y)$$

$$(x < 3 + y)\{x \leftarrow z + 1\} = (z + 1 < 3 + y)$$

$$(x < 3 + y)\{x \leftarrow y, y \leftarrow x\} = (y < 3 + x)$$

Interlude: Substitutions

For substitutions λ_1 and λ_2, the **composition** $\lambda_2 \circ \lambda_1$ has the effect of applying first λ_1 and then λ_2.

If

- $\lambda_1 = \{u_i \leftarrow e_i\}$
- $\lambda_2 = \{v_i \leftarrow f_i\}$

then

$$\lambda_2 \circ \lambda_1 = \{u_i \leftarrow e_i\lambda_2\} \cup \{v_i \leftarrow f_i \ : \ v_i \not\equiv u_j \text{ for any } j\}$$

Write $e(\lambda_2 \circ \lambda_1)$ as $(e\lambda_1)\lambda_2$ or $e(\lambda_1 \Box \lambda_2)$.

Interlude: Substitutions

Examples:

$$
\begin{aligned}
&(x+y)(\{y \leftarrow x+1\} \circ \{x \leftarrow y+1\}) \\
&\quad = (x+y)\{x \leftarrow (x+1)+1, y \leftarrow x+1\} \\
&\quad = ((x+1)+1)+(x+1) \\
&\quad = 2x+3
\end{aligned}
$$

$$
\begin{aligned}
&\{t \leftarrow a[j]\} \circ \{a \leftarrow a[j : a[j+1]]\} \\
&\quad = \{a \leftarrow a[j : a[j+1]], t \leftarrow a[j]\}
\end{aligned}
$$

$$
\begin{aligned}
&\{a \leftarrow a[j : a[j+1]], t \leftarrow a[j]\} \circ \{a \leftarrow a[j+1 : t]\} \\
&\quad = \{a \leftarrow a[j : a[j+1]][j+1 : a[j]], t \leftarrow a[j]\}
\end{aligned}
$$

Interlude: Substitutions

Composition of substitutions is **associative**.

$$\lambda_1 = \{u_i \leftarrow e_i\} \qquad \lambda_2 = \{v_i \leftarrow f_i\} \qquad \lambda_3 = \{w_i \leftarrow g_i\}$$

$$\begin{aligned}
\lambda_3 \circ (\lambda_2 \circ \lambda_1) &= \lambda_3 \circ (\{u_i \leftarrow e_i\lambda_2\} \cup \{v_i \leftarrow f_i \ : \ v_i \not\equiv u_j\}) \\
&= \{u_i \leftarrow (e_i\lambda_2)\lambda_3\} \cup \{v_i \leftarrow f_i\lambda_3 \ : \ v_i \not\equiv u_j\} \\
&\quad \cup \{w_i \leftarrow g_i \ : \ w_i \not\equiv u_j, \ w_i \not\equiv v_j\}
\end{aligned}$$

$$\begin{aligned}
(\lambda_3 \circ \lambda_2) \circ \lambda_1 &= (\{v_i \leftarrow f_i\lambda_3\} \cup \{w_i \leftarrow g_i \ : \ w_i \not\equiv v_j\}) \circ \lambda_1 \\
&= \underbrace{\{u_i \leftarrow e_i(\lambda_3 \circ \lambda_2)\}}_{\text{compute directly}} \cup \{v_i \leftarrow f_i\lambda_3 \ : \ v_i \not\equiv u_j\} \\
&\quad \cup \{w_i \leftarrow g_i \ : \ w_i \not\equiv u_j, \ w_i \not\equiv v_j\}
\end{aligned}$$

$$\Rightarrow \ \lambda_3 \circ (\lambda_2 \circ \lambda_1) = (\lambda_3 \circ \lambda_2) \circ \lambda_1$$

$\Rightarrow$ Compose forwards or backwards — no difference!

Translation: Composing Transitions

Composition of transitions

$$\langle \ell_1, \ell_2, \rho_{\tau_1} \rangle;\ \langle \ell_2, \ell_3, \rho_{\tau_2} \rangle$$

for

$$\rho_{\tau_1} :\ G_1 \wedge U_1 \qquad \lambda_1 = \mathsf{s}(U_1)$$
$$\rho_{\tau_2} :\ G_2 \wedge U_2 \qquad \lambda_2 = \mathsf{s}(U_2)$$

is

$$\langle \ell_1, \ell_3, \rho_{\tau} \rangle$$

where

$$\rho_\tau \ = \ G_1 \ \wedge \ G_2\lambda_1 \ \wedge \ \mathsf{s}^{-1}(\lambda_1 \circ \lambda_2) \ \wedge \ \mathsf{pres}(\ldots)$$

Composition of substitutions is associative.

$\Rightarrow$ Composition of transitions is associative.

$\Rightarrow$ Compose forwards or backwards — no difference!

Example 1: Composing Transitions

ℓ_1: $y := x + 1$; $\quad\quad$ $\tau_1 : \ \langle \ell_1, \ell_2, y' = x + 1 \wedge \mathsf{pres}(\ldots) \rangle$

ℓ_2: $x := y + 1$; $\quad\quad$ $\tau_2 : \ \langle \ell_2, \ell_3, x' = y + 1 \wedge \mathsf{pres}(\ldots) \rangle$

ℓ_3:

$$\tau_1 ; \tau_2 \ = \ \langle \ell_1, \ell_3, x' = x + 2 \ \wedge \ y' = x + 1 \ \wedge \ \mathsf{pres}(\ldots) \rangle$$

$$\lambda_1 \ = \ \mathsf{s}(U_1) \ = \ \{ y \leftarrow x + 1 \}$$
$$\lambda_2 \ = \ \mathsf{s}(U_2) \ = \ \{ x \leftarrow y + 1 \}$$
$$\lambda_1 \circ \lambda_2 = \{ x \leftarrow (x + 1) + 1, y \leftarrow x + 1 \}$$

$$G_1 \equiv G_2 \equiv \mathbf{true}$$
$$\mathsf{s}^{-1}(\lambda_1 \circ \lambda_2) \ = \ x' = x + 2 \ \wedge \ y' = x + 1$$
$$\Downarrow$$
$$G_1 \ \wedge \ G_2 \lambda_1 \ \wedge \ \mathsf{s}^{-1}(\lambda_1 \circ \lambda_2) \ = \ x' = x + 2 \ \wedge \ y' = x + 1$$

Example 2: Composing Transitions

m_1: `if` $(\ell > u)$ $\quad \tau_1^T : \langle m_1, m_2,\ \ell > u \ldots \rangle$

$\quad \tau_1^F : \langle m_1, m_3,\ \ell \leq u \ldots \rangle$

m_2: `return` **false**; $\quad \tau_2 : \langle m_2, m_f,\ \mathsf{rv}' = \textbf{false} \ldots \rangle$

m_3: $m := (\ell + u)$ `div` 2; $\quad \tau_3 : \langle m_3, m_4,\ m' = \frac{\ell+u}{2} \ldots \rangle$

m_4: `if` $(a[m] = e)$ $\quad \tau_4^T : \langle m_4, m_5,\ a[m] = e \ldots \rangle$

$\quad \tau_4^F : \langle m_4, m_6,\ a[m] \neq e \ldots \rangle$

m_5: `return` **true**; $\quad \tau_5 : \langle m_5, m_f,\ \mathsf{rv}' = \textbf{true} \ldots \rangle$

m_6: `if` $(a[m] < e)$ $\quad \tau_6^T : \langle m_6, m_7,\ a[m] < e \ldots \rangle$

$\quad \tau_6^F : \langle m_6, m_8,\ a[m] \geq e \ldots \rangle$

m_7: `return` $\textsc{BSearch}(a, m+1, u, e)$;

$\tau_7 : \langle m_7, m_f,\ \mathsf{rv}' = \textsc{BSearch}(a, m+1, u, e) \ldots \rangle$

m_8: `return` $\textsc{BSearch}(a, \ell, m-1, e)$;

$\tau_8 : \langle m_8, m_f,\ \mathsf{rv}' = \textsc{BSearch}(a, \ell, m-1, e) \ldots \rangle$

$\ldots : \wedge\ \mathsf{pres}(\ldots)$

Example 2: Composing Transitions

Let $\varphi_0 :\ a'_0 = a \ \wedge\ \ell'_0 = \ell \ \wedge\ u'_0 = u \ \wedge\ e'_0 = e.$

$\tau_0; \tau_1^F; \tau_3; \tau_4^T; \tau_5$:

$\tau_0; \tau_1^F \quad \langle m_0, m_3,\ \varphi_0 \ \wedge\ \ell \leq u \ldots \rangle$

$; \tau_3 \quad \langle m_0, m_4,\ \varphi_0 \ \wedge\ \ell \leq u \ \wedge\ m' = \frac{\ell+u}{2} \ldots \rangle$

$; \tau_4^T \quad \mathsf{s}(\tau_0; \tau_1^F; \tau_3) = \{m \leftarrow \frac{\ell+u}{2}\} = \lambda_1$

$\langle m_0, m_5,\ \varphi_0 \ \wedge\ \ell \leq u \ \wedge\ m' = \frac{\ell+u}{2} \wedge\ (a[m] = e)\lambda_1 \ \ldots \rangle$

$\langle m_0, m_5,\ \varphi_0 \ \wedge\ \ell \leq u \ \wedge\ m' = \frac{\ell+u}{2} \wedge\ a[\frac{\ell+u}{2}] = e \ldots \rangle$

$; \tau_5 \quad \mathsf{s}(\tau_0; \tau_1^F; \tau_3; \tau_5) = \{m \leftarrow \frac{\ell+u}{2}\} = \lambda_1$

$\mathsf{s}(\tau_5) = \{rv \leftarrow \mathbf{true}\} = \lambda_2$

$\langle m_0, m_f,\ \varphi_0 \ \wedge\ \ell \leq u \ \wedge\ \mathsf{s}^{-1}(\lambda_1 \circ \lambda_2) \ldots \rangle$

$$\left\langle m_0, m_f,\ \left(\begin{array}{l} \varphi_0 \ \wedge\ \ell \leq u \ \wedge\ m' = \frac{\ell+u}{2} \wedge\ a[\frac{\ell+u}{2}] = e \\ \wedge\ \mathsf{rv}' = \mathbf{true} \ \wedge\ \mathsf{pres}(\ldots) \end{array} \right) \right\rangle$$

Example 3: Composing Transitions

ℓ_1: $t := a[j]$; $\quad\quad$ $\tau_1 : \ \langle \ell_1, \ell_2, t' = a[j] \wedge \mathsf{pres}(\ldots) \rangle$

ℓ_2: $a[j] := a[j+1]$; $\quad\quad$ $\tau_2 : \ \langle \ell_2, \ell_3, a' = a[j : a[j+1]] \wedge \mathsf{pres}(\ldots) \rangle$

ℓ_3: $a[j+1] := t$; $\quad\quad$ $\tau_3 : \ \langle \ell_3, \ell_4, a' = a[j+1 : t]] \wedge \mathsf{pres}(\ldots) \rangle$

