Department of Computer Science
Technical University of Cluj-Napoca

Artificial Intelligence

Laboratory activity

Name: Patica Andreea-Alexandra and Popa Adriana
Group: 30433
Email: popaadriana2606@gmail.com

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

Contents

Table 1: Lab scheduling

Activity	**Deadline**
Searching agents, Linux, Latex, Python, Pacman	W_1
Uninformed search	W_2
Informed Search	W_3
Adversarial search	W_4
Propositional logic	W_5
First order logic	W_6
Inference in first order logic	W_7
Knowledge representation in first order logic	W_8
Classical planning	W_9
Contingent, conformant and probabilistic planning	W_{10}
Multi-agent planing	W_{11}
Modelling planning domains	W_{12}
Planning with event calculus	W_{14}

0.0.0.0.1 Lab organisation.

1. Laboratory work is 25% from the final grade.
2. There are three deliverables in total: 1. Search, 2. Logic, 3. Planning.
3. Before each deadline, you have to send your work (latex documentation/code) at moodle.cs.utcluj.ro
4. We use Linux and Latex
5. Plagiarism: Don't be a cheater! Cheating affects your colleagues, scholarships and a lot more.

Chapter 1

A1: Search

Chapter 2

A2: Logics

2.1 Brief Introduction

For this assignment we decided to solve a series of comparison puzzles which are based on propositional logic. The problems were chosen based on their difficulty level, as well as, their interesting and challenging content. In order of their difficulty, the problems are the following :

1. Who is taller? (link)
2. Youngest or eldest? (link)
3. Wide Awake (link)
4. Just Married (link)

As we formed teams of two for this assignment, we decided that each member would solve a set consisting of one easy problem and one hard problem.

2.2 Implementation

In order to solve this problem, we used the First Order Logic method in which each sentence, or statement, is broken down into a subject and a predicate. The predicate modifies or defines the properties of the subject.

2.2.1 First Problem: Who is Taller?

The first problem consists of 10 questions about people's heights. In this section we will shortly resume the implementation, show the implementation and explain the solution for each of them.

1. James is taller than Kate and Carly. Sammy is shorter than Kate. Natalie is shorter than Kate and Sammy, however Sammy is shorter than Carly. Who is the shortest?

In this problem there could be identified 5 people: James, Kate, Carly, Sammy and Natalie. We defined a predicate, *Shorter(x,y)*, which established which person is shorter between x and y. This means that x is shorter than y and y is not short. We defined a function *Short(x,y)*, which established that x is short.

Then the problem was broken into smaller pieces and we used propositional logic. The statements are:

- James is taller than Kate and Carly.
- Sammy is shorter than Kate.
- Natalie is shorter than Kate and Sammy,
- however Sammy is shorter than Carly.

The final question is: ”Who is the shortest?”. The final and correct answer is Natalie.

2. Rachel and Jessica are taller than John. Maria is shorter than Lara and Lara is taller than Sally. John is taller than Lara. Who is the tallest out of John, Maria, Lara and Sally?

In this problem there could be identified six people: Rachel, Jessica, John, Maria, Lara and Sally. We defined a predicate, *Taller(x,y)*, which established which person is taller between x and y. In this case, x will be taller than y. This means that y is not tall. We defined a function, *Tall(x)*, which establishes that x is tall.

Then the problem was broken into smaller pieces and we used propositional logic. The statements are:

- Rachel and Jessica are taller than John.
- Maria is shorter than Lara and Lara is taller than Sally. This means that Lara is taller than Maria.
- John is taller than Lara.

The final question is: ”Who is the tallest out of John, Maria, Lara and Sally?”. This means that we should exclude Rachel and Jessica which are obviously the tallest. For this, we created equivalent statements which include only the four: John, Maria, Lara and Sally. There are four statements, like the number of possible scenarios, and only one of them is true.

The final and correct answer is John.

3. Joy is taller than Cassie. Joy is taller than Brian who is taller than Cassie and Geoff. Geoff is shorter than Val who is shorter than Cassie. Is Val the shortest?

In this problem there could be identified 5 people: Joy, Cassie, Brian, Geoff and Val. We defined a predicate, *Shorter(x,y)*, which established which person is shorter between x and y. This means that x is shorter than y and y is not short. We defined a function *Short(x,y)*, which established that x is short.

Then the problem was broken into smaller pieces and we used propositional logic. The statements are:

- Joy is taller than Cassie.
- Joy is taller than Brian who is taller than Cassie and Geoff.
- Geoff is shorter than Val who is shorter than Cassie.

The final question is: ”Is Val the shortest?”. The final and correct answer is No, Geoff is the shortest.

4. Robert is taller than Jordan. Robert and Jordan are shorter than Lorraine. Robert is taller than Peter who is taller than Kylie. Jordan is taller than Peter. Who is the tallest?

In this problem there could be identified five people: Robert, Jordan, Lorraine, Peter, Kylie. We defined a predicate, *Taller(x,y)*, which established which person is taller between x and y. In this case, x will be taller than y. This means that y is not tall. We defined a function, *Tall(x)*, which establishes that x is tall.

Then the problem was broken into smaller pieces and we used propositional logic. The statements are:

- Robert is taller than Jordan.
- Robert and Jordan are shorter than Lorraine. This means that Lorraine is taller than Robert and Jordan.
- Robert is taller than Peter who is taller than Kylie.
- Jordan is taller than Peter

The final question is: "Who is the tallest?". The final and correct answer is Lorraine.

5. Luke is taller than Bianca who is taller than Nathan. Nathan is shorter than Cathy who is shorter than Bianca. Who is the shortest?

In this problem there could be identified four people: Luke, Bianca, Nathan, Cathy. We defined a predicate, *Shorter(x,y)*, which established which person is shorter between x and y. In this case, x will be shorter than y. This means that y is not short. We defined a function, *Short(x)*, which establishes that x is short.

Then the problem was broken into smaller pieces and we used propositional logic. The statements are:

- Luke is taller than Bianca who is taller than Nathan. This means that Bianca is shorter than Luke and Nathan is shorter than Bianca.
- Nathan is shorter than Cathy who is shorter than Bianca.

The final question is: "Who is the shortest?". The final and correct answer is Nathan.

6. Danny is taller than Lisa who is shorter than Elise who is shorter than Danny. Who is the shortest?

In this problem there could be identified three people: Danny, Lisa and Elise. We defined a predicate, *Shorter(x,y)*, which established which person is shorter between x and y. In this case, x will be shorter than y. This means that y is not short. We defined a function, *Short(x)*, which establishes that x is short.

The problem is short, so there was no need of breaking it into statements. The only statement was :

- Danny is taller than Lisa who is shorter than Elise who is shorter than Danny.

The final question is: "Who is the shortest?". The final and correct answer is Lisa.

7. Brett is taller than Shannon who is taller than Rene. Brett is also taller than Mel. Who is the shortest?

In this problem there could be identified four people: Brett, Shannon, Rene, Mel. We defined a predicate, *Shorter(x,y)*, which established which person is shorter between x and y.

In this case, x will be shorter than y. This means that y is not short. We defined a function, *Short(x)*, which establishes that x is short.

Then the problem was broken into smaller pieces and we used propositional logic. The statements are:

- Brett is taller than Shannon who is taller than Rene. It means that Shannon is shorter than Brett and Rene is shorter than Shannon.
- Brett is also taller than Mel.

The final question is: "Who is the shortest?". The question cannot be answered due to the fact that the problem does not provide enough information. From the first sentence results that Rene is shorter in the group of three (Brett, Shannon, Rene). The second statement implies that Mel is shorter than Brett. Although, this information does not help us into finding the answer as there is no information between Rene and Mel, only the one which states that Brett is taller than both of them.

The final and correct answer is "Not enough info".

8. Sophie is shorter than Harry. Will is shorter than Mike who is shorter than Harry. Harry and Will are shorter than Fiona. Who is the tallest?

In this problem there could be identified five people: Sophie, Harry, Will, Mike and Fiona. We defined a predicate, *Taller(x,y)*, which established which person is taller between x and y. In this case, x will be taller than y. This means that y is not tall. We defined a function, *Tall(x)*, which establishes that x is tall.

Then the problem was broken into smaller pieces and we used propositional logic. The statements are:

- Sophie is shorter than Harry.
- Will is shorter than Mike who is shorter than Harry.
- Harry and Will are shorter than Fiona.

The final question is: "Who is the tallest?". The final and correct answer is Fiona.

9. Tom is shorter than Matt and Tamara. Matt is shorter than Ben. Tamara is taller than Ben. Who is the tallest?

In this problem there could be identified four people: Tom, Matt, Tamara, Ben. We defined a predicate, *Taller(x,y)*, which established which person is taller between x and y. In this case, x will be taller than y. This means that y is not tall. We defined a function, *Tall(x)*, which establishes that x is tall.

Then the problem was broken into smaller pieces and we used propositional logic. The statements are:

- Tom is shorter than Matt and Tamara. This means that Matt is taller than Tom.
- Matt is shorter than Ben. It means that Ben is taller than Matt.
- Tamara is taller than Ben.

