Knowledge Representation and Reasoning
Coursework II

Due Friday 27th December 2020

Brandon Bennett

This coursework counts for 30% of the total marks for this module.

Part A: Sequent Calculus (10 marks)

Give a *Gentzen Sequent Calculus* proof of the following sequents, using the rules provided below:[1]

(i) $H \wedge Q,\ G \ \vdash \ (\neg H \rightarrow P) \wedge (G \vee R)$ [5 marks]

(ii) $\forall x[\neg P(a,x)] \ \vdash \ \forall x[\exists y[\neg P(y,x)]]$ [5 marks]

Rules:

$$\frac{\textit{Axiom}}{\alpha,\ \Gamma \ \vdash \ \alpha,\ \Delta}$$

$$\frac{\Gamma,\ \neg\alpha \vee \beta \ \vdash \ \Delta}{\Gamma,\ \alpha \ \rightarrow \ \beta \ \vdash \ \Delta}\,[\rightarrow\vdash r.w.] \qquad \frac{\Gamma \ \vdash \ \neg\alpha \vee \beta,\ \Delta}{\Gamma \ \vdash \ \alpha \ \rightarrow \ \beta,\ \Delta}\,[\vdash\rightarrow r.w.]$$

$$\frac{\alpha,\ \beta,\ \Gamma \ \vdash \ \Delta}{(\alpha \wedge \beta),\ \Gamma \ \vdash \ \Delta}\,[\wedge\vdash] \qquad \frac{\Gamma \vdash \alpha,\ \Delta \quad \textit{and} \quad \Gamma \vdash \beta,\ \Delta}{\Gamma \ \vdash (\alpha \wedge \beta),\ \Delta}\,[\vdash\wedge]$$

$$\frac{\alpha, \Gamma \vdash \Delta \quad \textit{and} \quad \beta, \Gamma \vdash \Delta}{(\alpha \vee \beta),\ \Gamma \ \vdash \ \Delta}\,[\vee\vdash] \qquad \frac{\Gamma \ \vdash \ \alpha,\ \beta,\ \Delta}{\Gamma \ \vdash \ (\alpha \ \vee \ \beta),\ \Delta}\,[\vdash\vee]$$

$$\frac{\Gamma \ \vdash \ \alpha,\ \Delta}{\neg\alpha,\ \Gamma \ \vdash \ \Delta}\,[\neg\vdash] \qquad \frac{\Gamma,\ \alpha \ \vdash \ \Delta}{\Gamma \ \vdash \ \neg\alpha,\ \Delta}\,[\vdash\neg]$$

$$\frac{\forall x[\Phi(x)],\ \Phi(k),\ \Gamma \ \vdash \ \Delta}{\forall x[\Phi(x)],\ \Gamma \ \vdash \ \Delta}\,[\forall\vdash] \qquad \frac{\Gamma \ \vdash \ \Phi(k), \Delta}{\Gamma \ \vdash \ \forall x[\Phi(x)],\ \Delta}\,[\vdash\forall]\ \dagger$$

† where κ cannot occur anywhere in the lower sequent.

$$\frac{\neg\forall x[\neg\Phi(x)],\ \Gamma \ \vdash \ \Delta}{\exists x[\Phi(x)],\ \Gamma \ \vdash \ \Delta}\,[\exists\vdash r.w.] \qquad \frac{\Gamma \ \vdash \ \neg\forall x[\neg\Phi(x)],\ \Delta}{\Gamma \ \vdash \ \exists x[\Phi(x)],\ \Delta}\,[\vdash\exists r.w.]$$

[1] This rule set is essentially the same as the one presented in the lectures/slides, but for simplicity I am including the axiom and rewrite rules along with the other rules.

1 Part B: Axioms and Models (10 Marks)

Your aim in answering this question is to specify *models* satisfying a given axiom set in terms of a *Python data structure.*

I shall first present a set of axioms and explain how we can use graph diagrams to represent its models. I shall then show how these models can also be represented in Python.

Axioms Describing a Road Network

The following symbols and formulae describe the existence of cities, towns and villages, and the road connections between them, in an imaginary land:

Symbols Used:

- $C(x)$ — x is a city.
- $T(x)$ — x is a town.
- $V(x)$ — x is a village.
- $x > y$ — x is larger than y.
- $R(x, y)$ — There is a road between x and y.
- $x = y$ — x is the same thing as y (equality).

