EPIC (Journal of Electrical Power, Instrumentation and Control)
Teknik Elektro – Universitas Pamulang
Vol. 8, No. 2, Bulan Desember, Tahun 2025, Hal 302 – 312
http://openjournal.unpam.ac.id/index.php/jit
DOI: 10.32493/epic.v8i2.58721

pISSN 2615-0646
eISSN 2614-8595

Implementation of Constraint Satisfaction Problem Approach for Solving the Zebra Puzzle with Prolog

Tri Bagus Kurniawan[1], Muhammad Risqi Nuryana[2] , Resi Sujiwo Bijokangko[3]

[1]Bachelor's Degree Department Electrical Engineering, Faculty of Electrical Engineering
[1]Mercu Buana University, Jl. Meruya Selatan No.1, DKI Jakarta, Indonesia
[2]Master's Department Electrical Engineering, Faculty of Bioscience, Technology and Information
[2]Atmajaya Catholic University Indonesia, Jl Jend Sudirman Kav. 21, DKI Jakarta, Indonesia
[3]Doctoral Department Electrical Engineering, Faculty of Electrical Engineering
[3]Mercu Buana University, Jl. Meruya Selatan No.1, DKI Jakarta, Indonesia

[1]41423120040@student.mercubuana.ac.id
[2]muhamma.12025004200@student.atmajaya.ac.id
[3]67325010004@student.mercubuana.ac.id

ARTICLE INFORMATION

submitted : 23-11-2025
revised : 01-12-2025
accepted : 25-12-2025
published : 31-12-2025

ABSTRACT

Constraint Satisfaction Problem (CSP) Is a fundamental concept in artificial intelligence used to solve combinatorial problems by satisfying a set of constraints. The Zebra Puzzle is a classical example that involves assigning attributes such as nationality, house color, pet, drink, and candy to five houses based on logical clues. This study aims to implement CSP techniques using the Prolog programming language to solve the Zebra Puzzle. The methodology includes classifying all relevant attributes, encoding constraints into Prolog predicates, and applying helper functions to represent adjacency and ordering relationships. The puzzle is modeled as a list of structured facts, and Prolog's inference engine is used to derive consistent solutions. The program successfully assigns all attributes to the correct house positions. For example, the Englishman lives in the red house, the Spaniard owns the dog, and the Ukrainian drinks tea. The Norwegian lives in the first house and drinks water, while the Japanese owns the zebra. These results confirm the effectiveness of Prolog in solving structured logic puzzles using CSP. The study concludes that logic programming is a reliable tool for modeling and solving constraint-based problems. This implies broader applicability of CSP in intelligent systems.

Keywords : Constain Satifiscation Problem; Zebra Puzzle; Prolog; Knowledge Representation

EPIC (Journal of Electrical Power, Instrumentation and Control)
Teknik Elektro – Universitas Pamulang
Vol. 8, No. 2, Bulan Desember, Tahun 2025, Hal 302 – 312
http://openjournal.unpam.ac.id/index.php/jit
DOI: 10.32493/epic.v8i2.58721
pISSN 2615-0646
eISSN 2614-8595

INTRODUCTION

Industries and sectors have begun to adopt AI technologies in contemporary society. AI assists people in achieving new levels of productivity. Examples of productivity AI include ChatGPT, Gemini, and Hailou. AI's ability to enhance productivity stems from the capacity to overcome challenges rapidly and formulate solutions with high precision (Siti Masrichah, 2023)

The Constraint Satisfaction Problem (CSP) is one of the most impactful techniques within AI for resolving complicated challenges. CSP is the ability to mathematically construct problems through the lens of variables with value domains and a set of conditions within the systems that need to remain intact (Jendreiko, 2024). In that manner, solutions are attainable through the productive satisfaction of each condition. Scheduling, planning, optimizations, and even the resolution of analytical puzzles are some applications of CSP (Szomiu & Groza, 2021)

One example of a puzzle that is often used as a case study to illustrate how CSP works is the Zebra Puzzle. Solving this puzzle requires the use of logic to assign the correct characteristics to each of the 5 individuals (house, pet, drink, candies, and nationality) based on the clues given to you (Berman et al., 2024). Using CSP modeling, this puzzle can be solved reliably and efficiently every time (Shah et al., n.d.). This research is based on the Zebra Puzzle and solves it using the Prolog programming language, which is considered a good programming language for logic programming and rule based inference.

