# Clue Deduction: Professor Plum Teaches Logic

**Todd W. Neller**
Gettysburg College
Department of Computer Science
Gettysburg, PA 17325
tneller@gettysburg.edu

**Zdravko Markov**
Central Connecticut State University
markovz@ccsu.edu

**Ingrid Russell**
University of Hartford
irussell@hartford.edu

## Abstract

In this paper, we describe curricular materials that use the game of Clue to teach basic concepts of propositional logic. While there are many ways of teaching concepts of logic, this approach is distinct in that it is (1) goal-oriented, culminating in a fun project where students implement expert Clue deduction, (2) minimalistic, covering many important concepts within the simple domain of propositional logic, and (3) extensible, inviting students to extend the project further in several significant directions through advanced study of knowledge representation and reasoning topics.

## Introduction

This paper presents curricular materials for teaching knowledge representation and reasoning in an introductory artificial intelligence course. The project, "Clue Deduction: an introduction to satisfiability reasoning", is part of a larger project *Machine Learning Laboratory Experiences for Introducing Undergraduates to Artificial Intelligence (MLExAI)*[1]. An overview of this NSF-funded work and samples of other course materials developed under this grant are published in (Russell *et al.* 2005; Markov *et al.* 2005).

There are many ways to teach knowledge representation and reasoning (KR&R). The approach we describe here has a number of distinguishing features. First, it is **goal directed**, with the goal being to develop expert artificial intelligence for reasoning about the popular board game Clue®[2], a mystery-themed game of deduction. By featuring one of the last half-century's most popular board games in the world, we have chosen a goal that is both **fun** and **accessible**. Further, we have chosen to deal exclusively with *propositional logic only*, covering most core KR&R concepts without first-order logic complexities. Drawing from both AI texts and mathematics journals, we offer a high-quality selection of word problems for propositional logic. Finally, the core student project is **extensible** in several important ways, creating incentives for further exploration in constraint satisfaction and boolean satisfiability reasoning engines.

In the remainder of this paper, we describe the structure of our curricular materials at a high level, and present our preliminary teaching experiences using them. Handouts and supporting software are available at the project website[3].



[1] http://uhaweb.hartford.edu/compsci/ccli/

[2] A.k.a. Cluedo® in other parts of the world


Figure 1: The Game of Clue

## The Game of Clue

In this section we describe the game of Clue and explain why it is well-suited to the teaching and application of propositional logic.

The murder-mystery themed board game of Clue was invented by Anthony E. Pratt and first produced in the U.K. by Waddingtons as "Cluedo" in 1949. Since that time, Clue became one of the most popular family board games of the last half century.

In the back-story of the game, Mr. Boddy has been murdered in one of the rooms of his mansion Tudor Close. The game's object is to be the first to correctly deduce the correct murder suspect, weapon, and room. Three to six players arbitrarily assume roles of possible suspects, but a player's guilt or innocence does not affect play.

A deck of 21 cards depicts each of six possible suspects, six possible weapons, and nine possible rooms. One of each of these three card types is randomly chosen and placed, unseen by any player, into an envelope called the *case file*.

[3] http://uhaweb.hartford.edu/compsci/ccli/Clue/

These three cards are the unknown murder suspect, weapon, and room for this game. The remaining 18 cards are then shuffled and dealt clockwise to players. (In a four- or five-player game, cards will not be dealt evenly.)

Each turn, a player rolls dice to move about the board, visiting rooms and making suggestions about the contents of the case file. When a player makes a suggestion, opponents clockwise indicate that they cannot disprove the suggestion until an opponent has one or more of the suggested cards and must reveal one privately to the suggester. Each player may declare one accusation in the game, checking the case file to see if the contents have been correctly named. If correct, the player wins; if not, the player loses and continues to disprove suggestions when possible.

Thus, Clue is primarily a game of information, and successful players are skilled at gaining more information than their opponents, using it to deduce or guess further information. Atomic sentences for our propositional reasoning are of the form "Card $c$ is in place $p$," denoted $c_p$.

Students often share fond memories of playing Clue as children. Indeed, Clue is often considered a simple children's game. This is perhaps due to the fact that the game's detective notepads are too small to make substantial notes, and instructions for use of detective notepads merely encourage the player to simply mark the cards the player has seen. However, most information available in the game concerns cards *not* seen. It is important to note what is *not* held by an opponent, or that one of three suggested cards was shown by one opponent to another. In our experiences, few Clue players make such notes or appreciate the logic puzzles each game presents.

