INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD)
© 2026 IJNRD | Volume 11, Issue 7, July 2026 | ISSN: 2456-4184 | IJNRD.ORG

Knights harder than knaves: a view of proof complexity on set of support

[1]Ruo Ando, [2]Yoshiyasu Takefuji

[1]National Institute of Informatics, [2]Keio University
[1] 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, [2] 5322 Endō, Fujisawa, Kanagawa 252-0882

Abstract : This paper investigates proof complexity in the Knights and Knaves problem under the set of support strategy using a resolution-based theorem prover. The problem is formulated as a satisfiability problem in first-order logic, and represented using an n-ary tree structure. Hyper-resolution and the set of support strategy are employed to generate proofs, and the number of generated clauses is analyzed as a measure of computational cost. Experiments were conducted on 1,296 possible patterns of truth-tellers and liars. Although the histogram of the number of generated clauses exhibits a unimodal distribution, by classifying the cases according to the number of knights, ranging from zero to four, we discovered an interesting underlying structure. The distributions of computational complexity for cases involving zero to four knights overlap considerably. In other words, even when the number of knights differs, cases yielding the same number of generated clauses can be found across a broad range of the distributions. Most notably, the variance in the zero-knight case is remarkably smaller than those observed in the cases involving one to four knights. We examine this distinctive feature from the perspectives of circularity in inference and fixed-point theory.

IndexTerms **– CNF-SAT, hyper resolution, set of support, self-reference**

INTRODUCTION

The Knights and Knaves puzzle can be formulated as a finite SAT problem by representing each character as a Boolean variable, where 1 denotes a truth-teller and 0 a liar. If a person A makes a statement P, this is expressed as

$$A \leftrightarrow P.$$

Solving the puzzle corresponds to finding an assignment that satisfies the conjunction of all such constraints. For example, if A says B is a liar" and B says A is a truth-teller," we obtain

$$A \leftrightarrow \neg B, \quad B \leftrightarrow A.$$

Determining whether these formulas are satisfiable is equivalent to solving the puzzle. If a satisfying assignment exists, the instance is SAT; otherwise, it is UNSAT.

```
1: list(usable).
2: P(G(A)) | P(G(B)) | P(G(C)) | P(G(D)).
3:
4: -P(G(A)) | -P(G(B)).
5: -P(G(A)) | -P(G(C)).
6: -P(G(A)) | -P(G(D)).
7:
8: -P(G(B)) | -P(G(C)).
9: -P(G(B)) | -P(G(D)).
10: -P(G(C)) | -P(G(D)).
11:
12: P(G(x)) | P(I(x)).
13: -P(G(x)) | -P(I(x)).
14:
15: -P(y) | -P(says(x,y)) | P(T(x)).
16: end_of_list.
```

This script is a set of clauses for a first-order logic theorem prover such as OTTER, encoding constraints on individuals, their properties, and the relation between statements and truth. The section between list(usable). and end_of_list. defines the usable clause set. Line 2 states that at least one of A, B, C, D has property G, while lines 4--10 prohibit any pair from simultaneously having G. Together, these clauses enforce that exactly one of the four individuals has property G.

Lines 12 and 13 introduce a binary classification for any object x: it must belong to either G or I, but not both. This forms an exclusive partition, similar to the distinction between truth-tellers and liars. Line 15 encodes a rule relating statements and truth: it corresponds to P(y) and P{says(x,y)) -> P(T(x)), meaning that if a statement y is true and x asserts y, then x has property T.

Overall, the script represents a logical framework combining exclusive properties, cardinality constraints, and a rule connecting statements to speaker attributes. The first part resembles a finite SAT structure over a fixed set of individuals, while the last clause introduces a general first-order rule involving variables.

FORMULATION

3.1 Population and Sample

A rooted tree (or n-ary tree) is a hierarchical graph structure consisting of nodes and edges, with one distinguished root node. Each node can have zero or more children, and there is exactly one path from the root to every node. The structure is connected and contains no cycles, meaning it forms an acyclic tree graph. Nodes with children are called internal nodes, while nodes without children are called leaf nodes.

