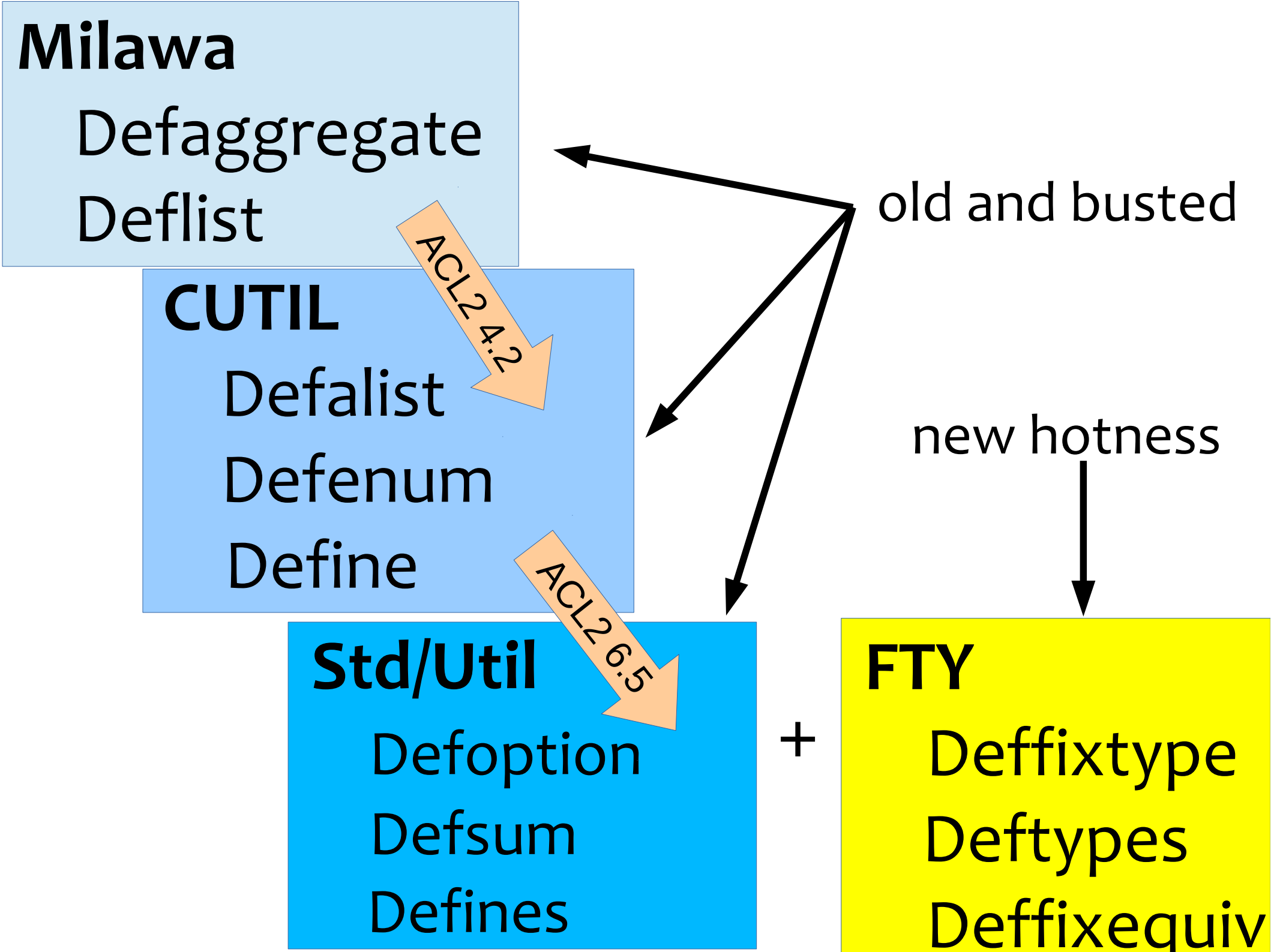
Fix Your Types

Sol Swords, Jared Davis

ACL2 Workshop, October 2015

CenTaur Technology


Milawa
Defaggregate
Deflist
old and busted
ACL2 4.2
CUTIL
Defalist
Defenum
Define
new hotness
ACL2 6.5
Std/Util
Defoption
Defsum
Defines
+
FTY
Deffixtype
Deftypes
Deffixequiv

I really like types.

