# Efficient Cross-Domain Learning of Complex Skills

Nan Li, William W. Cohen, and Kenneth R. Koedinger

School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh PA 15213 USA
{nli1,wcohen,koedinger}@cs.cmu.edu

**Abstract.** Building an intelligent agent that simulates human learning of math and science could potentially benefit both education, by contributing to the understanding of human learning, and artificial intelligence, by advancing the goal of creating human-level intelligence. However, constructing such a learning agent currently requires significant manual encoding of prior domain knowledge; in addition to being a poor model of human acquisition of prior knowledge, manual knowledge-encoding is both time-consuming and error-prone. Recently, we proposed an efficient algorithm that automatically acquires domain-specific prior knowledge in the form of deep features. We integrate this deep feature learner into a machine-learning agent, SimStudent. To evaluate the generality of the proposed approach and the effect of integration on prior knowledge, we carried out a controlled simulation study in three domains, fraction addition, equation solving, and stoichiometry, using problems solved by human students. The results show that the integration reduces SimStudent's dependence over domain-specific prior knowledge, while maintains SimStudent's performance.

**Keywords:** deep feature learning, learner modeling, transfer learning

## 1 Introduction

Education in the $21^{st}$ century will be increasingly about helping students not just to learn content but also to become better learners. In order to achieve this goal, we need to better understand the process of human knowledge acquisition and how students are different in their abilities to learn. Hence, a considerable amount of research (e.g., [6, 1, 5, 7]) has been carried out in building intelligent agents that model human learning of math and science. Although such agents are able to produce intelligent behavior requiring less knowledge engineering than before, agent developers still need to encode a nontrivial amount of domain-specific prior knowledge. Such manual encoding of prior knowledge can be time-consuming and error-prone. Moreover, providing domain-specific prior knowledge to the intelligent agents is less cognitively plausible, as students do not necessarily know such prior knowledge before class. An intelligent system that models automatic

knowledge acquisition without domain-specific prior knowledge could be helpful both in reducing the effort in knowledge engineering intelligent systems and in advancing the cognitive science of human learning.

Previous work in cognitive science [2] showed that one of the key factors that differentiates experts and novices in a field is that experts view the world in terms of deep functional features (e.g., coefficient and constant in algebra, molecular ratio in stoichiometry), while novices only view it in terms of shallow perceptual features (e.g., integer in an expression). We [3] have recently developed a learning algorithm that acquires deep features automatically with only domain-independent knowledge (e.g., what is an integer) as input. We integrate this deep feature learning algorithm into a machine-learning agent, *SimStudent* [5], to let it have this major component of human expertise acquisition. To evaluate how deep feature learner affects learning performance as well as prior knowledge requirement, we carried out a controlled simulation study in three math and science domains: fraction addition, equation solving, and stoichiometry.

## 2 A Brief Review of SimStudent

SimStudent is a machine-learning agent that inductively learns skills to solve problems from demonstrated solutions and from problem solving experience. In the rest of this section, we will briefly review SimStudent. For full details, please refer to [4]. In this paper, we will use stoichiometry as an illustrative example. Stoichiometry is a branch of chemistry that deals with the relative quantities of reactants and products in chemical reactions. In the stoichiometry domain, SimStudent is asked to solve problems such as "How many moles of atomic oxygen ($O$) are in 250 grams of $P_4O_{10}$? (Hint: the molecular weight of $P_4O_{10}$ is *283.88 g* $P_4O_{10}$ */ mol* $P_4O_{10}$.)".

During the learning process, given the current state of the problem (e.g., *1 mol* $COH_4$ *has ? mol H*), SimStudent first tries to propose a plan for the next step (e.g., *(bind ?element (get-substance "? mol H")) (bind ?output (molecular-ratio "1 mol* $COH_4$*" ?element))*) based on the skill knowledge it has acquired. If it finds a plan and receives positive feedback, it continues to the next step. If the proposed next step is incorrect, the tutor sends negative feedback to SimStudent and demonstrates a correct next step. Then, SimStudent attempts to learn or modify its skill knowledge accordingly. If it has not learned enough skill knowledge and fails to find a plan, a correct next step is directly demonstrated to SimStudent.

