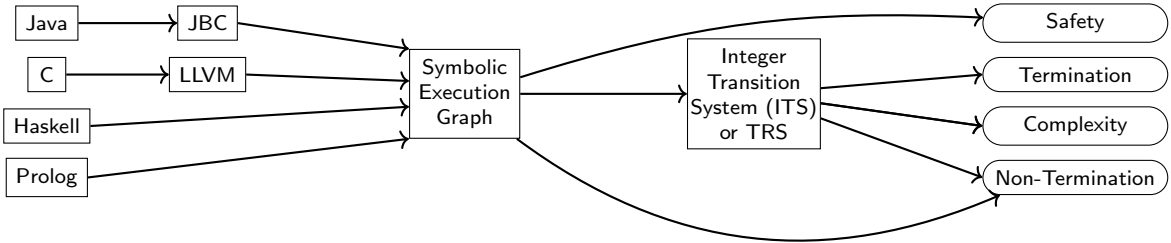
Proving Termination of C Programs with Lists

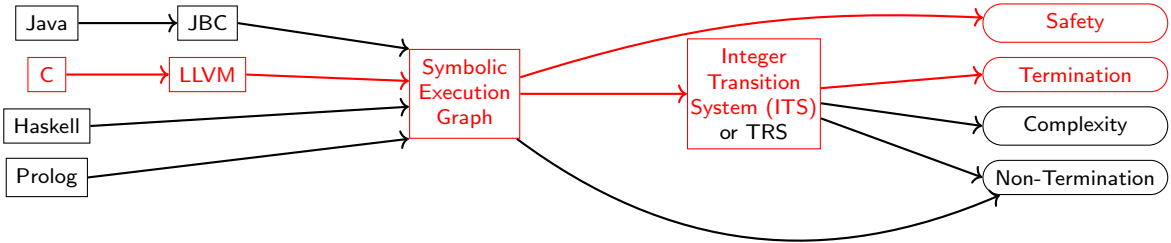
Jera Hensel Jürgen Giesl

LuFG Informatik 2, RWTH Aachen University, Germany

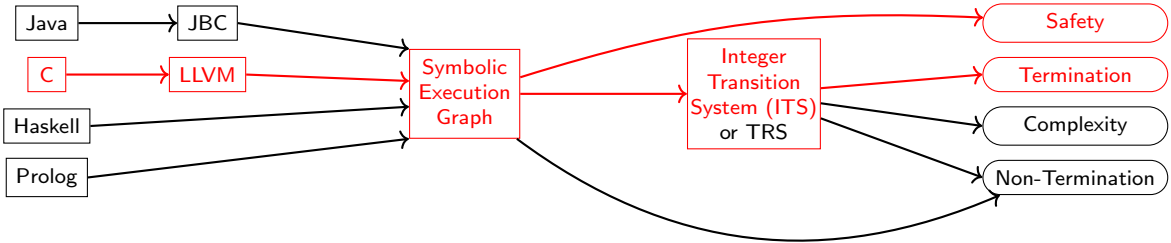
Termination and Complexity Analysis in AProVE



Termination and Complexity Analysis in AProVE

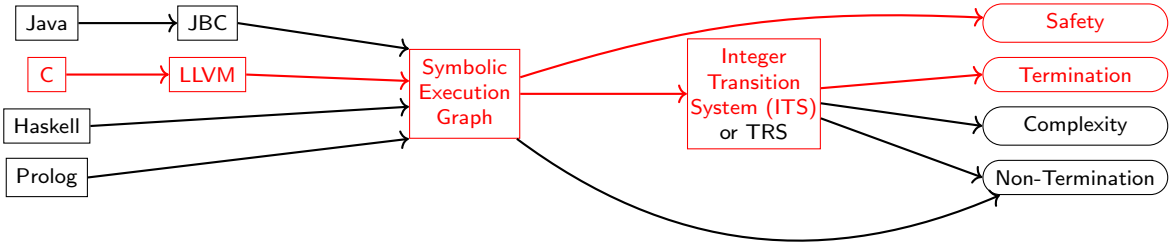


Termination and Complexity Analysis in AProVE



- Handling of C programs with explicit pointer arithmetic

Termination and Complexity Analysis in AProVE



- Handling of C programs with explicit pointer arithmetic

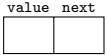
- **Drawback:**
  C termination tools have very restricted support for dynamic data structures

C Programs with Lists

```
struct list {
 unsigned int value;
 struct list* next;  }
```

C Programs with Lists

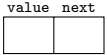
```
struct list {
 unsigned int value;
 struct list* next;  }
```



C Programs with Lists

```
struct list {
 unsigned int value;
 struct list* next;  }
```

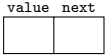
```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
```



C Programs with Lists

```
struct list {
 unsigned int value;
 struct list* next;  }
```

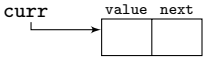
```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {

 }
 ...
```



C Programs with Lists

```
struct list {
 unsigned int value;
 struct list* next;  }
```

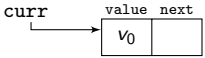
```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
  curr = malloc(sizeof(struct list));



                                      }
 ...
```



C Programs with Lists

```
struct list {
 unsigned int value;
 struct list* next;  }
```

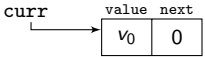
```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();

                                    }
 ...
```



C Programs with Lists

```
struct list {
 unsigned int value;
 struct list* next;  }
```

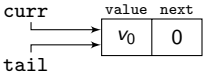
```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
                                      }
 ...
```



C Programs with Lists

```
struct list {
 unsigned int value;
 struct list* next;  }
```

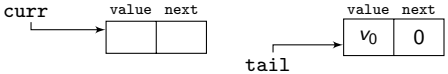
```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
  tail = curr;                          }
 ...
```



C Programs with Lists

```
struct list {
 unsigned int value;
 struct list* next;  }
```

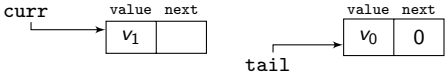
```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
  tail = curr;                          }
 ...
```



C Programs with Lists

```
struct list {
 unsigned int value;
 struct list* next;  }
```

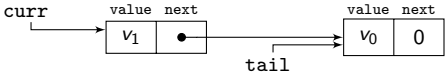
```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
  tail = curr;                          }
 ...
```



C Programs with Lists

```
struct list {
 unsigned int value;
 struct list* next;  }
```

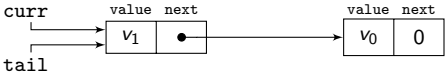
```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
  tail = curr;                          }
 ...
```



C Programs with Lists

```
struct list {
 unsigned int value;
 struct list* next;  }
```

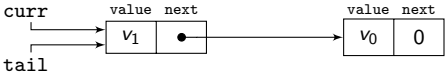
```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
  tail = curr;                          }
 ...
```



C Programs with Lists

```
struct list {
 unsigned int value;
 struct list* next;  }
```

```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
  tail = curr;                          }
 ...
```



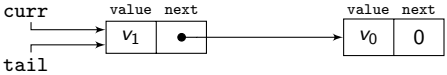
⇒ list of `n` non-deterministic numbers

C Programs with Lists

```
struct list {
 unsigned int value;
 struct list* next;  }
```

```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
  tail = curr;                          }
 ...
```

New technique to infer
list invariants (LI)



$\Rightarrow$ list of `n` non-deterministic numbers

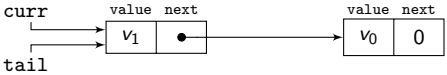
C Programs with Lists

```
struct list {
 unsigned int value;
 struct list* next;  }
```

New technique to infer
list invariants (*LI*)

- *LI* abstracts from detailed information about lists

```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
  tail = curr;                          }
 ...
```



$\Rightarrow$ list of `n` non-deterministic numbers

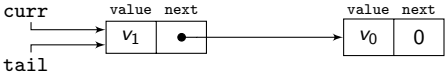
C Programs with Lists

```
struct list {
 unsigned int value;
 struct list* next;  }
```

```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
  tail = curr;                          }
 ...
```

New technique to infer
list invariants (*LI*)

- *LI* abstracts from detailed information about lists
- *LI* needed to obtain finite complete symbolic execution graphs



$\Rightarrow$ list of `n` non-deterministic numbers

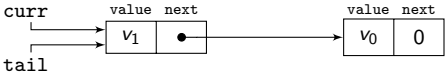
C Programs with Lists

```
struct list {
 unsigned int value;
 struct list* next;  }
```

```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
  tail = curr;                          }
 ...
```

New technique to infer
list invariants (*LI*)

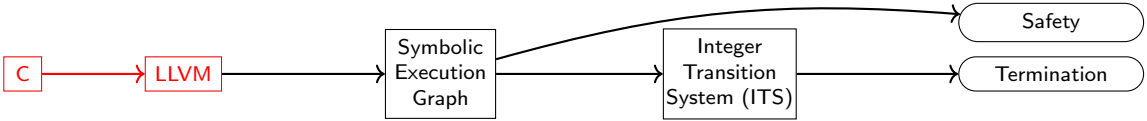
- *LI* abstracts from detailed information about lists
- *LI* needed to obtain finite complete symbolic execution graphs
- *LI* expresses crucial properties for memory safety & termination



$\Rightarrow$ list of `n` non-deterministic numbers

From C to LLVM

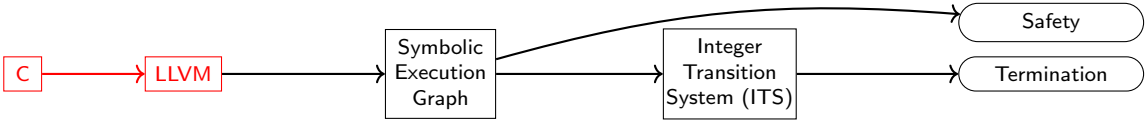
```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
  tail = curr;                        }
 ...
```



From C to LLVM

```
list = type { i32, list* }

define i32 @main() { ...
cmp: 0: k = load i32, i32* k_ad
     1: kltn = icmp ult i32 k, n
     2: br i1 kltn, label body, label init
body:0: mem = call i8* @malloc(i64 16)
     ...
     9: kinc = add i32 k, 1
     10:store i32 kinc, i32* k_ad
     11:br label cmp                ... }
```

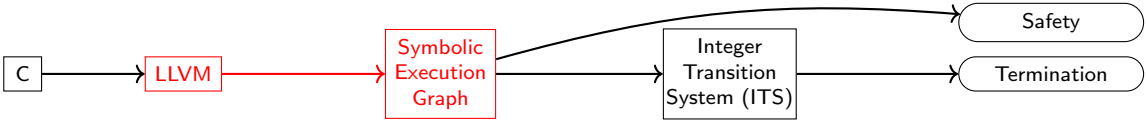
```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
  tail = curr;                        }
 ...
```



Abstract States

```
list = type { i32, list* }

define i32 @main() { ...
cmp: 0: k = load i32, i32* k_ad
     1: kltn = icmp ult i32 k, n
     2: br i1 kltn, label body, label init
body:0: mem = call i8* @malloc(i64 16)
     ...
     9: kinc = add i32 k, 1
     10:store i32 kinc, i32* k_ad
     11:br label cmp              ... }
```

```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
  tail = curr;                        }
 ...
```



Abstract States

```
list = type { i32, list* }

define i32 @main() { ...
cmp: 0: k = load i32, i32* k_ad
     1: kltn = icmp ult i32 k, n
     2: br i1 kltn, label body, label init
body:0: mem = call i8* @malloc(i64 16)
     ...
     9: kinc = add i32 k, 1
     10:store i32 kinc, i32* k_ad
     11:br label cmp              ... }
```

Abstract state a:

Abstract States

```
list = type { i32, list* }

define i32 @main() { ...
cmp: 0: k = load i32, i32* k_ad
     1: kltn = icmp ult i32 k, n
     2: br i1 kltn, label body, label init
body:0: mem = call i8* @malloc(i64 16)
     ...
     9: kinc = add i32 k, 1
     10:store i32 kinc, i32* k_ad
     11:br label cmp              ... }
```

$(\mathtt{cmp}, 0) \Longleftarrow pos$

Abstract state *a*:

pos: program position (block, next instruction)

Abstract States

```
list = type { i32, list* }

define i32 @main() { ...
cmp: 0: k = load i32, i32* k_ad
     1: kltn = icmp ult i32 k, n
     2: br i1 kltn, label body, label init
body:0: mem = call i8* @malloc(i64 16)
     ...
     9: kinc = add i32 k, 1
     10:store i32 kinc, i32* k_ad
     11:br label cmp              ... }
```

$(\mathtt{cmp}, 0) \Longleftarrow pos$

$\{\mathtt{curr} = x_{\mathtt{mem}}, \mathtt{kinc} = x_{\mathtt{kinc}}, \ldots\} \Longleftarrow PV$

Abstract state a:

pos: program position (block, next instruction)

PV: program variables → symbolic variables

Abstract States

```
list = type { i32, list* }

define i32 @main() { ...
cmp: 0: k = load i32, i32* k_ad
     1: kltn = icmp ult i32 k, n
     2: br i1 kltn, label body, label init
body:0: mem = call i8* @malloc(i64 16)
     ...
     9: kinc = add i32 k, 1
     10:store i32 kinc, i32* k_ad
     11:br label cmp              ... }
```

$(\mathtt{cmp}, 0) \Longleftarrow pos$

$\{\mathtt{curr} = x_{\mathtt{mem}}, \mathtt{kinc} = x_{\mathtt{kinc}}, \ldots\} \Longleftarrow PV$

$\{[\![x_{\mathtt{k\_ad}}, x^{end}_{\mathtt{k\_ad}}]\!], \ldots\} \Longleftarrow AL$

Abstract state a:

pos: program position (block, next instruction)

PV: program variables → symbolic variables

AL: allocation list $[\![v_1, v_2]\!]$

Abstract States

```
list = type { i32, list* }

define i32 @main() { ...
cmp: 0: k = load i32, i32* k_ad
     1: kltn = icmp ult i32 k, n
     2: br i1 kltn, label body, label init
body:0: mem = call i8* @malloc(i64 16)
     ...
     9: kinc = add i32 k, 1
     10:store i32 kinc, i32* k_ad
     11:br label cmp              ... }
```

$(\mathtt{cmp}, 0)$	$\Longleftarrow$ *pos*
$\{\mathtt{curr} = x_{\mathtt{mem}}, \mathtt{kinc} = x_{\mathtt{kinc}}, \ldots\}$	$\Longleftarrow$ *PV*
$\{[\![x_{\mathtt{k\_ad}}, x^{end}_{\mathtt{k\_ad}}]\!], \ldots\}$	$\Longleftarrow$ *AL*
$\{x^{end}_{\mathtt{k\_ad}} = x_{\mathtt{k\_ad}} + 3, \ldots\}$	$\Longleftarrow$ *KB*

Abstract state *a*:

- *pos*: program position (block, next instruction)
- *PV*: program variables → symbolic variables
- *AL*: allocation list $[\![v_1, v_2]\!]$
- *KB*: FO-knowledge base over symbolic variables

Abstract States

```
list = type { i32, list* }

define i32 @main() { ...
cmp: 0: k = load i32, i32* k_ad
     1: kltn = icmp ult i32 k, n
     2: br i1 kltn, label body, label init
body:0: mem = call i8* @malloc(i64 16)
     ...
     9: kinc = add i32 k, 1
     10:store i32 kinc, i32* k_ad
     11:br label cmp              ... }
```

$$
\begin{array}{ll}
(\mathtt{cmp}, 0) & \Longleftarrow \mathit{pos} \\
\{\mathtt{curr} = x_{\mathtt{mem}}, \mathtt{kinc} = x_{\mathtt{kinc}}, \ldots\} & \Longleftarrow \mathit{PV} \\
\{[\![x_{\mathtt{k\_ad}}, x^{\mathit{end}}_{\mathtt{k\_ad}}]\!], \ldots\} & \Longleftarrow \mathit{AL} \\
\{x^{\mathit{end}}_{\mathtt{k\_ad}} = x_{\mathtt{k\_ad}} + 3, \ldots\} & \Longleftarrow \mathit{KB} \\
\{x_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} x_{\mathtt{kinc}}, \ldots\} & \Longleftarrow \mathit{PT}
\end{array}
$$

Abstract state a:

- pos: program position (block, next instruction)
- PV: program variables → symbolic variables
- AL: allocation list $[\![v_1, v_2]\!]$
- KB: FO-knowledge base over symbolic variables
- PT: points-to atoms $v_1 \hookrightarrow_{\mathtt{type}} v_2$

Abstract States

```
list = type { i32, list* }

define i32 @main() { ...
cmp: 0: k = load i32, i32* k_ad
     1: kltn = icmp ult i32 k, n
     2: br i1 kltn, label body, label init
body:0: mem = call i8* @malloc(i64 16)
     ...
     9: kinc = add i32 k, 1
     10:store i32 kinc, i32* k_ad
     11:br label cmp              ... }
```

$$
\begin{array}{ll}
(\mathtt{cmp}, 0) & \Longleftarrow \mathit{pos} \\
\{\mathtt{curr} = x_{\mathtt{mem}}, \mathtt{kinc} = x_{\mathtt{kinc}}, \ldots\} & \Longleftarrow \mathit{PV} \\
\{[\![x_{\mathtt{k\_ad}}, x^{end}_{\mathtt{k\_ad}}]\!], \ldots\} & \Longleftarrow \mathit{AL} \\
\{x^{end}_{\mathtt{k\_ad}} = x_{\mathtt{k\_ad}} + 3, \ldots\} & \Longleftarrow \mathit{KB} \\
\{x_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} x_{\mathtt{kinc}}, \ldots\} & \Longleftarrow \mathit{PT}
\end{array}
$$

Abstract state a: ERR or

- pos: program position (block, next instruction)
- PV: program variables → symbolic variables
- AL: allocation list $[\![v_1, v_2]\!]$
- KB: FO-knowledge base over symbolic variables
- PT: points-to atoms $v_1 \hookrightarrow_{\mathtt{type}} v_2$

Abstract States

```
list = type { i32, list* }

define i32 @main() { ...
cmp: 0: k = load i32, i32* k_ad
     1: kltn = icmp ult i32 k, n
     2: br i1 kltn, label body, label init
body:0: mem = call i8* @malloc(i64 16)
     ...
     9: kinc = add i32 k, 1
     10:store i32 kinc, i32* k_ad
     11:br label cmp                ... }
```

$(\mathtt{cmp}, 0) \Longleftarrow pos$

$\{\mathtt{curr} = x_{\mathtt{mem}}, \mathtt{kinc} = x_{\mathtt{kinc}}, \ldots\} \Longleftarrow PV$

$\{[\![x_{\mathtt{k\_ad}}, x^{end}_{\mathtt{k\_ad}}]\!], \ldots\} \Longleftarrow AL$

$\{x^{end}_{\mathtt{k\_ad}} = x_{\mathtt{k\_ad}} + 3, \ldots\} \Longleftarrow KB$

$\{x_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} x_{\mathtt{kinc}}, \ldots\} \Longleftarrow PT$

Abstract state a: ERR or

pos: program position (block, next instruction)

PV: program variables → symbolic variables

AL: allocation list $[\![v_1, v_2]\!]$

KB: FO-knowledge base over symbolic variables

PT: points-to atoms $v_1 \hookrightarrow_{\mathtt{type}} v_2$

LI: $v_{\mathtt{ad}} \overset{v_\ell}{\hookrightarrow}_{\mathtt{ty}} [(off_i : \mathtt{ty}_i : v_i..\hat{v}_i)]_{i=1}^{m}$

Abstract States

```
list = type { i32, list* }

define i32 @main() { ...
cmp: 0: k = load i32, i32* k_ad
     1: kltn = icmp ult i32 k, n
     2: br i1 kltn, label body, label init
body:0: mem = call i8* @malloc(i64 16)
     ...
     9: kinc = add i32 k, 1
     10:store i32 kinc, i32* k_ad
     11:br label cmp                ... }
```

$(\mathtt{cmp}, 0)$	$\Longleftarrow$ *pos*
$\{\mathtt{curr} = x_{\mathtt{mem}}, \mathtt{kinc} = x_{\mathtt{kinc}}, \ldots\}$	$\Longleftarrow$ *PV*
$\{[\![x_{\mathtt{k\_ad}}, x^{end}_{\mathtt{k\_ad}}]\!], \ldots\}$	$\Longleftarrow$ *AL*
$\{x^{end}_{\mathtt{k\_ad}} = x_{\mathtt{k\_ad}} + 3, \ldots\}$	$\Longleftarrow$ *KB*
$\{x_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} x_{\mathtt{kinc}}, \ldots\}$	$\Longleftarrow$ *PT*
$\{x_{\mathtt{mem}} \overset{x_\ell}{\hookrightarrow}_{\mathtt{list}} [(0 : \mathtt{i32} : x_{\mathtt{nd}}..\hat{x}_{\mathtt{nd}}), (8 : \mathtt{list*} : x_{\mathtt{next}}..0)]\}$	

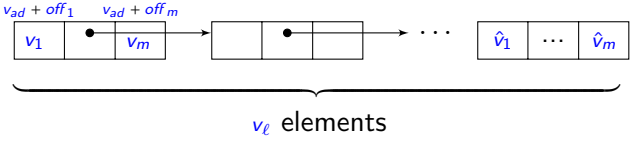
Abstract state a: ERR or

- pos: program position (block, next instruction)
- PV: program variables → symbolic variables
- AL: allocation list $[\![v_1, v_2]\!]$
- KB: FO-knowledge base over symbolic variables
- PT: points-to atoms $v_1 \hookrightarrow_{\mathtt{type}} v_2$
- LI: $v_{\mathtt{ad}} \overset{v_\ell}{\hookrightarrow}_{\mathtt{ty}} [(off_i : \mathtt{ty}_i : v_i..\hat{v}_i)]_{i=1}^{m}$

Abstract States

```
list = type { i32, list* }

define i32 @main() { ...
cmp: 0: k = load i32, i32* k_ad
     1: kltn = icmp ult i32 k, n
     2: br i1 kltn, label body, label init
body:0: mem = call i8* @malloc(i64 16)
     ...
     9: kinc = add i32 k, 1
     10:store i32 kinc, i32* k_ad
     11:br label cmp                ... }
```

$$
\begin{array}{ll}
(\mathtt{cmp}, 0) & \Longleftarrow \mathit{pos} \\
\{\mathtt{curr} = x_{\mathtt{mem}}, \mathtt{kinc} = x_{\mathtt{kinc}}, \ldots\} & \Longleftarrow \mathit{PV} \\
\{[\![x_{\mathtt{k\_ad}}, x^{end}_{\mathtt{k\_ad}}]\!], \ldots\} & \Longleftarrow \mathit{AL} \\
\{x^{end}_{\mathtt{k\_ad}} = x_{\mathtt{k\_ad}} + 3, \ldots\} & \Longleftarrow \mathit{KB} \\
\{x_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} x_{\mathtt{kinc}}, \ldots\} & \Longleftarrow \mathit{PT} \\
\{x_{\mathtt{mem}} \overset{x_\ell}{\hookrightarrow}_{\mathtt{list}} [(0 : \mathtt{i32} : x_{\mathtt{nd}}..\hat{x}_{\mathtt{nd}}), (8 : \mathtt{list*} : x_{\mathtt{next}}..0)]\} &
\end{array}
$$



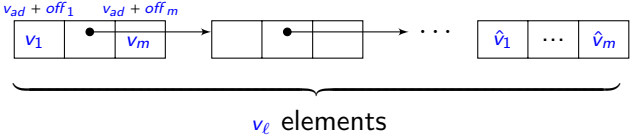
$$\mathit{LI}: \quad v_{\mathtt{ad}} \overset{v_\ell}{\hookrightarrow}_{\mathtt{ty}} [(\mathit{off}_i : \mathtt{ty}_i : v_i..\hat{v}_i)]_{i=1}^{m}$$