Signature

```
(svex->a4vec x env masks memo) → (mv res memo1)
```

Arguments

x — Guard (svex-p x).
env — Guard (svex-a4vec-env-p env).
masks — Guard (svex-mask-alist-p masks).
memo — Guard (svex-aig-memotable-p memo).

Returns

res — Type (a4vec-p res).
memo1 — Type (svex-aig-memotable-p memo1).

(student->name x)

vs.

(caddadr x)

Type Safety

Not a string

```
:/xdoc/preprocess.lisp

@@ -1081,7 +1081,14 @@ baz
           (mv nil sexpr)))
        ((mv err vals state) (acl2::unsound-eval sexpr))
        ((when err)
-        (mv (str::cat "Error: failed to evaluate @(`...`): " err)
+        (mv (str::cat "Error: failed to evaluate @(`...`): "
+                      ;; BOZO this isn't right, we really want something like
+                      ;; ~@, but we don't have that unless we use ACL2's
+                      ;; built-in fmt-to-string stuff, which I don't want to
+                      ;; use due to the problems described in
+                      ;; fmt-to-str-orig.lisp...  For now, we can at least just
+                      ;; stringify the error in a dumb way.
+                      (str::pretty err :config (str::make-printconfig :home-packa
             acc
             state))
        (ret (cond ((atom vals)
```

rocks are carnivores

dynamic typing saves time?
recertifying...

master

jaredcdavis authored on Jun 30 1 parent f5bc959 commi

Showing 1 changed file with 1 addition and 1 deletion.

Not a context **Not a topics-fal**

2 books/xdoc/preprocess.lisp

```
@@ -1121,7 +1121,7 @@ ba
1121 1121                kind (str::pretty sexpr))
1122 1122           acc state)))
1123 1123
1124      -(defun preprocess-eval (str topics-fal context base-pkg kpa state acc)
     1124 +(defun preprocess-eval (str context topics-fal base-pkg kpa state acc)
1125 1125    "Returns (MV ACC STATE)"
1126 1126    (b* (((mv errmsg sexpr state) (preprocess-eval-parse str base-pkg state))
1127 1127         ((when errmsg)
```

fix horrible, stupid bug

master

jaredcdavis authored on Jul 8, 2014 — 1 parent 0296d39 com

ragerdl committed on Sep 1, 2014

Not a tokenkind

Showing 1 changed file with **1 addition** and **1 deletion**.

2 books/centaur/vl/loader/parser/modules.lisp

```
@@ -106,7 +106,7 @@
106 106     :count strong
107 107     (seqw tokens warnings
108 108           (when (eq (vl-loadconfig->edition config) :verilog-2005)
109     -           (:= (vl-match-token :vl-always))
    109 +           (:= (vl-match-token :vl-kwd-always))
110 110             (return :vl-always))
111 111           (kwd := (vl-match-some-token '(:vl-kwd-always
112 112                                          :vl-kwd-always_comb
```

