Modular I/O Reasoning in DimSum

Alex Loitzl[1]

[1]Institute of Science and Technology Austria (ISTA)

March, 2025

Modular I/O Reasoning

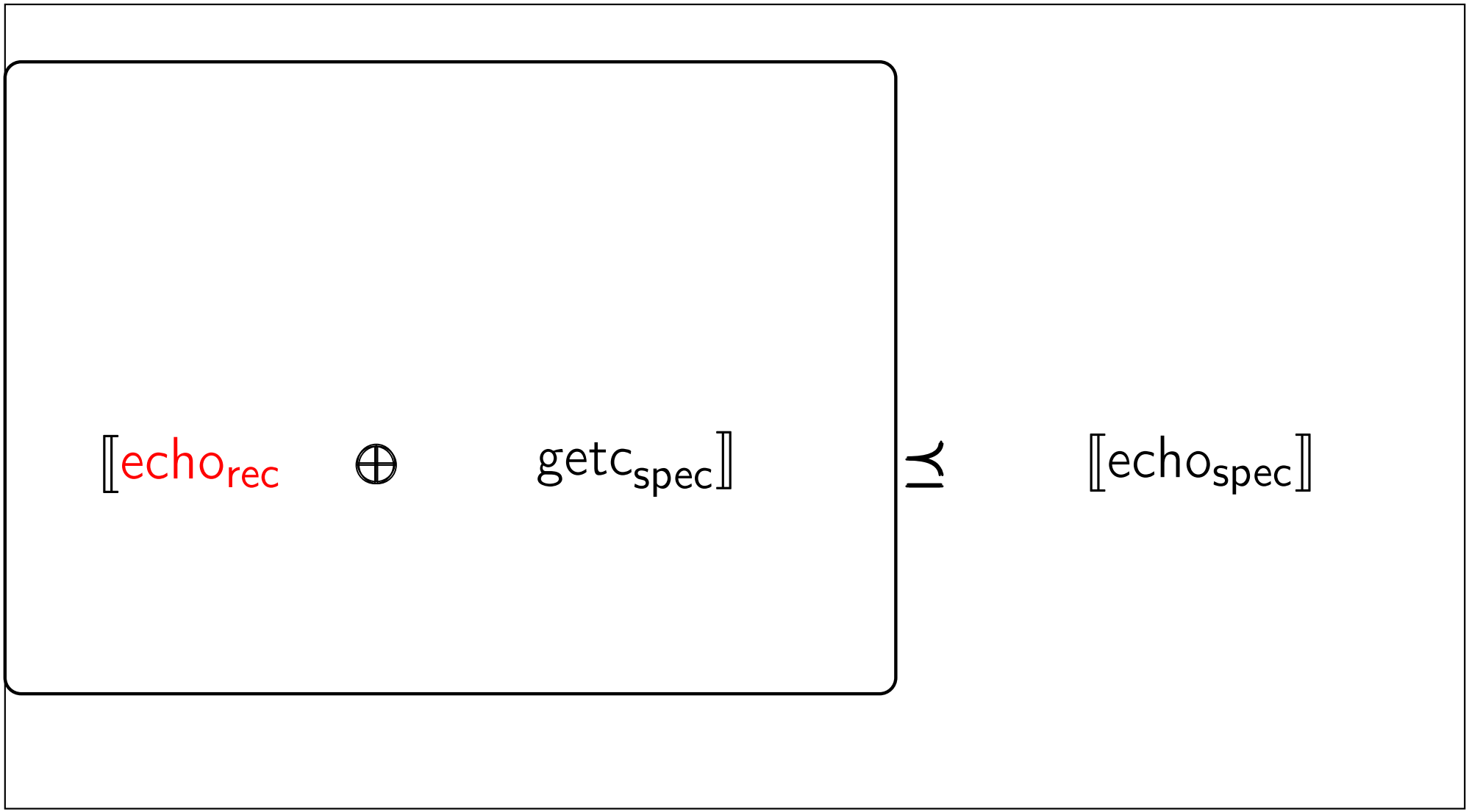
Multi-language Reasoning in DimSum

Rotation Project

Summary

- Formally verified compiler
  - Proof covers all optimizations
  - Correct w.r.t. the modeled semantics
- Discrepancies between hardware and model
  - Cannot implement correct calling conventions
  - Cannot support TriCore architecture
- Suboptimal code generation
  - Inserted moves
  - Higher register pressure

Example


$[\![\mathsf{echo}_{\mathsf{rec}} \oplus \mathsf{getc}_{\mathsf{spec}}]\!]$
$\preceq$
$[\![\mathsf{echo}_{\mathsf{spec}}]\!]$

Example

```
int echo () :=
  let c := getc();
  putc(c);
  return 0;
```

$\oplus_0$

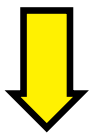
```
getc_spec :=
  Spec.forever(
  TExists '(f, vs, h);
  TVis (In, Call f vs h);;
  TAssume (f = "getc");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TVis (Out, Return v h)).
```

$\prec_0$

```
echo_getc_spec :=
  TExists '(f, vs, h);
  Tvis (In, Call f vs h);;
  TAssume (f = "echo");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TCallRet "putc" [v] h;
  TVis (Out, Return 0 h);;
  TUb.
```

Example

(Call f vs h)



```
int echo () :=
  let c := getc();
  putc(c);
  return 0;
```

⊕

```
getc_spec :=
  Spec.forever(
  TExists '(f, vs, h);
  TVis (In, Call f vs h);;
  TAssume (f = "getc");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TVis (Out, Return v h)).
```

0

≻

```
echo_getc_spec :=
  TExists '(f, vs, h);
  Tvis (In, Call f vs h);;
  TAssume (f = "echo");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TCallRet "putc" [v] h;
  TVis (Out, Return 0 h);;
  TUb.
```

0

Example

Call f vs h

```
int echo () :=
  let c := getc();
  putc(c);
  return 0;
```

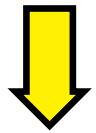
⊕

```
getc_spec :=
  Spec.forever(
  TExists '(f, vs, h);
  TVis (In, Call f vs h);;
  TAssume (f = "getc");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TVis (Out, Return v h)).
```

0

≻

(Call f vs h)



```
echo_getc_spec :=
  TExists '(f, vs, h);
  Tvis (In, Call f vs h);;
  TAssume (f = "echo");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TCallRet "putc" [v] h;
  TVis (Out, Return 0 h);;
  TUb.
```

