A MAS for Teaching Computational Logic

Jose Alberto Maestro-Prieto[1], Mª Aránzazu Simón-Hurtado[1],
Juan F. de-Paz-Santana[2], and Gabriel Villarrubia-González[2]

Dept. de Informática, Universidad de Valladolid, Spain
{jose,arancha}@infor.uva.es
Dept. de Informática y Automática, Universidad de Salamanca, Spain
{fcofds,gvg}@usal.es

Abstract. In this paper, an Intelligent Tutoring System (ITS) for teaching computational logic called SIAL is described. Several basic topics in computational logic are covered. The more complex part in SIAL is the module in charge of the diagnosis, which performs model-based diagnosis although sometimes, a knowledge-based (expertise) model is necessary in order to yield a more accurate diagnosis. The inherent complexity of the ITS is approached using a Multi-Agent System (MAS). The classical approach in ITS, which divides them into four independent modules, is adapted to a MAS creating an agent for each module and other agent for any other subsystem needed. The results obtained from an experiment of usage of SIAL are presented.

Keywords: Multi-Agent Systems, Intelligent Tutoring Systems, Computational Logic, Model Based Diagnosis, Knowledge Based Diagnosis.

1 Introduction

As software and hardware systems increase its complexity, processes that are more complex are involved in the specification and design of this kind of systems. Using logic-based descriptions is a suitable solution in this case [1]. Some kinds of logic are powerful enough to make complete descriptions and other complex tasks such as model checking, theorem proving and feature verification. Moreover, some of them can be automated, such as First Order Logic (FOL). However, dealing with practical problems of Automated Theorem Proving (ATP) such as memory limits, speeds or other performance constraints is not easy, as it can be seen in the CADE[1] ATP System Competition (CASC) reports [2].

The more complex theorem provers, such as General Proof [3] or OpenProof [4], provides a serious environment to use formal specifications. However, as it was stated by [5, 1], such kind of complex theorem provers will usually need to be guided, as they could not always be able to reach the

[1] CADE: International Conference on Automated Deduction.

S. Omatu et al. (Eds.): *Distrib. Computing & Artificial Intelligence,* AISC 217, pp. 209–217.
DOI: 10.1007/978-3-319-00551-5_26

solution without external (expert) assistance. Thus, using logic makes raise other problems: firstly, it cannot be automated completely, and hence, it is necessary people with a background good enough to be able to use it in order to get the correct results and interpret the correct conclusions.

The Intelligent Tutoring Systems (ITS) approach to self-paced learning can be valid to enhance the knowledge acquisition about Computational Logic and ATP. An ITS is a computer program based on cognitivist theories [6] that usually behaves proposing exercises to the student, and adapting its behavior (the exercises proposed, tips provided, changing the objectives if a lack of knowledge is detected, etc.) as the student acquire more knowledge and skills. An ITS usually exhibits an intelligent behavior and it is able of detecting of the student mistakes, diagnosing the detected errors, assessing the student knowledge, planning the set of exercises having into account the actual knowledge of the student, and maybe others.

An ITS fulfils most of the requirements to become a Multi-Agent System (MAS). The typical design of an ITS splits the functionality in separated, independent modules working with each other for achieving the ITS goals. In [7] agents are described as flexible problem solvers, operating in an environment over which they have only partial control and observability. One of the goals of a MAS is to construct systems capable of autonomous and flexible decision-making, and of cooperating with other systems [8]. Using the MAS approach can help to organize the complex behavior of the ITS. In [9, 10, 11, 12, 13] different proposals for implementing an ITS as a MAS can be found.

The next sections are organized as follows: computational logic is introduced, then SIAL is presented and the MAS architecture is sketched, a simple example is included and finally, the conclusions.

