

Modular I/O Reasoning in DimSum

Alex Loitzl¹

¹Institute of Science and Technology Austria (ISTA)

March, 2025

- 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

$$\llbracket \text{echo}_{\text{rec}} \oplus \text{getc}_{\text{spec}} \rrbracket \sqsubseteq \llbracket \text{echo}_{\text{spec}} \rrbracket$$

```
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)).
```

 0
 \sqsubseteq

```
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

(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

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

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

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

(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

(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

```
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  $\leftarrow$  TGet;
    TPut (v + 1);;
    TVis (Out, Return v h)).
```

0

\sqsubseteq

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

0

```
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)).
```

0

\sqsubseteq

```
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

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

(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

(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

(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

(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

(Call "getc" [] 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  $\leftarrow$  TGet;  
    TPut (v + 1);;  
    TVis (Out, Return v h)).
```

0

\sqsubseteq

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

0

(Call "getc" [] 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  $\leftarrow$  TGet;  
    TPut (v + 1);;  
    TVis (Out, Return v h)).
```

0

\sqsubseteq

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

0

```
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)).
```

 0
 \sqsubseteq

```
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

```
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

 \sqsubseteq

```
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


```
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)).
```

(Return 0 h)



1

 \sqsubseteq

```
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

(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

(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

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

(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

```
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



(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

```
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



(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

```
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



(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


```
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



(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

```
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



(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

```
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

(Return v h)



(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)).
```



```
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.
```

(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

(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

(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)).
```



```
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.
```

```
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


```
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

\sqsubseteq

```
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

RTL Transition

$$\frac{c(pc) = \lfloor \text{op}_{RTL}(op, \vec{x}, y, pc') \rfloor \quad \text{eval_op}(_, _, op, M(\vec{x})) = \lfloor v \rfloor}{_ \vdash S(_, _, _, pc, M, _) \xrightarrow{\varepsilon} S(_, _, _, pc', M[y \leftarrow v], _)}$$

LTL Transition

$$\frac{\text{eval_op}(_, _, op, L(\vec{p})) = \lfloor v \rfloor}{_ \vdash B(_, _, _, \text{op}_{LTL}(op, \vec{p}, q) :: bb, L, _) \xrightarrow{\varepsilon} B(_, _, _, bb, L[q \leftarrow v], _)}$$

	vpr	mesa	fuzz1	fuzz2	fuzz3
arm_hard	-0.83%	-1.77%	-4.78%	-4.7%	-4.7%
arm_soft	-0.2%	-0.71%	-0.2%	+0.19%	+0.27%

- Improved model of the Arm assembly semantics
- Proved all architectures correct w.o. changing their semantics
- New and more general register allocator
- Enable future support for TriCore architecture
- Small positive impact on code generation

- ① Background
- ② Project
- ③ RTL to LTL
- ④ Evaluation

	Remaining		Inserted		Reloads		Spills	
	C	C ^P	C	C ^P	C	C ^P	C	C ^P
vpr	4557	4557	165	0	275	275	298	297
mesa	13414	13420	939	0	1401	1276	2265	2133
fuzz1	119	118	40	0	17	17	17	15
fuzz2	404	404	148	0	115	115	74	65
fuzz3	1515	1515	533	0	456	461	267	226