Department of Computer Science
Technical University of Cluj-Napoca

Artificial Intelligence

Name: Flanja Tudor Calin
Email: tudorflanja@gmail.com

Teaching Assistant: Adrian Groza
Adrian.Groza@cs.utcluj.ro

Contents

Chapter 1

Logics

1.1 Introduction

Welcome to the world of First-Order Logic (FOL) logic puzzles! These puzzles are fascinating challenges that require a keen understanding of logical reasoning and deduction. FOL, a fundamental component of mathematical logic, serves as the backbone for these brain-teasers, offering an engaging and thought-provoking way to exercise cognitive abilities.

FOL puzzles present scenarios where you, as the solver, must apply logical rules and deductions to unravel intricate relationships, constraints, and patterns. They are designed to test your ability to make precise and systematic inferences based on given information, ultimately arriving at a logically consistent solution.

In this documentation, we dive into the fundamentals of FOL logic puzzles, providing you with a comprehensive understanding of their structure, rules, and solving strategies, applied in the next 4 puzzles.

The command to run the "Placings" puzzle is "prover9 -f placings" and for the rest of them is "mace4 -f nameOfFile". The first puzzle is run with Prover9 because it generates a large model and Mace4 does not have the "power" to perform the task and the model is generated in the "placings.out" file.

1.2 Logic puzzles

1.2.1 Placings

Task

List the order in which each person finished.

1. Tommy Tombstone finished after Lance Lamers and Brett Brown but before Mitch Monday.

2. Peter Poultry finished before Daniel Dusk and Lance Lamers.

3. Sam Sunny finished after Peter Poultry and before Jack Jill and Harry Hills.

4. Keri Kernel finished after Peter Poultry, Mitch Monday and Tommy Tombstone.

5. Lance Lamers finished after Brett Brown and Daniel Dusk, but before Jack Jill and Mitch Monday.
6. Mitch Monday finished after Sam Sunny and Brett Brown.
7. Brett Brown finished before Jack Jill, Mitch Monday and Peter Poultry.
8. Daniel Dusk finished before Keri Kernel and Tommy Tombstone, but after Sam Sunny.
9. Jack Jill finished before Keri Kernel, Tommy Tombstone and Mitch Monday, but after Peter Poultry and Daniel Dusk.
10. Harry Hills finished before Mitch Monday but after Lance Lamers, Jack Jill and Tommy Tombstone.

Solving

The code for this task is provided in the Appendix part of the project. This section of the documentation will cover only the explanations.

Prerequisites

The code provided represents a logical puzzle formulated in First-Order Logic (FOL) to determine the order in which a group of individuals finished a race. The task involves deducing the order of completion based on a set of clues and constraints provided.

Clues and Constraints

1. **Differentiation**: The different predicate establishes that each person is distinct from the others. This ensures uniqueness in the order of finish.
2. **Succession**: The next predicate denotes the sequential relationship between individuals. It defines the order in which one person finished immediately before another.
3. **Individual Assignment**: Each person is assigned to one and only one position in the finishing order. This is represented by predicates like TommyTombstone(x), LanceLamers(x), etc.
4. **Task Rules**: These rules express the relationships and order constraints derived from the given statements about the finishing positions of the individuals.

Solution

The provided FOL logic translates the verbal clues into logical statements, linking the individuals' finishing positions:

- The task rules (labeled 1 to 10) represent the deductions derived from the verbal statements about the order in which people finished relative to each other.
- Succession: The next predicate denotes the sequential relationship between individuals. It defines the order in which one person finished immediately before another.
- Individual Assignment: Each person is assigned to one and only one position in the finishing order. This is represented by predicates like TommyTombstone(x), LanceLamers(x), etc.

- Task Rules: These rules express the relationships and order constraints derived from the given statements about the finishing positions of the individuals.

1.2.2 First and Last Names

Task

The first names of five people are Beverly, Charles, Monica, Nelson, and Ruth. Their last names are Atwood, Porter, Stafford, Thompson, and Ward. Match each person first and last names.

1. Ruth and Porter went hiking yesterday with Monica and Stafford.
2. Beverly enrolled her son, and Atwood enrolled her daughter, at the same nursery school.
3. Thompson and Ruth don't have children.
4. Charles, who has been in the hospital for the past three days, was visited today by Porter and Nelson.

Solving

The code for this task is provided in the Appendix part of the project. This section of the documentation will cover only the explanations.