Abstract States

```
list = type { i32, list* }

define i32 @main() { ...
cmp: 0: k = load i32, i32* k_ad
     1: kltn = icmp ult i32 k, n
     2: br i1 kltn, label body, label init
body:0: mem = call i8* @malloc(i64 16)
     ...
     9: kinc = add i32 k, 1
     10:store i32 kinc, i32* k_ad
     11:br label cmp                ... }
```

$$
\begin{array}{ll}
(\mathtt{cmp}, 0) & \Longleftarrow \mathit{pos} \\
\{\mathtt{curr} = x_{\mathtt{mem}}, \mathtt{kinc} = x_{\mathtt{kinc}}, \ldots\} & \Longleftarrow \mathit{PV} \\
\{[\![x_{\mathtt{k\_ad}}, x^{end}_{\mathtt{k\_ad}}]\!], \ldots\} & \Longleftarrow \mathit{AL} \\
\{x^{end}_{\mathtt{k\_ad}} = x_{\mathtt{k\_ad}} + 3, \ldots\} & \Longleftarrow \mathit{KB} \\
\{x_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} x_{\mathtt{kinc}}, \ldots\} & \Longleftarrow \mathit{PT} \\
\{x_{\mathtt{mem}} \overset{x_\ell}{\hookrightarrow}_{\mathtt{list}} [(0 : \mathtt{i32} : x_{\mathtt{nd}}..\hat{x}_{\mathtt{nd}}), (8 : \mathtt{list*} : x_{\mathtt{next}}..0)]\} &
\end{array}
$$

- represents `struct ty` with m fields



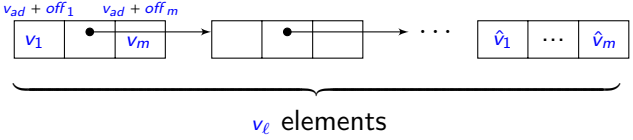
$$\mathit{LI}: \quad v_{\mathtt{ad}} \overset{v_\ell}{\hookrightarrow}_{\mathtt{ty}} [(\mathit{off}_i : \mathtt{ty}_i : v_i..\hat{v}_i)]_{i=1}^{m}$$

Abstract States

```
list = type { i32, list* }

define i32 @main() { ...
cmp: 0: k = load i32, i32* k_ad
     1: kltn = icmp ult i32 k, n
     2: br i1 kltn, label body, label init
body:0: mem = call i8* @malloc(i64 16)
     ...
     9: kinc = add i32 k, 1
     10:store i32 kinc, i32* k_ad
     11:br label cmp                ... }
```

$$
\begin{array}{ll}
(\mathtt{cmp}, 0) & \Longleftarrow \mathit{pos} \\
\{\mathtt{curr} = x_{\mathtt{mem}}, \mathtt{kinc} = x_{\mathtt{kinc}}, \ldots\} & \Longleftarrow \mathit{PV} \\
\{[\![x_{\mathtt{k\_ad}}, x^{end}_{\mathtt{k\_ad}}]\!], \ldots\} & \Longleftarrow \mathit{AL} \\
\{x^{end}_{\mathtt{k\_ad}} = x_{\mathtt{k\_ad}} + 3, \ldots\} & \Longleftarrow \mathit{KB} \\
\{x_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} x_{\mathtt{kinc}}, \ldots\} & \Longleftarrow \mathit{PT} \\
\{x_{\mathtt{mem}} \overset{x_\ell}{\hookrightarrow}_{\mathtt{list}} [(0 : \mathtt{i32} : x_{\mathtt{nd}}..\hat{x}_{\mathtt{nd}}), (8 : \mathtt{list*} : x_{\mathtt{next}}..0)]\} &
\end{array}
$$

- represents `struct ty` with m fields
- corresponds to list of length v_ℓ



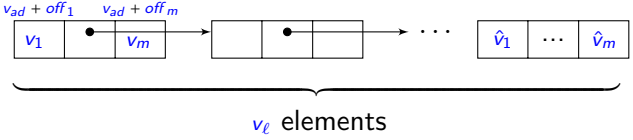
$$\mathit{LI}: \quad v_{\mathtt{ad}} \overset{v_\ell}{\hookrightarrow}_{\mathtt{ty}} [(\mathit{off}_i : \mathtt{ty}_i : v_i..\hat{v}_i)]_{i=1}^{m}$$

Abstract States

```
list = type { i32, list* }

define i32 @main() { ...
cmp: 0: k = load i32, i32* k_ad
     1: kltn = icmp ult i32 k, n
     2: br i1 kltn, label body, label init
body:0: mem = call i8* @malloc(i64 16)
     ...
     9: kinc = add i32 k, 1
     10:store i32 kinc, i32* k_ad
     11:br label cmp                ... }
```

$(\mathtt{cmp}, 0)$	$\Longleftarrow$ *pos*
$\{\mathtt{curr} = x_{\mathtt{mem}}, \mathtt{kinc} = x_{\mathtt{kinc}}, \ldots\}$	$\Longleftarrow$ *PV*
$\{[\![x_{\mathtt{k\_ad}}, x^{end}_{\mathtt{k\_ad}}]\!], \ldots\}$	$\Longleftarrow$ *AL*
$\{x^{end}_{\mathtt{k\_ad}} = x_{\mathtt{k\_ad}} + 3, \ldots\}$	$\Longleftarrow$ *KB*
$\{x_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} x_{\mathtt{kinc}}, \ldots\}$	$\Longleftarrow$ *PT*
$\{x_{\mathtt{mem}} \overset{x_\ell}{\hookrightarrow}_{\mathtt{list}} [(0 : \mathtt{i32} : x_{\mathtt{nd}}..\hat{x}_{\mathtt{nd}}), (8 : \mathtt{list*} : x_{\mathtt{next}}..0)]\}$	

- represents `struct ty` with m fields
- corresponds to list of length v_ℓ
- v_{ad} points to first list element



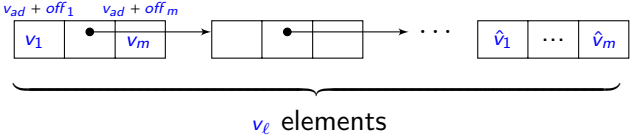
LI: $v_{\mathtt{ad}} \overset{v_\ell}{\hookrightarrow}_{\mathtt{ty}} [(off_i : \mathtt{ty}_i : v_i..\hat{v}_i)]_{i=1}^{m}$

Abstract States

```
list = type { i32, list* }

define i32 @main() { ...
cmp: 0: k = load i32, i32* k_ad
     1: kltn = icmp ult i32 k, n
     2: br i1 kltn, label body, label init
body:0: mem = call i8* @malloc(i64 16)
     ...
     9: kinc = add i32 k, 1
     10:store i32 kinc, i32* k_ad
     11:br label cmp                ... }
```

$$
\begin{array}{ll}
(\mathtt{cmp}, 0) & \Longleftarrow \mathit{pos} \\
\{\mathtt{curr} = x_{\mathtt{mem}}, \mathtt{kinc} = x_{\mathtt{kinc}}, \ldots\} & \Longleftarrow \mathit{PV} \\
\{[\![x_{\mathtt{k\_ad}}, x^{end}_{\mathtt{k\_ad}}]\!], \ldots\} & \Longleftarrow \mathit{AL} \\
\{x^{end}_{\mathtt{k\_ad}} = x_{\mathtt{k\_ad}} + 3, \ldots\} & \Longleftarrow \mathit{KB} \\
\{x_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} x_{\mathtt{kinc}}, \ldots\} & \Longleftarrow \mathit{PT} \\
\{x_{\mathtt{mem}} \overset{x_\ell}{\hookrightarrow}_{\mathtt{list}} [(0 : \mathtt{i32} : x_{\mathtt{nd}}..\hat{x}_{\mathtt{nd}}), (8 : \mathtt{list*} : x_{\mathtt{next}}..0)]\} &
\end{array}
$$

- represents `struct ty` with m fields
- corresponds to list of length v_ℓ
- v_{ad} points to first list element
- i-th field at address $v_{ad} + off_i$ has type $\mathtt{ty}_i$

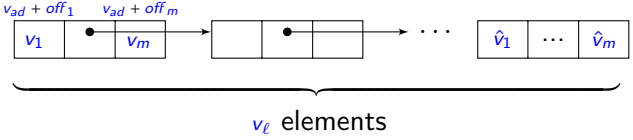


$$\mathit{LI}: \quad v_{\mathtt{ad}} \overset{v_\ell}{\hookrightarrow}_{\mathtt{ty}} [(\mathit{off}_i : \mathtt{ty}_i : v_i..\hat{v}_i)]_{i=1}^{m}$$

Abstract States

```
list = type { i32, list* }

define i32 @main() { ...
cmp: 0: k = load i32, i32* k_ad
     1: kltn = icmp ult i32 k, n
     2: br i1 kltn, label body, label init
body:0: mem = call i8* @malloc(i64 16)
     ...
     9: kinc = add i32 k, 1
     10:store i32 kinc, i32* k_ad
     11:br label cmp                ... }
```

$(\mathtt{cmp}, 0)$	$\Longleftarrow$ *pos*
$\{\mathtt{curr} = x_{\mathtt{mem}}, \mathtt{kinc} = x_{\mathtt{kinc}}, \ldots\}$	$\Longleftarrow$ *PV*
$\{[\![x_{\mathtt{k\_ad}}, x^{end}_{\mathtt{k\_ad}}]\!], \ldots\}$	$\Longleftarrow$ *AL*
$\{x^{end}_{\mathtt{k\_ad}} = x_{\mathtt{k\_ad}} + 3, \ldots\}$	$\Longleftarrow$ *KB*
$\{x_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} x_{\mathtt{kinc}}, \ldots\}$	$\Longleftarrow$ *PT*
$\{x_{\mathtt{mem}} \overset{x_\ell}{\hookrightarrow}_{\mathtt{list}} [(0 : \mathtt{i32} : x_{\mathtt{nd}}..\hat{x}_{\mathtt{nd}}), (8 : \mathtt{list*} : x_{\mathtt{next}}..0)]\}$	



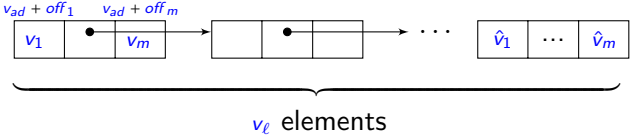
- represents `struct ty` with m fields
- corresponds to list of length v_ℓ
- v_{ad} points to first list element
- i-th field at address $v_{ad} + off_i$ has type $\mathtt{ty}_i$
- one recursive field where $\mathtt{ty}_j = \mathtt{ty*}$

LI: $v_{\mathtt{ad}} \overset{v_\ell}{\hookrightarrow}_{\mathtt{ty}} [(off_i : \mathtt{ty}_i : v_i..\hat{v}_i)]_{i=1}^{m}$

Abstract States

```
list = type { i32, list* }

define i32 @main() { ...
cmp: 0: k = load i32, i32* k_ad
     1: kltn = icmp ult i32 k, n
     2: br i1 kltn, label body, label init
body:0: mem = call i8* @malloc(i64 16)
     ...
     9: kinc = add i32 k, 1
     10:store i32 kinc, i32* k_ad
     11:br label cmp                ... }
```

$(\mathtt{cmp}, 0)$	$\Longleftarrow$	*pos*
$\{\mathtt{curr} = x_{\mathtt{mem}}, \mathtt{kinc} = x_{\mathtt{kinc}}, \ldots\}$	$\Longleftarrow$	*PV*
$\{[\![x_{\mathtt{k\_ad}}, x^{end}_{\mathtt{k\_ad}}]\!], \ldots\}$	$\Longleftarrow$	*AL*
$\{x^{end}_{\mathtt{k\_ad}} = x_{\mathtt{k\_ad}} + 3, \ldots\}$	$\Longleftarrow$	*KB*
$\{x_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} x_{\mathtt{kinc}}, \ldots\}$	$\Longleftarrow$	*PT*
$\{x_{\mathtt{mem}} \overset{x_\ell}{\hookrightarrow}_{\mathtt{list}} [(0 : \mathtt{i32} : x_{\mathtt{nd}}..\hat{x}_{\mathtt{nd}}), (8 : \mathtt{list*} : x_{\mathtt{next}}..0)]\}$		



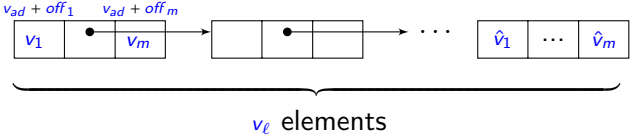
- represents `struct ty` with m fields
- corresponds to list of length v_ℓ
- v_{ad} points to first list element
- i-th field at address $v_{ad} + off_i$ has type $\mathtt{ty}_i$
- one recursive field where $\mathtt{ty}_j = \mathtt{ty*}$
- v_i: value of i-th field in first list element

LI: $v_{\mathtt{ad}} \overset{v_\ell}{\hookrightarrow}_{\mathtt{ty}} [(off_i : \mathtt{ty}_i : v_i..\hat{v}_i)]_{i=1}^{m}$

Abstract States

```
list = type { i32, list* }

define i32 @main() { ...
cmp: 0: k = load i32, i32* k_ad
     1: kltn = icmp ult i32 k, n
     2: br i1 kltn, label body, label init
body:0: mem = call i8* @malloc(i64 16)
     ...
     9: kinc = add i32 k, 1
     10:store i32 kinc, i32* k_ad
     11:br label cmp                ... }
```

$(\mathtt{cmp}, 0)$	$\Longleftarrow$	*pos*
$\{\mathtt{curr} = x_{\mathtt{mem}}, \mathtt{kinc} = x_{\mathtt{kinc}}, \ldots\}$	$\Longleftarrow$	*PV*
$\{[\![x_{\mathtt{k\_ad}}, x^{end}_{\mathtt{k\_ad}}]\!], \ldots\}$	$\Longleftarrow$	*AL*
$\{x^{end}_{\mathtt{k\_ad}} = x_{\mathtt{k\_ad}} + 3, \ldots\}$	$\Longleftarrow$	*KB*
$\{x_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} x_{\mathtt{kinc}}, \ldots\}$	$\Longleftarrow$	*PT*
$\{x_{\mathtt{mem}} \overset{x_\ell}{\hookrightarrow}_{\mathtt{list}} [(0 : \mathtt{i32} : x_{\mathtt{nd}}..\hat{x}_{\mathtt{nd}}), (8 : \mathtt{list*} : x_{\mathtt{next}}..0)]\}$		



- represents `struct ty` with m fields
- corresponds to list of length v_ℓ
- v_{ad} points to first list element
- i-th field at address $v_{ad} + off_i$ has type $\mathtt{ty}_i$
- one recursive field where $\mathtt{ty}_j = \mathtt{ty*}$
- v_i: value of i-th field in first list element
- $\hat{v}_i$: value of i-th field in last list element

LI: $v_{\mathtt{ad}} \overset{v_\ell}{\hookrightarrow}_{\mathtt{ty}} [(off_i : \mathtt{ty}_i : v_i..\hat{v}_i)]_{i=1}^{m}$

Abstract States

```
list = type { i32, list* }

define i32 @main() { ...
cmp: 0: k = load i32, i32* k_ad
     1: kltn = icmp ult i32 k, n
     2: br i1 kltn, label body, label init
body:0: mem = call i8* @malloc(i64 16)
     ...
     9: kinc = add i32 k, 1
     10:store i32 kinc, i32* k_ad
     11:br label cmp                ... }
```

- $\langle a \rangle$: FO formula for abstract state a

$(\mathtt{cmp}, 0)$	$\Longleftarrow$	*pos*
$\{\mathtt{curr} = x_{\mathtt{mem}}, \mathtt{kinc} = x_{\mathtt{kinc}}, \ldots\}$	$\Longleftarrow$	*PV*
$\{[\![x_{\mathtt{k\_ad}}, x^{end}_{\mathtt{k\_ad}}]\!], \ldots\}$	$\Longleftarrow$	*AL*
$\{x^{end}_{\mathtt{k\_ad}} = x_{\mathtt{k\_ad}} + 3, \ldots\}$	$\Longleftarrow$	*KB*
$\{x_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} x_{\mathtt{kinc}}, \ldots\}$	$\Longleftarrow$	*PT*
$\{x_{\mathtt{mem}} \overset{x_\ell}{\hookrightarrow}_{\mathtt{list}} [(0 : \mathtt{i32} : x_{\mathtt{nd}}..\hat{x}_{\mathtt{nd}}), (8 : \mathtt{list*} : x_{\mathtt{next}}..0)]\}$		

- represents `struct ty` with m fields
- corresponds to list of length v_ℓ
- v_{ad} points to first list element
- i-th field at address $v_{ad} + off_i$ has type $\mathtt{ty}_i$
- one recursive field where $\mathtt{ty}_j = \mathtt{ty*}$
- v_i: value of i-th field in first list element
- $\hat{v}_i$: value of i-th field in last list element

Abstract States

```
list = type { i32, list* }

define i32 @main() { ...
cmp: 0: k = load i32, i32* k_ad
     1: kltn = icmp ult i32 k, n
     2: br i1 kltn, label body, label init
body:0: mem = call i8* @malloc(i64 16)
     ...
     9: kinc = add i32 k, 1
     10:store i32 kinc, i32* k_ad
     11:br label cmp                ... }
```

- $\langle a \rangle$: FO formula for abstract state a
  contains KB
  and consequences of AL, PT, and LI

$(\mathtt{cmp}, 0)$	$\Longleftarrow$ *pos*
$\{\mathtt{curr} = x_{\mathtt{mem}}, \mathtt{kinc} = x_{\mathtt{kinc}}, \ldots\}$	$\Longleftarrow$ *PV*
$\{[\![x_{\mathtt{k\_ad}}, x^{end}_{\mathtt{k\_ad}}]\!], \ldots\}$	$\Longleftarrow$ *AL*
$\{x^{end}_{\mathtt{k\_ad}} = x_{\mathtt{k\_ad}} + 3, \ldots\}$	$\Longleftarrow$ *KB*
$\{x_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} x_{\mathtt{kinc}}, \ldots\}$	$\Longleftarrow$ *PT*
$\{x_{\mathtt{mem}} \overset{x_\ell}{\hookrightarrow}_{\mathtt{list}} [(0 : \mathtt{i32} : x_{\mathtt{nd}}..\hat{x}_{\mathtt{nd}}), (8 : \mathtt{list*} : x_{\mathtt{next}}..0)]\}$	

- represents `struct ty` with m fields
- corresponds to list of length v_ℓ
- v_{ad} points to first list element
- i-th field at address $v_{ad} + off_i$ has type $\mathtt{ty}_i$
- one recursive field where $\mathtt{ty}_j = \mathtt{ty*}$
- v_i: value of i-th field in first list element
- $\hat{v}_i$: value of i-th field in last list element

Abstract States

```
list = type { i32, list* }

define i32 @main() { ...
cmp: 0: k = load i32, i32* k_ad
     1: kltn = icmp ult i32 k, n
     2: br i1 kltn, label body, label init
body:0: mem = call i8* @malloc(i64 16)
     ...
     9: kinc = add i32 k, 1
     10:store i32 kinc, i32* k_ad
     11:br label cmp              ... }
```

$$
\begin{array}{ll}
(\mathtt{cmp}, 0) & \Longleftarrow \mathit{pos} \\
\{\mathtt{curr} = x_{\mathtt{mem}}, \mathtt{kinc} = x_{\mathtt{kinc}}, \ldots\} & \Longleftarrow \mathit{PV} \\
\{[\![x_{\mathtt{k\_ad}}, x^{end}_{\mathtt{k\_ad}}]\!], \ldots\} & \Longleftarrow \mathit{AL} \\
\{x^{end}_{\mathtt{k\_ad}} = x_{\mathtt{k\_ad}} + 3, \ldots\} & \Longleftarrow \mathit{KB} \\
\{x_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} x_{\mathtt{kinc}}, \ldots\} & \Longleftarrow \mathit{PT} \\
\{x_{\mathtt{mem}} \overset{x_\ell}{\hookrightarrow}_{\mathtt{list}} [(0 : \mathtt{i32} : x_{\mathtt{nd}}..\hat{x}_{\mathtt{nd}}), (8 : \mathtt{list*} : x_{\mathtt{next}}..0)]\} &
\end{array}
$$

- $\langle a \rangle$: FO formula for abstract state a
  contains KB
  and consequences of AL, PT, and LI
- abstract state a represents set of concrete states (formal definition using separation logic)

- represents `struct ty` with m fields
- corresponds to list of length v_ℓ
- v_{ad} points to first list element
- i-th field at address $v_{ad} + \mathit{off}_i$ has type $\mathtt{ty}_i$
- one recursive field where $\mathtt{ty}_j = \mathtt{ty*}$
- v_i: value of i-th field in first list element
- $\hat{v}_i$: value of i-th field in last list element

Symbolic Execution

```
define i32 @main() { ...
cmp:
  0:k = load i32, i32* k_ad
  1:kltn = icmp ult i32 k, n
  2:br i1 kltn, label body, label init
body:                                  ...}
```

```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
 ...
```

Symbolic Execution

```
define i32 @main() { ...
cmp:
  0:k = load i32, i32* k_ad
  1:kltn = icmp ult i32 k, n
  2:br i1 kltn, label body, label init
body:                                  ...}
```

```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
 ...
```

Symbolic Execution

```
define i32 @main() { ...
cmp:
  0:k = load i32, i32* k_ad
  1:kltn = icmp ult i32 k, n
  2:br i1 kltn, label body, label init
body:                                  ...}
```

```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
 ...
```

Symbolic Execution

```
define i32 @main() { ...
cmp:
  0:k = load i32, i32* k_ad
  1:kltn = icmp ult i32 k, n
  2:br i1 kltn, label body, label init
body:                                  ...}
```

```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
 ...
```

Symbolic Execution

```
define i32 @main() { ...
cmp:
  0:k = load i32, i32* k_ad
  1:kltn = icmp ult i32 k, n
  2:br i1 kltn, label body, label init
body:                                  ...}
```

```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
 ...
```

A

$\{\mathtt{n} = v_{\mathtt{n}}, \mathtt{k\_ad} = v_{\mathtt{k\_ad}}, \ldots\}$ $\Longleftarrow$ *pos*

$\{[\![v_{\mathtt{k\_ad}}, v^{end}_{\mathtt{k\_ad}}]\!]\}$ $\Longleftarrow$ *PV*

$\{v^{end}_{\mathtt{k\_ad}} = v_{\mathtt{k\_ad}} + 3, \ldots\}$ $\Longleftarrow$ *AL*

$\{v_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} 0\}$ $\Longleftarrow$ *KB*

$\Longleftarrow$ *PT*

Symbolic Execution

```
define i32 @main() { ...
cmp:
  0:k = load i32, i32* k_ad
  1:kltn = icmp ult i32 k, n
  2:br i1 kltn, label body, label init
body:                                  ...}
```

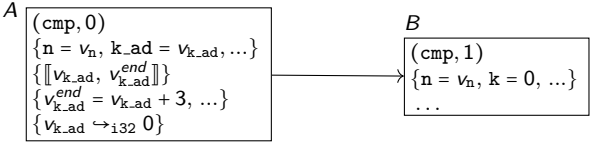
```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
 ...
```

$$
A \quad \boxed{\begin{array}{l}
(\texttt{cmp}, 0) \\
\{\texttt{n} = v_{\texttt{n}},\ \texttt{k\_ad} = v_{\texttt{k\_ad}}, \ldots\} \\
\{[\![v_{\texttt{k\_ad}},\ v^{end}_{\texttt{k\_ad}}]\!]\} \\
\{v^{end}_{\texttt{k\_ad}} = v_{\texttt{k\_ad}} + 3, \ldots\} \\
\{v_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} 0\}
\end{array}}
\begin{array}{l}
\Longleftarrow pos \\
\Longleftarrow PV \\
\Longleftarrow AL \\
\Longleftarrow KB \\
\Longleftarrow PT
\end{array}
$$