fix bug in bit-use-set handling of statements

master

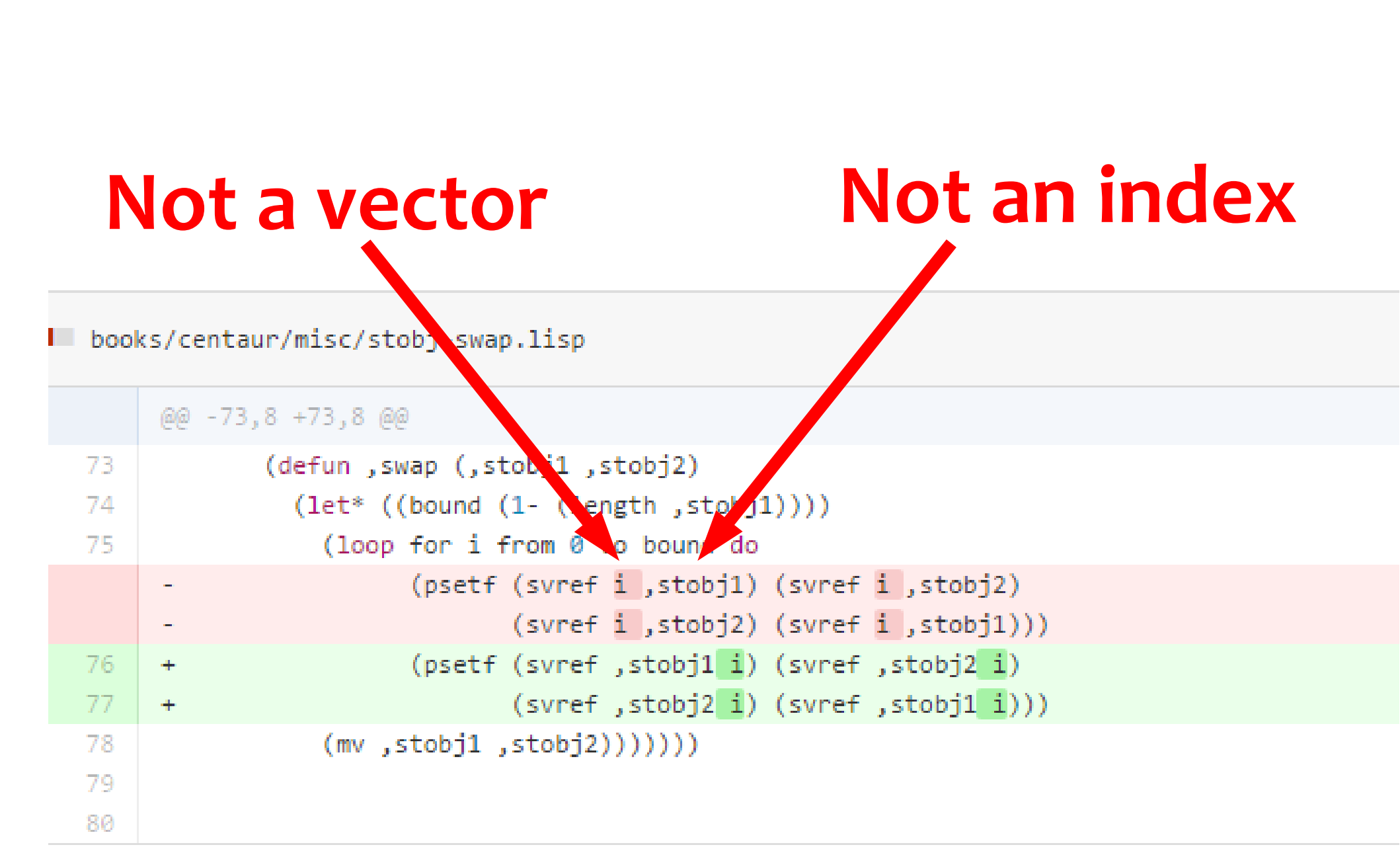
jaredcdavis authored on Dec 30, 2014 1 parent 066a

Showing 1 changed file with 1 addition and 1 deletion.

