Note: As in previous assignments, CSC 444 problems may be tried by CSC 244 students for bonus points (at half their point-value). And as before, we trust you to work independently on this assignment (though by all means consult the TAs, the instructor, and each other about principles underlying the problems, as long as you don’t share specific ways of solving them).

Problem 1 (Resolution strategies)

(a) Consider the following set of clauses:
1. $P(A) \lor Q(A) \lor \neg Q(A)$
2. $P(x) \lor Q(x) \lor R(f(x))$
3. $P(y) \lor \neg R(y)$
4. $\neg P(A) \lor Q(A)$
5. $P(B) \lor \neg Q(A) \lor R(B)$

(i) Which clauses would be eliminated by what rules (among the clause elimination strategies we discussed, applied repeatedly if possible) and why?

(ii) What does the result of this process tell us about the satisfiability of clauses 1-5? Why?

(b) (i) Intuitively, why is the Set-of-Support (SoS) strategy effective in general (where the set of support consists of the clauses derived from the denial of the conclusion)?

(ii) Suppose we attempt a resolution refutation using a depth-first strategy, starting with a resolution step in which one of the clauses was derived from the denial of the conclusion. Will the refutation attempt conform with an SoS strategy? Why or why not?

(iii) Give the simplest set of unsatisfiable clauses that you can make up for which unit resolution (used as the only rule) would fail to derive the empty clause. Show that for the more general resolution method, the empty clause is indeed derivable.

(iv) Apart from speed, what is another advantage of the model generation / elimination method, in terms of the information it can provide about a set of clauses?
Problem 2 (Specialist inference)

A depth-first (preorder) tree-numbering scheme was described in class that enables constant-time determination of the relationship between two types in a type hierarchy (under certain assumptions, whether they are incompatible types, or whether one is a generalization of the other).

A similar scheme can be applied to part hierarchies, i.e., hierarchies that specify what top-level parts an object (or type of object) is composed of, what subparts these top-level parts are composed of, and so on.

(a) One issue that arises is that we can view part structure in multiple ways. For instance, instead of using “superficial anatomy” for the human body (head, neck, torso, arms, legs, and each of their parts, such as upper arm, lower arm, hand, etc.), we can take a systemic perspective, dividing the body into skeleton, cardiovascular system, musculature, nervous system, gastrointestinal system, and so forth.

If you were programming a system for representing part structure, (i) how would you modify the depth-first numbering scheme to allow for multiple views? (ii) What relationships would the numbering scheme enable you to determine? (iii) Would this be a constant-time operation? Why or why not? (iv) Would you be able to determine whether, for instance, the human neck and the human skeleton have any part in common (in reality, these are the cervical vertebrae). If not, what’s the problem? If so, how? Do your best to answer the preceding questions, using diagrams if that helps.

[Hints: Assume that the superficial and systemic perspectives are both tree-structured, with each node representing a body part, and its children (for non-leaf nodes) representing a division of that part into disjoint, exhaustive subparts. The two perspectives have the same root — i.e., the entire human, but after that some nodes (vertices) in the two perspectives may be distinct for the two perspectives, while others might be shared. An obvious constraint for the two perspectives, given that they are exhaustive subdivisions of the body, is that ultimately the root-to-leaf paths in both perspectives should lead to the same set of leaf (terminal) nodes — the smallest parts at which the subdivisions are considered to terminate.]

(b) (CSC 444 only) Another potential problem concerns that of numerous similar parts. For instance, a centipede may have 70 legs, and a human brain may have $10^{11}$ neurons. What’s the problem here, in answering questions about whether or not such-and-such a type of thing has such-and-such parts (and how two specified types of parts are related)? And what might be a possible approach to solving it? Keep in mind that we may want to further specify the structure of a centipede’s leg, or a neuron. Answer in about paragraph or a little more, using diagrams if that helps.

(c) The “tree of parts” approach to representing part structure (as in (a)) doesn’t tell
us much about the geometry of the parts of an object, except that all the subparts of a part \( X \) must be geometrically contained in the region occupied by \( X \).

We could combine a tree-of-parts approach with a 3-D modelling approach, to enable reasoning about both part structure and geometric/topological properties such as relative sizes, angles, and containment. For instance, we can imagine a 3-D regular grid of “voxels” (volume elements), sufficiently fine-grained so that the smallest parts we want to model will occupy at least one voxel. We then use “spatial occupancy” labeling, where the label of a voxel is the name of the smallest part that the voxel belongs to (if any). Then, using the tree-of-parts structure, we can also infer what higher-level parts the voxel belongs to. Let’s assume that the part structure is a unique partitioning structure, i.e., each non-elementary part is divided into mutually exclusive, jointly exhaustive parts.