Table 1 N-ary tree of the problem

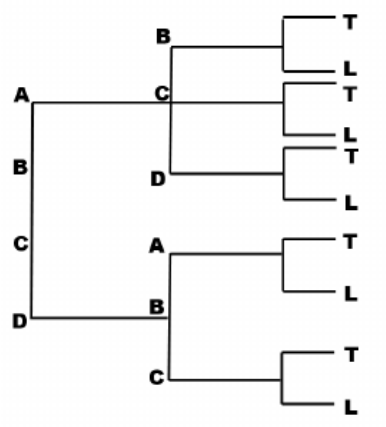


Table 1 shows the Knights and Liars problem represented as an n-ary tree. The first level of branches indicates which of A–D is the speaker. The second level indicates whom speakers A–D are talking about. The leaf nodes indicate whether the referenced person is honest or a liar.These three levels of branching make it possible to represent the Knights and Liars problem. For example, the first branch in Figure 1 indicates that speaker A says that B is honest. In clause form, this can be written as follows:

```
P(says(A,T(B))).
```

When the Knights and Liars problem with four speakers is represented using an n-ary tree, the tree consists of a total of 24 branches (leaves).

3.1 Hyperresolution

The hyper-resolution inference rule is most frequently used one. Hyper-resolution takes a non-positive clause which is called as the nucleus and simultaneously infers each of its negative literals. Those negative literals are called as the satellites. Hyper-resolution can be regarded as a sequence of positive binary resolution steps yielding a positive clause.

$$\frac{(\neg L_1 \vee \cdots \vee \neg L_m \vee L_{m+1} \vee \cdots \vee L_l),\ K_1, \ldots, K_m}{(L_{m+1} \vee \cdots \vee L_l \vee K_1' \vee \cdots \vee K_m')\sigma}$$

The general scheme is:

$$K_{1,l}, ., K1, n$$

...

$$Km, l, ., Km, n$$

$$\frac{\{\neg L_1, ., \neg L_{m+1}, L_l\}\exists\sigma.\sigma}{= mgu([|K_1|, ., |K_{m,1}|], [|L_1|, ., |L_m|])}$$

$$\{K_{1,2}, ., K_{1,n}, K_{m,2}, .K_{m,n}, L_{m+1}, ., L_l\}\sigma$$

In the list above, K_i denotes clause and L_i denotes literal. Hyper-resolution is applied to a set of m unit clauses {K1} ... {Km} and a single nucleus {L1, ..., Lm+1} consisting of m + 1 literals. Hyper-resolution is most frequently adopted inference rule in the situations where equality substitutions do not play a major role. Hyper-resolution is intuitively natural to human reasoning.

3.1 Set of support

The set of support strategy \cite{Wos2} guides the reasoning program to select from the clauses characterizing the question under research to be put in the initial set of support list which is denoted as $ list(sos)$ in OTTER. The corresponding restriction prevents the reasoning program from adopting an inference rule to a set of clauses of which all clauses are complement of the set of support. Consequently, each clause generated and retained is appended to $list(sos)$. In \cite{Wos2}, experimentally, it is pointed out that the most effective choice for the initial set of support is based on the special hypothesis and the denial of the theorem under research. The second best choice is the denial of the theorem itself.

We divide the set of clauses used in the proof into:

$$S = U \cup T,$$

Let \(G\) be the proposition to be proved. Instead of proving

$$U \models G$$

directly, we show that

$$U \cup \{\neg G\}$$

is unsatisfiable. Thus,

$$U \models G \quad \Longleftrightarrow \quad U \cup \{\neg G\} \models \bot.$$

The initial set of support is

$$T_0 = \mathrm{CNF}(\neg G).$$

The central formulas of the Set-of-Support strategy are the parent-clause restriction

$$C_1 \in T \vee C_2 \in T,$$

and the update rule

$$T_{k+1} = \rho\left(T_k \cup \{\mathrm{Res}(C, D) \mid C \in T_k,\ D \in U \cup T_k\}\right).$$

The Set-of-Support strategy reduces unnecessary resolution steps by focusing the proof search on clauses related to the negated goal \cite{Wos2}.