Applying logical reasoning to all basic propositional information in the game allows a player to make surprising expert deductions, reaching a winning accusation in a fraction of the time of a novice. More than a children's game, Clue is revealed as game of deeper deduction. Between nostalgia and mastery of a popular game, students enjoy a positive, fun experience as they learn fundamental concepts of logic.

## Propositional Logic

Our curricular materials begin with a concise presentation of the syntax and semantics of propositional logic. Examples are based on facts in the game of Clue and a simple liars/truth-tellers problem. A central running example in the introduction, conjunctive normal form, and resolution theorem proving is as follows:

> **Example Problem:** Suppose that liars always speak what is false, and truth-tellers always speak what is true. Further suppose that Amy, Bob, and Cal are each either a liar or truth-teller. Amy says, "Bob is a liar." Bob says, "Cal is a liar." Cal says, "Amy and Bob are liars." Which, if any, of these people are truth-tellers?

Using Clue facts and this problem, we cover atomic sentences, operators ($\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$), literals, propositional logic's BNF grammar, truth assignments, (un)satisfiability, models, validity, tautologies, entailment, and logical equivalence. Step by step, we show that the knowledge base of the example problem above can be expressed as:

$$\{A \Leftrightarrow \neg B,\ B \Leftrightarrow \neg C,\ C \Leftrightarrow \neg A \wedge \neg B\}$$

where the atomic sentences have the following interpretation:

- $A$ - Amy is a truth-teller.
- $B$ - Bob is a truth-teller.
- $C$ - Cal is a truth-teller.

## Conjunctive Normal Form (CNF)

In order to teach resolution theorem proving and understand the most common interface to propositional logic automated theorem provers, we demonstrate conversion to conjunctive normal form (CNF) using the liars/truth-tellers knowledge base:

1. **Eliminate $\Leftrightarrow$.** Replace each occurrence of $s_1 \Leftrightarrow s_2$ with the equivalent $(s_1 \Rightarrow s_2) \wedge (s_2 \Rightarrow s_1)$.

$$\left\{ \begin{array}{l} A \Rightarrow \neg B,\ \neg B \Rightarrow A,\ B \Rightarrow \neg C,\ \neg C \Rightarrow B, \\ C \Rightarrow \neg A \wedge \neg B,\ \neg A \wedge \neg B \Rightarrow C \end{array} \right\}$$

2. **Eliminate $\Rightarrow$.** Replace each occurrence of $s_1 \Rightarrow s_2$ with the equivalent $\neg s_1 \vee s_2$.

$$\left\{ \begin{array}{l} \neg A \vee \neg B,\ \neg\neg B \vee A,\ \neg B \vee \neg C,\ \neg\neg C \vee B, \\ \neg C \vee \neg A \wedge \neg B,\ \neg(\neg A \wedge \neg B) \vee C \end{array} \right\}$$

3. **Move $\neg$ inward.** We demonstrate de Morgan's law and double $\neg$ elimination.

$$\left\{ \begin{array}{l} \neg A \vee \neg B,\ B \vee A,\ \neg B \vee \neg C,\ C \vee B, \\ \neg C \vee \neg A \wedge \neg B,\ A \vee B \vee C \end{array} \right\}$$

4. **Distribute $\vee$ over $\wedge$.**

$$\left\{ \begin{array}{l} \neg A \vee \neg B,\ B \vee A,\ \neg B \vee \neg C,\ C \vee B, \\ \neg C \vee \neg A,\ \neg C \vee \neg B,\ A \vee B \vee C \end{array} \right\}$$

   Expressed as a set of sets of literals:

$$\left\{ \begin{array}{l} \{\neg A, \neg B\},\ \{B, A\},\ \{\neg B, \neg C\},\ \{C, B\}, \\ \{\neg C, \neg A\},\ \{\neg C, \neg B\},\ \{A, B, C\} \end{array} \right\}$$

