Oliver: an OnLine Inference and VERification system

Andy Wildenberg[1], *Christelle Scharff*[2]

Abstract — *We describe Oliver, the Online inference and verification system for propositional logic. It provides a web-based interface for teaching propositional logic proofs, and accepts any valid direct proof. Oliver provides instant feedback to students as to whether each step is correct or not, encourages experimentation by students and is integrated within the WeBWorK system for online grading and support. Oliver randomizes problems for students to reduce plagiarism and cheating. It has been successfully used to teach propositional logic to several thousand students at SUNY Stony Brook, and is very popular among students and faculty.*

Index Terms — *Distance learning, Inference System, Interactive Proofs, Proof Strategy, Proof Verification, Propositional Logic, Web.*

INTRODUCTION

Propositional logic is a core subject in several different fields. According to the ACM standards for computer science education, Discrete Structures is a required course, typically taught in one or two semesters [4,15,16]. One subtopic of Discrete Structures is propositional logic. Proving in propositional logic forms the basis for more complicated proofs in set theory and number theory later on in the Discrete Structures curriculum. Proof-based topics also appear in standard curricula for Mathematics, Philosophy, Electrical Engineering and Computer Engineering.

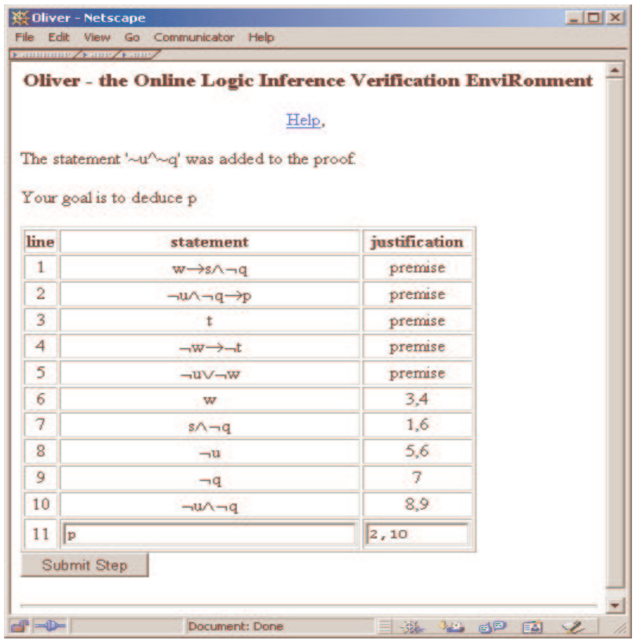
As a new framework for abstract symbol manipulation, propositional logic is typically difficult for students [7-10] – students typically underestimate or overestimate what can be done at any given step within a proof. Cognitive psychologists estimate that only 1 to 4% of the population are able without explicit training to correctly apply the principles of formal logic in situations where it is appropriate to do so [5,12]. Visualization in assisting the reasoning process through the use of software [1,3,11] and web-based can help students learn and enjoy logic [2,3]. Previous interactive systems for doing proofs using strict inference systems and strategies given by the user are Jape [11] and Logic Tutor [1].

We introduce Oliver [13], a web-based system for teaching propositional logic proofs. Students using Oliver enter their proofs online interactively one step at a time with the system checking each step. Feedback is instantaneous, and students can do the same proof multiple times, experimenting with different strategies of solution to see which are most effective. Problem randomization means that no two students get the same proof, although everyone gets an equivalent proof. This helps students draw the line between helping each other and just copying answers. It has been integrated in the WeBWorK system [14] for online grading and support.

Oliver has been used for four semesters at SUNY Stony Brook for teaching propositional logic to more than 2000 computer science, information science and computer engineering students, and will be used at Pace University starting in Fall 2002. Student feedback has been nearly universally positive – students enjoy using the system and feel it teaches them well.

DESCRIPTION OF THE SYSTEM

Oliver provides a web-based interface for teaching traditional propositional logic. Operators are conjunction ($\wedge$), negation (~), inclusive disjunction ($\vee$), implication ($\rightarrow$) and double implication ($\leftrightarrow$). Proofs are of arbitrary complexity, allowing an unlimited number of premises, steps in the proof, and individual statements of any length.



Oliver - the Online Logic Inference Verification EnviRonment

Help,

The statement '~u^~q' was added to the proof.

