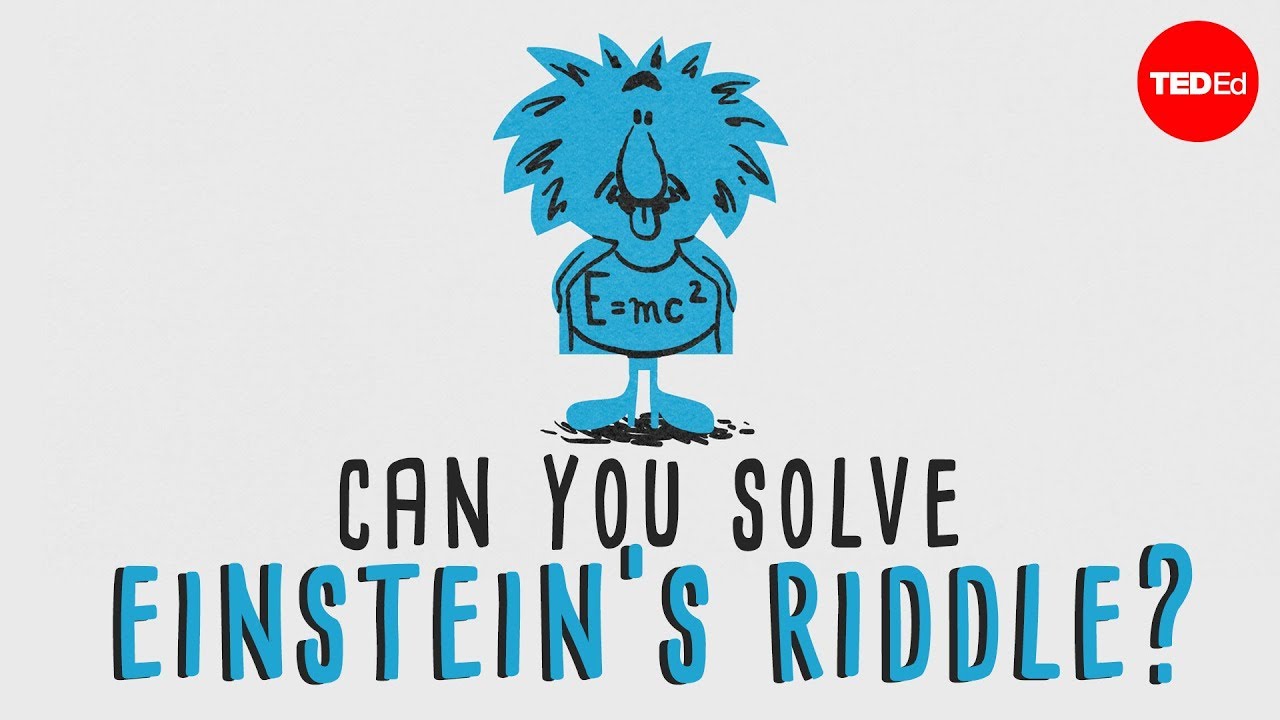
**Solving logic puzzles with Prolog: A case study of Einstein’s Riddle**

