Introduction to Artificial Intelligence (CS-487)
Assignment #3

Due on Dec 15, 2022

Professor I. Tsamardinos

University of Crete
Department of Computer Science

Nikolaos Kougioulis (ID 1285)

December 14, 2022

Exercise 1

Represent the following sentences in first-order logic, using a consistent vocabulary (which you must define):

1. Some students took French in spring 2001.
2. Every student who takes French passes it.
3. Only one student took Greek in spring 2001.
4. The best score in Greek is always higher than the best score in French.
5. Every person who buys a policy is smart.
6. No person buys an expensive policy.
7. There is an agent who sells policies only to people who are not insured.
8. There is a barber who shaves all men in town who do not shave themselves.
9. A person born in the UK, each of whose parents is a UK citizen or a UK resident, is a UK citizen by birth.
10. A person born outside the UK, one of whose parents is a UK citizen by birth, is a UK citizen by descent.
11. Politicians can fool some of the people all of the time, and they can fool all of the people some of the time, but they can't fool all of the people all of the time.
12. All Greeks speak the same language. (Use Speaks(x, l) to mean that person x speaks language l.)

Solution:

1. Student(x): whether x is a student, HasCourse(x,y,z): Student x takes course z on semester y

$$\exists\ s : \text{Student}(s) \wedge \text{HasCourse}(s, Spring2001, French)$$

2. Passes(x,y,z): Student x passes course z on semester y

$$\forall s,\ sem : \text{Student}(s) \wedge \text{HasCourse}(s, sem, French) \Rightarrow \text{Passes}(s, sem, French)$$

3. This is a bit trickier, as we need a way to write "only one" in first-order logic. One could write

$$\exists\ !s : \text{Student}(s) \wedge \text{HasCourse}(s, Spring2001, Greek)$$

Intuitively, there is only one element x satisfying property P if-f for every other element y satisfying P we have $x = y$. That is, $\exists\ x : P(x) \wedge (\forall y : P(y) \Rightarrow x = y)$, or $\exists\ x :\ \forall y : P(y) \Leftrightarrow x = y$. So overall

$$\exists\ s : (\text{Student}(s) \land \text{HasCourse}(s, Spring2001, Greek)) \land (\forall s' : (\text{Student}(s') \land \text{HasCourse}\,(s', Spring2001, Greek) \Rightarrow s = s'))$$

4. Score(x,y,z): The Grade of student x on semester y on course z

$$\exists\ s :\ \forall s', sem :\ \text{Score}(s, sem, Greek) > \text{Score}(s', sem, French)$$

5. Person(x): whether x is a person, Buys(x,y): Person x buys item i, Policy(i): whether item i is a policy, Smart(x): whether person x is smart

$$\forall\ p : \text{Person}(p) \land (\exists\ i : \text{Buys}(p, i) \land \text{Policy}(i)) \Rightarrow \text{Smart}(p)$$

6. Expensive(x): whether item x is expensive

$$\nexists\ p, i : \text{Person}(p) \land \text{Buys}(p, i) \land \text{Policy}(i) \Rightarrow \text{Expensive}(i) \qquad \overset{\text{equiv.}}{\Longrightarrow}$$

$$\forall\ p, i : \text{Person}(p) \land \text{Buys}(p, i) \land \text{Policy}(i) \Rightarrow \neg\text{Expensive}(i)$$

7. Agent(x): whether x is an agent or not, Sells(x,y,z): x sells item y to z, Insured(z): whether z is insured

$$\exists\ a, pol, person : \text{Agent}(a) \land \text{Sells}(a, pol, person) \land \text{Policy}(pol) \land \text{Person}(person) \Rightarrow \neg\text{Insured}(person)$$

8. Barber(x): whether x is a barber, Man(y): whether y is a man, Shaves(z): whether z shaves himself, BarberShaves(x,y): whether (barber) x shaves y

$$\exists\ b : \text{Barber}(b) \land (\forall p :\ \text{Man}(p) \land \neg\text{Shaves}(p) \Rightarrow \text{BarberShaves}(b, p))$$

9. Born(x,c): person x born in country c, Parent(x,y): parent of x is y, Citizen(x,c): person x is citizen (at birth) of country c, Resident(x,c): person x is a resident of country c

$$\begin{aligned}\forall\ p : \text{Person}(p) \land \text{Born}(p, UK) \land (\forall p_1, p_2 : (\text{Parent}(p, p_1) \land \text{Parent}(p, p_2)))\\ \land (\text{Citizen}(p_1, UK) \land \text{Citizen}(p_2, UK))\\ \lor (\text{Resident}(p_1, UK) \land \text{Resident}(p_2, UK)) \Rightarrow \text{Citizen}(p, UK)\end{aligned}$$