Prerequisites

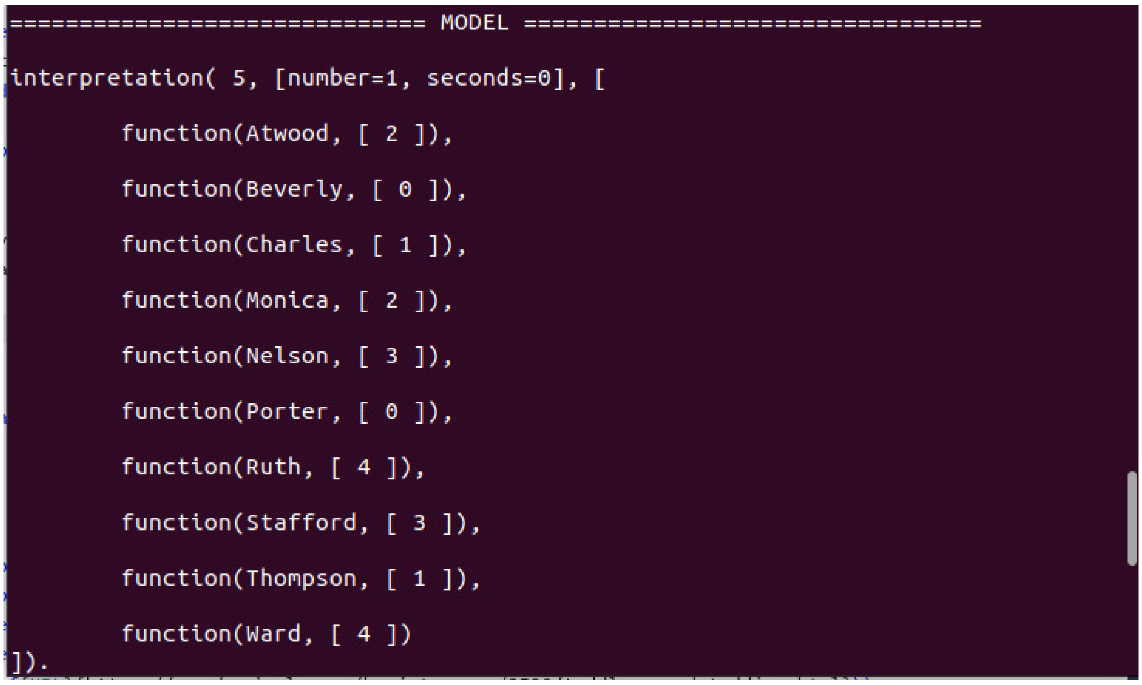
The code represents a logical puzzle where the objective is to match the first and last names of five people: Beverly, Charles, Monica, Nelson, and Ruth. Each person is associated with one first name and one last name from the set of possible options: Atwood, Porter, Stafford, Thompson, and Ward.

Clues and Constraints

1. **Hiking Companions**: Monica and Ruth went hiking yesterday. Deduction: Monica and Ruth did not have the last names Porter or Stafford due to their hiking companionship.
2. **Enrollment at Nursery School**: Beverly and Atwood enrolled their children in the same nursery school. Deduction: Beverly's last name cannot be Atwood.
3. **Children Status**: Thompson and Ruth do not have children. Deduction: Eliminates certain last name possibilities for Thompson and Ruth.
4. **Hospital Visit**: Charles, who has been hospitalized, was visited by Porter and Nelson. Deduction: Charles's last name cannot be Porter, and Nelson's last name also cannot be Porter or Stafford.
5. **Gender Differentiation**: Charles and Nelson are males; the others are female. Deduction: Certain last names are eliminated for Charles and Nelson based on the gender differentiation.

Solution

The FOL logic rules eliminate potential combinations of first and last names based on the clues provided. These rules are used to create constraints that determine which combinations are valid or invalid according to the given information.



```
============================== MODEL =================================

interpretation( 5, [number=1, seconds=0], [

        function(Atwood, [ 2 ]),

        function(Beverly, [ 0 ]),

        function(Charles, [ 1 ]),

        function(Monica, [ 2 ]),

        function(Nelson, [ 3 ]),

        function(Porter, [ 0 ]),

        function(Ruth, [ 4 ]),

        function(Stafford, [ 3 ]),

        function(Thompson, [ 1 ]),

        function(Ward, [ 4 ])
]).
```

Figure 1.1: Mace4 demonstration for the problem

1.2.3 Toddlers and Teddies

Task

At the local play group for babies and toddlers, I was asking the mothers about the number of teddies each of their children has.

1. The four children are aged 1, 2, 3 and 4.
2. Remarkably, the children have one, two, three and four teddies, although not necessarily respectively.
3. James has more teddies than his age.
4. John is older than Matthew.
5. Curiously only one child has the same number of teddies as his age.
6. Paul has less teddies than John, and the child aged 3 has two teddies.
7. Paul is the youngest.

How old is each child and how many teddies do they have each?

Solving

The code for this task is provided in the Appendix part of the project. This section of the documentation will cover only the explanations.

Prerequisites

The code encapsulates a logical puzzle concerning the number of teddies each child, aged 1, 2, 3, and 4, has at a local playgroup. The children's names are James, John, Matthew, and Paul.

Clues and Constraints

1. **Teddies and Ages**: Each child has a unique number of teddies, ranging from 1 to 4, but not necessarily in ascending order. James has more teddies than his age, and only one child has the same number of teddies as their age.
2. **Age Relations**: John is older than Matthew. The child aged 3 has two teddies, and Paul is the youngest.

Solution

The deductions made based on the clues are as follows:

1. Paul is the youngest, so he is 1 year old.
2. James has more teddies than his age, indicating he's either 2 or 3 years old.
3. John being older than Matthew, and Paul being 1 year old, limits Matthew to 2 or 3 years old, and John to 3 or 4 years old.
4. The child aged 3 (Matthew) has two teddies, excluding James from being 3 years old. James is 2 years old.
5. John, being older than Matthew, becomes 4 years old, leaving Matthew as 3 years old.