ℓ_4:

$$\tau_1; \tau_2 \ = \ \langle \ell_1, \ell_3, a' = a[j : a[j+1]] \ \wedge \ t' = a[j] \ \wedge \ \mathsf{pres}(\ldots) \rangle$$

$$\lambda_1 \ = \ \mathsf{s}(U_1) \ = \ \{t \leftarrow a[j]\}$$
$$\lambda_2 \ = \ \mathsf{s}(U_2) \ = \ \{a \leftarrow a[j : a[j+1]]\}$$
$$\lambda_1 \circ \lambda_2 = \{a \leftarrow a[j : a[j+1]], t \leftarrow a[j]\}$$

$$G_1 \equiv G_2 \equiv \mathbf{true}$$
$$\mathsf{s}^{-1}(\lambda_1 \circ \lambda_2) \ = \ a' = a[j : a[j+1]] \ \wedge \ t' = a[j]$$
$$\Downarrow$$
$$G_1 \ \wedge \ G_2 \lambda_1 \ \wedge \ \mathsf{s}^{-1}(\lambda_1 \circ \lambda_2) \ = \ a' = a[j : a[j+1]] \ \wedge \ t' = a[j]$$

Example 3: Composing Transitions

ℓ_1: $t := a[j]$; $\qquad\qquad$ $\tau_1 : \ \langle \ell_1, \ell_2, t' = a[j] \wedge \mathsf{pres}(\ldots) \rangle$

ℓ_2: $a[j] := a[j+1]$; $\qquad$ $\tau_2 : \ \langle \ell_2, \ell_3, a' = a[j : a[j+1]] \wedge \mathsf{pres}(\ldots) \rangle$

ℓ_3: $a[j+1] := t$; $\qquad\quad$ $\tau_3 : \ \langle \ell_3, \ell_4, a' = a[j+1 : t]] \wedge \mathsf{pres}(\ldots) \rangle$

ℓ_4:

$$(\tau_1; \tau_2); \tau_3 \ =$$
$$\langle \ell_1, \ell_4, a' = a[j : a[j+1]][j+1 : a[j]] \ \wedge \ t' = a[j] \ \wedge \ \mathsf{pres}(\ldots) \rangle$$

$$\lambda_1 \ = \ \mathsf{s}(U_{1,2}) \ = \ \{a \leftarrow a[j : a[j+1]], t \leftarrow a[j]\}$$
$$\lambda_2 \ = \ \mathsf{s}(U_3) \ = \ \{a \leftarrow a[j+1 : t]\}$$
$$\lambda_1 \circ \lambda_2 = \{a \leftarrow a[j : a[j+1]][j+1 : a[j]], t \leftarrow a[j]\}$$

$$G_1 \equiv G_2 \equiv \mathbf{true}$$
$$\mathsf{s}^{-1}(\lambda_1 \circ \lambda_2) \ = \ a' = a[j : a[j+1]][j+1 : a[j]] \ \wedge \ t' = a[j]$$

Example 4: Composing Transitions

ℓ_0: `if` $(a[j] > a[j+1])$ $\quad\quad$ τ_0^T : $\langle \ell_0, \ell_1, a[j] > a[j+1] \wedge \mathsf{pres}(\ldots)\rangle$

ℓ_1: $t := a[j];$ $\quad\quad$ τ_1 : $\langle \ell_1, \ell_2, t' = a[j] \wedge \mathsf{pres}(\ldots)\rangle$

ℓ_2: $a[j] := a[j+1];$ $\quad\quad$ τ_2 : $\langle \ell_2, \ell_3, a' = a[j : a[j+1]] \wedge \mathsf{pres}(\ldots)\rangle$

ℓ_3: $a[j+1] := t;$ $\quad\quad$ τ_3 : $\langle \ell_3, \ell_4, a' = a[j+1 : t]] \wedge \mathsf{pres}(\ldots)\rangle$

ℓ_4:

$$\tau_0; (\tau_1; \tau_2; \tau_3) \;=\; \left\langle \begin{array}{l} \ell_0, \ell_4, \\ a[j] > a[j+1] \;\wedge\; a' = a[j : a[j+1]][j+1 : a[j]] \\ \quad \wedge\; t' = a[j] \;\wedge\; \mathsf{pres}(\ldots) \end{array} \right\rangle$$

Translation: Programs

Represent programs as a set of transition systems

$$\{T_1, \ldots, T_n\}$$

Simplifications:

- No global variables.
  - No loss of generality.
  - Simulate globals with function arguments.
- Pass-by-value semantics.
  - Handling aliasing is complicated and out of scope of class.
  - Current research addressing aliasing.
  - Return multiple values via `struct` type.

BinarySearch: To a Transition System

```
@pre |a| ≥ 0 ∧ sorted(a, 0, |a| − 1)
@post rv ↔ (∃i ∈ [0, |a| − 1]) a[i] = e
bool BinarySearch(int[] a, int e) {
  return BSearch(a, 0, |a| − 1, e);
}

@pre 0 ≤ ℓ ∧ u < |a| ∧ sorted(a, 0, |a| − 1)
@post rv ↔ (∃i ∈ [ℓ, u]) a[i] = e
bool BSearch(int[] a, int ℓ, int u, int e) {
  int m;
  if (ℓ > u) return false;
  else {
    m := (ℓ + u) div 2;
    if (a[m] = e) return true;
    else if (a[m] < e) return BSearch(a, m + 1, u, e);
    else return BSearch(a, ℓ, m − 1, e);
  }
}
```

BinarySearch: To a Transition System

m_1: `if` $(\ell > u)$ — $\tau_1^T : \langle m_1, m_2,\ \ell > u \ldots \rangle$

— $\tau_1^F : \langle m_1, m_3,\ \ell \leq u \ldots \rangle$

m_2: `return` **false**; — $\tau_2 : \langle m_2, m_f,\ \mathsf{rv}' = \mathbf{false} \ldots \rangle$

m_3: $m := (\ell + u)$ `div` 2; — $\tau_3 : \langle m_3, m_4,\ m' = \frac{\ell+u}{2} \ldots \rangle$

m_4: `if` $(a[m] = e)$ — $\tau_4^T : \langle m_4, m_5,\ a[m] = e \ldots \rangle$

— $\tau_4^F : \langle m_4, m_6,\ a[m] \neq e \ldots \rangle$

m_5: `return` **true**; — $\tau_5 : \langle m_5, m_f,\ \mathsf{rv}' = \mathbf{true} \ldots \rangle$

m_6: `if` $(a[m] < e)$ — $\tau_6^T : \langle m_6, m_7,\ a[m] < e \ldots \rangle$

— $\tau_6^F : \langle m_6, m_8,\ a[m] \geq e \ldots \rangle$

m_7: `return` $\textsc{BSearch}(a, m+1, u, e)$;

$$\tau_7 : \langle m_7, m_f,\ \mathsf{rv}' = \textsc{BSearch}(a, m+1, u, e) \ldots \rangle$$

m_8: `return` $\textsc{BSearch}(a, \ell, m-1, e)$;

$$\tau_8 : \langle m_8, m_f,\ \mathsf{rv}' = \textsc{BSearch}(a, \ell, m-1, e) \ldots \rangle$$

$\ldots : \wedge\ \mathsf{pres}(\ldots)$

BinarySearch: To a Transition System

Initial transition:

$$\tau_0 : \langle m_0, m_1,\ a'_0 = a\ \wedge\ \ell'_0 = \ell\ \wedge u' = u\ \wedge\ e' = e\ \wedge \mathsf{pres}(\ldots)\rangle$$

Four composed transitions from m_0 to m_f:

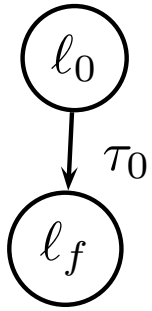
$$\tau_0; \tau_1^T; \tau_2$$
$$\tau_0; \tau_1^F; \tau_3; \tau_4^T; \tau_5$$
$$\tau_0; \tau_1^F; \tau_3; \tau_4^F; \tau_6^T; \tau_7$$
$$\tau_0; \tau_1^F; \tau_3; \tau_4^F; \tau_6^F; \tau_8$$

BinarySearch: To a Transition System

@pre $|a| \geq 0 \ \wedge \ \mathsf{sorted}(a, 0, |a| - 1)$

@post rv $\leftrightarrow \ (\exists i \in [0, |a| - 1])\ a[i] = e$

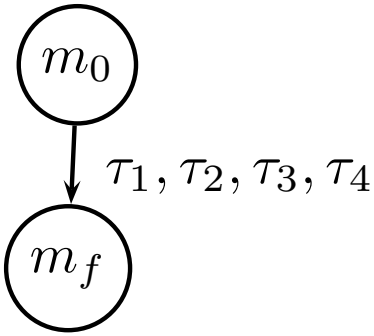
```
bool BinarySearch(int[] a, int e) {
  return BSearch(a, 0, |a| − 1, e);
}
```



$$\rho_{\tau_0}: \quad \begin{array}{l} a'_0 = a \ \wedge \ e'_0 = e \ \wedge \ |a|'_0 = |a| \\ \wedge \ \mathsf{rv}' = \textsc{BSearch}(a, 0, |a| - 1, e) \ \wedge \ \mathsf{pres}(\ldots) \end{array}$$

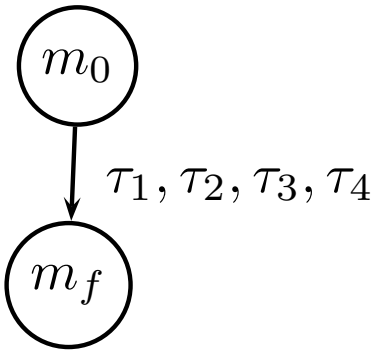
BinarySearch: To a Transition System

```
@pre 0 ≤ ℓ ∧ u < |a| ∧ sorted(a, 0, |a| − 1)
@post rv ↔ (∃i ∈ [ℓ, u]) a[i] = e
bool BSearch(int[] a, int ℓ, int u, int e) {
  int m;
  if (ℓ > u) return false;
  else {
    m := (ℓ + u) div 2;
    if (a[m] = e) return true;
    else if (a[m] < e) return BSearch(a, m + 1, u, e);
    else return BSearch(a, ℓ, m − 1, e);
  }
}
```



BinarySearch: To a Transition System

Let $\varphi_0 : \; a'_0 = a \; \wedge \; \ell'_0 = \ell \; \wedge \; u'_0 = u \; \wedge \; e'_0 = e.$



$$\rho_{\tau_1} : \; \varphi_0 \; \wedge \; \ell > u \; \wedge \; \mathsf{rv}' = \mathbf{false} \; \wedge \; \mathsf{pres}(\ldots)$$