Not tagged

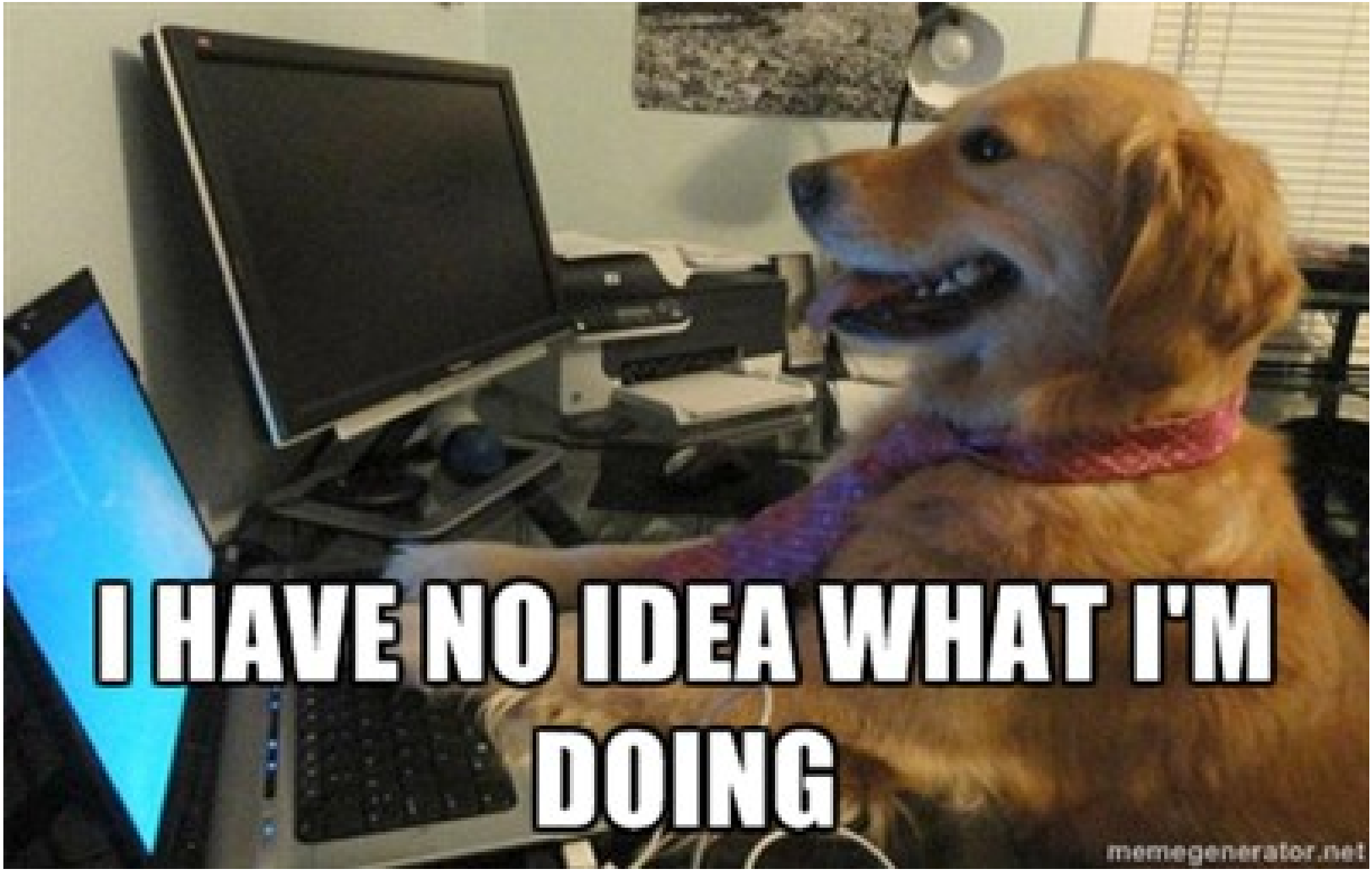
2 books/centaur/vl/lint/bit-use-set.lisp

```
@@ -1017,7 +1017,7 @@ warnings."
1017 1017          (warnings (vl-warninglist-fix warnings))
1018 1018
1019 1019          ((when (vl-atomicstmt-p x))
1020      -         (case (tag (vl-stmt-kind x))
     1020 +         (case (vl-stmt-kind x)
1021 1021            ;; - Nothing to do for null statements.
1022 1022            ;; - Don't think we want to do anything for eventtrigger
1023 1023            ;; - Don't think we want to do anything for deassign sta
```



books/centaur/misc/stobj-swap.lisp

```
@@ -73,8 +73,8 @@
73          (defun ,swap (,stobj1 ,stobj2)
74            (let* ((bound (1- (length ,stobj1))))
75              (loop for i from 0 to bound do
-                   (psetf (svref i ,stobj1) (svref i ,stobj2)
-                          (svref i ,stobj2) (svref i ,stobj1)))
76 +                 (psetf (svref ,stobj1 i) (svref ,stobj2 i)
77 +                        (svref ,stobj2 i) (svref ,stobj1 i)))
78            (mv ,stobj1 ,stobj2))))))
79
80
```



But I think types help me.

What I want…

- Easily introduce new types
- Safely change existing types
- Write type-safe programs
- Run them efficiently
- Reason about them effectively

