1 Logic

Consider the following set of facts:

1. Something is a legally owned pet if and only if (1) it is a pet and (2) there exists a person who owns the pet.
2. jan is person.
3. skippy is a pet.
4. skippy is owned by jan.
5. fido is a pet.
6. fido is not owned by anybody.

1.1 First Order Logic Conversion

Using the predicates pet(X), person(X), LegallyOwned(X), and OwnedBy(X,Y) to indicate when X is owned by Y, convert the above statements to first order logic.

1.2 CNF conversion

Convert your answer to the previous part to CNF, paying particular attention to the fact that your answer should contain no quantifiers and no implications. (Your answer should be consistent with CNF in all other ways too, but we called these to points out based upon mistakes students have made in the past.)

1.3 Resolution Proof 1

Using your answer to the previous section, provide a resolution proof, using proof by refutation, that skippy is a legally owned pet.

In class, we described a resolution proof tree as a nice way to write down resolution proofs, but it's not necessarily the most compatible with typed solutions. For a format that is easy to type up, we suggestion you sue the following: List all of the CNF sentences from your answer to the the preceding part as a numbered list. For each step in your proof, add a new numbered line of the form:

```
#: Resolve line #a with line #b using substitution {...} to get:
```

where“#a”and “#b”are the lines you have combined. In this way, your proof will be a sequence of increasing line numbers that count up from 1 until you reach the final line of your your proof which should look like:

```
Resolve line with #c with #d using substitution {...} to get: NIL
```

where, again, #c and #d are lines you have used in the final step of your proof.

1.4 Resolution Proof 2

Provide a resolution proof by refutation that fido is not a legally owned pet.

2 Planning Warmup

One of the peculiar and sometimes frustrating things about classical planning is how limiting the language is for describing actions. In this problem, we'll consider the most basic planning language: STRIPS. STRIPS action descriptions involve preconditions, and ADD list, and a DELETE list. As described in class, the preconditions are a list of predicates which must all be found the database for the action to be used. The ADD list is a list of predicates that are added, and DELETE list is a list of predicates that are deleted.

Strips action descriptions must not include negation, quantification, disjunction, equality or arithmetic. **Any answer that includes any of these will get ZERO points.** Strips actions also cannot rely upon an logical rules that are assumed to be in the database. In summary, STRIPS planning cannot make inferences; it can perform lookups (to check preconditions), add thing, and delete things.

Our warmup planning problem involves moving three humans from Earth to Mars using a shuttle. This is a trivial planning problem, so it's really meant as an exercise to get you familiar with the notation and limitations of the language. Let's assume that there are k humans on earth, and that any 3 will suffice to solve the problem. How we describe the actions and the goal when we can't do arithmetic and can't use disjunctions?

One way to approach this is to add some additional propositions (predicates with 0 arguments) that we can use to do counting for us by brute force. Suppose our initial state is:

- on(earth, h1) ... on(earth, h10)
- person(h1) ... person(h10)
- OnEarth-10
- OnMars-0
- ShuttlePos(Earth)
- ShuttleContents(Empty)

We can use the proposition OnEarth-10 to indicate that there are 10 people on earth, and we'll need the predicates person(X) to make sure that we can't do silly things like moving the earth.

Now let's consider the actions we'll need. A natural way to think about this would be to have actions like:

- LoadShuttle(X) - If X is a person, the person and the swhuttle are both at earth, and shuttle is empty, then this person is loaded onto the shuttle.
- UnloadShuttle(X) - If X is a person, the person in the shuttle, and the shuttle is at mars, then the person is offloaded onto mars.
- GoToMars - If the shuttle is at earth, then it goes to Mars
- GoToEarth - If the shuttle is at Mars, then it goes to earth

The problem with this is that it doesn't give us a way of counting the number of people at each location. Without math, the only way to do this is to make 11 copies each of load and unload, one for each possible number of people at each location.

2.1 Moving the Shuttle

Write out the full STRIPS action description for GoToMars and GotoEarth.

2.2 Loading and Unloading

We don't want to make you write out 22 different actions descriptions, so provide a generic action description for one of the LoadShuttle actions, e.g., some LoadShuttle-i(X) action for some ($1 \leq i \leq 10$), which would apply only when OnEarth-i is true. Be careful with your ADD list and DELETE to make sure that you maintain a consistent state of the database. Finally, do the same for some UnLoadShuttle-i(X).

2.3 Goal State

What is the goal state?

3 Klingons and Humans

The previous problem was sort of a silly planning problem because it was obvious how to move people to Mars given the rules. Let's consider a more challenging problem where we start with three Humans h1...H3 and three Klingons k1...k3 on Earth. Our goal is to get all 6 to Mars, but we have some complicated rules we must obey:

- We have a shuttlecraft that can hold a maximum of two people (Klingons and Humans are both "people" for this question.)
- The shuttlecraft must have at least one person on board to move between locations.
- It is forbidden to leave two or more Klingons alone with just a single human on Earth or Mars because the Klingons will gang up on the lone human. (We don't count people in the shuttle in this rule. For example, it would be OK to have two Klingons on Mars and a single human in the shuttle while the shuttle is on Mars. Note, however, that such cases would never be part of a valid plan anyway since they would either arise from an illegal state or result in an illegal state when the human in unloaded.)

To handle the two spots in the shuttlecraft, we'll assume that there are two bays in the shuttle which must loaded in a particular order: bay2 can only be loaded if bay1 is not empty, and bay2 (if occupied) must be unloaded before bay1.