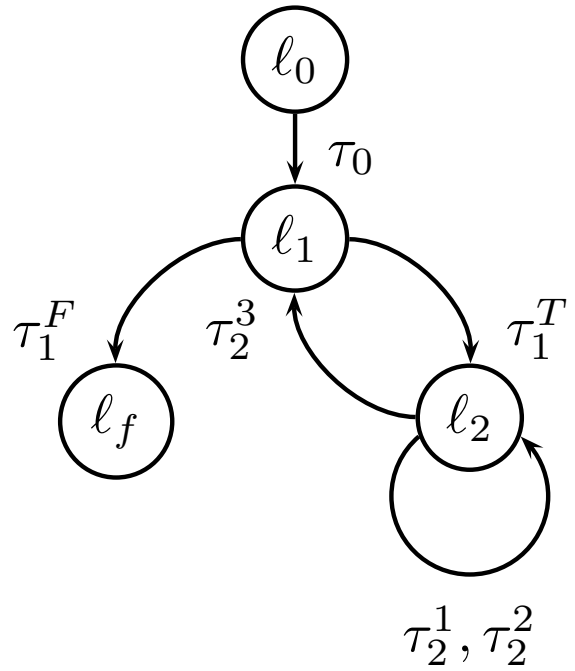
$$\rho_{\tau_2} : \; \begin{array}{l} \varphi_0 \; \wedge \; \ell \leq u \; \wedge \; m' = \frac{\ell+u}{2} \; \wedge \; a[\frac{\ell+u}{2}] = e \\ \wedge \; \mathsf{rv}' = \mathbf{true} \; \wedge \mathsf{pres}(\ldots) \end{array}$$

$$\rho_{\tau_3} : \; \begin{array}{l} \varphi_0 \; \wedge \; \ell \leq u \; \wedge \; m' = \frac{\ell+u}{2} \; \wedge \; a[\frac{\ell+u}{2}] \neq e \; \wedge \; a[\frac{\ell+u}{2}] < e \\ \wedge \; \mathsf{rv}' = \textsc{BSearch}(a, \frac{\ell+u}{2} + 1, u, e) \; \wedge \; \mathsf{pres}(\ldots) \end{array}$$

$$\rho_{\tau_4} : \; \begin{array}{l} \varphi_0 \; \wedge \; \ell \leq u \; \wedge \; m' = \frac{\ell+u}{2} \; \wedge \; a[\frac{\ell+u}{2}] \neq e \; \wedge \; a[\frac{\ell+u}{2}] \geq e \\ \wedge \; \mathsf{rv}' = \textsc{BSearch}(a, \ell, \frac{\ell+u}{2} - 1, e) \; \wedge \; \mathsf{pres}(\ldots) \end{array}$$

BubbleSort: To a Transition System

```
@pre |a| ≥ 0
@post sorted(rv, 0, |a| − 1)
int[] BubbleSort(int[] a) {
  int i, j, t;
  for (i := |a| − 1; i > 0; i := i − 1) {
    for (j := 0; j < i; j := j + 1) {
      if (a[j] > a[j + 1]) {
        t := a[j];
        a[j] := a[j + 1];
        a[j + 1] := t;
      }
    }
  }
  return a;
}
```



$\langle \{a, |a|, a_0, |a|_0, i, j, t\}, \langle \ell_0, \text{@pre} \rangle, \langle \ell_f, \text{@post} \rangle, \{\tau_0, \tau_1^T, \tau_1^F, \tau_2^1, \tau_2^2, \tau_2^3\} \rangle$

BubbleSort: To a Transition System

$$\rho_{\tau_0}: \quad a'_0 = a \wedge |a|' = |a|_0 \wedge i' = |a| - 1 \wedge \mathsf{pres}(\ldots)$$

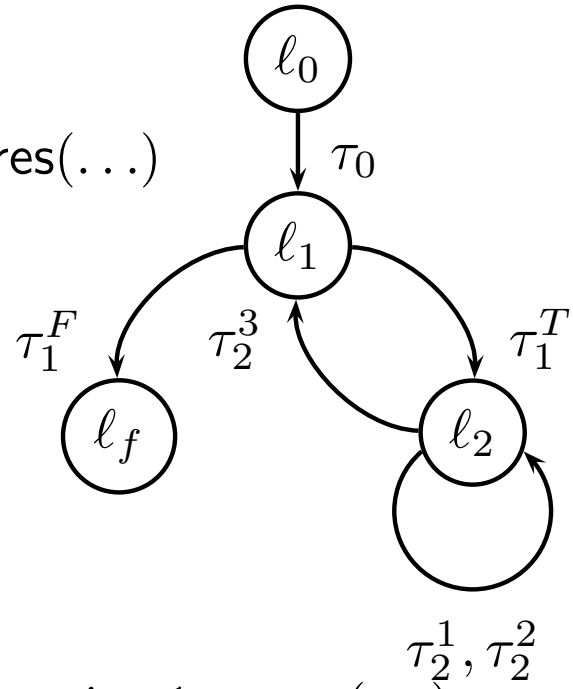
$$\rho_{\tau_1^T}: \quad i > 0 \wedge j' = 0 \wedge \mathsf{pres}(\ldots)$$

$$\rho_{\tau_1^F}: \quad i \leq 0 \wedge \mathsf{rv}' = a \wedge \mathsf{pres}(\ldots)$$

$$\rho_{\tau_2^1}: \quad \begin{array}{l} j < i \wedge a[j] > a[j+1] \wedge t' = a[j] \\ \wedge\ a' = a[j : a[j+1]][j+1 : a[j]] \wedge j' = j+1 \wedge \mathsf{pres}(\ldots) \end{array}$$

$$\rho_{\tau_2^2}: \quad j < i \wedge a[j] \leq a[j+1] \wedge j' = j+1 \wedge \mathsf{pres}(\ldots)$$

$$\rho_{\tau_2^3}: \quad j \geq i \wedge i' = i - 1 \wedge \mathsf{pres}(\ldots)$$



Outline

1. Transition Systems
2. ⇒ Partial Correctness
3. Total Correctness

BinarySearch: Motivation

```
bool BinarySearch(int[] a, int e) {
  return BSearch(a, 0, |a| − 1, e);
}

bool BSearch(int[] a, int ℓ, int u, int e) {
  int m;
  if (ℓ > u) return false;
  else {
    m := (ℓ + u) div 2;
    if (a[m] = e) return true;
    else if (a[m] < e) return BSearch(a, m + 1, u, e);
    else return BSearch(a, ℓ, m − 1, e);
  }
}
```

Does BinarySearch return true iff a contains e?

BubbleSort: Motivation

```
int[] BubbleSort(int[] a) {
  int i, j, t;
  for (i := |a| − 1; i > 0; i := i − 1) {
    for (j := 0; j < i; j := j + 1) {
      if (a[j] > a[j + 1]) {
        t := a[j];
        a[j] := a[j + 1];
        a[j + 1] := t;
      }
    }
  }
  return a;
}
```

Does BubbleSort return a sorted array?

Motivation

Given: Transition system $T : \langle \mathcal{V}, \Theta : \langle \ell_0, \theta \rangle, \Omega : \langle \ell_f, \omega \rangle, \mathcal{T} \rangle$

Goal:

Prove that if

initially θ holds

then

if ℓ_f is reached, then ω holds.

Strategy:

- Annotate each location with an **assertion**.
- Prove that the assertions are **inductive**.

BubbleSort: Partial Correctness

@pre $|a| \geq 0$

@post sorted(rv, $0, |a| - 1$)

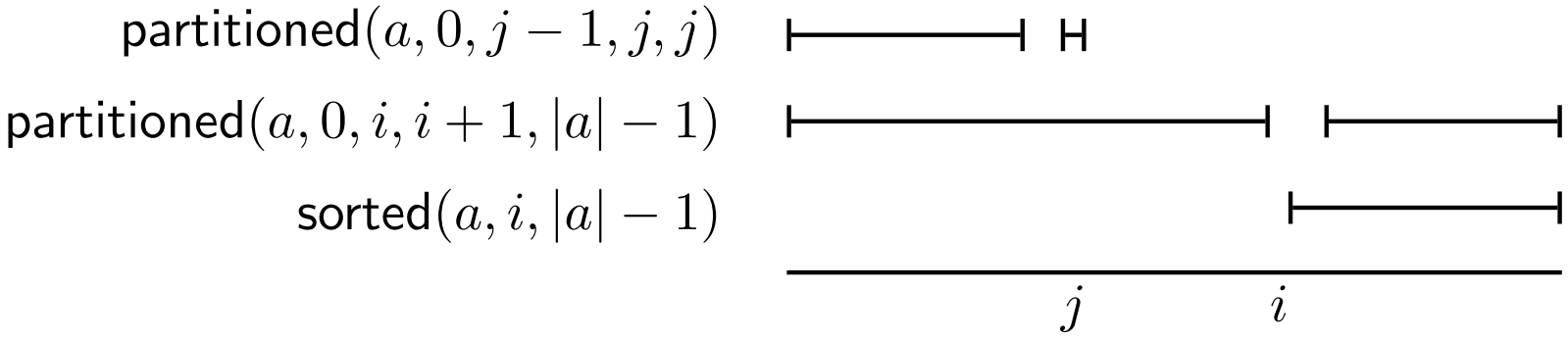
```
int[] BubbleSort(int[] a) {
  int i, j, t;
  for @ [−1 ≤ i < |a| ∧ |a| = |a|_0
         ∧ partitioned(a, 0, i, i + 1, |a| − 1) ∧ sorted(a, i, |a| − 1)]
  (i := |a| − 1; i > 0; i := i − 1)
      for @ [1 ≤ i < |a| ∧ 0 ≤ j ≤ i ∧ |a| = |a|_0
             ∧ partitioned(a, 0, i, i + 1, |a| − 1)
             ∧ partitioned(a, 0, j − 1, j, j) ∧ sorted(a, i, |a| − 1)]
      (j := 0; j < i; j := j + 1)
        if (a[j] > a[j + 1]) {
          t := a[j]; a[j] := a[j + 1]; a[j + 1] := t;
        }
  return a;
}
```

BubbleSort: Partial Correctness

Predicates:

- $\mathsf{sorted}(a, \ell, u)$: array a is sorted in range $[\ell, u]$
- $\mathsf{partitioned}(a, \ell_1, u_1, \ell_2, u_2)$:
  when $\ell_1 \leq u_1 < \ell_2 \leq u_2$,
  for $i \in [\ell_1, u_1]$ and $j \in [\ell_2, u_2]$,
  $a[i] \leq a[j]$

At the top of the inner loop:



Assertion Map

For transition system T, $\mathsf{locs}(T)$ are its locations.

Assertion Map:
Given $T : \langle \mathcal{V}, \Theta, \Omega, \mathcal{T} \rangle$,

$$\mu : \mathsf{locs}(T) \rightarrow \mathcal{A}(\mathcal{V})$$

maps T's locations to assertions over T's variables.
Special locations:

- $\mu(\ell_0) = \theta$
- $\mu(\ell_f) = \omega$

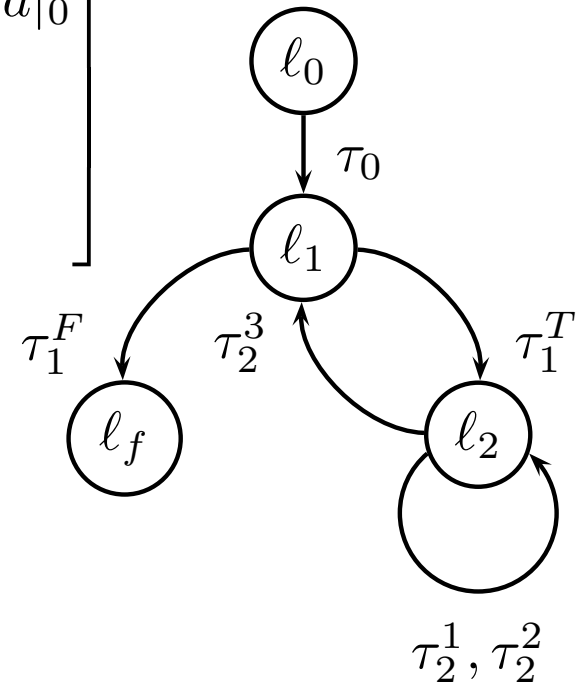
BubbleSort: Assertion Map

$$\mu(\ell_0) = |a| \geq 0$$

$$\mu(\ell_1) = \begin{bmatrix} -1 \leq i < |a| \ \wedge \ |a| = |a|_0 \\ \wedge \ \mathsf{partitioned}(a, 0, i, i+1, |a|-1) \ \wedge \ \mathsf{sorted}(a, i, |a|-1) \end{bmatrix}$$

$$\mu(\ell_2) = \begin{bmatrix} 1 \leq i < |a| \ \wedge \ 0 \leq j \leq i \ \wedge \ |a| = |a|_0 \\ \wedge \ \mathsf{partitioned}(a, 0, i, i+1, |a|-1) \\ \wedge \ \mathsf{partitioned}(a, 0, j-1, j, j) \\ \wedge \ \mathsf{sorted}(a, i, |a|-1) \end{bmatrix}$$

$$\mu(\ell_f) = \mathsf{sorted}(\mathsf{rv}, 0, |a|-1)$$



Invariant Map

An assertion map μ for $T : \langle \mathcal{V}, \Theta, \Omega, \mathcal{T} \rangle$ is an **invariant map** if

for each $\tau : \langle \ell, \ell', \rho_\tau \rangle \in \mathcal{T}$,
if $\mu(\ell)$ holds now,
then taking τ makes $\mu(\ell')$ hold next:

$$(\forall \mathcal{V} \cup \mathcal{V}')[\mu(\ell) \wedge \rho_\tau \ \rightarrow \ \mu(\ell')']$$

Conditions are called **verification conditions** (VCs).