Your goal is to deduce p

line	statement	justification
1	$w \rightarrow s \wedge \neg q$	premise
2	$\neg u \wedge \neg q \rightarrow p$	premise
3	t	premise
4	$\neg w \rightarrow \neg t$	premise
5	$\neg u \vee \neg w$	premise
6	w	3,4
7	$s \wedge \neg q$	1,6
8	$\neg u$	5,6
9	$\neg q$	7
10	$\neg u \wedge \neg q$	8,9
11	p	2, 10

Submit Step

FIGURE 1
A SCREENSHOT OF THE OLIVER SYSTEM IN USE.

[1] Department of Computer Science, SUNY Stony Brook, Stony Brook, NY 11794-4400, awilden@cs.sunysb.edu
[2] Corresponding Author – Pace University, New York City Campus, Computer Science and Information System Department, New York, NY 10028, csharff@pace.edu

Oliver is a unique system in several ways. Proofs are done with standard operators compatible with the systems taught by most discrete mathematics textbooks. At SUNY Stony Brook, Oliver was used coupled with the theory taught in [6]. Currently, the help system for Oliver is a single web page that introduces the symbols for the different logical operators and shows a sample proof. Beyond that, students refer to their textbooks. Oliver can be adapted to other styles of proofs, such as analytic tableaux or resolution proofs.

At each step Oliver asks students for the next line of the proof, and a list of one or two lines that it is derived from. Oliver accepts any statement that is a logical consequence of the specified lines. This gives students an additional flexibility in doing the proofs that makes Oliver easier to use than a system based on rules.

Oliver proofs are more natural to write than in other systems and more like the kinds of proofs that people write when they are doing proofs by hand. The added flexibility comes from Oliver's method of evaluating when a statement is legal: instead of using a rigid set of rules, it actually computes the truth table to determine that the statement is a logical conclusion of the user specified premises. Computational complexity is limited because the number of propositional variables of the problems given to students is limited to five or six.

To reduce plagiarism, Oliver customizes the problems for each student. For any given problem, the order of the premises and the labels of the literals is permuted depending on the student. Positive and negative literals are also swapped at random. The result is that all students have equally difficult problems, yet it is often more time-consuming for students to derive the correspondence between proofs than it is to completely do the proof from scratch.

In fact, many students do not seem to notice that the proofs are related to each other. On the class bulletin board when students ask for help on normal homework problems, they typically just say the section and problem number. With Oliver problems, they post the full text of the problem, because they know that without it people will not be able to help them.

One drawback of the truth table system is that Oliver sometimes accepts "illegal" proofs. Of course, the proofs are legal in the sense that each step is a logical consequence of the premises specified. However, sometimes it would be preferable for a step to be broken up into several smaller pieces corresponding to the application of an inference rule (modus ponens, modus tollens, disjunctive addition...). For example

1. $s \rightarrow (p \wedge (q \vee r))$	Premise
2. $\sim p$	Premise
3. $\sim s$	1,2

would not be accepted by in most strict rule-based systems for propositional logic as a correct step in a proof, because there are too many things that have been combined. A more acceptable proof might look like

1. $s \rightarrow (p \wedge (q \vee r))$	premise
2. $\sim p$	premise
3. $\sim(p \wedge (q \vee r))$	2, contradiction
4. $\sim s$	1,3, modus tollens

or in some cases, line 3 could be even split up further.

For future reference, all completed proofs are recorded in a database. This allows a detailed analysis of the style that the students are using when doing proofs. For example, consider a proof with four premises $p_1 \ldots p_4$ and conclusion c. Under Oliver's current system, the following proof is acceptable

1. p_1	Premise
2. p_2	Premise
3. p_3	Premise
4. p_4	Premise
5. $p_1 \wedge p_2$	1,2
6. $p_3 \wedge p_4$	3,4
7. $p_1 \wedge p_2 \wedge p_3 \wedge p_4$	5,6
8. c	7

which would clearly not be acceptable in a propositional logic class. However, because of the database, it is possible to look back and confirm that nobody has actually tried to do such a proof.

Finally, while the Oliver system can be used independently of other software packages, it has been loosely integrated with the WeBWorK environment, which provides student authentication and bookkeeping features to automatically record the scores. It would be easy to write similar bridge software to integrate it with other packages such as WebCT or Blackboard.

IMPLEMENTATION DETAILS

The explosion of the web browser and Internet connectivity has drastically changed the way university students do work and find information. Nearly all students have access to a computer with a web browser; most have one in their rooms. In a world where everyone is used to working online in their rooms, Oliver lets them do this.

The web-based interface allows students to use Oliver without installing any custom software or learning how to use a new application. End-user support is simplified because it is assumed (and easy to verify) that the browser is functioning.

