

# Taking Control of Redundancy in Scripted Tutorial Dialogue <sup>1</sup>

Pamela W. JORDAN <sup>2</sup>, Patricia ALBACETE and Kurt VANLEHN  
*Learning Research and Development Center, University of Pittsburgh*

**Abstract.** We describe extensions to a finite-state dialogue manager and its author scripting language that enable control over content repetition during tutorial dialogue. The problem of controlling redundancy requires the specification of complex conditions on the discourse history that are beyond the capabilities of the intended users of the scripting language. We show that the problem can be addressed by adding semantic labelling and the marking of optional steps and difficulty levels to the scripting language and heuristic algorithms to the dialogue manager.

**Keywords.** Dialogue Management, Authoring tools

## 1. Introduction

One of the many challenges in building intelligent tutoring systems that interact with students via natural language dialogue is selecting a dialogue management approach for which course content can be easily authored by non-technical users while still maximizing adaptability to the context. Our initial approach to dialogue management in the WHY-ATLAS tutoring system [13] focused on simplifying the authoring task and can be loosely categorized as a finite state model. Finite state models are appropriate for dialogues in which the task to be discussed is well-structured and the dialogue is to be system-led [8] as was the case for WHY-ATLAS. This initial approach proved successful with regard to ease of authoring [4,10] but ultimately proved not to be sufficiently sensitive to the discourse context.

Instructors use a scripting language [4] to author the content and form of the finite state network that represents the space of dialogue moves the tutor can make given the student's response to a previous move. The scripting language allows the author to specify a multi-step hierarchically-organized recipe (a type of plan structure defined in AI planning) for covering a topic or part of the problem solving for which the student needs help. Recipes are higher level goals that are defined as a sequence of any combination of primitives and recipes [16]. In WHY-ATLAS a step which is a primitive is a state in the finite state network. The author also specifies classes of answers to any tutor questions asked in a primitive and appropriate state transitions for each class of answer. Thus each answer is an arc between two states in the network. By default, if no transition is specified for a step then the transition is to the next step in the recipe. Authors can label

---

<sup>1</sup>This work was funded by NSF grant 0325054 and ONR grant N00014-00-1-0600.

<sup>2</sup>Correspondence to: 3939 O'Hara Street, Pittsburgh PA, 15260, USA; E-mail: pjordan@pitt.edu.

primitives with goals but the scripting language does not distinguish state transition information from goal information at the primitive level. This means that the goal labels for primitives must be unique if the originally scripted line of reasoning is to be recovered on-demand from the network.

When an author scripts dialogues to support tutoring for multiple problems that the student is to solve, the author should not pre-suppose what will have been seen by the student or how well the student responded to previous questions. Such reactions need to be encoded as conditions on the discourse context. However, adding conditions to the scripting language moves the authoring task closer to a programming task and potentially makes it too difficult for many instructors.

While we initially chose to ignore the need to specify more complex conditions on the context in order to make the task one that any instructor is likely to be able to do, the trade-off is redundancy in the material discussed with a student. Since students work on multiple problems during tutoring and all these problems share some subset of domain concepts with at least one other problem, a student might see similar content many different times. Although redundancy can be beneficial, if used inappropriately it can be detrimental to attention and to the quality of the solutions produced during problem solving [14,6].

During reviews of the WHY-ATLAS transcripts<sup>1</sup>, we found that when the system repeats content (whether in the same problem or across problems), students will often still answer but will also additionally append insults or displays of annoyance (“up, you idiot”, “same, like I said”), or expressions of confusion (“i don’t know what u want me to say.”). Or they may suspect that they are being misunderstood and try to solve the problem by doing such things as oversimplifying responses (“lightweight car massive truck patch frictionless ice head-on collision vehicle impact force greater change motion”). At other times they simply stop answering (“I don’t know” or null response). The loss of motivation and the loss of credibility of a tutor are expected to have some detrimental effect on learning.