EXPERIMENTAL RESULT

Table 2 shows a histogram constructed from 1,296 possible numbers of generated nodes. Overall, the histogram exhibits a unimodal shape, with a peak roughly in the range of 68–76 generated nodes. However, as shown later in Figures 2 and 3, the distributions of the number of generated nodes (in terms of both mean and variance) differ depending on whether there are one to four honest participants. Therefore, the unimodal shape observed in Figure 1 is considered to be a composite of these four distributions. In addition, when focusing only on the higher range of generated node counts (88–105), the distribution exhibits a slightly long-tailed shape.

Table 2 Histogram of generated clauses

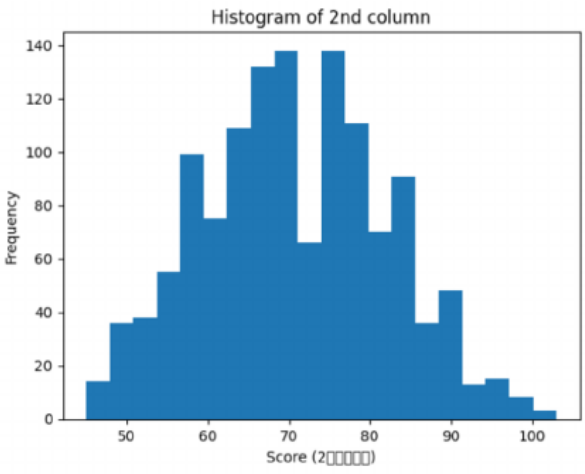


Table 3 is a graph in which the number of generated clauses is plotted on the vertical axis, while the 16 possible combinations of knights and liars are plotted on the horizontal axis. This makes it possible to observe how the characteristics of the distribution of generated clauses, such as the mean and variance, change depending on the number of knights. From A to E, the proportion of knights increases from 0 to 4. Looking at the averages, there are clear differences among A through E, and as the number of knights increases, the computational cost (the number of generated clauses) tends to rise. Furthermore, even when the number of knights is the same, variations can be observed in both the mean and variance of the generated clause count depending on which individuals are knights.

In general, the variance of the generated clause count tends to become larger when there are two or three knights.
When there are no knights, shown on the far right of the plot, both the mean and the variance of the generated clause count are the smallest. It can also be observed that, as the number of knights increases from 0 to 4, the mean number of generated clauses gradually increases, whereas the variance does not increase significantly.

