

Using Couplets to Understand Pursue and Consume Programs

David S. Touretzky
Version of May 15, 2017

When a page contains multiple Pursue rules and multiple Consume rules, how can we predict the order in which objects will actually be consumed? One way is by identifying the pursue-and-consume “couplets” in the rules. Here is how to find the couplets in the Star2 world. Start with the rules for Star2:

- [1] WHEN see **heart** DO move toward
- [2] WHEN bumped **heart** DO eat it
- [3] WHEN see **coin** DO move toward
- [4] WHEN bumped **coin** DO eat it



Step 1: Mark the pursue rules with “P” and the consume rules with “C”.

- P [1] WHEN see **heart** DO move toward
- C [2] WHEN bumped **heart** DO eat it
- P [3] WHEN see **coin** DO move toward
- C [4] WHEN bumped **coin** DO eat it

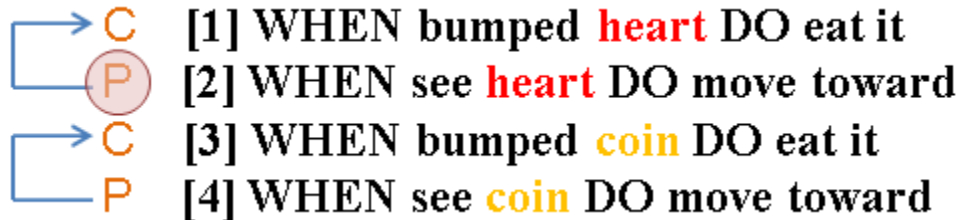
Step 2: Draw an arrow from each Pursue rule to the matching Consume rule.

-  P [1] WHEN see **heart** DO move toward
- C [2] WHEN bumped **heart** DO eat it
-  P [3] WHEN see **coin** DO move toward
- C [4] WHEN bumped **coin** DO eat it

Step 3: Find the earliest Pursue rule. That object will be pursued first.

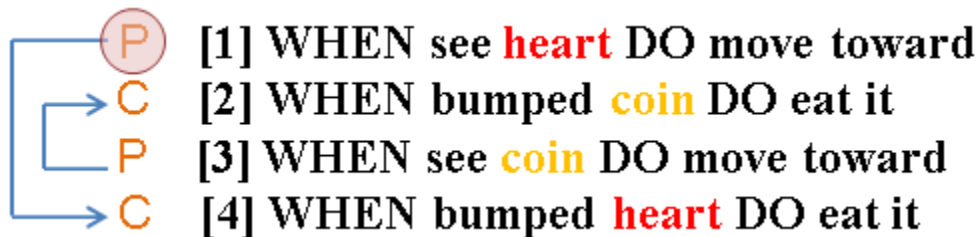
-  **P** [1] WHEN see **heart** DO move toward
- C [2] WHEN bumped **heart** DO eat it
-  P [3] WHEN see **coin** DO move toward
- C [4] WHEN bumped **coin** DO eat it

The Second Law of Kodu tells us that switching the order of the rules in a couplet has no effect on the character's behavior: Consume-and-Pursue programs behave the same way as Pursue-and-Consume ones. The only difference is that the arrows run the other way when we draw the diagram. It still holds that the earliest Pursue rule determines the first object that will be pursued and consumed: hearts in this case.

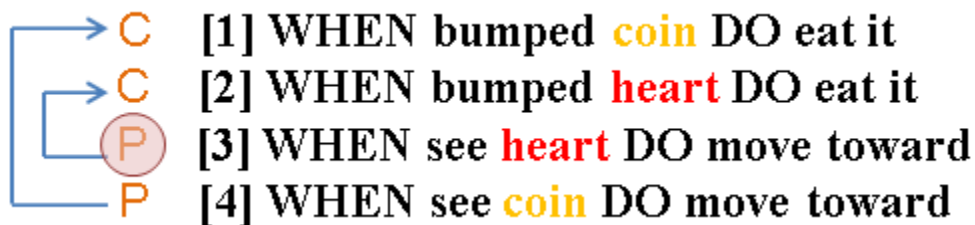


The Third Law of Kodu tells us that the earliest Pursue rule will always win until there is nothing left for it to pursue. Then the next Pursue rule can take effect. In this example, only when all the hearts are gone will the character pursue coins.