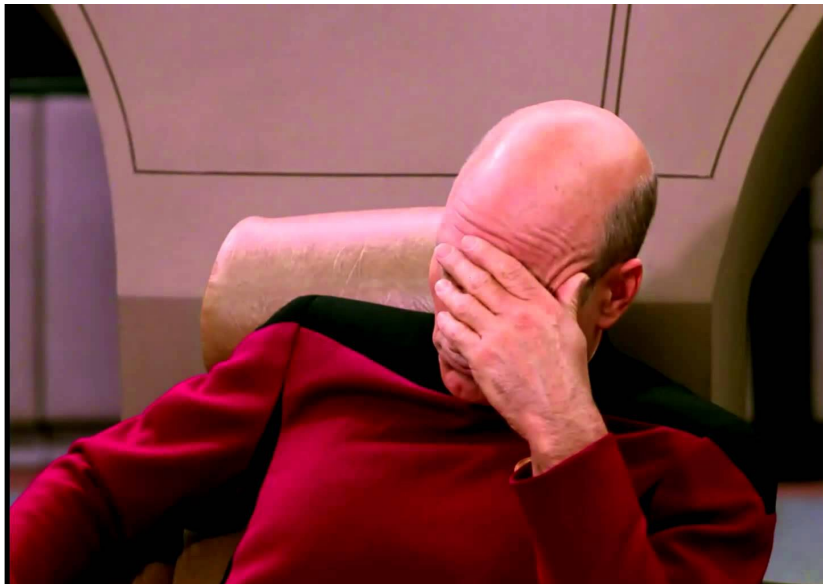
Example

```
struct student {
   string name;
   int age;
   …
};
```


```
struct student {
   string name;
   int age;
   …
};
```

```
(defund student-p (x)  …)
(defund student->name (x) (first x))
(defund student->age (x) (second x))
(defund make-student (name age) (list
```

(defund student-p (
(defund student->na
(defund student->ag
(defund make-stude



(defthm booleanp-of-student-p ...)
(defthm student-p-of-make-student ...
(defthm stringp-of-student->name ...)
(defthm natp-of-student->age ...)
(defthm student->name-of-make-stud
(defthm student->age-of-make-studen

Easily introduce new types

Std/Util

```
(defaggregate student
  ((name stringp)
   (age    natp) ...))
```

Alternatives

defdata, defstructure, defrec

Safely change existing types

```
(defaggregate student
  (;; (name stringp)
    (first stringp)
    (last stringp)
    (age natp) …))
```

```
(cw “Little ~s0 was tardy!” (student->name x))
```

Write type-safe programs

```
(define reportcard ((x student-p))
  :returns (report report-p)
  …  (cat …
        “Obviously ” x.name
        “ is far ahead of most ” x.age
        “ year-olds in many important
          areas.” …)  …))
```

Run them efficiently

Inlining

Layout options (tagless, illegible)

Honsing options

Type safety handled statically

So far so good!

- ✸ Easily introduce new types
- ✸ Safely change existing types
- ✸ Write type-safe programs
- ✸ Run them efficiently
- ? Reason about them effectively

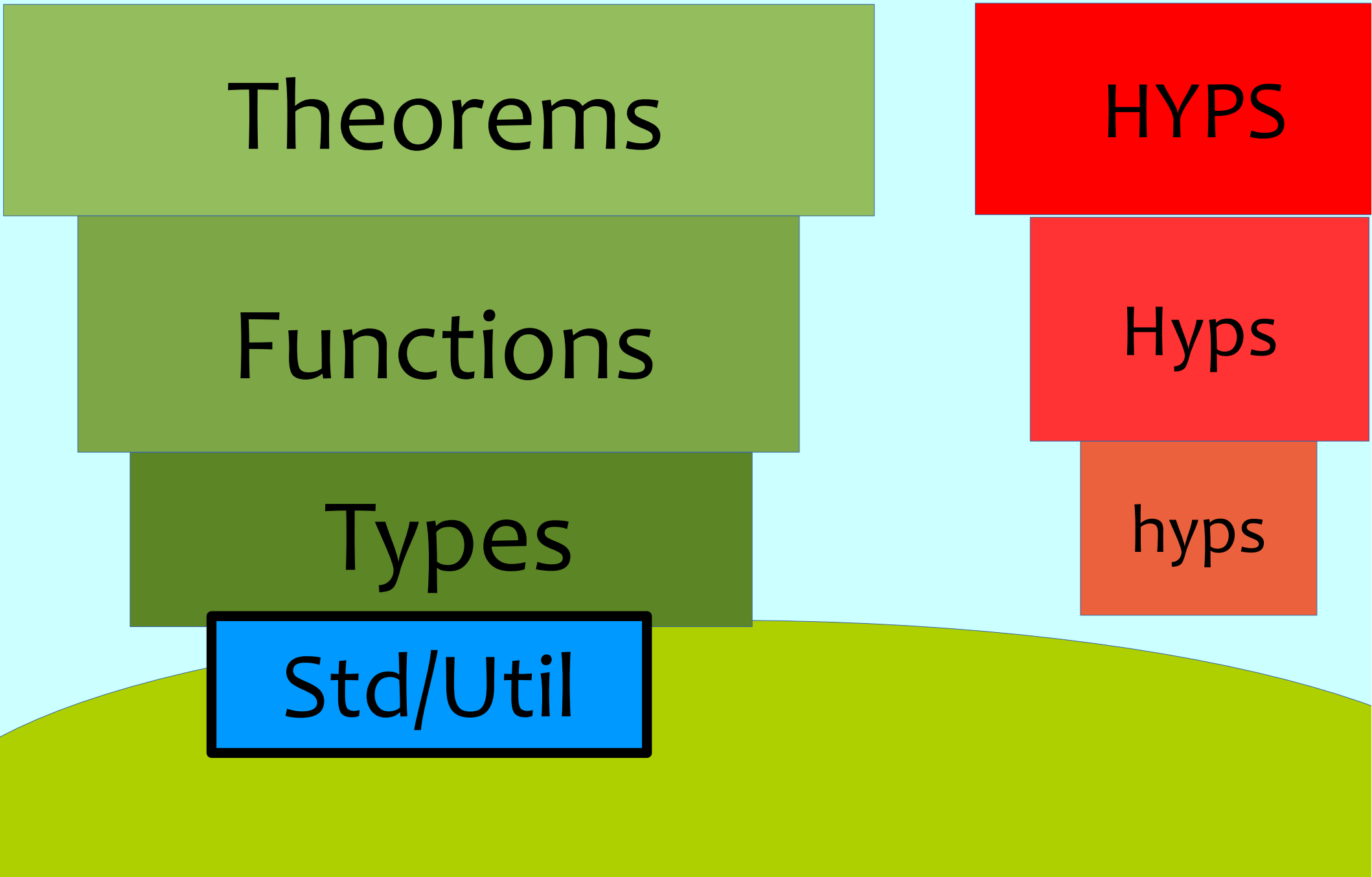
Reason about them effectively

```
(defthm student->name-of-make-student
  (equal  (student->name (make-student
                                  :name name
                                  :age age ...))
          name))

(defthm student->age-of-make-student
  (equal  (student->age (make-student ...))
          age))
```

