Homework 5—Build a More Complex Cognitive Tutor - Part 2
05-432/05-832 Personalized Online Learning, Fall 2018
Due: Fri, Nov 9, 11:59pm (via Canvas)
Maximum points: 70

Resources downloadable from Canvas:

- Some background material for learning about the task domain
- Two versions of the tutor interface (bug we saw has been fixed) ☺ One version is for offline use, which may be handy at times.
- Behavior graphs to be used for learning the task and testing the cognitive model (note: these graphs do not exhaustively enumerate all solution paths)
- **A** partial model, together with commentary that gives suggestions for how to extend the partial model.

You might simply replace your old package with the new package that is provided. It has everything that the old package has, and more.

Step 2: Complete the model

Complete the incomplete model that is provided so that the tutor can recognize *all* possible solutions for the two problems (foxglove and lupine)***, not just those in the behavior graphs.***

The incomplete rule-based tutor that you start with does no more than accept the strains to be crossed. It does not do any abductive reasoning about genotypes and does not accept Partial Conclusion or Final Conclusions – it is your job to implement the rules for this purpose. You get to do the fun part. The templates and bootstrap rule are complete, but the rule set is not.

Start by studying the description of the model provided below. See if you can follow this structure as you run the partial model that is provided. See if the Conflict Tree makes sense to you at each step and whether you can map it to the model's subgoal structure, described below. For this purpose, you will find it helpful to turn on tracing of the conflict tree using console command setTracerLogFlag("conflict_tree"). Tip: A good strategy for getting to see the whole Conflict Tree is to (deliberately) enter incorrect input. The model tracer will have to generate the whole tree as searches for possible model-generated observable actions that equal the student's.

Then write the missing rules based on your cognitive task analysis. The foxglove problem needs fewer rules than the lupine problem, so you could work on that first.

How to structure the model

The model captures reasoning about three strains' genotypes based on the results of a series of crosses (i.e., the phenotypes of the offspring). The student can select the crosses in any order. ***For each cross, the model works through four subgoals in order:***

1. select-cross
   The student enters the two strains to be crossed. The tutor prints the phenotypes for the offspring. Observable actions by the student: entering the first and second strain of the cross. Tutor-performed action: tutor displays the result of the cross in the interface.
2. infer-from-cross
   a. The tutor generates the inferences that can be drawn just from the information gathered about the last cross, and just about the two strains being crossed; these inferences are the rules for which you did the cognitive task analysis, see above;
3. make-inferences
   a. generate additional inferences, now also potentially involving the third strain; these inferences are based on the partial and final conclusions that have been drawn so far;
4. enter-inferences
   Observable actions: entering any partial and full conclusions that can be drawn

- Each subgoal can extend across multiple steps; conversely, one step can involve more than one subgoal. It is important that you understand the relation between steps and subgoals (i.e., which subgoal happens at which step). By "step" we mean a problem step with a single observable action (i.e., the usual meaning of this term). You can study these relations even with the partial model that has been provided. A good way might be to add console.log() messages on the rhs of those rules that change the subgoal, and then work your way through a couple of cross sequences. As part of the deliverables (see below), you are asked to annotate a couple of paths in the given behavior graphs with the subgoals.
- The abductive reasoning, that is, generation of partial and final conclusions related to the genotype of each strain, happens under subgoals #2 and #3. We also use the term "inferencing" to refer to this abductive reasoning. When the student enters a new cross, all inferencing happens *before* the student enters any new partial and final conclusions. These inferences are stored in working memory. A set of "write" rules models the actual entering of these inferences on the interface, drawing on the stored inferences. The model was structured in this way so it can "know" when the student is done with the inferences for the current cross (as is done by the doneWithMandatoryInput rule that is provided). Frankly, structuring the model so that it can enforce entering all interferences for the current cross before moving on to the next cross is a surprisingly difficult modeling challenge.
- What should the Conflict Tree look like for the step of entering the second strain of the next cross? The answer to this question may surprise you. This chain looks as follows, as you might verify in the running tutor:

    - one rule activation for accepting the second strain (given in productionRules.nools)
    - one rule activation for making inferences directly from the offspring (i.e., an "inferFromCross" rule that you will need to write)
    - zero or more rule activations for making additional inferences (i.e., the "makeInferences" rules that you will need to write)
    - activation of a stop rule (doneInferencing, given in productionRules.nools); this rule is needed to end the chain
- What's unusual is that the *first* rule activation in this chain generates the observable action of entering the second strain in the interface. Normally, the *last* rule in a chain generates the observable action – the model tracer stops chaining when it finds a rule that specifies an observable action. So how can this be? As you might verify, the rules that accept the second strain do not call halt().

You need to implement the following rules:

- 6 rules for drawing inferences directly from the result of crossing two strains (the inferFromCross rules);
- 5 rules for drawing further inferences (the makeInferences rules)
- 5 rules for entering partial and full conclusions (the enterInferences rules);
- 1 done rule

Questions regarding how working memory has been structured

- How does the model represent the different subgoals listed above? I.e., how are they represented in working memory?
- How does the model represent the inferences that can be made about the plant strains under study, that is, the five different partial and final conclusions listed in MendelBackwardsV4.ppt p. 13 and 14, which are the same ones you can see in the interface, in the menus for "Partial Conclusions" and "Final Conclusions"?
- Why is it useful to have facts of type Cross?
- How does the model distinguish between conclusions (partial or final) that have been inferred but not written versus those that have both been inferred *and* written?
- Why is it important to make that distinction?
- How are the interface elements represented in working memory and how are they linked to the problem fact?
- Why does fact type Cross have both an offspringExt and an offspringInt slot? What do these slots have in common and how are they different from each other?

