‘Gen‘ is the generic projection generator for the strict part of the list projection domain:

```
  Gen a b c = Nil a U Cons b (c U Gen a b c)
```

We could add a fourth parameter ‘d‘ to catch the non-strict part by adding ‘d U ..‘ to the front of the definition, where ‘d‘ comes from {FAIL, ABS}. There is a clear 1:1 relationship between the two parts though. It’s worth restricting ‘c‘ in the same way (can only be FAIL or ABS) since the projections generated when ‘c‘ is ID or STR are not particularly interesting thanks to the following properties:

```
  STR U NIL a == STR U Cons b c == STR, for all a, b, c
  ID U a = ID, for all a
```

Since the argument to NIL is a projection into a two-point domain, it must be restricted to ID or FAIL.

Now we can enumerate all (2x4x2=16) possible argument sets to Gen and rewrite them in terms of more readable generators:

```
  FNF a = Nil ID U Cons a (ABS U FNF a)
  INF a = Cons a (ABS U FNF a)
  FIN a = Nil ID U Cons a (FIN a)

  Gen FAIL FAIL FAIL = FAIL
  Gen FAIL FAIL ABS  = FAIL
  Gen FAIL ABS  FAIL = FAIL *
  Gen FAIL STR  FAIL = FAIL *
  Gen FAIL ID   FAIL = FAIL *

  * These projections would recognize infinite lists as "acceptable" but finite lists as
    "unacceptable", if they were not the same as FAIL

  Gen FAIL ABS  ABS  = INF ABS  == "LazyIgnStream"
    e.g. f x:xs = 3 : f xs

  Gen FAIL STR  ABS  = INF STR  == "AccumUntil"
    e.g. f x:xs = if x == 3 then True else f xs
     or ‘head‘ (?)

  Gen FAIL ID   ABS  = INF ID   == "LazyStream"
    e.g. f p x:xs = if p then x else 0 : f xs

  Gen ID   FAIL ABS  = FIN FAIL == NIL ID
  Gen ID   FAIL FAIL = FNF FAIL == NIL ID
    [] is the only acceptable arg

  Gen ID   ABS  FAIL = FIN ABS  == "Spine"
    e.g. length

  Gen ID   ABS  ABS  = FNF ABS  == "InfSpine"
    e.g. longerThan10 _ []   = False
         longerThan10 l _:xs =
           if l > 10 then True else longerThan10 (l + 1) xs

  Gen ID   STR  FAIL = FIN STR  == "AccumAll"
    e.g. sum

  Gen ID   STR  ABS  = FNF STR  == "AccumSome"
    e.g.  prodToZ []   = 1
          prodToZ x:xs = if (x == 0) then 1 else x * prodToZ xs

  Gen ID   ID   FAIL = FIN ID   == "AccumPartOfAll"
    e.g. sumEvenIdx _ [] = 0
         sumEvenIdx i (h:t) = sumEvenIdx (i + 1) t + (if even i then h else 0)

  Gen ID   ID   ABS  = FNF ID   == "WHNF" == STR
    e.g. takeOneIf p [] = Nothing
         takeOneIf p (x:_) = if p then Just x else Nothing
```