Symbolic Execution

```
define i32 @main() { ...
cmp:
  0:k = load i32, i32* k_ad
  1:kltn = icmp ult i32 k, n
  2:br i1 kltn, label body, label init
body:                                  ...}
```

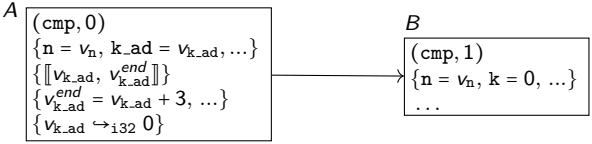
```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
 ...
```



Symbolic Execution

```
define i32 @main() { ...
cmp:
  0:k = load i32, i32* k_ad
  1:kltn = icmp ult i32 k, n
  2:br i1 kltn, label body, label init
body:                                  ...}
```

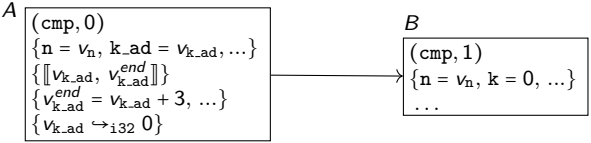
```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
 ...
```



Symbolic Execution

```
define i32 @main() { ...
cmp:
  0:k = load i32, i32* k_ad
  1:kltn = icmp ult i32 k, n
  2:br i1 kltn, label body, label init
body:                                  ...}
```

```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
 ...
```

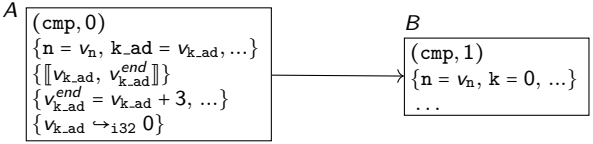


Symbolic execution rule for `x = icmp ult i32` t_1, t_2

Symbolic Execution

```
define i32 @main() { ...
cmp:
  0:k = load i32, i32* k_ad
  1:kltn = icmp ult i32 k, n
  2:br i1 kltn, label body, label init
body:                                  ...}
```

```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
 ...
```



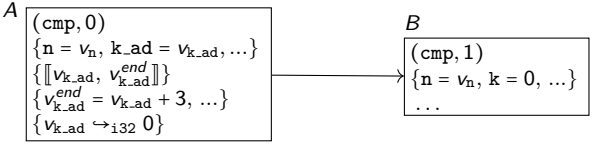
Symbolic execution rule for `x = icmp ult i32` t_1, t_2

- set `x` to `1` if $\models \langle a \rangle \Rightarrow (PV(t_2) > PV(t_1))$

Symbolic Execution

```
define i32 @main() { ...
cmp:
  0:k = load i32, i32* k_ad
  1:kltn = icmp ult i32 k, n
  2:br i1 kltn, label body, label init
body:                                  ...}
```

```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
 ...
```



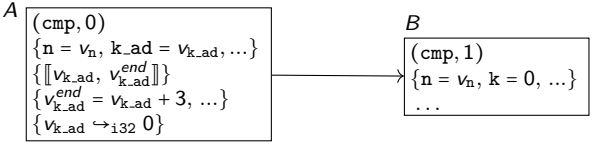
Symbolic execution rule for `x = icmp ult i32` t_1, t_2

- set `x` to 1 if $\models \langle a \rangle \Rightarrow (PV(t_2) > PV(t_1))$
- set `x` to 0 if $\models \langle a \rangle \Rightarrow (PV(t_2) \leq PV(t_1))$

Symbolic Execution

```
define i32 @main() { ...
cmp:
  0:k = load i32, i32* k_ad
  1:kltn = icmp ult i32 k, n
  2:br i1 kltn, label body, label init
body:                                  ...}
```

```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
 ...
```



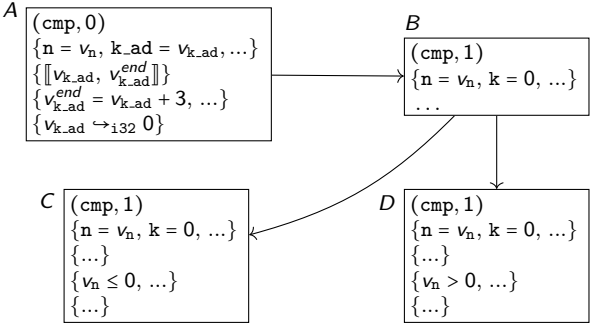
Symbolic execution rule for `x = icmp ult i32` t_1, t_2

- set `x` to 1 if $\models \langle a \rangle \Rightarrow (PV(t_2) > PV(t_1))$
- set `x` to 0 if $\models \langle a \rangle \Rightarrow (PV(t_2) \leq PV(t_1))$
- otherwise: case analysis

Symbolic Execution

```
define i32 @main() { ...
cmp:
  0:k = load i32, i32* k_ad
  1:kltn = icmp ult i32 k, n
  2:br i1 kltn, label body, label init
body:                                  ...}
```

```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
 ...
```



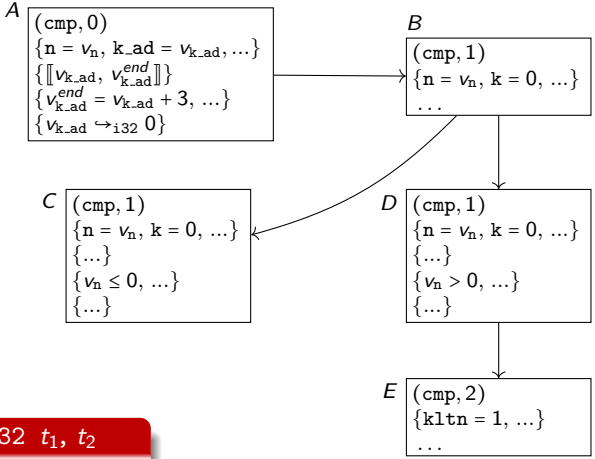
Symbolic execution rule for `x = icmp ult i32` t_1, t_2

- set `x` to 1 if $\models \langle a \rangle \Rightarrow (PV(t_2) > PV(t_1))$
- set `x` to 0 if $\models \langle a \rangle \Rightarrow (PV(t_2) \leq PV(t_1))$
- otherwise: case analysis

Symbolic Execution

```
define i32 @main() { ...
cmp:
  0:k = load i32, i32* k_ad
  1:kltn = icmp ult i32 k, n
  2:br i1 kltn, label body, label init
body:                                  ...}
```

```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
 ...
```



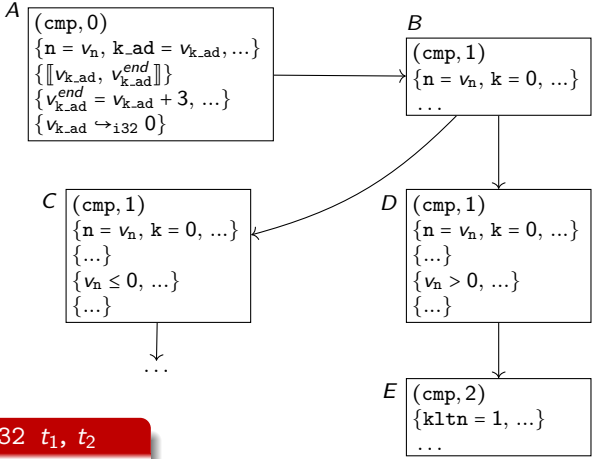
Symbolic execution rule for `x = icmp ult i32` t_1, t_2

- set `x` to 1 if $\models \langle a \rangle \Rightarrow (PV(t_2) > PV(t_1))$
- set `x` to 0 if $\models \langle a \rangle \Rightarrow (PV(t_2) \leq PV(t_1))$
- otherwise: case analysis

Symbolic Execution

```
define i32 @main() { ...
cmp:
  0:k = load i32, i32* k_ad
  1:kltn = icmp ult i32 k, n
  2:br i1 kltn, label body, label init
body:                                  ...}
```

```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
 ...
```



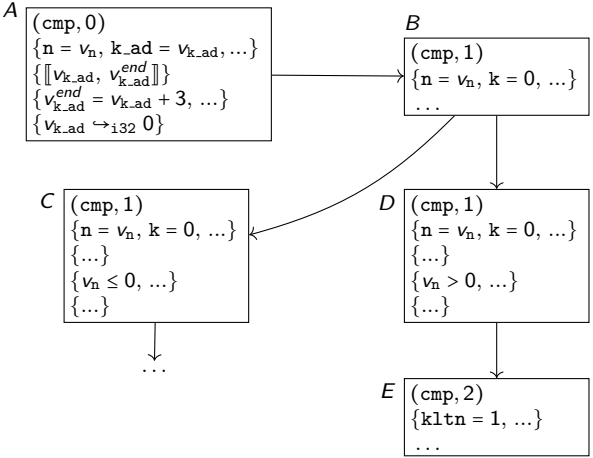
Symbolic execution rule for `x = icmp ult i32` t_1, t_2

- set `x` to 1 if $\models \langle a \rangle \Rightarrow (PV(t_2) > PV(t_1))$
- set `x` to 0 if $\models \langle a \rangle \Rightarrow (PV(t_2) \leq PV(t_1))$
- otherwise: case analysis

Symbolic Execution

```
define i32 @main() { ...
cmp:
  0:k = load i32, i32* k_ad
  1:kltn = icmp ult i32 k, n
  2:br i1 kltn, label body, label init
body:                                  ...}
```

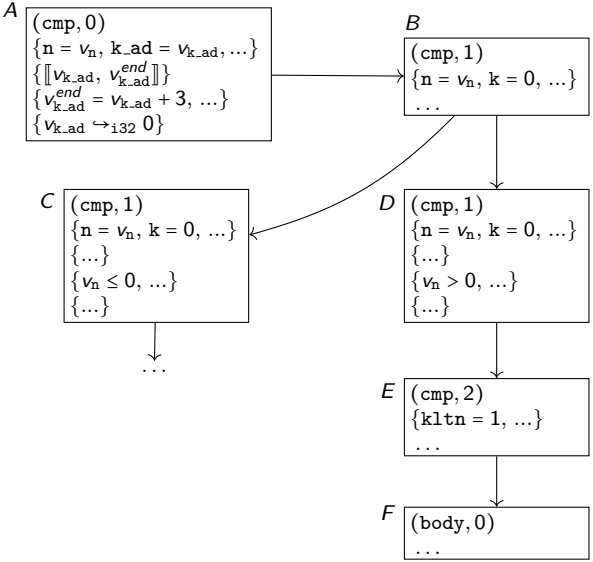
```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
 ...
```



Symbolic Execution

```
define i32 @main() { ...
cmp:
  0:k = load i32, i32* k_ad
  1:kltn = icmp ult i32 k, n
  2:br i1 kltn, label body, label init
body:                                  ...}
```

```
int main() {
 unsigned int n = nondet_uint();
 struct list* tail = NULL;
 struct list* curr;
 for (unsigned int k = 0; k < n; k++) {
 ...
```



Symbolic Execution

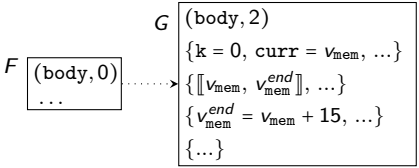
```
cmp:                    check "k < n"
body:0: mem = call i8* @malloc(i64 16)
     1: curr = bitcast i8* mem to list*
```

F $\boxed{(\text{body}, 0) \quad \ldots}$

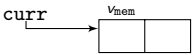
```
for (unsigned int k = 0; k < n; k++)   {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
  tail = curr;                          }
```

Symbolic Execution

```
cmp:                    check "k < n"
body:0: mem = call i8* @malloc(i64 16)
     1: curr = bitcast i8* mem to list*
```

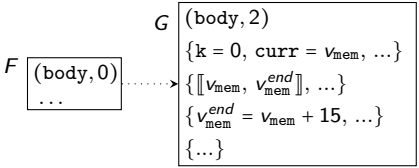



```
for (unsigned int k = 0; k < n; k++)   {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
  tail = curr;                          }
```

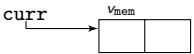


Symbolic Execution

```
cmp:                    check "k < n"
body:0: mem = call i8* @malloc(i64 16)
     1: curr = bitcast i8* mem to list*
     2: nondet = call i32 @nondet_uint()
     3: curr_val =  getelementptr list,
                    list* curr, i32 0, i32 0
     4: store i32 nondet, i32* curr_val
```



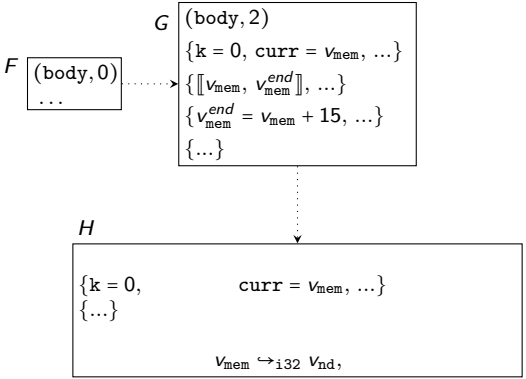

```
for (unsigned int k = 0; k < n; k++)   {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
  tail = curr;                          }
```

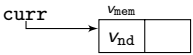


Symbolic Execution

```
cmp:                    check "k < n"
body:0: mem = call i8* @malloc(i64 16)
     1: curr = bitcast i8* mem to list*
     2: nondet = call i32 @nondet_uint()
     3: curr_val =  getelementptr list,
                    list* curr, i32 0, i32 0
     4: store i32 nondet, i32* curr_val
```

```
for (unsigned int k = 0; k < n; k++)   {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
  tail = curr;                          }
```

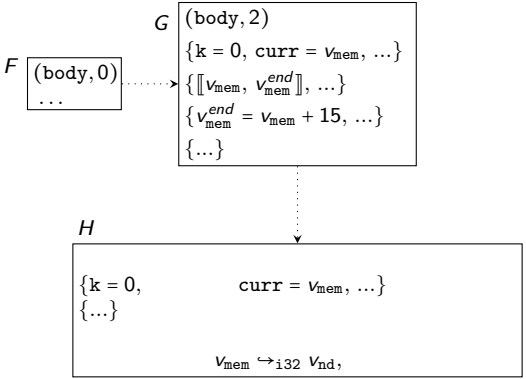


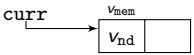


Symbolic Execution

```
cmp:                    check "k < n"
body:0: mem = call i8* @malloc(i64 16)
     1: curr = bitcast i8* mem to list*
     2: nondet = call i32 @nondet_uint()
     3: curr_val =  getelementptr list,
                    list* curr, i32 0, i32 0
     4: store i32 nondet, i32* curr_val
     5: tail = load list*, list** tail_ptr
     6: curr_next = getelementptr list,
                    list* curr, i32 0, i32 1
     7: store list* tail, list** curr_next
```

```
for (unsigned int k = 0; k < n; k++)   {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
  tail = curr;                          }
```

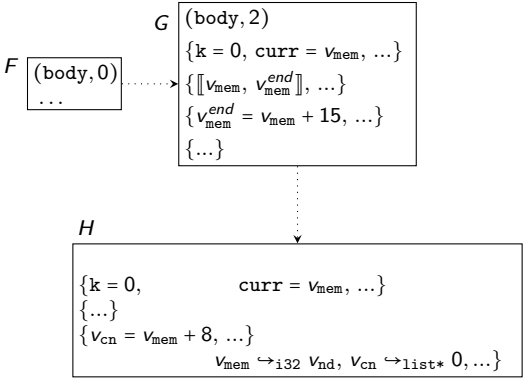


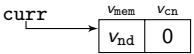


Symbolic Execution

```
cmp:                    check "k < n"
body:0: mem = call i8* @malloc(i64 16)
     1: curr = bitcast i8* mem to list*
     2: nondet = call i32 @nondet_uint()
     3: curr_val =  getelementptr list,
                    list* curr, i32 0, i32 0
     4: store i32 nondet, i32* curr_val
     5: tail = load list*, list** tail_ptr
     6: curr_next = getelementptr list,
                    list* curr, i32 0, i32 1
     7: store list* tail, list** curr_next
```

```
for (unsigned int k = 0; k < n; k++)   {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
  tail = curr;                          }
```

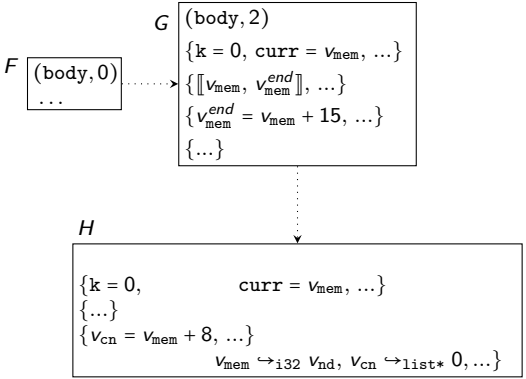


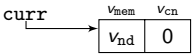


Symbolic Execution

```
cmp:                    check "k < n"
body:0: mem = call i8* @malloc(i64 16)
     1: curr = bitcast i8* mem to list*
     2: nondet = call i32 @nondet_uint()
     3: curr_val =  getelementptr list,
                    list* curr, i32 0, i32 0
     4: store i32 nondet, i32* curr_val
     5: tail = load list*, list** tail_ptr
     6: curr_next = getelementptr list,
                    list* curr, i32 0, i32 1
     7: store list* tail, list** curr_next
     8: store list* curr, list** tail_ptr
     9: kinc = add i32 k, 1
```

```
for (unsigned int k = 0; k < n; k++)   {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
  tail = curr;                          }
```

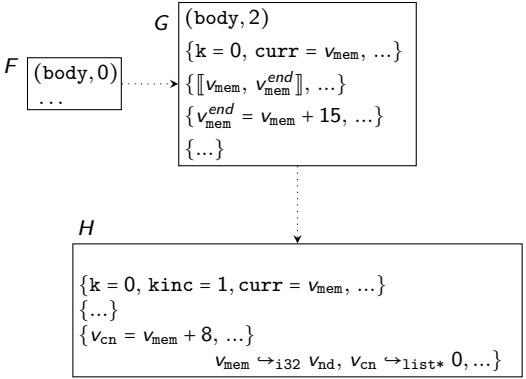


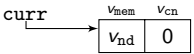


Symbolic Execution

```
cmp:                    check "k < n"
body:0: mem = call i8* @malloc(i64 16)
     1: curr = bitcast i8* mem to list*
     2: nondet = call i32 @nondet_uint()
     3: curr_val =  getelementptr list,
                    list* curr, i32 0, i32 0
     4: store i32 nondet, i32* curr_val
     5: tail = load list*, list** tail_ptr
     6: curr_next = getelementptr list,
                    list* curr, i32 0, i32 1
     7: store list* tail, list** curr_next
     8: store list* curr, list** tail_ptr
     9: kinc = add i32 k, 1
```

```
for (unsigned int k = 0; k < n; k++)   {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
  tail = curr;                          }
```

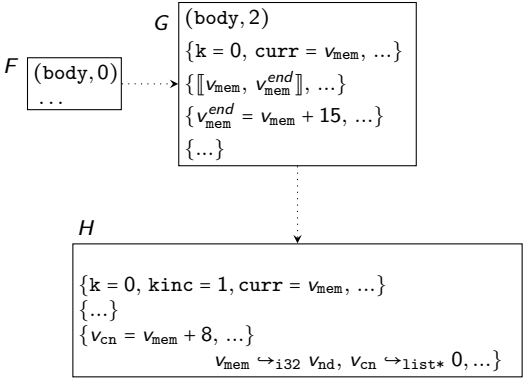


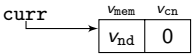


Symbolic Execution

```
cmp:                    check "k < n"
body:0: mem = call i8* @malloc(i64 16)
     1: curr = bitcast i8* mem to list*
     2: nondet = call i32 @nondet_uint()
     3: curr_val =  getelementptr list,
                    list* curr, i32 0, i32 0
     4: store i32 nondet, i32* curr_val
     5: tail = load list*, list** tail_ptr
     6: curr_next = getelementptr list,
                    list* curr, i32 0, i32 1
     7: store list* tail, list** curr_next
     8: store list* curr, list** tail_ptr
     9: kinc = add i32 k, 1
     10:store i32 kinc, i32* k_ad
```

```
for (unsigned int k = 0; k < n; k++)   {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
  tail = curr;                          }
```

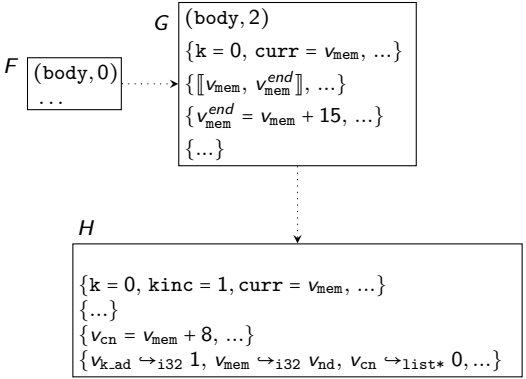


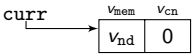


Symbolic Execution

```
cmp:                    check "k < n"
body:0: mem = call i8* @malloc(i64 16)
     1: curr = bitcast i8* mem to list*
     2: nondet = call i32 @nondet_uint()
     3: curr_val =  getelementptr list,
                    list* curr, i32 0, i32 0
     4: store i32 nondet, i32* curr_val
     5: tail = load list*, list** tail_ptr
     6: curr_next = getelementptr list,
                    list* curr, i32 0, i32 1
     7: store list* tail, list** curr_next
     8: store list* curr, list** tail_ptr
     9: kinc = add i32 k, 1
    10:store i32 kinc, i32* k_ad
```

```
for (unsigned int k = 0; k < n; k++)   {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
  tail = curr;                          }
```

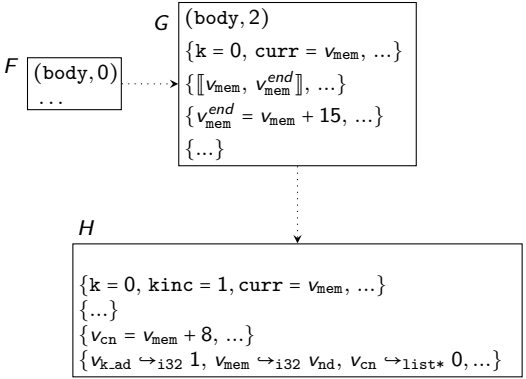


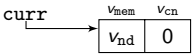


Symbolic Execution

```
cmp:                    check "k < n"
body:0: mem = call i8* @malloc(i64 16)
     1: curr = bitcast i8* mem to list*
     2: nondet = call i32 @nondet_uint()
     3: curr_val =  getelementptr list,
                    list* curr, i32 0, i32 0
     4: store i32 nondet, i32* curr_val
     5: tail = load list*, list** tail_ptr
     6: curr_next = getelementptr list,
                    list* curr, i32 0, i32 1
     7: store list* tail, list** curr_next
     8: store list* curr, list** tail_ptr
     9: kinc = add i32 k, 1
     10:store i32 kinc, i32* k_ad
     11:br label cmp
```

```
for (unsigned int k = 0; k < n; k++)   {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
  tail = curr;                          }
```

