Computational Logic - Assignment 2

Alessandro Marostica

November 3, 2023

Exercise 2

A bijective function is both injective and surjective. We define a function $f : D \to C$. We then express its injectivity and surjectivity respectively as such:

$$\forall x_1 \in D \, \forall x_2 \in D((x_1 \neq x_2) \Longrightarrow f(x_1) \neq f(x_2))$$
$$\forall y \in C \, \exists x \in D(y = f(x))$$

We can then introduce conjunction between the two formulas to descirbe a bijective funciton:

$$\forall x_1 \in D \, \forall x_2 \in D((x_1 \neq x_2) \Longrightarrow f(x_1) \neq f(x_2)) \; \wedge \forall y \in C \, \exists x \in D(y = f(x))$$

Exercise 3

To determine the validity of the given formula $(\forall x P(x) \to \forall x Q(x)) \to \forall x(P(x) \to Q(x))$ we need to demonstrate that it is always true, were we to find a counterexample where the formula is false, the it is not valid. Let's take, for example, the domain to be the set of all natural numbers ($\mathbb{N}$), let P(x) be "x is even" and Q(x) be "x is sivisible by 4". Let's now evaluate the lefthand operand $\forall x P(x) \to \forall x Q(x)$, this can be read as "for each x, x is even then for each x, x is divisible by 4". Both operands of this implication are false and, since a false antecedent can imply anything, this side is vacuously true. Now on to the rightmost operand $\forall x(P(x) \to Q(x))$. This could be read as "for each x such that x is even, x is also divisible by 4", but this is not always true since there are even numbers not divisible by 4 in $\mathbb{N}$. In conclusion, we have a counterexample to the validity of this formula, making it invalid.

Exercise 4

A syllogism applies deductive reasoning to conclude something based on two propositions that are asserted or assumed to be true. Aristotle labeled the different types of premise with letters, A meaning a universal affirmative premise. **BARBARA** is a mnemonic name which indicates the form AAA of the syllogism. We can formalize the BARBARA pattern in FO logic as follows (natural language, then first order logic):
Premises:

All A's are B's: $\forall x(A(x) \to B(x))$
All B's are C's: $\forall x(B(x) \to C(x))$

Conclusion:

All A's are C's: $\forall x(A(x) \to C(x))$

Let's now assume x_0 as an arbitrary element of a domain. Then

$$A(x_0) \rightarrow B(x_0)\ B(x_0) \rightarrow C(x_0)$$

Now, using the transitive property of implication we can deduce:

$$A(x_0) \rightarrow C(x_0)$$

Since x was chosen arbitrarily, we can generalize the result to the entre domain:

$$\forall x(A(x) \rightarrow C(x))$$

□

In a more formal way:

1.		$\forall x(A(x) \rightarrow B(x))$	Premise
2.		$\forall x(B(x) \rightarrow C(x))$	Premise
3.	x_0	$A(x_0) \rightarrow B(x_0)$	$\forall xe$ 1
4.		$B(x_0) \rightarrow C(x_0)$	$\forall xe$ 2
5.		$A(x_0) \rightarrow C(x_0)$	Transitive property
6.		$\forall x(A(x) \rightarrow C(x))$	$\forall xi$ 3 – 5

Exercise 5

The indecidability problem for FO logic states that It is undecidable whether a first order logic formula is provable (or true under all possible interpretations). A quick way do prove this theorem os to show that the decidability problem is at least as hard as the Halting problem, a known undecidable problem. We assume the existence of a decision procedure for determining the validity of FO logic formulas. This way we can now create a program which utilizes said procedure to solve the Halting Problem. This program takes as input another program P and an input string x and determines whether P halts on input x. The constructed program works like this: It constructs a FOL formula based on the behaviour of P on input x. It then uses the FOL validity checker to determine whether this formula is valid. If the FOL formula is valid, it concludes that P halts on input x. If the FOL formula is not valid, it concludes that P does not halt on input x. If the assumed FOL validity checker correctly determines the validity of FOL formulas, then our constructed program can solve the Halting Problem. However, we know the Halting problem to be an undecidable one, therefore are assumption that a general algorithm can determine the validity of FOL formulas must be wrong.