The final question is: "Who is the tallest?". The final and correct answer is Tamara.

10. Jen is taller than Sue who is taller than Dick. Alan is taller than Dick. Jen and Sue are shorter than Alan. Betty is shorter than Dick. Who is the tallest?

In this problem there could be identified five people: Jen, Seu, Dick, Alan and Betty. We defined a predicate, *Taller(x,y)*, which established which person is taller between x and y. In this case, x will be taller than y. This means that y is not tall. We defined a function, *Tall(x)*, which establishes that x is tall.
Then the problem was broken into smaller pieces and we used propositional logic. The statements are:

- Jen is taller than Sue who is taller than Dick.
- Alan is taller than Dick.
- Jen and Sue are shorter than Alan.
- Betty is shorter than Dick.

The final question is: "Who is the tallest?". The final and correct answer is Allan.

2.2.2 Second Problem: Youngest or eldest?

The Second problem consists of 10 questions about people's age. In this section we will shortly resume the implementation, show the implementation and explain the solution for each of them.

1. Harmony is older than Connor and younger than Maranda. Isabelle is younger than Edwardo and Maranda. Edwardo is older than Maranda. If Harmony and Isabelle are the same age, who's youngest?

In this problem there could be identified five people: Harmony, Connor, Maranda, Eduardo and Isabelle. We defined a predicate, *Younger(x,y)*, which established which person is younger between x and y. In this case, x will be younger than y. This means that y is not young. We defined a function, *Young(x)*, which establishes that x is young.
Then the problem was broken into smaller pieces and we used propositional logic. The statements are:

- Harmony is older than Connor and younger than Maranda.
- Isabelle is younger than Edwardo and Maranda.
- Edwardo is older than Maranda.
- If Harmony and Isabelle are the same age,

The final question is: "If Harmony and Isabelle are the same age, who's youngest?". The final and correct answer is Connor.

2. Ontario is older than Waryn. Waryn is younger than Anastasia and younger than Desmond. Desmond is older than Ontario and Alonso, who are older than Anastasia. If Alonso and Ontario are twins, who's oldest?

In this problem there could be identified five people: Ontario, Waryn, Anastasia, Desmond and Alonso. We defined a predicate, *Older(x,y)*, which established which person is older between x and y. In this case, x will be older than y. This means that y is not old. We defined a function, *Old(x)*, which establishes that x is old.

Then the problem was broken into smaller pieces and we used propositional logic. The statements are:

- Ontario is older than Waryn.
- Waryn is younger than Anastasia and younger than Desmond. It means that Anastasia is older than Waryn and Desmond is older than Anastasia.
- Desmond is older than Ontario and Alonso, who are older than Anastasia.
- Alonso and Ontario are twins. This means that none of them is older, i.e. Alonso is not older than Ontario and Ontario is not older than Alonso.

The final question is: "Who is the oldest?". The final and correct answer is Desmond.

3. Quentyn is older than Porfirio and younger than Caitlyn. Porfirio is younger than Joslyn and younger than Caitlyn. If Quentyn Is older than Joslyn, who's oldest?

In this problem there could be identified four people: Quentyn, Profirio, Caitlyn and Joslyn. We defined a predicate, *Older(x,y)*, which established which person is older between x and y. In this case, x will be older than y. This means that y is not old. We defined a function, *Old(x)*, which establishes that x is old.

Then the problem was broken into smaller pieces and we used propositional logic. The statements are:

- Quentyn is older than Porfirio and younger than Caitlyn.
- Porfirio is younger than Joslyn and younger than Caitlyn.
- If Quentyn Is older than Joslyn,

The final question is: "If Quentyn Is older than Joslyn, who's oldest?". The final and correct answer is Caitlyn.

4. Brendyn is older than Liam and Fergus. Marcell is older than Fergus and younger than Liam. If Liam has a brother named Lucian, who is older than Brendyn, who's youngest?

In this problem there could be identified five people: Brendyn, Liam, Marcell, Fergus and Lucian. We defined a predicate, *Younger(x,y)*, which established which person is younger between x and y. In this case, x will be younger than y. This means that y is not young. We defined a function, *Young(x)*, which establishes that x is young.

Then the problem was broken into smaller pieces and we used propositional logic. The statements are:

- Brendyn is older than Liam and Fergus. It means that Fergus is younger than Brendyn and Liam is younger than Brendyn.
- Marcell is older than Fergus and younger than Liam. This implies the fact that Fergus is younger than Marcell.

- Lucian is older than Brendyn. From this results that Brendyn is younger than Lucian.

The final question is: "Who is the youngest?". The final and correct answer is Fergus.

5. Cheveyo and Ethan are older than Devlyn and Victoria. Maria is older than Victoria, but younger than both Ethan and Devlyn. Cheveyo is older than Maria. If he is also younger than Ethan, who's youngest?

In this problem there could be identified five people: Cheveyo, Ethan, Devlyn, Victoria and Maria. We defined a predicate, *Younger(x,y)*, which established which person is younger between x and y. In this case, x will be younger than y. This means that y is not young. We defined a function, *Young(x)*, which establishes that x is young.
Then the problem was broken into smaller pieces and we used propositional logic. The statements are:

- Cheveyo and Ethan are older than Devlyn and Victoria.
- Maria is older than Victoria, but younger than both Ethan and Devlyn.
- Cheveyo is older than Maria.
- If he is also younger than Ethan,

The final question is: "If he is also younger than Ethan, who's youngest?". The final and correct answer is Victoria.

6. Minnie is older than Justin and Tiffany. Jaymee is younger than Justin and older than Tiffany. Georgette is younger than Minnie and older than Tiffany. If Jaymee and Georgette are the same age, who's oldest?

In this problem there could be identified five people: Minnie, Justin, Tiffany, Jaymee and Justin. We defined a predicate, *Older(x,y)*, which established which person is older between x and y. In this case, x will be older than y. This means that y is not old. We defined a function, *Old(x)*, which establishes that x is old.
Then the problem was broken into smaller pieces and we used propositional logic. The statements are:

- Minnie is older than Justin and Tiffany.
- Jaymee is younger than Justin and older than Tiffany. It implies that Justin is older than Jaymee.
- Georgette is younger than Minnie and older than Tiffany. This means that Minnie is older than Georgette.
- Jaymee and Georgette are the same age. From this results than Jaymee is not older than Georgette and Georgette is not older than Jaymee.

The final question is: "Who is the oldest?". The final and correct answer is Minnie.

7. Elysha is older than Logan and younger than Edward. Jo is older than Michael. Edy is older than Michael and younger than Jo. Michael is older than Edward and Elysha. If Edward and Elysha are brother and sister, and Edy is their father, who's youngest?

In this problem there could be identified six people: Elysha, Logan, Edward, Jo, Michael and Edy. We defined a predicate, *Younger(x,y)*, which established which person is younger between x and y. In this case, x will be younger than y. This means that y is not young. We defined a function, *Young(x)*, which establishes that x is young.

Then the problem was broken into smaller pieces and we used propositional logic. The statements are:

- Elysha is older than Logan and younger than Edward.
- Jo is older than Michael.
- Edy is older than Michael and younger than Jo.
- Michael is older than Edward and Elysha.
- If Edward and Elysha are brother and sister, and Edy is their father,

The final question is: "who's youngest?". The final and correct answer is Logan.

8. Chris is younger than Ruth and Sam, but older than Stephanie. Ruth is older than Stephanie and younger than Curtis. Curtis is younger than Sam and older than Chris. If Chris has a brother named Brian who's younger than Stephanie, who's oldest?

In this problem there could be identified six people: Chris, Ruth, Sam, Stephanie, Curtis and Brian. We defined a predicate, *Older(x,y)*, which established which person is older between x and y. In this case, x will be older than y. This means that y is not old. We defined a function, *Old(x)*, which establishes that x is old.

Then the problem was broken into smaller pieces and we used propositional logic. The statements are:

- Chris is younger than Ruth and Sam, but older than Stephanie. It means that Ruth is older than Chris.
- Ruth is older than Stephanie and younger than Curtis. This implies that Curtis is older than Ruth.
- Curtis is younger than Sam and older than Chris. This means that Sam is older than Curtis.
- Brian is younger than Stephanie, then Stephanie is older than Brian.

The final question is: "Who is the oldest?". The final and correct answer is Sam.

9. Roshontia and Eric are older than Justine. Jadira is older than Colleen and Roshontia. If Justine is older than Colleen, that makes her oldest.

In this problem there could be identified five people: Roshontia, Eric, Justine, Jadira and Coleen. We defined a predicate, *Older(x,y)*, which established which person is older between x and y. In this case, x will be older than y. This means that y is not old. We defined a function, *Old(x)*, which establishes that x is old.

Then the problem was broken into smaller pieces and we used propositional logic. The statements are:

- Roshontia and Eric are older than Justine.

- Jadira is older than Colleen and Roshontia.
- If Justine is older than Colleen,

The final question is: "If Justine is older than Colleen, that makes her oldest.". There are still valid models even if Justine is not the oldest and Eric or Jadira are, so she may not be the oldest. Additional information is needed to prove who is the oldest.