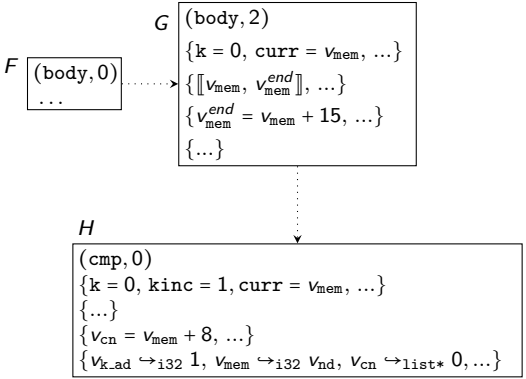


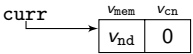


Symbolic Execution

```
cmp:                    check "k < n"
body:0: mem = call i8* @malloc(i64 16)
     1: curr = bitcast i8* mem to list*
     2: nondet = call i32 @nondet_uint()
     3: curr_val =  getelementptr list,
                    list* curr, i32 0, i32 0
     4: store i32 nondet, i32* curr_val
     5: tail = load list*, list** tail_ptr
     6: curr_next = getelementptr list,
                    list* curr, i32 0, i32 1
     7: store list* tail, list** curr_next
     8: store list* curr, list** tail_ptr
     9: kinc = add i32 k, 1
     10:store i32 kinc, i32* k_ad
     11:br label cmp
```

```
for (unsigned int k = 0; k < n; k++)   {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
  tail = curr;                          }
```





Generalization and Inference of List Invariants

$\cdots$

H

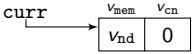
$(\texttt{cmp}, 0)$

$\{\quad \texttt{k} = 0, \texttt{kinc} = 1, \texttt{curr} = v_{\texttt{mem}}, \ldots\}$

$v_{\texttt{cn}} = v_{\texttt{mem}} + 8, \ldots\}$

$\{v_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} 1, v_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} v_{\texttt{nd}}, v_{\texttt{cn}} \hookrightarrow_{\texttt{list*}} 0, \ldots\}$

H :



Generalization and Inference of List Invariants

$\cdots$

H

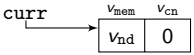
$(\texttt{cmp}, 0)$

$\{\texttt{n} = v_{\texttt{n}},\ \texttt{k} = 0,\ \texttt{kinc} = 1, \texttt{curr} = v_{\texttt{mem}}, \ldots\}$

$\{v_{\texttt{n}} > 0, \qquad v_{\texttt{cn}} = v_{\texttt{mem}} + 8, \ldots\}$

$\{v_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} 1, v_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} v_{\texttt{nd}}, v_{\texttt{cn}} \hookrightarrow_{\texttt{list*}} 0, \ldots\}$

H :



Generalization and Inference of List Invariants

$\cdots$

H

$(\texttt{cmp}, 0)$

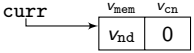
$\{\texttt{n} = v_{\texttt{n}},\ \texttt{k} = 0,\ \texttt{kinc} = 1, \texttt{curr} = v_{\texttt{mem}}, \ldots\}$

$\{[\![v_{\texttt{mem}},\ v_{\texttt{mem}}^{end}]\!],\ \ldots\}$

$\{v_{\texttt{n}} > 0, v_{\texttt{mem}}^{end} = v_{\texttt{mem}} + 15, v_{\texttt{cn}} = v_{\texttt{mem}} + 8, \ldots\}$

$\{v_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} 1, v_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} v_{\texttt{nd}}, v_{\texttt{cn}} \hookrightarrow_{\texttt{list*}} 0, \ldots\}$

H :



Generalization and Inference of List Invariants

$\ldots$

H

$(\texttt{cmp}, 0)$

$\{\texttt{n} = v_{\texttt{n}},\ \texttt{k} = 0,\ \texttt{kinc} = 1,\ \texttt{curr} = v_{\texttt{mem}}, \ldots\}$

$\{[\![v_{\texttt{mem}},\ v_{\texttt{mem}}^{end}]\!], \ldots\}$

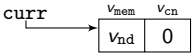
$\{v_{\texttt{n}} > 0,\ v_{\texttt{mem}}^{end} = v_{\texttt{mem}} + 15,\ v_{\texttt{cn}} = v_{\texttt{mem}} + 8, \ldots\}$

$\{v_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} 1,\ v_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} v_{\texttt{nd}},\ v_{\texttt{cn}} \hookrightarrow_{\texttt{list*}} 0, \ldots\}$

I

$(\texttt{cmp}, 0)$

H :



Generalization and Inference of List Invariants

$\ldots$

H

$(\texttt{cmp}, 0)$

$\{\texttt{n} = v_{\texttt{n}},\ \texttt{k} = 0,\ \texttt{kinc} = 1, \texttt{curr} = v_{\texttt{mem}}, \ldots\}$

$\{[\![v_{\texttt{mem}},\ v_{\texttt{mem}}^{end}]\!], \ldots\}$

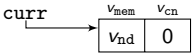
$\{v_{\texttt{n}} > 0, v_{\texttt{mem}}^{end} = v_{\texttt{mem}} + 15, v_{\texttt{cn}} = v_{\texttt{mem}} + 8, \ldots\}$

$\{v_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} 1, v_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} v_{\texttt{nd}}, v_{\texttt{cn}} \hookrightarrow_{\texttt{list*}} 0, \ldots\}$

I

$(\texttt{cmp}, 0)$

H :



I :

Generalization and Inference of List Invariants

$\ldots$

H

$(\texttt{cmp}, 0)$

$\{\texttt{n} = v_{\texttt{n}},\ \texttt{k} = 0,\ \texttt{kinc} = 1, \texttt{curr} = v_{\texttt{mem}}, \ldots\}$

$\{[\![v_{\texttt{mem}},\ v_{\texttt{mem}}^{end}]\!], \ldots\}$

$\{v_{\texttt{n}} > 0, v_{\texttt{mem}}^{end} = v_{\texttt{mem}} + 15, v_{\texttt{cn}} = v_{\texttt{mem}} + 8, \ldots\}$

$\{v_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} 1, v_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} v_{\texttt{nd}}, v_{\texttt{cn}} \hookrightarrow_{\texttt{list*}} 0, \ldots\}$

I

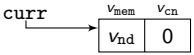
$(\texttt{cmp}, 0)$

$\{\texttt{n} = v_{\texttt{n}},\ \texttt{k} = 1, \texttt{kinc} = 2,$

$\{v_{\texttt{n}} > 1,$

$\{v_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} 2,$

H :



I :

Generalization and Inference of List Invariants

...

H

$(\texttt{cmp}, 0)$

$\{\texttt{n} = v_{\texttt{n}},\ \texttt{k} = 0,\ \texttt{kinc} = 1, \texttt{curr} = v_{\texttt{mem}}, \ldots\}$

$\{[\![v_{\texttt{mem}},\ v_{\texttt{mem}}^{end}]\!], \ldots\}$

$\{v_{\texttt{n}} > 0, v_{\texttt{mem}}^{end} = v_{\texttt{mem}} + 15, v_{\texttt{cn}} = v_{\texttt{mem}} + 8, \ldots\}$

$\{v_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} 1, v_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} v_{\texttt{nd}}, v_{\texttt{cn}} \hookrightarrow_{\texttt{list*}} 0, \ldots\}$

I

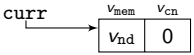
$(\texttt{cmp}, 0)$

$\{\texttt{n} = v_{\texttt{n}},\ \texttt{k} = 1, \texttt{kinc} = 2, \texttt{curr} = w_{\texttt{mem}}, \ldots\}$

$\{v_{\texttt{n}} > 1,$

$\{v_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} 2,$

H :



I :

Generalization and Inference of List Invariants

$\ldots$

H

$(\texttt{cmp}, 0)$

$\{\texttt{n} = v_{\texttt{n}},\ \texttt{k} = 0,\ \texttt{kinc} = 1, \texttt{curr} = v_{\texttt{mem}}, \ldots\}$

$\{[\![v_{\texttt{mem}},\ v_{\texttt{mem}}^{end}]\!],\ \ldots\}$

$\{v_{\texttt{n}} > 0, v_{\texttt{mem}}^{end} = v_{\texttt{mem}} + 15, v_{\texttt{cn}} = v_{\texttt{mem}} + 8, \ldots\}$

$\{v_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} 1, v_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} v_{\texttt{nd}}, v_{\texttt{cn}} \hookrightarrow_{\texttt{list*}} 0, \ldots\}$

I

$(\texttt{cmp}, 0)$

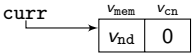
$\{\texttt{n} = v_{\texttt{n}},\ \texttt{k} = 1, \texttt{kinc} = 2, \texttt{curr} = w_{\texttt{mem}}, \ldots\}$

$\{[\![v_{\texttt{mem}},\ v_{\texttt{mem}}^{end}]\!], [\![w_{\texttt{mem}},\ w_{\texttt{mem}}^{end}]\!],\ \ldots\}$

$\{v_{\texttt{n}} > 1,$

$\{v_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} 2,$

H :



I :

Generalization and Inference of List Invariants

$\ldots$

H

$(\texttt{cmp}, 0)$

$\{\texttt{n} = v_{\texttt{n}},\ \texttt{k} = 0,\ \texttt{kinc} = 1, \texttt{curr} = v_{\texttt{mem}}, \ldots\}$

$\{[\![v_{\texttt{mem}},\ v_{\texttt{mem}}^{end}]\!],\ \ldots\}$

$\{v_{\texttt{n}} > 0, v_{\texttt{mem}}^{end} = v_{\texttt{mem}} + 15, v_{\texttt{cn}} = v_{\texttt{mem}} + 8, \ldots\}$

$\{v_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} 1, v_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} v_{\texttt{nd}}, v_{\texttt{cn}} \hookrightarrow_{\texttt{list*}} 0, \ldots\}$

I

$(\texttt{cmp}, 0)$

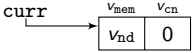
$\{\texttt{n} = v_{\texttt{n}},\ \texttt{k} = 1, \texttt{kinc} = 2, \texttt{curr} = w_{\texttt{mem}}, \ldots\}$

$\{[\![v_{\texttt{mem}},\ v_{\texttt{mem}}^{end}]\!], [\![w_{\texttt{mem}},\ w_{\texttt{mem}}^{end}]\!],\ \ldots\}$

$\{v_{\texttt{n}} > 1, v_{\texttt{mem}}^{end} = v_{\texttt{mem}} + 15,\quad v_{\texttt{cn}} = v_{\texttt{mem}} + 8,$
$\quad w_{\texttt{mem}}^{end} = w_{\texttt{mem}} + 15,\ w_{\texttt{cn}} = w_{\texttt{mem}} + 8, \ldots\}$

$\{v_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} 2,$

H :



I :

Generalization and Inference of List Invariants

$\cdots$

H

$(\texttt{cmp}, 0)$

$\{\texttt{n} = v_{\texttt{n}},\ \texttt{k} = 0,\ \texttt{kinc} = 1, \texttt{curr} = v_{\texttt{mem}}, \ldots\}$

$\{[\![v_{\texttt{mem}},\ v_{\texttt{mem}}^{end}]\!],\ \ldots\}$

$\{v_{\texttt{n}} > 0, v_{\texttt{mem}}^{end} = v_{\texttt{mem}} + 15, v_{\texttt{cn}} = v_{\texttt{mem}} + 8, \ldots\}$

$\{v_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} 1, v_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} v_{\texttt{nd}}, v_{\texttt{cn}} \hookrightarrow_{\texttt{list*}} 0, \ldots\}$

$\cdots$

I

$(\texttt{cmp}, 0)$

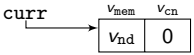
$\{\texttt{n} = v_{\texttt{n}},\ \texttt{k} = 1, \texttt{kinc} = 2, \texttt{curr} = w_{\texttt{mem}}, \ldots\}$

$\{[\![v_{\texttt{mem}},\ v_{\texttt{mem}}^{end}]\!], [\![w_{\texttt{mem}},\ w_{\texttt{mem}}^{end}]\!],\ \ldots\}$

$\{v_{\texttt{n}} > 1, v_{\texttt{mem}}^{end} = v_{\texttt{mem}} + 15,\quad v_{\texttt{cn}} = v_{\texttt{mem}} + 8,$
$\quad w_{\texttt{mem}}^{end} = w_{\texttt{mem}} + 15,\ w_{\texttt{cn}} = w_{\texttt{mem}} + 8, \ldots\}$

$\{v_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} 2, v_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} v_{\texttt{nd}},\quad v_{\texttt{cn}} \hookrightarrow_{\texttt{list*}} 0,$
$\quad w_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} w_{\texttt{nd}}, w_{\texttt{cn}} \hookrightarrow_{\texttt{list*}} v_{\texttt{mem}}, \ldots\}$

H :



I :

Generalization and Inference of List Invariants

$\ldots$

H

$(\texttt{cmp}, 0)$

$\{\texttt{n} = v_{\texttt{n}}, \texttt{k} = 0, \texttt{kinc} = 1, \texttt{curr} = v_{\texttt{mem}}, \ldots\}$

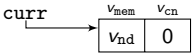
$\{[\![v_{\texttt{mem}}, v_{\texttt{mem}}^{end}]\!], \ldots\}$

$\{v_{\texttt{n}} > 0, v_{\texttt{mem}}^{end} = v_{\texttt{mem}} + 15, v_{\texttt{cn}} = v_{\texttt{mem}} + 8, \ldots\}$

$\{v_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} 1, v_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} v_{\texttt{nd}}, v_{\texttt{cn}} \hookrightarrow_{\texttt{list*}} 0, \ldots\}$

J

H :



I

$(\texttt{cmp}, 0)$

$\{\texttt{n} = v_{\texttt{n}}, \texttt{k} = 1, \texttt{kinc} = 2, \texttt{curr} = w_{\texttt{mem}}, \ldots\}$

$\{[\![v_{\texttt{mem}}, v_{\texttt{mem}}^{end}]\!], [\![w_{\texttt{mem}}, w_{\texttt{mem}}^{end}]\!], \ldots\}$

$\{v_{\texttt{n}} > 1, v_{\texttt{mem}}^{end} = v_{\texttt{mem}} + 15, \quad v_{\texttt{cn}} = v_{\texttt{mem}} + 8,$
$\quad w_{\texttt{mem}}^{end} = w_{\texttt{mem}} + 15, \; w_{\texttt{cn}} = w_{\texttt{mem}} + 8, \ldots\}$

$\{v_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} 2, v_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} v_{\texttt{nd}}, \quad v_{\texttt{cn}} \hookrightarrow_{\texttt{list*}} 0,$
$\quad w_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} w_{\texttt{nd}}, w_{\texttt{cn}} \hookrightarrow_{\texttt{list*}} v_{\texttt{mem}}, \ldots\}$

I :

Generalization and Inference of List Invariants

$\ldots$

H

$(\texttt{cmp}, 0)$

$\{\texttt{n} = v_{\texttt{n}}, \texttt{k} = 0, \texttt{kinc} = 1, \texttt{curr} = v_{\texttt{mem}}, \ldots\}$

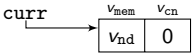
$\{[\![v_{\texttt{mem}}, v_{\texttt{mem}}^{end}]\!], \ldots\}$

$\{v_{\texttt{n}} > 0, v_{\texttt{mem}}^{end} = v_{\texttt{mem}} + 15, v_{\texttt{cn}} = v_{\texttt{mem}} + 8, \ldots\}$

$\{v_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} 1, v_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} v_{\texttt{nd}}, v_{\texttt{cn}} \hookrightarrow_{\texttt{list*}} 0, \ldots\}$

J

H :



I

$(\texttt{cmp}, 0)$

$\{\texttt{n} = v_{\texttt{n}}, \texttt{k} = 1, \texttt{kinc} = 2, \texttt{curr} = w_{\texttt{mem}}, \ldots\}$

$\{[\![v_{\texttt{mem}}, v_{\texttt{mem}}^{end}]\!], [\![w_{\texttt{mem}}, w_{\texttt{mem}}^{end}]\!], \ldots\}$

$\{v_{\texttt{n}} > 1, v_{\texttt{mem}}^{end} = v_{\texttt{mem}} + 15, \quad v_{\texttt{cn}} = v_{\texttt{mem}} + 8,$
$\quad w_{\texttt{mem}}^{end} = w_{\texttt{mem}} + 15, \; w_{\texttt{cn}} = w_{\texttt{mem}} + 8, \ldots\}$

$\{v_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} 2, v_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} v_{\texttt{nd}}, \quad v_{\texttt{cn}} \hookrightarrow_{\texttt{list*}} 0,$
$\quad w_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} w_{\texttt{nd}}, w_{\texttt{cn}} \hookrightarrow_{\texttt{list*}} v_{\texttt{mem}}, \ldots\}$

I :

Generalization and Inference of List Invariants

...

H

$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = v_{\mathtt{n}},\ \mathtt{k} = 0,\ \mathtt{kinc} = 1, \mathtt{curr} = v_{\mathtt{mem}}, \ldots\}$

$\{[\![v_{\mathtt{mem}},\ v_{\mathtt{mem}}^{end}]\!], \ldots\}$

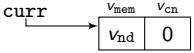
$\{v_{\mathtt{n}} > 0, v_{\mathtt{mem}}^{end} = v_{\mathtt{mem}} + 15, v_{\mathtt{cn}} = v_{\mathtt{mem}} + 8, \ldots\}$

$\{v_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} 1, v_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} v_{\mathtt{nd}}, v_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} 0, \ldots\}$

J

$(\mathtt{cmp}, 0)$

H :



I

$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = v_{\mathtt{n}},\ \mathtt{k} = 1, \mathtt{kinc} = 2, \mathtt{curr} = w_{\mathtt{mem}}, \ldots\}$

$\{[\![v_{\mathtt{mem}},\ v_{\mathtt{mem}}^{end}]\!], [\![w_{\mathtt{mem}},\ w_{\mathtt{mem}}^{end}]\!], \ldots\}$

$\{v_{\mathtt{n}} > 1, v_{\mathtt{mem}}^{end} = v_{\mathtt{mem}} + 15,\quad v_{\mathtt{cn}} = v_{\mathtt{mem}} + 8,$
$\quad w_{\mathtt{mem}}^{end} = w_{\mathtt{mem}} + 15,\ w_{\mathtt{cn}} = w_{\mathtt{mem}} + 8, \ldots\}$

$\{v_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} 2, v_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} v_{\mathtt{nd}},\quad v_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} 0,$
$\quad w_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} w_{\mathtt{nd}}, w_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} v_{\mathtt{mem}}, \ldots\}$

I :

Generalization and Inference of List Invariants

...

H $(\texttt{cmp}, 0)$

$\{\texttt{n} = v_{\texttt{n}}, \texttt{k} = 0, \texttt{kinc} = 1, \texttt{curr} = v_{\texttt{mem}}, \ldots\}$

$\{[\![v_{\texttt{mem}}, v_{\texttt{mem}}^{end}]\!], \ldots\}$

$\{v_{\texttt{n}} > 0, v_{\texttt{mem}}^{end} = v_{\texttt{mem}} + 15, v_{\texttt{cn}} = v_{\texttt{mem}} + 8, \ldots\}$

$\{v_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} 1, v_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} v_{\texttt{nd}}, v_{\texttt{cn}} \hookrightarrow_{\texttt{list*}} 0, \ldots\}$

J $(\texttt{cmp}, 0)$

I $(\texttt{cmp}, 0)$

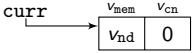
$\{\texttt{n} = v_{\texttt{n}}, \texttt{k} = 1, \texttt{kinc} = 2, \texttt{curr} = w_{\texttt{mem}}, \ldots\}$

$\{[\![v_{\texttt{mem}}, v_{\texttt{mem}}^{end}]\!], [\![w_{\texttt{mem}}, w_{\texttt{mem}}^{end}]\!], \ldots\}$

$\{v_{\texttt{n}} > 1, v_{\texttt{mem}}^{end} = v_{\texttt{mem}} + 15, \quad v_{\texttt{cn}} = v_{\texttt{mem}} + 8,$
$\quad w_{\texttt{mem}}^{end} = w_{\texttt{mem}} + 15, \; w_{\texttt{cn}} = w_{\texttt{mem}} + 8, \ldots\}$

$\{v_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} 2, v_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} v_{\texttt{nd}}, \quad v_{\texttt{cn}} \hookrightarrow_{\texttt{list*}} 0,$
$\quad w_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} w_{\texttt{nd}}, w_{\texttt{cn}} \hookrightarrow_{\texttt{list*}} v_{\texttt{mem}}, \ldots\}$

H :



I :

Generalization and Inference of List Invariants

...

H

$(\texttt{cmp}, 0)$

$\{\texttt{n} = v_{\texttt{n}},\ \texttt{k} = 0,\ \texttt{kinc} = 1,\ \texttt{curr} = v_{\texttt{mem}}, \ldots\}$

$\{[\![v_{\texttt{mem}},\ v_{\texttt{mem}}^{end}]\!], \ldots\}$

$\{v_{\texttt{n}} > 0,\ v_{\texttt{mem}}^{end} = v_{\texttt{mem}} + 15,\ v_{\texttt{cn}} = v_{\texttt{mem}} + 8, \ldots\}$

$\{v_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} 1,\ v_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} v_{\texttt{nd}},\ v_{\texttt{cn}} \hookrightarrow_{\texttt{list*}} 0, \ldots\}$

J

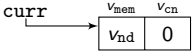
$(\texttt{cmp}, 0)$

$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{k}},\ \texttt{kinc} = x_{\texttt{kinc}},$

$\{\ldots\}$

$\{x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,$

H :



I

$(\texttt{cmp}, 0)$

$\{\texttt{n} = v_{\texttt{n}},\ \texttt{k} = 1, \texttt{kinc} = 2, \texttt{curr} = w_{\texttt{mem}}, \ldots\}$

$\{[\![v_{\texttt{mem}},\ v_{\texttt{mem}}^{end}]\!], [\![w_{\texttt{mem}},\ w_{\texttt{mem}}^{end}]\!], \ldots\}$

$\{v_{\texttt{n}} > 1,\ v_{\texttt{mem}}^{end} = v_{\texttt{mem}} + 15,\quad v_{\texttt{cn}} = v_{\texttt{mem}} + 8,$
$\quad w_{\texttt{mem}}^{end} = w_{\texttt{mem}} + 15,\ w_{\texttt{cn}} = w_{\texttt{mem}} + 8, \ldots\}$

$\{v_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} 2,\ v_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} v_{\texttt{nd}},\quad v_{\texttt{cn}} \hookrightarrow_{\texttt{list*}} 0,$
$\quad w_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} w_{\texttt{nd}}, w_{\texttt{cn}} \hookrightarrow_{\texttt{list*}} v_{\texttt{mem}}, \ldots\}$

I :

Generalization and Inference of List Invariants

...

H

$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = v_{\mathtt{n}},\ \mathtt{k} = 0,\ \mathtt{kinc} = 1, \mathtt{curr} = v_{\mathtt{mem}}, \ldots\}$

$\{[\![v_{\mathtt{mem}},\ v_{\mathtt{mem}}^{end}]\!], \ldots\}$

$\{v_{\mathtt{n}} > 0, v_{\mathtt{mem}}^{end} = v_{\mathtt{mem}} + 15, v_{\mathtt{cn}} = v_{\mathtt{mem}} + 8, \ldots\}$

$\{v_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} 1, v_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} v_{\mathtt{nd}}, v_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} 0, \ldots\}$

J

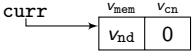
$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = x_{\mathtt{n}},\ \mathtt{k} = x_{\mathtt{k}},\ \mathtt{kinc} = x_{\mathtt{kinc}},$

$\{\ldots\}$

$\{x_{\mathtt{n}} > x_{\mathtt{k}},\ x_{\mathtt{kinc}} = x_{\mathtt{k}} + 1,$

H :



I :

I

$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = v_{\mathtt{n}},\ \mathtt{k} = 1, \mathtt{kinc} = 2, \mathtt{curr} = w_{\mathtt{mem}}, \ldots\}$

$\{[\![v_{\mathtt{mem}},\ v_{\mathtt{mem}}^{end}]\!], [\![w_{\mathtt{mem}},\ w_{\mathtt{mem}}^{end}]\!], \ldots\}$

$\{v_{\mathtt{n}} > 1, v_{\mathtt{mem}}^{end} = v_{\mathtt{mem}} + 15,\quad v_{\mathtt{cn}} = v_{\mathtt{mem}} + 8,$
$\quad w_{\mathtt{mem}}^{end} = w_{\mathtt{mem}} + 15,\ w_{\mathtt{cn}} = w_{\mathtt{mem}} + 8, \ldots\}$

$\{v_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} 2, v_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} v_{\mathtt{nd}},\quad v_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} 0,$
$\quad w_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} w_{\mathtt{nd}}, w_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} v_{\mathtt{mem}}, \ldots\}$

Generalization and Inference of List Invariants

...