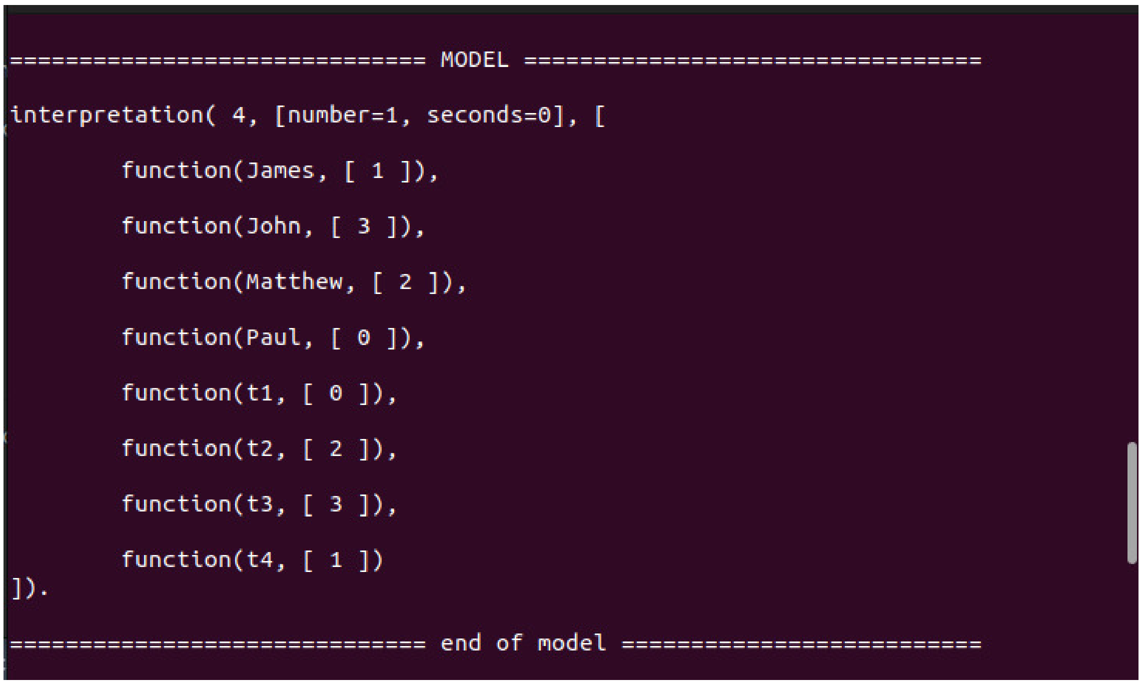
Using these deductions, the code assigns each child's age and the number of teddies they have accordingly: Paul is 1 year old with 1 teddy, James is 2 years old with 4 teddies, Matthew is 3 years old with 2 teddies, and John is 4 years old with 3 teddies. This solution satisfies all the given clues and constraints.

1.2.4 Books, books and more books

Task

Jake, John, Joe, and Jack each have a different favorite book series. These include "Harry Potter", "The Lord of the Rings", "Sherlock Holmes", and "The Hardy Boys". Using the clues, figure out who likes which book series.

1. Jake and Joe love mysteries, while John and Jack prefer magic and fantasy.
2. "Joe" is the name of a character in Joe's favorite series.
3. John's Halloween costume, based on his favorite series, includes glasses and a black robe.



```
============================== MODEL =================================

interpretation( 4, [number=1, seconds=0], [

        function(James, [ 1 ]),

        function(John, [ 3 ]),

        function(Matthew, [ 2 ]),

        function(Paul, [ 0 ]),

        function(t1, [ 0 ]),

        function(t2, [ 2 ]),

        function(t3, [ 3 ]),

        function(t4, [ 1 ])
]).

============================== end of model ==========================
```

Figure 1.2: Mace4 demonstration for the problem

Solving

The code for this task is provided in the Appendix part of the project. This section of the documentation will cover only the explanations.

Prerequisites

The code presents a logic puzzle where four individuals, Jake, John, Joe, and Jack, have distinct favorite book series: "Harry Potter," "The Lord of the Rings," "Sherlock Holmes," and "The Hardy Boys."

Clues and Constraints

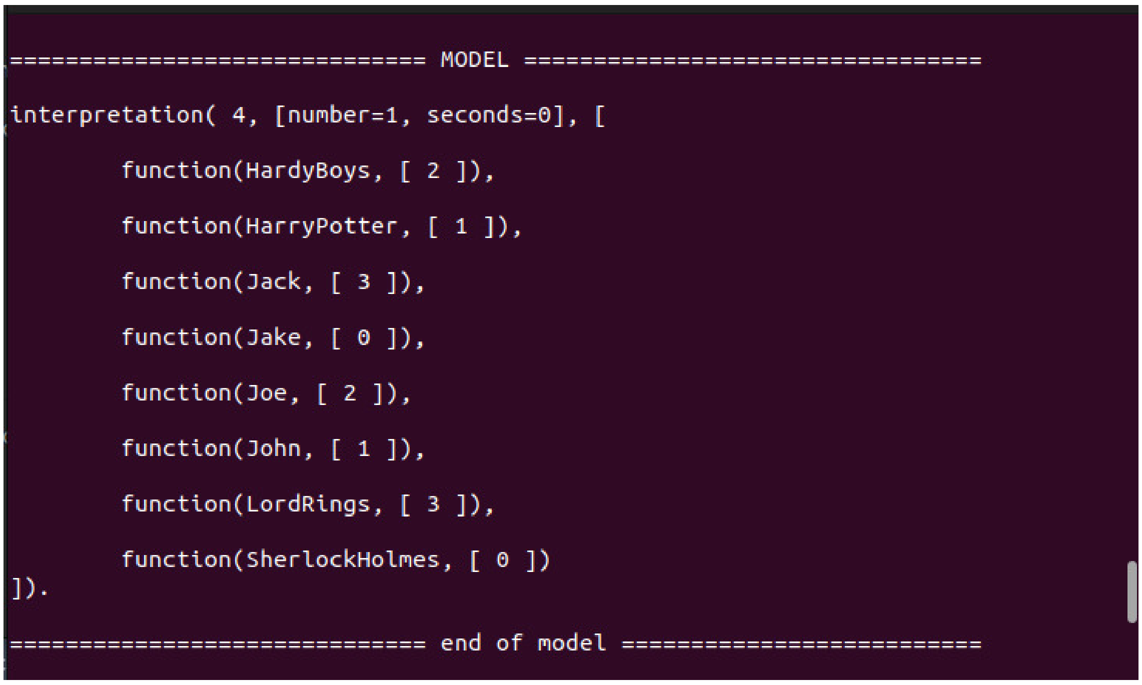
1. **Genre Preferences**: Jake and Joe favor mystery series ("Sherlock Holmes" or "The Hardy Boys"), while John and Jack prefer magical and fantasy series ("Harry Potter" or "The Lord of the Rings").