## Resolution Theorem Proving

This then sets the stage for teaching resolution theorem proving. We describe *reductio ad absurdum*, i.e. proof by contradiction, and generalized *modus ponens*. We then take our example knowledge base and show how to prove that Cal is a liar by assuming he is a truth-teller and deriving a contradiction through resolution:

| | | | |
|---|---|---|---|
| (1) | $\{\neg A, \neg B\}$ | | Knowledge base |
| (2) | $\{B, A\}$ | | |
| (3) | $\{\neg B, \neg C\}$ | | |
| (4) | $\{C, B\}$ | | |
| (5) | $\{\neg C, \neg A\}$ | | |
| (6) | $\{\neg C, \neg B\}$ | | |
| (7) | $\{A, B, C\}$ | | |
| (8) | $\{C\}$ | | Assumed negation |
| (9) | $\{\neg A\}$ | (5),(8) | Derived clauses |
| (10) | $\{B\}$ | (2),(9) | |
| (11) | $\{\neg C\}$ | (3),(10) | |
| (12) | $\{\}$ | (8),(11) | *Contradiction!* |

## SATSolver, DIMACS CNF Format, and zChaff

Our project is structured in such a way that students may do as little as represent knowledge, and may go so far as to implement the entire reasoning system. We make use of zChaff[4], a free, efficient, complete, DPLL-style satisfiability solver (Moskewicz *et al.* 2001). Any solver may be used that accepts the standard DIMACS CNF format and reports whether the given knowledge base is satisfiable or not. We thus treat the underlying satisfiability reasoning engine as a black box which students may choose to implement for an even richer educational experience. Indeed, students working on this project at Gettysburg College implemented WalkSAT to replace zChaff as their satisfiability reasoning black box.

The instructor may choose leave the solver as an abstract black box, or delve into the chosen solver's algorithm. This project is intended as a flexible, customizable starting point that is minimally a knowledge acquisition and representation exercise, but could also motivate a deeper inquiry into reasoning algorithms.

We give a detailed description of `SATSolver`, a Java interface to zChaff, using Donald Knuth's *literate programming*[5] style. Using the tool `noweb`[6] we created a single source document in which code is defined in modular chunks throughout the body of the teaching materials. This source is used both to create the typeset project documentation, and to extract and assemble the corresponding source code. Whereas traditional commenting intersperses text throughout code, literate programming intersperses code throughout text, and gives the reader assurance that the code described and supplied is the same.

The complete implementation of the `SATSolver` is thus presented piece by piece, so that the user can make any changes necessary should a solver other than zChaff be desired. In the "Related Work" section, we discuss the tradeoffs of applying Prolog to Clue.

## Propositional Logic Exercises

At this point, students are prepared to exercise knowledge representation, resolution theorem proving, and automated theorem proving through the `SATSolver`. Drawing from a diverse set of artificial intelligence texts, logic texts, and mathematics journals, we selected and adapted eight excellent problems in propositional reasoning. Students are asked to approach each problem in four parts:

(a) Express all relevant problem facts as a propositional logic knowledge base. Clearly explain the meaning of the propositional symbols.

(b) Convert the propositional logic knowledge base to CNF.

(c) Use resolution theorem proving to solve the problem.

(d) Solve the problem computationally with a SAT solver.

A student successfully completing a few of these exercises should have a firm grasp of the main concepts up to this point.

## Clue Reasoner

Given the SATSolver and exercises in knowledge representation and conversion to CNF, we are now ready to create an expert Clue reasoner. A skeleton class `ClueReasoner` is supplied which contains all but knowledge needed for the knowledge base, so this is essentially a knowledge acquisition and representation exercise.

We mentioned earlier that each Clue atomic sentence $c_p$ symbolizes the statement "The card $c$ is in place $p$.". There is an atomic sentence for each place and card pair. For DIMACS CNF format, we provide a unique integer, i.e. a Gödel number[7], for each atomic sentence as follows. Suppose we have a place index $i_p$ and a card index $i_c$. Then the integer corresponding to $c_p$ is $i_p \times \texttt{numCards} + i_c + 1$. Since $i_c \in [0, \texttt{numCards} - 1]$ and $i_p \in [0, \texttt{numPlayers}]$ (including the case file), then each atomic sentence has a unique number from 1 through $(\texttt{numPlayers} + 1) \times \texttt{numCards}$.

For example, consider the atomic sentence $\text{pi}_{\text{wh}}$. The player Mrs. White ("wh") has index 2. The lead pipe card ("pi") has index 10. There are 21 cards. Therefore, the integer corresponding to the atomic sentence $c_p$ is $2 \times 21 + 10 + 1 = 53$.

The knowledge of Clue can be divided into two types: initial knowledge and play knowledge. Initial knowledge is that which is known right before the first turn and includes:

- Each card is in exactly one location.
- Exactly one card of each category is in the case file.
- You know your hand of cards.

