User Constraints in a Metacomputing Environment

Luc Alexander Lüthi

January 4th, 2026

Contents

1 The Limits of Bind/Comp Algebra **1**

2 Factoring Fix **2**

1 The Limits of Bind/Comp Algebra

Compile time execution cannot be looped over, all looping logic must happen at the deepest level of nested compilation, and the changes must propagate upward. This lesson was learned the hard way. The first attempt at a constraint protocol involved a registry which lived in its own compile time environment.

```
(bind constraint_address () 0)

(bind constraint_setup ()
    (comp constraint
        ((uid (x) (
            (reg x)
            (mov x (constraint_address))
            (mov (at x) 1)
            (mov r11 x)
            (add x x 8)
            (mov (at x) 8)
            (mov x 0)
            (int x))))))
```

Errorful predicate constraint functions can then be registered with the system with the expectation that they will be sequenced on any invocation of the (info key val) protocol hook. In the latest version of un a new special form (err "message") has been added to allow user error initiation, predicates are expected to emit this special form when their logical invariants are violated, thus triggering a compilation error for the metaprogram.

```
(bind constraint (predicate)
    (comp constraint
        ((uid (x y) (
            (reg x)
            (reg y)
```

```
            (mov x (constraint_address))
            (add x x 8)
            (mov y (at x))
            (add y y 10)
            (mov (at x) y)
            (add x x y)
            (sub x x 8)
            (reif y predicate)
            (mov (at x) 1)
            (add x x 8)
            (mov (at x) y)
            (mov r11 (constraint_address))
            (mov x 0)
            (int x))))))

(bind info (key val)
    (comp constraint
        ((uid (loop end i n temp pred) (
            (reg i)
            (reg n)
            (mov i 0)
            (mov n (constraint_address))
            (reg temp)
            (mov temp n)
            (add temp temp 10)
            (mov n (at n))
            (reg pred)
            (label loop)
                (cmp i n)
                (jge end)
                (add i i 10)
                (mov pred ((flat (fetch_constraint temp)) (key val)))
                (add temp temp 10)
                (jmp loop)
            (label end))))))
```

There exists a simple mistake in the logic wherein you cannot dispatch symbols from a compile time environment dynamically, it must be done upfront instead, which means generating the desired loop code as an unrolled set of instructions within the constraint virtual machine itself.

2 Factoring Fix

The setup is nearly identical, but this time the info hook does all the work, generating s expressions in bytecode to run a sequence of constraint predicates which can error the compilation process.

```
(bind collect_constraints (key val)
    (comp constraint
        ((uid (i n x y rkey rval target tagged ptr loop exit) (
            (reg i)
```

```
            (reg n)
            (reg x)
            (reg y)
            (reg tagged)
            (reg target)
            (reg ptr)
            (reg rkey)
            (reg rval)
            (reg ptr)
            (reif rkey key)
            (reif rval val)
            (mov x (constraint_address))
            (add x x 8)
            (mov y (at x))
            (mov n y)
            (add x x 8)
            (mov target 80)
            (mov r11 target)
            (shr y y 3)
            (mov (at target) y)
            (add target target 8)
            (mov ptr FFF)
            (label loop)
                (cmp i n)
                (jge exit)
                (mov y (at x))
                (mov tagged ptr)
                (tag_ptr tagged)
                (mov (at target) tagged)
                (mov (at ptr) 3)
                (add ptr ptr 8)
                (mov (at ptr) y)
                (add ptr ptr 8)
                (mov (at ptr) rkey)
                (add ptr ptr 8)
                (mov (at ptr) rval)
                (add ptr ptr 8)
                (add target target 8)
                (add i i 8)
                (add x x 8)
                (jmp loop)
            (label exit)
            (mov x 0)
            (int x))))))

(bind info (key val)
    (comp vm ((
        (collect_constraints key val)
        (mov r0 0)
        (mov (at r0) 0)
        (int r0)))))
```

The hook can then be leveraged anywhere in the desired domain specific language for ad hoc constaints.

```
(bind define (name args body)
    (bind name (args) (
        (info "called" name)
        (body))))
```

Here is an example of a constraint predicate.

```
(use "constraint.un")

(sink database_write)
(source database_read)

(define update_user data
    ((uid (x) (
        (reg x)
        (mov x (database_read data))
        (database_write data)))))

(bind vacuous (key val)
    ((err "type error")))

(bind main ()
    ((uid (x) (
        (reg x)
        (mov x (constraint_setup))
        (mov x (constraint vacuous))
        (update_user ("john" "password"))
        (mov x 0)
        (int x)))))

(main)
```