0

Example

Call f vs h

```
int echo () :=
  let c := getc();
  putc(c);
  return 0;
```

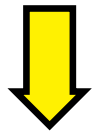
⊕

```
getc_spec :=
  Spec.forever(
  TExists '(f, vs, h);
  TVis (In, Call f vs h);;
  TAssume (f = "getc");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TVis (Out, Return v h)).
```

0

≻

(Call f vs h)



```
echo_getc_spec :=
  TExists '(f, vs, h);
  Tvis (In, Call f vs h);;
  TAssume (f = "echo");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TCallRet "putc" [v] h;
  TVis (Out, Return 0 h);;
  TUb.
```

0

Example

Call f vs h

```
int echo () :=
  let c := getc();
  putc(c);
  return 0;
```

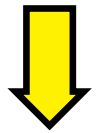
⊕

```
getc_spec :=
  Spec.forever(
  TExists '(f, vs, h);
  TVis (In, Call f vs h);;
  TAssume (f = "getc");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TVis (Out, Return v h)).
```

0

≻

(Call f vs h)



```
echo_getc_spec :=
  TExists '(f, vs, h);
  Tvis (In, Call f vs h);;
  TAssume (f = "echo");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TCallRet "putc" [v] h;
  TVis (Out, Return 0 h);;
  TUb.
```

0

Example

(Call "echo" vs h)

```
int echo () :=
  let c := getc();
  putc(c);
  return 0;
```

⊕

```
getc_spec :=
  Spec.forever(
  TExists '(f, vs, h);
  TVis (In, Call f vs h);;
  TAssume (f = "getc");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TVis (Out, Return v h)).
```

0

↾

```
echo_getc_spec :=
  TExists '(f, vs, h);
  Tvis (In, Call f vs h);;
  TAssume (f = "echo");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TCallRet "putc" [v] h;
  TVis (Out, Return 0 h);;
  TUb.
```

0

Example

```
(Call "echo" [] h)
```

```
int echo () :=
  let c := getc();
  putc(c);
  return 0;
```

⊕

```
getc_spec :=
  Spec.forever(
  TExists '(f, vs, h);
  TVis (In, Call f vs h);;
  TAssume (f = "getc");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TVis (Out, Return v h)).
```

0 ⊸

```
echo_getc_spec :=
  TExists '(f, vs, h);
  Tvis (In, Call f vs h);;
  TAssume (f = "echo");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TCallRet "putc" [v] h;
  TVis (Out, Return 0 h);;
  TUb.
```

0

Example

```
int echo () :=
  let c := getc();
  putc(c);
  return 0;
```

⊕

```
getc_spec :=
  Spec.forever(
  TExists '(f, vs, h);
  TVis (In, Call f vs h);;
  TAssume (f = "getc");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TVis (Out, Return v h)).
```

0

≻

```
echo_getc_spec :=
  TExists '(f, vs, h);
  Tvis (In, Call f vs h);;
  TAssume (f = "echo");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TCallRet "putc" [v] h;
  TVis (Out, Return 0 h);;
  TUb.
```

0

Example

```
int echo () :=
  let c := getc();
  putc(c);
  return 0;
```

⊕

```
getc_spec :=
  Spec.forever(
  TExists '(f, vs, h);
  TVis (In, Call f vs h);;
  TAssume (f = "getc");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TVis (Out, Return v h)).
```

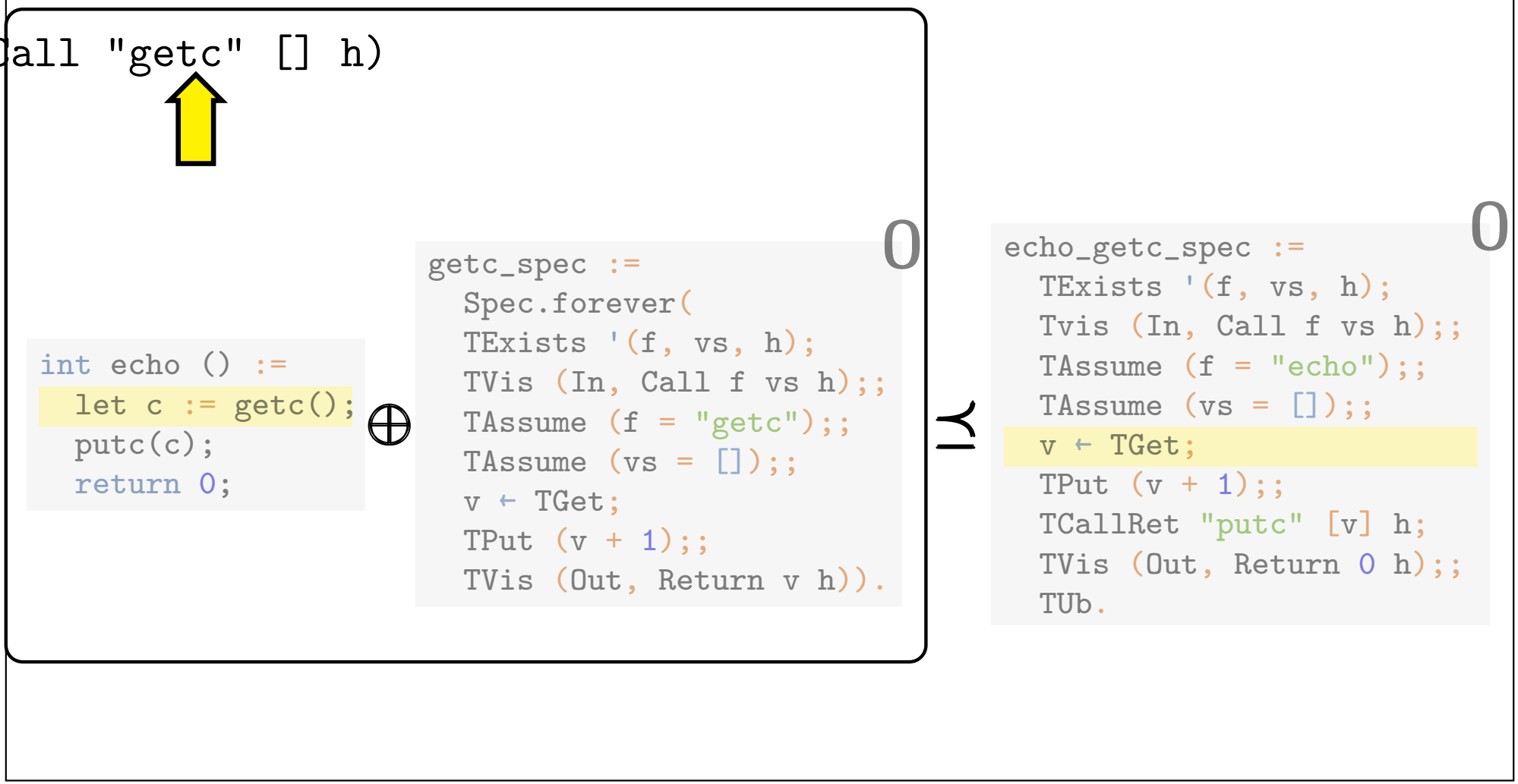
0

≻

```
echo_getc_spec :=
  TExists '(f, vs, h);
  Tvis (In, Call f vs h);;
  TAssume (f = "echo");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TCallRet "putc" [v] h;
  TVis (Out, Return 0 h);;
  TUb.
```

