Improving User Taught Task Models*

Phillip Michalak and James Allen

University of Rochester
Rochester, New York

Abstract. Task models are essential components in many approaches to user modelling because they provide the context with which to interpret, predict, and respond to user behavior. The quality of such models is critical to their ability to support these functions. This paper describes work on improving task models that are automatically acquired from demonstration. Modifications to a standard planning algorithm are described and applied to an example learned task model, showing the utility of incorporating plan-based reasoning into task learning systems.

1 Introduction and Related Work

Task models are essential components in many approaches to user modelling because they provide the context with which to interpret, predict, and respond to user behavior. Intelligent tutoring systems (e.g. [1]) need to model the tasks that trainees will perform so that they can determine flawed behavior in order to offer timely and appropriate remediation. Systems that adapt content to users (e.g. [2]) must understand the tasks that the user is trying to perform in order to provide appropriate content when it is needed. Plan recognition approaches (e.g. [3]) attempt to explain and predict behavior by constructing explanations that are consistent with task models and prior observation. The quality of the underlying task models determines their ability to support the interpretation, prediction, and response functions mentioned above. This paper describes work that improves the detail and accuracy of task models that have been learned from demonstration.

Typically task models are painstakingly constructed by hand for each domain, a difficult task in its own right and one complicated by the fact that domains are often incompletely or inaccurately specified through knowledge engineering. This paper describes work based on the Procedure Learning On the Web (PLOW) system ([4]), which continues a recent trend of automatic task model acquisition. Some previous machine learning methods (e.g. [5]) use version spaces to represent demonstration ambiguity and require multiple training examples to eliminate it.

* This material is based upon work supported by National Science Foundation, grant #IIS-0328811, DARPA, sub-contract from West Florida, grant #IHMC-UR-07-01 and support from DARPA with a subaward from SRI International. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of above named organizations.

Non-version space approaches (e.g. [6]) face similar difficulties; multiple examples are required to generalize from specific examples to a level that captures the essence of the training examples. In contrast, the PLOW system requires only a single demonstration to learn a task model because it leverages natural language. A play by play task description given as it is demonstrated provides three key types of information unrecoverable by observation alone: task parameterization information, sub-task structure, and semantic annotation for observed action.

In addition to their primary role in task execution, PLOW's task models can also serve as the basis for various user modelling techniques. This paper describes a mechanism for improving task models to better support those roles. Specifically, this mechanism discovers missing precondition and effect information in the task representations and relaxes the implicit total order on task steps that arises from a sequential demonstration. The Diligent system ([7]) recovered similar information through a simulated experimentation approach, but the plan based approach yields finer grained detail when primitive task models are known a priori. Section 2 describes the method by which missing information is recovered from models learned by PLOW, and Sect. 3 provides a brief example of this method.

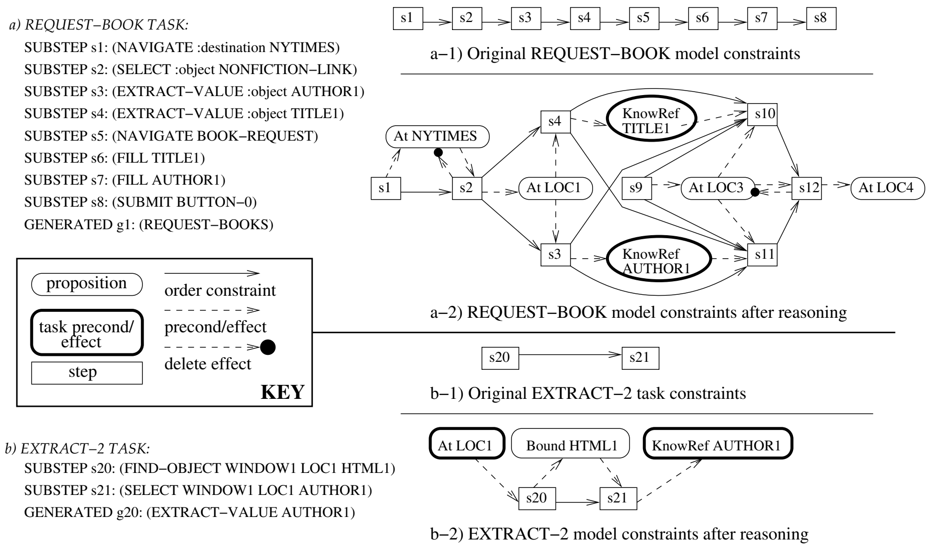
2 Improving Learned Task Models

This section describes a mechanism for improving task model quality and thus ability to support user modelling functions. Each PLOW task model is translated into a specially crafted planning domain and then analyzed by a modified Graphplan [8] algorithm which recovers missing information about necessary task preconditions, effects, and order constraints. This process is a recursive one; improved task models are used in the analysis of higher level task models.