10. Preston is older than Corissa and younger than Pedro. Corissa is older than Andrea and Madison. Andrea and Madison are older than Chloe. Andrea is younger than Caspian and older than Madison. Pedro is older than Corissa and Madison, but younger than Caspian. If Preston is Caspian's son, who's youngest?

In this problem there could be identified seven people: Preston, Corissa, Pedro, Andrea, Madison, Chloe and Caspian. We defined a predicate, *Younger(x,y)*, which established which person is younger between x and y. In this case, x will be younger than y. This means that y is not young. We defined a function, *Young(x)*, which establishes that x is young.

Then the problem was broken into smaller pieces and we used propositional logic. The statements are:

- Preston is older than Corissa and younger than Pedro. This means that Corissa is younger than Preston.
- Corissa is older than Andrea and Madison. It implies that Andrea and Madison are younger than Corissa.
- Andrea and Madison are older than Chloe, then Chloe is younger than Andrea and Madison.
- Andrea is younger than Caspian and older than Madison. From this results that Madison is younger than Andrea.
- Pedro is older than Corissa and Madison, but younger than Caspian, then Corissa and Madison are younger than Pedro.
- Preston is Caspian's son. It means that Preston is younger than Caspian.

The final question is: "Who is the youngest?". The final and correct answer is Chloe.

2.2.3 Third Problem: Wide Awake

The third problem is about people calling at night for different reasons. the task is to figure out who called at each hour and why.

The friends: Barbara, Carla, Edna, Fiona, Mary The times: 12:00 a.m. (midnight), 1:00 a.m., 2:00 a.m., 3:00 a.m., 4:00 a.m. The reasons: caffeine, worry, late exercise, hunger, snoring spouse

The clues:

A - "I was so busy today," said one caller, "that I couldn't do my weightlifting until after supper, and now I'm wired." She told me this some time before Fiona called.

B - The late exerciser called at some point after the person who had drunk ten cups of coffee that evening and was suffering from caffeine overload.

C - Carla called first.

D - Carla is not the one dieting to lose weight and too famished to sleep.

E - Mary called at some point before the coffee-drinker and also at some point before the person suffering from worry (about an upcoming trial).

F - Edna, a Mormon, avoids all stimulant beverages.

For the hours they called we defined predicates *Called 12am(x)* ,*Called 1am(x)*, *Called 2am(x)*, *Called 3am(x)*, *Called 4am(x)* to determine which person called at which hour. We deduce that each person called once and everybody called at least once.

For the hours they called we defined predicates *Exercise(x)* ,*Worry(x)*, *Hunger(x)*, *Coffee(x)*, *Snoring(x)* to determine the reason each person called. We deduce that each person had one reason and everybody had a reason.

Then the problem was broken into smaller pieces and we used propositional logic. The statements are:

- ”I was so busy today,” said one caller, ”that I couldn't do my weightlifting until after supper, and now I'm wired.” She told me this some time before Fiona called. -¿ We considered that the caller called before Fiona and he was the Exerciser. We eliminated the cases where Fiona was the weightlifter.

- The late exerciser called at some point after the person who had drunk ten cups of coffee that evening and was suffering from caffeine overload. -¿ We eliminated the cases where the coffee drinker called before the exerciser.

- Carla called first.

- Carla is not the one dieting to lose weight and too famished to sleep.

- Mary called at some point before the coffee-drinker and also at some point before the person suffering from worry (about an upcoming trial). -¿ We eliminated the cases where Mary was the coffee-drinker or the worried one, and also the cases where she called after them.

- Edna, a Mormon, avoids all stimulant beverages.

The final and correct answer is:

- Carla called at 12am and she has a snoring husband.

- Mary called at 1am and was kept awake by hunger.

- Barbara called at 2am and she drank too much coffee.

- Edna called at 3am and she was the exerciser.

- Fiona called at 4am and she was worried.

2.2.4 Fourth Problem: Just Married

The Fourth problem is a problem about just-married people. Five couples go on honeymoons to five different cities. The task is to figure out who is married to whom and where they went on their honeymoon.

There are five husbands: Ben, Todd, Brandon, Buddy and Jack. There are five wives: Lucy, Emma, Tabitha, Brandy and Ann. The five honeymoon cities are: Rome, Munich, London, Madrid and Paris.

The clues:

1. Ben's spouse did not go to Paris
2. Emma is not married to Brandon and did not travel to London or Rome
3. Jack visited Munich
4. Brandon and his new bride visited London
5. Todd is married to Lucy
6. Ann is not married to Jack or Ben and did not go to Madrid
7. The new bride with the shortest name did not go to Paris
8. Buddy is married to Brandy, they did not go to Rome or Paris

For this problem there is defined a predicate, *Married(x,y)*, which established which person is married to whom. This implies that x is married to y and y is married to x. Then all possibilities are considered. The possibility of being married to more persons or being married to oneself are eliminated.

For the honeymoon cities there was defined a function representing every city, *Rome(x)*, *Munich(x)*, *London(x)*, *Madrid(x)*, *Paris(x)*. Then there is defined the relation between married people, i.e. if x is married to y and x visited a city, then y visited the same city. There cannot be two couples who visited the same city and a couple cannot visit more than one city.

Then the problem was broken into smaller pieces and we used propositional logic. The statements are:

- Ben's spouse did not go to Paris. For this condition, we considered all the cases in which Ben could be married to and stated that the wife in each did not go to Paris.
- Emma is not married to Brandon and did not travel to London or Rome. We eliminated the case in which Emma is not married to Brandon and did not go to London and Rome.
- Jack visited Munich.
- Brandon and his new bride visited London. For this condition, we considered all the cases in which Brandon could be married to and stated that the wife went to London.
- Todd is married to Lucy.
- Ann is not married to Jack or Ben and did not go to Madrid.
- The new bride with the shortest name did not go to Paris. The bride with the shortest name is Ann.

- Buddy is married to Brandy, they did not go to Rome or Paris.

The final and correct answer is:

- Brandon is married to Ann and they went to London.
- Buddy is married to Brandy and they went to Madrid.
- Jack is married to Emma and they went to Munich.
- Todd is married to Lucy and they went to Paris.
- Ben is married to Tabitha and they went to Rome.

Chapter 3

A3: Planning

Bibliography

Appendix A

Your original code

Don't be a cheater! Cheating affects your colleagues, scholarships and a lot more. This section should contain only code developed by you, without any line re-used from other sources. This section helps me to correctly evaluate your amount of work and results obtained.

A.1

A.2 Logics

A.2.1 First Problem: Who is taller?

A.2.1.1 James is taller than Kate and Carly. Sammy is shorter than Kate. Natalie is shorter than Kate and Sammy, however Sammy is shorter than Carly. Who is the shortest?

```
assign(report_stderr, 2).
set(ignore_option_dependencies). % GUI handles dependencies

if(Prover9). % Options for Prover9
  assign(max_seconds, 60).
end_if.

if(Mace4). % Options for Mace4
  assign(domain_size, 5).
  assign(start_size, 5).
  assign(end_size, 5).
  assign(max_models, -1).
  assign(max_seconds, 60).
end_if.

formulas(assumptions).

James = 0.
Kate = 1.
Carly = 2.
Sammy = 3.
Natalie = 4.
```

```
Short(James) | Short(Kate) | Short(Carly) | Short(Sammy) | Short(Natalie).
Shorter(x,y) -> -Short(y).

Shorter(x,y) & Shorter(y,z) -> Shorter(x,z).

% James is taller than Kate and Carly
Shorter(Kate,James).
Shorter(Carly,James).

% Sammy is shorter than Kate.
Shorter(Sammy,Kate).

% Natalie is shorter than Kate and Sammy,
Shorter(Natalie,Kate).
Shorter(Natalie,Sammy).

% however Sammy is shorter than Carly
Shorter(Sammy,Carly).

end_of_list.

formulas(goals).

% Who is the shortest?
Short(Natalie).

end_of_list.
```

A.2.1.2 Rachel and Jessica are taller than John. Maria is shorter than Lara and Lara is taller than Sally. John is taller than Lara. Who is the tallest out of John, Maria, Lara and Sally?

```
formulas(assumptions).

Rachel != Jessica.
Rachel != John.
Rachel != Maria.
Rachel != Lara.
Rachel != Sally.

Jessica != John.
Jessica != Maria.
Jessica != Lara.
Jessica != Sally.

John != Maria.
John != Lara.
John != Sally.
```

```
Maria != Lara.
Maria != Sally.

Lara != Sally.

Tall(Rachel) | Tall(Jessica) | Tall(John) | Tall(Maria) | Tall(Lara) | Tall(
   ↪ Sally).
Taller(x,y) -> -Tall(y).
Taller(x,y) & Taller(y,z) -> Taller(x,z).

%Rachel and Jessica are taller than John
Taller(Rachel, John).
Taller(Jessica, John).


%Maria is shorter than Lara and Lara is taller than Sally.
Taller(Lara, Maria).
Taller(Lara, Sally).


%John is taller than Lara
Taller(John, Lara).




m1 | m2 | m3 | m4.

m1 <-> Taller(John, Maria) & Taller(John, Lara) & Taller(John, Sally).
m2 <-> Taller(Maria, John) & Taller(Maria, Lara) & Taller(Maria, Sally).
m3 <-> Taller(Lara, John) & Taller(Lara, Maria) & Taller(Lara, Sally).
m4 <-> Taller(Sally, John) & Taller(Sally, Maria) & Taller(Sally, Lara).


m1 -> -m2 & -m3 & -m4.
m2 -> -m1 & -m3 & -m4.
m3 -> -m1 & -m2 & -m4.
m4 -> -m1 & -m2 & -m3.


end_of_list.

formulas(goals).

end_of_list.
```

