**CECS 545-50 Exercises 8.6, 8.15, 8.16**

**Ryan Schron**

**Exercise 8.6**

Vocabulary:

Student(x), Person(x), Agent(x), Barber(x), Expensive(x), Politician(x), Insured(x), Smart(x) : Defines x as a member of a category

F, G : Represents French and Greek respectively

UK: Represents the United Kingdom

Birth: Represent the reason citizenship by birth

Take(x, c, s) : student x, course c, semester s

Pass(x, c) : student x, course c

Score(x, c, s): student x, course c, semester s

Buys(x, y, z): x buys y from z

Shaves(x, y): x shaves y

Sells(x, y , z): x sells y to z

Citizen(x, c, r): x is a citizen of country c for reason r

Resident(x, c): x is a citizen of country c

Birthplace(x, c): x is born in country c

Parent(x, z): z is the parent of x

Fools(x, y , t): x fools y at time t

1. Some students took French in spring 2001

ꓱx: Student(x) ꓥ Take(x, F, Spring 2001)

1. Every student who takes French passes it

ꓯx,s : Student(x) ꓥ Take(x, F, Spring 2001) -> Pass(x, F, s)

1. Only one student took Greek in spring 2001

ꓱx : Student(x) ꓥ Take(x, G, Spring 2001) ꓥ ꓯy ≠ x -> ⌐ Take(y, G, Spring 2001)

1. The best score in Greek is always higher than the best score in French

ꓯsꓱxꓯy: Score(x, G, s) > Score(y, F, s)

1. Every person who buys a policy is smart

ꓯx: Person(x) ꓥ (ꓱy, ꓱz: Policy(y) ꓥ Buys(x, y, z)) -> Smart(x)

1. No person buys an expensive policy

ꓯx, ꓯy, ꓯz: Person(x) ꓥ Policy(y) ꓥ Expensive(y) -> ⌐ Buys(x, y, z)

1. There is no agent who sells policies only to people who are not insured

ꓯx, ꓯy, ꓯz: Agent(x) ꓥ Policy(y) ꓥ ⌐ Sells(x, y, z) -> (Person(z) ꓥ ⌐ Insured(z))

1. There is a barber who shaves all men in town who do not shave themselves

ꓱx, ꓯy: Barber(x) ꓥ Man(y) ꓥ ⌐Shaves(y, y) -> Shaves(x,y)

1. A person born in the UK, each whose parents is a UK citizen or a UK resident, is a UK citizen by birth

ꓯx, ꓯy: (Person(x) ꓥ Birthplace(x, UK)) ꓥ (Person(y) ꓥ (Citizen(y, UK, r) ꓦ Resident(y, UK)) ꓥ Parent(x, y) -> Citizen(x, UK, Birth)

1. A person outside the UK, one of whose parents is a UK citizen by birth, is a UK citizen by descent.

ꓯx, ꓱy: Person(x) ꓥ ⌐Birthplace(x, UK) ꓥ (Parent(x,y) ꓥ (Citizen(y, UK, r) ꓦ Resident(y, UK)) -> Citizen(x, UK, Birth)

1. 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.

ꓯx Politician(x) -> (ꓱy ꓯt Person(y) ꓥ Fools(x, y, t)) ꓥ (ꓱt ꓯy Person(y) ꓥ Fools(x, y, t)) -> Fools(x, y, t) ꓥ ⌐(ꓯy ꓯt Person(y) -> Fools(x, y, t))

**Exercise 8.15**

The problem with the proposed definition of adjacent squares in the Wumpus world is that it does not work on the boundaries and would require additional conditions to represent such cases.

**Exercise 8.16**

The axioms required would be

ꓯs1 Smells(s1) <-> ꓱs2 Adjacent(s1, s2) ꓥ In(Wumpus, s2)

That is, for every square that smells, there exist and adjacent square containing the Wumpus.

ꓱs1 In(Wumpus, s1) ꓥ ꓯs\_2 ≠ s1 -> ⌐ In(Wumpus, s2)

A property of uniqueness is also required by definition. That is, there exists a square (s1) that contains a Wumpus, and thus every other square must not contain the Wumpus.