2. **Character Reference**: "Joe" is a character in Joe's favorite series ("The Hardy Boys").

3. **Halloween Costume**: John's Halloween costume, based on his favorite series, involves glasses and a black robe (indicative of "Harry Potter").

Solution

Using these deductions, the code assigns each individual to their respective favorite book series: Jake favors "Sherlock Holmes," John prefers "Harry Potter," Joe's favorite is "The Hardy Boys," and Jack's preference is "The Lord of the Rings." This solution aligns with all the provided clues and constraints.



```
============================== MODEL =================================

interpretation( 4, [number=1, seconds=0], [

        function(HardyBoys, [ 2 ]),

        function(HarryPotter, [ 1 ]),

        function(Jack, [ 3 ]),

        function(Jake, [ 0 ]),

        function(Joe, [ 2 ]),

        function(John, [ 1 ]),

        function(LordRings, [ 3 ]),

        function(SherlockHolmes, [ 0 ])
]).

============================== end of model ==========================
```

Figure 1.3: Mace4 demonstration for the problem

1.3 Graphical User Interface

1.3.1 User Interface Features

The user interface presents a window with four buttons, each associated with a logic puzzle. Clicking a button displays the puzzle's description and prompts users to reveal the solution.

1.3.2 Displaying Text

The `display_text()` function shows the puzzle description and the answer when the associated button is clicked. It uses Tkinter's Label and Button widgets.

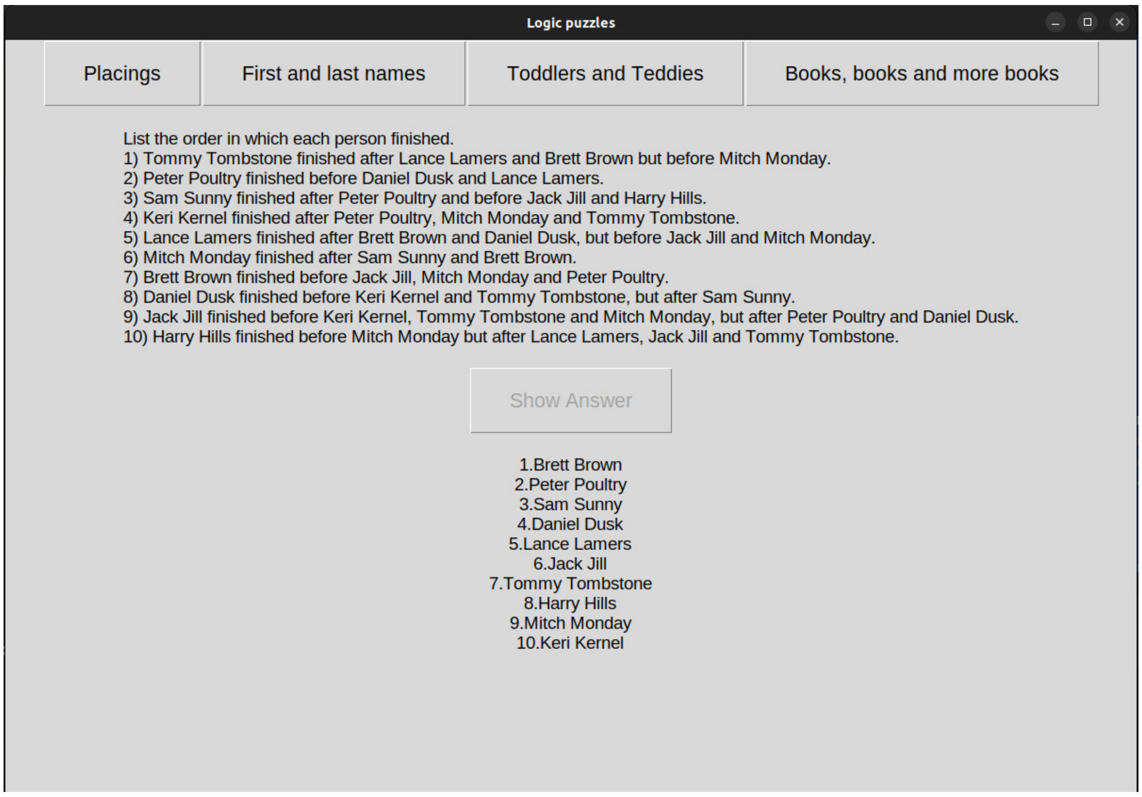
1.3.3 Button Functions

Each button is linked to a specific puzzle and calls a corresponding function when clicked (`button1_clicked()`, `button2_clicked()`, etc.).

1.3.4 Conclusion

The user interface provides an interactive way to solve logic puzzles by displaying the puzzle descriptions and their solutions upon button click.

