U051SS2EU : <@U5ZAJ15P0> umm that's kind of what clojure.test does when you run the tests
U051SS2EU : are you talking about doing this outside unit tests?
U5ZAJ15P0 : it only seems to work fell for some predicates
U5ZAJ15P0 : no, running the tests
U5ZAJ15P0 : for example, if the predicate is any arbitrary function it won't report much
U5ZAJ15P0 : I wrote a dumb, poor "schema" toy lib
U5ZAJ15P0 : and getting a report such as:```
expected: (v (get flat-target k))
  actual: false
```

U5ZAJ15P0 : (v is a predicate function)
U5ZAJ15P0 : I'll try with the approach you suggest
U5ZAJ15P0 : (that's running it with `test/is`)
U051SS2EU : ```Clojure 1.9.0-alpha15+user=&gt; (require '[clojure.test :as t] '[schema.core :as s])
nil
+user=&gt; (t/is (nil? (s/check [Number] ["a"])) "expected a number")

FAIL in () (NO_SOURCE_FILE:2)
expected a number
expected: (nil? (s/check [Number] ["a"]))
  actual: (not (nil? [(not (instance? java.lang.Number "a"))]))
false
```

U5ZAJ15P0 : Maybe I could call do-report manually instance
U5ZAJ15P0 : mmh
U5ZAJ15P0 : hang on
U5ZAJ15P0 : seems to work, ok, nevermind
U051SS2EU : if you test a predicate, all you get in the error is that the predicate didn't return the true/false you expected
U051SS2EU : ```+user=&gt; (defmacro handle [obj prop fun] `(.setValue ~(symbol (str (name obj) "/" (name prop))) (fi javafx.event.ActionEvent ~(symbol "event") ~fun)))
#'user/handle
+user=&gt; (macroexpand '(handle a :onAction (fn [x] nil)))
(. a/onAction setValue (user/fi javafx.event.ActionEvent event (fn [x] nil)))
``` I don't know if it's cider, but something's misbehaving

U2APCNHCN : ...ah, it didn't compile and didn't say anythign
U051SS2EU : oh - right-  macroexpand silently outputs a form that doesn't start with a known macro
U051SS2EU : even if the thing in it isn't defined
U2APCNHCN : Back to the drawing board, I guess. Apparently I'm still too dumb to do macros.
U051SS2EU : Cider magic introduces ambiguities and special cases about what's been evaluated and what hasn't, it's an easy mistake
U051SS2EU : and macroexpand doesn't care if the things inside it are known, it just expands known macros and leaves the rest alone
U2APCNHCN : Ok so it expands with `C-x e` in the editor, but not in the REPL, which is set to be the same ns. Huh.
U2APCNHCN : Heh ^^ But I wouldn't want to miss it. It's weird, but it's great.
U2APCNHCN : ...like Emacs, I guess :stuck_out_tongue:
```