At an implementation level, the Oliver system contains two pieces: a propositional logic engine for checking inferences and generating random problems, and a servlet-based interface allowing access to the system through standard web browsers. The Java servlet interface is easy to install – support for servlets already comes standard on many web servers for Unix, Windows and MacOS systems.

In general, servlets run very efficiently, and in particular, Oliver places a minimal load on system resources. Furthermore, by putting most of the logic on the server, there are almost no demands on the client machine. Any web browser can be used with the system without the user installing any additional software. Old machines (provided they can run a web browser) can be used as clients just as easily as brand-new ones. Server-based systems also increase the security of the system and provide centralized scoring facilities. Additionally, the system logs all completed proofs, enabling a more detailed analysis of student solution patterns later on.

PEDAGOGICAL GOALS

The Oliver system is designed to target several different pedagogical goals. Our main goals in Oliver are to help students learn to employ formal reasoning methods, access logic and enjoy it. Visualization [3,11] through the use of web-based or graphical interface is considered helpful in assisting students in learning and enjoying logic [2,3]. Using a tool lets students experiment while doing proofs. They can attempt the same proof several times, trying different strategies and methods. Because they have instant feedback, they can be less conservative in their work – they know that if they make a mistake, they can correct it and move on.

Flexible evaluation, immediate feedback for beginners

We didn't want Oliver to be too rigid and frustrating for students. With Oliver, students are taught standard inference rules in class but the implemented mechanism in Oliver does not pedantically check the application of rules – instead it is based on use of truth tables and only confirms that a given step is a valid consequence of the stated premises. The advantage is that Oliver's proofs more closely match the proofs the students would obtain by hand. Many proving systems disallow multiple steps at a time. Using such systems to verify human proofs is frustrating because the system is too pedantic, and for beginning students, this frustration causes problems. With its truth-table mechanism, Oliver allows more "natural" proofs, i.e. proofs that closely match what a human would write down without a machine checking every step. For example, it might be natural to rearrange $(p\wedge q\wedge r) \rightarrow s$ to be $(r\wedge q\wedge p) \rightarrow s$, but few rule-based systems will allow such a manipulation in a single step.

Through its web-based interface, Oliver offers a way to visualize each step of the non-deterministic construction and discovery of the considered proof. Cognitively, this representation is more powerful than the more text-based representation. From this point of view, Oliver qualifies as one of the "envisioned tool(s) that would facilitate the abstract subject of logic, and thereby work more effectively with it" mentioned in [2,3]. The graphical web-based interface makes learning reasoning methods and logic enjoyable for the student.

Students can attempt a problem in Oliver as many times as they want, even after successfully completing it. All correct solutions are recorded in a central database for further analysis. In the database there are several cases of students trying to optimize their solution to the most compact form, attempting multiple tacks to solve the problem.

When doing a proof, students are stopped as soon as they make a mistake, and can not progress until the mistake has been corrected. This stops "begging the question" and "jumping to conclusion" types of errors. It also permits them to achieve incremental learning.

Another main advantage of Oliver is its instant feedback. Before Oliver, students typically had to wait 2-3 weeks from when they completed the homework to when they got their grades. Grading proofs for professors and teaching assistants was not an easy task. Many students would walk into a test, and not be able to write anything down for the symbolic proof.

Empirically, proof quality and scores have improved dramatically since we started using Oliver. On a propositional logic test question this semester, almost all the students who scored more than 10/12 points completed all the Oliver questions. More than half of the students who scored less than 5/15 on the Oliver problems scored obviously less than 2/12 on the test.

Integration with standard environments

Oliver has been integrated into the WeBWorK environment. Developed by the University of Rochester, WeBWorK is also a web-based method for homework problems, and provides many advanced features that are emulated in Oliver, including instant feedback, individualized problem sets, and the evaluation of symbolic answers (i.e. formulas). Additionally, WeBWorK provides authentication and grading capabilities, an advanced scripting mechanism for rapidly creating new problem sets, and a mature infrastructure for web-based evaluation.

WeBWorK is used to teach a variety of continuous mathematics (algebra, calculus, vector calculus) and physical science (physics, chemistry) classes at major universities across the country. We use it for discrete mathematics and functional programming homeworks. The integration of Oliver into WeBWorK permits use to use all of WeBWorK's infrastructure capabilities (delivering, grading, posting solutions of homeworks) and of control of plagiarism and cheating.

In addition, all correct solutions are recorded to a database, allowing automated analysis of solutions for some statistics at a later date. The use of WeBWorK permits us to control plagiarism and cheating. In WeBWorK problems are permuted and do not use the same data. Oliver has also been designed to avoid plagiarism and cheating.

