

lib/basic/fmaps_unittest.ath

```

1 load "fmaps"
2
3 open FMap
4
5 (define M1 [[1 --> 'a] [2 --> 'b] [1 --> 'c]])
6 (define M2 [['a --> true] ['b --> false] ['foo --> true]])
7 (eval M2 o M1)
8
9 (define [t1 t2] [(alist->fmap M2) (alist->fmap M1)])
10
11 define capitals :=
12   [['paris --> 'france] ['tokyo --> 'japan] ['cairo --> 'egypt]]
13
14 define countries :=
15   [['france --> 'europe] ['algeria --> 'africa] ['japan --> 'asia]]
16
17 (eval countries o capitals)
18
19 #(falsify composition-is-comm 20)
20
21 #(falsify composition-is-assoc 20)
22
23 #(falsify (close ((m3 o m2) o m1) = m3 o (m2 o m1))) 20)
24
25 #(falsify comp2-is-comm 10)
26
27 #(falsify comp2-is-assoc 80)
28
29 # (falsify comp2-app-lemma 10)
30
31 (eval [[1 --> 'a] [2 --> 'b]] <-> [[1 --> 'a] [3 --> 'c]])
32
33 (eval [[1 --> 'a] [2 --> 'b]] <-> [[1 --> 'a] [2 --> 'foo] [3 --> 'c]])
34
35 define compatible-theorem-1 := (forall m . m <-> m)
36
37 (falsify compatible-theorem-1 20)
38
39 define compatible-theorem-2 := (forall m1 m2 . m1 <-> m2 <==> m2 <-> m1)
40
41 (running-time (lambda () (falsify compatible-theorem-2 10)) 0)
42 # with new evall: 4.22
43
44 define compatible-theorem-3 := (forall m1 m2 m3 . m1 <-> m2 & m2 <-> m3 ==> m1 <-> m3)
45
46 # (falsify compatible-theorem-3 10)
47
48 (define [s t hyp] [(apply ?tail:(Map 'T1 'T2)
49                       ?k:'T1)
50
51                   (apply ([?key:'T1 val] ++ ?tail:(Map 'T1 'T2))
52                           ?k:'T1)
53
54                   (?key:'T1 /= ?k:'T1)])/
55
56 (assume hyp
57   (!chain [s = t [apply-axioms]]))
58
59
60 define M' := [[1 --> 'a] [2 --> 'bar] [1 --> 'c]]
61
62 (eval agree-on M M' [1])
63 (eval agree-on M M' [2])
64
65 define ag-conjecture-1 :=
66   (forall S m . agree-on m m S)
67

```

```

68 # Older map identity:
69 #assert* map-identity :=
70 # (m1 = m2 <==> dom m1 = dom m2 & (agree-on m1 m2 dom m1));
71
72 define agree-characterization :=
73   (forall S m1 m2 . (agree-on m1 m2 S) <==> forall x . x in S ==> m1 applied-to x = m2 applied-to x)
74
75 # by-induction agree-characterization {
76 #   (S as Set.null) =>
77 #     pick-any m1 m2
78 #     let {dir1 := assume hyp := (agree-on m1 m2 null)
79 #         pick-any x
80 #           (!chain [(x in S) ==> false [Set.null-characterization]
81 #                 ==> (m1 applied-to x = m2 applied-to x) [prop-taut]]);
82 #         dir2 := assume hyp := (forall x . x in null ==> m1 applied-to x = m2 applied-to x)
83 #           (!chain-> [true ==> (agree-on m1 m2 S) [agree-on-axioms]])}
84 #     (!equiv dir1 dir2)
85 # | (S as (Set.insert h t)) =>
86 #   let {dir1 := assume hyp := (agree-on m1 m2 S)
87 #       }
88
89 #(falsify agree-characterization 10)
90 #(!induction* agree-characterization)
91
92 (eval (empty-map:(Map Int Int) at 1))
93
94 (eval M applied-to 1)
95 (eval M applied-to 2)
96 (eval M applied-to 97)
97 (eval M - 1 applied-to 2)
98 (eval M - 1 applied-to 1)
99
100 (define M1 [[1 --> 'a] [2 --> 'b] [1 --> 'c]])
101
102 (define M2 [['a --> true] ['b --> false] ['foo --> true]])
103
104 (eval M2 o M1)
105
106 #(falsify composition-is-assoc 10)
107
108 define [n] := [?n:N]
109
110 let {m := (alist->fmap [[1 --> 2] [2 --> 3] [3 --> 1]]);
111     _ := (print "\nm iterated once: " (eval m ^ 1));
112     _ := (print "\nm iterated twice: " (eval m ^ 2));
113     _ := (print "\nm iterated thrice: " (eval m ^ 3))}
114 (print "\nAre m and m^3 identical?: " (eval m = m ^ 3))
115
116 (eval [[1 --> 'a] [2 --> 'b]] <-> [[1 --> 'a] [3 --> 'c]])
117
118 (eval [[1 --> 'a] [2 --> 'b]] <-> [[1 --> 'a] [2 --> 'foo] [3 --> 'c]])
119
120 define compatible-theorem-1 := (forall m . m <-> m)
121
122 (falsify compatible-theorem-1 20)
123
124 define compatible-theorem-2 := (forall m1 m2 . m1 <-> m2 <==> m2 <-> m1)
125
126 #(running-time (lambda () (falsify compatible-theorem-2 50)) 0)
127 # with new evall: 4.22
128
129 define compatible-theorem-3 := (forall m1 m2 m3 . m1 <-> m2 & m2 <-> m3 ==> m1 <-> m3)
130
131 #(falsify compatible-theorem-3 10)
132
133 #(define remove-correctness
134 # (forall m x . (m - x) applied-to x = NONE))
135
136 #(holds? remove-correctness)
137 #(!induction* remove-correctness)

```

```
138
139 #(falsify remove-correctness 100)
140
141 #(define conj
142 # (close (agree-on m1 m2 A) <==> m1 ^ A = m2 ^ A))
143 #? (close (agree-on m1 m2 A) <==> m1 |^ A = m2 |^ A))
144
145 #(!induction* conj)
```