The constants, initial facts, goals, and operators of a specially crafted planning domain are extracted from each (sub-)task model. The planning domain constants come from a task model's constants and parameters. Task model parameters are treated as constants because this analysis never considers a particular parameterized instance; no values are ever assigned to the parameters during reasoning. Task model constants and parameters typically result from spoken and web browser interaction (e.g. typing "http://www.nytimes.com/" into an instrumented web browser or saying "Put the title of the book here").

The planning domain initial facts come from task model preconditions. Essentially, the reasoning algorithm assumes that the task model preconditions are true, and attempts to construct plans that achieve its goals by using the steps specified in the task. Accordingly, the effects and the steps of a task model are formulated as planning domain goals. Treating the steps as goals forces the planner to consider only plans that include those steps.

The operators of the planning domain are summaries of the task models that the system knows about. Specifically, those task models that achieve one or more steps of the task being analyzed are included as domain operators. Operator definitions take task parameters, preconditions, and effects to be their variables, preconditions, and effects, respectively.



a) REQUEST–BOOK TASK:
SUBSTEP s1: (NAVIGATE :destination NYTIMES)
SUBSTEP s2: (SELECT :object NONFICTION–LINK)
SUBSTEP s3: (EXTRACT–VALUE :object AUTHOR1)
SUBSTEP s4: (EXTRACT–VALUE :object TITLE1)
SUBSTEP s5: (NAVIGATE BOOK–REQUEST)
SUBSTEP s6: (FILL TITLE1)
SUBSTEP s7: (FILL AUTHOR1)
SUBSTEP s8: (SUBMIT BUTTON–0)
GENERATED g1: (REQUEST–BOOKS)

b) EXTRACT–2 TASK:
SUBSTEP s20: (FIND–OBJECT WINDOW1 LOC1 HTML1)
SUBSTEP s21: (SELECT WINDOW1 LOC1 AUTHOR1)
GENERATED g20: (EXTRACT–VALUE AUTHOR1)

Fig. 1. Task models for (a) requesting the purchase of a best-selling non-fiction book and its sub-task (b) extracting the author name from a web page, before and after analysis

The resulting planning domain represents the basic constraints of the task model: that the specified steps occur and that the necessary effects hold after task completion when the task preconditions are assumed to be true.

The planning algorithm takes as input standard type, initial condition, goal, and operator descriptions derived from task models as described above. Extensions to the original Graphplan algorithm infer missing information about preconditions, effects, and operator order, as well as rate the quality of each solution. The plan graph of Fig. 2 depicts the minimum cost solution for the author name extraction sub-task of Fig. 1-b; it will be used to illustrate the extensions described below.

The algorithm first augments the planning domain's initial facts to include all facts that enable sub-step achieving operators. This operation guarantees that all of the task sub-steps are achievable, but possibly at the expense of domain integrity. For example, Fig. 2 shows that the additional preconditions (shaded) enable a text selection action in the first time step, despite the fact that it should occur *after* the object finding action.

The second extension modifies the search algorithm to terminate once an acceptable *set* of solutions has been found or it can be guaranteed that none will ever be found. Intuitively, invalid solutions that are enabled by augmenting the initial preconditions should be pruned; the search algorithm assigns a cost to each solution based on the number of *abductive* preconditions that it uses, and