0

Example

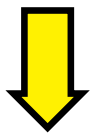

all "getc" [] h)
int echo () :=
let c := getc();
putc(c);
return 0;
⊕
getc_spec :=
Spec.forever(
TExists '(f, vs, h);
TVis (In, Call f vs h);;
TAssume (f = "getc");;
TAssume (vs = []);;
v ← TGet;
TPut (v + 1);;
TVis (Out, Return v h)).
0
≻
echo_getc_spec :=
TExists '(f, vs, h);
Tvis (In, Call f vs h);;
TAssume (f = "echo");;
TAssume (vs = []);;
v ← TGet;
TPut (v + 1);;
TCallRet "putc" [v] h;
TVis (Out, Return 0 h);;
TUb.
0

Example

```
int echo () :=
  let c := getc();
  putc(c);
  return 0;
```

⊕

(Call "getc" [] h)



```
getc_spec :=
  Spec.forever(
  TExists '(f, vs, h);
  TVis (In, Call f vs h);;
  TAssume (f = "getc");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TVis (Out, Return v h)).
```

0

⊸

```
echo_getc_spec :=
  TExists '(f, vs, h);
  Tvis (In, Call f vs h);;
  TAssume (f = "echo");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TCallRet "putc" [v] h;
  TVis (Out, Return 0 h);;
  TUb.
```

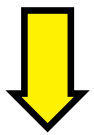
0

Example

```
int echo () :=
  let c := getc();
  putc(c);
  return 0;
```

⊕

```
(Call "getc" [] h)
```



```
getc_spec :=
  Spec.forever(
  TExists '(f, vs, h);
  TVis (In, Call f vs h);;
  TAssume (f = "getc");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TVis (Out, Return v h)).
```

0

⊸

```
echo_getc_spec :=
  TExists '(f, vs, h);
  Tvis (In, Call f vs h);;
  TAssume (f = "echo");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TCallRet "putc" [v] h;
  TVis (Out, Return 0 h);;
  TUb.
```

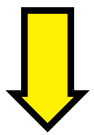
0

Example

```
int echo () :=
  let c := getc();
  putc(c);
  return 0;
```

⊕

```
(Call "getc" [] h)
```



```
getc_spec :=
  Spec.forever(
  TExists '(f, vs, h);
  TVis (In, Call f vs h);;
  TAssume (f = "getc");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TVis (Out, Return v h)).
```

0

⊸

```
echo_getc_spec :=
  TExists '(f, vs, h);
  Tvis (In, Call f vs h);;
  TAssume (f = "echo");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TCallRet "putc" [v] h;
  TVis (Out, Return 0 h);;
  TUb.
```

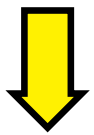
0

Example

```
int echo () :=
  let c := getc();
  putc(c);
  return 0;
```

⊕

(Call "getc" [] h)



0

```
getc_spec :=
  Spec.forever(
  TExists '(f, vs, h);
  TVis (In, Call f vs h);;
  TAssume (f = "getc");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TVis (Out, Return v h)).
```

⊸

```
echo_getc_spec :=
  TExists '(f, vs, h);
  Tvis (In, Call f vs h);;
  TAssume (f = "echo");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TCallRet "putc" [v] h;
  TVis (Out, Return 0 h);;
  TUb.
```

0

Example

```
(Call "getc" [] h)
```

```
int echo () :=
  let c := getc();
  putc(c);
  return 0;
```

⊕

```
getc_spec :=
  Spec.forever(
  TExists '(f, vs, h);
  TVis (In, Call f vs h);;
  TAssume (f = "getc");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TVis (Out, Return v h)).
```

0

⊸

```
echo_getc_spec :=
  TExists '(f, vs, h);
  Tvis (In, Call f vs h);;
  TAssume (f = "echo");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TCallRet "putc" [v] h;
  TVis (Out, Return 0 h);;
  TUb.
```

0

Example

```
int echo () :=
  let c := getc();
  putc(c);
  return 0;
```

⊕

```
(Call "getc" [] h)
```

0

```
getc_spec :=
  Spec.forever(
  TExists '(f, vs, h);
  TVis (In, Call f vs h);;
  TAssume (f = "getc");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TVis (Out, Return v h)).
```

⊸

```
echo_getc_spec :=
  TExists '(f, vs, h);
  Tvis (In, Call f vs h);;
  TAssume (f = "echo");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TCallRet "putc" [v] h;
  TVis (Out, Return 0 h);;
  TUb.
```

0

Example

```
int echo () :=
  let c := getc();
  putc(c);
  return 0;
```

⊕

```
getc_spec :=
  Spec.forever(
  TExists '(f, vs, h);
  TVis (In, Call f vs h);;
  TAssume (f = "getc");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TVis (Out, Return v h)).
```

0

≻

```
echo_getc_spec :=
  TExists '(f, vs, h);
  Tvis (In, Call f vs h);;
  TAssume (f = "echo");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TCallRet "putc" [v] h;
  TVis (Out, Return 0 h);;
  TUb.
```

0

Example

```
int echo () :=
  let c := getc();
  putc(c);
  return 0;
```

⊕

```
getc_spec :=
  Spec.forever(
  TExists '(f, vs, h);
  TVis (In, Call f vs h);;
  TAssume (f = "getc");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TVis (Out, Return v h)).
```