Our initial state will look like this:

- human(h1)
- human(h2)
- human(h3)
- klingon(k1)
- klingon(k2)
- klingon(k3)
- on(earth, h1)
- on(earth, h2)
- on(earth, h3)
- on(earth, k1)
- on(earth, k2)
- on(earth, k3)
- bay1(empty)
- bay2(empty)
- location(earth)

- location(mars)
- HOnEarth3
- HOnMars0
- KOnEarth3
- KOnMars0
- ShuttlePos(earth)

Although not shown in the initial state, we'll use in(bay1, X) to indicate that X is in bay 1, and in(bay2, X) to indicate that X is in bay2. To work around the lack of math in STRIPS, we'll need many different versions of our actions.

3.1 Actions

Write down STRIPS action descriptions for the following cases:

- EarthLoadHumanShuttleBay1-ij(H) - The preconditions for this are that the shuttle is at earth, H is a Human, we are able to load bay 1 (because both bay 1 and bay 2 are empty), there are i Humans on Earth and j Klingons on Earth. Noting that we can't do arithmetic in STRIPS, you should write this out for a specific i and j. You can pick i and j to be (almost) anything you want, but keep in mind that you must not pick an i and j that would allow you to violate the rules about leaving Humans with Klingons.
- MarsUnloadKlingongShuttleBay1-ij(K) - The preconditions for this are that the shuttle is at Mars, K is a Klingon, we are able to unload bay 1 (because somebody is in it, and bay 2 is empty), there are i humans on Mars and j Klingons on Mars. Noting that we can't do arithmetic in STRIPS, you should write this out for a specific i and j. You can pick i and j to be (almost) anything you want, but keep in mind that you must not pick an i and j that would allow you to violate the rules about leaving Humans with Klingons.

3.2 Number of Actions

Observe that we prevent illegal configurations of humans and Klingons by only having actions that result in legal states. For example, we would not have an action: EarthLoadHumanShuttleBay1-23(H) because this would leave 1 human with 3 Klingons. You can also assume that we don't bother defining actions with preconditions that would imply we are starting the action from an illegal state. Taking this into account, and including the GoToEarth and GoToMars actions, how many actions in total does the approach described above produce? Your answer should count all legal combinations, and not just those used by a particular solution. (Don't solve the next question, then come back to this one and remove the actions your solution didn't need.)

3.3 Solution

Write out a valid plan which achieves the goal state:

- KOnMars3

- HOnMars3

After completing this exercise, you are probably thinking that there must be a better way to do this. Difficulties in dealing with numerical conditions indeed contributed to a subfield of planning that involves planning with numerical resources. Such extensions move classical planning towards the closely related field of *scheduling* since time is really just one of many possible numerical resources that actions can influence. For an example of a real-life system that combines concepts you have learned in class, check out the Spike Scheduling and Planning System, which combines planning, scheduling and CSPs for the Hubble telescope.

4 Probability

Consider the distribution over 4 binary random variables shown below:

A	B	C	D	P
0	0	0	0	$\frac{1}{8}$
0	1	0	0	$\frac{1}{32}$
1	0	0	0	$\frac{1}{32}$
1	1	0	0	$\frac{1}{32}$
0	0	1	0	$\frac{1}{32}$
0	1	1	0	$\frac{1}{16}$
1	0	1	0	$\frac{1}{16}$
1	1	1	0	$\frac{1}{8}$
0	0	0	1	$\frac{1}{8}$
0	1	0	1	$\frac{1}{32}$
1	0	0	1	$\frac{1}{32}$
1	1	0	1	$\frac{1}{32}$
0	0	1	1	$\frac{1}{32}$
0	1	1	1	$\frac{1}{16}$
1	0	1	1	$\frac{1}{16}$
1	1	1	1	$\frac{1}{8}$

4.1 Conditional Probabilities

Compute $P(Ab|Cd)$. Observe that A and C are capitalized, but b and d are not. Think carefully about how many numbers we are asking for.

4.2 Conditional Independence

It's easy to see from how the table is presented (the last 8 probabilities for the d case mirror those of the $\overline{d}$ case) that for any of the other variables, such as A, $P(A|D) = P(A)$, i.e., knowing D tells us nothing about A. It's less salient from the structure of the table that reverse is also true: $P(D|A) = P(D)$. Prove that, in general, for any two variables X and Y, if $P(X|Y) = P(X)$, then $P(Y|X) = P(Y)$.

5 Bayes Rule in the Real World

Some people may be confused about how the following two pieces of information can be both be true:

- Vaccination reduces the risk of infection.
- Most people who are infected are vaccinated.

Let's look at some numbers to see how this can be. Suppose that the probability of getting a disease if you are vaccinated is reduced by a factor of 4, i.e., $P(d|v) = \frac{1}{4} \times P(d|\overline{v})$, and $P(d|\overline{v})$ is 0.4. Further, assume that the unconditional probability of the getting the disease is 0.15. What fraction of people must be vaccinated for most infected people to also be vaccinated, i.e., $P(v|d) > 0.5$? (Note: This question is phrased as a bound where you assume $P(v|d) > 0.5$, and then bound $p(v)$, and you can answer the question this way, but it turns out there's enough information already provided for you to compute an exact value of $P(v|d)$ which, conveniently, happens to be larger than 0.5. If you go that route, you can compute an exact value of $P(v)$ as well.)