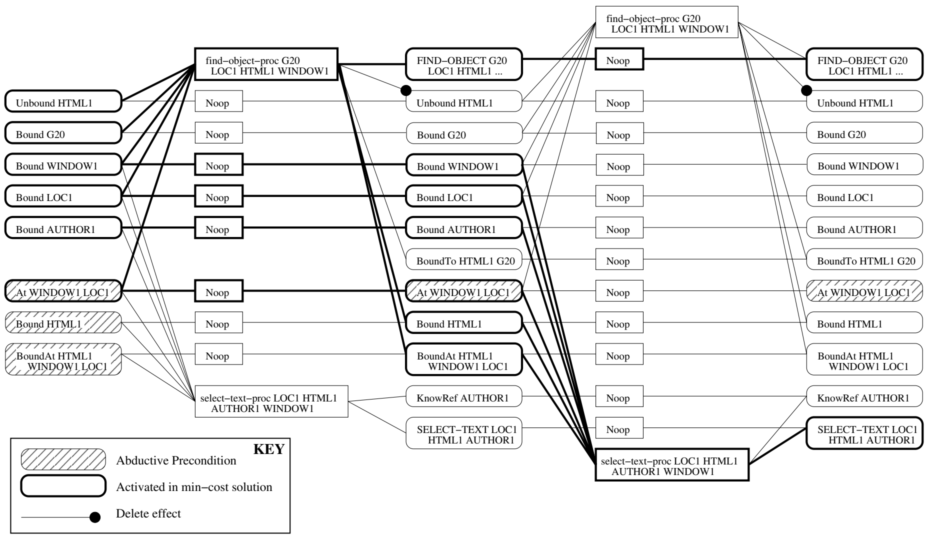

find–object–proc G20 LOC1 HTML1 WINDOW1
FIND–OBJECT G20 LOC1 HTML1 ...
Unbound HTML1
Bound G20
Bound WINDOW1
Bound LOC1
Bound AUTHOR1
BoundTo HTML1 G20
At WINDOW1 LOC1
Bound HTML1
BoundAt HTML1 WINDOW1 LOC1
KnowRef AUTHOR1
SELECT–TEXT LOC1 HTML1 AUTHOR1
select–text–proc LOC1 HTML1 AUTHOR1 WINDOW1
Noop
KEY
Abductive Precondition
Activated in min–cost solution
Delete effect

Fig. 2. A plan graph corresponding to the planning domain of the sub-task depicted in Fig. 1-b. The minimum cost solution is accented in bold.

accepts only minimum cost solutions. The incorrect solution described above is pruned because it must assume more facts than the bold-faced minimum cost solution of Fig. 2.

The third extension to the basic Graphplan algorithm computes the consistent characteristics of the minimal cost solutions: binary order constraints, assumed preconditions, and effects. The resulting sets are those elements that are present in every valid minimal cost solution. Since the example of Fig. 2 has just a single minimum cost solution, its preconditions and effects are taken to be necessary. These sets are output by the planning algorithm, and can be contrasted with the declared preconditions, effects, and order constraints of the original task.

3 Example

This section briefly describes the algorithm as it operates on the book request task model depicted in Fig. 1-a. The algorithm first decomposes the task model into its constituent sub-steps and recursively analyzes them. The author extraction sub-task, for example, is translated into a planning domain and analyzed as depicted in Fig. 2. An additional precondition **(At WINDOW1 LOC1)** and effect **(KnowRef AUTHOR1)** are recovered from the minimal cost solution. The other sub-steps of the book request task are likewise translated to planning domains and analyzed. Specifically, predicates of the form **(KnowRef <ITEM>)** are recovered as effects of the selection actions and as preconditions for the form filling actions. Analysis of the high level task resumes with updated planning operators reflecting these discoveries. The algorithm determines that:

the extract sequences precede the fill sequences, the extract steps can occur in either order, and the field filling steps can occur in either order. These refinements are shown graphically in Fig. 1-a.

The refined model more precisely characterizes the task. For example, a plan recognizer using the improved task model can, among other things, recognize task instances in which sub-tasks are performed in different orders. Additionally, the augmented precondition and effect detail of the improved model can be used to infer reasons why each individual action might be taken, allowing novel plans to be attributed to the user.

4 Conclusion

This paper presents a novel use of planning to recover missing information in task models, and describes its operation on a simple task model learned from a single demonstration with the PLOW system. The improved task models that result from this analysis are useful for understanding and reasoning about user behavior. Though this treatment analyzes procedures after they have been learned in their entirety, it is a straightforward modification to analyze procedures incrementally as new information is learned (e.g. additional sub-task specifications, preconditions, effects, etc.).

References

1. Litman, D.J., Silliman, S.: Itspoke: An intelligent tutoring spoken dialogue system. In: Proceedings of the Human Language Technology Conference: 4th Meeting of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL), Boston, MA (2004)
2. Rich, C., Sidner, C.: Collagen: A collaboration manager for software interface agents. User. Modeling and User. Adapted Interaction 8(3/4), 315–350 (1998)
3. Carberry, S.: Plan Recognition in Natural Language Dialogue. MIT Press, Cambridge (1990)
4. Jung, H., Allen, J., Chambers, N., Galescu, L., Swift, M., Taysom, W.: One-shot procedure learning from instruction and observation. In: Proceedings of the International FLAIRS Conference (FLAIRS-2006): Special Track on Natural Language and Knowledge Representation, AAAI Press, Stanford, California, USA (2006)
5. Lau, T., Domingos, P., Weld, D.: Learning programs from traces using version space algebra. In: Proceedings of the 2nd International Conference on Knowledge Capture (K-CAP) (2003)
6. Bauer, M.: Towards the automatic acquisition of plan libraries. In: European Conference on Artificial Intelligence, pp. 484–488 (1998)
7. Angros, R., Johnson, W.L., Rickel, J., Scholer, A.: Learning domain knowledge for teaching procedural skills. In: AAMAS, pp. 1372–1378. ACM Press, New York (2002)
8. Blum, A., Furst, M.: Fast planning through planning graph analysis. Artificial Intelligence 90, 281–300 (1997)