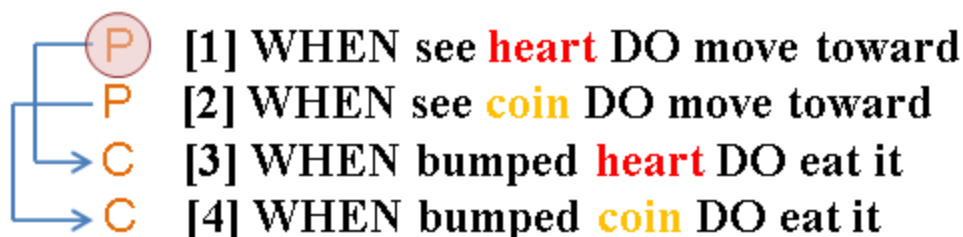
Again, it's the order of the Pursue rules that matters. The Consume rules can appear anywhere. The version below behaves the same as the previous versions, because the Pursue rule for hearts still comes first. Notice that the arrows now run in opposite directions; this makes no difference.



Here's a version where all the Consume rules come first and the Pursue rules come last. But again, all that matters is which is the earliest Pursue rule.



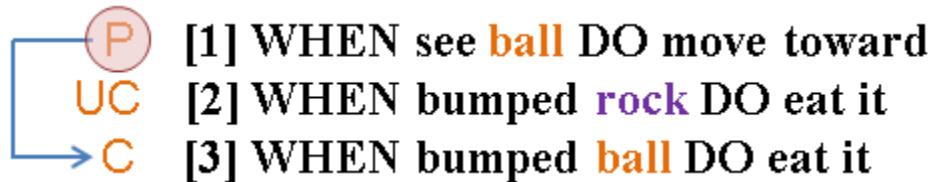
The arrows can even cross. It makes no difference.



Reasoning About Tricky Cases

Case 1: Unpaired Consume rules (due to a missing Pursue rule) will starve.

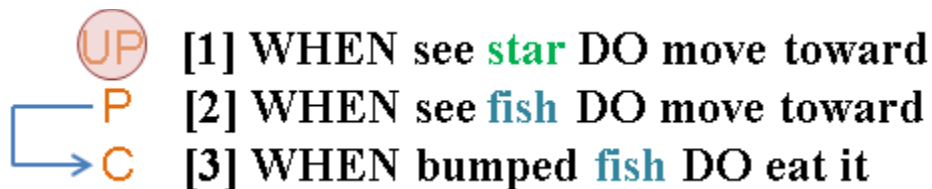
If a Consume rule has no matching Pursue rule, put a “U” before the “C”, indicating that it is an Unpaired Consume rule. The object mentioned in that rule will never be reached, so it will never be consumed. In the example below, the character will eat all the balls but it will ignore the rocks.



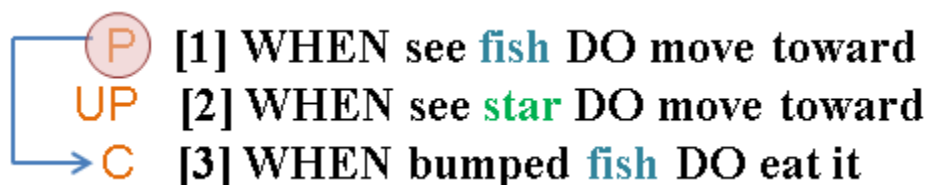
Actually, it can still eat a rock if it bumps into one by accident. But if that accident doesn't happen, rule 2 will “starve” because there is no Pursue rule to feed it.

Case 2: An unpaired Pursue rule (missing Consume rule) will leave the character stuck at the first thing that rule pursues.

If a Pursue rule has no matching Consume rule, put a “U” before the “P”, indicating that it is an Unpaired Pursue rule. This rule will take the character to the closest matching object, but if there is no Consume rule, the closest matching object never changes. So the character gets stuck there.



If the Unpaired Pursue rule is not the earliest Pursue rule, the character won't get stuck right away. It will pursue and consume the first kind of object normally. But when it runs out of those and begins pursuing something for which there is no consume rule, it will get stuck.



Case 3: Shared Consume rule (fed by two Pursue rules).

Makes the program shorter but doesn't affect behavior, which is controlled by the Pursue rules. (Details will be supplied in a later version of this document.)

Case 4: Shared Pursue rule (one Pursue rule feeding two Consume rules).

Both types of objects will be pursued at the same time, so the order can be mixed. Also, allows for two different consume actions, e.g., eat red apples and squash blue ones. (Details will be supplied later.)