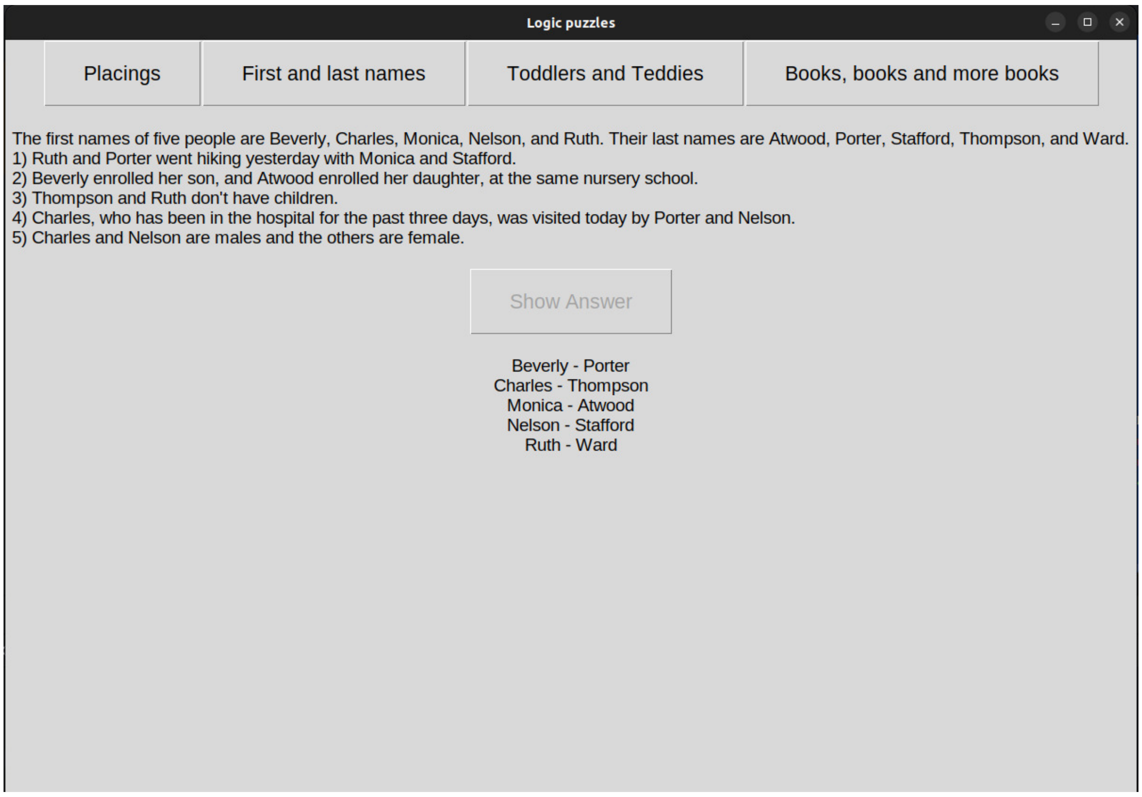
1.3.5 GUI



List the order in which each person finished.
1) Tommy Tombstone finished after Lance Lamers and Brett Brown but before Mitch Monday.
2) Peter Poultry finished before Daniel Dusk and Lance Lamers.
3) Sam Sunny finished after Peter Poultry and before Jack Jill and Harry Hills.
4) Keri Kernel finished after Peter Poultry, Mitch Monday and Tommy Tombstone.
5) Lance Lamers finished after Brett Brown and Daniel Dusk, but before Jack Jill and Mitch Monday.
6) Mitch Monday finished after Sam Sunny and Brett Brown.
7) Brett Brown finished before Jack Jill, Mitch Monday and Peter Poultry.
8) Daniel Dusk finished before Keri Kernel and Tommy Tombstone, but after Sam Sunny.
9) Jack Jill finished before Keri Kernel, Tommy Tombstone and Mitch Monday, but after Peter Poultry and Daniel Dusk.
10) Harry Hills finished before Mitch Monday but after Lance Lamers, Jack Jill and Tommy Tombstone.

```
1.Brett Brown
2.Peter Poultry
3.Sam Sunny
4.Daniel Dusk
5.Lance Lamers
6.Jack Jill
7.Tommy Tombstone
8.Harry Hills
9.Mitch Monday
10.Keri Kernel
```

