

COMP 681 Formal Methods Spring 2008 Assignment 5

1. You are given the following predicates.

child(x, y) means x is a child of y

sibling(x, y) means x is a sibling of y

female(x) means x is a female

married_to(x, y) means x is married to y

Define the following predicate in terms of these. Note that there are two ways x can be the aunt of y .

aunt(x, y) means x is the aunt of y

2. Use Venn diagrams to evaluate the validity of the following syllogisms. If the syllogism is valid, explain why you cannot draw a diagram in which the premises are true yet the conclusion is false.

a. No A 's are B 's.

No C 's are B 's.

Some A 's are not C 's.

b. No A 's are B 's.

Some C 's are B 's.

Some C 's are not A 's.

3. Translate the following argument into the language of predicate logic and use a Venn diagram to show that it is valid. Be sure to indicate the meanings of the predicate symbols you use, and explain why the argument is valid.

Some cars and some trucks administered by the University are state-owned and some are owned by the university. Everything owned by the University is bargain-priced. Bargain-priced vehicles are dangerous. So, since cars and trucks are vehicles, the University owns some dangerous things.

4. Encode the following statements in the language of predicate logic using the indicated predicates. Be sure to indicate the meanings of the predicate symbols you use. The encodings do not involve nested quantifiers.

a. *Old dogs and tame cats chase mice but don't bother people as long as they're cared for.*

Let

old(x) mean x is old

dog(x) mean x is a dog

tame(x) mean x is tame

cat(x) mean x is a cat

chases_mice(x) mean x chases mice

bother_people(x) mean x bothers people

cared_for(x) mean x is cared for

- b. *Some beer is healthy when drunk sparingly and some is unhealthy when drunk by pregnant women or old men.*

Let

beer(x) mean *x is beer*

healthy(x) mean *x is healthy*

drunk_sparingly(x) mean *x is drunk sparingly*

drunk_by_pgs(x) mean *x is drunk by pregnant women*

drunk_by_om(x) mean *x is drunk by old men*

5. Write abstract programs that specify the following.

- a. *As long as x has a niece or a nephew, make y equal to a niece of x or, if x has no niece, make y equal to a nephew of x.*

Use

sister(x, y) = x is a sister of y.

brother(x, y) = x is a brother of y.

son(x, y) = x is a son of y.

daughter(x, y) = x is a daughter of y.

- b. Consider a line

$$y = m * x + b$$

and a circle

$$(x-x_0)^2 + (y-y_0)^2 = r^2$$

As long as the line and the circle intersect, make the values of u and v such that (u,v) is that point of intersection of the line and the circle that is closest to the origin. (Recall that a line and a circle, if they intersect, intersect at one or two points.) Treat m , b , x_0 , y_0 , and r as global (read-only) variables.

6. For each of the following statements,

- encode it into the language of typed predicate logic and
- translate the result into a formula with untyped quantifiers.

- a. *Courses with prerequisites are not taken by first-semester students who have not taken related high-school courses.*

Let

Course be the type for courses

Student be the type for students

prereq(x, y) mean *x is a prerequisite for y* (intending $x, y : \text{Course}$)

first_semester(x) means *x is a first-semester (student)*

high_school(x) mean *x is a high-school (course)*

related_to(x, y) mean *x is related to y* (intending $x, y : \text{Course}$)

takes(x, y) mean *x (a student) takes y (a course)*

- b. *Some buildings have passed the fire inspection yet, if they catch fire, some in them will be in danger unless the fire is quickly extinguished.*

Let

Building be the type for buildings

Inspector be the type for inspectors

Fire be the type for fires

Person be the type for persons

passed_for_fire(x, y) mean x (an inspector) *has passed* y (a building) *in a fire inspection*

burns_in(x, y) mean x (a fire) *burns in* y (a building)

extinguished_quickly(x) mean x (a fire) *is quickly extinguished*

threatens(x, y) mean x (a fire) *threatens* y (a person)

7. Consider the statement

Any person without a permit who parks a commercial truck on a residential street is fined.

- a. Encode this into the language of predicate logic with types and constraints. Include a constraint with each quantifier. Let

Person be the type for persons

Truck be the type for trucks

Street be the type for streets

has_permit(x) mean x *has a permit*

residential(x) mean x *is residential*

commercial(x) mean x *is residential*

parks(x, y, z) mean x *parks* y *on* z

fined(x) mean x *is fined*

- b. Translate the result in a into a formula without constraints (but with typed quantifiers).
- c. Translate the result in b into a formula without typed quantifiers.