A.2.1.3 Joy is taller than Cassie. Joy is taller than Brian who is taller than Cassie and Geoff. Geoff is shorter than Val who is shorter than Cassie. Is Val the shortest?

```
assign(report_stderr, 2).
set(ignore_option_dependencies). % GUI handles dependencies

if(Prover9). % Options for Prover9
  assign(max_seconds, 60).
end_if.

if(Mace4). % Options for Mace4
  assign(domain_size, 5).
  assign(start_size, 5).
  assign(end_size, 5).
  assign(max_models, -1).
  assign(max_seconds, 60).
end_if.

formulas(assumptions).

Joy = 0.
Cassie = 1.
Brian = 2.
Geoff = 3.
Val = 4.

Short(Joy) | Short(Cassie) | Short(Brian) | Short(Geoff) | Short(Val).
Shorter(x,y) & Shorter(y,z) -> Shorter(x,z).
Shorter(x,y) -> -Short(y).

%Joy is taller than Cassie
Shorter(Cassie,Joy).

%Joy is taller than Brian who is taller than Cassie and Geoff
Shorter(Brian,Joy).
Shorter(Cassie,Brian).
Shorter(Geoff,Brian).

%Geoff is shorter than Val who is shorter than Cassie.
Shorter(Geoff,Val).
Shorter(Val,Cassie).

end_of_list.

formulas(goals).

%Is Val the shortest?

-Short(Val). %No
%Short(Geoff). %Geoff is the shortest

end_of_list.
```

A.2.1.4 Robert is taller than Jordan. Robert and Jordan are shorter than Lorraine. Robert is taller than Peter who is taller than Kylie. Jordan is taller than Peter. Who is the tallest?

```
formulas(assumptions).

Robert != Jordan.
Robert != Lorraine.
Robert != Peter.
Robert != Kylie.

Jordan != Lorraine.
Jordan != Peter.
Jordan != Kylie.

Lorraine != Peter.
Lorraine != Kylie.

Peter != Kylie.

Tall(Robert) | Tall(Jordan) | Tall(Lorraine) | Tall(Peter) | Tall(Kylie).
Taller(x,y) -> -Tall(y).
Taller(x,y) & Taller(y,z) -> Taller(x,z).

%Robert is taller than Jordan
Taller(Robert, Jordan).

%Robert and Jordan are shorter than Lorraine.
Taller(Lorraine, Robert).
Taller(Lorraine, Jordan).

%Robert is taller than Peter who is taller than Kylie.
Taller(Robert, Peter).
Taller(Peter, Kylie).

%Jordan is taller than Peter.
Taller(Jordan, Peter).

end_of_list.

formulas(goals).

end_of_list.
```

A.2.1.5 Luke is taller than Bianca who is taller than Nathan. Nathan is shorter than Cathy who is shorter than Bianca. Who is the shortest?

```
formulas(assumptions).
```

```
Luke != Bianca.
Luke != Nathan.
Luke != Cathy.

Bianca != Nathan.
Bianca != Cathy.

Nathan != Cathy.

Short(Luke) | Short(Bianca) | Short(Nathan) | Short(Cathy).
Shorter(x,y) -> -Short(y).
Shorter(x,y) & Shorter(y,z) -> Shorter(x,z).

%Luke is taller than Bianca who is taller than Nathan.
Shorter(Bianca, Luke).
Shorter(Nathan, Bianca).

%Nathan is shorter than Cathy who is shorter than Bianca.
Shorter(Nathan, Cathy).
Shorter(Cathy, Bianca).

end_of_list.

formulas(goals).

end_of_list.
```

A.2.1.6 Danny is taller than Lisa who is shorter than Elise who is shorter than Danny. Who is the shortest?

```
assign(report_stderr, 2).
set(ignore_option_dependencies). % GUI handles dependencies

if(Prover9). % Options for Prover9
  assign(max_seconds, 60).
end_if.

if(Mace4). % Options for Mace4
  assign(domain_size, 3).
  assign(start_size, 3).
  assign(end_size, 3).
  assign(max_seconds, 60).
end_if.

formulas(assumptions).

Danny = 0.
Lisa = 1.
Elise = 2.
```

```
Short(Danny) | Short(Lisa) | Short(Elise).
Shorter(x,y) & Shorter(y,z) -> Shorter(x,z).
Shorter(x,y) -> -Short(y).

%Danny is taller than Lisa who is shorter than Elise who is shorter than
   ↪ Danny.
Shorter(Lisa,Danny).
Shorter(Lisa,Elise).
Shorter(Elise,Danny).

end_of_list.

formulas(goals).

% Who is the shortest?
Short(Lisa).

end_of_list.
```

A.2.1.7 Brett is taller than Shannon who is taller than Rene. Brett is also taller than Mel. Who is the shortest?

```
formulas(assumptions).

Brett != Shannon.
Brett != Rene.
Brett != Mel.

Shannon != Rene.
Shannon != Mel.

Rene != Mel.

Short(Brett) | Short(Shannon) | Short(Rene) | Short(Mel).
Shorter(x,y) -> -Short(y).
Shorter(x,y) & Shorter(y,z) -> Shorter(x,z).


%Brett is taller than Shannon who is taller than Rene.
Shorter(Shannon, Brett).
Shorter(Rene, Shannon).

%Brett is also taller than Mel.
Shorter(Mel, Brett).
s
end_of_list.

formulas(goals).
```

```
end_of_list.
```

A.2.1.8 Sophie is shorter than Harry. Will is shorter than Mike who is shorter than Harry. Harry and Will are shorter than Fiona. Who is the tallest?

```
assign(report_stderr, 2).
set(ignore_option_dependencies). % GUI handles dependencies

if(Prover9). % Options for Prover9
  assign(max_seconds, 60).
end_if.

if(Mace4). % Options for Mace4
  assign(domain_size, 5).
  assign(start_size, 5).
  assign(end_size, 5).
  assign(max_seconds, 60).
end_if.

formulas(assumptions).

Sophie = 0.
Harry = 1.
Will = 2.
Mike = 3.
Fiona = 4.

Tall(Sophie) | Tall(Harry) | Tall(Will) | Tall(Mike) | Tall(Fiona).
Taller(x,y) & Taller(y,z) -> Taller(x,z).
Taller(x,y) -> -Tall(y).

%Sophie is shorter than Harry.
Taller(Harry,Sophie).

% Will is shorter than Mike who is shorter than Harry.
Taller(Mike,Will).
Taller(Harry,Mike).

%Harry and Will are shorter than Fiona.
Taller(Fiona,Harry).
Taller(Fiona,Will).

end_of_list.

formulas(goals).

%Who is the tallest?
Tall(Fiona).
```

```
end_of_list.
```

A.2.1.9 Tom is shorter than Matt and Tamara. Matt is shorter than Ben. Tamara is taller than Ben. Who is the tallest?

```
formulas(assumptions).

Tom != Matt.
Tom != Tamara.
Tom != Ben.

Matt != Tamara.
Matt != Ben.

Tamara != Ben.

Tall(Tom) | Tall(Matt) | Tall(Tamara) | Tall(Ben).
Taller(x,y) -> -Tall(y).
Taller(x,y) & Taller(y,z) -> Taller(x,z).


%Tom is shorter than Matt and Tamara
Taller(Matt, Tom).
Taller(Tamara, Tom).

%Matt is shorter than Ben.
Taller(Ben, Matt).

%Tamara is taller than Ben.
Taller(Tamara, Ben).


%Tamara

end_of_list.

formulas(goals).

end_of_list.
```

A.2.1.10 Jen is taller than Sue who is taller than Dick. Alan is taller than Dick. Jen and Sue are shorter than Alan. Betty is shorter than Dick. Who is the tallest?

```
assign(report_stderr, 2).
set(ignore_option_dependencies). % GUI handles dependencies
```

```
if(Prover9). % Options for Prover9
  assign(max_seconds, 60).
end_if.

if(Mace4). % Options for Mace4
  assign(domain_size, 5).
  assign(start_size, 5).
  assign(end_size, 5).
  assign(max_seconds, 60).
end_if.

formulas(assumptions).

Jen = 0.
Sue = 1.
Dick = 2.
Alan = 3.
Betty = 4.

Tall(Jen) | Tall(Sue) | Tall(Dick) | Tall(Alan) | Tall(Betty).
Taller(x,y) & Taller(y,z) -> Taller(x,z).
Taller(x,y) -> -Tall(y).

% Jen is taller than Sue who is taller than Dick.
Taller(Jen,Sue).
Taller(Sue,Dick).

% Alan is taller than Dick.
Taller(Alan,Dick).

% Jen and Sue are shorter than Alan.
Taller(Alan,Jen).
Taller(Alan,Sue).

% Betty is shorter than Dick.
Taller(Dick,Betty).

end_of_list.

formulas(goals).

% Who is the tallest?
Tall(Alan).

end_of_list.
```

A.2.2 Second Problem: Youngest or eldest?