Reason about them effectively

```
(defthm student-p-of-make-student
  (implies (and (stringp name)
                (natp age))
           (student-p (make-student ...))))

(defthm stringp-of-student->name
  (implies (student-p x)
           (stringp (student->name x))))
```

So What?



End Result

Std/Util

```
(defthm vl-plainarglist-p-of-vl-partition-plainarg
 (b* (((mv okp warnings plainargs)
       (vl-partition-plainarg arg port-width insts mod
                              ialist elem warnings)))
   (implies
     (and okp
          (force (vl-plainarg-p arg))
          (force (posp port-width))
          (force (posp insts))
          (force (vl-module-p mod))
          (force (equal ialist (vl-moditem-alist mod))))
        (vl-plainarglist-p plainargs))))
```

Tedious
Slow
Unreliable

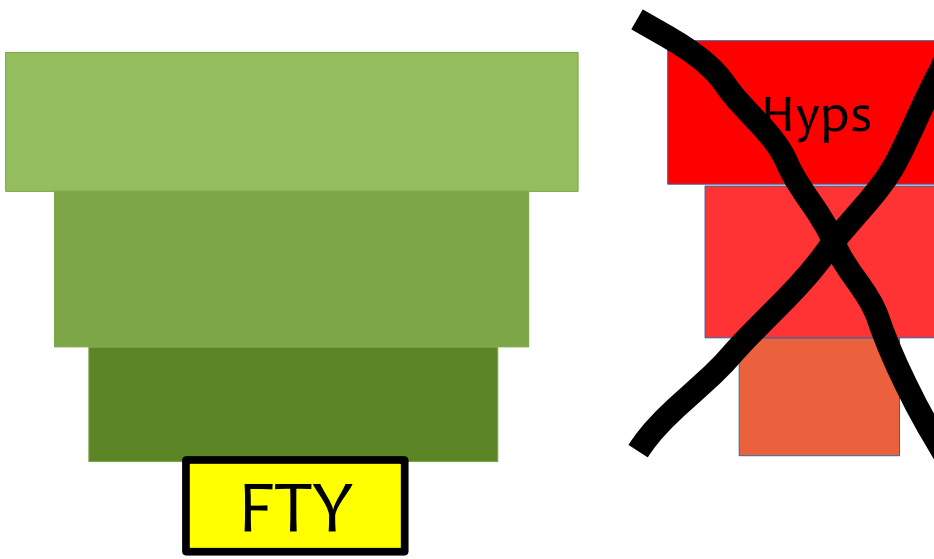
```
(and okp
     (force (vl-plainarg-p arg))
     (force (posp port-width))
     (force (posp insts))
     (force (vl-module-p mod))
     (force (equal ialist (vl-moditem-alist mod))))
  (vl-plainarglist-p plainargs))))
```