*Șorecău Adrian-Vasile, 927*

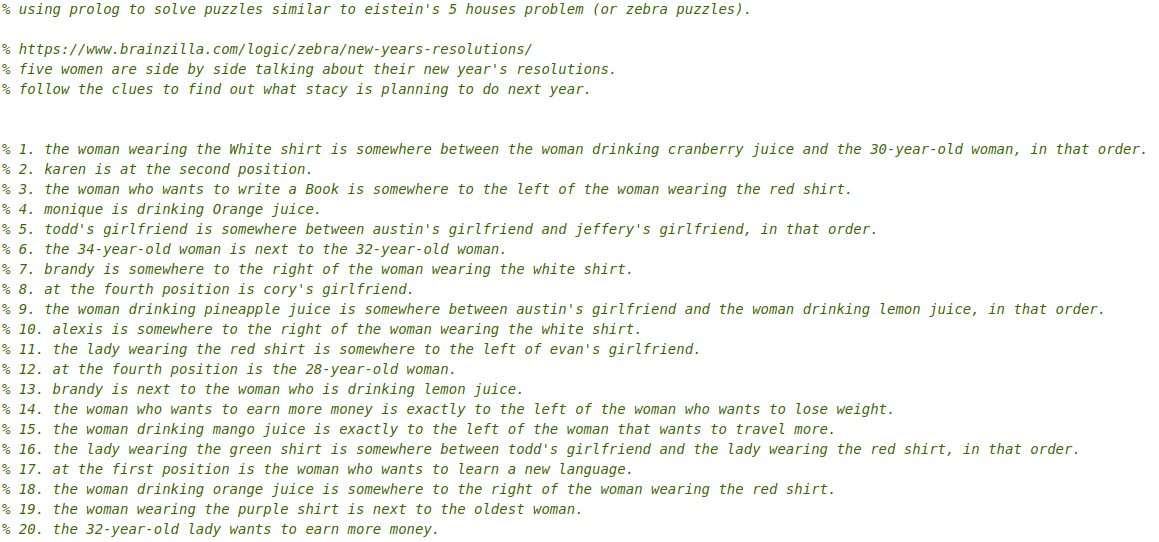


1. **Introduction**

**Logical programming languages**, exemplified by **Prolog**, offer a powerful paradigm for problem-solving by *expressing rules and relationships declaratively*. Prolog, with its *goal-driven execution model*, excels in tackling complex problems with intricate logical structures. One compelling application of this approach is evident in the realm of logic puzzles, where systematic deduction is crucial.

**Einstein's Riddle**, often referred to as the **Zebra Puzzle**, serves as an intriguing challenge requiring individuals to deduce relationships among a group of entities based on provided clues and constraints. As a logical problem, the Zebra Puzzle becomes an ideal candidate for showcasing the capabilities of Prolog. In this project, we explore how Prolog's unique features, including its syntax and backtracking mechanism, enable an elegant representation and efficient resolution of the intricate logical relationships posed by Einstein's Riddle. This investigation aims to highlight the effectiveness of logical programming languages in solving real-world problems with nuanced logical structures.

1. **Einstein’s Riddle: An Overview**

**** **Einstein's Riddle** is a logic puzzle that revolves around a scenario with a group of distinct entities, often people, each possessing unique attributes. The puzzle typically provides a set of clues or statements offering partial information about the entities' characteristics, such as their professions, nationalities, pet preferences, and the locations of their residences. The objective is to deduce the precise attributes of each entity and their respective placements within the group based on the given clues. The challenge lies in systematically applying deductive reasoning to unravel the complex web of relationships, ultimately arriving at a comprehensive and coherent solution that satisfies all the constraints provided in the puzzle.

1. **Logic Programming and Prolog**
   1. **Logic Programming**

**Logic programming** is a computational paradigm that focuses on representing and solving problems through the formulation of logical rules and relationships. Unlike traditional imperative programming, logic programming languages, such as Prolog, enable users to express the desired outcome declaratively, allowing the system to handle the logical reasoning necessary to achieve the specified goal. This approach is particularly effective for solving problems with intricate logical structures, making logic programming a valuable tool in various domains.

* 1. **Prolog: A brief introduction**

**Prolog**, short for "Programming in Logic," is a prominent logic programming language known for its simplicity and expressiveness in dealing with complex logical relationships. In Prolog, programs are constructed using facts and rules that define relationships and constraints. The language's distinctive feature is its goal-driven execution model, where the programmer specifies the desired outcome, and Prolog autonomously searches for a solution by satisfying the given logical conditions. Prolog's elegant syntax and efficient backtracking mechanism make it well-suited for tasks involving complex logical reasoning, such as solving logic puzzles like Einstein's Riddle.