2 Teaching Computational Logic

FOL allows representing knowledge in a domain by stating the general relationships in that domain (called axioms) and some concrete facts for representing the specific problem. Thus, a problem can be declaratively described just stating facts and relationships, a process called formalization. It is also possible to obtain new knowledge using logic rules. Logic rules are transformation operations that get logic expressions (such as axioms and maybe facts), and yield new expressions, which it is said to be inferred by the rule. Sound, valid rules yield sound, valid new knowledge from the existing one. Sound logic rules, axioms and facts together with a proper strategy (or an algorithm) can let you prove a theorem. Automated theorem proving was a landmark in the development of the Artificial Intelligence (AI) which still appears in the standard curricula for an introductory course to AI -for example, the proposed in [14]-, as an example of an automated reasoning procedure. Classical theorem proving is based on the use of the Resolution Rule, and may be others. In order to use the Resolution rule, logical expressions should

have a specific format, termed clause form, which can be achieved by applying a well-known sequence of logical transformations to a Well-Formed Formula (WFF). At the final of the nine steps, a set of clauses are obtained. A clause is just a disjunctive set of literals. A literal is a predicate, maybe negated.

The general idea of adding a graphical representation to text descriptions is still valid [15]. Graphics can help to ease the learning of concepts and procedures and to improve the knowledge transfer, and it can be more helpful for novices [16].

This is probably the first time the students come face to face with a declarative programming language [17]. There exist several tools which goal is to ease the learning of logic and to get practice solving logical problems, such as HyperProof [18], Deep Thought [19] or WinKE [20]. More recently, a new graphical representation based on puzzle pieces for the SWI-Prolog interpreter output has been proposed in [17]. All these tools share a feature: it is available some kind of graphical output for representing the developed internal process.

Formalization has also been approached using ITSs, for example in [21]. However, to the best of our knowledge, transforming a WFF into the clause form has only been approached twice. In [21] it is described a computer program able to process a WFF up to one specific step (of the nine steps process for obtaining the clause form from a WFF), on demand of the student. Then, the student can check its own, hand written, response and visually comparing it against the result yielded by the program. The other alternative is our own solution, called SLI, which is also able to expose the individual steps of the process for transforming a set of WFF into a set of clauses.

3 SIAL: An ITS for Computational Logic

SIAL is an ITS designed to perform as a practical tool, that automatically can check the student's solution to a proposed exercise. SIAL provides computational support in order to help the student to fix some concepts and procedures in a practical way; the student is proposed exercises that should be solved using usual techniques and methods of computational logic.

SIAL is designed to be able to propose exercises ranging from obtaining the clause form from FOL expressions up to being able to use the hyperresolution rule. SIAL is organized in 15 thematic levels (Table 1). The levels are arranged from the simplest skills to the most complex ones. Levels 1 to 6 include converting a set of WFF to clause form, predicate unification, binary resolution, resolution refutation and the factoring rule. These levels constitute the basic skills to be acquired, thus forming the main basis for the following levels.

Levels 7 to 12 deal with methods for selecting clauses to yield a new resolvent and reducing the number of generated clauses. Pure literals, tautologies and subsumed clauses should be detected and removed. In addition, the set

of support strategy is included as a way to control the progressive increase for the generated clauses. Level 12 is devoted to the hyperresolution rule.

SIAL levels can be grouped depending on the interaction the student is allowed to do. Levels 1 to 6 are designed for novice students. These levels are more guided and the student is encouraged towards a stepwise interaction. Despite this, student interaction is still wide enough, and this makes it impossible obtaining an accurate diagnosis in every case. Levels 7 to 12 are designated for middle-level students and tje guidance is weakened. In these levels, some restrictions are removed and students can provide a solution without some intermediate steps. There exists also an automatic mode in which SIAL solves a problem proposed by the student. SIAL behaves as a usual ATP program.

SIAL is designed to increase the complexity of the developed process in two ways: firstly, exercises should be arranged into each level from the easier to the most difficult. Second, the first three levels only deal with one process (although a complex process), whereas levels from 4 to 12 are defined to combine several processes. This approach makes the learning process progressive as upper levels lie on previous ones. Each level above the level 4 can assume the functionality of each level below it. In this way, the user must integrate the knowledge already practiced jointly with a new technique, in the current problem solving process.