(i) In sketchy algorithmic terms, how could we determine whether one part (at any structural level) is adjacent to (touching) another?

(ii) In sketchy algorithmic terms, how could we determine whether one part completely surrounds another (in the 3-D sense, the way the skin of an orange surrounds its flesh, or the way the cornea surrounds the lens of your eye? (Hint: Perhaps think in terms of a label-propagation process.)

**Problem 3: Natural Logic (NLog) inference**

(a) How could NLog perform the following inferences? Be sure to specify the polarity of relevant syntactic environments, any implicative properties you are using, and any concept specializations or generalizations that you are using. Note that some specialization/generalization relations may be given as explicit premises of form “All \( P \) are \( Q \)” or “All \( P \) are \( \neg Q \)”, where \( P \) and \( Q \) are concepts (typically predicative words or simple phrases); while others, especially for logical relations like those between quantifiers or boolean connectives need to be assumed in addition.

Several great tenors are Scottish; therefore

Some great singers are British.

No qualified medical doctor fails to get a well-paid job; therefore,

No highly qualified oncologist fails to get a paid job.

Smith knew that ITEL managed to win the service contract in 1992; therefore,


(b) Can you see a way for NLog to perform the following inference (in multiple steps)? Explain how this would go. *Strong hints:* First, note that every sentence of form “\( \forall \ P \) are \( Q \)” can be regarded as a generalization of a predicate \( P \) to a predicate \( Q \), no matter how many words are involved in \( P \) and \( Q \). Second, try to make use of the conservativity property of all quantifiers \( \Omega \): “\( \Omega \ P \) are \( Q \)” implies “\( \Omega \ P \) are
Finally, note that dropping a conjunct from a conjunction generalizes it. Again make reference to positive and negative environments as appropriate.

Every Italian tenor wants to be great; and
Some Italian tenors are great; therefore,
Some Italian tenors want to be great

**Problem 4: Satisfiability testing**

Apply the DPLL procedure to the following clauses, showing successive steps, and being careful to abide by the constraints on choices made by DPLL:

\[ A, \neg A \lor B \lor C, \neg A \lor B \lor \neg C, C \lor \neg D, E \lor F, E \lor \neg F \lor \neg G, E \lor F \lor G, D \lor \neg E, \neg D \lor \neg E \]

**Problem 5: Frames**

(a) Frames seem well-suited for entities that can be described in terms of a set of more or less independent features or component entities. For example, for a particular women’s singles tennis match, we might have the following information: This was a US Open Tennis final, the first-place winner was Emma Raducanu (nationality Britain), the second-place winner was Leylah Fernandez (nationality Canada), there were 2 sets with respective scores 6-4, 6-3, the date was Sept. 11, 2021, and the place was Arthur Ashe Stadium, New York.

Provide (i) an individual frame for this match; and (ii) a definitional frame for the concept of a women’s singles tennis match, using the syntax illustrated in the class notes (taking care to use keywords, bracketing, and upper/lower case as required by the syntax). Assume that there are either 2 or 3 sets, of which 2 are winning sets. Each winning set score is one of 6-0, 6-1, 6-2, 6-3, 6-4, or 7-6, and any losing set score is 0-6, 1-6, 2-6, 3-6, 4-6, or 6-7.

(b) Marvin Minky’s “Frame systems” paper ultimately led to the development of description logics, and the version of frames presented in class to some extent reflects this development. However, that version seems rather more limited than what Minsky had in mind. Recall that Minsky expected frames to be able to encode the expected configuration of entities such as certain kinds of children’s blocks (seen from different angles) or rooms (such as a bedroom or classroom).

Here’s something of this sort, but simpler: The concept of a caged canary. This should encode some properties of the canary (color, sex, age), the cage (its size and perhaps shape), and essential accessories like a perch, a feeder station, and a water tray. Furthermore – and for this you’ll probably want to extend what is allowed in the “value” position of frame slots a bit – the bird is (normally) in the cage, and the accessories are normally in the cage as well, and the bird is typically on the perch. Explain the extensions you find necessary (if any) beyond the syntax specified in class.