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

Azalea Gui
Peter Lin

October 3, 2021

Part 1: Predicate Logic

1.
   1. When $D_1 = [0, \infty)$
      Statement 1 is True because every number $x \in D_1$ is smaller than a $y \in D_1$ (For example, $y = x + 1 > x$).
      Statement 2 is False because when $y = 0$, there isn't an $x \in D_1$ smaller than y.
   2. When $D_2 = \mathbb{Z}$
      Statement 1 is True because every integer x is smaller than some integer y (For example, $y = x + 1 > x$).
      Statement 2 is True because every integer y is greater than some integer x (For example, $x = y - 1 < y$).
   3. When $D_3 = \{0\}$
      Statement 1 is False because when $x = 0$, there isn't a $y \in D_3$ greater than x.
      Statement 2 is False because when $y = 0$, there isn't an $x \in D_3$ smaller than y.
2.
   1. $P(x) : 1 < x < 8$ where $x \in S$
   2. $Q(x) : x < 8$ where $x \in S$

   When $x \leq 1$ and $x \geq 8$, $P(x)$ is False, so $P(x) \land Q(x)$ would be False and $P(x) \implies Q(x)$ would be True.
   When $1 < x < 8$, $P(x)$ and $Q(x)$ are both True, so both $P(x) \land Q(x)$ and $P(x) \implies Q(x)$ would be True.
   Since $S = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$,
   Statement 3 would be False because $P(x) \land Q(x)$ is False when $x = 0$,
   Statement 4 would be True because $P(x) \implies Q(x)$ is true for all values of $x \in S$.
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:

   $$MeetingTimesConflict(m_1, m_2) : m_1[0] == m_2[0] \land m_1[2] > m_2[1] \land m_2[2] > m_1[1] \qquad \text{where } m_1, m_2 \in M$$

   Predicate for sections conflicting:

   $$SectionsConflict(s_1, s_2) : (s_1[1] = \text{'Y'} \lor s_2[1] = \text{'Y'} \lor s_1[1] = s_2[1]) \land \exists m_1 \in s_1[2], \exists m_2 \in s_2[2],\ \text{s.t. } MeetingTimesConflict(m_1, m_2) \qquad \text{where } s_1, s_2 \in S$$

   Predicate for valid schedule:

   $$IsValidSchedule(sc) : \forall s_1, s_2 \in sc.sections, SectionsConflict(s_1, s_2) \Rightarrow s_1 = s_2 \qquad \text{where } sc \in SC$$

   (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: $\forall s \in c[2], ...$

   Predicate for section-schedule compatibility:

   $$IsCompatibleSection(sc, s) : \forall s_1 \in sc.sections, \neg SectionsConflict(s, s_1) \qquad \text{where } sc \in SC, s \in S$$

   Predicate for course-schedule compatibility:

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

   (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.