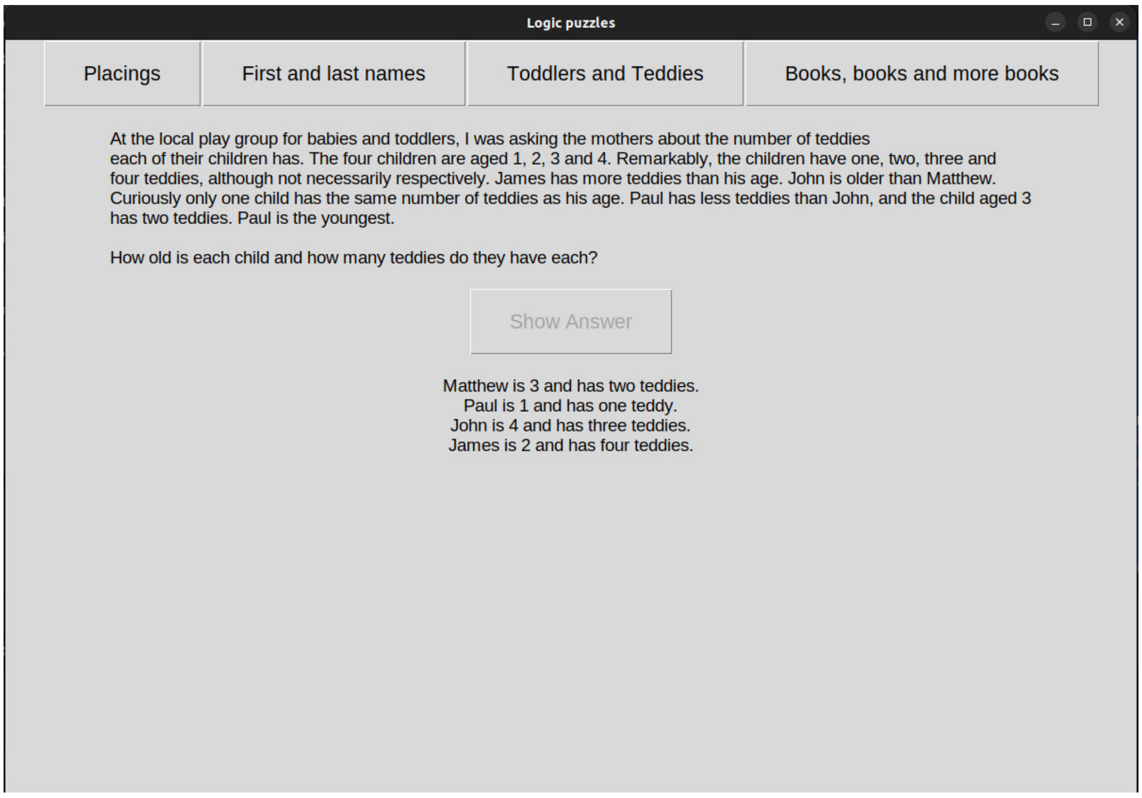
Figure 1.4: GUI for “Placings” problem



The first names of five people are Beverly, Charles, Monica, Nelson, and Ruth. Their last names are Atwood, Porter, Stafford, Thompson, and Ward.
1) Ruth and Porter went hiking yesterday with Monica and Stafford.
2) Beverly enrolled her son, and Atwood enrolled her daughter, at the same nursery school.
3) Thompson and Ruth don't have children.
4) Charles, who has been in the hospital for the past three days, was visited today by Porter and Nelson.
5) Charles and Nelson are males and the others are female.

```
Beverly - Porter
Charles - Thompson
Monica - Atwood
Nelson - Stafford
Ruth - Ward
```

Figure 1.5: GUI for “First and Last Names” problem

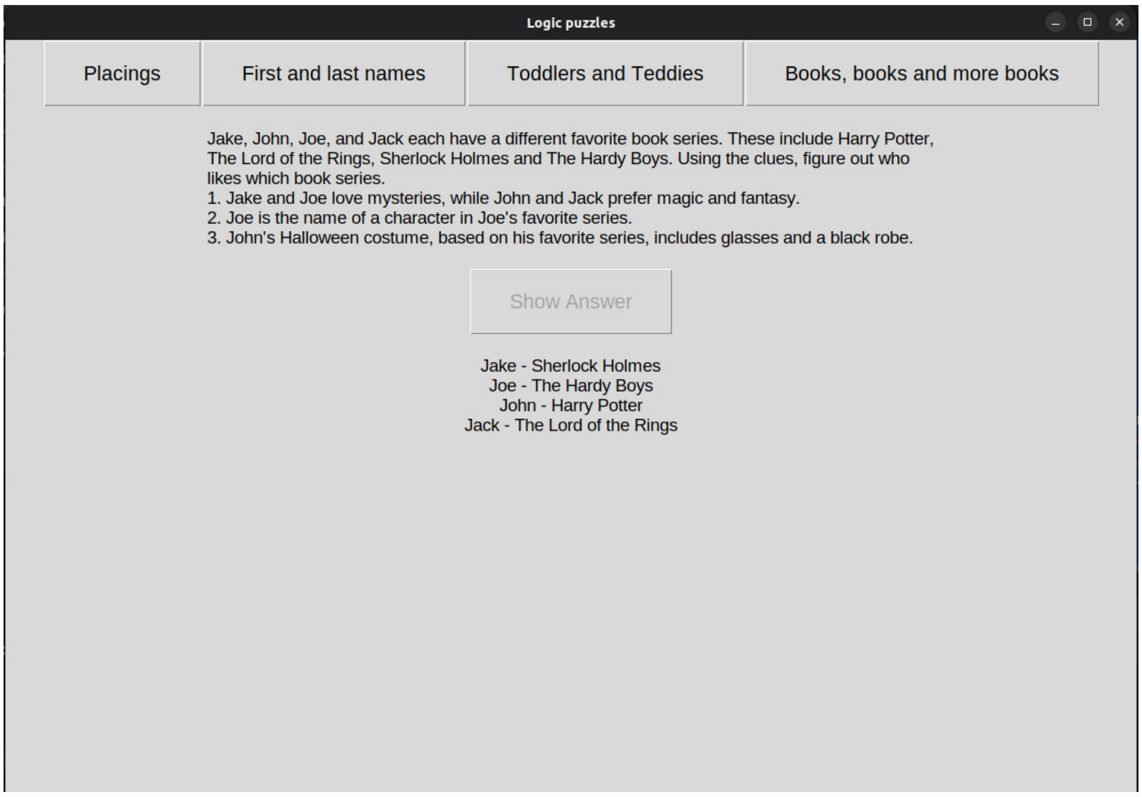


At the local play group for babies and toddlers, I was asking the mothers about the number of teddies each of their children has. The four children are aged 1, 2, 3 and 4. Remarkably, the children have one, two, three and four teddies, although not necessarily respectively. James has more teddies than his age. John is older than Matthew. Curiously only one child has the same number of teddies as his age. Paul has less teddies than John, and the child aged 3 has two teddies. Paul is the youngest.

How old is each child and how many teddies do they have each?

Matthew is 3 and has two teddies.
Paul is 1 and has one teddy.
John is 4 and has three teddies.
James is 2 and has four teddies.

Figure 1.6: GUI for “Toddlers and Teddies” problem



Jake, John, Joe, and Jack each have a different favorite book series. These include Harry Potter, The Lord of the Rings, Sherlock Holmes and The Hardy Boys. Using the clues, figure out who likes which book series.
1. Jake and Joe love mysteries, while John and Jack prefer magic and fantasy.
2. Joe is the name of a character in Joe's favorite series.
3. John's Halloween costume, based on his favorite series, includes glasses and a black robe.