One solution for solving CSP puzzles is to use Prolog. Prolog itself is a deductive programming language that operates under a specific form of logic called first order predicate logic (Buscaroli et al., 2023). This programming language was first created by Alain Colmerauer and Robert Kowalski in the 1970s (Körner et al., 2022). Since then, Prolog has become one of the most prominent programming languages in Knowledge-Based Systems and has since migrated to Artificial Intelligence (AI) in fields such as expert systems, rule-based systems, and Natural Language Processing (Yang et al., 2024).

Prolog have acontribution to artificial intelligence is exceptional because, among programming languages, it is declarative, while most others are procedural and concerned with how to accomplish a task. Prolog programmers merely have to specify the facts and the logical rules that characterize the system's knowledge (Schrijvers et al., 2023). Then, using strategies such as depth first search and backtracking, the Prolog inference engine looks for answers or tests knowledge-based hypotheses and validates them. Its remarkable efficiencies are useful for undertaking functions in fields that are predominantly logical in nature and require the manipulation of symbols, such as expert systems, early NLP, and, of course, the implementation and resolution of CSP models (Grohe et al., 2018; Thompson et al., 2025; White et al., 2024).

Then, considering that many people are now using Prolog to solve logic puzzles, one platform that can be used is SWISH (Wielemaker et al., 2021). Using SWISH enables collaboration, learning, and

EPIC (Journal of Electrical Power, Instrumentation and Control)
Teknik Elektro – Universitas Pamulang
Vol. 8, No. 2, Bulan Desember, Tahun 2025, Hal 302 – 312
http://openjournal.unpam.ac.id/index.php/jit
DOI: 10.32493/epic.v8i2.58721
pISSN 2615-0646
eISSN 2614-8595

experimentation because it is browser-based and does not require local installation. With SWISH, Prolog can also become more user-friendly and accessible to a wider audience, both for academic and research purposes. Based on the previous explanation, SWISH itself can be defined as a web interface that allows users to write, run, and share Prolog programs interactively (*SWISH -- Prolog_tutorials.Swinb*, n.d.; Wielemaker et al., 2015)

The presence of SWISH strengthens Prolog's position as a logic programming language that is not only classic, but also adaptive to modern technological developments. This makes SWISH a bridge between logic programming theory and the practice of global knowledge sharing (Wielemaker et al., 2020).

LITTERATURE STUDY

A. Zebra Puzzle Constraint Satisfaction Problem (CSP)

The Zebra Puzzle, also known as Einstein's Riddle, is a classic case in logic that serves as an excellent case study for testing reasoning and AI algorithms. Its basic structure involves assigning variables (such as house, color, nationality, animal, drink, and candies) to specific value domains based on a series of interrelated logical constraints (Jamine, 2024).

In the context of artificial intelligence, such puzzles are used as reliable benchmarks. Studies show that good logic puzzles have key characteristics, namely that all information provided is necessary and there is no redundant information. This property makes the Zebra Puzzle a challenging and ideal problem for deep reasoning tasks, such as Natural Language Inference (NLI) and machine comprehension (Szomiu & Groza, 2021).

Computational solving of the Zebra Puzzle is effectively reduced to a Constraint Satisfaction Problem (CSP). CSP modeling requires the definition of:

- Variables (e.g., the positions of houses 1 through 5).
- Domains (possible values for each variable, such as colors: red, green, blue, etc.).
- Constraints (logical rules connecting variables, e.g., “The Englishman lives in the red house”).

The goal of CSP is to find one or more assignments of values to the variables such that all constraints are satisfied. Thus, the Zebra Puzzle is a fundamental example in validating the effectiveness of logic-based computational approaches to solving industrial problems such as scheduling, resource allocation, or data analysis (Krokhin & Opršal, 2022).