Based on the demonstration, SimStudent learns a set of production rules as its skill knowledge. The left side of Figure 1 shows an example of a production rule learned by SimStudent in a readable format[1]. A production rule indicates "where" to look for information in the interface, "how" to change the problem state, and "when" to apply a rule. For example, the rule to "calculate how many moles of $H$ are in 1 mole of $COH_4$" shown at the left side of Figure 1 would be read as "given the current value (*1 mol* $COH_4$) and the question (*? mol H*),

[1] The actual production rule uses a LISP format.

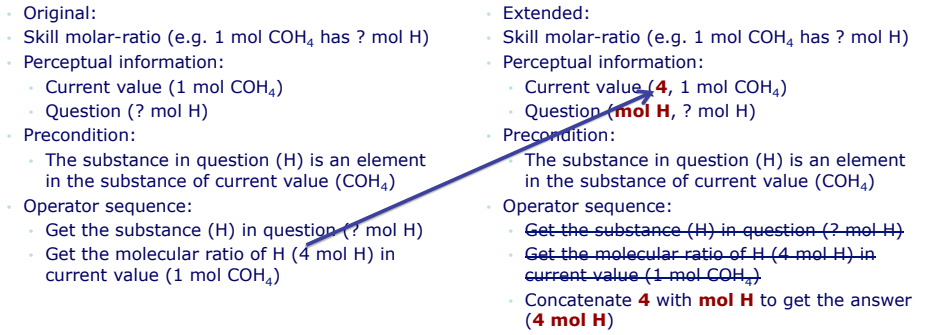



**Fig. 1.** Original and extended production rules for divide in a readable format.

when the substance in question (*H*) is an element in the substance ($COH_4$), then get the substance in question (*H*), and compute the molecular ratio of *H* (*4 mol H*) in $COH_4$".

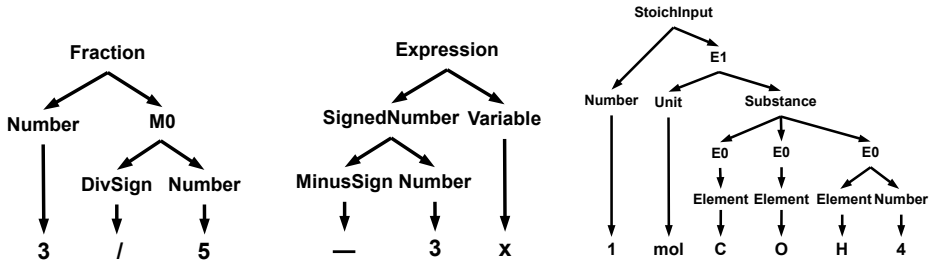
## 3 A Brief Description of Integrating Deep Feature Learning into SimStudent

To learn the "how" part in the production rules, SimStudent requires a set of operator functions given as prior knowledge. For instance, *(molecular-ratio ?val1 ?val2)* is an operator function. It generates the number of moles of an individual substance that each mole of input substance has, based on molecular ratio of input substance. There are two groups of operator functions: domain-specific operator functions (e.g., *(molecular-ratio ?val1 ?val2)*) and domain-general operator functions (e.g., *(copy-string ?val)*). Domain-specific operator functions are more complicated skills, which human students may not know in advance.

Many of the domain-specific operator functions are extraction operators that extract deep features from the input. In order to reduce SimStudent's dependence on such domain-specific operator functions, we use a deep feature learner [3] to acquire the deep features automatically, and then extend the "where" (perceptual information) part to include these deep features as needed. As presented at the right side of Figure 1, in addition to the original current value *1 mol* $COH_4$ and the question *? mol H*, SimStudent automatically adds the molecular ratio of *H* (*4*) into the perceptual information part. Then, the "how" (operator sequence) part does not need the three domain-specific operators any more. Instead, SimStudent can directly concatenate the molecular ratio (*4*) with the rest part in question (*mol H*).

