NetLogo 5 Tasks – referencing#1

outline

This is in response to a couple of questions asked about NetLogo tasks after release of trial versions of NetLogo 5.

I have experimented with tasks because I am curious about performance gains with NLoops (the partial OO system for NetLogo) – see www.agent-domain.org.

testing

Presented here is a simple testbed using a table to access & then run netlogo procedures via a nick-name.

We use a table called **pmap** which maps {nick-name => task-closure}

So, given 3 netlogo procedures defined as...

```
to-report add2 [a b]
  report (a + b)
end

to-report sub2 [a b]
  report (a - b)
end

to-report sub1 [x]
  report (x * -1)
end
```

We can set up nick-names “plus”, “minus” & “neg” which map to task *closures* for these as follows...

```
to set-map
  set pmap table:make
  table:put pmap "plus"  (task add2)
  table:put pmap "minus" (task sub2)
  table:put pmap "neg"   (task sub1)
end
```

To run/test the use of nick-names we build the following...

```
to call [fname arg1 arg2]
  set-map
  print (runresult (table:get pmap fname) arg1 arg2)
end
```

Results are as follows...

```
observer> call "plus" 4 5
9
observer> call "minus" 4 5
-1
observer> call "neg" 4 5
-4
```

comments

- mostly functions as expected – no problem storing accessing task closures;
- over-supplying arguments to task closures is ok (extra args are ignored);
- under-supplying args to closures (not shown in above tests) generates errors;

further investigation

- how (most sensibly) to build a call mechanism which takes a variable number of arguments.

full Netlogo code

```
extensions [table]

globals [pmap]        ;; pmap is (name -> proc) map

to set-map
  set pmap table:make
  table:put pmap "plus"  (task add2)
  table:put pmap "minus" (task sub2)
  table:put pmap "neg"   (task sub1)
end

to call [fname arg1 arg2]
  set-map
  print (runresult (table:get pmap fname) arg1 arg2)
end

;-------------------------------
; sample callers
;-------------------------------

to-report add2 [a b]
  report (a + b)
end

to-report sub2 [a b]
  report (a - b)
end

to-report sub1 [x]
  report (x * -1)
end
```