4 The ITS as a Multi-agents System

SIAL is composed of several agents which are capable of interact with the student while collect, process and evaluate new data together with historical data. The agents of SIAL know the student history and they use these data to adapt the training plan. SIAL architecture is based on the classical four modules ITS architecture [22]. The same as in [13], classical modules

Table 1 Description of each level defined in SIAL, guiding and topic classification

Level	Guiding	Topic	Description
1	Strong	Single	Getting the clause form (without Skolemization process).
2	Strong	Single	Getting the clause form (with Skolemization process).
3	Strong	Single	Unification procedure.
4	Strong	Compound	Resolution rule.
5	Strong	Compound	Strongly guided refutation resolution.
6	Strong	Compound	Factoring rule.
7	Weak	Compound	Weakly guided refutation resolution.
8	Weak	Compound	Pure literal removing.
9	Weak	Compound	Tautology removing.
10	Weak	Compound	Support set strategy.
11	Weak	Compound	Subsumption.
12	Weak	Compound	Hyperresolution (positive/negative hyperresolution).

have become in agents. Together with these main agents, any other support subsystem used in SIAL has also been wrapped as an agent of the system.

4.1 The Main Agents

The **interface** agent is responsible for showing the environment for problem solving and controlling the user interaction. Besides the program interface, the interface agent also contains the *Help manager*. In order to develop its functionality, the interface agent must communicate with the others main agents, those acting as the expertise module, the student model and the pedagogical module.

The agent performing the **expertise model** is responsible for interpreting the actions the student takes. It is composed of two automated theorem provers that represent the domain model which detect mistakes and identify errors. It also includes an *expert system engine* (CLIPS) for helping in error identification. This agent is in charge of inferring the student's knowledge by analyzing the student's actions. SIAL applies up to two different processes in order to obtain an accurate diagnosis [23]: a model-based diagnosis, based on ATP, to ensure the student's answer validity and maybe obtain a diagnosis and, whenever it is not possible to get an accurate diagnosis, an expertise-based diagnosis (using expert systems) is tried.

The agent in charge of the **student model** is responsible for maintaining a representation of the student's knowledge. It stores the actions taken by the student, the mistakes and errors identified, the exercises solved and so on, in a database.

The agent acting as the **pedagogical module** is responsible for the instructional support. The pedagogical module agent uses the information contained in the student model in order to plan a sequence of exercises, to decide if a reinforcement exercise is necessary or whether any advice message should be sent to the student. It also decides when a student is promoted to the next level.

A knowledge-based implementation has been chosen for the pedagogical behavior. The knowledge for selecting the next exercise to be shown is represented as an expert system.

4.2 The Support Agents

Together with the main agents, SIAL also has some other agents in order to wrap other software programs and obtain some typical features in MAS architecture such as, an autonomous behavior, high modularity and interoperability. Wrapping the external programs as agents can also help to replace some of these programs, by newer versions or by other programs without changing the whole application.

The expertise model agent rely on three other agents, both of them are automated theorem provers (SLI and OTTER) and the other is the *expert*

Table 2 Number of errors identified by SIAL, grouped by exercise level and diagnosis method

SIAL Level	N. errors	% Diagnosed by Formal Model	% Diagnosed by E. S.
1	1626	84′38	15′62
2	1957	87′79	12′21
3	976	100′00	
4	261	100′00	
5	402	100′00	
7	482	100′00	
Total	5702	91′35	8′65

system engine CLIPS. The wrapping agents for SLI and OTTER make the communication uniform between them and the other agents despite the fact that both provers do not share a common interface. In addition, some of these agents communicate with several agents, for instance, the CLIPS agent should communicate with the expertise model agent and with pedagogical module agent. Providing them with a high-level communication protocol (a subset of the KQML standard language [24]) eases the communication among them.

The student model agent also interacts with a database. The database has also been wrapped by its own agent. In one hand, this allows to change the physical database reducing the changes in the whole program. SIAL has been used with several Databases: MySQL, MS SQLServer and PostgreSQL whereas MS Access is used when developing. This also allows running SIAL maintaining a central database. SIAL agents can communicate via standard HTTP requests and hence, the agent database can be placed in the computer containing the physical database whereas the others agents and specially, the program interface, can be placed in other computer.

5 A Practical Experiment

SIAL has been used in the practical sessions of an introductory course of Artificial Intelligence. 33 students regularly assisted to practical sessions and use SIAL together with SWI-Prolog to solve exercises. The experiment consisted of 120 practical exercises classified in levels (from 1 to 6) some of them mandatory (61) and others (59) for reinforcement. Each student made an average of 70.16 exercises ($\sigma = 18.56$). Data collected about the errors identified during the experiment is shown in Table 2.

