

'C'ing is Believing?
Anthony Dardis, Hofstra University
July 3, 1996

It seems to me that Fodor's account of a sufficient condition for a symbol 'C's meaning, say, tokens of the letter 'a', in a particular system, is incorrect, since the condition can be satisfied even though tokens of 'C' have no meaning. The argument is a computer program that has the causal structure Fodor specifies.

Data Structure:

a list of rules, each of which is a pair of letters, eg
aC
bD
eG

Two constraints are put on the list:

- i. for any letter, there is no more than one rule which has that letter as its second element;
- ii. if a rule is deleted from the rule set, no rule can be added which has the same second letter as the deleted rule. (This is enforced by the user-interface, say.)

Procedures:

- (p1) an input routine that reads characters from the keyboard
- (p2) a routine that at random intervals picks a subset of the rules; this subset will be used by routine (p3, part b)
- (p3) an output routine that runs when a letter is typed, that
 - (a) usually checks to see whether the new character matches the antecedent (ie the first character) of any of the rules; if it does, then it outputs the second character of the rule
 - (b) but every once in a while (say, randomly, for about 5% of the characters input) picks the second character of one of the rules in the subset of the rules generated by procedure (p2) and outputs that instead. This is the "error generator."

I submit the following are true, supposing that 'aC' is initially one of the rules:

1. "'a's cause 'C's" is a law
2. an 'a' actually causes a 'C'

3. For all letters Y which are not 'a's, if Y actually causes a 'C', then Ys causing 'C's is asymmetrically dependent on Xs causing 'C's
4. the output of this program is meaningless.

Comments:

(1.) 'a's cause 'C's as long as the 'aC' rule is in the rule set and (p2) functions correctly. (If there is any sensible way to interpret Fodor's condition 1. then (??) it's no more unlikely that it's a law that 'a's cause 'C's than that it's a law that cows cause cow thoughts in me.) The program could token the output of (p2) as some kind of intermediate stage rather than output it directly; that way, the tokens of 'C' seem more "internal."

(2.) someone has to type an 'a'.

(3.) is made true by the structure of procedures (p2) and (p3). The rules and procedure (p3) ensure a causal connection between input and output, so 'a's cause 'C's. The 'C's that are not caused by 'a's are output because the rule is present (hence because 'a's cause 'C's); so if the rule were absent then nothing would cause 'C's (since the rule list can't be updated with a new rule that has 'C' as its second element). The part of asymmetric dependence that says if non-'a's didn't cause 'C's then 'a's would still cause 'C's is ensured by the random subset selector procedure (p2): sometimes the subset won't include the 'aC' rule, and then no 'C's will be output by the "error generator," but that has no effect on procedure (p3, part a), which (usually) outputs a 'C' when an 'a' is typed in.

(4.) I won't argue for this.

If you are sympathetic to Block's "liberalism" complaint about functionalism, and to Putnam's hi-tech version of it in the appendix to Representation and Reality, then you will accept that any thing satisfies any functional role you care to specify. So these 'C's have the functional role of beliefs. So 'C'ing is believing!