Play knowledge is knowledge gained during events of the game:

- A player cannot refute a suggestion.
- A player refutes your suggestion by showing you a card.
- A player refutes another player's suggestion by showing them a card privately.
- A player makes an accusation and shares whether or not it was correct.

Whether initial or play knowledge, it is added immediately clause by clause to the knowledge base. Satisfiability testing is done with the knowledge base and each possible literal in turn in order to determine where cards are, are not, or possibly may be located. The results are printed in a table.

Students at Gettysburg College were the first to be assigned this project. To our surprise, the reasoning performance of the `ClueReasoner` was beyond that of the "expert" computer players of Atari's commercial product *Clue: Murder at Boddy Mansion*. In one game, several turns before an expert computer player was able to make a correct accusation, the assigned project system was able to deduce the contents of the case file given only the information known to a non-playing observer who *never sees any cards*.

[4] http://www.princeton.edu/%7Echaff/zchaff.html

[5] www.literateprogramming.com

[6] http://www.eecs.harvard.edu/%7Enr/noweb/

[7] http://en.wikipedia.org/wiki/G%C3%B6del_number

Students should be encouraged by the fact that they can both comprehend and implement a state-of-the-art reasoning system.

## Improving Performance

Until now, no effort was made (programmatically or on paper) to delete unnecessary clauses from the knowledge base. After completion of the project, we describe three simple types of unnecessary clauses that are often eliminated in the preprocessing stage of a reasoning engine: equivalent clauses, subsumed clauses, and tautology clauses. (We do not cover pure literal elimination, but supply references for further reading.) Examples are given from our liar/truth-teller example.

We also use the liar/truth-teller example to concretely show how the results of deduction may be added to a knowledge base, subsuming prior knowledge, streamlining the knowledge base, and improving reasoning performance. Thus, we naturally lead into a discussion of *deductive learning*.

Machine learning is the unifying theme for this set of introductory AI projects developed for the NSF. In his text *Machine Learning* (Mitchell 1997), Tom Mitchell defines *machine learning* as follows:

> A computer is said to learn from experience $E$ with respect to some class of tasks $T$ and performance measure $P$, if its performance at tasks in $T$, as measured by $P$, improves with experience $E$.

Having shown deductive learning to improve average expected performance of the reasoning task, we contrast deductive learning with *inductive learning* and discuss their inverse relation. Finally, we point out that our system is "told" facts, and that *knowledge acquisition* is a third important experience for augmenting and improving a knowledge-based system.

## Advanced Projects

For students who have completed this project, we provide brief descriptions of three possible directions students can proceed for advanced work.

First, we point out the fact that all previous reasoning omits important initial knowledge: the number of cards dealt to each player. This is common knowledge, and can be very important in practice. For example, suppose six players are playing with three cards each. You have been shown one of a player's cards, and have deduced 18 cards that the player does not have. This leaves two unknowns out of 21 cards. However, since the player holds three cards and you know 18 cards which are not held by the player, then all remaining cards must be held by the player. If you know all the cards a player does not have, you know all the cards a player does have, and vice versa.

There are a number of ways a student can represent this information about the number of cards dealt to each player. A first approach is to represent it in CNF, either in a simple combinatorially large form, or a compact adder-circuit form. A second approach is to perform such reasoning at a meta-level, repeatedly performing all other deductions until the meta-level can deduce nothing more. A third approach is to operate directly with *pseudo-boolean constraints* as a constraint satisfaction problem. Indeed, this would provide a good segue to the topic of constraint satisfaction.

A second possible advanced project would be to replace zChaff with the student's own simple DPLL-type solver. Whereas the project thus far has been largely concerned with knowledge representation, an implementation of the underlying reasoning engine would be an excellent educational experience as well.

Similarly, as a third possible advanced project, a student might opt to experiment with the implementation of a simple stochastic local search (SLS) based reasoning engine (e.g. WalkSAT). This was the direction taken at Gettysburg College. The practical application of incomplete WalkSAT search to Clue reasoning gave students a feel for the tradeoff of speed versus quality of the reasoning result. The more iterations WalkSAT is given, the more likely the algorithm will find satisfying truth assignments when they exist, answering knowledge base queries correctly.

## Assessment

This project was first assigned to Gettysburg College students in the fall of 2004. Projects were tested with a simulated game scenario unknown to the students. Of eleven projects, five had complete and correct deductions. Most other projects were missing one type of fact. For example, in adding a player's hand information to the knowledge base, a few students failed to add what cards were *not* in the player's hand.