Jake - Sherlock Holmes
Joe - The Hardy Boys
John - Harry Potter
Jack - The Lord of the Rings

Figure 1.7: GUI for “Books, books and more books” problem

Bibliography

https://www.braingle.com/brainteasers/29710/first-and-last-names.html
https://www.braingle.com/brainteasers/36278/placings.html
https://www.braingle.com/brainteasers/3598/toddlers-and-teddies.html
https://www.braingle.com/brainteasers/44631/books-books-and-more-books.html
https://moodle.cs.utcluj.ro/course/view.php?id=620
https://www.javatpoint.com/first-order-logic-in-artificial-intelligence

Chapter 2

Original code

The code shown below is for the "First and Last Names" puzzle:

```
set(arithmetic).
assign(domain_size, 5).

list(distinct).

[Beverly, Charles, Monica, Nelson, Ruth].
[Atwood, Porter, Stafford, Thompson, Ward].

end_of_list.

formulas(assumptions).

Beverly = 0.
Charles = 1.
Monica = 2.
Nelson = 3.
Ruth = 4.

%Clue 1. Monica's and Ruth's last name is not Porter, neither Stafford
%because they went hiking together
Monica != Porter.
Monica != Stafford.
Ruth != Porter.
Ruth != Stafford.

%Clue 2. Beverly's last name is not Atwood, because both have enrolled their
%child to the nursery school

%Atwood can not be a male
Beverly != Atwood.

%Clue 3.
Beverly != Thompson.
Ruth != Atwood.
Ruth != Thompson.
```

```
%Clue 4. Charles's last name is not Porter, because he was visited in the
%hospital by him
Charles != Porter.
Nelson != Porter.
Charles != Stafford.

%Clue 5
Charles != Atwood.
Nelson != Atwood.

end_of_list.
```

The code shown bellow is for the "Placings" puzzle:

```
formulas(assumptions).

 different(person1,person2).
 different(person1,person3).
 different(person1,person4).
 different(person1,person5).
 different(person1,person6).
 different(person1,person7).
 different(person1,person8).
 different(person1,person9).
 different(person1,person10).

 different(person2,person3).
 different(person2,person4).
 different(person2,person5).
 different(person2,person6).
 different(person2,person7).
 different(person2,person8).
 different(person2,person9).
 different(person2,person10).

 different(person3,person4).
 different(person3,person5).
 different(person3,person6).
 different(person3,person7).
 different(person3,person8).
 different(person3,person9).
 different(person3,person10).

 different(person4,person5).
 different(person4,person6).
 different(person4,person7).
 different(person4,person8).
 different(person4,person9).
 different(person4,person10).

 different(person5,person6).
 different(person5,person7).
 different(person5,person8).
 different(person5,person9).
 different(person5,person10).

 different(person6,person7).
 different(person6,person8).
 different(person6,person9).
 different(person6,person10).

 different(person7,person8).
 different(person7,person9).
 different(person7,person10).

 different(person8,person9).
 different(person8,person10).

 different(person9,person10).

 different(x,y) -> different(y,x).

 %next

-next(person1, person1).
-next(person2, person2).
-next(person3, person3).
-next(person4, person4).
-next(person5, person5).
-next(person6, person6).
-next(person7, person7).
-next(person8, person8).
-next(person9, person9).
-next(person10, person10).

 next(person1, person2).
-next(person1, person3).
-next(person1, person4).
-next(person1, person5).
-next(person1, person6).
-next(person1, person7).
-next(person1, person8).
-next(person1, person9).
-next(person1, person10).

 next(person2, person3).
-next(person2, person4).
-next(person2, person5).
-next(person2, person6).
-next(person2, person7).
-next(person2, person8).
```

```
-next(person2, person9).
-next(person2, person10).

 next(person3, person4).
-next(person3, person5).
-next(person3, person6).
-next(person3, person7).
-next(person3, person8).
-next(person3, person9).
-next(person3, person10).

 next(person4, person5).
-next(person4, person6).
-next(person4, person7).
-next(person4, person8).
-next(person4, person9).
-next(person4, person10).

 next(person5, person6).
-next(person5, person7).
-next(person5, person8).
-next(person5, person9).
-next(person5, person10).

 next(person6, person7).
-next(person6, person8).
-next(person6, person9).
-next(person6, person10).

 next(person7, person8).
-next(person7, person9).
-next(person7, person10).

 next(person8, person9).
-next(person8, person10).

next(person9, person10).

-next(person2, person1).
-next(person3, person1).
-next(person4, person1).
-next(person5, person1).
-next(person6, person1).
-next(person7, person1).
-next(person8, person1).
-next(person9, person1).
-next(person10, person1).

-next(person3, person2).
-next(person4, person2).
```

```
-next(person5, person2).
-next(person6, person2).
-next(person7, person2).
-next(person8, person2).
-next(person9, person2).
-next(person10, person2).

-next(person4, person3).
-next(person5, person3).
-next(person6, person3).
-next(person7, person3).
-next(person8, person3).
-next(person9, person3).
-next(person10, person3).

-next(person5, person4).
-next(person6, person4).
-next(person7, person4).
-next(person8, person4).
-next(person9, person4).
-next(person10, person4).

-next(person6, person5).
-next(person7, person5).
-next(person8, person5).
-next(person9, person5).
-next(person10, person5).

-next(person7, person6).
-next(person8, person6).
-next(person9, person6).
-next(person10, person6).

-next(person8, person7).
-next(person9, person7).
-next(person10, person7).

-next(person9, person8).
-next(person10, person8).

-next(person10, person9).

%next(x, y) -> next(y, x).

TommyTombstone(x) | LanceLamers(x) | BrettBrown(x) | MitchMonday(x) | PeterPoultry(x)
| DanielDusk(x) | SamSunny(x) | JackJill(x) | HarryHills(x) | KeriKernel(x).

TommyTombstone(x) & TommyTombstone(y) -> -different(x, y).
LanceLamers(x) & LanceLamers(y) -> -different(x, y).
BrettBrown(x) & BrettBrown(y) -> -different(x, y).
```

```
MitchMonday(x) & MitchMonday(y) -> -different(x, y).
PeterPoultry(x) & PeterPoultry(y) -> -different(x, y).
DanielDusk(x) & DanielDusk(y) -> -different(x, y).
SamSunny(x) & SamSunny(y) -> -different(x, y).
JackJill(x) & JackJill(y) -> -different(x, y).
HarryHills(x) & HarryHills(y) -> -different(x, y).
KeriKernel(x) & KeriKernel(y) -> -different(x, y).

%tasks

%1
LanceLamers(x) & TommyTombstone(y) -> next(x,y).
BrettBrown(x) & TommyTombstone(y) -> next(x,y).
TommyTombstone(x) & MitchMonday(y) -> next(x,y).

%2
PeterPoultry(x) & DanielDusk(y) -> next(x,y).
PeterPoultry(x) & LanceLamers(y) -> next(x,y).

%3
PeterPoultry(x) & SamSunny(y) -> next(x,y).
SamSunny(x) & JackJill(y) -> next(x,y).
SamSunny(x) & HarryHills(y) -> next(x,y).

%4
PeterPoultry(x) & KeriKernel(y) -> next(x,y).
MitchMonday(x) & KeriKernel(y) -> next(x,y).
TommyTombstone(x) & KeriKernel(y) -> next(x,y).

%5
BrettBrown(x) & LanceLamers(y) -> next(x,y).
DanielDusk(x) & LanceLamers(y) -> next(x,y).
LanceLamers(x) & JackJill(y) -> next(x,y).
LanceLamers(x) & MitchMonday(y) -> next(x,y).

%6
SamSunny(x) & MitchMonday(y) -> next(x,y).
BrettBrown(x) & MitchMonday(y) -> next(x,y).

%7
BrettBrown(x) & JackJill(y) -> next(x,y).
BrettBrown(x) & MitchMonday(y) -> next(x,y).
BrettBrown(x) & PeterPoultry(y) -> next(x,y).

%8
DanielDusk(x) & KeriKernel(y) -> next(x,y).
DanielDusk(x) & TommyTombstone(y) -> next(x,y).
SamSunny(x) & DanielDusk(y) -> next(x,y).

%9
```

```
JackJill(x) & KeriKernel(y) -> next(x,y).
JackJill(x) & TommyTombstone(y) -> next(x,y).
JackJill(x) & MitchMonday(y) -> next(x,y).
PeterPoultry(x) & JackJill(y) -> next(x,y).
DanielDusk(x) & JackJill(y) -> next(x,y).

%10
HarryHills(x) & MitchMonday(y) -> next(x,y).
LanceLamers(x) & HarryHills(y) -> next(x,y).
JackJill(x) & HarryHills(y) -> next(x,y).
TommyTombstone(x) & HarryHills(y) -> next(x,y).
```