Here are a few more examples to demonstrate how the extended "where" part enables the removal of domain-specific operator functions, while maintaining efficient skill knowledge acquisition. Figure 2 shows the parse trees of example input strings acquired by the deep feature learner. The deep features are associated with nonterminal symbols in the parse trees.

In fraction addition, one of the important operator functions in this domain is getting the denominator of the addend (i.e., *(get-denominator ?val)*). Figure 2(a) shows an example parse tree for *3/5*. The extended SimStudent can




(a) (b) (c)

**Fig. 2.** Example parse trees learned by the deep feature learner in three domains, a) fraction addition, b) equation solving, c) stoichiometry.

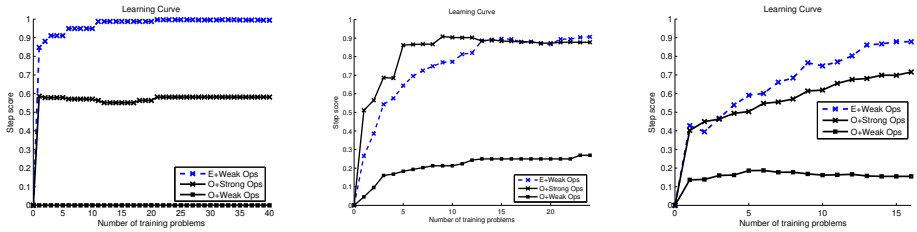
directly get the denominator *5* from the non-terminal symbol *Number* in rule *M0 → 1.0, DivSign, Number*. Then, the operator function *(get-denominator ?val)* is replaced by a more general operator function *(copy-string ?val)*. Another important domain-specific operator function in equation solving is getting the coefficient of some expression (i.e., *(get-coefficient ?val)*). With the deep feature learner, the coefficient of an expression can be extracted by directly taking the signed number (i.e., *SignedNumber*) in rule *Expression → 1.0, SignedNumber, Variable*. Again, the domain-specific operator function *(get-coefficient ?val)* is replaced by the domain-general operator function *(copy-string ?val)*. As mentioned before, *(molecular-ratio ?val0 ?val1)* is a domain-specific operator function used in stoichiometry. Instead of programming this operator function, after integrated with deep feature learning, the output can now be generated by taking the "*Number*" in grammar rule *E0 → 0.5 Element, Number*, and then concatenating with the unit *mol* and the individual substance "*Element*". Thus, the original operator function *(molecular-ratio ?val0 ?val1)* is replaced by the domain-general operator function concatenation (i.e., *(concat ?val2 ?val3)*).

## 4 Experimental Study

To further quantitatively evaluate the amount of required prior knowledge encoding and the learning effectiveness of SimStudent, we carried out a controlled simulation study in the above three domains: fraction addition, equation solving, and stoichiometry.

**Methods:** We compare three versions of SimStudent: two original SimStudents without deep feature learning, and one extended SimStudent with deep feature learning. One of the original SimStudents is given both domain-general and domain-specific operator functions (*O+Strong Ops*). The other is given only domain-general operator functions (*O+Weak Ops*). The extended SimStudent is also only given domain-general operator functions (*E+Weak Ops*).

In each domain, the three SimStudents are trained on 12 problem sequences over the same set of problems in different orders. Both training and testing problems are gathered from classroom studies on human students. SimStudent




(a) (b) (c)

**Fig. 3.** Learning curves of three SimStudents in three domains, a) fraction addition, b) equation solving, c) stoichiometry.

is tutored by automatic tutors that are similar to those used by human students. The number of training and testing problems is listed in Table 1.

| Domain Name | # of Training Problems | # of Testing Problems |
|---|---|---|
| Fraction Addition | 40 | 6 |
| Equation Solving | 24 | 11 |
| Stoichiometry | 16 | 3 |