Our solution for controlling redundancy is to share the task of specifying conditioning on context between the author and the dialogue management software by making the author’s added task one of labelling rather than of programming. Authors are asked to optionally label dialogue moves with similar content with a consistent labelling scheme of their own choosing and mark the difficulty level of a move relative to those with similar labelling. Given this additional information and heuristic algorithms, the dialogue manager has the additional information it needs to more wisely use redundancy. It can now check the dialogue history for previous uses of a label and find out how often the content has been presented and how well the student responded in each of those cases. It allows the dialogue manager to either skip moves or to select moves that are more or less challenging based on the student’s previous performance on that same labelled move. This addition is similar to what is suggested in contingent tutoring [15].

In this paper, we focus on the changes we have made to the scripting language and to the dialogue manager. First we review the WHY-ATLAS system and the old scripting language and dialogue manager. Next we describe the extensions to the scripting language and how the dialogue manager uses this additional information to provide additional conditioning on the context. During the discussion we show two examples of op-

---

<sup>1</sup>We reviewed 110 system-student dialogue sessions in which one session covers one physics problem. All quoted examples are verbatim from these transcripts.

tionally enhanced scripts and how they adapt to the context. We conclude with a preliminary evaluation of instructors' ability to use the extended scripting language and our plans for evaluating the effectiveness of the resulting dialogues.

## 2. The WHY-ATLAS System

Question: Suppose you are running in a straight line at constant speed. You throw a pumpkin straight up. Where will it land? Explain.

Explanation: While I am running, both the pumpkin and I are moving with the same speed. Once I throw the pumpkin up, it no longer has anything to thrust it forward in the either the horizontal or vertical direction. Therefore, it will fall to the ground behind me.

**Figure 1.** The statement of the problem and a verbatim student explanation.

The WHY-ATLAS system covers 5 qualitative physics problems on introductory mechanics. When the system presents one of these problems to a student, it asks that she type an answer and explanation and informs her it will analyze her final response and discuss it with her. One of the problems WHY-ATLAS covers is shown in Figure 1 and the student response shown is a first response from our corpus of students' problem-solving sessions. The student response in this case is an instance of the often-observed *impetus* misconception: If there is no force on a moving object, it slows down. In a majority of student responses, the only flaw is that the response is incomplete. Details about how the essay is analyzed are addressed in [7,5,9] and are beyond the scope of this paper.

### 2.1. Dialogue Management

Given the results of the essay analysis, which is a list of topic labels, the WHY-ATLAS dialogue subsystem leads a discussion about those topics. It uses a robust parsing approach (CARMEL [11]) to understand the student's input and match it to the expected inputs, and a reactive planner (APE [2]) to manage the dialogue where choosing the next dialogue move depends upon the student's answer.

There are 3 types of dialogue recipes that were scripted for WHY-ATLAS; 1) a walkthrough of the entire problem solution, 2) short elicitations of particular pieces of knowledge or 3) remediations. Walkthrough recipes are selected when the student is unable to provide much in response to the question or when the system understands little of what the student wrote. Short elicitations are selected if the student's response is partially complete in order to encourage the student to fill in missing pieces of the explanation. Remediations are selected if errors or misconceptions are detected in the student's response to the question. During the course of the top-level recipe type selected, pushes to recipes for subdialogues that are of the same three types (i.e. walkthrough, elicitation or remediation) are possible but typically are limited to remediations.

After the discussion based on the top-level recipe is complete (may have pushed to and popped from many recipes for subdialogues during the course of the main recipe), the system will either address an additional fault in the essay or ask that the student revise her explanation before moving on to any other flaws already identified. The cycle of explanation revision and follow-up discussion continues until no flaws remain in the student's most recent essay.

## 2.2. Dialogue Scripts

Dialogues are represented as finite state networks with a stack (i.e. a pushdown automaton). States correspond to primitives that produce tutor utterances, arcs correspond to correct student responses or cases in which no response is expected, and pushes to vague or incorrect student responses. Pushes call a subdialogue and pops return from one.

The scripting language defines primitive actions and recipes. A primitive is defined to be a tutoring goal that is a leaf node in a plan tree and an associated natural language string that realizes that primitive tutoring goal. A primitive may encode a tutor explanation or a question for eliciting a particular piece of information or both.