Only 270 (4,74%) out of the 5702 detected errors were syntactic errors. All the other errors were errors due to a wrong proposed solution: wrong expressions, wrong use of operators, lack of literals, wrong use of rules, and so on. Hence, the implemented interface in SIAL seem to be able to minimize the number of errors due to expression manipulation. The huge difference in the number of errors detected between the first two levels and the others can

be explained because of the complexity of transforming WFF into the clause form with respect to the unification and resolution processes. The expert systems are designed to deal with errors at levels 1 and 2 (clause form), and they are invoked only after the model based diagnosis. The model based diagnosis diagnoses the 91,35% of the errors and provides SIAL with a great detection power and accuracy.

The students received a survey once they have solved some exercises using SIAL and SWI-Prolog. The survey inquired students about their opinion about usefulness, usability and a comparison of SIAL and SWI-Prolog. Students were also asked to mark SIAL. Questions have four possible answers.

Most of the students said that SIAL is very useful (31.82%) or useful (54,54%) for learning computational logic. Only 1 student (4,54%) said that SIAL is not useful at all.

All the students said that they have solved exercises using both SIAL and SWI-Prolog. A 31.82% of the students said that SIAL cannot be replaced by SWI-Prolog, and a 50.0% said that SIAL only can be replaced very partially by SWI-Prolog. None said that SIAL and SWI-Prolog were equal.

The 77.27% of the students said that SIAL is very easy to use and the 13.64% of the students said that is easy to use. None of the students said that SIAL is difficult to use. SIAL was marked as B (Good) by the 90.91% of the students. The rest of the students (9.09%) mark SIAL as C (Pass).

6 Conclusions

SIAL, an ITS for learning computational logic, has been described. SIAL has been implemented as a MAS. The different modules of the ITS have been converted in agents together with the auxiliary subsystems. One of the most important modules in SIAL is the expertise module, which is made up of two automatic theorem provers and expert systems. The chosen approach allows us to provide students with an accurate tutor. The chosen model-based approach lets SIAL detect the most of user's errors accurately, inform the user about its existence, and provide some hint to localize and fix the mistake.

The results obtained in the practical experiment are promising. The student's answers to the survey show a clear difference between SIAL and SWI-Prolog. Student's answers also say that SIAL is useful for learning computational logic and it is easy to use.

References

1. Bundy, A., Moore, J.D., Zinn, C.: An Intelligent Tutoring System for Induction Proofs. In: Melis, E., Scott, D., et al. (eds.) CADE-17 Workshop on Automated Deduction in Education, pp. 4–13 (2000)
2. Sutcliffe, G.: The CADE-23 Automated Theorem Proving System Competition - CASC-23. AI Communications 25, 49–63 (2012)