1. **Translating Einstein’s Riddle into Prolog**

To solve puzzles similar to Einstein's 5 houses problem, exemplified by the **New Year's Resolutions** scenario, we leverage the power of Prolog to represent and deduce relationships systematically. The puzzle features five women discussing their resolutions, each with distinctive attributes such as shirt color, name, resolution, boyfriend, age, and preferred juice. The Prolog code encapsulates these entities and their relationships, effectively translating the puzzle into a set of logical rules and constraints.

****

In the Prolog code, predicates like **somewhereToTheLeft**, **somewhereToTheRight**, **exactlyToTheLeft**, **nextTo**, and **somewhereBetween** encode the spatial relationships and constraints outlined in the puzzle's clues.



The **women** predicate defines the characteristics of each woman, and the **stacys\_resolution** predicate is the query to find out Stacy's resolution. By executing the program, Prolog uses its goal-driven execution and backtracking mechanism to explore possible configurations of the entities until a solution that satisfies all constraints is found.

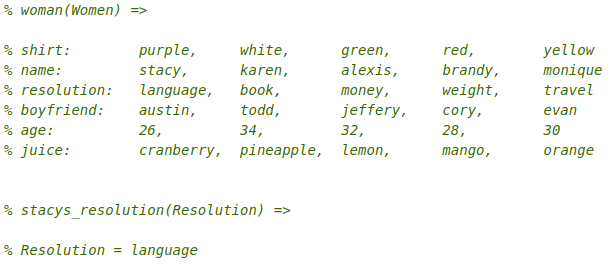
The defined relationships in the Prolog code mirror the logical deductions required to crack Einstein's Riddle, showcasing Prolog's effectiveness in solving complex logic puzzles with precision. The results of executing the program reveal Stacy's resolution, demonstrating the application of logical programming to unravel intricate scenarios posed by such puzzles.