H

$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = v_{\mathtt{n}},\ \mathtt{k} = 0,\ \mathtt{kinc} = 1, \mathtt{curr} = v_{\mathtt{mem}}, \ldots\}$

$\{[\![v_{\mathtt{mem}},\ v_{\mathtt{mem}}^{end}]\!], \ldots\}$

$\{v_{\mathtt{n}} > 0, v_{\mathtt{mem}}^{end} = v_{\mathtt{mem}} + 15, v_{\mathtt{cn}} = v_{\mathtt{mem}} + 8, \ldots\}$

$\{v_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} 1, v_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} v_{\mathtt{nd}}, v_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} 0, \ldots\}$

J

$(\mathtt{cmp}, 0)$

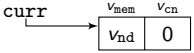
$\{\mathtt{n} = x_{\mathtt{n}},\ \mathtt{k} = x_{\mathtt{k}},\ \mathtt{kinc} = x_{\mathtt{kinc}},$

$\{\ldots\}$

$\{x_{\mathtt{n}} > x_{\mathtt{k}},\ x_{\mathtt{kinc}} = x_{\mathtt{k}} + 1,$

$\{x_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} x_{\mathtt{kinc}}, \ldots\}$

H :



I

$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = v_{\mathtt{n}},\ \mathtt{k} = 1, \mathtt{kinc} = 2, \mathtt{curr} = w_{\mathtt{mem}}, \ldots\}$

$\{[\![v_{\mathtt{mem}},\ v_{\mathtt{mem}}^{end}]\!], [\![w_{\mathtt{mem}},\ w_{\mathtt{mem}}^{end}]\!], \ldots\}$

$\{v_{\mathtt{n}} > 1, v_{\mathtt{mem}}^{end} = v_{\mathtt{mem}} + 15,\quad v_{\mathtt{cn}} = v_{\mathtt{mem}} + 8,$
$\quad w_{\mathtt{mem}}^{end} = w_{\mathtt{mem}} + 15,\ w_{\mathtt{cn}} = w_{\mathtt{mem}} + 8, \ldots\}$

$\{v_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} 2, v_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} v_{\mathtt{nd}},\quad v_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} 0,$
$\quad w_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} w_{\mathtt{nd}}, w_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} v_{\mathtt{mem}}, \ldots\}$

I :

Generalization and Inference of List Invariants

$\ldots$

H

$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = v_{\mathtt{n}},\ \mathtt{k} = 0,\ \mathtt{kinc} = 1, \mathtt{curr} = v_{\mathtt{mem}}, \ldots\}$

$\{[\![v_{\mathtt{mem}},\ v_{\mathtt{mem}}^{end}]\!], \ldots\}$

$\{v_{\mathtt{n}} > 0, v_{\mathtt{mem}}^{end} = v_{\mathtt{mem}} + 15, v_{\mathtt{cn}} = v_{\mathtt{mem}} + 8, \ldots\}$

$\{v_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} 1, v_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} v_{\mathtt{nd}}, v_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} 0, \ldots\}$

J

$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = x_{\mathtt{n}},\ \mathtt{k} = x_{\mathtt{k}},\ \mathtt{kinc} = x_{\mathtt{kinc}},$

$\{\ldots\}$

$\{x_{\mathtt{n}} > x_{\mathtt{k}},\ x_{\mathtt{kinc}} = x_{\mathtt{k}} + 1,$

$\{x_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} x_{\mathtt{kinc}}, \ldots\}$

I

$(\mathtt{cmp}, 0)$

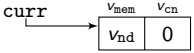
$\{\mathtt{n} = v_{\mathtt{n}},\ \mathtt{k} = 1, \mathtt{kinc} = 2, \mathtt{curr} = w_{\mathtt{mem}}, \ldots\}$

$\{[\![v_{\mathtt{mem}},\ v_{\mathtt{mem}}^{end}]\!], [\![w_{\mathtt{mem}},\ w_{\mathtt{mem}}^{end}]\!], \ldots\}$

$\{v_{\mathtt{n}} > 1, v_{\mathtt{mem}}^{end} = v_{\mathtt{mem}} + 15,\quad v_{\mathtt{cn}} = v_{\mathtt{mem}} + 8,$
$\quad w_{\mathtt{mem}}^{end} = w_{\mathtt{mem}} + 15,\ w_{\mathtt{cn}} = w_{\mathtt{mem}} + 8, \ldots\}$

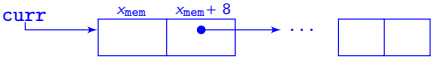
$\{v_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} 2, v_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} v_{\mathtt{nd}},\quad v_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} 0,$
$\quad w_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} w_{\mathtt{nd}}, w_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} v_{\mathtt{mem}}, \ldots\}$

H :



I :

J :



Generalization and Inference of List Invariants

$\cdots$

H

$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = v_{\mathtt{n}},\ \mathtt{k} = 0,\ \mathtt{kinc} = 1,\ \mathtt{curr} = v_{\mathtt{mem}}, \ldots\}$

$\{[\![v_{\mathtt{mem}},\ v_{\mathtt{mem}}^{end}]\!], \ldots\}$

$\{v_{\mathtt{n}} > 0,\ v_{\mathtt{mem}}^{end} = v_{\mathtt{mem}} + 15,\ v_{\mathtt{cn}} = v_{\mathtt{mem}} + 8, \ldots\}$

$\{v_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} 1,\ v_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} v_{\mathtt{nd}},\ v_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} 0, \ldots\}$

J

$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = x_{\mathtt{n}},\ \mathtt{k} = x_{\mathtt{k}},\ \mathtt{kinc} = x_{\mathtt{kinc}},$

$\{\ldots\}$

$\{x_{\mathtt{n}} > x_{\mathtt{k}},\ x_{\mathtt{kinc}} = x_{\mathtt{k}} + 1,$

$\{x_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} x_{\mathtt{kinc}}, \ldots\}$

I

$(\mathtt{cmp}, 0)$

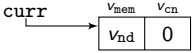
$\{\mathtt{n} = v_{\mathtt{n}},\ \mathtt{k} = 1, \mathtt{kinc} = 2, \mathtt{curr} = w_{\mathtt{mem}}, \ldots\}$

$\{[\![v_{\mathtt{mem}},\ v_{\mathtt{mem}}^{end}]\!], [\![w_{\mathtt{mem}},\ w_{\mathtt{mem}}^{end}]\!], \ldots\}$

$\{v_{\mathtt{n}} > 1,\ v_{\mathtt{mem}}^{end} = v_{\mathtt{mem}} + 15,\quad v_{\mathtt{cn}} = v_{\mathtt{mem}} + 8,$
$\quad w_{\mathtt{mem}}^{end} = w_{\mathtt{mem}} + 15,\ w_{\mathtt{cn}} = w_{\mathtt{mem}} + 8, \ldots\}$

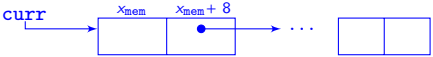
$\{v_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} 2,\ v_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} v_{\mathtt{nd}},\quad v_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} 0,$
$\quad w_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} w_{\mathtt{nd}}, w_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} v_{\mathtt{mem}}, \ldots\}$

H :



I :

J :



Generalization and Inference of List Invariants

...

H

$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = v_{\mathtt{n}},\ \mathtt{k} = 0,\ \mathtt{kinc} = 1,\ \mathtt{curr} = v_{\mathtt{mem}}, \ldots\}$

$\{[\![v_{\mathtt{mem}},\ v_{\mathtt{mem}}^{end}]\!], \ldots\}$

$\{v_{\mathtt{n}} > 0,\ v_{\mathtt{mem}}^{end} = v_{\mathtt{mem}} + 15,\ v_{\mathtt{cn}} = v_{\mathtt{mem}} + 8, \ldots\}$

$\{v_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} 1,\ v_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} v_{\mathtt{nd}},\ v_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} 0, \ldots\}$

J

$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = x_{\mathtt{n}},\ \mathtt{k} = x_{\mathtt{k}},\ \mathtt{kinc} = x_{\mathtt{kinc}},\ \mathtt{curr} = x_{\mathtt{mem}}, \ldots\}$

$\{\ldots\}$

$\{x_{\mathtt{n}} > x_{\mathtt{k}},\ x_{\mathtt{kinc}} = x_{\mathtt{k}} + 1,$

$\{x_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} x_{\mathtt{kinc}}, \ldots\}$

I

$(\mathtt{cmp}, 0)$

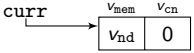
$\{\mathtt{n} = v_{\mathtt{n}},\ \mathtt{k} = 1, \mathtt{kinc} = 2, \mathtt{curr} = w_{\mathtt{mem}}, \ldots\}$

$\{[\![v_{\mathtt{mem}},\ v_{\mathtt{mem}}^{end}]\!], [\![w_{\mathtt{mem}},\ w_{\mathtt{mem}}^{end}]\!], \ldots\}$

$\{v_{\mathtt{n}} > 1,\ v_{\mathtt{mem}}^{end} = v_{\mathtt{mem}} + 15,\quad v_{\mathtt{cn}} = v_{\mathtt{mem}} + 8,$
$\quad w_{\mathtt{mem}}^{end} = w_{\mathtt{mem}} + 15,\ w_{\mathtt{cn}} = w_{\mathtt{mem}} + 8, \ldots\}$

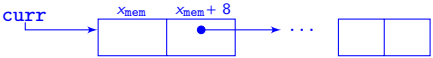
$\{v_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} 2,\ v_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} v_{\mathtt{nd}},\quad v_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} 0,$
$\quad w_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} w_{\mathtt{nd}}, w_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} v_{\mathtt{mem}}, \ldots\}$

H :



I :

J :



Generalization and Inference of List Invariants

...

H

$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = v_{\mathtt{n}},\ \mathtt{k} = 0,\ \mathtt{kinc} = 1, \mathtt{curr} = v_{\mathtt{mem}}, \ldots\}$

$\{[\![v_{\mathtt{mem}},\ v_{\mathtt{mem}}^{end}]\!], \ldots\}$

$\{v_{\mathtt{n}} > 0, v_{\mathtt{mem}}^{end} = v_{\mathtt{mem}} + 15, v_{\mathtt{cn}} = v_{\mathtt{mem}} + 8, \ldots\}$

$\{v_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} 1, v_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} v_{\mathtt{nd}}, v_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} 0, \ldots\}$

J

$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = x_{\mathtt{n}},\ \mathtt{k} = x_{\mathtt{k}},\ \mathtt{kinc} = x_{\mathtt{kinc}},\ \mathtt{curr} = x_{\mathtt{mem}}, \ldots\}$

$\{\ldots\}$

$\{x_{\mathtt{n}} > x_{\mathtt{k}},\ x_{\mathtt{kinc}} = x_{\mathtt{k}} + 1,$

$\{x_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} x_{\mathtt{kinc}}, \ldots\}$

I

$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = v_{\mathtt{n}},\ \mathtt{k} = 1, \mathtt{kinc} = 2, \mathtt{curr} = w_{\mathtt{mem}}, \ldots\}$

$\{[\![v_{\mathtt{mem}},\ v_{\mathtt{mem}}^{end}]\!], [\![w_{\mathtt{mem}},\ w_{\mathtt{mem}}^{end}]\!], \ldots\}$

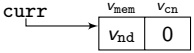
$\{v_{\mathtt{n}} > 1, v_{\mathtt{mem}}^{end} = v_{\mathtt{mem}} + 15,\quad v_{\mathtt{cn}} = v_{\mathtt{mem}} + 8,$

$w_{\mathtt{mem}}^{end} = w_{\mathtt{mem}} + 15,\ w_{\mathtt{cn}} = w_{\mathtt{mem}} + 8, \ldots\}$

$\{v_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} 2, v_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} v_{\mathtt{nd}},\quad v_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} 0,$

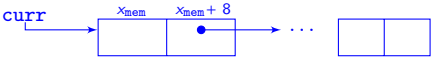
$w_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} w_{\mathtt{nd}}, w_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} v_{\mathtt{mem}}, \ldots\}$

H :



I :

J :



Generalization and Inference of List Invariants

...

H

$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = v_{\mathtt{n}},\ \mathtt{k} = 0,\ \mathtt{kinc} = 1, \mathtt{curr} = v_{\mathtt{mem}}, \ldots\}$

$\{[\![v_{\mathtt{mem}},\ v_{\mathtt{mem}}^{end}]\!], \ldots\}$

$\{v_{\mathtt{n}} > 0, v_{\mathtt{mem}}^{end} = v_{\mathtt{mem}} + 15, v_{\mathtt{cn}} = v_{\mathtt{mem}} + 8, \ldots\}$

$\{v_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} 1, v_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} v_{\mathtt{nd}}, v_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} 0, \ldots\}$

J

$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = x_{\mathtt{n}},\ \mathtt{k} = x_{\mathtt{k}},\ \mathtt{kinc} = x_{\mathtt{kinc}},\ \mathtt{curr} = x_{\mathtt{mem}}, \ldots\}$

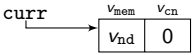
$\{\ldots\}$

$\{x_{\mathtt{n}} > x_{\mathtt{k}},\ x_{\mathtt{kinc}} = x_{\mathtt{k}} + 1,$

$\{x_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} x_{\mathtt{kinc}}, \ldots\}$

$\{x_{\mathtt{mem}} \hookrightarrow_{\mathtt{list}} [(0: \qquad\qquad), (8: \qquad\qquad)]\}$

H :



I

$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = v_{\mathtt{n}},\ \mathtt{k} = 1, \mathtt{kinc} = 2, \mathtt{curr} = w_{\mathtt{mem}}, \ldots\}$

$\{[\![v_{\mathtt{mem}},\ v_{\mathtt{mem}}^{end}]\!], [\![w_{\mathtt{mem}},\ w_{\mathtt{mem}}^{end}]\!], \ldots\}$

$\{v_{\mathtt{n}} > 1, v_{\mathtt{mem}}^{end} = v_{\mathtt{mem}} + 15, \quad v_{\mathtt{cn}} = v_{\mathtt{mem}} + 8,$

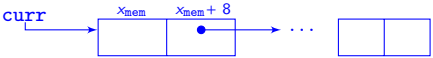
$w_{\mathtt{mem}}^{end} = w_{\mathtt{mem}} + 15,\ w_{\mathtt{cn}} = w_{\mathtt{mem}} + 8, \ldots\}$

$\{v_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} 2, v_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} v_{\mathtt{nd}}, \quad v_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} 0,$

$w_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} w_{\mathtt{nd}}, w_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} v_{\mathtt{mem}}, \ldots\}$

I :

J :



Generalization and Inference of List Invariants

$\cdots$

H

$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = v_{\mathtt{n}},\ \mathtt{k} = 0,\ \mathtt{kinc} = 1, \mathtt{curr} = v_{\mathtt{mem}}, \ldots\}$

$\{[\![v_{\mathtt{mem}},\ v_{\mathtt{mem}}^{end}]\!], \ldots\}$

$\{v_{\mathtt{n}} > 0, v_{\mathtt{mem}}^{end} = v_{\mathtt{mem}} + 15, v_{\mathtt{cn}} = v_{\mathtt{mem}} + 8, \ldots\}$

$\{v_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} 1, v_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} v_{\mathtt{nd}}, v_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} 0, \ldots\}$

J

$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = x_{\mathtt{n}},\ \mathtt{k} = x_{\mathtt{k}},\ \mathtt{kinc} = x_{\mathtt{kinc}},\ \mathtt{curr} = x_{\mathtt{mem}}, \ldots\}$

$\{\ldots\}$

$\{x_{\mathtt{n}} > x_{\mathtt{k}},\ x_{\mathtt{kinc}} = x_{\mathtt{k}} + 1,$

$\{x_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} x_{\mathtt{kinc}}, \ldots\}$

$\{x_{\mathtt{mem}} \xrightarrow{x_\ell}_{\mathtt{list}} [(0: \qquad\qquad), (8: \qquad\qquad)]\}$

I

$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = v_{\mathtt{n}},\ \mathtt{k} = 1, \mathtt{kinc} = 2, \mathtt{curr} = w_{\mathtt{mem}}, \ldots\}$

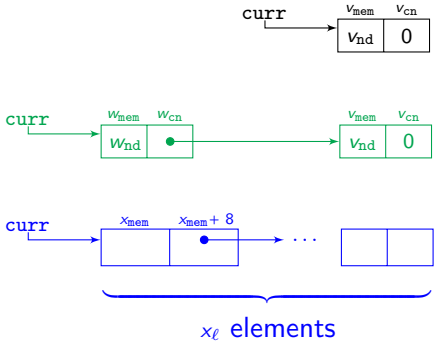
$\{[\![v_{\mathtt{mem}},\ v_{\mathtt{mem}}^{end}]\!], [\![w_{\mathtt{mem}},\ w_{\mathtt{mem}}^{end}]\!], \ldots\}$

$\{v_{\mathtt{n}} > 1, v_{\mathtt{mem}}^{end} = v_{\mathtt{mem}} + 15,\quad v_{\mathtt{cn}} = v_{\mathtt{mem}} + 8,$

$w_{\mathtt{mem}}^{end} = w_{\mathtt{mem}} + 15,\ w_{\mathtt{cn}} = w_{\mathtt{mem}} + 8, \ldots\}$

$\{v_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} 2, v_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} v_{\mathtt{nd}},\quad v_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} 0,$

$w_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} w_{\mathtt{nd}}, w_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} v_{\mathtt{mem}}, \ldots\}$



Generalization and Inference of List Invariants

$\cdots$

H

$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = v_{\mathtt{n}},\ \mathtt{k} = 0,\ \mathtt{kinc} = 1, \mathtt{curr} = v_{\mathtt{mem}}, \ldots\}$

$\{[\![v_{\mathtt{mem}},\ v_{\mathtt{mem}}^{end}]\!], \ldots\}$

$\{v_{\mathtt{n}} > 0, v_{\mathtt{mem}}^{end} = v_{\mathtt{mem}} + 15, v_{\mathtt{cn}} = v_{\mathtt{mem}} + 8, \ldots\}$

$\{v_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} 1, v_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} v_{\mathtt{nd}}, v_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} 0, \ldots\}$

J

$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = x_{\mathtt{n}},\ \mathtt{k} = x_{\mathtt{k}},\ \mathtt{kinc} = x_{\mathtt{kinc}},\ \mathtt{curr} = x_{\mathtt{mem}}, \ldots\}$

$\{\ldots\}$

$\{x_{\mathtt{n}} > x_{\mathtt{k}},\ x_{\mathtt{kinc}} = x_{\mathtt{k}} + 1,\ x_{\ell} = x_{\mathtt{kinc}},\ x_{\ell} \geq 1, \ldots\}$

$\{x_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} x_{\mathtt{kinc}}, \ldots\}$

$\{x_{\mathtt{mem}} \xhookrightarrow{x_{\ell}}_{\mathtt{list}} [(0: \qquad\qquad), (8: \qquad\qquad)]\}$

I

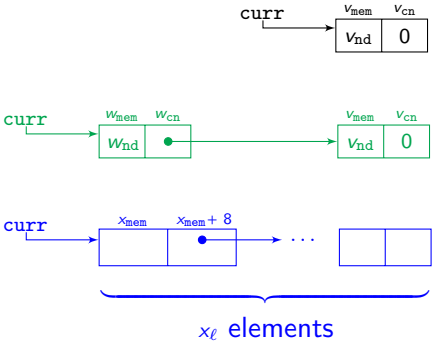
$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = v_{\mathtt{n}},\ \mathtt{k} = 1, \mathtt{kinc} = 2, \mathtt{curr} = w_{\mathtt{mem}}, \ldots\}$

$\{[\![v_{\mathtt{mem}},\ v_{\mathtt{mem}}^{end}]\!], [\![w_{\mathtt{mem}},\ w_{\mathtt{mem}}^{end}]\!], \ldots\}$

$\{v_{\mathtt{n}} > 1, v_{\mathtt{mem}}^{end} = v_{\mathtt{mem}} + 15,\quad v_{\mathtt{cn}} = v_{\mathtt{mem}} + 8,$
$\quad w_{\mathtt{mem}}^{end} = w_{\mathtt{mem}} + 15,\ w_{\mathtt{cn}} = w_{\mathtt{mem}} + 8, \ldots\}$

$\{v_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} 2, v_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} v_{\mathtt{nd}},\quad v_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} 0,$
$\quad w_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} w_{\mathtt{nd}}, w_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} v_{\mathtt{mem}}, \ldots\}$



Generalization and Inference of List Invariants

...

H

$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = v_{\mathtt{n}},\ \mathtt{k} = 0,\ \mathtt{kinc} = 1,\ \mathtt{curr} = v_{\mathtt{mem}}, \ldots\}$

$\{[\![v_{\mathtt{mem}},\ v_{\mathtt{mem}}^{end}]\!], \ldots\}$

$\{v_{\mathtt{n}} > 0,\ v_{\mathtt{mem}}^{end} = v_{\mathtt{mem}} + 15,\ v_{\mathtt{cn}} = v_{\mathtt{mem}} + 8, \ldots\}$

$\{v_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} 1,\ v_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} v_{\mathtt{nd}},\ v_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} 0, \ldots\}$

...

I

$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = v_{\mathtt{n}},\ \mathtt{k} = 1, \mathtt{kinc} = 2, \mathtt{curr} = w_{\mathtt{mem}}, \ldots\}$

$\{[\![v_{\mathtt{mem}},\ v_{\mathtt{mem}}^{end}]\!], [\![w_{\mathtt{mem}},\ w_{\mathtt{mem}}^{end}]\!], \ldots\}$

$\{v_{\mathtt{n}} > 1,\ v_{\mathtt{mem}}^{end} = v_{\mathtt{mem}} + 15,\quad v_{\mathtt{cn}} = v_{\mathtt{mem}} + 8,$
$\quad w_{\mathtt{mem}}^{end} = w_{\mathtt{mem}} + 15,\ w_{\mathtt{cn}} = w_{\mathtt{mem}} + 8, \ldots\}$

$\{v_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} 2,\ v_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} v_{\mathtt{nd}},\ \ v_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} 0,$
$\quad w_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} w_{\mathtt{nd}}, w_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} v_{\mathtt{mem}}, \ldots\}$

J

$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = x_{\mathtt{n}},\ \mathtt{k} = x_{\mathtt{k}},\ \mathtt{kinc} = x_{\mathtt{kinc}},\ \mathtt{curr} = x_{\mathtt{mem}}, \ldots\}$

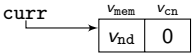
$\{\ldots\}$

$\{x_{\mathtt{n}} > x_{\mathtt{k}},\ x_{\mathtt{kinc}} = x_{\mathtt{k}} + 1,\ x_{\ell} = x_{\mathtt{kinc}},\ x_{\ell} \geq 1, \ldots\}$

$\{x_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} x_{\mathtt{kinc}}, \ldots\}$

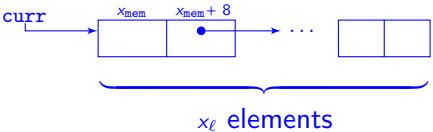
$\{x_{\mathtt{mem}} \xhookrightarrow{x_{\ell}}_{\mathtt{list}} [(0: \qquad\qquad), (8: \qquad\qquad)]\}$

H :



I :

J :



Generalization and Inference of List Invariants

...