**Table 1.** Number of training problems and testing problems presented to SimStudent.

We evaluate the performance of SimStudent with two measurements. We use the number of domain-specific and domain-general operator functions used in three domains to measure the amount of prior knowledge engineering needed. In addition, we count the number of lines of Java code developed for each operator functions, and use this as a secondary measurement to assess the amount of knowledge engineering. To assess learning effectiveness, we define a *step score* for each step in the testing problem. Among all next steps proposed by SimStudent, we count the number of next steps that are correct, and compute the step score as the number of correct next steps proposed divided by the total number of correct steps plus the number of incorrect next steps proposed. This measurement evaluates the quality of production rules in terms of both precision and recall.

**Experimental Results:** Not surprisingly, only the original SimStudent given the strong set of operator functions (*O+Strong Ops*) uses domain-specific operator functions. Across three domains, it requires at least as many operator functions as the extended SimStudent without domain-specific operator functions (*E+Weak Ops*). Moreover, since domain-specific operator functions are not reusable across domains, the original SimStudent with domain-specific operator functions (*O+Strong Ops*) requires nearly twice as many operator functions (31 vs. 17) as that of the extended SimStudent (*E+Weak Ops*) needed. The total number of lines of code required for the operator functions used by the extended SimStudent (*E+Weak Ops*) is 645, whereas the total number of lines of code programmed for the original SimStudent (*O+Strong Ops*) is 6789, which is more than ten times the size of the code needed by the extended SimStudent.

Learning curves of the three SimStudents are presented in Figure 3. Across three domains, without domain-specific prior knowledge, the original SimStudent (*O+Weak Ops*) is not able to achieve a step score more than 0.3. Given domain-specific operator functions, the original SimStudent (*O+Strong Ops*) is able to perform reasonably well. It obtains a step score around 0.85 in equation solving. However, its performance is still not as good as the extended SimStudent. Given all training problems, the extended SimStudent (*E+Weak Ops*) performs slightly better than the original SimStudent with domain-specific prior knowledge in equation solving. It (*E+Weak Ops*) achieves significantly ($p < 0.0001$) better step scores than the original SimStudent given domain-specific operator functions (*O+Strong Ops*) in two other domains. Hence, we conclude that the extended SimStudent acquires skill knowledge, which is as or more effective than the original SimStudent, while requiring less prior knowledge engineering.

## 5 Concluding Remarks

To summarize, we presented a novel approach that integrates a deep feature leaner into a simulated student, SimStudent, and demonstrated with examples how the integrated deep feature learner reduces prior knowledge engineering effort across three domains. We then carried out a controlled simulation study to quantitatively measure the amount of prior knowledge engineering and the learning efficiency, and showed that the extended SimStudent achieved better or comparable performance than the original SimStudent, without requiring encoding of domain-specific prior knowledge.

## References

1. Anzai, Y., Simon, H.A.: The theory of learning by doing. Psychological Review 86(2), 124–140 (1979)
2. Chi, M.T.H., Feltovich, P.J., Glaser, R.: Categorization and representation of physics problems by experts and novices. Cognitive Science 5(2), 121–152 (June 1981)
3. Li, N., Cohen, W.W., Koedinger, K.R.: A computational model of accelerated future learning through feature recognition. In: ITS'10. pp. 368–370 (2010)
4. Li, N., Cohen, W.W., Koedinger, K.R.: Integrating representation learning and skill learning in a human-like intelligent agent. Tech. Rep. CMU-MLD-12-1001, Carnegie Mellon University (January 2012)
5. Matsuda, N., Lee, A., Cohen, W.W., Koedinger, K.R.: A computational model of how learner errors arise from weak prior knowledge. In: Proceedings of Conference of the Cognitive Science Society (2009)
6. Neves, C.M., Anderson, J.R.: Knowledge compilation: Mechanisms for the automatization of cognitive skills. pp. 57–84 (1981)
7. VanLehn, K.: Mind Bugs: The Origins of Procedural Misconceptions. MIT Press, Cambridge, MA, USA (1990)