Semantics for Classical Predicate Logic (Part I)[*]

Hans Halvorson

Formal logic begins with the assumption that the validity of an argument depends only on its logical form, and not on its content. In particular, if two arguments have the same logical form, then either they are both valid, or they are both invalid. But what is logical form? Our first approach (before the midsemester break) was to identify logical form with sentence connecting words, such as "and", "or", and "if ... then ...". But we found that there are some arguments whose validity is not due to sentence connecting words; instead, these arguments are valid because of certain quantifier phrases, such as "all ..." or "some ...".

We have now proposed four rules of inference for quantifier phrases: U-Elimination, U-Introduction, E-Introduction, and E-Elimination. It should seem reasonably clear that these rules are acceptable, i.e. will not lead you from true premises to a false conclusion. But since the rules are somewhat subtle, we should provide an independent method for checking their validity. Furthermore, it is not at all clear whether or not these rules form a "complete" system — i.e. if there is a sense in which they can be used to reproduce *any* valid argument.

Let's consider first the question of the safety of these rules, and compare with the case of the sentence connective rules. How do we know that the rule &E is safe? Roughly speaking, we know that in every situation where $P \wedge Q$ is true, P is also true. So, the inference "$P \wedge Q$ therefore P" can never lead from truth to falsity. But what do we mean by "every situation"? We made the notion of a "situation" precise by means of a truth assignment to propositional variables $P, Q, R, \ldots$. We stipulated that these propositional variables do not have any logical relations (e.g. P does not logically imply $-Q$), and so any combination of their truth values is a possibility we must consider.

Conversely, how do we know that the argument form "$P \vee Q$ therefore P" is *invalid*? We know that this argument form is is invalid because there are *instances* of that form that have true premises and false conclusions, for example:

> Tilghman is either Harvard's or Princeton's President. Therefore Tilghman is Harvard's President.

In fact, the point of this example is that there are sentences P and Q such that $P \vee Q$ is true, while P is false. So, to give a counterexample, we only need specify the corresponding row in the truth table for "$\vee$".

[*]Version 0.1, revised April 6, 2012. Please report typos to: hhalvors@princeton.edu.

It is obvious how to give "informal" counterexamples to predicate logic arguments: just replace predicate letters with real predicates, and replace name letters with real names. For example, to show that "$\exists xFx$ therefore Fm" is invalid, simply note that somebody is a Princeton student ($Fx \equiv x$ is a Princeton student) but John Tesh is not a Princeton student ($m \equiv$ John Tesh). The main question of this Chapter is whether this heuristic method for finding counterexamples can be transformed into something systematic and efficient, something like truth tables.

So far the following is clear: to give a counterexample to a predicate logic argument, one must replace the predicate letters with real predicates, and the name letters with real names. It might then seem that once we replace the predicate letters with predicates and the name letters with names, then each sentence is either true or false. But actually, there is one small complication: unless we specify a "domain of quantification," then it is not clear what the quantifiers mean. For example, is it true or false that there was a Danish prince who loved Ophelia? Is it true or false that there are infinitely many prime numbers? Is it true that the color red exists?

Now you might say that "$\forall x$" should always be translated as "For all x", where the "all" is taken to apply to just those things that actually do exist. So then "$\forall xFx$" would simply mean that all actually existing things are F. But this seems too strict. For example, it seems that we should also be able to use quantifiers to translate sentences such as "all of Jane Austen's characters are complex."

In order to deal with this and other related issues, we require that each time an argument is translated into quantifier form, one must provide a "domain of quantification," abbreviated "DoQ". For example, we can show that the argument form

$$\frac{\forall x(Fx \vee Gx)}{\forall xFx \vee \forall xGx} \qquad (*)$$

is invalid by specifying the natural numbers as the DoQ, and by setting "$Fx \equiv x$ is even" and "$Gx \equiv x$ is odd."

1 From Predicates to Sets

In order to give counterexamples in propositional logic, we employed a simplifying strategy: replace sentences with their truth values. In predicate logic, we will also employ a simplifying strategy: replace predicates with their extensions.

Definition. The *extension* of a predicate is just the collection of those things to which the predicate truly applies.