Five students were independently interviewed and surveyed by an external evaluator for the NSF. Results were very positive. On a 1-5 scale ("strongly disagree", "disagree somewhat", "neither agree nor disagree", "agree somewhat", "strongly agree"), students responded to the questions of Figure 2.

Although this sample size is very small, the preliminary results of Figure 2 are encouraging. The curricular materials were revised according to recommendations by student participants and reviewing faculty. Additional broader assessment of the revised materials is forthcoming as the revised materials are currently being tested across a diverse set of institutions.

## Related Work

Prolog implementations of simple Clue reasoning systems are described in (Neufeld 2002) and (Emond & Paulissen 1986). A detailed implementation is described in (Emond & Paulissen 1986), highlighting the strengths and weaknesses of working with Prolog. Perhaps the greatest strength is that Prolog as a declarative logic programming language is well-suited to such knowledge representation and reasoning programming tasks.

However, two representational weaknesses in the implementation were highlighted by the authors. First, a lack of loop detection necessitates the creation of unintuitive, separate `has` and `owns` predicates which force acyclic search. That is, awkward necessary procedural "hacks"

| Question | Mean |
|---|---|
| Requirements for the student project were clearly presented and easy to follow. | 4.6 |
| The time allowed for the completion of the project was sufficient. | 5.0 |
| The student project was interesting to work on. | 4.2 |
| The student project contributed to my overall understanding of the material in the course. | 4.4 |
| The student project was at an appropriate level of difficulty given my knowledge of computer science and programming. | 4.6 |
| After taking this course I feel that I have a good understanding of the fundamental concepts in Artificial Intelligence. | 4.4 |
| Based on my experience with this course, I would like to learn more about the field of Artificial Intelligence. | 4.2 |
| The Artificial Intelligence problem solving techniques covered in this course are valuable. | 4.8 |
| I have a firm grasp of the problem solving techniques covered in this course. | 4.4 |
| I am confident that I can identify opportunities to apply these problem solving techniques. | 4.2 |
| I am confident that I can apply these problem solving techniques to different problems. | 4.2 |
| I had a positive learning experience in this course. | 4.8 |

Figure 2: Preliminary assessment

were necessary for successful search. Second, the restriction to Horn clauses necessitated predicates such as `has_at_least_one_of_three`.

Prolog would certainly lend itself to a more compact representation. However, assuming the students are do not already know Prolog, instructors should carefully weigh the cost of teaching a new programming language in the context of an AI course.

## NSF Project

This Clue project is one of several projects developed for *Machine Learning Laboratory Experiences for Introducing Undergraduates to Artificial Intelligence (MLExAI)*, a joint effort between the University of Hartford, Central Connecticut State University, and Gettysburg College. Instructors can access projects like this Clue project through the project website[8].

The importance of AI in the undergraduate computer science curriculum is illustrated by the Computing Curricula 2001 recommendation of ten core units in AI (Engel & Roberts 2001; Russell & Neller 2003). It is generally recognized that an undergraduate introductory AI course is challenging to teach (Hearst 1995). This is, in part, due to the diverse and seemingly disconnected core topics that are typically covered. Recently, work has been done to address the diversity of topics covered in the course and to create a theme-based approach. Several faculty have been working to integrate Robotics into the AI course (Kumar 2001; Kumar & Meeden 1998). Russell and Norvig use an agent-centered approach (Russell & Norvig 2003), while Nilsson uses an evolutionary-based approach (Nilsson 1998).

In MLExAI, we have created an adaptable framework for the presentation of core AI topics through a unifying theme of *machine learning*. A suite of hands-on projects have been developed, each involving the design, implementation, and/or use of an applied learning system.

Our work will undoubtedly draw comparison to the introductory text by Nils Nilsson, *Artificial Intelligence: a New Synthesis*, which takes an evolutionary, agent-based approach to topic unification with a heavy emphasis on machine learning (Nilsson 1998). The text is geared toward a student with significant prior coursework in mathematics. Being mathematically focused, the text places almost no emphasis on application of ideas through implementation. Indeed, the preface states, "Although some pseudocode algorithms are presented, this book is not an AI programming and implementation book."

In contrast, our experiential approach, as exemplified by this Clue project, allows for varying levels of mathematical sophistication with the implementation of concepts being central to the learning process.

## Conclusion

Creating a state-of-the-art reasoning engine for one of the world's most popular (and least analyzed) board games is truly a unique and rewarding experience. Moreover, the ability to accomplish this in the context of an introductory AI course while teaching core concepts of knowledge representation and reasoning (KR&R) is both rewarding and encouraging.