B. The Central Role of Prolog in Logic Programming for CSP

Prolog (Programming in Logic) is a programming language based on first-order predicate logic and has become a pillar in the development of knowledge-based systems and AI. Prolog's declarative approach emphasizing what needs to be accomplished logic rather than how to achieve it is well suited for CSP modeling (Jendreiko, 2024).

a. Constraint Representation and Automatic Inference

EPIC (Journal of Electrical Power, Instrumentation and Control)
Teknik Elektro – Universitas Pamulang
Vol. 8, No. 2, Bulan Desember, Tahun 2025, Hal 302 – 312
http://openjournal.unpam.ac.id/index.php/jit
DOI: 10.32493/epic.v8i2.58721
pISSN 2615-0646
eISSN 2614-8595

Horn clauses and predicates are how Prolog describes the logical rules of puzzles. For example, the positional restriction "Spanish people have dogs" can be modeled using predicates that link nationality and pets in a data structure. To automatically explore the solution space, the Prolog inference engine uses a combination of resolution and backtracking (Berman et al., 2024; *ZebraPuzzles.Com*, n.d.). If a set of specific value assignments is found to violate a constraint, Prolog will backtrack and try a different solution path. The ability to handle backtracking automatically frees programmers from the burden of having to implement search algorithms.

b. Utilization of CLP(FD) in SWI-Prolog

Modern implementations of Prolog, particularly SWI-Prolog, offer highly sophisticated libraries, such as Constraint Logic Programming over Finite Domains, commonly abbreviated as CLP(FD). This library transforms CSP problems from simple backtracking searches into highly efficient constraint propagation problems. CLP(FD) works by running the domain of possible values immediately after a constraint is applied, well before the search process is performed, referring to the concept of CLP(FD) in SWI-Prolog. This approach drastically reduces computation time, making the Zebra Puzzle solution computationally reliable and efficient. Prolog's ability to integrate with mathematical and symbolic tools is also evident in other areas, such as Automatic Differentiation (AD) and Automated Theorem Proving (ATP), where the Prolog environment is well-suited for combining statistical and symbolic learning methods (Schrijvers et al., 2023; Zombori et al., 2020).

C. SWISH: Democratizing the Implementation of SWI-Prolog

Although Prolog is theoretically powerful, its development has become more accessible through SWISH. SWISH is a platform that also serves as the basis for advanced Semantic Web frameworks such as ClioPatria. The most significant improvement in terms of accessibility comes from the development of SWISH (SWI-Prolog for Sharing). SWISH is arguably an innovative web front-end for SWI-Prolog. This platform represents a natural evolution of the programming environment, shifting from traditional text-based or GUI interfaces to interactive and collaborative web-based interfaces, similar to Jupyter Notebooks or R-Studio (Wielemaker et al., 2021).

By using SWISH, there are many advantages to implementing Zebra Puzzle through SWISH, here are some advantages of implementing zebra puzzles using SWISH

- Global Reach: With the cloud-based nature of SWISH, there are no local installation barriers, allowing anyone to run Prolog programs through a browser. This also enables code and results sharing among multiple users.
- Interactive Prolog Explanation: SWISH allows the creation and sharing of documents that describe Zebra Puzzle Prolog solutions with text, run Prolog code, and display results immediately.
- Leveraging Results: Using contemporary web technology,

EPIC (Journal of Electrical Power, Instrumentation and Control)
Teknik Elektro – Universitas Pamulang
Vol. 8, No. 2, Bulan Desember, Tahun 2025, Hal 302 – 312
http://openjournal.unpam.ac.id/index.php/jit
DOI: 10.32493/epic.v8i2.58721
pISSN 2615-0646
eISSN 2614-8595

SWISH can format and display Prolog output. This is particularly helpful with the output of Zebra Puzzle solutions, which consist of a complete list of Prolog variables. It is much easier for readers and learners to understand and work with a coherent table of solutions than with a messy and unorganized list of Prolog variables.

In short, SWISH does more than just demonstrate the solution to the Zebra Puzzle with SWI-Prolog. It also redefines the level of complexity at which such problems can be taught and shared in academic and research systems.

RESEARCH METHOD

The research method used in this study is qualitative and questions for the zebra puzzle are as follows:

"**Consider the following logic puzzle: In five houses, each with a different color, live five persons of different nationalities, each of whom prefers a different brand of candy, a different drink, and a different pet. Given the following facts, the questions to answer are 'Where does the zebra live, and in which house do they drink water?**"

The Englishman lives in the red house. The Spaniard owns the dog.

The Norwegian lives in the first house on the left.

The green house is immediately to the right of the ivory house.