Recall that recipes are higher level goals that are defined as a sequence of any combination of primitives and recipes [16]. This representational approach is widely used in computational linguistics since problem-solving dialogues and text are believed to be hierarchically structured and to reflect the problem-solving structure of the task being discussed [3]. Tutorial intentions or goals should be associated with both recipes and primitives. In this way, the author may encode alternative ways of achieving the same tutorial intention.

For each primitive tutoring goal, the scripting language also includes information on what to expect from the student so that information on how to respond appropriately to the student can also be included in the script. Possible student responses are categorized as expected correct answers, vague answers and a set of expected typical wrong answers. For completeness, the author is expected to always include a class for unrecognized responses as well. Every vague and wrong answer and the default class have associated with them a list of tutorial goals that must be achieved by the dialogue manager in order to respond appropriately to that answer class.

## 3. Controlling Redundancy

What is redundant depends on the student's history so the goal is to adequately track content across all tutoring sessions with a student. We have added three types of optional information to the scripting language that will help with tracking content and controlling redundancies: 1) semantic labels 2) optional steps within a multi-step recipe 3) difficulty levels for recipes and primitives. We will discuss each in more detail below.

### 3.1. Semantic Labels and Optional Steps

The original scripting language denigrates the goal labels for primitives with respect to their planning origins by collapsing goal labels and arc pointers. This was done mostly because authors had difficulty associating a goal with every step and found it easier to think of these labels as pointers. But an arc pointer is limited to a specific state while goals are meant to be relevant to multiple states. Thus not only is the power of multiple ways of achieving a goal lost at the primitive level so is the knowledge that primitives from different recipes may cover similar content.

Because the dialogue manager does not have any information on the meaning of the content encoded in the network, it cannot detect repetitions with sufficient reliability to reduce repetition or possibly even push the students with increasingly more challenging

questions. So while the dialogue manager does track the dialogue history by recording 1) what has been contributed to the dialogue by both the student and tutor, and 2) the language interpreter's classification of student responses to tutor questions, it does not have access to the meaning of the tutor's turn. So context conditioning is strictly limited to the previous student response, and the dialogue manager can not skip steps that were made obsolete by its own previous actions or earlier student responses.

To solve this problem, we added semantic labels for primitives and recipes so that all primitives and recipes with similar content are recognizable to the dialogue manager and we added markers for optional steps that can be skipped given the proper context. The semantic labels used are up to the author. The author can make the label meaningful to him or not (e.g. *elicit-force* vs *sem1*) but it has no actual meaning to the system. The system only looks for exact label matches between a turn that is about to be delivered and one or more previous turns.

We cannot always skip redundant material because redundancy has a beneficial role to play in task-oriented dialogue. The more relevant roles for tutoring are that it either brings a piece of knowledge into focus so that inferences are easier to make, or emphasizes a piece of knowledge. We also know that for learning, repetition of a piece of knowledge that a student is having difficulty grasping is sometimes helpful. Given these roles, the location of the redundancy with respect to time and how the student previously performed are considered.

As an example, the following script includes semantic labels (i.e. *:sem <label>*) and optional steps (i.e. *:step\**). Here we will assume the remediation recipes each use the same labels for semantics as for goal names.

```
(goal detailed-analyze-forces
  :sem detailed-analyze-forces
  (:step
    "Try to name all the forces acting on the pumpkin after
    it is thrown."
    :answers
    (("gravity")("air resistance" remind-negligible)
     ("$anything else$" help-id-forces))
  (:step*
    :sem help-id-forces
    "Are there any other forces on the pumpkin?"
    :answers
    (("no")("$anything else$" help-id-forces))
  (:step*
    :sem remind-negligible
    "Why is there no force on the pumpkin due to air?"
    :answers
    (("negligible")("$anything else$")))
  (:step "So gravity is the only force on the pumpkin")
  ....)
```

Here the second and third steps are marked as optional. There are two ways in which an optional step can be skipped. The first is if a semantic label is in the immediate discourse history. In the above, the semantic label *help-id-forces* would be in the immediate discourse history if the student's answer in the previous step was not recognized (i.e. categorized as answer class "\$anything else\$") and a push was made to the remediation recipe for that class, *help-id-forces*. The same is true for *remind-negligible*. The second

way of skipping is if the semantic label is in the non-immediate history and the student did well with it when last encountered.