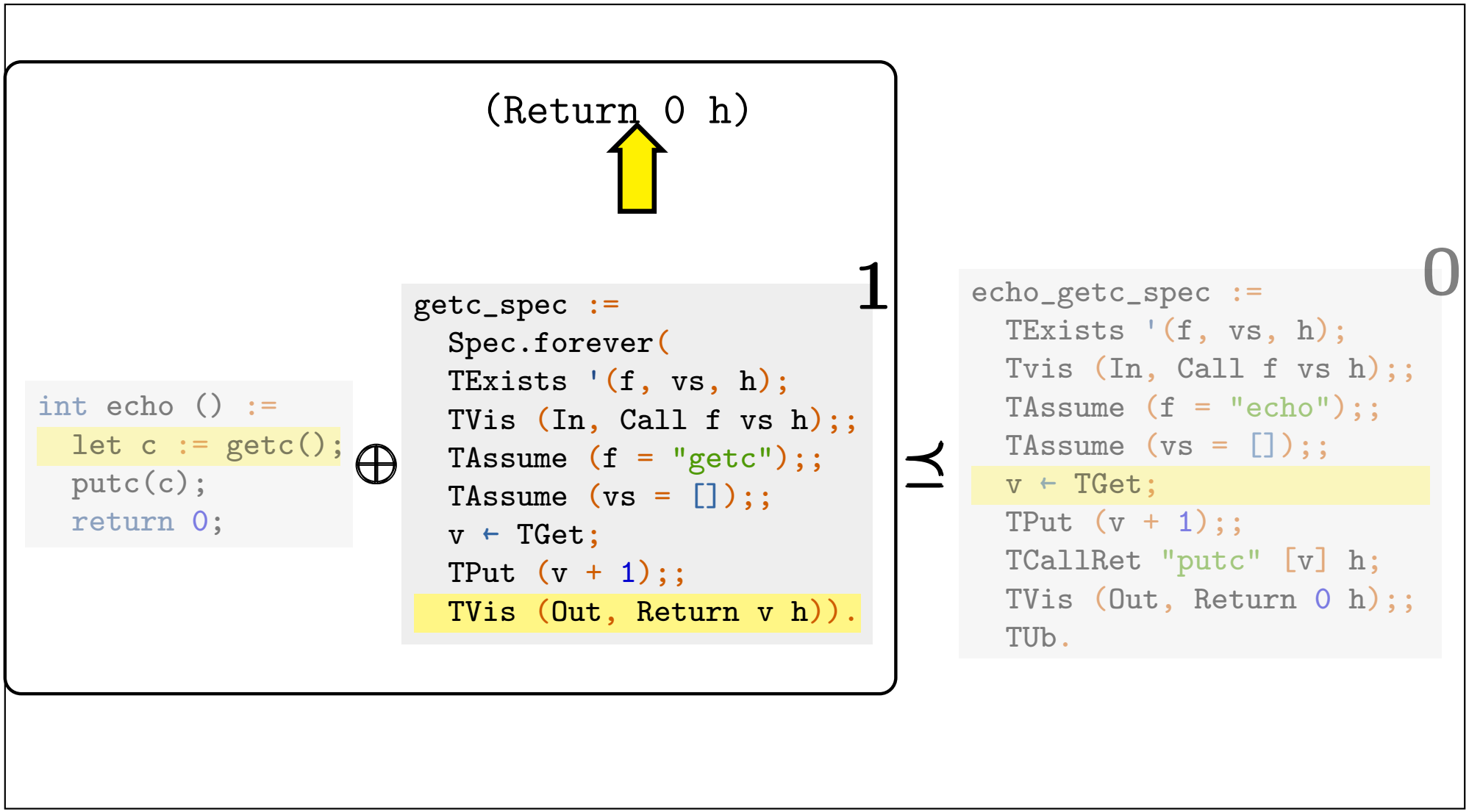
1

≺1

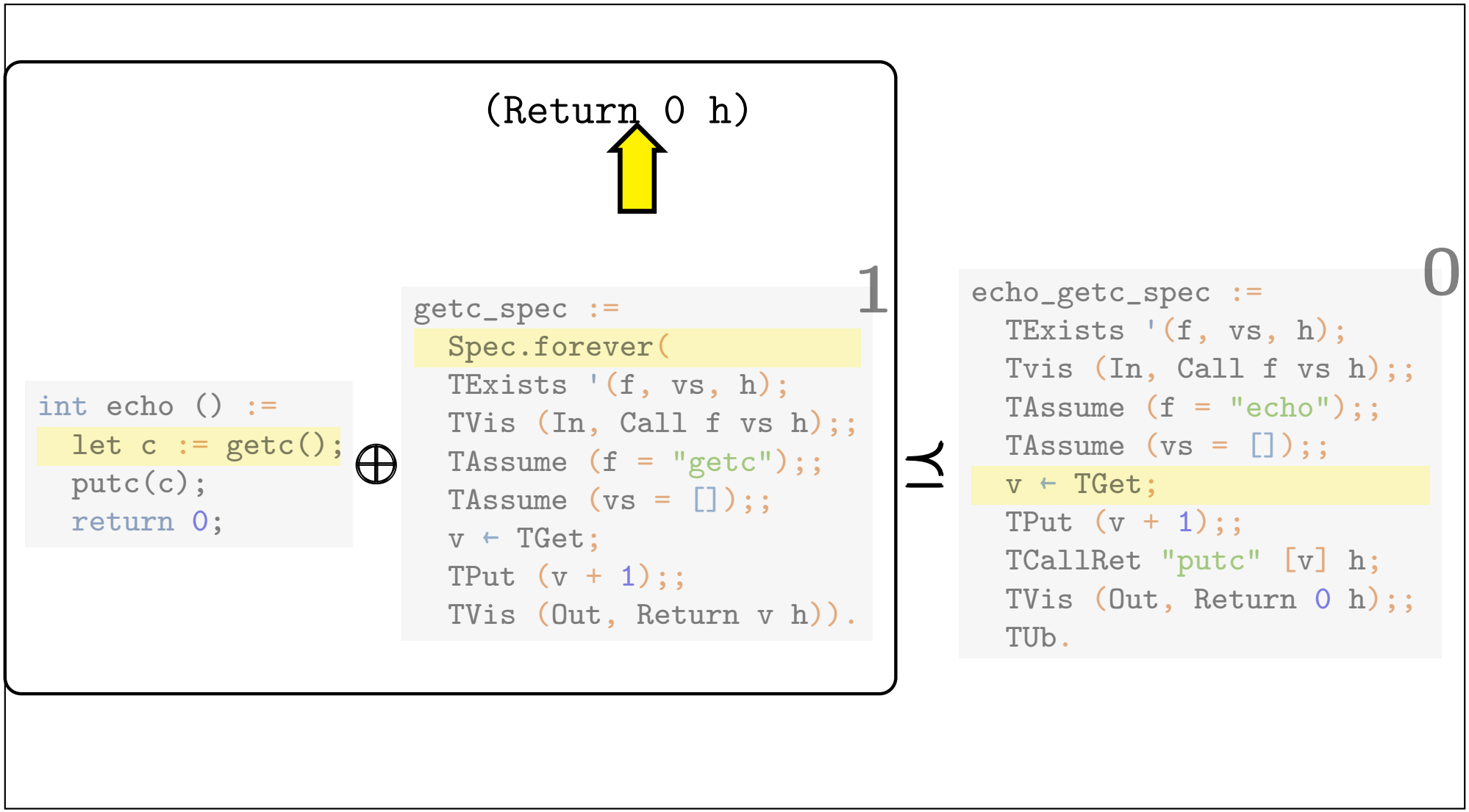
```
echo_getc_spec :=
  TExists '(f, vs, h);
  Tvis (In, Call f vs h);;
  TAssume (f = "echo");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TCallRet "putc" [v] h;
  TVis (Out, Return 0 h);;
  TUb.
```