The man who eats Hershey bars lives in the house next to the man with the fox.

Kit Kats are eaten in the yellow house.

The Norwegian lives next to the blue house. The Smarties eater owns snails.

The Snickers eater drinks orange juice. The Ukrainian drinks tea.

The Japanese eats Milky Ways.

Kit Kats are eaten in a house next to the house where the horse is kept.

Coffee is drunk in the green house.

Milk is drunk in the middle house."

I will directly try to see if it is possible in the zebra puzzle to find out who owns the zebra and who drinks the water based on the questions and clues provided in the previous questions using swi-prolog. For make it easier to understand the steps, please refer to the Flowchart in Figure 1.

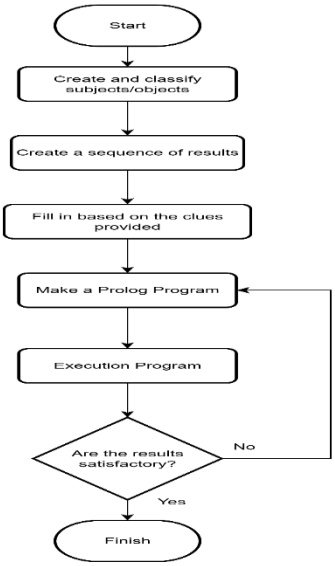

Figure 1 Flowchart Research

A. Create and Classifi Subject/Object

The first step in this research is to classify each subject/object, for example, what types of drinks, what types of pets, etc.

EPIC (Journal of Electrical Power, Instrumentation and Control)
Teknik Elektro – Universitas Pamulang
Vol. 8, No. 2, Bulan Desember, Tahun 2025, Hal 302 – 312
http://openjournal.unpam.ac.id/index.php/jit
DOI: 10.32493/epic.v8i2.58721
pISSN 2615-0646
eISSN 2614-8595

After classification, the results are as shown in Table 1 below.

Table 1 Table of Subject and Object Classification

House Order	1	2	3	4	5
Nationality	Japanese	Norwegian	Englishman	Ukranian	Spaniard
Pet	Snail	Horse	Fox	Dog	Zebra
Drink	Milk	Coffee	Orange Juice	Tea	Water
Candy	Kitkat	Hershey	Smarties	Snickers	Milky ways
House Colour	Yellow	Blue	Red	Ivory	Green

B. Create a Sequence Result

After completing the modeling stage of the Zebra Puzzle problem and dividing the variables into five categories: House Color, Nationality, Drink, Pet, and Snack, the next step is to design a logical and systematic structure. This logical and systematic structure becomes the blueprint for programming using Prolog. This structure is also essential for the implementation of the Constraint Satisfaction Problem, commonly abbreviated as CSP.

The grouping for each house characteristic to be represented is as follows:

House order: House color, Nationality, Drink, Pet, Snack

Justification for the Selection of Order. This grouping has a direct influence on the ease of modeling constraints in Prolog.

Consistency of Representation. Each house (positions 1 to 5) in Prolog will be represented in the form of a list or a single data structure. Therefore, if the grouping has the same and fixed order (for example, (Color, Nationality, Drink, PET, Candies), it will be easier for programmers to control each house property managed only through an index, which will certainly make it easier for programmers to maintain the code.

Mapping Constraints: Constraints in Zebra Puzzle example (“English people live in red houses”) will always combine two or more of these categories. With a specific order of houses, constraints can be modeled in a structured way. For example, to represent all houses, we will use a list of length five, where each five houses are represented by their own list, in a specific attribute order as above.

Application of CSP Principles: The regularity of the structure allows for better optimization to be performed on the more expensive parts of the CSP solving algorithm. This is especially true when we use libraries such as CLP(FD) in SWI-Prolog. All categories (Color, Nationality, etc.) act as value Domains to be searched. The regularity in the structure order ensures that each domain variable (e.g., “Color of the house in position 3”) is correctly and positively placed in the correct order in the solution data structure, allowing for easier constraint propagation and more efficient labeling by Prolog.

C. Fill in Based the Clue

After that, of course, we need to make a list and include the clues given in the question. There are a total of 11 clues that we can use as a reference in determining the answer.

