Model Construction and “QuTe”

1. In SL, recall that we used fully-developed, open paths of trees to construct interpretations showing the incorrectness of target sequents corresponding to the opening layout of the tree. In fact, this was the key to our demonstration of the soundness of tableau with respect to the semantics of SL.

2. We can do much the same with QuTe with respect to the semantics of PL. That is, we can use open, fully developed paths of trees to construct models which serve as relevant counterexamples to target sequents. The key is to look to the atomic sentences that appear on the left of an open path, and to assign denotations and predicate extensions that will serve to make them (and only them!) true.

   The construction of an appropriate model from a fully-developed open path is actually quite straightforward. The domain will be a set of objects \{ThingA, ThingB, ThingC, …\} corresponding to (and respectively denoted by) each of the name letters a,b,c… that appear in the formulas along that open path. Thus every object in the domain will be denoted by one, and only one, name letter, and every name letter will have an object that it denotes. Moreover, we determine predicate extensions by looking to the atomic formulas lying along the path. Specifically, if an atomic formula (e.g., Fcb) appears on the left side of the path, then we include the objects denoted by its name letters in the extension of the predicate (<ThingC, ThingB> in the extension of F). If an atomic formula appears on the right, we simply make sure that the objects it denotes does NOT belong in the extension of its predicate letter. Any other atomic formulas can be settled willy-nilly.

3. Don’t forget, however, that the full-development of a path requires that all of the “all-shot” formulas on it must be developed for every name that ever appears on that path. Not only can this lead to a proliferation or “explosion” of developments, some paths might never be able to be developed fully as such development might further uncover new one-shot formulas that need to be developed in turn. Moreover, when one does not know whether or not a given entailment holds, the aim of closing off branches efficiently can work against the aim of systematically developing paths fully, and one might well have little to no idea which is the best strategy to pursue. Indeed, short of actually closing off a path, there is no general algorithmic procedure for determining beforehand whether or not a given setup can be closed, or whether it could simply be developed on forever.

[Decidability]

4. Finally, observe also that this method for constructing models from open branches is not guaranteed to come up with an interpretation that has the fewest members in its domain required to make all the formulas on the left true and the formulas on the right false. Sometimes one can come up with a smaller domain by “recycling” the denotations of names along an open path (that is, using the same object in the domain as the referent for more than one name). However, once again there is no algorithmic procedure for determining just when this can be done. Furthermore, there are also cases in which formulas can all be made appropriately true or false in
a finite domain, even though their development in tableau would result in a non-terminating path containing a never-ending series of one-shot developments and a corresponding never-ending introduction of new names.

[For the following exercises, I’ve abbreviated strings of quantifiers of the same type such as \( \forall x \forall y \forall z \) simply as \( \forall xyz \).]

1. Use fully developed open branches of trees to construct interpretations (models) showing that the following sequents are incorrect.

   a. \( \exists x Fxa, \exists x Gax \models \exists x (Fxa \& Gax) \)

   b. \( \exists x (Fxa \& Gax) \models \exists x (Fxa \& Gxa) \)

   c. \( \exists x \forall y Fxy \models \forall x Fxx \)

   d. \( \forall x Fxa \rightarrow \exists x Fxa, \exists x Fxa \models \exists x Fx \)

   e. \( \exists x (Fx \& \exists y Gxy), \forall xy (Gxy \rightarrow Hx) \models \forall x (Fx \rightarrow Hx) \)

   f. \( \forall x (\forall y Fxy \rightarrow Lx) \models \forall xy (Fxy \rightarrow Lx) \)

   g. \( \forall x (\exists y Fxy \rightarrow \exists y Gxy), \forall x (Hx \rightarrow \exists y Fxy) \models \forall x (Hx \rightarrow \forall y Gxy) \)

2. These are a little less straightforward. Construct models showing that the following sequents are incorrect.

   a. \( \forall xyz ((\neg Rxy \& \neg Ryz) \rightarrow \neg Rxz) \models \forall x Rxx \)

   b. \( \forall xyz ((\neg Rxy \& \neg Ryz) \rightarrow \neg Rxz) \models \forall x \neg Rxx \)

   c. \( \forall xyz ((Rxy \& Ryz) \rightarrow Rxz) \models \forall xyz ((\neg Rxy \& \neg Ryz) \rightarrow \neg Rxz) \)

   d. \( \forall xyz ((Rxy \& Ryz) \rightarrow Rxz), \forall xy (Rxy \rightarrow Ryx) \models \forall xyz ((\neg Rxy \& \neg Ryz) \rightarrow \neg Rxz) \)

   e. \( \exists x Fx, \forall x (Fx \rightarrow \exists y (Rxy \& Fy)), \forall x (Fx \rightarrow \exists y (Rxy \& \neg Fy)) \models \forall xy (Rxy \rightarrow (Fx \lor Fy)) \lor \forall x (Fx \rightarrow \exists y Ryx) \)

   f. \( \forall x (Fx \rightarrow \exists y Gxy) \models \exists x \forall y (Fx \rightarrow Gxy) \)
3. Determine whether the following sets of sentences (or “constraints”) are consistent or inconsistent. If they are consistent, demonstrate whether they are \textit{finitely} consistent by providing a model with a finite domain which satisfies those constraints. [As a challenge, try to find a finite model with the minimum number of members in its domain.]