FTY

Discipline + Automation

for eliminating type hypotheses

```
(defthm vl-plainarglist-p-of-vl-partition-plainarg
  (b* (((mv okp warnings plainargs)
        (vl-partition-plainarg arg port-width insts ...)))
    (vl-plainarglist-p plainargs)))
```



Treat bad inputs consistently

Functions on Numbers < + - * /

$$x*(y+z) = x*y + x*z$$

Functions on Naturals	zp, nth, logbitp
Functions on Strings	char, string-append
Functions on Conses	car, cdr, endp
Functions on Sets	set::head, set::union

Fixing Function

How to handle bad inputs

1. typep(fix(x))
2. typep(x) → fix(x) = x

Example

1. natp(nfix(x))
2. natp(x) → nfix(x) = x

Fixing Equivalence

equiv(x,y) == fix(x) = fix(y)

```
(defun nat-equiv (x y)
  (equal (nfix x) (nfix y)))

(defequiv nat-equiv)

(defthm nth-respects-nat-equiv
   (implies (nat-equiv n1 n2)
            (equal (nth n1 x) (nth n2 x))))
```

FTY Discipline

1. Types should have a corresponding fixing function and equivalence relation.

FTY

Discipline

2. Typed functions should produce equal results for any equivalent inputs.

Fix your inputs before using them

FTY

Automation

1. Introducing types with fixing functions and equivalences

2. Proving congruences for functions that use these types

Automation

Deffixtype – Type Database

Basetypes – Definitions for basic types

Deftypes – Automation for new types

Deffixequiv – Automated congruences

Base Types

Type Name	Recognizer	Fix	Equiv
bit	bitp	bfix	bit-equiv
nat	natp	nfix	nat-equiv
int	integerp	ifix	int-equiv
rational	rationalp	rfix	rational-equiv
acl2-number	ACL2-numberp	fix	number-equiv
true-list	true-listp	list-fix	list-equiv

Deftypes

```
(defprod student
  ((name stringp)
   (age     natp) ...))
```

Styled after std/util!

```
(defund student-p (x)  ...)
(defund student->name (x) (string-fix (f
(defund student->age (x) (nfix (second
(defund make-student (name age) (list
(defund student-fix (x) (if (student-p x)
(defund student-equiv (x y) (equal (stu
(defthm student-fix-when-student-p (i
(defthm student-p-of-student-fix (stud
(defequiv student-equiv :hints((“Goal”
```

Deftypes

defprod
deflist
defalist
defoption
deftagsum
deftranssum
defflexsum

+ Mutual Recursion!

Congruences

```
(define multiply-and-add ((a natp)
                          (b natp)
                          (c natp))
  ...)

(deffixequiv multiply-and-add)
```

or even fully automatically with a hook

Automation

Deffixtype – Type Database

Basetypes – Definitions for basic types

Deftypes – Automation for new types

Deffixequiv – Automated congruences

Fix Your Types

Sol Swords Jared Davis

Centaur Technology, Inc.
7600-C N. Capital of Texas Hwy, Suite 300
Austin, TX 78731

{sswords,jared}@centtech.com

When using existing ACL2 datatype frameworks, many theorems require type hypotheses. These hypotheses slow down the theorem prover, are tedious to write, and are easy to forget. We describe a principled approach to types that provides strong type safety and execution efficiency while avoiding

Jump to append Search files

ACL2::macro-libraries

Fty

[books]/centaur/fty/top.lisp

FTY
Package

FTY is a macro library for introducing new data types and writing type-safe programs in ACL2. It automates a systematic discipline for working with types that allows for both efficient reasoning and execution.

FTY, short for *fixtype*, is a library for type-safe programming in ACL2. It provides significant automation for introducing new data types and using data types according to the "fixtype discipline." Following this discipline allows you to write type-safe programs that support efficient reasoning (by minimizing the need for type-related hypotheses) and also have good execution efficiency.

Thanks!