EPIC (Journal of Electrical Power, Instrumentation and Control)
Teknik Elektro – Universitas Pamulang
Vol. 8, No. 2, Bulan Desember, Tahun 2025, Hal 302 – 312
http://openjournal.unpam.ac.id/index.php/jit
DOI: 10.32493/epic.v8i2.58721
pISSN 2615-0646
eISSN 2614-8595

```
%petunjuk dari soal
%The Englishman lives in the red house
member(isi(_, red, englishman, _, _, _), Penghuni),
%The Spaniard owns the dog
member(isi(_, _, spaniard, _, dog, _), Penghuni),
%The Norwegian lives in the first house
member(isi(1, _, norwegian, _, _, _), Penghuni),
%Green is right of ivory
right_of(isi(_, green, _, _, _, _), isi(_, ivory, _, _, _, _), Penghuni),
% Hershey next to fox
next_to(isi(_, _, _, _, hershey), isi(_, _, _, _, fox, _), Penghuni),
% Kit Kats in yellow house
member(isi(_, yellow, _, _, _, kitkat), Penghuni),
% Norwegian next to blue
next_to(isi(_, _, norwegian, _, _, _), isi(_, blue, _, _, _, _), Penghuni),
% Smarties eater owns snails
member(isi(_, _, _, _, snail, smarties), Penghuni),
% Snickers -> orange juice
member(isi(_, _, _, orangejuice, _, snickers), Penghuni),
% Ukrainian drinks tea
member(isi(_, _, ukrainian, tea, _, _), Penghuni),
% Japanese eats Milky Ways
member(isi(_, _, japanese, _, _, milkyways), Penghuni),
% KitKat next to horse
next_to(isi(_, _, _, _, horse, _), isi(_, _, _, _, _, kitkat), Penghuni),
% Coffee in green house
member(isi(_, green, _, coffee, _, _), Penghuni),
% Milk in middle house
member(isi(3, _, _, milk, _, _), Penghuni),                    % Milk in middle house
```

Figure 2 Insert Clue in Prolog

D. Make a Prolog Program

After entering the clues provided, the next step is to create a program with the aim of answering the questions based on those clues, namely who owns the zebra and who drink the water. In this case, I created a program by defining the question as a question with a definite answer, so that Prolog would only provide one definite answer, not multiple answers with several possibilities. For easier understanding of the details, please refer to Figure 3.

```
solve :-
Penghuni = [
    isi(1, _, _, _, _, _),
    isi(2, _, _, _, _, _),
    isi(3, _, _, _, _, _),
    isi(4, _, _, _, _, _),
    isi(5, _, _, _, _, _)
],
```

Figure 3 Question Form in Prolog Programs

E. Execution Program

At this stage, I will execute the program that has been created and also create a program to display the results of this program. In the program that I created here, I made it complete so that the results will describe in detail all homeowners, candies, pets, and drinks.

```
% ---------- result ----------
nl, write('===== RESULT ====='), nl,
print_penghuni(Penghuni),
nl,
member(isi(_, _, Who_Have_Zebra, _, zebra, _), Penghuni),
format('~w is person who have zebra.~n', [Who_Have_Zebra]),
member(isi(_, _, Who_Drink_Water, water, _, _), Penghuni),
format('~w is a person who drinking a water.~n', [Who_Drink_Water]).
```

Figure 4 Program for Execution

RESULT AND DISCUSSION

The result of this research is that when we run the code ?-solve. on SWISH, SWISH will immediately provide an answer. This can be done because, as shown in Figure 3. I put the word solve:- which means the command to answer questions that have been declared in the solve:- syntax. So after we run ?-solve. It will show like figure 5.

```
===== RESULT =====
House Number 1: The Colour is yellow, Nationallity norwegian, Drink _10682, Pet fox, Eat kitkat
House Number 2: The Colour is blue, Nationallity ukrainian, Drink tea, Pet horse, Eat hershey
House Number 3: The Colour is red, Nationallity englishman, Drink milk, Pet snail, Eat smarties
House Number 4: The Colour is ivory, Nationallity spaniard, Drink orangejuice, Pet dog, Eat snickers
House Number 5: The Colour is green, Nationallity japanese, Drink coffee, Pet _10764, Eat milkyways

japanese is person who have zebra.
norwegian is a person who drinking a water.
```