A.2.2.1 Harmony is older than Connor and younger than Maranda. Isabelle is younger than Edwardo and Maranda. Edwardo is older than Maranda. If Harmony and Isabelle are the same age, who's youngest?

```
assign(report_stderr, 2).
set(ignore_option_dependencies). % GUI handles dependencies

if(Prover9). % Options for Prover9
  assign(max_seconds, 60).
end_if.

if(Mace4). % Options for Mace4
  assign(domain_size, 5).
  assign(start_size, 5).
  assign(end_size, 5).
  assign(max_models, -1).
  assign(max_seconds, 60).
end_if.

formulas(assumptions).

Harmony = 0.
Connor = 1.
Maranda = 2.
Isabelle = 3.
Edwardo = 4.

Youngest(Harmony) | Youngest(Connor) | Youngest(Maranda) | Youngest(Isabelle)
    ↪ | Youngest(Edwardo).
Younger(x,y) -> -Youngest(y).
Younger(x,y) & Younger(y,z) -> Younger(x,z).

%Harmony is older than Connor and younger than Maranda.
Younger(Connor,Harmony).
Younger(Harmony,Maranda).

%Isabelle is younger than Edwardo and Maranda.
Younger(Isabelle,Edwardo).
Younger(Isabelle,Maranda).

%Edwardo is older than Maranda.
Younger(Maranda,Edwardo).

%If Harmony and Isabelle are the same age,
Younger(Connor,Isabelle).
Younger(Isabelle,Maranda).
Younger(Harmony,Edwardo).
Younger(Harmony,Maranda).
```

```
end_of_list.

formulas(goals).

%who's youngest?
Youngest(Connor).

end_of_list.
```

A.2.2.2 Ontario is older than Waryn. Waryn is younger than Anastasia and younger than Desmond. Desmond is older than Ontario and Alonso, who are older than Anastasia. If Alonso and Ontario are twins, who's oldest?

```
formulas(assumptions).

Ontario != Waryn.
Ontario != Anastasia.
Ontario != Desmond.
Ontario != Alonso.

Waryn != Anastasia.
Waryn != Desmond.
Waryn != Alonso.

Anastasia != Desmond.
Anastasia != Alonso.

Desmond != Alonso.

Old(Ontario) | Old(Waryn) | Old(Anastasia) | Old(Desmond) | Old(Alonso).
Older(x,y) -> -Old(y).
Older(x,y) & Older(y,z) -> Older(x,z).


%Ontario is older than Waryn.
Older(Ontario, Waryn).

%Waryn is younger than Anastasia and younger than Desmond.
Older(Anastasia, Waryn).
Older(Desmond, Waryn).

%Desmond is older than Ontario and Alonso, who are older than Anastasia.
Older(Desmond, Ontario).
Older(Desmond, Alonso).
Older(Ontario, Anastasia).
Older(Alonso, Anastasia).

%Alonso and Ontario are twins
```

```
-Older(Alonso, Ontario) & -Older(Ontario, Alonso).


end_of_list.

formulas(goals).

end_of_list.
```

A.2.2.3 Quentyn is older than Porfirio and younger than Caitlyn. Porfirio is younger than Joslyn and younger than Caitlyn. If Quentyn Is older than Joslyn, who's oldest?

```
assign(report_stderr, 2).
set(ignore_option_dependencies). % GUI handles dependencies

if(Prover9). % Options for Prover9
  assign(max_seconds, 60).
end_if.

if(Mace4). % Options for Mace4
  assign(domain_size, 4).
  assign(start_size, 4).
  assign(end_size, 4).
  assign(max_models, -1).
  assign(max_seconds, 60).
end_if.

formulas(assumptions).

Quentyn= 0.
Porfirio = 1.
Caitlyn = 2.
Joslyn = 3.

Oldest(Quentyn) | Oldest(Porfirio) | Oldest(Caitlyn) | Oldest(Joslyn).
Older(x,y) -> -Oldest(y).
Older(x,y) & Older(y,z) -> Older(x,z).

%Quentyn is older than Porfirio and younger than Caitlyn.
Older(Quentyn,Porfirio).
Older(Caitlyn,Quentyn).

%Porfirio is younger than Joslyn and younger than Caitlyn.
Older(Joslyn,Porfirio).
Older(Caitlyn,Porfirio).

%If Quentyn Is older than Joslyn,
Older(Quentyn,Joslyn).
```

```
end_of_list.

formulas(goals).

%who's oldest?
Oldest(Caitlyn).

end_of_list.
```

A.2.2.4 Brendyn is older than Liam and Fergus. Marcell is older than Fergus and younger than Liam. If Liam has a brother named Lucian, who is older than Brendyn, who's youngest?

```
formulas(assumptions).

Brendyn != Liam.
Brendyn != Fergus.
Brendyn != Marcell.
Brendyn != Lucian.

Liam != Fergus.
Liam != Marcell.
Liam != Lucian.

Fergus != Desmond.
Fergus != Lucian.

Marcell != Lucian.


Young(Brendyn) | Young(Liam) | Young(Fergus) | Young(Marcell) | Young(Lucian).
Younger(x,y) -> -Young(y).
Younger(x,y) & Younger(y,z) -> Younger(x,z).


%Brendyn is older than Liam and Fergus.
Younger(Liam, Brendyn).
Younger(Fergus, Brendyn).

%Marcell is older than Fergus and younger than Liam.
Younger(Fergus, Marcell).
Younger(Marcell, Liam).

%Liam has a brother named Lucian
Younger(Brendyn, Lucian).


end_of_list.
```

```
formulas(goals).

end_of_list.
```

A.2.2.5 Cheveyo and Ethan are older than Devlyn and Victoria. Maria is older than Victoria, but younger than both Ethan and Devlyn. Cheveyo is older than Maria. If he is also younger than Ethan, who's youngest?

```
assign(report_stderr, 2).
set(ignore_option_dependencies). % GUI handles dependencies

if(Prover9). % Options for Prover9
  assign(max_seconds, 60).
end_if.

if(Mace4). % Options for Mace4
  assign(domain_size, 5).
  assign(start_size, 5).
  assign(end_size, 5).
  assign(max_models, -1).
  assign(max_seconds, 60).
end_if.

formulas(assumptions).

Cheveyo = 0.
Ethan = 1.
Devlyn = 2.
Victoria = 3.
Maria = 4.

Youngest(Cheveyo) | Youngest(Ethan) | Youngest(Devlyn) | Youngest(Victoria) |
   ↪ Youngest(Maria).
Younger(x,y) -> -Youngest(y).
Younger(x,y) & Younger(y,z) -> Younger(x,z).

%Cheveyo and Ethan are older than Devlyn and Victoria.
Younger(Cheveyo,Devlyn).
Younger(Victoria,Devlyn).
Younger(Cheveyo,Ethan).
Younger(Victoria,Ethan).

%Maria is older than Victoria, but younger than both Ethan and Devlyn.
Younger(Victoria,Maria).
Younger(Ethan,Maria).
Younger(Devlyn,Maria).

%Cheveyo is older than Maria.
```

```
Younger(Maria,Cheveyo).

%If he is also younger than Ethan,
Younger(Ethan,Cheveyo).

end_of_list.

formulas(goals).

% who's youngest?
Youngest(Victoria).

end_of_list.
```

A.2.2.6 Minnie is older than Justin and Tiffany. Jaymee is younger than Justin and older than Tiffany. Georgette is younger than Minnie and older than Tiffany. If Jaymee and Georgette are the same age, who's oldest?

```
formulas(assumptions).

Minnie != Justin.
Minnie != Tiffany.
Minnie != Jaymee.
Minnie != Georgette.

Justin != Tiffany.
Justin != Jaymee.
Justin != Georgette.

Tiffany != Jaymee.
Tiffany != Georgette.

Jaymee != Georgette.



Old(Minnie) | Old(Justin) | Old(Tiffany) | Old(Jaymee) | Old(Georgette).
Older(x,y) -> -Old(y).
Older(x,y) & Older(y,z) -> Older(x,z).


%Minnie is older than Justin and Tiffany.
Older(Minnie, Justin).
Older(Minnie, Tiffany).

%Jaymee is younger than Justin and older than Tiffany.
Older(Justin, Jaymee).
Older(Jaymee, Tiffany).
```

```
%Georgette is younger than Minnie and older than Tiffany.
Older(Minnie, Georgette).
Older(Georgette, Tiffany).

%Jaymee and Georgette are the same age
-Older(Jaymee, Georgette) & -Older(Georgette, Jaymee).

%Minnie

end_of_list.

formulas(goals).

end_of_list.
```