0

Example

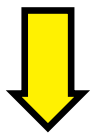

(Return 0 h)
1
0
int echo () :=
let c := getc();
putc(c);
return 0;
getc_spec :=
Spec.forever(
TExists '(f, vs, h);
TVis (In, Call f vs h);;
TAssume (f = "getc");;
TAssume (vs = []);;
v ← TGet;
TPut (v + 1);;
TVis (Out, Return v h)).
echo_getc_spec :=
TExists '(f, vs, h);
Tvis (In, Call f vs h);;
TAssume (f = "echo");;
TAssume (vs = []);;
v ← TGet;
TPut (v + 1);;
TCallRet "putc" [v] h;
TVis (Out, Return 0 h);;
TUb.

Example


(Return 0 h)
int echo () :=
let c := getc();
putc(c);
return 0;
getc_spec :=
Spec.forever(
TExists '(f, vs, h);
TVis (In, Call f vs h);;
TAssume (f = "getc");;
TAssume (vs = []);;
v ← TGet;
TPut (v + 1);;
TVis (Out, Return v h)).
1
echo_getc_spec :=
TExists '(f, vs, h);
Tvis (In, Call f vs h);;
TAssume (f = "echo");;
TAssume (vs = []);;
v ← TGet;
TPut (v + 1);;
TCallRet "putc" [v] h;
TVis (Out, Return 0 h);;
TUb.
0

Example

(Return 0 h)



```
int echo () :=
  let c := getc();
  putc(c);
  return 0;
```

⊕

```
getc_spec :=
  Spec.forever(
  TExists '(f, vs, h);
  TVis (In, Call f vs h);;
  TAssume (f = "getc");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TVis (Out, Return v h)).
```