Figure 5 Result

When you look closer, there are two columns with no answers house number 1 has no drink, and house number 5 has no pet. Based on this, the answer is that the Norwegian drinks water, and the Japanese keeps a zebra as a pet.Then the reason why the group shape can appear is that there are additional programs as shown in figure 6

```
% Print result nicely
print_penghuni([]).
print_penghuni([isi(No, Warna, Suku, Minuman, Hewan, Permen)|Rest]) :-
    format('House Number ~w: The Colour is ~w, Nationallity ~w, Drink ~w, Pet ~w, Eat ~w~n',
           [No, Warna, Suku, Minuman, Hewan, Permen]),
  print_penghuni(Rest).
```

Figure 6 Program For Print Result

Based on that program the print_penghuni function acts like a storyteller who has an extensive list of residents in the

EPIC (Journal of Electrical Power, Instrumentation and Control)
Teknik Elektro – Universitas Pamulang
Vol. 8, No. 2, Bulan Desember, Tahun 2025, Hal 302 – 312
http://openjournal.unpam.ac.id/index.php/jit
DOI: 10.32493/epic.v8i2.58721
pISSN 2615-0646
eISSN 2614-8595

house. They take out the list and check it carefully from the first house to the last. After finding a house, they read all the details contained within it, such as the house number and color, the owner's nationality, favorite drink, pets, candy, and so on. They organize their words and convey the information to the screen as if presenting the residents of the house. Instead of stopping after presenting a house, they continue to the next entry and repeat the same process. This narrative continues until all the houses on the list have been fully described. Once the list of residents has been fully described, the storyteller wisely knows to stop on its own and finish. Therefore, print_penghuni functions as an automatic storyteller that converts data in Prolog structure into a flowing narrative.

Then the reason why it says that Japanese have zebra and Norwegians drink water is based on the program in image 4, where you can see that I entered the question in the third subject in the order where it is the place of nationality, so when I changed the layout, for example to the fifth order, the answer no longer displayed Japanese and Norwegian but KitKat and Milky Way as shown in images 7 and 8.

```
% ---------- result ----------
nl, write('===== RESULT ====='), nl,
print_penghuni(Penghuni),
nl,
member(isi(_, _, _, _, zebra, Who_Have_Zebra), Penghuni),
format('~w is person who have zebra.~n', [Who_Have_Zebra]),
member(isi(_, _, _r, water, _, Who_Drink_Water), Penghuni),
format('~w is a person who drinking a water.~n', [Who_Drink_Water]).
```

Figure 7 Program to Make Candies the Subject of Answers

```
===== RESULT =====
House Number 1: The Colour is yellow, Nationallity norwegian, Drink _17242, Pet fox, Eat kitkat
House Number 2: The Colour is blue, Nationallity ukrainian, Drink tea, Pet horse, Eat hershey
House Number 3: The Colour is red, Nationallity englishman, Drink milk, Pet snail, Eat smarties
House Number 4: The Colour is ivory, Nationallity spaniard, Drink orangejuice, Pet dog, Eat snickers
House Number 5: The Colour is green, Nationallity japanese, Drink coffee, Pet _17324, Eat milkyways

milkyways is person who have zebra.
kitkat is a person who drinking a water.
```

Figure 8 Results Where Candies Become Objects

The reason why the zebra puzzle-solving experiment was successful is that when executing the “solve” predicate, Prolog's way of thinking is similar to that of an investigator who records the initial case with many suspects and interconnected clues, which in this case are five houses standing side by side. At this point, the five houses are like a blank canvas. The houses are numbered, but their attributes color, nationality, drink, pet, and candies are left blank. All values are unspecified and can be anything as long as they fit within the existing constraints.

Once this basic structure is in place, Prolog begins scanning the clues in the handout one by one. The handout contains clues that serve as constraints. For example, Prolog, having been told that “the Englishman lives in the red house,” must ensure that in one of the five houses, the nationality variable is set to Englishman, and the house color variable in the same position is red. Other clues such as “Spaniards have dogs” or “Ukrainian drink tea” limit the assignment of variables to other houses. Little by little, more and more variables are bound to specific values.