Axioms:

1. $R(x, y) \rightarrow (R(y, x) \wedge \neg(x = y))$
2. $\forall x \forall y[\ x > y \ \leftrightarrow \ ((C(x) \wedge (T(y) \vee V(y))) \vee (T(x) \wedge V(y)))]$
3. $\forall x[C(x) \vee T(x) \vee V(x)]$
4. $\exists x[C(x)]$
5. $\forall x \forall y[\ (C(x) \wedge C(y)) \rightarrow (x = y)\ ]$
6. $\exists x \exists y[\ T(x) \wedge T(y) \wedge \neg(x = y)\ ]$
7. $\forall x[C(x) \vee \exists y[\ y > x \wedge R(x, y)]\ ]$
8. $\forall x \forall y \forall z[\ (V(x) \wedge R(x, y) \wedge R(x, z)) \rightarrow y = z]$
9. $\forall x[T(x) \rightarrow \exists y_1 \exists y_2 \exists y_3[\ R(x, y_1) \wedge R(x, y_2) \wedge R(x, y_3) \wedge \ \neg(y_1 = y_2) \wedge \neg(y_1 = y_3) \wedge \neg(y_2 = y_3)\ ]$
10. $\forall x \forall y_1 \forall y_2 \forall y_3 \forall y_4[\ (R(x, y_1) \wedge R(x, y_2) \wedge R(x, y_3) \wedge R(x, y_4)) \ \rightarrow$
    $((y_1 = y_2) \vee (y_1 = y_3) \vee (y_1 = y_4) \vee (y_2 = y_3) \vee (y_2 = y_4) \vee (y_3 = y_4))\ ]$

B (i) Task

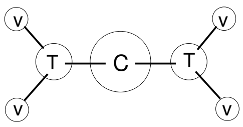
Translate each of the axioms 1-10 into an English sentence, in accordance with the given interpretation specification. [10 marks]

There is basically one mark per axiom that is correctly stated. However, up to 5 marks may be deducted for over-complex and unnatural sentences. Your answers do not need to be perfect English, but should not be just statements of the raw logical form, which no human would say.

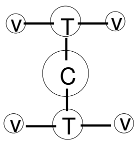
Graphical Representation of Models

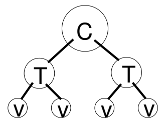
A model is a representation of a situation that enables one to determine whether any formula expressed using some particular predicates is true or false. The model must provide a way to interpret the predicates used in such formulae.

A model of the above axioms can be represented by a simple diagram, in which the entities and their types are represented by a circle labelled with letter (say `v` for villages `T` for towns and `C` for cities), and the road connections are represented by lines. The relative size of the places is represented by the size of the circles. For example, the following diagram represents a model, which does indeed satisfy the ten axioms given:



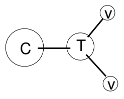
In a diagrammatic representation, there will usually be many different ways to represent the same model. For example, the two diagrams below are equivalent to the one given above. If you compare these three diagrams, you will see that they all contain the same number of each type of entity (city, town, village) and that the road connections between them are the same.

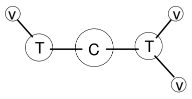


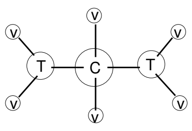


Some Non-Model Examples

It is important to understand that when we consider models of a *set of axioms*, these are models that satisfy ***all*** the axioms. For example, none of the following diagrams is a model of the axioms. Each of them fails to satisfy one of the given axioms (a different one in each case) but satisfies the other axioms.







Set Theoretic Representation of Models in Python

The more standard way that models are represented in mathematics and logic is by means of sets. Each entity is an atomic element; each predicate denotes a set of entities and each n-ary relation denotes a set of n-tuples.

It is actually very straightforward to represent models in Python, in a way which is almost the same as the standard set-theoretic representation. This is illustrated below, where a set of models is represented by a Python list, and each model is a dictionary, giving the domain of entities and the denotation of each non-logical symbol that occurs in the axioms that we are considering. Here, for clarity I am using the full words `City`, `Town`, `Village`, `Road`, corresponding to C, T, V and R in the first-order axioms. `DOMAIN` declares the domain of entities as a set of positive integers, and these occur in the denotations of the other symbols.

```
MODELS = [
  {
      "DOMAIN"  : {1,2,3,4,5,6,7},

      "City"    : {1},
      "Town"    : {2,3},
      "Village" : {4,5,6,7},

      "Road" : { (1,2), (2,1),
                 (1,3), (3,1),
                 (2,4), (4,2),
                 (2,5), (5,2),
                 (3,6), (6,3),
                 (3,7), (7,3),
               },

      ">" : { (1,2),(1,3),(1,4),(1,5),(1,6),(1,7),
              (2,4),(2,5),(2,6),(2,7),
              (3,4),(3,5),(3,6),(3,7),
            }
  },

  ## OTHER MODELS SHOULD BE ADDED HERE
]
```

B (ii) Task

Find all the models of the axiom set given above and represent them using Python in the format shown above. One correct model has already been given in the list, you need to add similar dictionary declarations representing each of the other non-equivalent models.