While first-order logic has its place in AI education, we believe that a student's first exposure to KR&R concepts should be in the simple context of propositional logic. With an increasing trend towards translating planning and constraint satisfaction problems to boolean satisfiability problems(Kautz & Selman 1996; Gent 2002), this approach also provides students with immediate practical tools to encourage application and further learning. Our minimalism is also motivated by the fact that introductory AI courses often present a diverse set of topics within tight time constraints.

Finally, we note that the Clue project ends at a crossroads. Students are invited to pursue further elaboration of the project into a variety of reasoning engines, including more general constraint satisfaction search. The extensibility of this project lends itself to a variety of instructional purposes.

Instructors are encouraged to visit the Clue project website[9], and explore other experiential teaching materials for introductory AI at the MLExAI project page[10].

[8] http://uhaweb.hartford.edu/compsci/ccli/

[9] http://cs.gettysburg.edu/%7Etneller/nsf/clue/index.html

[10] http://uhaweb.hartford.edu/compsci/ccli/

## Acknowledgment

This work is supported in part by National Science Foundation grant *DUE CCLI-A&I Award Number 0409497*.

## References

Emond, J.-C., and Paulissen, A. 1986. The art of deduction: a simple program that demonstrates the deductive power of prolog. *Byte* 11:207–214.

Engel, G., and Roberts, E., eds. 2001. *Computing Curricula 2001 Computer Science*. IEEE Press. Available at http://www.sigcse.org/cc2001/.

Gent, I. P. 2002. Arc consistency in SAT. In van Harmelen, F., ed., *Proceedings of the 15th Eureopean Conference on Artificial Intelligence, ECAI'2002, Lyon, France, July 2002*. IOS Press.

Hearst, M., ed. 1995. *Improving Instruction of Artificial Intelligence: Papers from the 1994 AAAI Fall Symposium*. AAAI Press.

Kautz, H., and Selman, B. 1996. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96) and The Eighth Annual Conference on Innovative Applications of Artificial Intelligence, August 4-8, 1996, Portland, Oregon*. AAAI Press.

Kumar, D., and Meeden, L. 1998. A robot laboratory for teaching artificial intelligence. In *Proceedings of the 29th SIGCSE technical symposium on computer science education*, 341–344.

Kumar, A. N. 2001. Using robotics in an undergraduate artificial intelligence course: An experience report. In *Proceedings of the 31st ASEE/IEEE Frontiers in Education Conference, October 10-13, 2001, Reno, NV*, 10–14. Session T4D.

Markov, Z.; Russell, I.; Neller, T.; and Coleman, S. 2005. Enhancing undergraduate ai courses through machine learning projects. In *Proceedings of the 35th ASEE/IEEE Frontiers in Education Conference (FIE'05), Indianapolis, IN*. IEEE Press.

Mitchell, T. M. 1997. *Machine Learning*. New York, New York, USA: McGraw-Hill.

Moskewicz, M.; Madigan, C.; Zhao, Y.; Zhang, L.; and Malik, S. 2001. Chaff: Engineering an efficient sat solver. In *Proceedings of the 39th Design Automation Conference (DAC 2001)*.

Neufeld, E. 2002. Clue as a testbed for automated theorem proving. In Cohen, R., and Spencer, B., eds., *Lecture Notes in Computer Science 2338: Advances in Artificial Intelligence: 15th Conference of the Canadian Society for Computational Studies of Intelligence, AI 2002, Calgary, Canada, May 27-29, 2002. Proceedings.* Berlin: Springer. 69–78.

Nilsson, N. 1998. *Artificial Intelligence: A New Synthesis*. Morgan Kaufmann.

Russell, I., and Neller, T. 2003. Implementing the intelligent systems knowledge units of computing curricula 2001. In *Proceedings of Frontiers in Education Conference (FIE 03), Boulder, Colorado, November 5-8, 2003*. IEEE Press.

Russell, S., and Norvig, P. 2003. *Artificial Intelligence: a modern approach, 2nd ed.* Upper Saddle River, NJ, USA: Prentice Hall.

Russell, I.; Markov, Z.; Neller, T.; Georgiopoulos, M.; and Coleman, S. 2005. Unifying an introduction to artificial intelligence course through machine learning laboratory experiences. In *Proceedings of the 25th American Society for Engineering Education Annual Conference and Exposition (ASEE'05)*. ASEE Press.