The code shown bellow is for the "Toddlers and Teddies" puzzle:

```
set(arithmetic).
assign(domain_size, 4).

list(distinct).

[James, John, Matthew, Paul].
[t1, t2, t3, t4].

end_of_list.

formulas(assumptions).

%Paul is the youngest, so he is 1 year old
%James has more teddies than his age and Paul is 1 year old, so James can be either
%2 or 3 years old
%John being older than Matthew and Paul being 1 means that Matthew can be only
%2 or 3 years old and that John can be 3 or 4
%The child aged 3 has two teddies, so that he can't be James so it's either
%John or Matthew -> James is 2 years old
%There remains two toddlers without an age assigned and they are John and Matthew.
%John being older than Matthew, it means that Matthew is 3 and John is 4
Paul = 0.
James = 1.
Matthew = 2.
John = 3.

%Child aged 3(Matthew) has 2 teddies
Matthew = t2.

%James(being 2) can have either 3 or 4 teddies
James = t3 | James = t4.

%Matthew and James both having a number of teddies different to their age, it remains
%that Paul and John have the same number of teddies as their age
Paul = t1 | John = t4.

%If Paul has 3 teddies, then John will have 4 teddies and James 1 that will be in
```

```
%contradiction with this statement: "James has more teddies than his age".
%Paul can't have 4 teddies because that way will have more than John(which is not
%allowed) That remains that John can't have 4 teddies
John != t4.
```

The code shown bellow is for the "Books, Books and more Books" puzzle:

```
set(arithmetic).
assign(domain_size, 4).

list(distinct).

[Jake, John, Joe, Jack].
[HarryPotter, LordRings, SherlockHolmes, HardyBoys].

end_of_list.

formulas(assumptions).

Jake = 0.
John = 1.
Joe = 2.
Jack = 3.

%1. Jake and Joe love mysteries(Sherlock Holmes and The Hardy Boys),
Jake = SherlockHolmes | Jake = HardyBoys.
Joe = SherlockHolmes | Joe = HardyBoys.
% while John and Jack prefer magic and fantasy(Harry Potter and Lord of the Rings).
John = HarryPotter | John = LordRings.
Jack = HarryPotter | Jack = LordRings.
%2. "Joe" is the name of a character in Joe's favorite series(Joe Hardy in The Hardy
$boys).
Joe = HardyBoys.
%3. John's Halloween costume, based on his favorite series, includes glasses and a
%black robe(Harry Potter).
John = HarryPotter.

end_of_list.
```