Prolog does not fill houses randomly. It uses unification and backtracking. Unification ensures that facts such as the relationship between color and nationality must be consistent. If at any point Prolog fills a house with a combination that is inconsistent with one of the clues, it does not stop or fail. Instead, it backtracks, goes back one step, and tries something else. This process is similar to someone trying to solve a logic puzzle, eliminating incorrect options and trying new combinations until all clues are satisfied.

Clues related to placement, such as “green is to the right of ivory” and “the Norwegian lives next to the blue house,” mean that Prolog must handle not only

EPIC (Journal of Electrical Power, Instrumentation and Control)
Teknik Elektro – Universitas Pamulang
Vol. 8, No. 2, Bulan Desember, Tahun 2025, Hal 302 – 312
http://openjournal.unpam.ac.id/index.php/jit
DOI: 10.32493/epic.v8i2.58721
pISSN 2615-0646
eISSN 2614-8595

values but also the relationships between houses. Prolog verifies the structure of the house list and ensures that specific houses are in the correct positions. It also backtracks if the combination it tries does not fit.

The program operates in a completely systematic manner. Each clue narrows down the possibilities until Prolog is left with only one combination of values that meets all the requirements. When all conditions are met, the Residents list is filled with consistent values. This is the result of the print_occupants predicate, where the user can see the complete layout of the houses, colors, nationalities, drinks, pets, and candies.

In this step of the program, two pieces of information are extracted from the structure: the names of the zebra owners and the names of those who drink water. This is done by searching for houses with the attribute of owning a zebra and houses with the attribute of drinking water. Since all variables are filled in, Prolog can provide an answer immediately and without ambiguity.

Overall, this program functions like a machine for logical reasoning. Instead of providing an algorithm with a set of step-by-step instructions, it is given a set of facts and relationships that explain those facts. Using matching, reasoning, and clear elimination, Prolog successfully concludes a single configuration. The beauty of Prolog, of course, lies in its ability to deduce answers from logical descriptions alone, without showing any procedural instructions. Thus, solving the Zebra Puzzle is more than just code execution; it represents an example of automatic reasoning with a set of interconnected logical rules and constraints.

CONCLUSION

The Prolog program used to solve the Zebra Puzzle operates by utilizing other Prolog logical thinking strategies, unification and backtracking. When the program is running in Prolog, it will first mark the five empty houses, then fill in and adjust the attributes of each house, modified according to the instructions. Each instruction functions as a constraint that must be met. Prolog will then gradually eliminate any inconsistent combinations, leaving only the appropriate combinations.

This solution search process is applied gradually, with Prolog trying various combinations using backtracking until it finds the right one. However, there is a slight error in the search pattern that affects the final part, meaning that Prolog can only display the answer up to three times. In addition, errors will occur when Prolog searches the results in the next search path. Therefore, Prolog can solve this puzzle using declarative logic, patterns, and rule structures that must be very subjective to the author. Prolog will be able to complete its solution search optimally when all clues and thought patterns are arranged in such a way that they are consistent and error-free.

REFERENCES

Berman, S., McKeown, K., & Ray, B. (2024). *Solving Zebra Puzzles Using Constraint-Guided Multi-Agent Systems* (arXiv:2407.03956). arXiv. https://doi.org/10.48550/arXiv.2407.03956

Buscaroli, R., Chesani, F., Giuliani, G., Loreti, D., & Mello, P. (2023). A Prolog application for reasoning on maths puzzles with diagrams. *Journal of Experimental & Theoretical Artificial Intelligence*, *35*(7), 1079–1099. https://doi.org/10.1080/0952813X.2022.2062456

Grohe, M., Guruswami, V., & Zivny, S. (2018). *The Constraint Satisfaction Problem: Complexity and Approximability (Dagstuhl Seminar*

EPIC (Journal of Electrical Power, Instrumentation and Control)
Teknik Elektro – Universitas Pamulang
Vol. 8, No. 2, Bulan Desember, Tahun 2025, Hal 302 – 312
http://openjournal.unpam.ac.id/index.php/jit
DOI: 10.32493/epic.v8i2.58721
pISSN 2615-0646
eISSN 2614-8595

18231) [Application/pdf]. 18 pages. https://doi.org/10.4230/DAGREP.8.6.1

Jamine, D. M. (2024). *LOGIC PUZZLES: UPAYA MENINGKATKAN COMPUTATIONAL THINKING SISWA SMK MENGGUNAKAN MODEL PUZZLE-BASED LEARNING PADA MATERI ALGORITMA*.