Note the similarity to **mathematical induction**.

Verification Conditions

Idea:

Prove correctness by proving validity of a (finite) set of first-order assertions.

Generating VCs:

- One for each transition $\langle \ell, \ell', \rho_\tau \rangle$:
  Prove that if $\mu(\ell)$ holds and ρ_τ is taken, then $\mu(\ell')$ holds.
- One for each function call $\textsc{f}(\overrightarrow{a})$:
  Prove that precondition of $\textsc{f}$ holds for arguments $\overrightarrow{a}$.

Verification Conditions

Notation:

$$\{\varphi\}\rho_\tau\{\psi\} \quad \Rightarrow \quad (\forall \mathcal{V} \cup \mathcal{V}')[\varphi \wedge \rho_\tau \ \rightarrow \ \psi']$$

Easy simplification: Use substitutions.
When ρ_τ can be expressed as $G(\mathcal{V}) \wedge U(\mathcal{V} \cup \mathcal{V}')$,

$$\{\varphi\}\rho_\tau\{\psi\} \quad \Rightarrow \quad (\forall \mathcal{V})[\varphi \wedge G \ \rightarrow \ \psi(\mathsf{s}(U))]$$

Example: $\textsc{BubbleSort}(\{\mu(\ell_0)\}\rho_{\tau_0}\{\mu(\ell_1)\})$

$$(\forall a, |a|, a_0, |a|_0, i, j, t, \mathsf{rv}, a', |a|', a'_0, |a|'_0, i', j', t', \mathsf{rv}')$$
$$\Big[\underbrace{|a| \geq 0}_{\mu(\ell_0)} \ \wedge \ \underbrace{a'_0 = a \ \wedge \ |a|' = |a|_0 \ \wedge \ i' = |a| - 1 \ \wedge \ \mathsf{pres}(\ldots)}_{\rho_{\tau_0}}$$
$$\Big[\underbrace{\begin{array}{ll} & \Big[-1 \leq i' < |a|' \ \wedge \ |a|' = |a|'_0 \\ \rightarrow & \Big[\wedge \ \mathsf{partitioned}(a', 0, i', i'+1, |a|'-1) \\ & \Big[\wedge \ \mathsf{sorted}(a', i', |a|'-1)\Big]\end{array}}_{\mu(\ell_1)'}\Big]$$

More simply:

$$(\forall a, |a|, |a|_0)$$
$$\Big[|a| > 0 \ \rightarrow$$
$$\Big[-1 \leq |a| - 1 < |a| \ \wedge \ |a| = |a|_0$$
$$\Big[\wedge \ \mathsf{partitioned}(a, 0, |a|-1, |a|, |a|-1)$$
$$\Big[\wedge \ \mathsf{sorted}(a, |a|-1, |a|-1)\Big]\Big]\Big]$$

BubbleSort: Verification Conditions

Five other VCs:

$$\{\mu(\ell_1)\}\rho_{\tau_1^T}\{\mu(\ell_2)\}$$

$$\{\mu(\ell_1)\}\rho_{\tau_1^F}\{\mu(\ell_f)\}$$

$$\{\mu(\ell_2)\}\rho_{\tau_2^1}\{\mu(\ell_2)\}$$

$$\{\mu(\ell_2)\}\rho_{\tau_2^2}\{\mu(\ell_2)\}$$

$$\{\mu(\ell_2)\}\rho_{\tau_2^3}\{\mu(\ell_1)\}$$

$\Rightarrow$ BubbleSort returns a sorted array

What about Function Calls?

If transition relation contains function calls,
replace according to postconditions in VC.

Calling function F. Called function G.

Function $\textsc{g}(\overrightarrow{p})$ has

- vector of formal parameters $\overrightarrow{p}$, $p_i \in \mathcal{F}^g$
- postcondition assertion $\omega(\mathcal{F}_0^g \cup \{\mathsf{rv}\})$

$$\langle \ell_1, \ell_2, v' = \textsc{g}(\overrightarrow{a}) \ \wedge \ \mathsf{pres}(\ldots) \rangle$$

$$\Downarrow$$

$$\langle \ell_1, \ell_2, v' = u \ \wedge \ \omega\{\mathsf{rv} \leftarrow u, \ \overrightarrow{p}_0 \leftarrow \overrightarrow{a}\} \ \wedge \ \mathsf{pres}(\ldots) \rangle$$

where u is a fresh variable.
$\overrightarrow{a}$ is the **formal arguments**, a vector of expressions over $\mathcal{V}^f$.

Example: Function Calls

$$\varphi_0 \ \wedge \ \ell \leq u \ \wedge \ m' = \tfrac{\ell+u}{2} \ \wedge \ a[\tfrac{\ell+u}{2}] \neq e \ \wedge \ a[\tfrac{\ell+u}{2}] < e$$
$$\wedge \ \mathsf{rv}' = \textsc{BSearch}(a, \tfrac{\ell+u}{2}+1, u, e) \ \wedge \ \mathsf{pres}(\ldots)$$

$$\Downarrow$$

$$\varphi_0 \ \wedge \ \ell \leq u \ \wedge \ m' = \tfrac{\ell+u}{2} \ \wedge \ a[\tfrac{\ell+u}{2}] \neq e \ \wedge \ a[\tfrac{\ell+u}{2}] < e \ \wedge \ \mathsf{rv}' = u$$
$$\wedge \ u \leftrightarrow (\exists i \in [\tfrac{\ell+u}{2}+1, u]) \ a[i] = e \ \wedge \ \mathsf{pres}(\ldots)$$

Verification Conditions: Function Calls

Need to generate VC for each function call.

Solution:

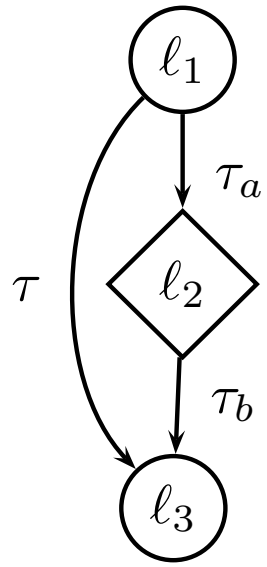
Generate **intermediate assertions** and proceed as usual.

Intermediate Assertions

So far, assertions only appear as

- preconditions (`@pre`)
- postconditions (`@post`)
- loop assertions (`while @() ...`, `for @() ...`)

Assertions can appear anywhere.



Circle nodes are initial, final, or loop nodes.

Diamond nodes are intermediate nodes.

$\tau = \tau_a; \tau_b$

VCs:

- $\{\mu(\ell_1)\}\rho_{\tau_a}\{\mu(\ell_2)\}$
- $\{\mu(\ell_1)\}\rho_{\tau}\{\mu(\ell_3)\}$

Intermediate Assertions: Function Preconditions

For function $\textsc{g}(\overrightarrow{p})$ with formal parameters $\overrightarrow{p}$ and statement

$$\ell_1\colon\ v := \textsc{g}(\overrightarrow{a});\ \ell_2$$

generate assertion

$$\ell_1'\colon\ \texttt{@}\ (\theta^g\{\overrightarrow{p} \leftarrow \overrightarrow{a}\});$$
$$\ell_1\colon\ v := \textsc{g}(\overrightarrow{a});$$
$$\ell_2\colon$$

Says that g's precondition is satisfied.

BinarySearch: Function Preconditions

@pre $|a| \geq 0 \ \wedge \ \mathsf{sorted}(a, 0, |a| - 1)$

@post rv $\leftrightarrow \ (\exists i \in [0, |a| - 1]) \ a[i] = e$

`bool` BinarySearch(`int[]` a, `int` e) {

  ℓ_1: @ $0 \leq 0 \ \wedge \ |a| - 1 < |a| \ \wedge \ \mathsf{sorted}(a, 0, |a| - 1)$;

  `return` BSearch$(a, 0, |a| - 1, e)$;

}

BinarySearch: Function Preconditions

```
@pre 0 ≤ ℓ ∧ u < |a| ∧ sorted(a, ℓ, u)
@post rv ↔ (∃i ∈ [ℓ, u]) a[i] = e
bool BSearch(int[] a, int ℓ, int u, int e) {
  int m;
  if (ℓ > u) return false;
  else {
    m := (ℓ + u) div 2;
    if (a[m] = e) return true;
    else if (a[m] < e) {
      m1: @ 0 ≤ m + 1 ∧ u < |a| ∧ sorted(a, m + 1, u);
      return BSearch(a, m + 1, u, e);
    }
    else {
      m2: @ 0 ≤ ℓ ∧ m − 1 < |a| ∧ sorted(a, ℓ, m − 1);
      return BSearch(a, ℓ, m − 1, e);
    }
  }
}
```

BinarySearch: Augmented Transition System

@pre $|a| \geq 0 \ \wedge \ \mathsf{sorted}(a, 0, |a| - 1)$

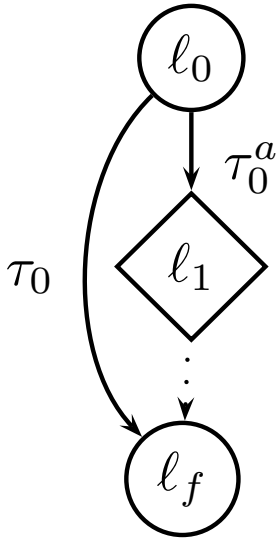
@post rv $\leftrightarrow \ (\exists i \in [0, |a| - 1])\ a[i] = e$

`bool` BinarySearch(`int[]` a, `int` e) {

  ℓ_1: @ $0 \leq 0 \ \wedge \ |a| - 1 < |a| \ \wedge \ \mathsf{sorted}(a, 0, |a| - 1)$;

  `return` BSearch$(a, 0, |a| - 1, e)$;

}



$\rho_{\tau_0^a}$: $\quad a'_0 = a \ \wedge \ e'_0 = e \ \wedge \ |a|'_0 = |a| \ \wedge \ \mathsf{pres}(\ldots)$