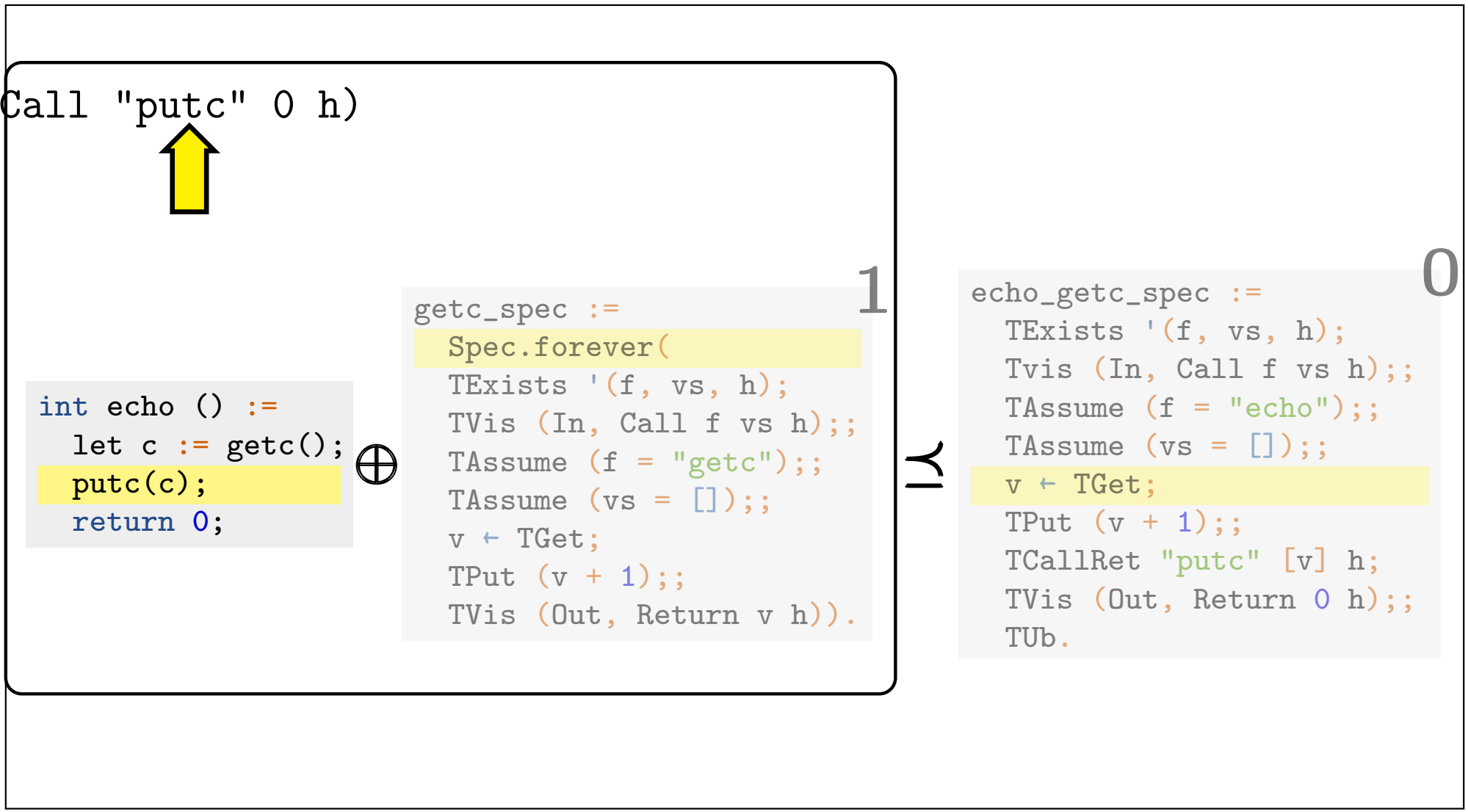
1

≺1

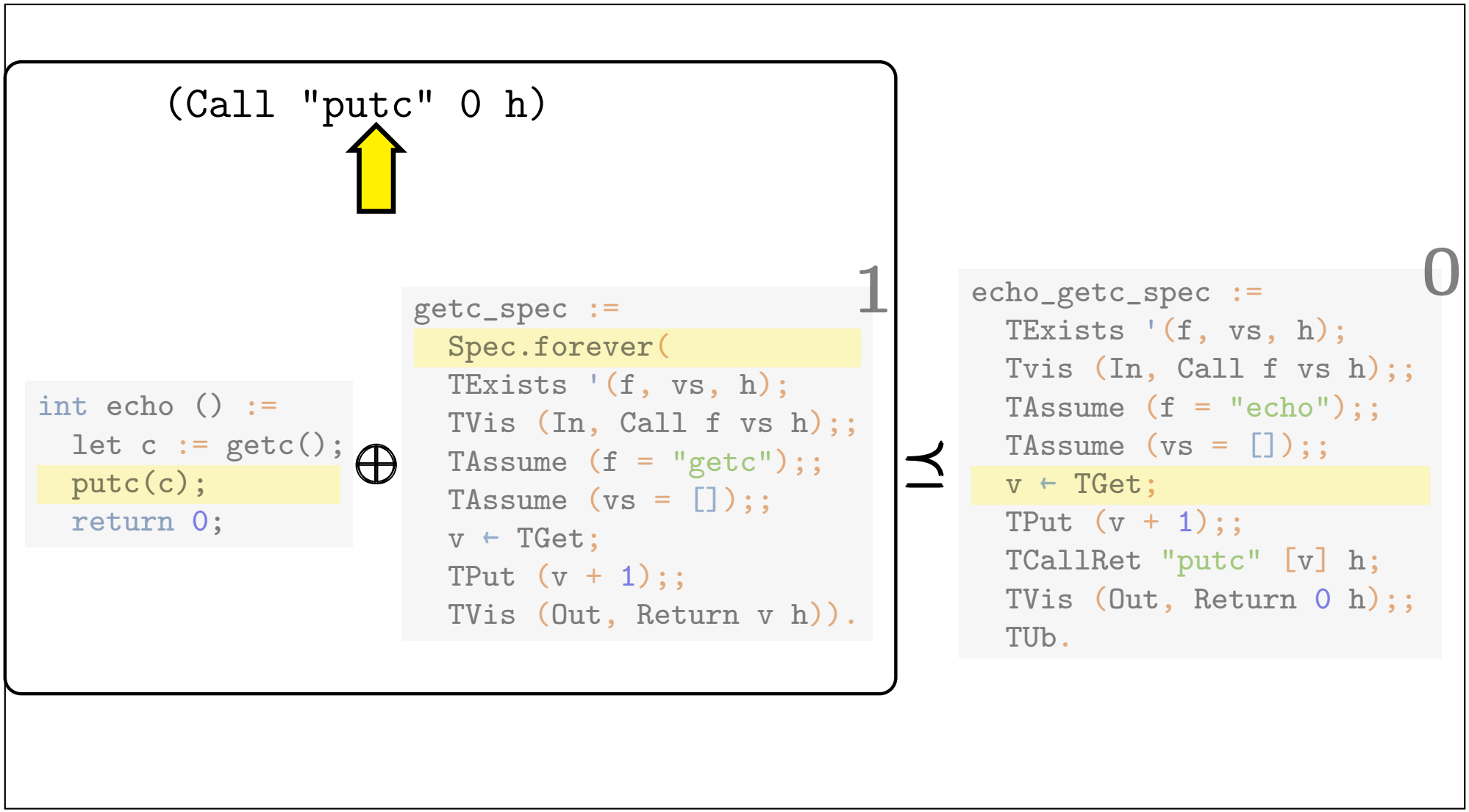
```
echo_getc_spec :=
  TExists '(f, vs, h);
  Tvis (In, Call f vs h);;
  TAssume (f = "echo");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TCallRet "putc" [v] h;
  TVis (Out, Return 0 h);;
  TUb.
```

0

Example


Call "putc" 0 h)
int echo () :=
let c := getc();
putc(c);
return 0;
getc_spec :=
Spec.forever(
TExists '(f, vs, h);
TVis (In, Call f vs h);;
TAssume (f = "getc");;
TAssume (vs = []);;
v ← TGet;
TPut (v + 1);;
TVis (Out, Return v h)).
1
echo_getc_spec :=
TExists '(f, vs, h);
Tvis (In, Call f vs h);;
TAssume (f = "echo");;
TAssume (vs = []);;
v ← TGet;
TPut (v + 1);;
TCallRet "putc" [v] h;
TVis (Out, Return 0 h);;
TUb.
0

Example


(Call "putc" 0 h)
int echo () :=
let c := getc();
putc(c);
return 0;
getc_spec :=
Spec.forever(
TExists '(f, vs, h);
TVis (In, Call f vs h);;
TAssume (f = "getc");;
TAssume (vs = []);;
v ← TGet;
TPut (v + 1);;
TVis (Out, Return v h)).
1
echo_getc_spec :=
TExists '(f, vs, h);
Tvis (In, Call f vs h);;
TAssume (f = "echo");;
TAssume (vs = []);;
v ← TGet;
TPut (v + 1);;
TCallRet "putc" [v] h;
TVis (Out, Return 0 h);;
TUb.
0

Example

```
int echo () :=
  let c := getc();
  putc(c);
  return 0;
```

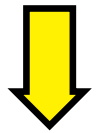
⊕

```
getc_spec :=
  Spec.forever(
  TExists '(f, vs, h);
  TVis (In, Call f vs h);;
  TAssume (f = "getc");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TVis (Out, Return v h)).
```

1

≻1

```
(Call "putc" 0 h)
```



```
echo_getc_spec :=
  TExists '(f, vs, h);
  Tvis (In, Call f vs h);;
  TAssume (f = "echo");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TCallRet "putc" [v] h;
  TVis (Out, Return 0 h);;
  TUb.
```

0

Example

```
int echo () :=
  let c := getc();
  putc(c);
  return 0;
```

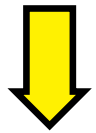
⊕

```
getc_spec :=
  Spec.forever(
  TExists '(f, vs, h);
  TVis (In, Call f vs h);;
  TAssume (f = "getc");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TVis (Out, Return v h)).
```