Jendreiko, C. (2024). *Generative Logic: Teaching Prolog as Generative AI in Art and Design*.

Körner, P., Leuschel, M., Barbosa, J., Costa, V. S., Dahl, V., Hermenegildo, M. V., Morales, J. F., Wielemaker, J., Diaz, D., Abreu, S., & Ciatto, G. (2022). Fifty Years of Prolog and Beyond. *Theory and Practice of Logic Programming*, *22*(6), 776–858. https://doi.org/10.1017/S1471068422000102

Krokhin, A., & Opršal, J. (2022). An invitation to the promise constraint satisfaction problem. *ACM SIGLOG News*, *9*(3), 30–59. https://doi.org/10.1145/3559736.3559740

Schrijvers, T., Berg, B. van den, & Riguzzi, F. (2023). *Automatic Differentiation in Prolog* (arXiv:2305.07878). arXiv. https://doi.org/10.48550/arXiv.2305.07878

Shah, K., Dikkala, N., Wang, X., & Panigrahy, R. (n.d.). *Causal language modeling can elicit search and reasoning capabilities on logic puzzles*.

Siti Masrichah. (2023). Ancaman Dan Peluang Artificial Intelligence (AI). *Khatulistiwa: Jurnal Pendidikan dan Sosial Humaniora*, *3*(3), 83–101. https://doi.org/10.55606/khatulistiwa.v3i3.1860

SWISH -- prolog_tutorials.swinb. (n.d.). Retrieved November 18, 2025, from https://swish.swi-prolog.org/example/prolog_tutorials.swinb

Szomiu, R., & Groza, A. (2021). *A Puzzle-Based Dataset for Natural Language Inference* (arXiv:2112.05742). arXiv. https://doi.org/10.48550/arXiv.2112.05742

Thompson, S., Candon, K., & Vázquez, M. (2025). The Social Context of Human–Robot Interactions. *Annual Review of Control, Robotics, and Autonomous Systems*. https://doi.org/10.1146/annurev-control-030623-015506

White, C., Dooley, S., Roberts, M., Pal, A., Feuer, B., Jain, S., Shwartz-Ziv, R., Jain, N., Saifullah, K., Naidu, S., Hegde, C., LeCun, Y., Goldstein, T., Neiswanger, W., Goldblum, M., & Ai, A. (2024). *LiveBench: A Challenging, Contamination-Free LLM Benchmark*.

Wielemaker, J., Beek, W., Hildebrand, M., & van Ossenbruggen, J. (2021). *ClioPatria: A SWI-Prolog Infrastructure for the Semantic Web*.

Wielemaker, J., Lager, T., & Riguzzi, F. (2015). *SWISH: SWI-Prolog for Sharing* (arXiv:1511.00915). arXiv. https://doi.org/10.48550/arXiv.1511.00915

Wielemaker, J., Riguzzi, F., Kowalski, B., Lager, T., Sadri, F., & Calejo, M. (2020). *Using SWISH to realise interactive web based tutorials for logic based languages*. https://doi.org/10.1017/S1471068418000522

Yang, X., Chen, B., & Tam, Y.-C. (2024). *Arithmetic Reasoning with LLM: Prolog Generation & Permutation* (arXiv:2405.17893). arXiv. https://doi.org/10.48550/arXiv.2405.17893

ZebraPuzzles.com: New Zebra Puzzles Every Day. (n.d.). Retrieved November 12, 2025, from https://www.zebrapuzzles.com/

EPIC (Journal of Electrical Power, Instrumentation and Control)
Teknik Elektro – Universitas Pamulang
Vol. 8, No. 2, Bulan Desember, Tahun 2025, Hal 302 – 312
http://openjournal.unpam.ac.id/index.php/jit
DOI: 10.32493/epic.v8i2.58721
pISSN 2615-0646
eISSN 2614-8595

Zombori, Z., Urban, J., & Brown, C. E. (2020). Prolog Technology Reinforcement Learning Prover: (System Description). In N. Peltier & V. Sofronie-Stokkermans (Eds.), *Automated Reasoning* (Vol. 12167, pp. 489–507). Springer International Publishing. https://doi.org/10.1007/978-3-030-51054-1_33