Dependently Typed Assembly for Secure Linking

William J. Bowman
Linking requires types

\begin{align*}
\text{import } f : A. & & \text{export } f : A. \\
\begin{array}{c}
\text{e} \\
\text{e}'
\end{array}
\end{align*}
Linking requires types

import f : A.  export f : A.

Type Checker:
You are go for linking.
Linking requires types

import f : A.

export f : B.

Type Checker:
Error, expected A but found B
Linking requires types

import f : A.  export f : B.

**Linker:**
Sure, there’s an f.
Linking requires types

```
export f : B.
```

Linker:
Sure, there’s an f.
Linking requires types

export f : B.

**Linker:**
Sure, there’s an f.
Theorem. (Type Preservation)

If $e : A$

then $e^+ : A^+$ compiles to
Linking requires types

`export f : B.`

**Linker/Type Checker:**
Error: expected A+ but found B+
Okay, I get it... abstractly
The Problem

```plaintext
export eq_mask: x:uint32_s → y:uint32_s →
   {z:uint32_s |
    if reveal x = reveal y
    then reveal z = 0xffffffff
    else reveal z = 0x0}
```
The Solution:
Dependently Typed Assembly
First, some syntax

Start of block

```
label_name(r1:int32, rk:Π(r1:int32), pf:r1 = 0):
  jmp rk;
```
First, some syntax

Start of block

Block preconditions

```plaintext
label_name(r1:int32, rk:Π(r1:int32), pf:r1 = 0):
    jmp rk;
```
First, some syntax

```
label_name(r1:int32, rk:Π(r1:int32), pf:r1 = 0):
  jmp rk;
```

Start of block  Block preconditions  Continuation/Post-conditions
One more thing

```
extern label_name : Π(r1:int32, rk:Π(r1:int32));
```
The Solution:

Dependently Typed Assembly

extern uint32_s : Type;

extern uint32_s=(x:uint32_s, y:uint32) : Prop;
The Solution:
Dependently Typed Assembly

dependent typedef uint32_s : Type;

dependent typedef uint32_s=(x:uint32_s, y:uint32) : Prop;

eq_mask(r1:uint32_s, r2:uint32_s, 
  rk:Π(r3:uint32_s, pf: 
    (And (r1 = r2 -> uint32_s= r3 0xffffffff) 
    (r1 != r2 -> uint32_s= r3 0x0)))));

....
The Solution:
Dependently Typed Assembly

extern uint32_s : Type;

extern uint32_s=(x:uint32_s, y:uint32) : Prop;

eq_mask(r1:uint32_s, r2:uint32_s,
  rk:Π(r3:uint32_s, pf:
    (And (r1 = r2 -> uint32_s= r3 0xffffffff)
    (r1 != r2 -> uint32_s= r3 0x0))):

r1 and r2 are live, secure Integers

....
The Solution:
Dependently Typed Assembly

extern uint32_s : Type;

post condition: r3 is secure integer, plus proof objects

eq_mask(r1:uint32_s, r2:uint32_s,
   rk:Π(r3:uint32_s, pf:
      (And (r1 = r2 -> uint32_s= r3 0xffffffff)
           (r1 != r2 -> uint32_s= r3 0x0)))))

....
Secure Linking

*Linker/Type checker:* Type error: e trying to branch secure integer.