Pablo,

We have been working to define what extensions/enhancements we would need to `dol[o/ark/ang]` to allow us to represent the structure of all the models we want to offer solution tools for in HARK.

The goals are:

1. Generality
    - We hope to define a syntax/structure rich enough to handle anything that qualifies as a discrete-time Bellman problem with both discrete and continuous states (or, at least, rich enough that it can eventually be extended to handle any such)
2. Flexibility
    - We want users to be able to hand-code whichever parts of the solution process they like (obviously, they *must* hand-code any parts that cannot be handled by the built-in solution methods
    - In principle, this allows the framework to handle any problem, even ones that built-in solvers will fail at
3. Modularity
    - It should be as easy as possible to swap in or out different model components
        - e.g., maybe you have a portfolio choice only once every third year

As best we can see, all of these objectives could be accomplished by extending dol[o/ark/ang] in a few specific ways.

The most important is that the your dolang yaml file defining a period of a problem would be modified to be able to represent a 'stage' of a problem:

- in which the end of period ('EOP') and beginning of period ('BOP') state are allowed to differ
- in which there are no `definitions` (those should be defined at the level of a 'period'

The of course would require the user to guaranteee that when they append a further (backwards) element to the existing list of solutions, the new element is guaranteed to generate the state variables needed by its successor.

What we need from Dolo:

1. A syntax for defining EOPand BOPvariables (or inheriting those definitions from a namespace)
    - $EOP_t \leftrightarrow BOP_{t+1}$

  - It is necessary to allow this, but not necessary to require it. That is, infinite horizon models with an identical problem in successive steps could omit explicit definition of EOP and BOP variables and the code would interpret their absence as a requirement that the problem is identical in the next step

2. Transitions can occur without a timestep increment
   - These correspond to multistage problems, like with labor, consumption, portfolio choice ...

3. Variables within a time period cannot reference those in another time period
   - This seems necessary for modularity

We want to construct both the solution and the simulation components of the problem as we move back from a final moment.

1. Backward/Expectation/Optimization: We are solving for chosen behavior giving beliefs (expectations) about the future

2. Forward/Experience/Simulation: We are experiencing the realization of the shocks that determine our states and making the precalculated choices

1. There are three types of stages:
   a) `transition` - (short '`trn`')
      - Evolution in chrontime from end of a period to the beginning of the next ('tick')
      - Example: a year of life
      - Backward: structure of BOP_{t+1} states creates structure of EOP_t states
        - e.g., k[t+1] grid defines a[t] grid
      - Forward: structure of EOP_t states creates structure of BOP_{t+1} states
        - e.g., a[t] distribution becomes k[t+1] distribution

   b) `optimization` - (short '`opt`')
      - Constructs the agent's choices

   c) `exogenous` shocks - short `exo`
      - Backward: Compute expectations required for model solution
      - Forward: Propagate distributions according to eqns of motion