Table 3 Patterns and the number of generated clauses

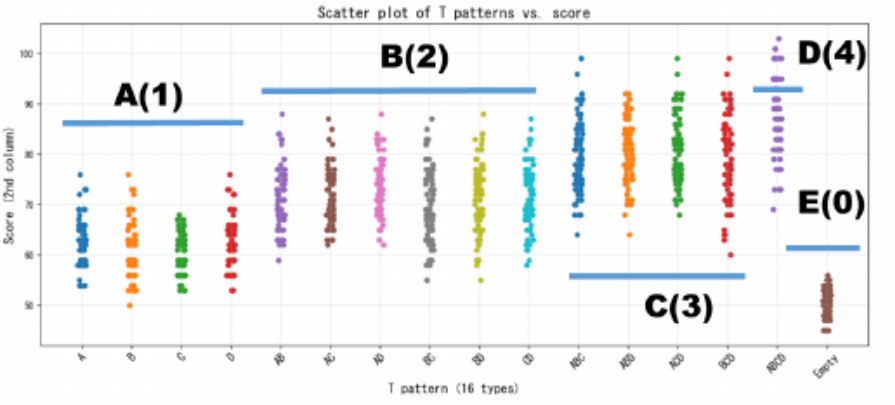
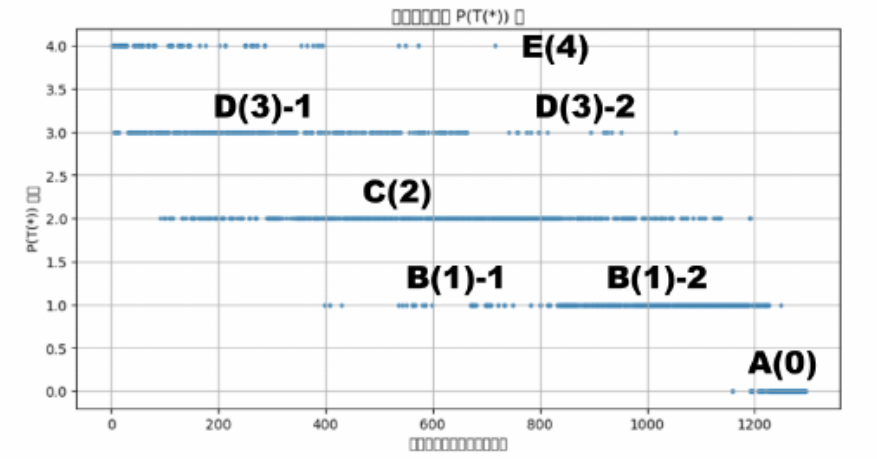


Table 4 plots all 1,296 patterns on the horizontal axis and the number of knights on the vertical axis. The 1,296 patterns on the horizontal axis are sorted in descending order according to the number of generated clauses. In other words, the left side of Figure 4 corresponds to patterns with a large number of generated clauses, whereas the right side corresponds to patterns with a small number of generated clauses.

Overall, the pattern group C(2), corresponding to cases with two knights, exhibits the largest variance. These patterns take relatively large values over a wide range, from approximately the top 100 to around the 1,200th position in terms of the number of generated clauses. The pattern groups with three knights (D) and one knight (B) exhibit a rough symmetry. Specifically, the D patterns are densely concentrated on the left side of the figure, while the right side becomes sparse. The density of D(3)-1 is greater than that of D(3)-2.

Conversely, the B pattern group is sparse on the left side and dense on the right side. That is, the density of B(1)-1 is smaller than that of B(1)-2. Finally, the pattern group with zero knights (A), similarly to Figure 3, has both a small mean and a small variance. The points corresponding to the generated clause counts and pattern numbers of group A are clustered in the lower-right region of Figure 4.

Table 4 Knights and the sorted number of clauses



DISCUSSION

Variability in computational complexity. Although the proposed method does not employ probabilistic techniques, variability in computational complexity is observed. Simply put, intuitively speaking, there are five possible numbers of honest individuals (0 to 4), and one would expect there to be only five distinct computational complexities corresponding to these. In reality, however, the computational complexity output varies depending on the number of honest individuals, exhibiting different distributions (mean and variance).

The reason for this is that the order of the symbols (literals) in the clauses influences the inference process, and furthermore, there is a possibility that some form of interaction occurs between clauses within the support set during the inference process. The problem

addressed in this paper is k-SAT, and whilst the computational complexity is generally defined as a scalar value of $ O(m 2^n)$, in practice it consists of a composite of five distributions.

Furthermore, these five distributions exhibit certain characteristics. The variance is greatest when there are two honest players, and smallest when there are zero honest players. This suggests that some form of interaction may occur during the derivation process of clauses within the support set. Furthermore, there are upper and lower bounds on the computational complexity: below a certain threshold, cases with one or more honest players cannot be handled, and conversely, above a certain threshold, cases with four or more honest players cannot be handled.

The situation in which all four speakers are liars is not merely a characterization of personalities; in mathematical logic, it can be regarded as a form of mutual self-reference or cyclic self-reference. When the four individuals make statements about one another, the truth value of each person's statement depends on the truth values of the others, and these dependencies may form a closed cycle. For example, if A speaks about B, B about C, C about D, and D about A, the dependency graph becomes A → B → C → D → A. In such a structure, determining the truth value of one individual eventually affects, and is affected by, that same individual through the cycle. Consequently, the entire system acquires a self-referential character.

