Miscellaneous utilities

Marc Coiffier

Contents

Reminder : all builtin functions can be found documented here

Exporting definitions

```
'defX { 1 dupn swap def export } def 'defX export
'synonym { $ def } defX
```

Navigating the environment

```
'show-context {
   "" hypotheses
   { dup variable type swap "%s : %v\n%s" format } each
   print pop
 } defX
'showdef { pattern-index 1 swapn swap index-insert set-pattern-index } defX
'vis { show-context "-------\n" printf show-stack } defX
```

Binders and contexts

```
'binder { 2 shaft { ${ swap } -> ${ } ${ } } } defX
'funs { swap reverse { swap '! $ binder } each exec } defX
'prods { swap reverse { swap '? $ binder } each exec } defX
'# { swap cons } defX
```

Constructing typed terms

```
'Type { 0 universe pull } defX
```

```
'foralls { { extro-forall } swap times } defX
'lambdas { { extro-lambda } swap times } defX
'applys { range { pop apply } each } defX
'applyl { { swap apply } each } defX
'recursor { dup 2 shaft -> variable mu ! } defX
```

```
'( '[ $ defX
') { ] applyl } defX
```

Managing the type environment

```
'-> { dup 1 swapn swap intro { ,{ dup } variable pull } def } defX
'! 'extro-lambda $ defX
'? 'extro-forall $ defX
```

Defining inductive constructors

```
'defconstr { 1 dupn swap showdef def } defX
```

Acting on list as stacks

```
'in-list {
  swap {
    ,{ } set-stack ${ }
    { ,{ ,{ stack } } set-stack ,{ stack } } exec
  } exec
} defX
```