10. Descent(x,c): person x is of c descent

$$\begin{aligned}\forall\ p : \text{Person}(p) \land \neg\text{Born}(p, UK) \land (\exists p_1, p_2 : (\text{Parent}(p, p_1) \lor \text{Parent}(p, p_2))\\ \land (\text{Citizen}(p_1, UK) \lor \text{Citizen}(p_2, UK)) \Rightarrow \text{Descent}(p, UK)\end{aligned}$$

11. Politician(x): whether x is a politician, Fool(x,y,t): x fools y on time t

    Politicians can fool some of the people all of the time:

$$\forall\ pol : \text{Politician}(pol) \land (\exists p, \forall t : \text{Fool}(pol, p, t) \land \text{Person}(p))$$

    They can fool all of the people some of the time:

$$\forall\ pol : \text{Politician}(pol) \land (\forall p, \exists t : \text{Fool}(pol, p, t) \land \text{Person}(p))$$

    They can't fool all of the people all of the time:

$$\forall\ pol : \text{Politician}(pol) \land (\forall p, t : \neg\text{Fool}(pol, p, t) \land \text{Person}(p))$$

    All together:

$$\begin{aligned}\forall\ pol : \text{Politician}(pol) \bigwedge &(\exists p, \forall t : \text{Fool}(pol, p, t) \land \text{Person}(p))\\ &\land (\forall p, \exists t : \text{Fool}(pol, p, t) \land \text{Person}(p))\\ &\land (\forall p, t : \neg\text{Fool}(pol, p, t) \land \text{Person}(p))\end{aligned}$$

12. Greek(x): whether x is Greek (not specified by Birth or by Descent), Speaks(x,l): x speaks language l (not specified as "Greek")

$$\forall\ p_1, p_2, l : \text{Greek}(p_1) \land \text{Greek}(p_2) \land \text{Speaks}(p_1, l) \Rightarrow \text{Speaks}(p_2, l)$$

Exercise 2

Write a general set of facts and axioms to represent the assertion "Wellington heard about Napoleon's death" and to correctly answer the question "Did Napoleon hear about Wellington's death?"

Solution:

Intuitively, if Wellington heard about Napoleon's death then it's obvious that Napoleon remains dead after his death, so he cannot hear about Wellington's death afterwards. So the answer to "Did Napoleon hear about Wellington's death?" is False.

In order to correctly write a general set of axioms and facts, we must also take into account time. Our reasoning to deduce the set of axioms and facts is the following:

Time embodies direction, that is, *asymmetry* (t_1 occurring after t_0 means that t_0 occured before t_1) and order, that is, *transitivity* (t_2 occurring after t_1 and t_1 occurring after t_0 implies that t_2 occurs after t_0). We assume time is discrete instead of continuous, such as minutes or hours. We also assume that Wellington and Napoleon have good judgement so they cannot hear about each others death unless that event really happened.

Wellington can only hear about Napoleon's death after the time of death. If Napoleon died at time t_0, then Wellington can only hear at a time $t_1 > t_0$. The atomic sentence HeardOfEvent(p, e, t) is interpreted as "Person p heard of event e on time t", and Death(p, t) as "Death of person p on time t". TimeHappened(e) returns the time event e occured. In our case of Wellington's or Napoleon's death.

Wellington can only hear about Napoleon's death if he is not dead himself, so this gives us another axiom and allows us to deduce that Wellington must not be dead at time t_0.

Our facts and axioms are:

- Time property is assymetric:
$$\forall\ t_0, t_1 :\ (t_1 > t_0) \Rightarrow \neg(t_0 > t_1)$$
- Time property is transitive:
$$\forall\ t_0, t_1, t_2 :\ (t_1 > t_0) \land (t_2 > t_1) \Rightarrow (t_2 > t_0)$$
- If a person hears an event on time t, he is not dead at time t
$$\forall\ p, e, t : \text{HeardOfEvent}(p, e, t) \Rightarrow \neg\text{Death}(p, t)$$
- If a person is dead on time t_0, he keeps being dead after t_0
$$\forall\ p, t_0 : \text{Death}(p, t_0) \Rightarrow (\forall t > t_0 : \text{Death}(p, t))$$
- A person can only hear an event after the time is has occurred
$$\forall\ p, e, t : \text{HeardOfEvent}(p, e, t) \Rightarrow t > \text{TimeHappened}(e)$$

  In our case of Wellington's or Napoleon's death,
$$\forall\ p, e,\ t > t_0 : \text{HeardOfEvent}(p, e, t) \Rightarrow t > \text{Death}(p, t_0)$$

- Death of p at a time t means that the Died event must be updated

  $$\exists\, p, t : \text{Death}(p, t_0) \Rightarrow \text{Died}(p)$$

  Where $e \equiv \text{Died}(p)$ is the event of person p to be True when p is dead, False otherwise. This is a way of performing bookkeeping (know if event happened) in order to pass the event as an argument when quering our KB (see on the next page). Another interpretation for this fact would be to perform skolemization with $t = t_0$, obtain $\text{Death}(p, t_0)$ and keep it as $e \equiv \text{Death}(p, t_0)$.

