PMT Additional Exercises (Week 7)

Nicholas Sim

November 16th, 2017

1 Logic: First-order translations

1.1 Easy translations

Defining appropriate constants, translate the following. You may use the following relations:

- person(x) means x is a person
- hungry(x) means x is hungry
- loves(x, y) means x loves y

We define an ordering.

1. All love is unreciprocated. *(anti-symmetry)*
2. If a person's beloved loves another, then so does the original person. *(transitivity)*
3. Nobody loves themselves. *(anti-reflexivity)*
4. Between any two people, one must love the other. *(totality)*
5. Everybody loves somebody. *(minimum element)*

Some mathematical translations (you may assume the ordering above holds)

6. There is a hungry person who is loved by all other hungry people.
7. There are (at least) three hungry people.
8. There are at most two hungry people.
9. There are infinitely many hungry people.

You might notice that unary relations can define sets (like the set of hungry people).

It's hard to discuss mathematics without functions. Once we have these we can construct algebraic structures, such as groups, with predicate logic.[1]

1.2 Graphs (also easy)

Suppose L is a language with equality and a 2-ary relation symbol E.

Define a first-order L-structure G, calling the elements in its domain *vertices*, using constant symbols (of L) $v_1, v_2, \ldots$. Write a formula such that `E` has the following properties in G:

1. `E` is irreflexive
2. `E` is symmetric

[1] Or we can (painfully!) use relations to stand in for functions, as you'll see next term.

Call G an undirected graph. Translate the following:

1. G is a complete graph
2. Any two vertices of G share exactly 1 common neighbour

(Erdös-Renyi friendship theorem:) In a finite graph, if any two vertices share precisely one common neighbour, then the graph is a *friendship graph* (or Dutch windmill graph).

Notice that we normally define the set of edges as a subset of $V \times V$. Like how a unary relation defines a subset of the domain (V here), a binary relation defines a subset of $V \times V$.

Why can't we express the following?

1. There is a path between v_1 and v_2
2. G is a connected graph

2 Solutions to translations

2.1 Easy translations

1. $(\forall x)(\forall y)\,(\text{loves}(x,y) \to \neg\text{loves}(y,x))$.
2. Note that I only require the original lover to be a person.
   $(\forall x)(\forall y)(\forall z)\,(\text{person}(x) \wedge \text{loves}(x,y) \wedge \text{loves}(y,z) \to \text{loves}(x,z))$
3. $(\forall x)\,(\text{person}(x) \to \neg\text{loves}(x,x))$
4. $(\forall x)(\forall y)\,(\text{person}(x) \wedge \text{person}(y) \to \text{loves}(x,y) \vee \text{loves}(y,x))$
5. $(\exists x)(\forall y)\text{loves}(y,x)$

   Important: for brevity below, I have declined to state that x, y, z may be people.

6. Realise that *hungriness* defines a subset of *people*. In fact, so does any unary relation.
   $(\exists x)\,(\text{hungry}(x) \wedge (\forall y)\,(y \neq x \wedge \text{hungry}(y) \to \text{loves}(y,x)))$ Note: $y \neq x$ is very important here.
7. $(\exists x)(\exists y)(\exists z)\,(x \neq y \wedge y \neq z \wedge x \neq z \wedge \text{hungry}(x) \wedge \text{hungry}(y) \wedge \text{hungry}(z))$
8. Negate the last statement.
9. Essentially, we are saying there is no finite number of hungry people. However, it would quickly get boring writing a formula similar to the previous parts.
   $\neg(\exists x)\,(\text{hungry}(x) \wedge (\forall y)\,(\text{hungry}(y) \to \text{loves}(x,y)))$ (why?)

   Or use induction. $(\forall x)\,(\text{hungry}(x) \to (\exists y)\,(\text{hungry}(y) \wedge \text{loves}(y,x))) \wedge (\exists z)\text{hungry}(z)$

   Satisfy yourself that these are equivalent.

2.2 Graph translations

1. $(\forall v_1)(\forall v_2)E(v_1,v_2)$
2. $(\forall v_1)(\forall v_2)(\exists v_3)(E(v_1,v_3) \wedge E(v_3,v_2))$

Why not:

1. Essentially, we are talking about the transitive closure of E. If we defined a new 2-ary relation `path`, it might look like this: $path(u,v) : E(u,v) \vee (\exists t)(path(u,t) \wedge E(t,v))$.
   Since we can't define add a new relation symbol to L, you'll quickly see that there's a way to express that there is a path of length n between two vertices, there isn't (finite length) formula that can express that there is a path between two vertices.
2. This follows from the previous section. The definition of a graph being connected is: pick any two vertices, then there is a path between them.