ρ_{τ_0} : $\quad \rho_{\tau_0^a} \ \wedge \ \mathsf{rv}' = u \ \wedge \ \mathsf{pres}(\ldots)$
$\wedge \ u \leftrightarrow (\exists i \in [0, |a| - 1])\ a[i] = e$

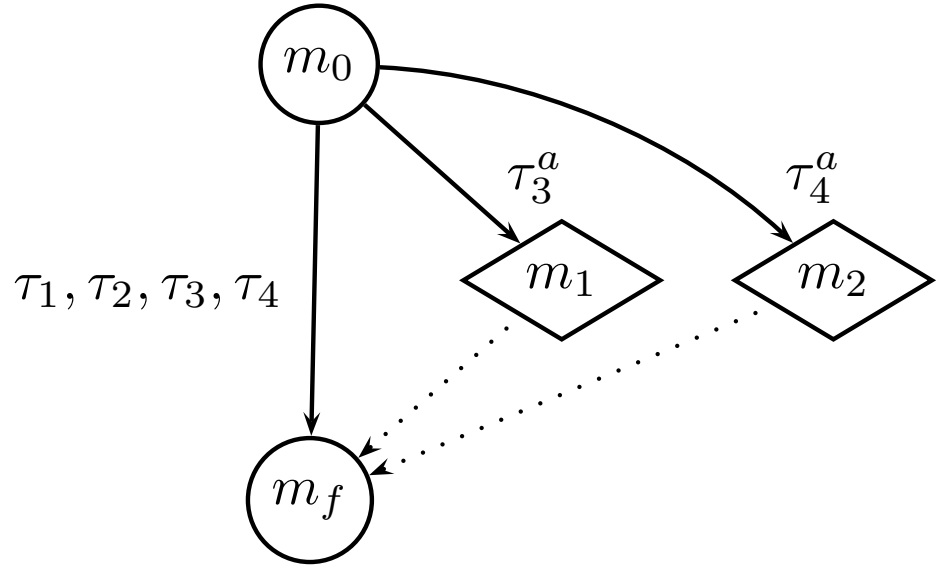
BinarySearch: Augmented Transition System

Let $\varphi_0 : \; a'_0 = a \;\wedge\; \ell'_0 = \ell \;\wedge\; u'_0 = u \;\wedge\; e'_0 = e.$

Intermediate transitions:

$$\rho_{\tau_3^a} : \quad \varphi_0 \;\wedge\; \ell \leq u \;\wedge\; m' = \tfrac{\ell+u}{2} \;\wedge\; a[\tfrac{\ell+u}{2}] \neq e \;\wedge\; a[\tfrac{\ell+u}{2}] < e \;\ldots$$

$$\rho_{\tau_4^a} : \quad \varphi_0 \;\wedge\; \ell \leq u \;\wedge\; m' = \tfrac{\ell+u}{2} \;\wedge\; a[\tfrac{\ell+u}{2}] \neq e \;\wedge\; a[\tfrac{\ell+u}{2}] \geq e \;\ldots$$



BinarySearch: Assertion Map

$$
\begin{aligned}
\mu(\ell_0) &= |a| \geq 0 \ \wedge \ \mathsf{sorted}(a, 0, |a| - 1) \\
\mu(\ell_1) &= 0 \leq 0 \ \wedge \ |a| - 1 < |a| \ \wedge \ \mathsf{sorted}(a, 0, |a| - 1) \\
\mu(\ell_f) &= \mathsf{rv} \ \leftrightarrow \ (\exists i \in [0, |a| - 1]) \ a[i] = e \\
\mu(m_0) &= 0 \leq \ell \ \wedge \ u < |a| \ \wedge \ \mathsf{sorted}(a, \ell, u) \\
\mu(m_1) &= 0 \leq m + 1 \ \wedge \ u < |a| \ \wedge \ \mathsf{sorted}(a, m + 1, u) \\
\mu(m_2) &= 0 \leq \ell \ \wedge \ m - 1 < |a| \ \wedge \ \mathsf{sorted}(a, \ell, m - 1) \\
\mu(m_f) &= \mathsf{rv} \ \leftrightarrow \ (\exists i \in [\ell, u]) \ a[i] = e
\end{aligned}
$$

BinarySearch: Verification Conditions

$$\{\mu(\ell_0)\}\rho_{\tau_0^a}\{\mu(\ell_1)\} \qquad \{\mu(m_0)\}\rho_{\tau_1}\{\mu(m_f)\}$$

$$\{\mu(\ell_0)\}\rho_{\tau_0}\{\mu(\ell_f)\} \qquad \{\mu(m_0)\}\rho_{\tau_2}\{\mu(m_f)\}$$

$$\{\mu(m_0)\}\rho_{\tau_3^a}\{\mu(m_f)\}$$

$$\{\mu(m_0)\}\rho_{\tau_3}\{\mu(m_f)\}$$

$$\{\mu(m_0)\}\rho_{\tau_4^a}\{\mu(m_f)\}$$

$$\{\mu(m_0)\}\rho_{\tau_4}\{\mu(m_f)\}$$

$\Rightarrow$ BinarySearch returns **true** iff array a contains element e

BinarySearch: $\{\mu(m_0)\}\rho_{\tau_2}\{\mu(m_f)\}$

$$(\forall a, \ell, u, e, a_0, \ell_0, u_0, e_0, m, a', \ell', u', e', a'_0, \ell'_0, u'_0, e'_0, m')$$
$$\left[\begin{array}{l} |a| \geq 0 \ \wedge \ \mathsf{sorted}(a, 0, |a| - 1) \\ \wedge\ \varphi_0 \ \wedge \ \ell \leq u \ \wedge \ m' = \frac{\ell+u}{2} \\ \wedge\ a[\frac{\ell+u}{2}] = e \ \wedge \ \mathsf{rv}' = \mathbf{true} \ \wedge \ \mathsf{pres}(\ldots) \\ \quad \rightarrow \ (\mathsf{rv}' \ \leftrightarrow \ (\exists i \in [0, |a|'_0 - 1])\ a'_0[i] = e'_0) \end{array}\right]$$

More simply:

$$\rho_{\tau_2} : \ G(\mathcal{V}) \ \wedge \ U(\mathcal{V} \cup \mathcal{V}')$$
$$(\forall \mathcal{V})[\mu(m_0) \ \wedge \ G \ \rightarrow \ \mu(m_f)(\mathsf{s}(U))]$$
$$(\forall a, \ell, u, e)$$
$$\left[\begin{array}{l} |a| \geq 0 \ \wedge \ \mathsf{sorted}(a, 0, |a| - 1) \wedge \ \ell \leq u \ \wedge \ a[\frac{\ell+u}{2}] = e \\ \quad \rightarrow \ (\exists i \in [0, |a| - 1])\ a[i] = e \end{array}\right]$$

Intermediate Assertions: Runtime Assertions

- **Array Bounds**: If statement contains array access $a[i]$, add assertion

$$\texttt{@}\ (0 \leq i \leq |a| - 1)$$

- **Divide-by-Zero**: If statement contains
  - real division e_1/e_2
  - integer division e_1 `div` e_2
  - integer mod $e_1 \% e_2$

  add assertion

$$\texttt{@}\ (e_2 \neq 0)$$

Optional: to prove lack of runtime exceptions.

BubbleSort: Intermediate Assertions

$$\texttt{for @}\begin{bmatrix}1 \le i < |a| \ \wedge\ 0 \le j \le i \ \wedge\ |a| = |a|_0 \\ \wedge\ \mathsf{partitioned}(a, 0, i, i+1, |a|-1) \\ \wedge\ \mathsf{partitioned}(a, 0, j-1, j, j) \ \wedge\ \mathsf{sorted}(a, i, |a|-1)\end{bmatrix}$$

$(j := 0;\ j < i;\ j := j + 1)\ \{$

  @ $0 \le j < |a| \ \wedge\ 0 \le j+1 < |a|$;

  `if` $(a[j] > a[j+1])\ \{$

    @ $0 \le j < |a|$;

    $t := a[j]$;

    @ $0 \le j < |a| \ \wedge\ 0 \le j+1 < |a|$;

    $a[j] := a[j+1]$;

    @ $0 \le j+1 < |a|$;

    $a[j+1] := t$;

  }

}

Compiler optionally generates assertions.

BinarySearch: Intermediate Assertions

@pre $|a| \geq 0 \ \wedge \ \mathsf{sorted}(a, 0, |a| - 1)$

@post rv $\leftrightarrow \ (\exists i \in [0, |a| - 1]) \ a[i] = e$

`bool` BinarySearch(`int[]` a, `int` e) {

  ℓ_1: @ $0 \leq 0 \ \wedge \ |a| - 1 < |a| \ \wedge \ \mathsf{sorted}(a, 0, |a| - 1)$;

  `return` BSearch$(a, 0, |a| - 1, e)$;

}

```
@pre 0 ≤ ℓ ∧ u < |a| ∧ sorted(a, ℓ, u)
@post rv ↔ (∃i ∈ [ℓ, u]) a[i] = e
bool BSearch(int[] a, int ℓ, int u, int e) {
  int m;
  if (ℓ > u) return false;
  else {
    m1: @ 2 ≠ 0;
    m := (ℓ + u) div 2;
    m2: @ 0 ≤ m < |a| − 1;
    if (a[m] = e) return true;
    m3: @ 0 ≤ m < |a| − 1;
    else if (a[m] < e) {
      m4: @ 0 ≤ m + 1 ∧ u < |a| ∧ sorted(a, m + 1, u);
      return BSearch(a, m + 1, u, e);
    }
    else {
      m5: @ 0 ≤ ℓ ∧ m − 1 < |a| ∧ sorted(a, ℓ, m − 1);
      return BSearch(a, ℓ, m − 1, e);
    }
  }
}
```

BinarySearch: Augmented Transition System

@pre $|a| \geq 0 \ \wedge \ \mathsf{sorted}(a, 0, |a| - 1)$

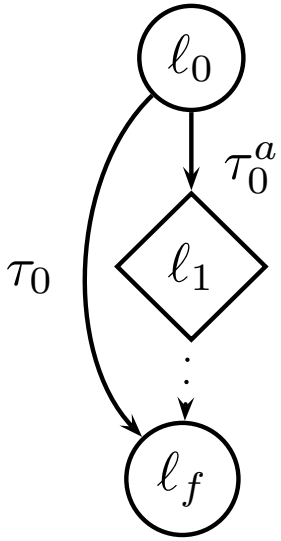
@post rv $\leftrightarrow \ (\exists i \in [0, |a| - 1])\ a[i] = e$

`bool` BinarySearch(`int[]` a, `int` e) {

  ℓ_1: @ $0 \leq 0 \ \wedge \ |a| - 1 < |a| \ \wedge \ \mathsf{sorted}(a, 0, |a| - 1)$;

  `return` BSearch$(a, 0, |a| - 1, e)$;