- Wellington heard about Napoleon's death:

  $$\exists\, t_0, t_1 : \text{HeardOfEvent}(Wellington, \text{Death}(Napoleon, t_0), t_1)$$

To answer the query "Did Napoleon hear about Wellington's death?", we can pass $\text{Died}(Wellington)$ to the HeardOfEvent predicate and perform the query

$$\text{ASK}(KB, \text{HeardofEvent}(Napoleon, \text{Died}(Wellington), t)$$

or as mentioned before, using the skolemization step

$$\text{ASK}(KB, \text{HeardofEvent}(Napoleon, e \equiv \text{Death}(Wellington, t_0), t)$$

Exercise 3

This question considers Horn KBs, such as the following:

$$P(F(x)) \Rightarrow P(x) \tag{1}$$

$$Q(x) \Rightarrow P(F(x)) \tag{2}$$

$$P(A) \tag{3}$$

$$Q(B) \tag{4}$$

Let FC be a breadth-first forward-chaining algorithm that repeatedly adds all consequences of currently satisfied rules; let BC be a depth-first left-to-right backward-chaining algorithm that tries clauses in the order given in the KB.

Which of the following are true?

1. FC will infer the literal Q(A).
2. FC will infer the literal P(B).
3. If FC has failed to infer a given literal, then it is not entailed by the KB.
4. BC will return true given the query P(B).
5. If BC does not return true given a query literal, then it is not entailed by the KB.

Solution:

Starting from known facts, Forward Chaining instantiates all the rules whose premises are satisfied, adding their conclusions to the known facts in the Knowledge Base using Modus Ponens, by Breadth-First Search. Backward Chaining works starting from the goal, and chains through rules to find known facts that support the proof, by Depth-First Search.

1. False. There is no rule in the Knowledge Base to infer the literal $Q(A)$.
2. True. Forward Chaining fires the rule $Q(x) \Rightarrow P(F(x))$ to infer $P(F(B))$ from the fact $Q(B)$. Then from the rule $P(F(x)) \Rightarrow P(x)$ FC infers $P(B)$ from the fact $P(F(B))$.
3. True, because every inference is an application of Generalized Modus Ponens and answers every query whose answers are entailed by any KB of definite clauses (completeness).
4. True. BC will fire $P(F(x)) \Rightarrow P(x)$, but it will not be able to apply it. Then, BC will try $Q(x) \Rightarrow P(F(x))$, applied with $\{x/B\}$, because the KB contains the fact $Q(B)$. This generates $P(F(B))$ which can be proved by applying $P(F(x)) \Rightarrow P(x)$ with $\{x/B\}$.
5. False, because BC may suffer from repeated states and incompleteness (Chapter 9, Section 4, p. 294)[1].

[1] Russell S. J. & Norvig P. (2020). *Artificial intelligence: A Modern Approach* (4th ed.), Pearson.

Exercise 4

Suppose a knowledge base contains just the following first-order Horn clauses:

$$\text{Ancestor}(\text{Mother}(x), x)$$

$$\text{Ancestor}(x, y) \wedge \text{Ancestor}(y, z) \Rightarrow \text{Ancestor}(x, z)$$

Consider a forward chaining algorithm that, on the j-th iteration, terminates if the KB contains a sentence that unifies with the query, else adds to the KB every atomic sentence that can be inferred from the sentences already in the KB after iteration $j - 1$.

1. For each of the following queries, say whether the algorithm will (1) give an answer (if so, write down that answer); or (2) terminate with no answer; or (3) never terminate.
   (a) Ancestor(Mother(y), John)
   (b) Ancestor(Mother(Mother(y)), John)
   (c) Ancestor(Mother(Mother(Mother(y))), Mother(y))
   (d) Ancestor(Mother(John), Mother(Mother(John)))
2. Can a resolution algorithm prove the sentence ¬ Ancestor(John, John) from the original knowledge base? Explain how, or why not.
3. Suppose we add the assertion that ¬(Mother(x) = x) and augment the resolution algorithm with inference rules for equality. Now what is the answer to (2)?

Solution:

1. (a) Yes, right by instantiating $\{y/\text{John}\}$, (b) Yes, on the second iteration, by instantiating $\{y/\text{John}\}$, (c) Yes, the empty set (d) No answer, as it does not terminate.
2. Resolution cannot prove the sentence of John not being an Ancestor of John, as it is does not follow from the knowledge base. So ¬ Ancestor(John, John) cannot be proved by Resolution.
3. Resolution is still unable to prove (2).