Problem customization/randomization

To this end, Oliver randomizes the problem for each student by permuting the order of the premises, permuting the variables used and randomly swapping positive and negative literals. For example, the following are three students' versions of the same proof

Student A	Student B	Student C
1. $p \rightarrow q$	1. $p \rightarrow \sim s$	1. v
2. $q \rightarrow r$	2. $\sim r$	2. $v \rightarrow \sim u$
3. p	3. $\sim r \rightarrow p$	3. $\sim u \rightarrow s$
$\therefore r$	$\therefore \sim s$	$\therefore s$

Note that the three proofs are completely equivalent to each other and require the same amount of work to solve (if students follow the same proof strategy). However, the correspondence between them is not obvious (and becomes less obvious as the proofs get more complicated). In practice, few students seem to even try to make the correspondence.

The randomized problems also seem to encourage study groups without encouraging plagiarism. In general, students have a hard time distinguishing between helping each other studying and copying answers, but when everyone has a different problem, the distinction is clear.

Faster and more accurate grading

The system is more accurate than hand grading. Grading propositional logic proofs is time consuming and error prone because there are so many correct proofs, and each step must be considered in turn. To expedite grading, SUNY Stony Brook graders typically graded only 1 or 2 complicated student problems for the homework assignment (out of an assignment of 6). Grading was inconsistent from TA to TA, resulting in students saying "but I did the same thing for the homework and the TA gave me full credit." The automated system is faster and more accurate. Students typically do 11 problems; many want more, and some even want the source code so they can play with the system at home.

USER FEEDBACK AND ANALYSIS

Oliver is a popular tool with teaching assistants and professors, but students seem to prefer it most of all. A survey taken for two consecutive semesters found broad-based support for web-based homework in general and Oliver in specific. Only 22% of students said they preferred paper-based homework to web-based homework. 71% said that Oliver was the most useful web-based homework, and 52% said logic was their favorite topic in the class. Only 8% of the class said they felt it was easy to cheat on the homework.

A topic on the bulletin board asking for user feedback for Oliver also gave similar support for the system. A typical response was

> "*The Oliver system really helped me out by practicing proofs and knowing if it was correct. One thing that I feel would help students would be to have to state the rule for each step that they complete.*"

Easily the most common request from students was to adapt the system to require you to name the rule used for a given step. Students know that they have to name the rules on the test, and wish that Oliver could check their work in that aspect with the same accuracy and immediacy as it checks their logic.

Students also seem to trust Oliver more than the graders. On the propositional logic exam, roughly 7% of students asked for a regrade of the propositional logic proof. During the homework less than 1% of the students wrote emails saying that Oliver wouldn't accept a valid step (further investigation confirmed that Oliver was right). Of course there were lots of emails asking for help in solving the proofs.

Since students know that the scores are high for the Oliver assignment and since they can see their scores as the go along, students are driven to get every proof right. This is different from the attitude toward written homeworks, where the standard is "good enough." As an example, this semester at Stony Brook 78% of the students attempted the Oliver assignment. The average score was 13.4/15 and 59% of the students who attempted the Oliver assignment got all of the problems correct. The same semester, on a paper homework covering similar material (and using a grading scale arguably much more generous than Oliver's), 84% of the students attempted the homework, but the average grade was 4.7/10 and only 17% of the class got 10/10.

FUTURE DIRECTIONS

While Oliver has been used several semesters as the primary method of teaching/evaluating propositional logic, its development is still an active project. There are several general categories of work for the immediate future, mostly involving modifying the existing tool according to the feedback of the users of Oliver, the results of our evaluations and the database of complete proofs to extend our pedagogical goals and improve the user experience.

Oliver does not ask (and can not verify) the names of the rules used for each step. It only checks that the conclusion is valid. The implementation of the use of inference rules will require a significant overhaul of the Oliver framework. The difficulty is that humans tend to group several steps together at a time, for example applying the commutative and associative properties of conjunction at the same time as performing an inference rule. This is one of the reasons that the Oliver system was built using truth tables. To get around this, we will develop a set of (sound) rules iteratively. By analyzing the database of solutions (currently at roughly 10k solutions), we plan to iteratively construct a set of rules by to identifying a common rule, figuring out what it is called, inserting that rule into the system, then identifying another common rule, and so on until all the proof steps in the database have been entered as rules, or determined to be too complex to be atomic rules.

Many students ask for more problems than the system currently has in it. The system could be modified to allow students to type in their own premises and conclusion, thus opening up the system. Students should be able to specify their premises.