### 3.2. Levels of Difficulty

While we know that repetition of difficult material can be beneficial, it should be gradually adjusted over time so that the student is providing the knowledge with decreasing assistance from the tutor. In addition the tutor could try to help the student achieve a deeper understanding. To address these possibilities, we added the specification of difficulty levels at the primitive and recipe levels to work in conjunction with semantic labels and optional steps. To encode difficulty levels, we use speech act labels to distinguish primitives with the same semantic labels and intent levels for recipes with the same semantic labels.

A speech-act [12] is a type of intention behind an utterance. The two most frequently discussed speech-acts in the literature are inform and request [1]. Inform is frequently realized as a declarative sentence while a request is frequently realized as a question. For tutoring, we further sub-divide the request speech-act by question-type and use the labels “Whyq” for why questions, “howq” for how questions, “ynq” for yes/no questions, and “whq” for all other questions (i.e. when, where, what). An example in which different questions types are used that involve the same concept is: “Why is the pumpkin’s acceleration downward?” vs. “Does the pumpkin have a downward acceleration?” vs. “What is the direction of the pumpkin’s acceleration?” vs. “The pumpkin accelerates downward.”

If primitives with the same semantic label are defined using different speech-act/question-types, the semantic label is already in the discourse history and the student was successful with the previous form, then a “harder” speech-act/question-type is selected (if it is available). The first speech-act/question-type defined for a step is the default one if the semantic label is not yet in the student’s history. Otherwise the question-types are organized by difficulty as follows (howq whyq whq ynq) from hardest to answer to easiest. So if the student always has trouble with the whyq and higher for a particular semantic label then the question-type that is selected (if available) is whq or lower. If the student got the previous question-type right for a semantic label then the selection heuristic will try the next hardest type available. Note that if the student gets it wrong then an easier type will be tried the next time and then a harder one the next if she is able to answer it. If the hardest question-type specified was previously tried and the student got the question right then the student will get an inform if one is available (the assumption is that she must already know this bit of knowledge now).

When multiple recipes have the same semantic label, an intent label indicates the difficulty level of the recipe. In this case, higher numbers indicate increased difficulty and 0 is reserved for a recipe that simply informs. For example, below are two recipes for goal G with the same semantic label *a*, two recipe intent levels (i.e. :intent <level>) and two speech-act/question-types for one step in the second recipe (i.e. :sa <speech-act/question-type>).

```
(goal G
  :sem a
  :intent 0
  "After the object is released, the only force acting on it is
  gravity. This force is called weight and is always present
  when an object is in a gravitational field.")
```

```

(goal G
  :sem a
  :intent 1
  (:step
    "What force is responsible for an object's weight?"
    :answers
    (("gravity")("$anything else$" forces-in-a-freefall-inform)))
  (:step
    :sa inform
    "The force of gravity is always present when an object is in
     a gravitational field such as the one produced by earth."
    :sa whq
    "When is gravity present?"
    :answers
    (("in gravitational field")
     ("anything else$" gravity-near-earth-inform)))
  ... )

```

The first time the recipe for goal G is initiated, label *a* is not in the discourse history. Thus the student gets G with intent level 1 and an inform for the second step which is the default since it is listed first. The second time that she needs G, she will get the intent level 1 version and the whq for the second step. If she needs G a third time, assuming she got all of the steps in the last version of G right, she will get intent level 0 with the assumption being that the content denoted by label *a* is now known and just needs to be brought into focus.

#### 4. Preliminary Evaluation and Conclusion

Two instructors who had previously used the original scripting language were asked to author new material for an upcoming experiment to compare tutoring systems and were asked to try to use the new options to reduce redundancy. Together they authored approximately 350 new recipes and added optional difficulty levels to 11% of these recipes. They also authored 645 new primitives and added semantic labels to 20% of these new recipes and primitives. Finally they marked 4% of the new primitives as optional steps. No optional question types were used. The instructors considered optional question types a low priority and ran out of time before having a chance to try using them.