A.2.2.7 Elysha is older than Logan and younger than Edward. Jo is older than Michael. Edy is older than Michael and younger than Jo. Michael is older than Edward and Elysha. If Edward and Elysha are brother and sister, and Edy is their father, who's youngest?

```
assign(report_stderr, 2).
set(ignore_option_dependencies). % GUI handles dependencies

if(Prover9). % Options for Prover9
  assign(max_seconds, 60).
end_if.

if(Mace4). % Options for Mace4
  assign(domain_size, 6).
  assign(start_size, 6).
  assign(end_size, 6).
  assign(max_models, -1).
  assign(max_seconds, 60).
end_if.

formulas(assumptions).

Elysha = 0.
Logan = 1.
Edward = 2.
Jo = 3.
Michael = 4.
Edy = 5.

Youngest(Elysha) | Youngest(Logan) | Youngest(Edward) | Youngest(Jo) |
   ↪ Youngest(Edy) | Youngest(Michael).
Younger(x,y) -> -Youngest(y).
Younger(x,y) & Younger(y,z) -> Younger(x,z).
```

```
%Elysha is older than Logan and younger than Edward.
Younger(Logan,Elysha).
Younger(Elysha,Edward).

%Jo is older than Michael.
Younger(Michael,Jo).

%Edy is older than Michael and younger than Jo.
Younger(Michael,Edy).
Younger(Edy,Jo).

%Michael is older than Edward and Elysha.
Younger(Edward,Michael).
Younger(Elysha,Michael).

%If Edward and Elysha are brother and sister, and Edy is their father,
Younger(Edward,Edy).
Younger(Elysha,Edy).

end_of_list.

formulas(goals).

% who's youngest?
Youngest(Logan).

end_of_list.
```

A.2.2.8 Chris is younger than Ruth and Sam, but older than Stephanie. Ruth is older than Stephanie and younger than Curtis. Curtis is younger than Sam and older than Chris. If Chris has a brother named Brian who's younger than Stephanie, who's oldest?

```
formulas(assumptions).

Chris != Ruth.
Chris != Sam.
Chris != Stephanie.
Chris != Curtis.
Chris != Brian.

Ruth != Sam.
Ruth != Curtis.
Ruth != Stephanie.
Ruth != Brian.

Sam != Curtis.
Sam != Stephanie.
Sam != Brian.
```

```
Stephanie != Curtis.
Stephanie != Brian.

Curtis != Brian.


Old(Chris) | Old(Ruth) | Old(Sam) | Old(Stephanie) | Old(Curtis) | Old(Brian).
Older(x,y) -> -Old(y).
Older(x,y) & Older(y,z) -> Older(x,z).


%Chris is younger than Ruth and Sam, but older than Stephanie.
Older(Ruth, Chris).
Older(Sam, Chris).
Older(Chris, Stephanie).

%Ruth is older than Stephanie and younger than Curtis.
Older(Ruth, Stephanie).
Older(Curtis, Ruth).

%Curtis is younger than Sam and older than Chris.
Older(Sam, Curtis).
Older(Curtis, Chris).

%Brian who's younger than Stephanie
Older(Stephanie, Brian).

%Sam

end_of_list.

formulas(goals).

end_of_list.
```

A.2.2.9 Roshontia and Eric are older than Justine. Jadira is older than Colleen and Roshontia. If Justine is older than Colleen, that makes her oldest.

```
assign(report_stderr, 2).
set(ignore_option_dependencies). % GUI handles dependencies

if(Prover9). % Options for Prover9
  assign(max_seconds, 60).
end_if.

if(Mace4). % Options for Mace4
  assign(domain_size, 5).
  assign(start_size, 5).
  assign(end_size, 5).
  assign(max_models, -1).
  assign(max_seconds, 60).
end_if.

formulas(assumptions).

Roshontia = 0.
Eric = 1.
Justine = 2.
Jadira = 3.
Colleen = 4.

Oldest(Roshontia) | Oldest(Eric) | Oldest(Justine) | Oldest(Jadira) | Oldest(
    ↪ Colleen).
Older(x,y) -> -Oldest(y).
Older(x,y) & Older(y,z) -> Older(x,z).

%Roshontia and Eric are older than Justine.
Older(Roshontia,Justine).
Older(Eric,Justine).

%Jadira is older than Colleen and Roshontia.
Older(Jadira,Colleen).
Older(Jadira,Roshontia).

%If Justine is older than Colleen,
Older(Justine,colleen).

%that makes her oldest.
-Oldest(Justine).
%There are still valid models even if Justine is not the oldest, so she may
    ↪ not be the oldest.
%additional information is needed to prove who is the oldest.

end_of_list.

formulas(goals).

end_of_list.
```

A.2.2.10 Preston is older than Corissa and younger than Pedro. Corissa is older than Andrea and Madison. Andrea and Madison are older than Chloe. Andrea is younger than Caspian and older than Madison. Pedro is older than Corissa and Madison, but younger than Caspian. If Preston is Caspian's son, who's youngest?

```
formulas(assumptions).
```

```
Preston != Corissa.
Preston != Pedro.
Preston != Andrea.
Preston != Madison.
Preston != Chloe.
Preston != Caspian.

Corissa != Pedro.
Corissa != Andrea.
Corissa != Madison.
Corissa != Chloe.
Corissa != Caspian.

Pedro != Andrea.
Pedro != Madison.
Pedro != Chloe.
Pedro != Caspian.

Andrea != Madison.
Andrea != Chloe.
Andrea != Caspian.

Madison != Chloe.
Madison != Caspian.

Chloe != Caspian.



Young(Preston) | Young(Corissa) | Young(Pedro) | Young(Andrea) | Young(Madison
   ↪ ) | Young(Chloe) | Young(Caspian).
Younger(x,y) -> -Young(y).
Younger(x,y) & Younger(y,z) -> Younger(x,z).


%Preston is older than Corissa and younger than Pedro.
Younger(Corissa, Preston).
Younger(Preston, Pedro).

% Corissa is older than Andrea and Madison.
Younger(Andrea, Corissa).
Younger(Madison, Corissa).

%Andrea and Madison are older than Chloe.
Younger(Chloe, Andrea).
Younger(Chloe, Madison).

%Andrea is younger than Caspian and older than Madison.
Younger(Andrea, Caspian).
Younger(Madison, Andrea).
```

```

%Pedro is older than Corissa and Madison, but younger than Caspian.
Younger(Corissa, Pedro).
Younger(Madison, Pedro).
Younger(Pedro, Caspian).

%Preston is Caspian's son
Younger(Preston, Caspian).

%Chloe.


end_of_list.

formulas(goals).

end_of_list.
```