Along the same lines, a proof editor for creating new proofs would be useful for professors. Currently, inserting new problems or modifying existing problems requires a recompilation of the system and some basic knowledge of Java. A web-based front end could easily be provided to allow non-technical professors to input new problems without recompiling the system.

Oliver currently only supports direct proofs. However, it would be simple to allow proof by contradiction within the Oliver framework. The only modifications would be to provide a mechanism to negate the conclusion, and to recognize when the empty clause has been derived.

There are several minor modifications that will improve the user experience and make it more concordant with WeBWorK's experience. Specifically, Oliver should record incomplete proofs and provide options for printing proofs and printing solutions once the due date for problems has passed. Students currently must complete an Oliver problem in a single session on the web. Most other problems in the WeBWorK system remember a student's answers from one session to the next. Also, WeBWorK allows you to print all of your problems at once, allowing the students to work on the assignments offline, and releases the solutions to the students once the due date has passed. While students can still print problems by printing out each Oliver problem on a separate page, it would be nice to give them a more compact representation.

We think also that a tighter integration of Oliver and WeBWorK is important. While the system does integrate with WeBWorK, it opens each Oliver problem in a separate window and requires students to copy a "ticket" (a nine character string that is a cryptographic certificate of completion) from Oliver to WeBWorK. While students don't seem to mind copying the ticket, they find the multiple windows confusing. A closer integration will eliminate both the need for tickets and the extra windows.

ACKNOWLEDGMENT

We would like to acknowledge Arnold Pizer and Mike Gage from the University of Rochester for developing the WeBWorK system and providing the expert support and advice helping us to integrate Oliver with their system.

REFERENCES

[1] Abraham, D., Crawford, L., Lesta, L. Merceron, A. and Yacef, K., "The Logic Tutor: A Multimedia Presentation", *Interactive Multimedia Electronic Journal*, 3(2), http://imej.wfu.edu/articles/2001/2/ Oct. 2001.

[2] Barwise K. and Etchemendy, J, Logic software from CSLI, http://www-csli.standford.edu/hp/Logic-software.html.

[3] Barwise K. and Etchemendy, J., *The language of first-order logic*, Cambridge University Press, 1993.

[4] *Computing Curricula 2001*, December 1, 2001, http://wwww.acm.org/sigcse/cc2001/

[5] Douglas, Ronald G., ed., "The logic of teaching calculus", In *Toward a lean and lively calculus: report of conference workshop to develop curriculum and teaching methods for calculus at the college level*, Washington D.C.: Mathematical Association of America, 1987, pp. 41-59.

[6] Epp, S. S., *Discrete Mathematics with Application*, Wadworth, 1995.

[7] Fung, P. and O'Shea, T., "Using software tools to learn formal reasoning: a first assessment", *CITE report no 168*, Open University, 1992.

[8] Fung, P., O'Shea, B., Gohson, D., Reeves, S. and Bornat, R., "Computer Science students' perception of learning formal reasoning", *International journal of Mathematical Education in Science and Technology* 24 (5) pp. 749-760.

[9] Fung, P., O'Shea, B., Godson, D., Reeves, D. and Bornat, R. "Why computer science students find formal reasoning frightening", *Journal of Computer Assisted Learning*, 10, pp. 240-250.

[10] Fung, P., O'Shea, B., Godson, D., Reeves, D. and Bornat, R. "Computer tools to teach formal reasoning", *Computers in Education*, 27(1), pp. 59-69.

[11] Bornat R. & Sufrin B.A. (1999) "Animating formal proof at the surface: the Jape proof calculator", The Computer Journal, 43(3), 177-192.

[12] *Math-thinking discussion group.* http://www.cs.geneseo.edu/~baldwin/math-thinking.

[13] Oliver: OnLine Inference and VERification system http://webwork.ug.cs.sunysb.edu:8080/servlet/Oliver.OliverStart

[14] Pizer, A., and Gage, M., *The WeBWorK System*, http://webwork.math.rochester.edu

[15] Roberts, E. Leblanc, R., Shackelford, R. Denning, P., Srimani, P. and Croos J., "Curriculum 2001: Interim Report from the ACM/IEE-CS Task Force", *Proceedings of the 30th SIGCSE Technical Symposium on Computer Science Education*, pp. 343--344, 199, New Orleans, Louisiana.

[16] Roberts, E., Cover, C., Chang, C., Engel, G., Mc Gettrick, A. and Wolz, U., "Computing Curricula 2001: How will it work for you?", *Proceedings of the 32nd SIGCSE Technical Symposium on Computer Science Education*, February 21-25, 2001, pp. 433-434.