The enhanced scripts are currently in use in the WHY-ATLAS system and when the current experiment is completed we will analyze these new dialogue transcripts to see if the reactions to better controlled redundancy are neutral as opposed to negative. In future experiments we will compare the learning gains of the enhanced dialogues to unenhanced ones.

We presented an enhanced dialogue manager and scripting language that is sensitive to scripted redundancy in a way that is theoretically beneficial to tutoring. We presented examples of enhanced scripts and discussed how they control redundancy. A preliminary evaluation of the extended scripting language showed that instructors were able to make immediate use of all but one new option. This suggests that we have met our goal of keeping the scripting task from becoming a programming task so that it is still doable by most instructors.

## References

- [1] Philip Cohen and Raymond Perrault. Elements of a Plan-Based Theory of Speech-Acts. *Cognitive Science*, 3:177–212, 1979.
- [2] Reva Freedman. Plan-based dialogue management in a physics tutor. In *Proceedings of the 6th Applied Natural Language Processing Conference*, 2000.
- [3] Barbara Grosz and Candace Sidner. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12:175–204, 1986.
- [4] Pamela Jordan, Carolyn Rosé, and Kurt VanLehn. Tools for authoring tutorial dialogue knowledge. In *Proceedings of AI in Education 2001 Conference*, 2001.
- [5] Pamela W. Jordan, Maxim Makatchev, and Kurt VanLehn. Combining competing language understanding approaches in an intelligent tutoring system. In *Proceedings of the Intelligent Tutoring Systems Conference*, 2004.
- [6] Pamela W. Jordan and Marilyn A. Walker. Deciding to remind during collaborative problem solving: Empirical evidence for agent strategies. In *Proceedings of AAAI-96*. AAAI Press, 1996.
- [7] Maxim Makatchev, Pamela Jordan, and Kurt VanLehn. Abductive theorem proving for analyzing student explanations and guiding feedback in intelligent tutoring systems. *Journal of Automated Reasoning: Special Issue on Automated Reasoning and Theorem Proving in Education*, 32(3):187–226, 2004.
- [8] Michael McTear. Spoken dialogue technology: enabling the conversational user interface. *ACM Comput. Surv.*, 34(1):90–169, 2002.
- [9] Uma Pappuswamy, Dumisizwe Bhembe, Pamela W. Jordan, and Kurt VanLehn. A multi-tier NL-knowledge clustering for classifying students' essays. In *Proceedings of 18th International FLAIRS Conference*, 2005.
- [10] Carolyn Rosé, Pamela Jordan, Michael Ringenberg, Stephanie Siler, Kurt VanLehn, and Anders Weinstein. Interactive conceptual tutoring in Atlas-Andes. In *Proceedings of AI in Education 2001 Conference*, 2001.
- [11] Carolyn P. Rosé. A framework for robust semantic interpretation. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 311–318, 2000.
- [12] John R. Searle. What Is a Speech Act. In Max Black, editor, *Philosophy in America*, pages 615–628. Cornell University Press, Ithaca, New York, 1965. Reprinted in *Pragmatics. A Reader*, Steven Davis editor, Oxford University Press, 1991.
- [13] Kurt VanLehn, Pamela Jordan, Carolyn Rosé, Dumisizwe Bhembe, Michael Böttner, Andy Gaydos, Maxim Makatchev, Umarani Pappuswamy, Michael Ringenberg, Antonio Roque, Stephanie Siler, and Ramesh Srivastava. The architecture of Why2-Atlas: A coach for qualitative physics essay writing. In *Proceedings of Intelligent Tutoring Systems Conference*, volume 2363 of *LNCIS*, pages 158–167. Springer, 2002.
- [14] Marilyn A. Walker. *Informational Redundancy and Resource Bounds in Dialogue*. PhD thesis, University of Pennsylvania, December 1993.
- [15] David Wood. Scaffolding, contingent tutoring and computer-supported learning. *International Journal of Artificial Intelligence in Education*, 12:280–292, 2001.
- [16] R. Michael Young, Martha E. Pollack, and Johanna D. Moore. Decomposition and causality in partial order planning. In *Second International Conference on Artificial Intelligence and Planning Systems*, 1994.