H

$(\texttt{cmp}, 0)$

$\{\texttt{n} = v_{\texttt{n}},\ \texttt{k} = 0,\ \texttt{kinc} = 1,\ \texttt{curr} = v_{\texttt{mem}}, \ldots\}$

$\{[\![v_{\texttt{mem}},\ v_{\texttt{mem}}^{end}]\!], \ldots\}$

$\{v_{\texttt{n}} > 0,\ v_{\texttt{mem}}^{end} = v_{\texttt{mem}} + 15,\ v_{\texttt{cn}} = v_{\texttt{mem}} + 8, \ldots\}$

$\{v_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} 1,\ v_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} v_{\texttt{nd}},\ v_{\texttt{cn}} \hookrightarrow_{\texttt{list*}} 0, \ldots\}$

J

$(\texttt{cmp}, 0)$

$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{k}},\ \texttt{kinc} = x_{\texttt{kinc}},\ \texttt{curr} = x_{\texttt{mem}}, \ldots\}$

$\{\ldots\}$

$\{x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ x_\ell = x_{\texttt{kinc}},\ x_\ell \geq 1, \ldots\}$

$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}}, \ldots\}$

$\{x_{\texttt{mem}} \xhookrightarrow{x_\ell}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}} .. \hat{x}_{\texttt{nd}}), (8 : \qquad)]\}$

I

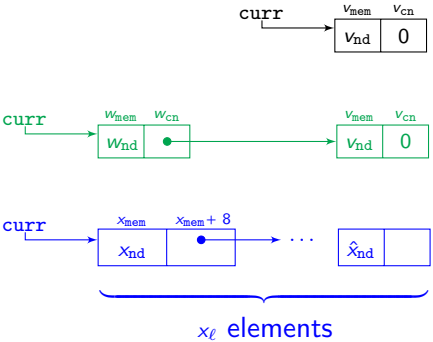
$(\texttt{cmp}, 0)$

$\{\texttt{n} = v_{\texttt{n}},\ \texttt{k} = 1, \texttt{kinc} = 2, \texttt{curr} = w_{\texttt{mem}}, \ldots\}$

$\{[\![v_{\texttt{mem}},\ v_{\texttt{mem}}^{end}]\!], [\![w_{\texttt{mem}},\ w_{\texttt{mem}}^{end}]\!], \ldots\}$

$\{v_{\texttt{n}} > 1,\ v_{\texttt{mem}}^{end} = v_{\texttt{mem}} + 15,\quad v_{\texttt{cn}} = v_{\texttt{mem}} + 8,$
$\quad w_{\texttt{mem}}^{end} = w_{\texttt{mem}} + 15,\ w_{\texttt{cn}} = w_{\texttt{mem}} + 8, \ldots\}$

$\{v_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} 2,\ v_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} v_{\texttt{nd}},\ \ v_{\texttt{cn}} \hookrightarrow_{\texttt{list*}} 0,$
$\quad w_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} w_{\texttt{nd}}, w_{\texttt{cn}} \hookrightarrow_{\texttt{list*}} v_{\texttt{mem}}, \ldots\}$



Generalization and Inference of List Invariants

$\cdots$

H

$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = v_{\mathtt{n}},\ \mathtt{k} = 0,\ \mathtt{kinc} = 1,\ \mathtt{curr} = v_{\mathtt{mem}}, \ldots\}$

$\{[\![v_{\mathtt{mem}},\ v^{end}_{\mathtt{mem}}]\!], \ldots\}$

$\{v_{\mathtt{n}} > 0,\ v^{end}_{\mathtt{mem}} = v_{\mathtt{mem}} + 15,\ v_{\mathtt{cn}} = v_{\mathtt{mem}} + 8, \ldots\}$

$\{v_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} 1,\ v_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} v_{\mathtt{nd}},\ v_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} 0, \ldots\}$

J

$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = x_{\mathtt{n}},\ \mathtt{k} = x_{\mathtt{k}},\ \mathtt{kinc} = x_{\mathtt{kinc}},\ \mathtt{curr} = x_{\mathtt{mem}}, \ldots\}$

$\{\ldots\}$

$\{x_{\mathtt{n}} > x_{\mathtt{k}},\ x_{\mathtt{kinc}} = x_{\mathtt{k}} + 1,\ x_\ell = x_{\mathtt{kinc}},\ x_\ell \geq 1, \ldots\}$

$\{x_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} x_{\mathtt{kinc}}, \ldots\}$

$\{x_{\mathtt{mem}} \xrightarrow{x_\ell}_{\mathtt{list}} [(0 : \mathtt{i32} : x_{\mathtt{nd}} .. \hat{x}_{\mathtt{nd}}), (8 : \qquad)]\}$

I

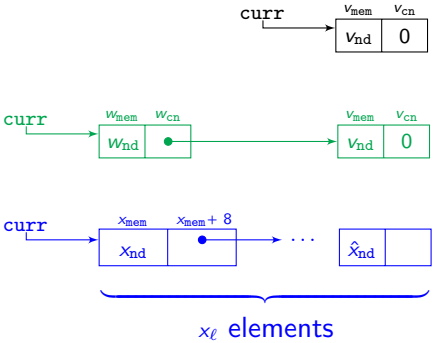
$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = v_{\mathtt{n}},\ \mathtt{k} = 1, \mathtt{kinc} = 2, \mathtt{curr} = w_{\mathtt{mem}}, \ldots\}$

$\{[\![v_{\mathtt{mem}},\ v^{end}_{\mathtt{mem}}]\!], [\![w_{\mathtt{mem}},\ w^{end}_{\mathtt{mem}}]\!], \ldots\}$

$\{v_{\mathtt{n}} > 1,\ v^{end}_{\mathtt{mem}} = v_{\mathtt{mem}} + 15,\quad v_{\mathtt{cn}} = v_{\mathtt{mem}} + 8,$
$\quad w^{end}_{\mathtt{mem}} = w_{\mathtt{mem}} + 15,\ w_{\mathtt{cn}} = w_{\mathtt{mem}} + 8, \ldots\}$

$\{v_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} 2,\ v_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} v_{\mathtt{nd}},\ \ v_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} 0,$
$\quad w_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} w_{\mathtt{nd}}, w_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} v_{\mathtt{mem}}, \ldots\}$



Generalization and Inference of List Invariants

$\cdots$

H

$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = v_{\mathtt{n}},\ \mathtt{k} = 0,\ \mathtt{kinc} = 1,\ \mathtt{curr} = v_{\mathtt{mem}}, \ldots\}$

$\{[\![v_{\mathtt{mem}},\ v_{\mathtt{mem}}^{end}]\!], \ldots\}$

$\{v_{\mathtt{n}} > 0,\ v_{\mathtt{mem}}^{end} = v_{\mathtt{mem}} + 15,\ v_{\mathtt{cn}} = v_{\mathtt{mem}} + 8, \ldots\}$

$\{v_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} 1,\ v_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} v_{\mathtt{nd}},\ v_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} 0, \ldots\}$

$\cdots$

J

$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = x_{\mathtt{n}},\ \mathtt{k} = x_{\mathtt{k}},\ \mathtt{kinc} = x_{\mathtt{kinc}},\ \mathtt{curr} = x_{\mathtt{mem}}, \ldots\}$

$\{\ldots\}$

$\{x_{\mathtt{n}} > x_{\mathtt{k}},\ x_{\mathtt{kinc}} = x_{\mathtt{k}} + 1,\ x_{\ell} = x_{\mathtt{kinc}},\ x_{\ell} \geq 1, \ldots\}$

$\{x_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} x_{\mathtt{kinc}}, \ldots\}$

$\{x_{\mathtt{mem}} \xrightarrow{x_{\ell}}_{\mathtt{list}} [(0 : \mathtt{i32} : x_{\mathtt{nd}}..\hat{x}_{\mathtt{nd}}), (8 : \mathtt{list*} : x_{\mathtt{next}}..0)]\}$

I

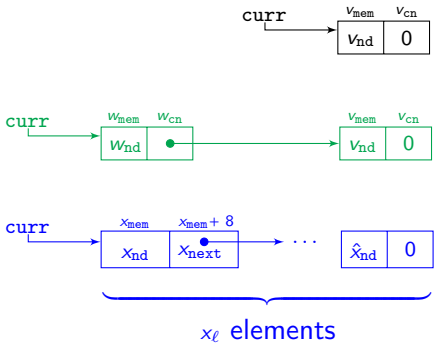
$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = v_{\mathtt{n}},\ \mathtt{k} = 1, \mathtt{kinc} = 2, \mathtt{curr} = w_{\mathtt{mem}}, \ldots\}$

$\{[\![v_{\mathtt{mem}},\ v_{\mathtt{mem}}^{end}]\!], [\![w_{\mathtt{mem}},\ w_{\mathtt{mem}}^{end}]\!], \ldots\}$

$\{v_{\mathtt{n}} > 1,\ v_{\mathtt{mem}}^{end} = v_{\mathtt{mem}} + 15,\quad v_{\mathtt{cn}} = v_{\mathtt{mem}} + 8,$
$\quad w_{\mathtt{mem}}^{end} = w_{\mathtt{mem}} + 15,\ w_{\mathtt{cn}} = w_{\mathtt{mem}} + 8, \ldots\}$

$\{v_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} 2,\ v_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} v_{\mathtt{nd}},\quad v_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} 0,$
$\quad w_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} w_{\mathtt{nd}},\ w_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} v_{\mathtt{mem}}, \ldots\}$



Modifying List Invariants

```
cmp: 0: k = load i32, i32* k_ad    ...
```

J

$(\texttt{cmp}, 0)$

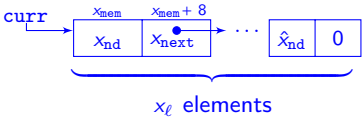
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{k}},\ \texttt{kinc} = x_{\texttt{kinc}},\ \ldots\}$

$\{\ldots\}$

$\{x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ \ldots\}$

$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}},\ \ldots\}$

$\{x_{\texttt{mem}} \xhookrightarrow{x_\ell}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$



Modifying List Invariants

```
cmp: 0: k = load i32, i32* k_ad    ...
```

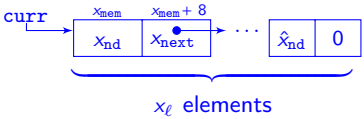
J

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{k}},\ \texttt{kinc} = x_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}},\ \ldots\}$
$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$

K

$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{kinc}},$

$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$



Modifying List Invariants

```
cmp: 0: k = load i32, i32* k_ad   ...
```

J

$(\texttt{cmp}, 0)$

$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{k}},\ \texttt{kinc} = x_{\texttt{kinc}},\ \ldots\}$

$\{\ldots\}$

$\{x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ \ldots\}$

$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}},\ \ldots\}$

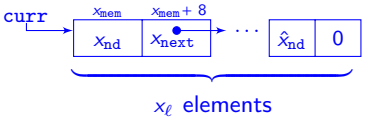
$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}} .. \hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}} .. 0)]\}$

$\ldots$

K

$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{kinc}},$

$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}} .. \hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}} .. 0)]\}$



Modifying List Invariants

```
cmp: 0: k = load i32, i32* k_ad    ...
body:0: mem = call i8* @malloc(i64 16)
     1: curr = bitcast i8* mem to list*
```

J

$(\texttt{cmp}, 0)$

$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{k}},\ \texttt{kinc} = x_{\texttt{kinc}},\ \ldots\}$

$\{\ldots\}$

$\{x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ \ldots\}$

$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}},\ \ldots\}$

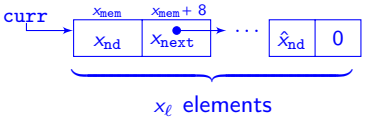
$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$

$\ldots$

K

$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{kinc}},$

$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$



Modifying List Invariants

```
cmp: 0: k = load i32, i32* k_ad    ...
body:0: mem = call i8* @malloc(i64 16)
     1: curr = bitcast i8* mem to list*
```

J

$(\texttt{cmp}, 0)$

$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{k}},\ \texttt{kinc} = x_{\texttt{kinc}},\ \ldots\}$

$\{\ldots\}$

$\{x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ \ldots\}$

$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}},\ \ldots\}$

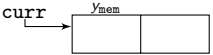
$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}} .. \hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}} .. 0)]\}$

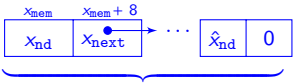
$\ldots$

K

$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{kinc}},$

$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}} .. \hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}} .. 0)]\}$





Modifying List Invariants

```
cmp: 0: k = load i32, i32* k_ad    ...
body:0: mem = call i8* @malloc(i64 16)
     1: curr = bitcast i8* mem to list*
```

J

$(\texttt{cmp}, 0)$

$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{k}},\ \texttt{kinc} = x_{\texttt{kinc}},\ \ldots\}$

$\{\ldots\}$

$\{x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ \ldots\}$

$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}},\ \ldots\}$

$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}} .. \hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}} .. 0)]\}$

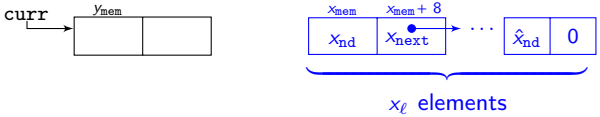
$\ldots$

K

$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{kinc}},\ \texttt{curr} = y_{\texttt{mem}},$

$\{[\![y_{\texttt{mem}},\ y_{\texttt{mem}}^{end}]\!],\ \ldots\}$

$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}} .. \hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}} .. 0)]\}$



Modifying List Invariants

```
cmp: 0: k = load i32, i32* k_ad    ...
body:0: mem = call i8* @malloc(i64 16)
     1: curr = bitcast i8* mem to list*
     2: nondet = call i32 @nondet_uint()
     3: curr_val =  getelementptr list,
                    list* curr, i32 0, i32 0
     4: store i32 nondet, i32* curr_val
```

J

$(\texttt{cmp}, 0)$

$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{k}},\ \texttt{kinc} = x_{\texttt{kinc}},\ \ldots\}$

$\{\ldots\}$

$\{x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ \ldots\}$

$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}},\ \ldots\}$

$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$

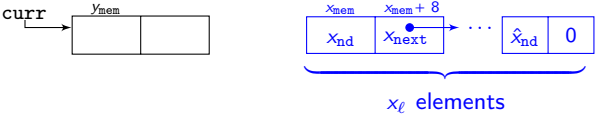
$\ldots$

K

$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{kinc}},\ \texttt{curr} = y_{\texttt{mem}},$

$\{[\![y_{\texttt{mem}},\ y_{\texttt{mem}}^{end}]\!],\ \ldots\}$

$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$



Modifying List Invariants

```
cmp: 0: k = load i32, i32* k_ad    ...
body:0: mem = call i8* @malloc(i64 16)
     1: curr = bitcast i8* mem to list*
     2: nondet = call i32 @nondet_uint()
     3: curr_val =  getelementptr list,
                    list* curr, i32 0, i32 0
     4: store i32 nondet, i32* curr_val
```

J

$(\texttt{cmp}, 0)$

$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{k}},\ \texttt{kinc} = x_{\texttt{kinc}},\ \ldots\}$

$\{\ldots\}$

$\{x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ \ldots\}$

$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}},\ \ldots\}$

$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}} .. \hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}} .. 0)]\}$

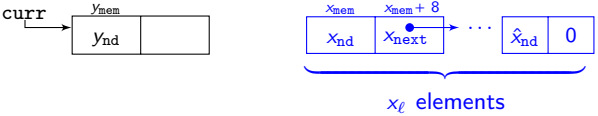
$\ldots$

K

$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{kinc}},\ \texttt{curr} = y_{\texttt{mem}},$

$\{[\![y_{\texttt{mem}},\ y_{\texttt{mem}}^{end}]\!],\ \ldots\}$

$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}} .. \hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}} .. 0)]\}$



Modifying List Invariants

```
cmp: 0: k = load i32, i32* k_ad    ...
body:0: mem = call i8* @malloc(i64 16)
     1: curr = bitcast i8* mem to list*
     2: nondet = call i32 @nondet_uint()
     3: curr_val =  getelementptr list,
                    list* curr, i32 0, i32 0
     4: store i32 nondet, i32* curr_val
```

J

$(\texttt{cmp}, 0)$

$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{k}},\ \texttt{kinc} = x_{\texttt{kinc}},\ \ldots\}$

$\{\ldots\}$

$\{x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ \ldots\}$

$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}},\ \ldots\}$

$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$

$\cdots$

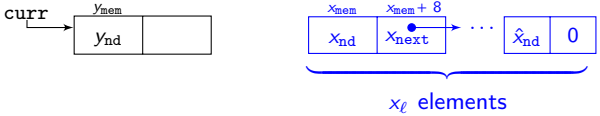
K

$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{kinc}},\ \texttt{curr} = y_{\texttt{mem}},$

$\{[\![y_{\texttt{mem}},\ y_{\texttt{mem}}^{end}]\!],\ \ldots\}$

$\{y_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} y_{\texttt{nd}},\ \ldots\}$

$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$



Modifying List Invariants

```
cmp: 0: k = load i32, i32* k_ad    ...
body:0: mem = call i8* @malloc(i64 16)
     1: curr = bitcast i8* mem to list*
     2: nondet = call i32 @nondet_uint()
     3: curr_val =  getelementptr list,
                    list* curr, i32 0, i32 0
     4: store i32 nondet, i32* curr_val
     5: tail = load list*, list** tail_ptr
     6: curr_next = getelementptr list,
                    list* curr, i32 0, i32 1
```

J

$(\mathtt{cmp}, 0)$

$\{\mathtt{n} = x_{\mathtt{n}},\ \mathtt{k} = x_{\mathtt{k}},\ \mathtt{kinc} = x_{\mathtt{kinc}},\ \ldots\}$

$\{\ldots\}$

$\{x_{\mathtt{n}} > x_{\mathtt{k}},\ x_{\mathtt{kinc}} = x_{\mathtt{k}} + 1,\ \ldots\}$

$\{x_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} x_{\mathtt{kinc}},\ \ldots\}$

$\{x_{\mathtt{mem}} \overset{x_\ell}{\hookrightarrow}_{\mathtt{list}} [(0 : \mathtt{i32} : x_{\mathtt{nd}}..\hat{x}_{\mathtt{nd}}), (8 : \mathtt{list*} : x_{\mathtt{next}}..0)]\}$

$\cdots$

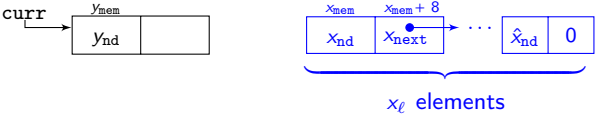
K

$\{\mathtt{n} = x_{\mathtt{n}},\ \mathtt{k} = x_{\mathtt{kinc}},\ \mathtt{curr} = y_{\mathtt{mem}},$

$\{[\![y_{\mathtt{mem}},\ y_{\mathtt{mem}}^{end}]\!],\ \ldots\}$

$\{y_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} y_{\mathtt{nd}},\ \ldots\}$

$\{x_{\mathtt{mem}} \overset{x_\ell}{\hookrightarrow}_{\mathtt{list}} [(0 : \mathtt{i32} : x_{\mathtt{nd}}..\hat{x}_{\mathtt{nd}}), (8 : \mathtt{list*} : x_{\mathtt{next}}..0)]\}$



Modifying List Invariants

```
cmp: 0: k = load i32, i32* k_ad   ...
body:0: mem = call i8* @malloc(i64 16)
     1: curr = bitcast i8* mem to list*
     2: nondet = call i32 @nondet_uint()
     3: curr_val =  getelementptr list,
                    list* curr, i32 0, i32 0
     4: store i32 nondet, i32* curr_val
     5: tail = load list*, list** tail_ptr
     6: curr_next = getelementptr list,
                    list* curr, i32 0, i32 1
```

J

$(\texttt{cmp}, 0)$

$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{k}},\ \texttt{kinc} = x_{\texttt{kinc}},\ \ldots\}$

$\{\ldots\}$

$\{x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ \ldots\}$

$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}},\ \ldots\}$

$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$

$\cdots$

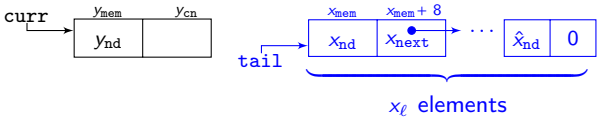
K

$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{kinc}},\ \texttt{curr} = y_{\texttt{mem}},$

$\{[\![y_{\texttt{mem}},\ y_{\texttt{mem}}^{end}]\!],\ \ldots\}$

$\{y_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} y_{\texttt{nd}},\ \ldots\}$

$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$



Modifying List Invariants

```
cmp: 0: k = load i32, i32* k_ad    ...
body:0: mem = call i8* @malloc(i64 16)
     1: curr = bitcast i8* mem to list*
     2: nondet = call i32 @nondet_uint()
     3: curr_val =  getelementptr list,
                    list* curr, i32 0, i32 0
     4: store i32 nondet, i32* curr_val
     5: tail = load list*, list** tail_ptr
     6: curr_next = getelementptr list,
                    list* curr, i32 0, i32 1
```

J

$(\texttt{cmp}, 0)$

$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{k}},\ \texttt{kinc} = x_{\texttt{kinc}},\ \ldots\}$

$\{\ldots\}$

$\{x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ \ldots\}$

$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}},\ \ldots\}$

$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$

$\ldots$

K

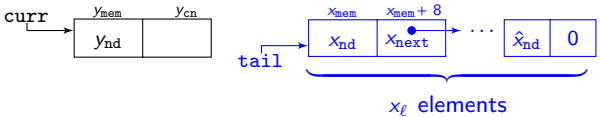
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{kinc}},\ \texttt{curr} = y_{\texttt{mem}},\ \texttt{tail} = x_{\texttt{mem}},\ \texttt{curr\_next} = y_{\texttt{cn}},\ \ldots\}$

$\{[\![y_{\texttt{mem}},\ y_{\texttt{mem}}^{end}]\!],\ \ldots\}$

$\{y_{\texttt{cn}} = y_{\texttt{mem}} + 8,\ \ldots\}$

$\{y_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} y_{\texttt{nd}},\ \ldots\}$

$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$



Modifying List Invariants

```
cmp: 0: k = load i32, i32* k_ad    ...
body:0: mem = call i8* @malloc(i64 16)
     1: curr = bitcast i8* mem to list*
     2: nondet = call i32 @nondet_uint()
     3: curr_val =  getelementptr list,
                    list* curr, i32 0, i32 0
     4: store i32 nondet, i32* curr_val
     5: tail = load list*, list** tail_ptr
     6: curr_next = getelementptr list,
                    list* curr, i32 0, i32 1
```

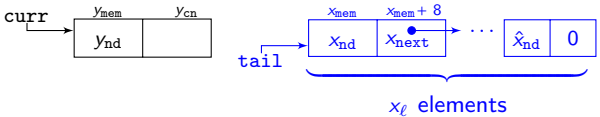
J

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}}, \texttt{k} = x_{\texttt{k}}, \texttt{kinc} = x_{\texttt{kinc}}, \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{k}}, x_{\texttt{kinc}} = x_{\texttt{k}} + 1, \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}}, \ldots\}$
$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}} .. \hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}} .. 0)]\}$

$\cdots$

K

$(\texttt{body}, 7)$
$\{\texttt{n} = x_{\texttt{n}}, \texttt{k} = x_{\texttt{kinc}}, \texttt{curr} = y_{\texttt{mem}}, \texttt{tail} = x_{\texttt{mem}}, \texttt{curr\_next} = y_{\texttt{cn}}, \ldots\}$
$\{[\![y_{\texttt{mem}}, y_{\texttt{mem}}^{end}]\!], \ldots\}$
$\{y_{\texttt{cn}} = y_{\texttt{mem}} + 8, \ldots\}$
$\{y_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} y_{\texttt{nd}}, \ldots\}$
$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}} .. \hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}} .. 0)]\}$