1

≻1

```
(Call "putc" 0 h)
```



```
echo_getc_spec :=
  TExists '(f, vs, h);
  Tvis (In, Call f vs h);;
  TAssume (f = "echo");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TCallRet "putc" [v] h;
  TVis (Out, Return 0 h);;
  TUb.
```

0

Example

```
int echo () :=
  let c := getc();
  putc(c);
  return 0;
```

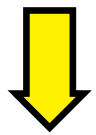
⊕

```
getc_spec :=
  Spec.forever(
  TExists '(f, vs, h);
  TVis (In, Call f vs h);;
  TAssume (f = "getc");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TVis (Out, Return v h)).
```

1

≺1

(Call "putc" 0 h)



```
echo_getc_spec :=
  TExists '(f, vs, h);
  Tvis (In, Call f vs h);;
  TAssume (f = "echo");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TCallRet "putc" [v] h;
  TVis (Out, Return 0 h);;
  TUb.
```

1

Example

```
int echo () :=
  let c := getc();
  putc(c);
  return 0;
```

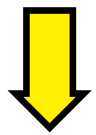
⊕

```
getc_spec :=
  Spec.forever(
  TExists '(f, vs, h);
  TVis (In, Call f vs h);;
  TAssume (f = "getc");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TVis (Out, Return v h)).
```

1

≺1

(Call "putc" 0 h)



```
echo_getc_spec :=
  TExists '(f, vs, h);
  Tvis (In, Call f vs h);;
  TAssume (f = "echo");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TCallRet "putc" [v] h;
  TVis (Out, Return 0 h);;
  TUb.
```

1

Example

```
int echo () :=
  let c := getc();
  putc(c);
  return 0;
```

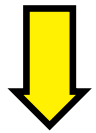
$\oplus$

```
getc_spec :=
  Spec.forever(
  TExists '(f, vs, h);
  TVis (In, Call f vs h);;
  TAssume (f = "getc");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TVis (Out, Return v h)).
```

1

$\succeq_1$

(Call "putc" 0 h)



```
echo_getc_spec :=
  TExists '(f, vs, h);
  Tvis (In, Call f vs h);;
  TAssume (f = "echo");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TCallRet "putc" [v] h;
  TVis (Out, Return 0 h);;
  TUb.
```

1

Example

```
int echo () :=
  let c := getc();
  putc(c);
  return 0;
```

$\oplus$

```
getc_spec :=
  Spec.forever(
  TExists '(f, vs, h);
  TVis (In, Call f vs h);;
  TAssume (f = "getc");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TVis (Out, Return v h)).
```

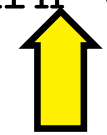
1

$\preceq_{\Sigma}$

```
echo_getc_spec :=
  TExists '(f, vs, h);
  Tvis (In, Call f vs h);;
  TAssume (f = "echo");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TCallRet "putc" [v] h;
  TVis (Out, Return 0 h);;
  TUb.
```

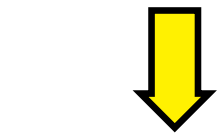
1

(Return v h)



Example

(Return v h)



```
int echo () :=
  let c := getc();
  putc(c);
  return 0;
```

$\oplus$

```
getc_spec :=
  Spec.forever(
  TExists '(f, vs, h);
  TVis (In, Call f vs h);;
  TAssume (f = "getc");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TVis (Out, Return v h)).
```

1

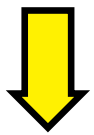
$\preceq$

```
echo_getc_spec :=
  TExists '(f, vs, h);
  Tvis (In, Call f vs h);;
  TAssume (f = "echo");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TCallRet "putc" [v] h;
  TVis (Out, Return 0 h);;
  TUb.
```

1

Example

(Return v h)



```
int echo () :=
  let c := getc();
  putc(c);
  return 0;
```

⊕

```
getc_spec :=
  Spec.forever(
  TExists '(f, vs, h);
  TVis (In, Call f vs h);;
  TAssume (f = "getc");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TVis (Out, Return v h)).
```

1

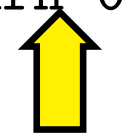
≺

```
echo_getc_spec :=
  TExists '(f, vs, h);
  Tvis (In, Call f vs h);;
  TAssume (f = "echo");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TCallRet "putc" [v] h;
  TVis (Out, Return 0 h);;
  TUb.
```

1

Example

(Return 0 h)



```
int echo () :=
  let c := getc();
  putc(c);
  return 0;
```

$\oplus$

```
getc_spec :=
  Spec.forever(
  TExists '(f, vs, h);
  TVis (In, Call f vs h);;
  TAssume (f = "getc");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TVis (Out, Return v h)).
```

