CSC110 Fall 2021 Assignment 2: Logic, Constraints, and Nested Data

Nathaniel Yong

October 11, 2021

Part 1: Predicate Logic

1. 1. Yes, D_1 could be the set of natural numbers. Statement 1 is true but statement 2 is false. For every x, you can take $y = x + 1$ and statement 1 is satisfied. However, statement 2 is false for $y = 0$, because there is obviously no natural number x where $x < 0$.
   2. Yes, D_2 could be the set of all integers. For every x in this set, you can take $y = x + 1$, and statement 1 is satisfied. For every y in this set, you can take $x = y - 1$, and statement 2 is satisfied.
   3. Yes, let $D_3 = \{0\}$. When $x = 0$, there is no y less than it. When $y = 0$, there is no x bigger than it.
2. 1. $P(x) : x > 2$, where $x \in S$
   2. $Q(x) : x > 1$, where $x \in S$

   These two predicates make statement 3 false and statement 4 true. Statement 3 is false, because if you take $x = 0$, $P(x) \land Q(x)$ evaluates to False $\land$ False, which is simply False. Statement 4 is true, because if $x \leq 2$, then $P(x)$ is false, and so the statement is vacuously true. However, if $x > 2$, then $P(x)$ is true, and $Q(x)$ must also be true because if $x > 2$ then obviously $x > 1$.
3. Complete this part in the provided `a2_part1.py` starter file. Do **not** include your solution in this file.
4. Complete this part in the provided `a2_part1.py` starter file. Do **not** include your solution in this file.

Part 2: Conditional Execution

Complete this part in the provided `a2_part2.py` starter file. Do **not** include your solution in this file.

Part 3: Generating a Timetable

1. Complete this part in the provided `a2_part3.py` starter file. Do **not** include your solution in this file.
2. (a) *IMPORTANT DEFINITIONS/NOTATION* (don't change this text!)

   We define the following sets:
   - C: the set of all possible courses
   - S: the set of all possible sections
   - M: the set of all possible meeting times
   - SC: the set of all possible schedules

   We also define the following notation for expressions involving the elements of these sets:
   - The first three (courses/sections/meeting times) are represented as tuples (as described in the assignment handout), and you can use the indexing operation on these values. For example, you could translate "every section term is in $\{'F','S','Y'\}$" into predicate logic as the statement:

   $$\forall s \in S,\ s[1] \in \{'F','S','Y'\}$$

   - The start and end times of a meeting time can be compared chronologically using the standard $<$, $\leq$, $>$, and $\geq$ operators.
   - For a section $s \in S$, $s[2]$ represents a tuple of meeting times. You may use standard set operations and quantifiers for these tuples (pretend they are sets). For example, we can say:
     - $\forall s \in S,\ s[2] \subseteq M$
     - $\forall s \in S,\ \forall m \in s[2],\ m[1] < m[2]$
   - Finally, for a schedule $sc \in SC$, you can use the notation $sc.sections$ to refer to a set of all sections in that schedule. You can use quantifiers with that set of schedules as well, e.g. $\forall s \in sc.sections$, ...

   Predicate for meeting times conflicting:

   $$MeetingTimesConflict(m_1, m_2) : (m_1[0] = m_2[0]) \land (m_1[1] \leq m_2[1] < m_1[2] \lor m_1[1] < m_2[2] \leq m_1[2]) \quad \text{where } m_1, m_2 \in M$$

   Predicate for sections conflicting:

   $$SectionsConflict(s_1, s_2) : (s_1[1] = s_2[1] \lor s_1[1] = `Y' \lor s_2[1] = `Y') \land (\exists m_1, m_2 \in s_1[2], MeetingTimesConflict(m_1, m_2)) \quad \text{where } s_1, s_2 \in S$$

   Predicate for valid schedule:

   $$IsValidSchedule(sc) : \forall s_1 \in sc.sections, \forall s_2 \in sc.sections, (s_1 \neq s_2 \implies \neg SectionsConflict(s_1, s_2)) \quad \text{where } sc \in SC$$

   (b) Complete this part in the provided `a2_part3.py` starter file. Do **not** include your solution in this file.
3. (a) You may use all notation from question 2(a). Note that a course $c \in C$ is a tuple, and $c[2]$ is a set of sections, and so can be quantified over: $\forall s \in c[2], \ldots$.

   Predicate for section-schedule compatibility:

   $$IsCompatibleSection(sc, s) : \forall s_2 \in sc.sections, \neg SectionsConflict(s, s_2) \quad \text{where } sc \in SC, s \in S$$

   Predicate for course-schedule compatibility:

   $$IsCompatibleCourse(sc, c) : \exists s \in c[2], IsCompatibleSection(sc, s) \quad \text{where } sc \in SC, c \in C$$

   (b) Complete this part in the provided `a2_part3.py` starter file. Do **not** include your solution in this file.

Part 4: Processing Raw Data

Complete this part in the provided `a2_part4.py` starter file. Do **not** include your solution in this file.