In contrast, if at least one of the four individuals is truthful, that person's statement serves as a fixed reference point for determining truth values. This partially breaks the cyclic dependency and allows the truth values of the remaining individuals to be determined sequentially. Therefore, while the "all four are liars" case constitutes a self-referential system governed entirely by negative dependencies, the presence of at least one truthful speaker transforms the problem into a more stable logical structure with a fixed point.

It deals with a more indirect form of self-reference. Even if \(A\) does not directly make a statement about itself, its judgment eventually returns to \(A\) through the following path:

$$A \to B \to C \to D \to A.$$

Therefore, this structure can be expressed as the following system of logical equations:

$$\begin{aligned} x_A &= f_A(x_B), \\ x_B &= f_B(x_C), \\ x_C &= f_C(x_D), \\ x_D &= f_D(x_A). \end{aligned}$$

By successively substituting these equations into one another, we obtain

$$\begin{aligned} x_A &= f_A(x_B) \\ &= f_A\left(f_B(x_C)\right) \\ &= f_A\left(f_B\left(f_C(x_D)\right)\right) \\ &= f_A\left(f_B\left(f_C\left(f_D(x_A)\right)\right)\right). \end{aligned}$$

If we define the composite function:

$$F = f_A \circ f_B \circ f_C \circ f_D,$$

then the equation can be written more simply as:

$$x_A = F(x_A).$$

Thus, the problem ultimately becomes a fixed-point problem. In other words, \(x_A\) must be a value that remains unchanged under the mapping \(F\).

This structure represents indirect self-reference. Although \(A\) does not refer to itself directly, the chain of dependencies forms a cycle, causing the judgment made by \(A\) to return to \(A\) itself. Therefore, the entire cyclic inference process can be understood as a search for a fixed point ofthe composite logical mapping \(F\).
In contrast, if at least one of the four individuals is truthful, that person's statement serves as a fixed reference point for determining truth values. This partially breaks the cyclic dependency and allows the truth values of the remaining individuals to be determined sequentially. Therefore, while the "all four are liars" case constitutes a self-referential system governed entirely by negative dependencies, the presence of at least one truthful speaker transforms the problem into a more stable logical structure with a fixed point.

Such problems can be formalized using Boolean variables and transformed into CNF-SAT instances. Moreover, the notion of cyclic self-reference underlying these puzzles is closely related to profound results in mathematical logic, including Gödel's incompleteness theorems and Halting problem. Thus, these seemingly simple liar puzzles provide an instructive example of fundamental concepts in modern logic and the theory of computation.

RELATED WORK

Logic puzzles involving truth-tellers, liars, and self-referential statements have long been studied as examples of formal reasoning. Smullyan presented numerous puzzles of this type and demonstrated how apparently informal statements can be translated into precise logical constraints [1]. These puzzles frequently involve dependencies among speakers, direct or indirect self-reference, and consistency conditions across multiple statements. More recently, Rakshit and Dwivedi examined algorithmic methods for solving logic puzzles by converting natural-language conditions into computational representations [2]. Their work demonstrates the close relationship between recreational logic puzzles and automated reasoning. However, the main objective of these studies is generally to identify a correct interpretation or solution, rather than to analyze the computational behavior of the inference process.

The theoretical foundation of resolution-based automated deduction was established by Robinson [3]. The resolution principle provides a uniform inference rule for proving the unsatisfiability of a set of clauses and has become one of the central techniques in automated theorem proving. Bachmair and Ganzinger later provided a systematic treatment of resolution theorem proving, including inference restrictions, clause ordering, redundancy elimination, and completeness [4]. These studies form the logical basis for clause-based reasoning systems.