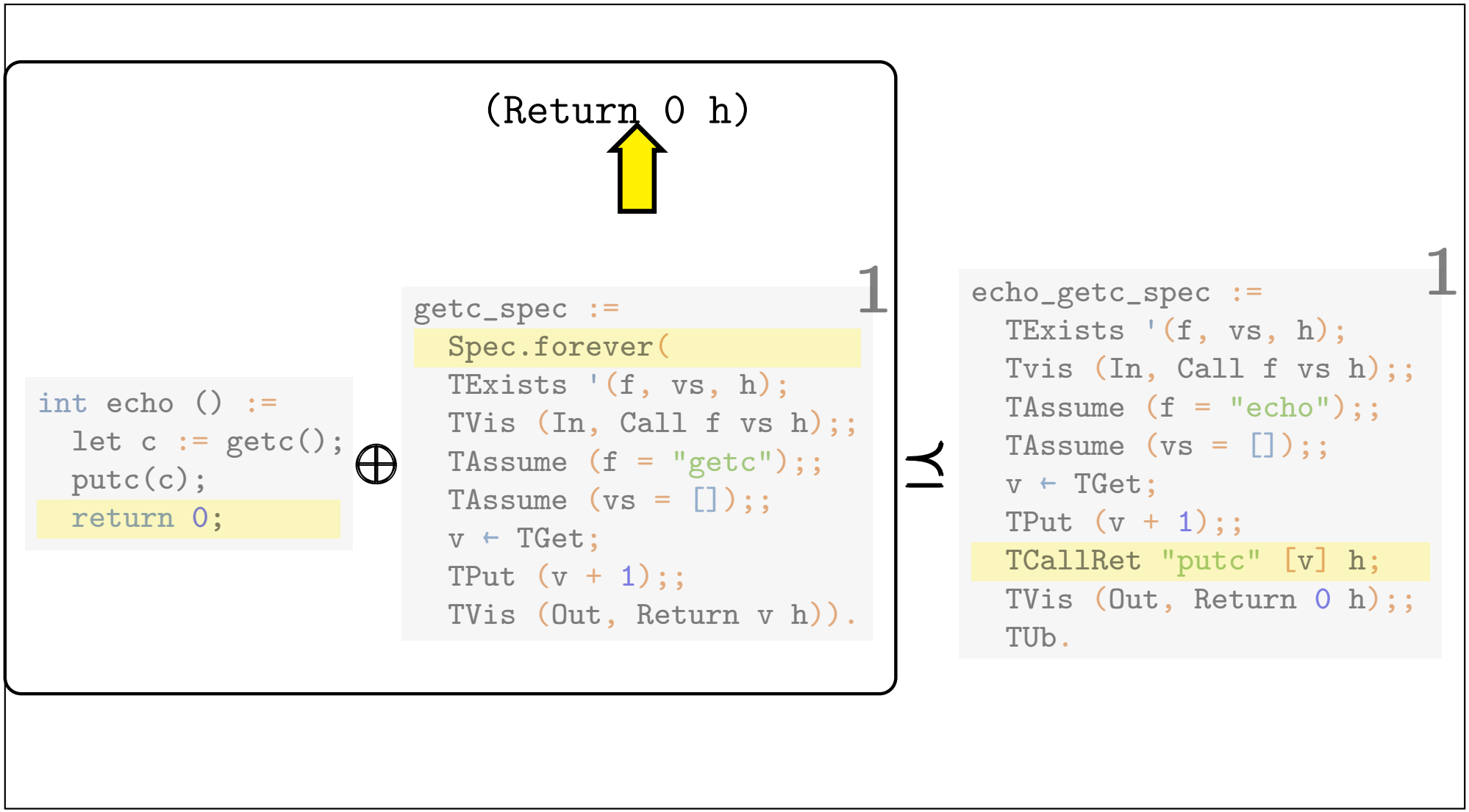
1

$\succeq_1$

```
echo_getc_spec :=
  TExists '(f, vs, h);
  Tvis (In, Call f vs h);;
  TAssume (f = "echo");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TCallRet "putc" [v] h;
  TVis (Out, Return 0 h);;
  TUb.
```

1

Example


(Return 0 h)
int echo () :=
let c := getc();
putc(c);
return 0;
getc_spec :=
Spec.forever(
TExists '(f, vs, h);
TVis (In, Call f vs h);;
TAssume (f = "getc");;
TAssume (vs = []);;
v ← TGet;
TPut (v + 1);;
TVis (Out, Return v h)).
1
echo_getc_spec :=
TExists '(f, vs, h);
Tvis (In, Call f vs h);;
TAssume (f = "echo");;
TAssume (vs = []);;
v ← TGet;
TPut (v + 1);;
TCallRet "putc" [v] h;
TVis (Out, Return 0 h);;
TUb.
1

Example

```
int echo () :=
  let c := getc();
  putc(c);
  return 0;
```

⊕

```
getc_spec :=
  Spec.forever(
  TExists '(f, vs, h);
  TVis (In, Call f vs h);;
  TAssume (f = "getc");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TVis (Out, Return v h)).
```

1

⪯

(Return 0 h)

```
echo_getc_spec :=
  TExists '(f, vs, h);
  Tvis (In, Call f vs h);;
  TAssume (f = "echo");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TCallRet "putc" [v] h;
  TVis (Out, Return 0 h);;
  TUb.
```

1

Example

```
int echo () :=
  let c := getc();
  putc(c);
  return 0;
```

$\oplus$

```
getc_spec :=
  Spec.forever(
  TExists '(f, vs, h);
  TVis (In, Call f vs h);;
  TAssume (f = "getc");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TVis (Out, Return v h)).
```

1

$\succ$

```
echo_getc_spec :=
  TExists '(f, vs, h);
  Tvis (In, Call f vs h);;
  TAssume (f = "echo");;
  TAssume (vs = []);;
  v ← TGet;
  TPut (v + 1);;
  TCallRet "putc" [v] h;
  TVis (Out, Return 0 h);;
  TUb.
```

1

Towards Hoare-style Reasoning

$[\![\mathsf{echo}_{\mathsf{rec}} \oplus \mathsf{getc}_{\mathsf{spec}}]\!]$ $\preceq$ $[\![\mathsf{echo}_{\mathsf{spec}}]\!]$

Towards Hoare-style Reasoning

$$[\![\mathsf{echo}_{\mathsf{rec}} \oplus \mathsf{getc}_{\mathsf{spec}}]\!] \approx\!> (\boldsymbol{\lambda} \kappa_t \sigma_t,$$

Towards Hoare-style Reasoning

$$[\![\mathsf{echo}_{\mathsf{rec}} \oplus \mathsf{getc}_{\mathsf{spec}}]\!] \approx\!> (\boldsymbol{\lambda} \kappa_t \sigma_t, [\![\mathsf{echo}_{\mathsf{spec}}]\!] \approx\!> (\boldsymbol{\lambda} \kappa_s \sigma_s,$$

Towards Hoare-style Reasoning

$$[\![\mathsf{echo}_{\mathsf{rec}} \oplus \mathsf{getc}_{\mathsf{spec}}]\!] \approx\!> (\boldsymbol{\lambda} \kappa_t \sigma_t, [\![\mathsf{echo}_{\mathsf{spec}}]\!] \approx\!> (\boldsymbol{\lambda} \kappa_s \sigma_s, \kappa_t = \kappa_s * \sigma_t \preceq \sigma_s))$$

Towards Hoare-style Reasoning

$$[\![\mathsf{echo}_{\mathsf{rec}} \oplus \mathsf{getc}_{\mathsf{spec}}]\!] \approx\!> \overbrace{(\boldsymbol{\lambda} \kappa_t \sigma_t, [\![\mathsf{echo}_{\mathsf{spec}}]\!] \approx\!> (\boldsymbol{\lambda} \kappa_s \sigma_s, \kappa_t = \kappa_s * \sigma_t \preceq \sigma_s))}^{\Pi_s}$$

Towards Hoare-style Reasoning

$$[\![\mathsf{echo}_{\mathsf{rec}} \oplus \mathsf{getc}_{\mathsf{spec}}]\!] \approx> \Pi_s$$

Towards Hoare-style Reasoning

$$[\![\mathsf{echo}_{\mathsf{rec}}]\!] \approx> \Pi_{\oplus}(\Pi_s)$$