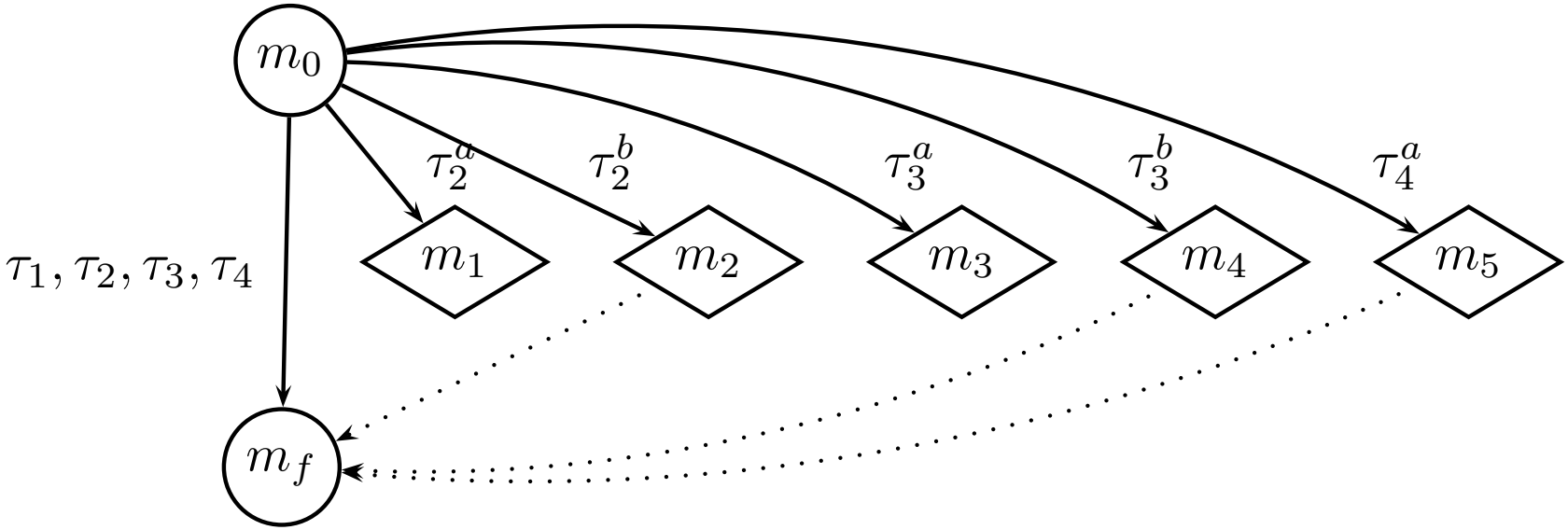
}



$\rho_{\tau_0^a}$: $\quad a'_0 = a \ \wedge \ e'_0 = e \ \wedge \ |a|'_0 = |a| \ \wedge \ \mathsf{pres}(\ldots)$

ρ_{τ_0} : $\quad \rho_{\tau_0^a} \ \wedge \ \mathsf{rv}' = u \ \wedge \ u \leftrightarrow (\exists i \in [0, |a| - 1])\ a[i] = e$

BinarySearch: Augmented Transition System



BinarySearch: Augmented Transition System

$$\rho_{\tau_1}: \quad a_0' = a \ \wedge \ \cdots \ \wedge \ \ell > u \ \wedge \ \mathsf{rv}' = \mathbf{false} \ \wedge \ \mathsf{pres}(\ldots)$$

$$\rho_{\tau_2^a}: \quad a_0' = a \ \wedge \ \cdots \ \wedge \ \ell \leq u \ \wedge \ \mathsf{pres}(\ldots)$$

$$\rho_{\tau_2^b}: \quad \rho_{\tau_2^a} \ \wedge \ m' = \tfrac{\ell+u}{2}$$

$$\rho_{\tau_2}: \quad \rho_{\tau_2^b} \ \wedge \ a[\tfrac{\ell+u}{2}] = e \ \wedge \ \mathsf{rv}' = \mathbf{true}$$

$$\rho_{\tau_3^a}: \quad \rho_{\tau_2^b} \ \wedge \ a[\tfrac{\ell+u}{2}] \neq e$$

$$\rho_{\tau_3^b}: \quad \rho_{\tau_3^a} \ \wedge \ a[\tfrac{\ell+u}{2}] < e$$

$$\rho_{\tau_3}: \quad \rho_{\tau_3^b} \ \wedge \ \mathsf{rv}' = u \ \wedge \ u \leftrightarrow (\exists i \in [\tfrac{\ell+u}{2}+1, u]) \ a[i] = e$$

$$\rho_{\tau_4^a}: \quad \rho_{\tau_3^a} \ \wedge \ a[\tfrac{\ell+u}{2}] \geq e$$

$$\rho_{\tau_4}: \quad \rho_{\tau_4^a} \ \wedge \ \mathsf{rv}' = u \ \wedge \ u \leftrightarrow (\exists i \in [\ell, \tfrac{\ell+u}{2}-1]) \ a[i] = e$$

BinarySearch: Assertion Map

$$
\begin{aligned}
\mu(\ell_0) &= |a| \geq 0 \ \wedge\ \mathsf{sorted}(a, 0, |a| - 1) \\
\mu(\ell_1) &= 0 \leq 0 \ \wedge\ |a| - 1 < |a| \ \wedge\ \mathsf{sorted}(a, 0, |a| - 1) \\
\mu(\ell_f) &= \mathsf{rv} \ \leftrightarrow\ (\exists i \in [0, |a| - 1])\ a[i] = e \\
\mu(m_0) &= 0 \leq \ell \ \wedge\ u < |a| \ \wedge\ \mathsf{sorted}(a, \ell, u) \\
\mu(m_1) &= 2 \neq 0 \\
\mu(m_2) &= 0 \leq m < |a| - 1 \\
\mu(m_3) &= 0 \leq m < |a| - 1 \\
\mu(m_4) &= 0 \leq m + 1 \ \wedge\ u < |a| \ \wedge\ \mathsf{sorted}(a, m + 1, u) \\
\mu(m_5) &= 0 \leq \ell \ \wedge\ m - 1 < |a| \ \wedge\ \mathsf{sorted}(a, \ell, m - 1) \\
\mu(m_f) &= \mathsf{rv} \ \leftrightarrow\ (\exists i \in [\ell, u])\ a[i] = e
\end{aligned}
$$

BinarySearch: Verification Conditions

$$\{\mu(\ell_0)\}\rho_{\tau_0^a}\{\mu(\ell_1)\} \qquad \{\mu(m_0)\}\rho_{\tau_1}\{\mu(m_f)\}$$

$$\{\mu(\ell_0)\}\rho_{\tau_0}\{\mu(\ell_f)\} \qquad \{\mu(m_0)\}\rho_{\tau_2^a}\{\mu(m_1)\}$$

$$\{\mu(m_0)\}\rho_{\tau_2^b}\{\mu(m_2)\}$$

$$\{\mu(m_0)\}\rho_{\tau_2}\{\mu(m_f)\}$$

$$\{\mu(m_0)\}\rho_{\tau_3^a}\{\mu(m_3)\}$$

$$\{\mu(m_0)\}\rho_{\tau_3^b}\{\mu(m_4)\}$$

$$\{\mu(m_0)\}\rho_{\tau_3}\{\mu(m_f)\}$$

$$\{\mu(m_0)\}\rho_{\tau_4^a}\{\mu(m_5)\}$$

$$\{\mu(m_0)\}\rho_{\tau_4}\{\mu(m_f)\}$$

$\Rightarrow$ BinarySearch is also correct WRT runtime assertions

Intermediate Assertions: Annotations

Finally, natural to assert expressions as close to context as possible.

Outline

1. Transition Systems
2. Partial Correctness

⇒ 3. Total Correctness

BubbleSort: Motivation

```
int[] BubbleSort(int[] a) {
  int i, j, t;
  for (i := |a| − 1; i > 0; i := i − 1) {
    for (j := 0; j < i; j := j + 1) {
      if (a[j] > a[j + 1]) {
        t := a[j];
        a[j] := a[j + 1];
        a[j + 1] := t;
      }
    }
  }
  return a;
}
```

Does BubbleSort always finish?

BinarySearch: Motivation

```
bool BinarySearch(int[] a, int e) {
  return BSearch(a, 0, |a| − 1, e);
}

bool BSearch(int[] a, int ℓ, int u, int e) {
  int m;
  if (ℓ > u) return false;
  else {
    m := (ℓ + u) div 2;
    if (a[m] = e) return true;
    else if (a[m] < e) return BSearch(a, m + 1, u, e);
    else return BSearch(a, ℓ, m − 1, e);
  }
}
```

Does BinarySearch always finish?

Motivation

Given: Transition system $T : \langle \mathcal{V}, \Theta : \langle \ell_0, \theta \rangle, \Omega : \langle \ell_f, \omega \rangle, \mathcal{T} \rangle$

Goal:

Also prove that if

initially θ holds

then

ℓ_f is reached.

Strategy:

- Prove that each loop is associated with a **well-founded relation**.
- Usually requires **supporting** invariant map.

BubbleSort: Termination

@pre $|a| \geq 0$

@post **true**

```
int[] BubbleSort(int[] a) {
  int i, j, t;
  ℓ1: for @ (i + 1 ≥ 0) ↓ (i + 1, i, 1) (i := |a| − 1; i > 0; i := i − 1)
    ℓ2: for @ (i − j ≥ 0) ↓ (i + 1, i − j, 0) (j := 0; j < i; j := j + 1)
      if (a[j] > a[j + 1]) {
        t := a[j]; a[j] := a[j + 1]; a[j + 1] := t;
      }
  return a;
}
```

BubbleSort: Termination

Prove that

$$\mu(\ell_1) = i + 1 \geq 0$$
$$\mu(\ell_2) = i - j \geq 0$$

is an invariant map.

Prove that

- range of each function at a $\downarrow$ is bounded from below;
- function values decrease from one $\downarrow$ to the next.

BinarySearch: Termination

```
@pre |a| ≥ 0
@post true
↓ (1, 0)
bool BinarySearch(int[] a, int e) {
  return BSearch(a, 0, |a| − 1, e);
}

@pre u − ℓ + 1 ≥ 0
@post true
↓ (0, u − ℓ + 1)
bool BSearch(int[] a, int ℓ, int u, int e) {
  int m;
  if (ℓ > u) return false;
  else {
    m := (ℓ + u) div 2;
    if (a[m] = e) return true;
    else if (a[m] < e) return BSearch(a, m + 1, u, e);
    else return BSearch(a, ℓ, m − 1, e);
  }
}
```

BinarySearch: Termination (Generated Annotations)

@pre $|a| \geq 0$

@post **true**

$\downarrow (1, 0)$

```
bool BinarySearch(int[] a, int e) {
```

ℓ_1: @ $(|a| - 1) - 0 + 1 \geq 0$

$\downarrow (0, (|a| - 1) - 0 + 1)$;

```
  return BSearch(a, 0, |a| − 1, e);
}
```

```
@pre u − ℓ + 1 ≥ 0
@post true
↓ (0, u − ℓ + 1)
bool BSearch(int[] a, int ℓ, int u, int e) {
  int m;
  if (ℓ > u) return false;
  else {
    m := (ℓ + u) div 2;
    if (a[m] = e) return true;
    else if (a[m] < e) {
      m1: @ u − (m + 1) + 1 ≥ 0
            ↓ (0, u − (m + 1) + 1);
      return BSearch(a, m + 1, u, e);
    }
    else {
      m2: @ (m − 1) − ℓ + 1 ≥ 0
            ↓ (0, (m − 1) − ℓ + 1);
      return BSearch(a, ℓ, m − 1, e);
    }
  }
}
```

BinarySearch: Termination

Prove that

$$\begin{aligned} \mu(\ell_0) &= |a| \geq 0 \\ \mu(\ell_1) &= (|a| - 1) - 0 + 1 \geq 0 \\ \mu(\ell_f) &= \textbf{true} \end{aligned}$$

$$\begin{aligned} \mu(m_0) &= u - \ell + 1 \geq 0 \\ \mu(m_1) &= u - (m+1) + 1 \geq 0 \\ \mu(m_2) &= (m-1) - \ell + 1 \geq 0 \\ \mu(m_f) &= \textbf{true} \end{aligned}$$

is an invariant map.