Although unrestricted resolution is complete, the generation of all possible resolvents can cause a rapid expansion of the search space. To control this growth, Wos, Robinson, and Carson proposed the set of support strategy [5]. This strategy requires at least one parent clause in each inference step to belong to a designated support set. Under appropriate conditions, it preserves completeness while reducing the number of unnecessary resolution steps. It is especially useful when the original axioms are regarded as consistent and clauses derived from the negation of the target theorem are placed in the support set.

The practical usefulness of resolution and the set of support strategy has been demonstrated in automated theorem-proving systems such as OTTER [6]. OTTER implements resolution, hyper-resolution, paramodulation, demodulation, and several search-control mechanisms. Its design illustrates that theorem-proving performance depends not only on logical validity but also on clause selection, formula representation, literal ordering, and inference strategy.

The computational difficulty of resolution has also been studied within proof complexity. Haken proved that resolution proofs of the pigeonhole principle require exponential size [7]. This result established that some unsatisfiable formulas are inherently difficult for resolution, regardless of implementation improvements. Ben-Sasson and Wigderson later showed a fundamental relationship between proof length and proof width [8]. Their results indicate that the size of a resolution proof is strongly affected by the structural properties of clauses and intermediate resolvents.

CONCLUSION

This study examined the proof complexity of the Knights and Knaves problem under the set of support strategy using a resolution-based theorem prover. The problem was formulated as a satisfiability problem in first-order logic and represented using an n-ary tree structure. Hyper-resolution and the set of support strategy were applied to investigate how different configurations of truth-tellers and liars influence the proof-generation process.

Experimental results on 1,296 patterns showed that the number of generated clauses strongly depends on the number and arrangement of knights. In particular, patterns containing two or three knights tended to exhibit larger variance in generated clause counts, indicating greater proof complexity and instability in the search process. In contrast, patterns with no knights showed both small mean and variance, suggesting relatively simple proof structures.

Although the histogram of the number of generated clauses is unimodal overall, the distribution exhibits distinct characteristics depending on the number of knights. When there are one to four knights, the range between the minimum and maximum numbers of generated clauses is approximately 780 to 1,100, although the regions of higher and lower density differ across the cases.The zero-knight case is an exception. In this case, the difference between the minimum and maximum values is limited to approximately 100, resulting in an extremely small variance. We hypothesize that this is because no fixed point exists in the circular inference process.

The results also revealed approximate symmetry between the distributions for one-knight and three-knight cases. Such tendencies imply that proof complexity in automated reasoning is not determined solely by problem size, but also by the structural distribution of logical constraints. Future work includes extending the analysis to larger numbers of participants, comparing different inference strategies, and applying statistical or probabilistic methods to characterize proof-search complexity in automated deduction systems.

REFERENCES

[1] Smullyan, R. M. What Is the Name of This Book? Prentice-Hall, 1978.
[2] Rakshit, U. and Dwivedi, N. “What Is the Title of This Paper? Solving Logic Puzzles Using Algorithms.” arXiv preprint arXiv:2309.13044, 2023.
[3] Robinson, J. A. “A Machine-Oriented Logic Based on the Resolution Principle.” Journal of the ACM, Vol. 12, No. 1, pp. 23-41, 1965.
[4] Bachmair, L. and Ganzinger, H. “Resolution Theorem Proving.” In Handbook of Automated Reasoning, A. Robinson and A. Voronkov, eds., Elsevier and MIT Press, pp. 19-99, 2001.

[5] Wos, L., Robinson, G. A., and Carson, D. F. "Efficiency and Completeness of the Set of Support Strategy in Theorem Proving." Journal of the ACM, Vol. 12, No. 4, pp. 536-541, 1965.
[6] McCune, W. OTTER 3.3 Reference Manual. Argonne National Laboratory, 2003.
[7] Haken, A. "The Intractability of Resolution." Theoretical Computer Science, Vol. 39, pp. 297-308, 1985.
[8] Ben-Sasson, E. and Wigderson, A. "Short Proofs Are Narrow—Resolution Made Simple." Journal of the ACM, Vol. 48, No. 2, pp. 149-169, 2001.