You will submit your Python file containing the list of models to a Gradescope automarker, which will check each model to determine your grade. [10 marks]

To gain full marks your list needs to contain all the possible models and you will lose marks for incorrect models or if you include more than one equivalent form of the same model.

Part C: Build Your Own Knowledge Base (30 marks)

For this part of the assignment you will create a *Knowledge Base* (KB) formulated as a *Prolog* program. As a starting point, you are given the program `brandons_kb.pl`, which contains an example knowledge base (expressed as a set of facts and a set of inference rules) as well as a simple but quite powerful inference mechanism. You can get to it via the following url:

https://swish.swi-prolog.org/p/brandons_kb.pl

You should examine the program and run it in SWISH (some other Prolog), before attempting the questions. In order to modify the code you need to copy it using the *fork* option in the SWISH "Save new version" dialogue, that you get to by choosing "Save ..." on the drop down "File" menu. You should rename your version to ***username*_kb.pl**.

Initial Experimentation

Before attempting this question, you are advised to do some initial experimentation with the given database and go through the following questions, which will help you think about how the system works:

1. Specify a rule which enforces the condition: "Cats hate all dogs except puppies".
   (Ensure that your rule gives the expected inferences.)

2. Specify a *relational composition* rule that will enable the implied fact
   `[rex, is_grandfather_of, champ]` to be inferred from the KB.

3. A programmer is considering adding the following "genesis" rule to the KB:
   ```
   rule( genesis, [[X, is, dog]] ==> [fatherof(X), is, dog] ).
   ```
   Add this rule to the KB and run the inference mechanism, before answering these questions:

   (a) What is the meaning/purpose of this rule?

   (b) Explain what happens when the inference system is run with this rule present and why it could be problematic.

4. Consider the function of the weak negation operator `\+` that can be applied to premisses of a rule. A weakly negated premiss takes the form `\+[r, ... ]` and means that the rule can only be applied if the fact `[r, ... ]` is not in the database. This can be useful but can lead to unforseen behaviour of the inference system when rule are reordered or further rules are added. Explain how this can happen.

Making your own Knowledge Base

To answer this question you must replace the simple example facts and rules with a more substantial KB of your own devising (you will probably want to keep the 'logic' rules for subclass reasoning).

Preparation: You must chose a domain for your KB. This should be focused enough so that it has a distinctive vocabulary of key concepts, relations and specific facts; but also substantial enough that moderately complex inference rules are required to draw relevant and interesting conclusions. I suggest you do a bit of experimentation with adding new information to the current KB before deciding on the domain your want to work on. Feel free to ask whether the domain seems suitable. Most domains should be fine, but I might possibly suggest narrowing or widening the domain.

C (i) Your Knowledge Base

Your answer to this question will be in the form of a submission of the actual code of your modified ***user-name*_kb.pl** file. This code should be submitted as a separate file on Gradescope.

The mark allocation for your KB will be divided up as follows:

1. **Fact Statements.** Your fact statements should involve all key properties and relations of the domain. The relevant concept hierarchy should be specified via the `subclass` relation, so that all subclass relations are either directly asserted or derivable by the inference mechanism. A selection of significant and/or illustrative facts concerning specific entities should also be given. [5 marks]

   You should divide the facts up into blocks of similar facts and include a short comment above each of these stating the kind of facts in the block. There could also be certain individual facts that are not within a group of similar facts.

2. **Rule Statements.** Within the restricted scope of your chosen domain and the limited time available, you should aim to give inference rules that are as comprehensive as possible. Generally applicable rules are preferable to those that are very specific. More credit will be given for diversity of rules than for large numbers of very similar rules. [7 marks]

   Above each rule statements should be a comment giving a brief explanation of the rule.

C (ii) Discussion of your KB and its Inferences

You answer should be in the form of a PDF report that you will upload to Gradescope. In your report answer the following questions:

1. Write an overview of your KB in about 200 words **+ 100 words per person in your group**. This should describe the scope of the domain and briefly discuss the power and limitations of the inference rules you have specified. [6 marks]

2. Pick out up to 3 of what you consider to be the most interesting inferences that are generated by your system. **You must consider at least one inference for each member of the group.** Choose cases such that each illustrates a different aspect of your rule system. In each case explain the rules that are used to generate the inference. [12 marks]

   The most interesting inferences will typically those that require a sequence of several inference steps involving several different rules and facts. When answering the question you should give a clear account of the steps involved in making the inference.

[60 marks total]