a. \( \forall x \neg Fxx, \forall x \exists y Fxy \)
b. \( \forall x \neg Fxx, \exists x \forall y Fxy \)
c. \( \forall x y (Fxy \rightarrow \neg Fyx), \forall x \exists y Fxy \)
d. \( \forall x y z ((Fxy \& Fyz) \rightarrow Fxz), \forall x \exists y Fxy \)
e. \( \forall x y z ((Fxy \& Fyz) \rightarrow \neg Fxz), \forall x \exists y Fxy \)
f. \( \forall x y z ((Fxy \& Fyz) \rightarrow \neg Fxz), \forall x y (Fxy \rightarrow \neg Fyx), \forall x \exists y Fxy \)
g. \( \forall x y z ((Fxy \& Fyz) \rightarrow Fxz), \forall x y z ((Fxy \& Fyz) \rightarrow \neg Fyz) \)
h. \( \forall x y z ((Fxy \& Fyz) \rightarrow Fxz), \forall x y z ((Fxy \& Fyz) \rightarrow \neg Fxz), \forall x \exists y Fxy \)
i. \( \forall x y z ((Fxy \& Fyz) \rightarrow Fxz), \forall x y (Fxy \rightarrow \neg Fyx), \forall x \exists y Fxy \)
j. \( \forall x \exists y (Fxy \& \neg Fyx) \)
k. \( \forall x \exists y (Fxy \& \neg Fyx), \exists x y z ((Fxy \& Fyz) \rightarrow Fxz) \)
l. \( \forall x \exists y (Fxy \& \neg Fyx), \forall x \exists y \neg(Fxy \lor Fyx) \)
m. \( \forall x \exists y (Fxy \& \neg Fyx), \forall x \exists y \neg(Fxy \lor Fyx), \forall x y z ((Fxy \& Fyz) \rightarrow Fxz) \)
n. \( \forall x \exists y (Fxy \& \neg Fyx), \forall x \exists y \neg(Fxy \lor Fyx), \forall x y z ((Fxy \& Fyz) \rightarrow Fxz), \forall x \exists y Fxy \)
o. \( \forall x \exists y (Fxy \& \neg Fyx), \forall x \exists y z \neg(Fxy \lor Fyx) \& \neg(Fxz \lor Fzx) \& (Fyz \& \neg Fzy) \)
p. \( \exists x y Fxy, \forall x y (Fxy \rightarrow \exists z(Fxz \& Fzy \& \neg Fyz)) \)
q. \( \forall x \exists y z ((Fxy \& Fxz) \& \neg(Fyx \& \neg Fzx) \& (Fyz \& \neg Fzy)), \forall x y (Fxy \lor Fyx) \)
r. \( \forall x \exists y(z ((Fxy \& Fxz) \& \neg(Fyx \& \neg Fzx) \& (Fyz \& \neg Fzy)), \forall x \exists y (Fxy \lor Fyx) \)
s. \( \forall x \exists y(z ((Fxy \& Fxz) \& \neg(Fyx \& \neg Fzx) \& (Fyz \& \neg Fzy)), \forall x y (Fxy \lor Fyx), \forall x y z ((Fxy \& Fyz) \rightarrow Fxz) \)
t. \( \forall x \exists y z ((Fxy \& Fxz) \& \neg(Fyx \& \neg Fzx) \& (Fyz \& \neg Fzy)), \forall x \exists y (Fxy \lor Fyx), \forall x y z ((Fxy \& Fyz) \rightarrow Fxz) \)
u. \( \forall x \exists y z ((Fxy \& Fxz) \& \neg(Fyx \& \neg Fzx) \& (Fyz \& \neg Fzy)), \forall x y z ((Fxy \& Fyz) \rightarrow Fxz), \forall x y z ((Fxy \lor Fyx) \lor (Fxz \lor Fzx) \lor (Fyz \lor Fyx)) \)
4. Come up with a pair of constraints that may be satisfied by a model with seven (or if you prefer, an eight) item domain, but not in any models with domains of fewer members. [Hint: think about what’s going on in j, l, q, and r above.]