Modifying List Invariants

```
cmp: 0: k = load i32, i32* k_ad   ...
body:0: mem = call i8* @malloc(i64 16)
     1: curr = bitcast i8* mem to list*
     2: nondet = call i32 @nondet_uint()
     3: curr_val =  getelementptr list,
                    list* curr, i32 0, i32 0
     4: store i32 nondet, i32* curr_val
     5: tail = load list*, list** tail_ptr
     6: curr_next = getelementptr list,
                    list* curr, i32 0, i32 1
     7: store list* tail, list** curr_next
```

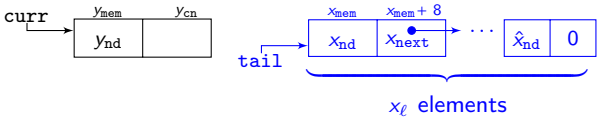
J

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{k}},\ \texttt{kinc} = x_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}},\ \ldots\}$
$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$

$\cdots$

K

$(\texttt{body}, 7)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{kinc}},\ \texttt{curr} = y_{\texttt{mem}},\ \texttt{tail} = x_{\texttt{mem}},\ \texttt{curr\_next} = y_{\texttt{cn}},\ \ldots\}$
$\{[\![y_{\texttt{mem}},\ y_{\texttt{mem}}^{end}]\!],\ \ldots\}$
$\{y_{\texttt{cn}} = y_{\texttt{mem}} + 8,\ \ldots\}$
$\{y_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} y_{\texttt{nd}},\ \ldots\}$
$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$



Modifying List Invariants

```
cmp: 0: k = load i32, i32* k_ad    ...
body:0: mem = call i8* @malloc(i64 16)
     1: curr = bitcast i8* mem to list*
     2: nondet = call i32 @nondet_uint()
     3: curr_val =  getelementptr list,
                    list* curr, i32 0, i32 0
     4: store i32 nondet, i32* curr_val
     5: tail = load list*, list** tail_ptr
     6: curr_next = getelementptr list,
                    list* curr, i32 0, i32 1
     7: store list* tail, list** curr_next
```

J

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}}, \texttt{k} = x_{\texttt{k}}, \texttt{kinc} = x_{\texttt{kinc}}, \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{k}}, x_{\texttt{kinc}} = x_{\texttt{k}} + 1, \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}}, \ldots\}$
$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}} .. \hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}} .. 0)]\}$

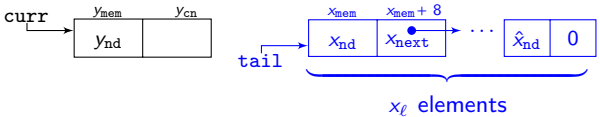
$\ldots$

K

$(\texttt{body}, 7)$
$\{\texttt{n} = x_{\texttt{n}}, \texttt{k} = x_{\texttt{kinc}}, \texttt{curr} = y_{\texttt{mem}}, \texttt{tail} = x_{\texttt{mem}}, \texttt{curr\_next} = y_{\texttt{cn}}, \ldots\}$
$\{[\![y_{\texttt{mem}}, y_{\texttt{mem}}^{end}]\!], \ldots\}$
$\{y_{\texttt{cn}} = y_{\texttt{mem}} + 8, \ldots\}$
$\{y_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} y_{\texttt{nd}}, \ldots\}$
$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}} .. \hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}} .. 0)]\}$

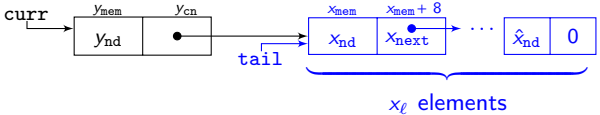
L

$(\texttt{body}, 8)$
$\ldots$
$\{y_{\texttt{cn}} = y_{\texttt{mem}} + 8,$
$\{\ldots\}$



Modifying List Invariants

```
cmp: 0: k = load i32, i32* k_ad    ...
body:0: mem = call i8* @malloc(i64 16)
     1: curr = bitcast i8* mem to list*
     2: nondet = call i32 @nondet_uint()
     3: curr_val =  getelementptr list,
                    list* curr, i32 0, i32 0
     4: store i32 nondet, i32* curr_val
     5: tail = load list*, list** tail_ptr
     6: curr_next = getelementptr list,
                    list* curr, i32 0, i32 1
     7: store list* tail, list** curr_next
```



J

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}}, \texttt{k} = x_{\texttt{k}}, \texttt{kinc} = x_{\texttt{kinc}}, \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{k}}, x_{\texttt{kinc}} = x_{\texttt{k}} + 1, \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}}, \ldots\}$
$\{x_{\texttt{mem}} \xhookrightarrow{x_\ell}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$

$\ldots$

K

$(\texttt{body}, 7)$
$\{\texttt{n} = x_{\texttt{n}}, \texttt{k} = x_{\texttt{kinc}}, \texttt{curr} = y_{\texttt{mem}}, \texttt{tail} = x_{\texttt{mem}}, \texttt{curr\_next} = y_{\texttt{cn}}, \ldots\}$
$\{[\![y_{\texttt{mem}}, y_{\texttt{mem}}^{end}]\!], \ldots\}$
$\{y_{\texttt{cn}} = y_{\texttt{mem}} + 8, \ldots\}$
$\{y_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} y_{\texttt{nd}}, \ldots\}$
$\{x_{\texttt{mem}} \xhookrightarrow{x_\ell}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$

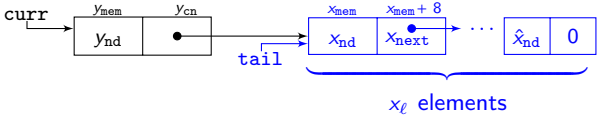
L

$(\texttt{body}, 8)$
$\ldots$
$\{y_{\texttt{cn}} = y_{\texttt{mem}} + 8,$
$\{\ldots\}$

Modifying List Invariants

```
cmp: 0: k = load i32, i32* k_ad    ...
body:0: mem = call i8* @malloc(i64 16)
     1: curr = bitcast i8* mem to list*
     2: nondet = call i32 @nondet_uint()
     3: curr_val =  getelementptr list,
                    list* curr, i32 0, i32 0
     4: store i32 nondet, i32* curr_val
     5: tail = load list*, list** tail_ptr
     6: curr_next = getelementptr list,
                    list* curr, i32 0, i32 1
     7: store list* tail, list** curr_next
```

Instead of adding $y_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} x_{\mathtt{mem}}$, extend list invariant.



J

$(\mathtt{cmp}, 0)$
$\{\mathtt{n} = x_{\mathtt{n}},\ \mathtt{k} = x_{\mathtt{k}},\ \mathtt{kinc} = x_{\mathtt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\mathtt{n}} > x_{\mathtt{k}},\ x_{\mathtt{kinc}} = x_{\mathtt{k}} + 1,\ \ldots\}$
$\{x_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} x_{\mathtt{kinc}},\ \ldots\}$
$\{x_{\mathtt{mem}} \overset{x_\ell}{\hookrightarrow}_{\mathtt{list}} [(0 : \mathtt{i32} : x_{\mathtt{nd}}..\hat{x}_{\mathtt{nd}}), (8 : \mathtt{list*} : x_{\mathtt{next}}..0)]\}$

$\ldots$

K

$(\mathtt{body}, 7)$
$\{\mathtt{n} = x_{\mathtt{n}},\ \mathtt{k} = x_{\mathtt{kinc}},\ \mathtt{curr} = y_{\mathtt{mem}},\ \mathtt{tail} = x_{\mathtt{mem}},\ \mathtt{curr\_next} = y_{\mathtt{cn}},\ \ldots\}$
$\{[\![y_{\mathtt{mem}},\ y_{\mathtt{mem}}^{end}]\!],\ \ldots\}$
$\{y_{\mathtt{cn}} = y_{\mathtt{mem}} + 8,\ \ldots\}$
$\{y_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} y_{\mathtt{nd}},\ \ldots\}$
$\{x_{\mathtt{mem}} \overset{x_\ell}{\hookrightarrow}_{\mathtt{list}} [(0 : \mathtt{i32} : x_{\mathtt{nd}}..\hat{x}_{\mathtt{nd}}), (8 : \mathtt{list*} : x_{\mathtt{next}}..0)]\}$

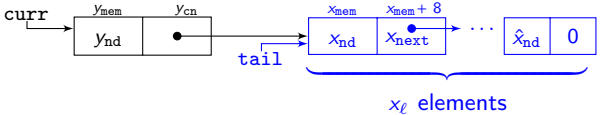
L

$(\mathtt{body}, 8)$
$\ldots$
$\{y_{\mathtt{cn}} = y_{\mathtt{mem}} + 8,$
$\{\ldots\}$

Modifying List Invariants

```
cmp: 0: k = load i32, i32* k_ad    ...
body:0: mem = call i8* @malloc(i64 16)
     1: curr = bitcast i8* mem to list*
     2: nondet = call i32 @nondet_uint()
     3: curr_val =  getelementptr list,
                    list* curr, i32 0, i32 0
     4: store i32 nondet, i32* curr_val
     5: tail = load list*, list** tail_ptr
     6: curr_next = getelementptr list,
                    list* curr, i32 0, i32 1
     7: store list* tail, list** curr_next
```

Instead of adding $y_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} x_{\mathtt{mem}}$, extend list invariant.



J

$(\mathtt{cmp}, 0)$
$\{\mathtt{n} = x_{\mathtt{n}},\ \mathtt{k} = x_{\mathtt{k}},\ \mathtt{kinc} = x_{\mathtt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\mathtt{n}} > x_{\mathtt{k}},\ x_{\mathtt{kinc}} = x_{\mathtt{k}} + 1,\ \ldots\}$
$\{x_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} x_{\mathtt{kinc}},\ \ldots\}$
$\{x_{\mathtt{mem}} \overset{x_\ell}{\hookrightarrow}_{\mathtt{list}} [(0 : \mathtt{i32} : x_{\mathtt{nd}}..\hat{x}_{\mathtt{nd}}), (8 : \mathtt{list*} : x_{\mathtt{next}}..0)]\}$

$\ldots$

K

$(\mathtt{body}, 7)$
$\{\mathtt{n} = x_{\mathtt{n}},\ \mathtt{k} = x_{\mathtt{kinc}},\ \mathtt{curr} = y_{\mathtt{mem}},\ \mathtt{tail} = x_{\mathtt{mem}},\ \mathtt{curr\_next} = y_{\mathtt{cn}},\ \ldots\}$
$\{[\![y_{\mathtt{mem}},\ y_{\mathtt{mem}}^{end}]\!],\ \ldots\}$
$\{y_{\mathtt{cn}} = y_{\mathtt{mem}} + 8,\ \ldots\}$
$\{y_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} y_{\mathtt{nd}},\ \ldots\}$
$\{x_{\mathtt{mem}} \overset{x_\ell}{\hookrightarrow}_{\mathtt{list}} [(0 : \mathtt{i32} : x_{\mathtt{nd}}..\hat{x}_{\mathtt{nd}}), (8 : \mathtt{list*} : x_{\mathtt{next}}..0)]\}$

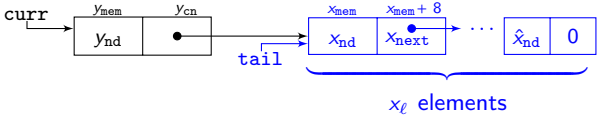
L

$(\mathtt{body}, 8)$
$\ldots$
$\{y_{\mathtt{cn}} = y_{\mathtt{mem}} + 8,$
$\{\ldots\}$
$\{y_{\mathtt{mem}} \overset{y_\ell}{\hookrightarrow}_{\mathtt{list}} [(0 : \mathtt{i32} : y_{\mathtt{nd}}..\hat{x}_{\mathtt{nd}}), (8 : \mathtt{list*} : x_{\mathtt{mem}}..0)]\}$

Modifying List Invariants

```
cmp: 0: k = load i32, i32* k_ad    ...
body:0: mem = call i8* @malloc(i64 16)
     1: curr = bitcast i8* mem to list*
     2: nondet = call i32 @nondet_uint()
     3: curr_val =  getelementptr list,
                    list* curr, i32 0, i32 0
     4: store i32 nondet, i32* curr_val
     5: tail = load list*, list** tail_ptr
     6: curr_next = getelementptr list,
                    list* curr, i32 0, i32 1
     7: store list* tail, list** curr_next
```

Instead of adding $y_{\mathtt{cn}} \hookrightarrow_{\mathtt{list*}} x_{\mathtt{mem}}$, extend list invariant.



J

$(\mathtt{cmp}, 0)$
$\{\mathtt{n} = x_{\mathtt{n}},\ \mathtt{k} = x_{\mathtt{k}},\ \mathtt{kinc} = x_{\mathtt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\mathtt{n}} > x_{\mathtt{k}},\ x_{\mathtt{kinc}} = x_{\mathtt{k}} + 1,\ \ldots\}$
$\{x_{\mathtt{k\_ad}} \hookrightarrow_{\mathtt{i32}} x_{\mathtt{kinc}},\ \ldots\}$
$\{x_{\mathtt{mem}} \xhookrightarrow{x_\ell}_{\mathtt{list}} [(0 : \mathtt{i32} : x_{\mathtt{nd}}..\hat{x}_{\mathtt{nd}}), (8 : \mathtt{list*} : x_{\mathtt{next}}..0)]\}$

$\ldots$

K

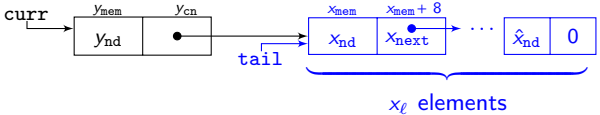
$(\mathtt{body}, 7)$
$\{\mathtt{n} = x_{\mathtt{n}},\ \mathtt{k} = x_{\mathtt{kinc}},\ \mathtt{curr} = y_{\mathtt{mem}},\ \mathtt{tail} = x_{\mathtt{mem}},\ \mathtt{curr\_next} = y_{\mathtt{cn}},\ \ldots\}$
$\{[\![y_{\mathtt{mem}},\ y_{\mathtt{mem}}^{end}]\!],\ \ldots\}$
$\{y_{\mathtt{cn}} = y_{\mathtt{mem}} + 8,\ \ldots\}$
$\{y_{\mathtt{mem}} \hookrightarrow_{\mathtt{i32}} y_{\mathtt{nd}},\ \ldots\}$
$\{x_{\mathtt{mem}} \xhookrightarrow{x_\ell}_{\mathtt{list}} [(0 : \mathtt{i32} : x_{\mathtt{nd}}..\hat{x}_{\mathtt{nd}}), (8 : \mathtt{list*} : x_{\mathtt{next}}..0)]\}$

L

$(\mathtt{body}, 8)$
$\ldots$
$\{y_{\mathtt{cn}} = y_{\mathtt{mem}} + 8,\ y_\ell = x_\ell + 1,\ \ldots\}$
$\{\ldots\}$
$\{y_{\mathtt{mem}} \xhookrightarrow{y_\ell}_{\mathtt{list}} [(0 : \mathtt{i32} : y_{\mathtt{nd}}..\hat{x}_{\mathtt{nd}}), (8 : \mathtt{list*} : x_{\mathtt{mem}}..0)]\}$

Modifying List Invariants

```
cmp: 0: k = load i32, i32* k_ad    ...
body:0: mem = call i8* @malloc(i64 16)
     1: curr = bitcast i8* mem to list*
     2: nondet = call i32 @nondet_uint()
     3: curr_val =  getelementptr list,
                    list* curr, i32 0, i32 0
     4: store i32 nondet, i32* curr_val
     5: tail = load list*, list** tail_ptr
     6: curr_next = getelementptr list,
                    list* curr, i32 0, i32 1
     7: store list* tail, list** curr_next
     8: store list* curr, list** tail_ptr
     9: kinc = add i32 k, 1
    10:store i32 kinc, i32* k_ad
    11:br label cmp
```



J

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\text{n}}, \texttt{k} = x_{\text{k}}, \texttt{kinc} = x_{\text{kinc}}, \ldots\}$
$\{\ldots\}$
$\{x_{\text{n}} > x_{\text{k}}, x_{\text{kinc}} = x_{\text{k}} + 1, \ldots\}$
$\{x_{\text{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\text{kinc}}, \ldots\}$
$\{x_{\text{mem}} \xhookrightarrow{x_\ell}_{\texttt{list}} [(0 : \texttt{i32} : x_{\text{nd}}..\hat{x}_{\text{nd}}), (8 : \texttt{list*} : x_{\text{next}}..0)]\}$

$\cdots$

K

$(\texttt{body}, 7)$
$\{\texttt{n} = x_{\text{n}}, \texttt{k} = x_{\text{kinc}}, \texttt{curr} = y_{\text{mem}}, \texttt{tail} = x_{\text{mem}}, \texttt{curr\_next} = y_{\text{cn}}, \ldots\}$
$\{[\![y_{\text{mem}}, y_{\text{mem}}^{end}]\!], \ldots\}$
$\{y_{\text{cn}} = y_{\text{mem}} + 8, \ldots\}$
$\{y_{\text{mem}} \hookrightarrow_{\texttt{i32}} y_{\text{nd}}, \ldots\}$
$\{x_{\text{mem}} \xhookrightarrow{x_\ell}_{\texttt{list}} [(0 : \texttt{i32} : x_{\text{nd}}..\hat{x}_{\text{nd}}), (8 : \texttt{list*} : x_{\text{next}}..0)]\}$

L

$(\texttt{body}, 8)$
$\ldots$
$\{y_{\text{cn}} = y_{\text{mem}} + 8, y_\ell = x_\ell + 1, \ldots\}$
$\{\ldots\}$
$\{y_{\text{mem}} \xhookrightarrow{y_\ell}_{\texttt{list}} [(0 : \texttt{i32} : y_{\text{nd}}..\hat{x}_{\text{nd}}), (8 : \texttt{list*} : x_{\text{mem}}..0)]\}$

Modifying List Invariants

M

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\text{n}},\ \texttt{k} = x_{\text{kinc}},\ \texttt{kinc} = y_{\text{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\text{n}} > x_{\text{kinc}},\ y_{\text{kinc}} = x_{\text{kinc}} + 1,\ \ldots\}$
$\{x_{\text{k\_ad}} \hookrightarrow_{\texttt{i32}} y_{\text{kinc}},\ \ldots\}$
$\{y_{\text{mem}} \overset{y_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : y_{\text{nd}}..\hat{x}_{\text{nd}}), (8 : \texttt{list*} : x_{\text{mem}}..0)]\}$

J

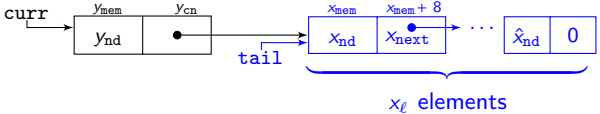
$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\text{n}},\ \texttt{k} = x_{\text{k}},\ \texttt{kinc} = x_{\text{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\text{n}} > x_{\text{k}},\ x_{\text{kinc}} = x_{\text{k}} + 1,\ \ldots\}$
$\{x_{\text{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\text{kinc}},\ \ldots\}$
$\{x_{\text{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\text{nd}}..\hat{x}_{\text{nd}}), (8 : \texttt{list*} : x_{\text{next}}..0)]\}$

$\ldots$

K

$(\texttt{body}, 7)$
$\{\texttt{n} = x_{\text{n}},\ \texttt{k} = x_{\text{kinc}},\ \texttt{curr} = y_{\text{mem}},\ \texttt{tail} = x_{\text{mem}},\ \texttt{curr\_next} = y_{\text{cn}},\ \ldots\}$
$\{[\![y_{\text{mem}},\ y_{\text{mem}}^{end}]\!],\ \ldots\}$
$\{y_{\text{cn}} = y_{\text{mem}} + 8,\ \ldots\}$
$\{y_{\text{mem}} \hookrightarrow_{\texttt{i32}} y_{\text{nd}},\ \ldots\}$
$\{x_{\text{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\text{nd}}..\hat{x}_{\text{nd}}), (8 : \texttt{list*} : x_{\text{next}}..0)]\}$

```
8: store list* curr, list** tail_ptr
9: kinc = add i32 k, 1
10:store i32 kinc, i32* k_ad
11:br label cmp
```



L

$(\texttt{body}, 8)$
$\ldots$
$\{y_{\text{cn}} = y_{\text{mem}} + 8,\ y_\ell = x_\ell + 1,\ \ldots\}$
$\{\ldots\}$
$\{y_{\text{mem}} \overset{y_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : y_{\text{nd}}..\hat{x}_{\text{nd}}), (8 : \texttt{list*} : x_{\text{mem}}..0)]\}$

Modifying List Invariants

M

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\mathrm{n}},\ \texttt{k} = x_{\texttt{kinc}},\ \texttt{kinc} = y_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\mathrm{n}} > x_{\texttt{kinc}},\ y_{\texttt{kinc}} = x_{\texttt{kinc}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} y_{\texttt{kinc}},\ \ldots\}$
$\{y_{\texttt{mem}} \xhookrightarrow{y_\ell}_{\texttt{list}} [(0 : \texttt{i32} : y_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{mem}}..0)]\}$

J

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\mathrm{n}},\ \texttt{k} = x_{\mathrm{k}},\ \texttt{kinc} = x_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\mathrm{n}} > x_{\mathrm{k}},\ x_{\texttt{kinc}} = x_{\mathrm{k}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}},\ \ldots\}$
$\{x_{\texttt{mem}} \xhookrightarrow{x_\ell}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$

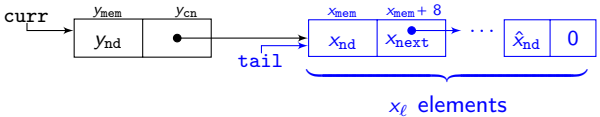
$\ldots$

K

$(\texttt{body}, 7)$
$\{\texttt{n} = x_{\mathrm{n}},\ \texttt{k} = x_{\texttt{kinc}},\ \texttt{curr} = y_{\texttt{mem}},\ \texttt{tail} = x_{\texttt{mem}},\ \texttt{curr\_next} = y_{\texttt{cn}},\ \ldots\}$
$\{[\![y_{\texttt{mem}},\ y_{\texttt{mem}}^{end}]\!],\ \ldots\}$
$\{y_{\texttt{cn}} = y_{\texttt{mem}} + 8,\ \ldots\}$
$\{y_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} y_{\texttt{nd}},\ \ldots\}$
$\{x_{\texttt{mem}} \xhookrightarrow{x_\ell}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$

L

$(\texttt{body}, 8)$
$\ldots$
$\{y_{\texttt{cn}} = y_{\texttt{mem}} + 8,\ y_\ell = x_\ell + 1,\ \ldots\}$
$\{\ldots\}$
$\{y_{\texttt{mem}} \xhookrightarrow{y_\ell}_{\texttt{list}} [(0 : \texttt{i32} : y_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{mem}}..0)]\}$



Modifying List Invariants

M

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{kinc}},\ \texttt{kinc} = y_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{kinc}},\ y_{\texttt{kinc}} = x_{\texttt{kinc}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} y_{\texttt{kinc}},\ \ldots\}$
$\{y_{\texttt{mem}} \overset{y_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : y_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{mem}}..0)]\}$

J

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{k}},\ \texttt{kinc} = x_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}},\ \ldots\}$
$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$

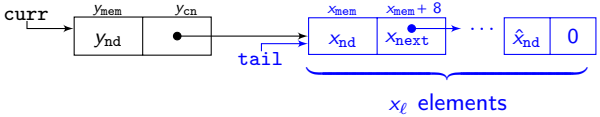
$\ldots$

K

$(\texttt{body}, 7)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{kinc}},\ \texttt{curr} = y_{\texttt{mem}},\ \texttt{tail} = x_{\texttt{mem}},\ \texttt{curr\_next} = y_{\texttt{cn}},\ \ldots\}$
$\{[\![y_{\texttt{mem}},\ y_{\texttt{mem}}^{end}]\!],\ \ldots\}$
$\{y_{\texttt{cn}} = y_{\texttt{mem}} + 8,\ \ldots\}$
$\{y_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} y_{\texttt{nd}},\ \ldots\}$
$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$