Prove that

- range of each function at a $\downarrow$ is bounded from below;
- function values decrease from one $\downarrow$ to the next.

Well-founded Relation

$(D, \prec)$: $\prec$ is **well-founded** if there is no infinite sequence

$$d_1, d_2, d_3, \ldots \quad \text{where } d_i \in D$$

such that

$$(\forall i)\ d_i \succ d_{i+1}$$

$(d_2 \prec d_1 \iff d_1 \succ d_2)$

Examples:

- $(\mathbb{Z}^+, <)$
  $1023 > 988 > 987 > \cdots > 16 > 0$
- $(\mathbb{L}, \prec)$ for lists $\mathbb{L}$ $\quad \ell_1 \prec \ell_2 \iff |\ell_1| < |\ell_2|$
  $[a; c; e] \succ [c; e] \succ []$

Lexicographic Well-founded Relation

Given well-founded relations over domains

$$(D_1, \prec_1), (D_2, \prec_2), \ldots, (D_k, \prec_k)$$

define **lexicographic well-founded relation** $\prec$ over

$$D = D_1 \times D_2 \times \cdots \times D_k$$

For $d = \langle d_1, d_2, \ldots, d_k \rangle,\ e = \langle e_1, e_2, \ldots, e_k \rangle \in D$

$$d \prec e \quad \Longleftrightarrow \quad (\exists i)\,[d_i \prec_i e_i \ \wedge\ (\forall j < i)\ d_j = e_j]$$

$$\begin{array}{ccccccc} \langle & d_1, & \ldots, & d_i, & \ldots, d_k & \rangle \\ & \| & & \| & \prec_i & \\ \langle & e_1, & \ldots, & e_i, & \ldots, e_k & \rangle \end{array}$$

Example: Lexicographic Well-founded Relation

$(\mathbb{Z}^+, <) \times (\mathbb{Z}^+, <)$:

New domain (pairs of integers)

$$D = \mathbb{Z}^+ \times \mathbb{Z}^+$$

and well-founded relation

$$(m_1, m_2) \prec (n_1, n_2)$$

$$\Updownarrow$$

$$m_1 < n_1 \ \vee \ (m_1 = n_1 \ \wedge \ m_2 < n_2)$$

$(7, 13) \succ (7, 11) \succ (6, 101) \succ (3, 1000) \succ (0, 0)$

Function Map

Given $T : \langle \mathcal{V}, \Theta, \Omega, \mathcal{T} \rangle$,

$$\nu : \mathsf{locs}(T) \to \mathcal{E}(\mathcal{V})$$

maps T's locations to functions over T's variables.

Each function

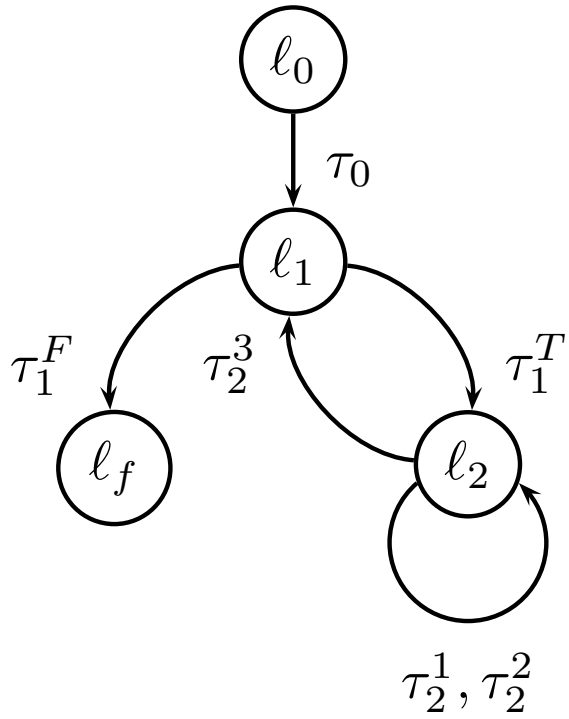
$$\nu(\ell) : \mathcal{V} \to \mathbb{Z}^n$$

maps a program state to a tuple of integers.

BubbleSort: Function Map

$\nu(\ell_1) = (i+1,\ i,\ 1)$

$\nu(\ell_2) = (i+1,\ i-j,\ 0)$



BinarySearch: Function Map

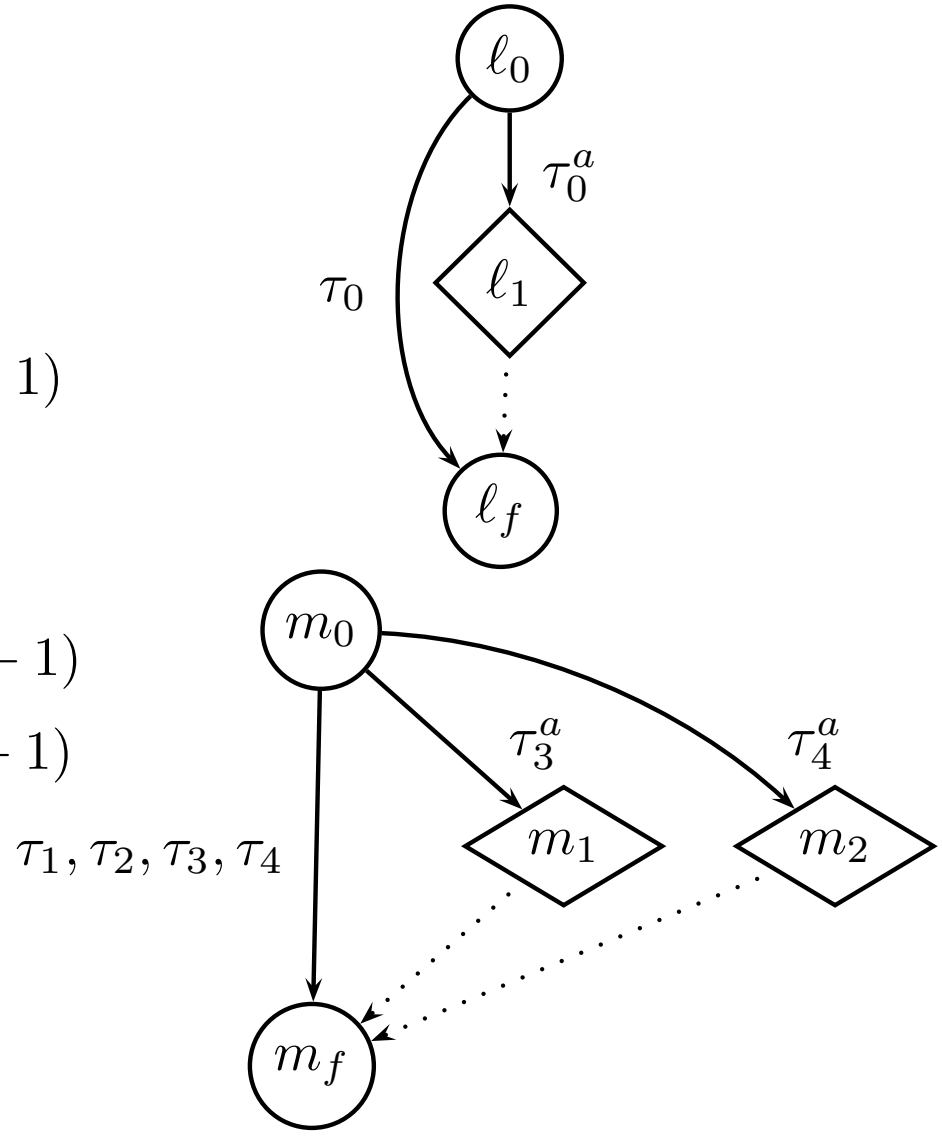
$\nu(\ell_0) = (1, 0)$

$\nu(\ell_1) = (0, (|a| - 1) - 0 + 1)$

$\nu(m_0) = (0, u - \ell + 1)$

$\nu(m_1) = (0, u - (m + 1) + 1)$

$\nu(m_2) = (0, (m - 1) - \ell + 1)$



Ranking Function Map

Given function map ν for $T : \langle \mathcal{V}, \Theta, \Omega, \mathcal{T} \rangle$.

ν is **ranking function map** if

there is an invariant map μ s.t.
for each $\tau : \langle \ell, \ell', \rho_\tau \rangle \in \mathcal{T}$

Well-founded

$$\mu(\ell) \;\Rightarrow\; \nu(\ell) \succ 0$$

Decreasing

$$\{\mu(\ell)\}\rho_\tau\{\nu(\ell')' \prec \nu(\ell)\}$$

Conditions are called **verification conditions** (VCs).

Verification Conditions

Recall:

$$\{\varphi\}\rho_\tau\{\psi\} \quad \Rightarrow \quad (\forall \mathcal{V} \cup \mathcal{V}')[\varphi \wedge \rho_\tau \ \rightarrow \ \psi']$$

With substitutions:
When ρ_τ can be expressed as $G(\mathcal{V}) \wedge U(\mathcal{V} \cup \mathcal{V}')$,

$$\{\mu(\ell)\}\rho_\tau\{\nu(\ell')' \prec \nu(\ell)\}$$

$$\Downarrow$$

$$(\forall \mathcal{V})[\varphi \wedge G \ \rightarrow \ \nu(\ell')(\mathsf{s}(U)) \prec \nu(\ell)]$$

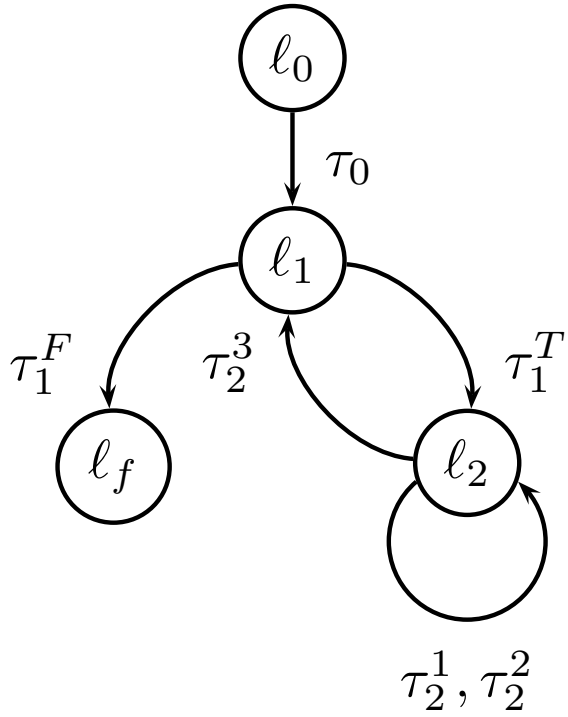
BubbleSort: Verification Conditions

Well-founded

$$\mu(\ell_1) \;\Rightarrow\; \nu(\ell_1) \succ 0$$
$$\mu(\ell_2) \;\Rightarrow\; \nu(\ell_2) \succ 0$$