3. Aspinall, D., Lüth, C., Winterstein, D.: A Framework for Interactive Proof. In: Kauers, M., Kerber, M., Miner, R., Windsteiger, W. (eds.) MKM/CALCULEMUS 2007. LNCS (LNAI), vol. 4573, pp. 161–175. Springer, Heidelberg (2007)
4. Barker-Plummer, D., Etchemendy, J., Liu, A., Murray, M., Swoboda, N.: Openproof - A Flexible Framework for Heterogeneous Reasoning. In: Stapleton, G., Howse, J., Lee, J. (eds.) Diagrams 2008. LNCS (LNAI), vol. 5223, pp. 347–349. Springer, Heidelberg (2008)
5. Aitken, J.: Problem Solving in Interactive Proof: A Knowledge-Modelling Approach. In: Wahlster, W. (ed.) Proceedings of the 12th European Conference on Artificial Intelligence, pp. 335–339. John Wiley & Sons, Budapest (1996)
6. Driscoll, M.P.: Psychological foundations of instructional design. In: Trends and Issues in Instructional Design and Technology, 2nd edn., pp. 36–44. Pearson Education, Inc., Upper Saddle River (2007)
7. Jennings, N.R., Wooldridge, M.: Agent-Oriented Software Engineering. In: Bradshaw, J. (ed.). AAAI/MIT Press (2000)
8. Rodríguez, S., Pérez-Lancho, B., de Paz, J., Bajo, J., Corchado, J.: Ovamah: Multiagent-based Adaptive Virtual Organizations. In: Proceedings of the 12th International Conference on Information Fusion, Seattle, USA, pp. 990–997 (2009)
9. Cheikes, B.: GIA: An agent-based architecture for Intelligent Tutoring Systems. In: Proceedings of the CIKM 1995 Workshop on Intelligent Information Agents (1995)
10. Capuano, N., Marsella, M., Salerno, S.: ABITS: An Agent Based Intelligent Tutoring System for Distance Learning. In: Proceedings of the International Workshop in Adaptative and Intelligent Web-based Educational Systems, pp. 17–28 (2000)
11. Hospers, M., Kroezen, E., Nijholt, A., op den Akker, R.: Developing a generic agent-based intelligent tutoring system. In: Devedzic, V., Spector, M., Sampson, D., Kinshuk, M. (eds.) The 3rd IEEE International Conference on Advanced Learning Technologies. IEEE Computer Society, Los Alamitos (2003)
12. Fernández-Caballero, A., Gascueña, J., Botella, F., Lazcorreta, E.: Distance learning by intelligent tutoring system. Part I: Agent-based architecture for user-centred adaptivity. In: Proceedings of the 7th International Conference on Enterprise Information Systems, pp. 75–82 (2005)
13. González, C., Burguillo, J., Vidal, J.C., Llamas, M., Rodríguez, D.: ITS-TB: An Intelligent Tutoring System to provide e-Learning in Public Health. In: Proceedings of the 16th EAEEIE Annual Conference on Innovation in Education for Electrical and Information Engineering, EIE (2005)
14. Association for Computing Machinery (ACM): Computer Science Curriculum 2008: An Interim Revision of the CS 2001 (2008)
15. Mayer, R.E., Moreno, R.: Nine Ways to Reduce Cognitive Load in Multimedia Learning. Educational Psychologist 38, 43–52 (2003)
16. Clark, R., Mayer, R.: e-Learning and the Science of Instruction: Proven Guidelines for Consumers and Designers of Multimedia Learning, 3rd edn. Pfeiffer (2011)
17. Mondshein, L., Sattar, A., Lorenzen, T.: Visualizing prolog: a "jigsaw puzzle" approach. ACM Inroads 1, 43–48 (2010)

18. Stenning, K., Cox, R., Oberlander, J.: Contrasting the Cognitive Effects of Graphical and Sentential Logic Teaching: Reasoning, Representation and Individual Differences. Language and Cognitive Processes 3, 333–354 (1995)
19. Croy, M.J.: Graphic Interface Design and Deductive Proof Construction. Journal of Computers in Mathematics and Science Teaching 18, 371–385 (1999)
20. Endriss, U.: The interactive learning environment winke for teaching deductive reasoning. In: Manzano, M. (ed.) Proceedings of the 1st International Congress on Tools for Teaching Logic. University of Salamanca (2000) (invited talk abstract)
21. Hatzilygeroudis, I., Giannoulis, C., Koutsojannis, C.: A Web-Based Education System for Predicate Logic. In: Proceedings of the IEEE International Conference on Advanced Learning Technologies, pp. 106–110. IEEE Computer Society, Washington, DC (2004)
22. Mitrovic, A., Martin, B., Suraweera, P.: Intelligent tutors for all: The constraint-based approach. IEEE Intelligent Systems 22, 38–45 (2007)
23. Ferrero, B., Fernández-Castro, I., Urretavizcaya, M.: Multiple Paradigms for a Generic Diagnostic Proposal. In: Gauthier, G., VanLehn, K., Frasson, C. (eds.) ITS 2000. LNCS, vol. 1839, p. 653. Springer, Heidelberg (2000)
24. Finin, T., Weber, J., Wiederhold, G., Genesereth, M., Fritzson, R., McGuire, J., Pelavin, R., Shapiro, S., Beck, C.: Specification of the KQML Agent-Communication Language. Technical Report, DARPA, DRAFT (1993)