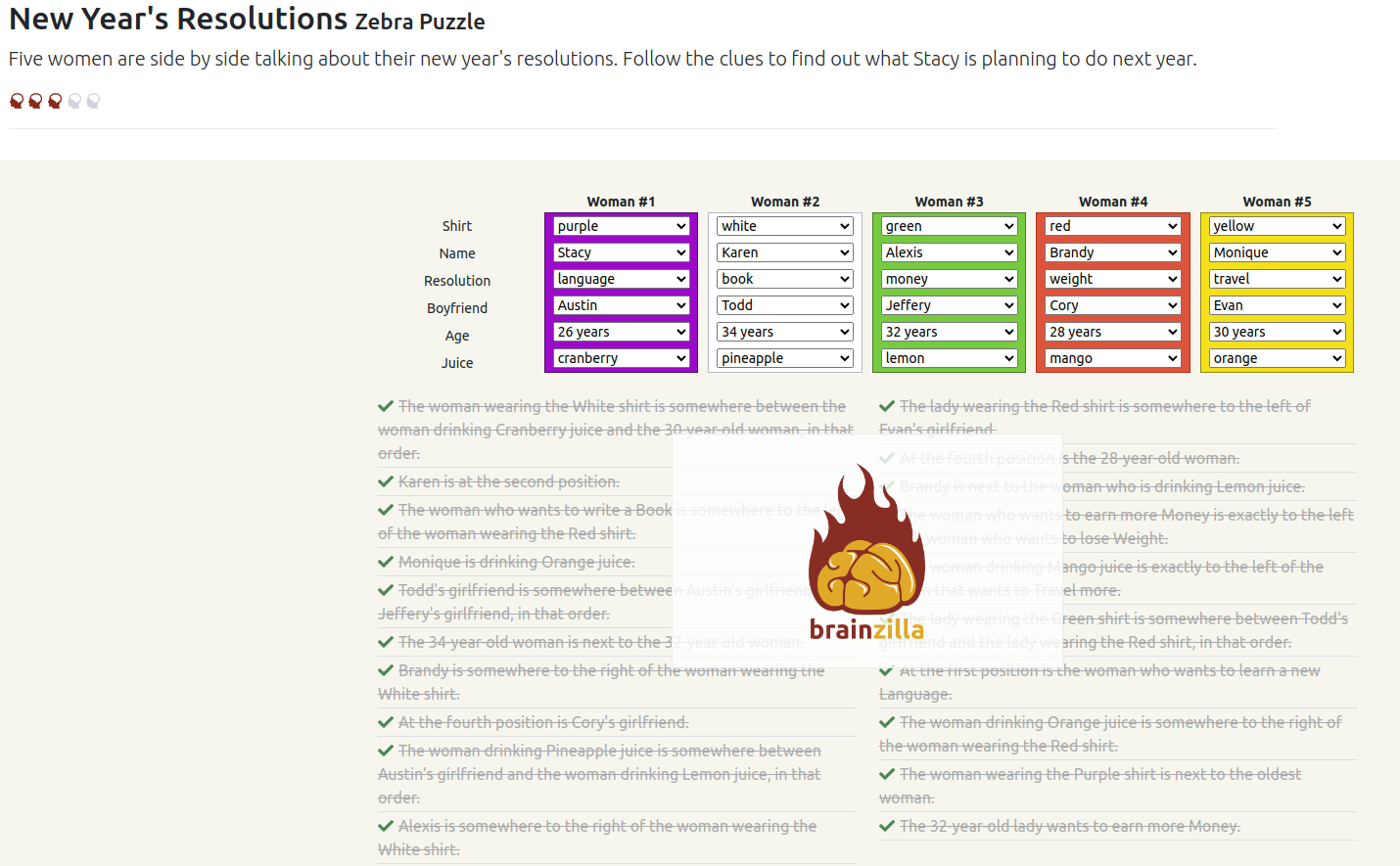
1. **Backtracking**

Prolog's inherent backtracking mechanism is a pivotal feature in navigating the complex solution space. When the initial path of deduction encounters a contradiction or fails to satisfy a particular constraint, Prolog seamlessly backtracks, exploring alternative paths to reach a valid solution. This iterative process continues until a consistent solution is found or all possibilities are exhausted. The synergy of goal-driven execution and backtracking in Prolog enables efficient exploration of the intricate relationships embedded in puzzles like the New Year's Resolutions scenario, showcasing the language's effectiveness in solving complex logical problems.

1. **Conclusion**

In conclusion, the application of Prolog to solve logic puzzles, exemplified by the New Year's Resolutions scenario, underscores the language's effectiveness in handling complex logical relationships. By translating the intricate rules and clues of the puzzle into Prolog code, we showcased the power of logic programming in representing and solving real-world problems. The goal-driven execution model and backtracking mechanism of Prolog allowed for a systematic exploration of possible solutions, ultimately revealing Stacy's resolution and highlighting the elegance of Prolog in tackling puzzles similar to Einstein's Riddle. This project provides a glimpse into the potential of logical programming languages, specifically Prolog, as valuable tools for unraveling intricate scenarios through precise and efficient deduction.





1. **References**

*Can you solve Einstein’s Riddle? - Dan Van der Vieren*, **YouTube**, uploaded by **TED-Ed**, 30.11.2015,

[Can you solve "Einstein’s Riddle"? - Dan Van der Vieren](https://www.youtube.com/watch?v=1rDVz_Fb6HQ&ab_channel=TED-Ed)

*Einstein or Zebra Problem*, **YouTube**, uploaded by **Max Sullivan**, 09.12.2013,

[Einstein or Zebra Problem](https://www.youtube.com/watch?v=zUnnvt9IpoA&t=105s&ab_channel=MaxSullivan)

**Kealan Parr**, 2021, *How to solve Einstein’s Five House Riddle*, **freeCodeCamp**, 8 September

*An Introduction to Prolog Programming*, **KATEDRA INFORMATIKY A VÝPOČETNÍ TECHNIKY**,

<https://www.kiv.zcu.cz/studies/predmety/uzi/Folie_Prolog/Intro_to_Prolog_prog.pdf>