L

$(\texttt{body}, 8)$
$\ldots$
$\{y_{\texttt{cn}} = y_{\texttt{mem}} + 8,\ y_\ell = x_\ell + 1,\ \ldots\}$
$\{\ldots\}$
$\{y_{\texttt{mem}} \overset{y_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : y_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{mem}}..0)]\}$



Modifying List Invariants

M

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{kinc}},\ \texttt{kinc} = y_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{kinc}},\ y_{\texttt{kinc}} = x_{\texttt{kinc}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} y_{\texttt{kinc}},\ \ldots\}$
$\{y_{\texttt{mem}} \overset{y_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : y_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{mem}}..0)]\}$

J

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{k}},\ \texttt{kinc} = x_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}},\ \ldots\}$
$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$

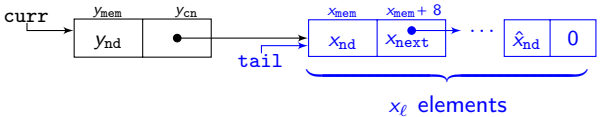
$\ldots$

K

$(\texttt{body}, 7)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{kinc}},\ \texttt{curr} = y_{\texttt{mem}},\ \texttt{tail} = x_{\texttt{mem}},\ \texttt{curr\_next} = y_{\texttt{cn}},\ \ldots\}$
$\{[\![y_{\texttt{mem}},\ y_{\texttt{mem}}^{end}]\!],\ \ldots\}$
$\{y_{\texttt{cn}} = y_{\texttt{mem}} + 8,\ \ldots\}$
$\{y_{\texttt{mem}} \hookrightarrow_{\texttt{i32}} y_{\texttt{nd}},\ \ldots\}$
$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$

L

$(\texttt{body}, 8)$
$\ldots$
$\{y_{\texttt{cn}} = y_{\texttt{mem}} + 8,\ y_\ell = x_\ell + 1,\ \ldots\}$
$\{\ldots\}$
$\{y_{\texttt{mem}} \overset{y_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : y_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{mem}}..0)]\}$



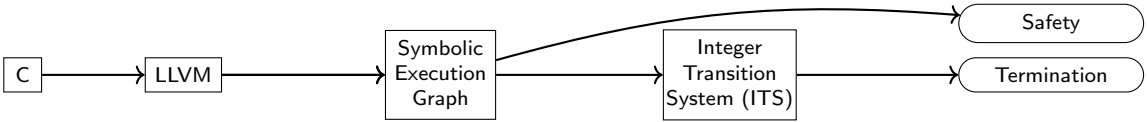
Safety

M

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{kinc}},\ \texttt{kinc} = y_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{kinc}},\ y_{\texttt{kinc}} = x_{\texttt{kinc}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} y_{\texttt{kinc}},\ \ldots\}$
$\{y_{\texttt{mem}} \xhookrightarrow{y_\ell}_{\texttt{list}} [(0 : \texttt{i32} : y_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{mem}}..0)]\}$

J

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{k}},\ \texttt{kinc} = x_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}},\ \ldots\}$
$\{x_{\texttt{mem}} \xhookrightarrow{x_\ell}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$



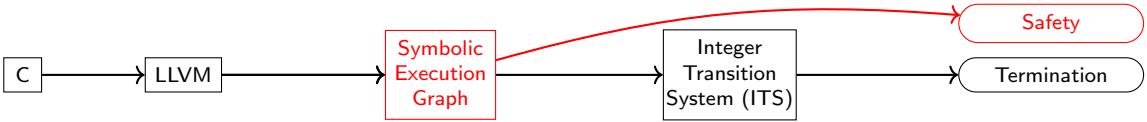
Safety

M

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{kinc}},\ \texttt{kinc} = y_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{kinc}},\ y_{\texttt{kinc}} = x_{\texttt{kinc}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} y_{\texttt{kinc}},\ \ldots\}$
$\{y_{\texttt{mem}} \xhookrightarrow{y_\ell}_{\texttt{list}} [(0 : \texttt{i32} : y_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{mem}}..0)]\}$

J

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{k}},\ \texttt{kinc} = x_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}},\ \ldots\}$
$\{x_{\texttt{mem}} \xhookrightarrow{x_\ell}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$



Safety

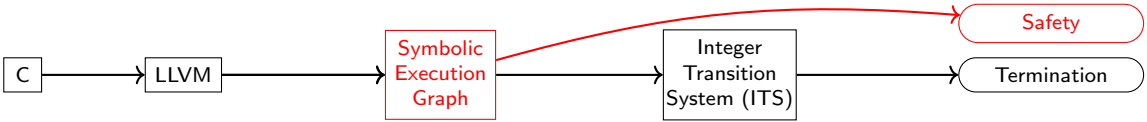
M

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{kinc}},\ \texttt{kinc} = y_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{kinc}},\ y_{\texttt{kinc}} = x_{\texttt{kinc}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} y_{\texttt{kinc}},\ \ldots\}$
$\{y_{\texttt{mem}} \xhookrightarrow{y_\ell}_{\texttt{list}} [(0 : \texttt{i32} : y_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{mem}}..0)]\}$

J

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{k}},\ \texttt{kinc} = x_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}},\ \ldots\}$
$\{x_{\texttt{mem}} \xhookrightarrow{x_\ell}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$

- Complete symbolic execution graph without *ERR*
  $\Longrightarrow$ Safety



Safety

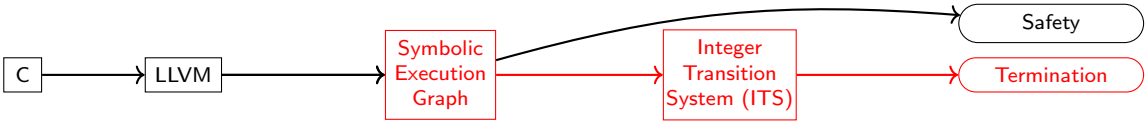
M

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{kinc}},\ \texttt{kinc} = y_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{kinc}},\ y_{\texttt{kinc}} = x_{\texttt{kinc}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} y_{\texttt{kinc}},\ \ldots\}$
$\{y_{\texttt{mem}} \xhookrightarrow{y_\ell}_{\texttt{list}} [(0 : \texttt{i32} : y_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{mem}}..0)]\}$

J

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{k}},\ \texttt{kinc} = x_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}},\ \ldots\}$
$\{x_{\texttt{mem}} \xhookrightarrow{x_\ell}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$

- Complete symbolic execution graph without *ERR*
  $\Longrightarrow$ Safety



Termination

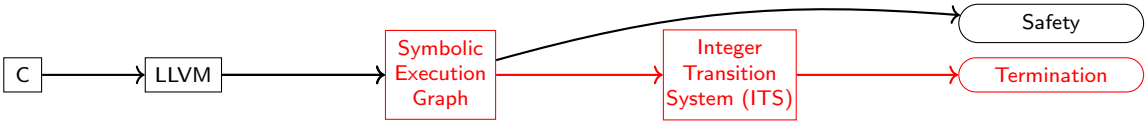
M

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{kinc}},\ \texttt{kinc} = y_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{kinc}},\ y_{\texttt{kinc}} = x_{\texttt{kinc}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} y_{\texttt{kinc}},\ \ldots\}$
$\{y_{\texttt{mem}} \overset{y_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : y_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{mem}}..0)]\}$

J

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{k}},\ \texttt{kinc} = x_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}},\ \ldots\}$
$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$

- Complete symbolic execution graph without *ERR*
  $\Longrightarrow$ Safety
- ITS from cycles of symbolic execution graph



Termination

M

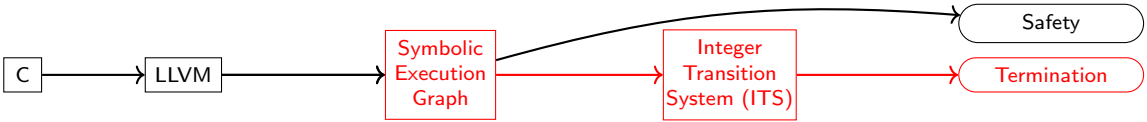
$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{kinc}},\ \texttt{kinc} = y_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{kinc}},\ y_{\texttt{kinc}} = x_{\texttt{kinc}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} y_{\texttt{kinc}},\ \ldots\}$
$\{y_{\texttt{mem}} \xhookrightarrow{y_\ell}_{\texttt{list}} [(0 : \texttt{i32} : y_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{mem}}..0)]\}$

J

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{k}},\ \texttt{kinc} = x_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}},\ \ldots\}$
$\{x_{\texttt{mem}} \xhookrightarrow{x_\ell}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$

ℓ_M ℓ_J

- Complete symbolic execution graph without *ERR*
  $\Longrightarrow$ Safety
- ITS from cycles of symbolic execution graph



Termination

M

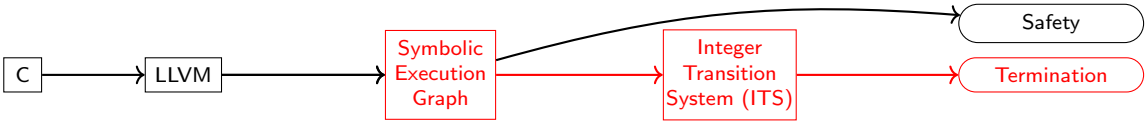
$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{kinc}},\ \texttt{kinc} = y_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{kinc}},\ y_{\texttt{kinc}} = x_{\texttt{kinc}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} y_{\texttt{kinc}},\ \ldots\}$
$\{y_{\texttt{mem}} \xhookrightarrow{y_\ell}_{\texttt{list}} [(0 : \texttt{i32} : y_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{mem}}..0)]\}$

J

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{k}},\ \texttt{kinc} = x_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}},\ \ldots\}$
$\{x_{\texttt{mem}} \xhookrightarrow{x_\ell}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$

$\ell_M \longleftrightarrow \ell_J$

- Complete symbolic execution graph without *ERR*
  $\Longrightarrow$ Safety
- ITS from cycles of symbolic execution graph



Termination

M

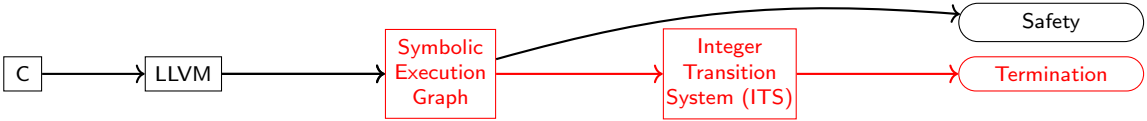
$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{kinc}},\ \texttt{kinc} = y_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{kinc}},\ y_{\texttt{kinc}} = x_{\texttt{kinc}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}},\ \ldots\}$
$\{y_{\texttt{mem}} \overset{y_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : y_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{mem}}..0)]\}$

J

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{k}},\ \texttt{kinc} = x_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}},\ \ldots\}$
$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$

$x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,$

- Complete symbolic execution graph without *ERR*
  $\Longrightarrow$ Safety
- ITS from cycles of symbolic execution graph



Termination

M

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{kinc}},\ \texttt{kinc} = y_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{kinc}},\ y_{\texttt{kinc}} = x_{\texttt{kinc}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} y_{\texttt{kinc}},\ \ldots\}$
$\{y_{\texttt{mem}} \xhookrightarrow{y_\ell}_{\texttt{list}} [(0 : \texttt{i32} : y_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{mem}}..0)]\}$

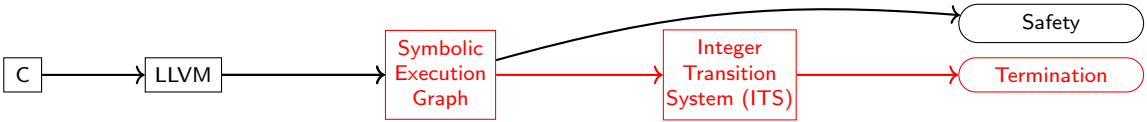
J

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{k}},\ \texttt{kinc} = x_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}},\ \ldots\}$
$\{x_{\texttt{mem}} \xhookrightarrow{x_\ell}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$

$\ell_M \longleftarrow \ell_J$

$x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ x'_{\texttt{n}} = x_{\texttt{n}},\ x'_{\texttt{k}} = x_{\texttt{k}},\ \ldots$

- Complete symbolic execution graph without *ERR*
  $\Longrightarrow$ Safety
- ITS from cycles of symbolic execution graph



Termination

M

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{kinc}},\ \texttt{kinc} = y_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{kinc}},\ y_{\texttt{kinc}} = x_{\texttt{kinc}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} y_{\texttt{kinc}},\ \ldots\}$
$\{y_{\texttt{mem}} \overset{y_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : y_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{mem}}..0)]\}$

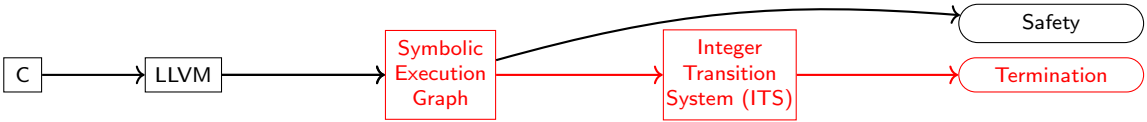
J

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{k}},\ \texttt{kinc} = x_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}},\ \ldots\}$
$\{x_{\texttt{mem}} \overset{x_\ell}{\hookrightarrow}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$

$\ell_M \longrightarrow \ell_J$

$x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ x'_{\texttt{n}} = x_{\texttt{n}},\ x'_{\texttt{k}} = x_{\texttt{k}},\ \ldots$

- Complete symbolic execution graph without *ERR*
  $\Longrightarrow$ Safety
- ITS from cycles of symbolic execution graph



Termination

M

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}}, \texttt{k} = x_{\texttt{kinc}}, \texttt{kinc} = y_{\texttt{kinc}}, \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{kinc}}, y_{\texttt{kinc}} = x_{\texttt{kinc}} + 1, \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}}, \ldots\}$
$\{y_{\texttt{mem}} \xhookrightarrow{y_\ell}_{\texttt{list}} [(0 : \texttt{i32} : y_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{mem}}..0)]\}$

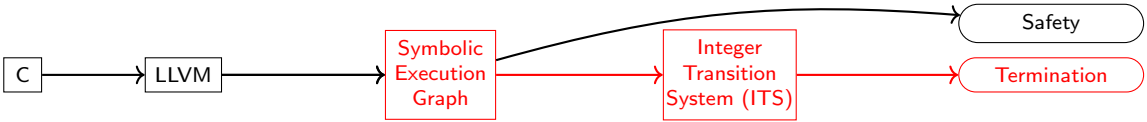
J

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}}, \texttt{k} = x_{\texttt{k}}, \texttt{kinc} = x_{\texttt{kinc}}, \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{k}}, x_{\texttt{kinc}} = x_{\texttt{k}} + 1, \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}}, \ldots\}$
$\{x_{\texttt{mem}} \xhookrightarrow{x_\ell}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$

ℓ_M $\xleftarrow{x'_{\texttt{n}} = x_{\texttt{n}},\ x'_{\texttt{k}} = x_{\texttt{kinc}},\ \ldots}$ ℓ_J

$x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ x'_{\texttt{n}} = x_{\texttt{n}},\ x'_{\texttt{k}} = x_{\texttt{k}},\ \ldots$

- Complete symbolic execution graph without *ERR*
  $\Longrightarrow$ Safety
- ITS from cycles of symbolic execution graph



Termination

M

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{kinc}},\ \texttt{kinc} = y_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{kinc}},\ y_{\texttt{kinc}} = x_{\texttt{kinc}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} y_{\texttt{kinc}},\ \ldots\}$
$\{y_{\texttt{mem}} \xhookrightarrow{y_\ell}_{\texttt{list}} [(0 : \texttt{i32} : y_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{mem}}..0)]\}$

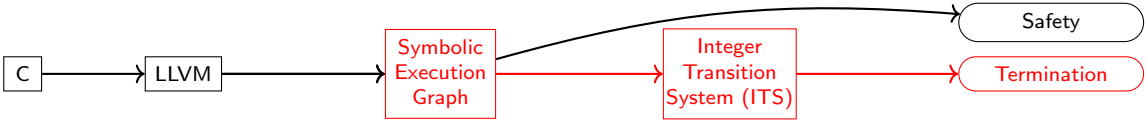
J

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{k}},\ \texttt{kinc} = x_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}},\ \ldots\}$
$\{x_{\texttt{mem}} \xhookrightarrow{x_\ell}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$

ℓ_M $\quad x'_{\texttt{n}} = x_{\texttt{n}},\ x'_{\texttt{k}} = x_{\texttt{kinc}},\ \ldots$ $\quad \ell_J$

$x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ x'_{\texttt{n}} = x_{\texttt{n}},\ x'_{\texttt{k}} = x_{\texttt{k}},\ \ldots$

- Complete symbolic execution graph without *ERR* $\Longrightarrow$ Safety
- ITS from cycles of symbolic execution graph
- ITS termination by existing tools & techniques



Termination

M

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{kinc}},\ \texttt{kinc} = y_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{kinc}},\ y_{\texttt{kinc}} = x_{\texttt{kinc}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} y_{\texttt{kinc}},\ \ldots\}$
$\{y_{\texttt{mem}} \xhookrightarrow{y_\ell}_{\texttt{list}} [(0 : \texttt{i32} : y_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{mem}}..0)]\}$

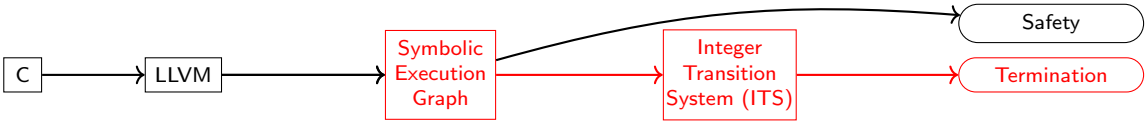
J

$(\texttt{cmp}, 0)$
$\{\texttt{n} = x_{\texttt{n}},\ \texttt{k} = x_{\texttt{k}},\ \texttt{kinc} = x_{\texttt{kinc}},\ \ldots\}$
$\{\ldots\}$
$\{x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ \ldots\}$
$\{x_{\texttt{k\_ad}} \hookrightarrow_{\texttt{i32}} x_{\texttt{kinc}},\ \ldots\}$
$\{x_{\texttt{mem}} \xhookrightarrow{x_\ell}_{\texttt{list}} [(0 : \texttt{i32} : x_{\texttt{nd}}..\hat{x}_{\texttt{nd}}), (8 : \texttt{list*} : x_{\texttt{next}}..0)]\}$

ℓ_M $\xleftrightarrow{x'_{\texttt{n}} = x_{\texttt{n}},\ x'_{\texttt{k}} = x_{\texttt{kinc}},\ \ldots}$ ℓ_J

$x_{\texttt{n}} > x_{\texttt{k}},\ x_{\texttt{kinc}} = x_{\texttt{k}} + 1,\ x'_{\texttt{n}} = x_{\texttt{n}},\ x'_{\texttt{k}} = x_{\texttt{k}},\ \ldots$

- Complete symbolic execution graph without *ERR*
  $\Longrightarrow$ Safety
- ITS from cycles of symbolic execution graph
- ITS termination by existing tools & techniques
  $\Longrightarrow$ LLVM/C program terminates



Proving Termination of C Programs with Lists

Inference and modification of list invariants (LI)

Proving Termination of C Programs with Lists

Inference and modification of list invariants (*LI*)

- *LI* abstracts from number of list elements

Proving Termination of C Programs with Lists

Inference and modification of list invariants (*LI*)

- *LI* abstracts from number of list elements
- *LI* preserves knowledge about
  some list contents and shape

Proving Termination of C Programs with Lists

Inference and modification of list invariants (*LI*)

- *LI* abstracts from number of list elements
- *LI* preserves knowledge about
  some list contents and shape
- *LI* contains information on
  memory arrangement of list fields

Proving Termination of C Programs with Lists

Inference and modification of list invariants (*LI*)

- *LI* abstracts from number of list elements
- *LI* preserves knowledge about some list contents and shape
- *LI* contains information on memory arrangement of list fields

```
int main() {
 ...
 for (unsigned int k = 0; k < n; k++) {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
  tail = curr;                          }
 struct list* ptr = tail;
 while(ptr != NULL) {
  ptr = *((struct list**)((void*)ptr +
        offsetof(struct list, next))); }}
```

Proving Termination of C Programs with Lists

Inference and modification of list invariants (*LI*)

- *LI* abstracts from number of list elements
- *LI* preserves knowledge about some list contents and shape
- *LI* contains information on memory arrangement of list fields
- *LI* needed for memory safety & termination

```
int main() {
 ...
 for (unsigned int k = 0; k < n; k++) {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
  tail = curr;                          }
 struct list* ptr = tail;
 while(ptr != NULL) {
  ptr = *((struct list**)((void*)ptr +
        offsetof(struct list, next))); }}
```

Proving Termination of C Programs with Lists

Inference and modification of list invariants (*LI*)

- *LI* abstracts from number of list elements
- *LI* preserves knowledge about some list contents and shape
- *LI* contains information on memory arrangement of list fields
- *LI* needed for memory safety & termination
  $\Longrightarrow$ ITS contains symbolic variables for list length and values of list fields

```
int main() {
 ...
 for (unsigned int k = 0; k < n; k++) {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
  tail = curr;                          }
 struct list* ptr = tail;
 while(ptr != NULL) {
  ptr = *((struct list**)((void*)ptr +
        offsetof(struct list, next))); }}
```

Proving Termination of C Programs with Lists

Inference and modification of list invariants (*LI*)

- *LI* abstracts from number of list elements
- *LI* preserves knowledge about some list contents and shape
- *LI* contains information on memory arrangement of list fields
- *LI* needed for memory safety & termination
  $\Longrightarrow$ ITS contains symbolic variables for list length and values of list fields

```
int main() {
 ...
 for (unsigned int k = 0; k < n; k++) {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
  tail = curr;                          }
 struct list* ptr = tail;
 while(ptr != NULL) {
  ptr = *((struct list**)((void*)ptr +
        offsetof(struct list, next))); }}
```

Implementation in AProVE

Proving Termination of C Programs with Lists

Inference and modification of list invariants (*LI*)

- *LI* abstracts from number of list elements
- *LI* preserves knowledge about some list contents and shape
- *LI* contains information on memory arrangement of list fields
- *LI* needed for memory safety & termination
  $\Longrightarrow$ ITS contains symbolic variables for list length and values of list fields

```
int main() {
 ...
 for (unsigned int k = 0; k < n; k++) {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
  tail = curr;                          }
 struct list* ptr = tail;
 while(ptr != NULL) {
  ptr = *((struct list**)((void*)ptr +
        offsetof(struct list, next))); }}
```

Implementation in AProVE

18 list programs in *SV-COMP* '22
18 list programs in *TermComp* '22

Proving Termination of C Programs with Lists

Inference and modification of list invariants (*LI*)

- *LI* abstracts from number of list elements
- *LI* preserves knowledge about some list contents and shape
- *LI* contains information on memory arrangement of list fields
- *LI* needed for memory safety & termination
  $\Longrightarrow$ ITS contains symbolic variables for list length and values of list fields

```
int main() {
 ...
 for (unsigned int k = 0; k < n; k++) {
  curr = malloc(sizeof(struct list));
  curr->value = nondet_uint();
  curr->next = tail;
  tail = curr;                          }
 struct list* ptr = tail;
 while(ptr != NULL) {
  ptr = *((struct list**)((void*)ptr +
        offsetof(struct list, next))); }}
```

Implementation in AProVE

18 list programs in *SV-COMP* '22
18 list programs in *TermComp* '22

	SV-C Term.	SV-C Non-T.	TermComp Term.
AProVE	7 (of 9)	5 (of 9)	17 (of 18)
UAutomizer	2 (of 9)	7 (of 9)	1 (of 18)