Extra credit (7 points)

Create a new problem involving wild flowers or other plants. The new problem should involve a plant species different from those in the given problems, for which it is reasonable to assume that color variations are controlled by a single gene.

Deliverables – please submit through Canvas

- Your tutor – please provide all tutor files within a single .zip
- A short report with:
    - Evidence that your tutor can handle *all* solutions to the two problems provided (foxglove and lupine). I.e., for each problem, show two screenshots of completed problems, including at least one solution for each problem that is not in the behavior graphs that are provided.
    - Documentation for the model
        - Behavior graphs for at least two problems with at least one path (start to done) annotated with the subgoals (i.e., each link should be annotated with the subgoals that are covered during the model's processing of that step) – no need for hi-fi superglossy diagrams, just keep it simple
        - Lo-fi sketch of a diagram that show working memory structure
        - English versions of all rules, included in the rules file
        - Comments on the meaning of all the facts and their slots, included in the types file
        - Answers to the "questions regarding how working memory has been structured" above
        - Answers to the
    - If you added a new problem, please also provide screenshots of two ways of solving it. Please also provide a listing of sources for the genetics knowledge on which the new problem is based.
    - Reflection on the strengths and limitations (in a pedagogical sense) of the tutor you built.
    - A few sentences on how you worked together as a team
    - What was hard about this assignment?
    - Gripes and raves about CTAT (this helps us make CTAT better)
- Please include all screen shots and diagrams within the report file, not in separate files.

Testing of your tutors by the course instructor

To test your tutors, I may run them with variations of some of the paths in the behavior graphs that were provided (e.g., same steps, different order) as well as one or two new paths for the current problems. Most likely, I will also include a test case of a different problem (i.e., different set of genotypes) that can be solved with the same rules, to test the generality of your model. I may do other things as well.

Suggestions for how to work together as a team

Given you work together with a student who is likely to have a different background, there is an opportunity to learn from each other. Please see it as your job not just to finish the assignment, but also to learn from each other (and help the other learn from you. The following are suggestions for how to divide up the work in a way that would provide many opportunities for learning from each other.

We suggest creating somewhat different roles for the T and E person on the team. As discussed in class, the T person has more technical experience, the E person has more experience in education, (instructional) design, psychology, business, and/or related fields. We realize that everyone is at least a bit of a T and a bit of an E.

E could take the lead on

- Domain explorations – reading up on Mendelian genetics, to the extent needed
- Cognitive task analysis and formulating English rules
- Creating test scenarios to be used during development and testing
- Testing whether the model works on held-out test scenarios (i.e., a subset of test scenarios not used during development; maybe T does not even know about these scenarios until the model is ready for testing)
- Testing whether the model has the required behavior, as specified in the assignment (in particular, that is does not let the student enter the next cross before the inferences from the previous cross have all been entered)
- Documentation (see under deliverables); in particular providing English versions for the actual Nools rules (these are likely to differ somewhat from those written for the initial cognitive task analysis)

E should also write and debug a small number of rules (one for two different subgoals), mentored by the T person. The idea is that E will learn to understand the model well, even if T does the lion's share of building it.

T could take the lead on

- Explore how the current partial model works
- Implement and debug new rules

Consult with E on documenting the rules; however, it may be better is E is the one who is doing the documenting, as this will force E to really understand the rules even if T is doing most of the writing; T could then give feedback

Suggested joint tasks:

- Going over the English rules created in cognitive task analysis (or even generating them together)
- Answering the "understand" questions
- Writing a small number of rules together

If you are both strong Ts, then perhaps the division of labor suggested above is not so useful. You could, instead, divide up the work by subgoal. However, we would still recommend working together on the domain exploreations and cognitive task analysis, as well as doing some pair programming or other joint work. For example, you might use a joint work session to write the first one or two rules for each subgoal, and divide and conquer from there. You might also make sure you systematically test each other's work.

Some things the model does not do (ideas for possible course projects here!)

- Showing pictures of the flowers in the interface would be a nice touch!
- After the student has entered a cross, the model does all the inferences that can be drawn in the same "cycle" that the second strain for the cross is entered. By contrast, the student might make them cycle by cycle, so model and student are not fully in synch. Structuring the model in this manner is OK, but a downside is the model might need some additions to give good hints and do accurate knowledge tracing. That is not part of the assignment, however.
- The model does not model a strategy for *selecting* the next cross that is most likely to be informative. Maybe an effective a strategy can be found, maybe not. It would be neat to study log data from the current tutor to see if students, across a sequence of tutor problems, develop strategies. (Potentially, there is such log data in DataShop.) Would be neat to figure this out and make the tutor capable of tutoring strategy.
- The model also does not model the simulation rules for determining the result of crossing each pair of strains (i.e., the phenotypes of the offspring). A more complete tutor would also have rules for calculating the offspring. But the abductive reasoning rule are more interesting/challenging, which is why we focused on those rules.
- Maybe it would help students who use this tutor if the tutor made reasoning about the underlying genetic processes more explicit – since learning those processes is the real learning goal. How to integrate such reasoning in the current tutor is an interesting challenge.