Decreasing

$$\{\mu(\ell_1)\}\rho_{\tau_1^T}\{\nu(\ell_2)' \prec \nu(\ell_1)\}$$
$$\{\mu(\ell_2)\}\rho_{\tau_2^1}\{\nu(\ell_2)' \prec \nu(\ell_2)\}$$
$$\{\mu(\ell_2)\}\rho_{\tau_2^1}\{\nu(\ell_2)' \prec \nu(\ell_2)\}$$
$$\{\mu(\ell_2)\}\rho_{\tau_2^3}\{\nu(\ell_1)' \prec \nu(\ell_2)\}$$



(plus VCs for invariant map)

BubbleSort: Verification Conditions

$$\rho_{\tau_1^T}: \quad i > 0 \wedge j' = 0 \wedge \mathsf{pres}(\ldots)$$

$$\rho_{\tau_2^1}: \quad \begin{array}{l} j < i \wedge a[j] > a[j+1] \wedge t' = a[j] \\ \wedge\ a' = a[j : a[j+1]][j+1 : a[j]] \wedge j' = j+1 \wedge \mathsf{pres}(\ldots) \end{array}$$

$$\rho_{\tau_2^2}: \quad j < i \wedge a[j] \leq a[j+1] \wedge j' = j+1 \wedge \mathsf{pres}(\ldots)$$

$$\rho_{\tau_2^3}: \quad j \geq i \wedge i' = i-1 \wedge \mathsf{pres}(\ldots)$$

BubbleSort: Verification Conditions

$$\mu(\ell_1) \;\Rightarrow\; \nu(\ell_1) \succ 0$$

$$(\forall \mathcal{V})[i+1 \geq 0 \;\rightarrow\; (i+1, i, 1) \succ 0]$$

$$(\forall \mathcal{V})[i+1 \geq 0 \;\rightarrow\; i+1 \geq 0 \;\wedge\; i \geq 0 \;\wedge\; 1 \geq 0]$$

$$\mu(\ell_2) \;\Rightarrow\; \nu(\ell_2) \succ 0$$

$$(\forall \mathcal{V})[i-j \geq 0 \;\rightarrow\; (i+1, i-j, 0) \succ 0]$$

$$(\forall \mathcal{V})[i-j \geq 0 \;\rightarrow\; i+1 \geq 0 \;\wedge\; i-j \geq 0 \;\wedge\; 0 \geq 0]$$

BubbleSort: Verification Conditions

$$\{\mu(\ell_1)\}\rho_{\tau_1^T}\{\nu(\ell_2)' \prec \nu(\ell_1)\}$$

$$(\forall \mathcal{V})[i+1 \geq 0 \ \wedge \ i > 0 \ \rightarrow \ (i+1, i-0, 0) \prec (i+1, i, 1)]$$

$$(\forall \mathcal{V}) \left[\begin{array}{l} i+1 \geq 0 \ \wedge \ i > 0 \ \rightarrow \\ \left[\begin{array}{l} i+1 < i+1 \ \vee \ (i+1 = i+1 \ \wedge \ i-0 < i) \\ \vee \ (i+1 = i+1 \ \wedge \ i = i-j \ \wedge \ 0 < 1) \end{array} \right] \end{array} \right]$$

$$\{\mu(\ell_2)\}\rho_{\tau_2^1}\{\nu(\ell_2)' \prec \nu(\ell_2)\}$$

$$(\forall \mathcal{V}) \left[\begin{array}{l} i-j \geq 0 \ \wedge \ j < i \ \wedge \ a[j] > a[j+1] \\ \rightarrow (i+1, i-(j+1), 0) \prec (i+1, i-j, 0) \end{array} \right]$$

BubbleSort: Verification Conditions

$$\{\mu(\ell_2)\}\rho_{\tau_2^2}\{\nu(\ell_2)' \prec \nu(\ell_2)\}$$

$$(\forall \mathcal{V}) \begin{bmatrix} i-j \geq 0 \ \wedge \ j < i \ \wedge \ a[j] \leq a[j+1] \\ \rightarrow \ (i+1, i-(j+1), 0) \prec (i+1, i-j, 0) \end{bmatrix}$$

$$\{\mu(\ell_2)\}\rho_{\tau_2^3}\{\nu(\ell_1)' \prec \nu(\ell_2)\}$$

$$(\forall \mathcal{V})[i-j \geq 0 \ \wedge j \geq i \ \rightarrow ((i-1)+1, i-1, 1) \prec (i+1, i-j, 0)]$$

$\Rightarrow$ BubbleSort always terminates

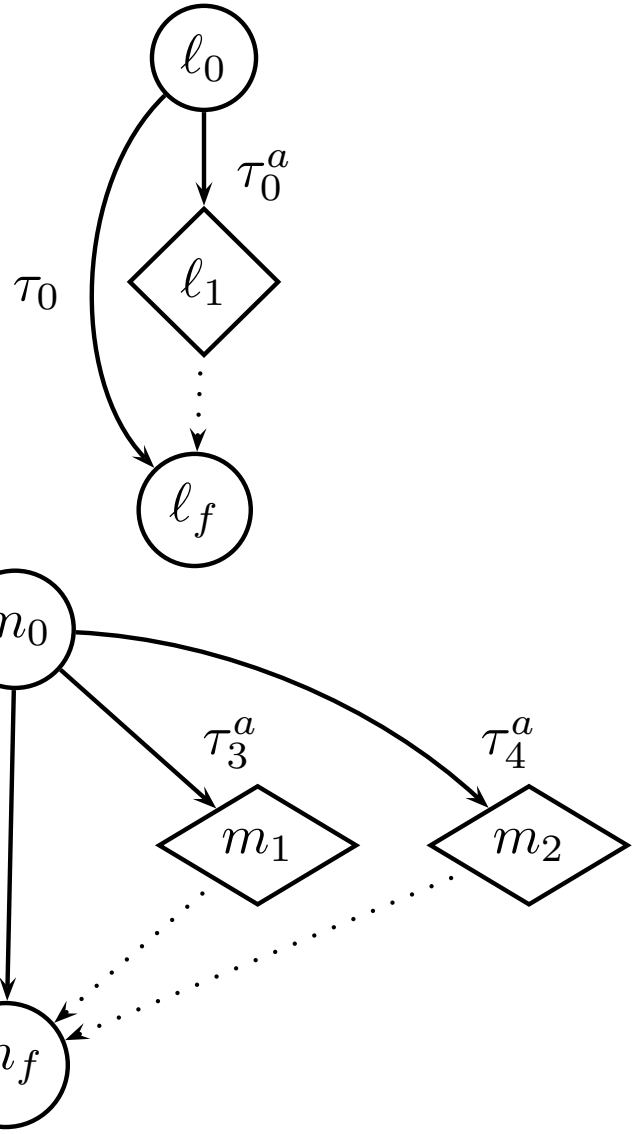
BinarySearch: Verification Conditions

Well-founded

$$\mu(\ell_0) \;\Rightarrow\; \nu(\ell_0) \succ 0$$
$$\mu(\ell_1) \;\Rightarrow\; \nu(\ell_1) \succ 0$$
$$\mu(m_0) \;\Rightarrow\; \nu(m_0) \succ 0$$
$$\mu(m_1) \;\Rightarrow\; \nu(m_1) \succ 0$$
$$\mu(m_2) \;\Rightarrow\; \nu(m_2) \succ 0$$

Decreasing

$$\{\mu(\ell_0)\}\rho_{\tau_0^a}\{\nu(\ell_1)' \prec \nu(\ell_0)\}$$
$$\{\mu(m_0)\}\rho_{\tau_3^a}\{\nu(m_1)' \prec \nu(m_0)\}$$
$$\{\mu(m_0)\}\rho_{\tau_4^a}\{\nu(m_2)' \prec \nu(m_0)\}$$



BinarySearch: Verification Conditions

$$\rho_{\tau_0^a}: \quad a'_0 = a \ \wedge \ e'_0 = e \ \wedge \ |a|'_0 = |a| \ \wedge \ \mathsf{pres}(\ldots)$$

$$\rho_{\tau_3^a}: \quad \varphi_0 \ \wedge \ \ell \leq u \ \wedge \ m' = \tfrac{\ell+u}{2} \ \wedge \ a[\tfrac{\ell+u}{2}] \neq e \ \wedge \ a[\tfrac{\ell+u}{2}] < e \ \wedge \ \mathsf{pres}(\ldots)$$

$$\rho_{\tau_4^a}: \quad \varphi_0 \ \wedge \ \ell \leq u \ \wedge \ m' = \tfrac{\ell+u}{2} \ \wedge \ a[\tfrac{\ell+u}{2}] \neq e \ \wedge \ a[\tfrac{\ell+u}{2}] \geq e \ \wedge \ \mathsf{pres}(\ldots)$$

BinarySearch: Verification Conditions

$$\mu(\ell_0) \;\Rightarrow\; \nu(\ell_0) \succ 0$$
$$(\forall \mathcal{V})[|a| \geq 0 \;\rightarrow\; (1, 0) \succ 0]$$

$$\mu(\ell_1) \;\Rightarrow\; \nu(\ell_1) \succ 0$$
$$(\forall \mathcal{V})[(|a| - 1) - 0 + 1 \geq 0 \;\rightarrow\; (0, (|a| - 1) - 0 + 1) \succ 0]$$

BinarySearch: Verification Conditions

$$\mu(m_0) \;\Rightarrow\; \nu(m_0) \succ 0$$
$$(\forall \mathcal{V})[u - \ell + 1 \geq 0 \;\rightarrow\; (0, u - \ell + 1) \succ 0]$$

$$\mu(m_1) \;\Rightarrow\; \nu(m_1) \succ 0$$
$$(\forall \mathcal{V})[u - (m+1) + 1 \geq 0 \;\rightarrow\; (0, u - (m+1) + 1) \succ 0]$$

$$\mu(m_2) \;\Rightarrow\; \nu(m_2) \succ 0$$
$$(\forall \mathcal{V})[(m-1) - \ell + 1 \geq 0 \;\rightarrow\; (0, (m-1) - \ell + 1) \succ 0]$$

BinarySearch: Verification Conditions

$$\{\mu(\ell_0)\}\rho_{\tau_0^a}\{\nu(\ell_1)' \prec \nu(\ell_0)\}$$
$$(\forall \mathcal{V})[|a| \geq 0 \ \rightarrow \ (0, (|a| - 1) - 0 + 1) \prec (1, 0)]$$

$$\{\mu(m_0)\}\rho_{\tau_3^a}\{\nu(m_1)' \prec \nu(m_0)\}$$
$$(\forall \mathcal{V}) \left[\begin{array}{c} u - \ell + 1 \geq 0 \ \wedge \ \ell \leq u \ \wedge \ a[\frac{\ell+u}{2}] \neq e \ \wedge \ a[\frac{\ell+u}{2}] < e \\ \rightarrow \ (0, u - (\frac{\ell+u}{2} + 1) + 1) \prec (0, u - \ell + 1) \end{array}\right]$$

$$\{\mu(m_0)\}\rho_{\tau_4^a}\{\nu(m_2)' \prec \nu(m_0)\}$$
$$(\forall \mathcal{V}) \left[\begin{array}{c} u - \ell + 1 \geq 0 \ \wedge \ \ell \leq u \ \wedge \ a[\frac{\ell+u}{2}] \neq e \ \wedge \ a[\frac{\ell+u}{2}] \geq e \\ \rightarrow \ (0, (\frac{\ell+u}{2} - 1) - \ell + 1) \prec (0, u - \ell + 1) \end{array}\right]$$

$\Rightarrow$ BinarySearch always terminates