A.2.3 Third Problem: Wide Awake

```
assign(report_stderr, 2).
set(ignore_option_dependencies). % GUI handles dependencies

if(Prover9). % Options for Prover9
  assign(max_seconds, 60).
end_if.

if(Mace4). % Options for Mace4
  assign(domain_size, 5).
  assign(start_size, 5).
  assign(end_size, 5).
  assign(max_models, -1).
  assign(max_seconds, 60).
end_if.

formulas(assumptions).

Barbara = 0.
Carla = 1.
Edna = 2.
Fiona = 3.
Mary = 4.

Exercise(Barbara) | Exercise(Carla) | Exercise(Edna) | Exercise(Fiona) |
   ↪ Exercise(Mary).
Coffee(Barbara) | Coffee(Carla) | Coffee(Edna) | Coffee(Fiona) | Coffee(Mary).
Worry(Barbara) | Worry(Carla) | Worry(Edna) | Worry(Fiona) | Worry(Mary).
Hunger(Barbara) | Hunger(Carla) | Hunger(Edna) | Hunger(Fiona) | Hunger(Mary).
```

```
Snoring(Barbara) | Snoring(Carla) | Snoring(Edna) | Snoring(Fiona) | Snoring(
    ↪ Mary).

Exercise(x) -> -Coffee(x) & -Worry(x) & -Hunger(x) & -Snoring(x).
Coffee(x) -> -Exercise(x) & -Worry(x) & -Hunger(x) & -Snoring(x).
Worry(x) -> -Coffee(x) & -Exercise(x) & -Hunger(x) & -Snoring(x).
Hunger(x) -> -Coffee(x) & -Worry(x) & -Exercise(x) & -Snoring(x).
Snoring(x) -> -Coffee(x) & -Worry(x) & -Hunger(x) & -Exercise(x).

Called_12am(Barbara) | Called_12am(Carla) | Called_12am(Edna) | Called_12am(
    ↪ Fiona) | Called_12am(Mary).
Called_1am(Barbara) | Called_1am(Carla) | Called_1am(Edna) | Called_1am(Fiona)
    ↪  | Called_1am(Mary).
Called_2am(Barbara) | Called_2am(Carla) | Called_2am(Edna) | Called_2am(Fiona)
    ↪  | Called_2am(Mary).
Called_3am(Barbara) | Called_3am(Carla) | Called_3am(Edna) | Called_3am(Fiona)
    ↪  | Called_3am(Mary).
Called_4am(Barbara) | Called_4am(Carla) | Called_4am(Edna) | Called_4am(Fiona)
    ↪  | Called_4am(Mary).

Called_12am(x) -> -Called_1am(x) & -Called_2am(x) & -Called_3am(x) & -
    ↪ Called_4am(x).
Called_1am(x) -> -Called_12am(x) & -Called_2am(x) & -Called_3am(x) & -
    ↪ Called_4am(x).
Called_2am(x) -> -Called_1am(x) & -Called_12am(x) & -Called_3am(x) & -
    ↪ Called_4am(x).
Called_3am(x) -> -Called_1am(x) & -Called_2am(x) & -Called_12am(x) & -
    ↪ Called_4am(x).
Called_4am(x) -> -Called_1am(x) & -Called_2am(x) & -Called_3am(x) & -
    ↪ Called_12am(x).

%A - "I was so busy today," said one caller, "that I couldn't do my
    ↪ weightlifting until after supper, and now I'm wired." She told me this
    ↪  some time before Fiona called.
-Exercise(Fiona).
Exercise(x) -> ((Called_12am(x) & (Called_1am(Fiona) | Called_2am(Fiona) |
    ↪ Called_3am(Fiona) | Called_4am(Fiona))) | (Called_1am(x) & (Called_2am(
    ↪ Fiona) | Called_3am(Fiona) | Called_4am(Fiona))) | (Called_2am(x) & (
    ↪ Called_3am(Fiona) | Called_4am(Fiona))) | (Called_3am(x) & Called_4am(
    ↪ Fiona))).

%B - The late exerciser called at some point after the person who had drunk
    ↪ ten cups of coffee that evening and was suffering from caffeine
    ↪ overload.
(Exercise(x) & Coffee(y)) -> ((Called_12am(y) & (Called_1am(x) | Called_2am(x)
    ↪  | Called_3am(x) | Called_4am(x))) | (Called_1am(y) & (Called_2am(x) |
    ↪ Called_3am(x) | Called_4am(x))) | (Called_2am(y) & (Called_3am(x) |
    ↪ Called_4am(x))) | (Called_3am(y) & Called_4am(x))).

%C - Carla called first.
```

```
Called_12am(Carla).

%D - Carla is not the one dieting to lose weight and too famished to sleep.
-Hunger(Carla).
-Exercise(Carla).

%E - Mary called at some point before the coffee-drinker and also at some
    ↪ point before the person suffering from worry (about an upcoming trial)
    ↪ .
-Coffee(Mary).
-Worry(Mary).
-Called_3am(Mary).
-Called_4am(Mary).
Coffee(x) -> ((Called_12am(Mary) & (Called_1am(x) | Called_2am(x) | Called_3am
    ↪ (x) | Called_4am(x))) | (Called_1am(Mary) & (Called_2am(x) | Called_3am(
    ↪ x) | Called_4am(x))) | (Called_2am(Mary) & (Called_3am(x) | Called_4am(x
    ↪ )))).
Worry(x) -> ((Called_12am(Mary) & (Called_1am(x) | Called_2am(x) | Called_3am(
    ↪ x) | Called_4am(x))) | (Called_1am(Mary) & (Called_2am(x) | Called_3am(x
    ↪ ) | Called_4am(x))) | (Called_2am(Mary) & (Called_3am(x) | Called_4am(x)
    ↪ ))).

%F - Edna, a Mormon, avoids all stimulant beverages.
-Coffee(Edna).

%To find out who called first, Fiona or Marry
%Mary called first.
m1 <-> ((Called_12am(Mary) & (Called_1am(Fiona) | Called_2am(Fiona) |
    ↪ Called_3am(Fiona) | Called_4am(Fiona))) | (Called_1am(Mary) & (
    ↪ Called_2am(Fiona) | Called_3am(Fiona) | Called_4am(Fiona))) | (
    ↪ Called_2am(Mary) & (Called_3am(Fiona) | Called_4am(Fiona))) | (
    ↪ Called_3am(Mary) & Called_4am(Fiona))).

end_of_list.

formulas(goals).

%1.Who called at midnight?
%Called_12am(Carla).

%2. Who called earlier, Mary or Fiona?
%m1.

%3. Who called me at 1:00 a.m.?
%Called_1am(Mary).

%4. Why could Fiona not sleep the other night?
%Worry(Fiona).

%5. And why could Carla not sleep the other night?
```

```
%Snoring(Carla).

%6. Now who was kept awake by hunger?
%Hunger(Mary).

%7. Who drank too much coffee and called me in a caffeine buzz?
%Coffee(Barbara).

%8. So, who rang at 2:00 a.m.?
%Called_2am(Barbara).

%9. And who called me at 3:00 a.m.?
%Called_3am(Edna).

%10. And lastly, who called at 4:00 a.m. (the final caller of the night)?
Called_4am(Fiona).

end_of_list.
```

A.2.4 Fourth Problem: Just Married

```
formulas(assumptions).



Ben != Todd.
Ben != Brandon.
Ben != Buddy.
Ben != Jack.
Ben != Lucy.
Ben != Emma.
Ben != Tabitha.
Ben != Brandy.
Ben != Ann.

Todd != Brandon.
Todd != Buddy.
Todd != Jack.
Todd != Lucy.
Todd != Emma.
Todd != Tabitha.
Todd != Brandy.
Todd != Ann.

Brandon != Buddy.
Brandon != Jack.
Brandon != Lucy.
Brandon != Emma.
Brandon != Tabitha.
Brandon != Brandy.
Brandon != Ann.

Buddy != Jack.
Buddy != Lucy.
Buddy != Emma.
Buddy != Tabitha.
Buddy != Brandy.
Buddy != Ann.

Jack != Lucy.
Jack != Emma.
Jack != Tabitha.
Jack != Brandy.
Jack != Ann.

Lucy != Emma.
Lucy != Tabitha.
Lucy != Brandy.
Lucy != Ann.

Emma != Tabitha.
Emma != Brandy.
Emma != Ann.

Tabitha != Brandy.
Tabitha != Ann.

Brandy != Ann.


Married(Ben, Lucy) | Married(Ben, Emma) | Married(Ben, Tabitha) | Married(Ben,
   ↪  Brandy) | Married(Ben, Ann).
Married(Todd, Lucy) | Married(Todd, Emma) | Married(Todd, Tabitha) | Married(
   ↪ Todd, Brandy) | Married(Todd, Ann).
Married(Brandon, Lucy) | Married(Brandon, Emma) | Married(Brandon, Tabitha) |
   ↪ Married(Brandon, Brandy) | Married(Brandon, Ann).
Married(Buddy, Lucy) | Married(Buddy, Emma) | Married(Buddy, Tabitha) |
   ↪ Married(Buddy, Brandy) | Married(Buddy, Ann).
Married(Jack, Lucy) | Married(Jack, Emma) | Married(Jack, Tabitha) | Married(
   ↪ Jack, Brandy) | Married(Jack, Ann).

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

Married(Ben, Lucy) -> -Married(Todd, Lucy) & -Married(Brandon, Lucy) & -
   ↪ Married(Buddy, Lucy) & -Married(Jack, Lucy).
Married(Ben, Emma) -> -Married(Todd, Emma) & -Married(Brandon, Emma) & -
   ↪ Married(Buddy, Emma) & -Married(Jack, Emma).
Married(Ben, Tabitha) -> -Married(Todd, Tabitha) & -Married(Brandon, Tabitha)
   ↪ & -Married(Buddy, Tabitha) & -Married(Jack, Tabitha).
```

```
Married(Ben, Brandy) -> -Married(Todd, Brandy) & -Married(Brandon, Brandy) & -
    ↪ Married(Buddy, Brandy) & -Married(Jack, Brandy).
Married(Ben, Ann) -> -Married(Todd, Ann) & -Married(Brandon, Ann) & -Married(
    ↪ Buddy, Ann) & -Married(Jack, Ann).

Married(Todd, Lucy) -> -Married(Ben, Lucy) & -Married(Brandon, Lucy) & -
    ↪ Married(Buddy, Lucy) & -Married(Jack, Lucy).
Married(Todd, Emma) -> -Married(Ben, Emma) & -Married(Brandon, Emma) & -
    ↪ Married(Buddy, Emma) & -Married(Jack, Emma).
Married(Todd, Tabitha) -> -Married(Ben, Tabitha) & -Married(Brandon, Tabitha)
    ↪ & -Married(Buddy, Tabitha) & -Married(Jack, Tabitha).
Married(Todd, Brandy) -> -Married(Ben, Brandy) & -Married(Brandon, Brandy) & -
    ↪ Married(Buddy, Brandy) & -Married(Jack, Brandy).
Married(Todd, Ann) -> -Married(Ben, Ann) & -Married(Brandon, Ann) & -Married(
    ↪ Buddy, Ann) & -Married(Jack, Ann).

Married(Brandon, Lucy) -> -Married(Ben, Lucy) & -Married(Todd, Lucy) & -
    ↪ Married(Buddy, Lucy) & -Married(Jack, Lucy).
Married(Brandon, Emma) -> -Married(Ben, Emma) & -Married(Todd, Emma) & -
    ↪ Married(Buddy, Emma) & -Married(Jack, Emma).
Married(Brandon, Tabitha) -> -Married(Ben, Tabitha) & -Married(Todd, Tabitha)
    ↪ & -Married(Buddy, Tabitha) & -Married(Jack, Tabitha).
Married(Brandon, Brandy) -> -Married(Ben, Brandy) & -Married(Todd, Brandy) & -
    ↪ Married(Buddy, Brandy) & -Married(Jack, Brandy).
Married(Brandon, Ann) -> -Married(Ben, Ann) & -Married(Todd, Ann) & -Married(
    ↪ Buddy, Ann) & -Married(Jack, Ann).

Married(Buddy, Lucy) -> -Married(Ben, Lucy) & -Married(Todd, Lucy) & -Married(
    ↪ Brandon, Lucy) & -Married(Jack, Lucy).
Married(Buddy, Emma) -> -Married(Ben, Emma) & -Married(Todd, Emma) & -Married(
    ↪ Brandon, Emma) & -Married(Jack, Emma).
Married(Buddy, Tabitha) -> -Married(Ben, Tabitha) & -Married(Todd, Tabitha) &
    ↪ -Married(Brandon, Tabitha) & -Married(Jack, Tabitha).
Married(Buddy, Brandy) -> -Married(Ben, Brandy) & -Married(Todd, Brandy) & -
    ↪ Married(Brandon, Brandy) & -Married(Jack, Brandy).
Married(Buddy, Ann) -> -Married(Ben, Ann) & -Married(Todd, Ann) & -Married(
    ↪ Brandon, Ann) & -Married(Jack, Ann).

Married(Jack, Lucy) -> -Married(Ben, Lucy) & -Married(Todd, Lucy) & -Married(
    ↪ Brandon, Lucy) & -Married(Buddy, Lucy).
Married(Jack, Emma) -> -Married(Ben, Emma) & -Married(Todd, Emma) & -Married(
    ↪ Brandon, Emma) & -Married(Buddy, Emma).
Married(Jack, Tabitha) -> -Married(Ben, Tabitha) & -Married(Todd, Tabitha) & -
    ↪ Married(Brandon, Tabitha) & -Married(Buddy, Tabitha).
Married(Jack, Brandy) -> -Married(Ben, Brandy) & -Married(Todd, Brandy) & -
    ↪ Married(Brandon, Brandy) & -Married(Buddy, Brandy).
Married(Jack, Ann) -> -Married(Ben, Ann) & -Married(Todd, Ann) & -Married(
    ↪ Brandon, Ann) & -Married(Buddy, Ann).
```

```
-Married(Ben, Todd) & -Married(Ben, Brandon) & -Married(Ben, Buddy) & -Married
    ↪ (Ben, Jack).
-Married(Todd, Brandon) & -Married(Todd, Buddy) & -Married(Todd, Jack).
-Married(Brandon, Buddy) & -Married(Brandon, Jack).
-Married(Buddy, Jack).

-Married(Lucy, Emma) & -Married(Lucy, Tabitha) & -Married(Lucy, Brandy) & -
    ↪ Married(Lucy, Ann).
-Married(Emma, Tabitha) & -Married(Emma, Brandy) & -Married(Emma, Ann).
-Married(Tabitha, Brandy) & -Married(Tabitha, Ann).
-Married(Brandy, Ann).


Rome(x) | Munich(x) | London(x) | Madrid(x) | Paris(x).

Rome(x) -> -Munich(x) & -London(x) & -Madrid(x) & -Paris(x).
Munich(x) -> -Rome(x) & -London(x) & -Madrid(x) & -Paris(x).
London(x) -> -Rome(x) & -Munich(x) & -Madrid(x) & -Paris(x).
Madrid(x) -> -Rome(x) & -Munich(x) & -London(x) & -Paris(x).
Paris(x) -> -Rome(x) & -Munich(x) & -London(x) & -Madrid(x).

Rome(x) & Married(x,y) -> Rome(y).
Munich(x) & Married(x,y) -> Munich(y).
London(x) & Married(x,y) -> London(y).
Madrid(x) & Married(x,y) -> Madrid(y).
Paris(x) & Married(x,y) -> Paris(y).

Rome(Ben) -> -Rome(Todd) & -Rome(Brandon) & -Rome(Buddy) & -Rome(Jack).
Rome(Todd) -> -Rome(Ben) & -Rome(Brandon) & -Rome(Buddy) & -Rome(Jack).
Rome(Brandon) -> -Rome(Todd) & -Rome(Ben) & -Rome(Buddy) & -Rome(Jack).
Rome(Buddy) -> -Rome(Todd) & -Rome(Brandon) & -Rome(Ben) & -Rome(Jack).
Rome(Jack) -> -Rome(Todd) & -Rome(Brandon) & -Rome(Buddy) & -Rome(Ben).

Munich(Ben) -> -Munich(Todd) & -Munich(Brandon) & -Munich(Buddy) & -Munich(
    ↪ Jack).
Munich(Todd) -> -Munich(Ben) & -Munich(Brandon) & -Munich(Buddy) & -Munich(
    ↪ Jack).
Munich(Brandon) -> -Munich(Todd) & -Munich(Ben) & -Munich(Buddy) & -Munich(
    ↪ Jack).
Munich(Buddy) -> -Munich(Todd) & -Munich(Brandon) & -Munich(Ben) & -Munich(
    ↪ Jack).
Munich(Jack) -> -Munich(Todd) & -Munich(Brandon) & -Munich(Buddy) & -Munich(
    ↪ Ben).

London(Ben) -> -London(Todd) & -London(Brandon) & -London(Buddy) & -London(
    ↪ Jack).
London(Todd) -> -London(Ben) & -London(Brandon) & -London(Buddy) & -London(
    ↪ Jack).
London(Brandon) -> -London(Todd) & -London(Ben) & -London(Buddy) & -London(
    ↪ Jack).
```

```
London(Buddy) -> -London(Todd) & -London(Brandon) & -London(Ben) & -London(
   ↪ Jack).
London(Jack) -> -London(Todd) & -London(Brandon) & -London(Buddy) & -London(
   ↪ Ben).

Madrid(Ben) -> -Madrid(Todd) & -Madrid(Brandon) & -Madrid(Buddy) & -Madrid(
   ↪ Jack).
Madrid(Todd) -> -Madrid(Ben) & -Madrid(Brandon) & -Madrid(Buddy) & -Madrid(
   ↪ Jack).
Madrid(Brandon) -> -Madrid(Todd) & -Madrid(Ben) & -Madrid(Buddy) & -Madrid(
   ↪ Jack).
Madrid(Buddy) -> -Madrid(Todd) & -Madrid(Brandon) & -Madrid(Ben) & -Madrid(
   ↪ Jack).
Madrid(Jack) -> -Madrid(Todd) & -Madrid(Brandon) & -Madrid(Buddy) & -Madrid(
   ↪ Ben).

Paris(Ben) -> -Paris(Todd) & -Paris(Brandon) & -Paris(Buddy) & -Paris(Jack).
Paris(Todd) -> -Paris(Ben) & -Paris(Brandon) & -Paris(Buddy) & -Paris(Jack).
Paris(Brandon) -> -Paris(Todd) & -Paris(Ben) & -Paris(Buddy) & -Paris(Jack).
Paris(Buddy) -> -Paris(Todd) & -Paris(Brandon) & -Paris(Ben) & -Paris(Jack).
Paris(Jack) -> -Paris(Todd) & -Paris(Brandon) & -Paris(Buddy) & -Paris(Ben).


%1. Ben's spouse did not go to Paris
(Married(Ben, Lucy) & -Paris(Lucy)) | (Married(Ben, Emma) & -Paris(Emma)) | (
   ↪ Married(Ben, Tabitha) & -Paris(Tabitha)) | (Married(Ben, Brandy) & -
   ↪ Paris(Brandy)) | (Married(Ben, Ann) & -Paris(Ann)).


%2. Emma is not married to Brandon and did not travel to London or Rome
-Married(Brandon, Emma) & -London(Emma) & -Rome(Emma).

%3. Jack visited Munich
Munich(Jack).

%4. Brandon and his new bride visited London
(Married(Brandon, Lucy) & London(Lucy)) | (Married(Brandon, Emma) & London(
   ↪ Emma)) | (Married(Brandon, Tabitha) & London(Tabitha)) | (Married(
   ↪ Brandon, Brandy) & London(Brandy)) | (Married(Brandon, Ann) & London(Ann
   ↪ )).

%5. Todd is married to Lucy
Married(Todd, Lucy).

%6. Ann is not married to Jack or Ben and did not go to Madrid
-Married(Jack, Ann) & -Married(Ben, Ann) & -Madrid(Ann).

%7. The new bride with the shortest name did not go to Paris
-Paris(Ann).
```

```
%8. Buddy is married to Brandy, they did not go to Rome or Paris
Married(Buddy, Brandy) & -Rome(Buddy) & -Rome(Brandy) & -Paris(Buddy) & -Paris
    ↪ (Brandy).


end_of_list.

formulas(goals).

end_of_list.
```