Example. The extension of the predicate "is even" (relative to the domain of natural numbers) is just the collection of even numbers. But if the DoQ is the set $\{1, 2, 3\}$ then the extension of the predicate "is even" is the singleton set $\{2\}$.

In order to give *formal* counterexamples to invalid arguments, we will make use of the mathematical theory called "set theory." Roughly speaking, set theory is the study of collections of things.

We use curly braces "{" and "}" around a list of things in order to denote the *set* of those things. For example, the set consisting of the numbers $2, 6$, and 17 is denoted by:

$$\{2, 6, 17\}$$

Axiom (Axiom of Extensionality:)**.** Two sets A and B are equal, written $A = B$, just in case they have the same members.

For example $\{2, 4\}$ and $\{4, 2\}$ are two different ways to write the unique set whose elements are 2 and 4. Similarly, $\{2, 2, 4\} = \{2, 4\}$ since the former set contains exactly the same elements as the latter.

Definition (Member)**.** The number 2 is a member of the set $\{2, 6, 17\}$. We can indicate this fact by writing $2 \in \{2, 6, 17\}$, where "$\in$" is shorthand for "is a member of."

Definition (Subset)**.** Every member of the set $\{2, 17\}$ is also a member of the set $\{2, 6, 17\}$. So, we say that $\{2, 17\}$ is a *subset* of $\{2, 6, 17\}$. And if we don't feel like writing out the words, we might just write $\{2, 17\} \subseteq \{2, 6, 17\}$ as a shorthand.

Note that is does *not* make sense to write $2 \subseteq \{2, 6, 17\}$, because 2 is not itself a set. On the other hand, it does make sense to write $\{2\} \subseteq \{2, 6, 17\}$. The number "2" and the singleton set "$\{2\}''$ are two different things.

Definition (Empty Set)**.** There is a set that does not contain anything at all. This set is called the *empty set*, and it is denoted by $\emptyset$. The empty set is the one and only set that is contained in all other sets.

Definition. If A and B are sets, we use $A \cap B$ to denote the set that consists of those things that are in both A and B. We use $A \cup B$ to denote the set that consists of those things that are in either A or B. Finally, we use $B - A$ to denote those things that are in the set B but not in the set A.

2 Formal Counterexamples

Now we revisit the concepts of counterexamples and interpretations, this time with an eye toward precision and rigor.

Definition. An *interpretation* of a sentence (or sentences) of the predicate calculus consists of:

(a.) A set X, called the *domain of quantification.*

(b.) An assignment $\underline{\text{Ref}}$ of names to elements of X.

(c.) An assignment $\underline{\text{Ext}}$ of predicate letters to subsets of X.

Definition. A *formal counterexample* to an argument in predicate logic is an interpretation where the premises of the argument are true, and the conclusion of the argument is false.

I have not yet given a precise definition of when a sentence is true relative to an interpretation. I have not done so because the definition turns out to be quite complicated, and your intuitions will be sufficient for most cases.

Example. Consider the interpretation with domain $X = \{1, 2, 3, 4\}$ and

$$\begin{aligned}&\text{Ref}(m) = 1, \quad \text{Ref}(n) = 1, \quad \text{Ref}(o) = 2,\\ &\underline{\text{Ext}}(Fx) = \{1, 2, 3\} \quad \underline{\text{Ext}}(Gx) = \{3, 4\} \quad \underline{\text{Ext}}(Hx) = \emptyset.\end{aligned}$$

Then, it follows that:

- $\exists x(Fx \wedge Gx)$ is true relative to this interpretation, since the element 3 is in both $\underline{\text{Ext}}(Fx)$ and $\underline{\text{Ext}}(Gx)$.
- $\forall x(Fx \to Gx)$ is false relative to this interpretation, since the element 1 is in $\underline{\text{Ext}}(Fx)$ but not in $\underline{\text{Ext}}(Gx)$.
- $\forall x(Fx \vee Gx)$ is true relative to this interpretation, since every element in the domain is in either $\underline{\text{Ext}}(Fx)$ or $\underline{\text{Ext}}(Gx)$.
- $\forall x(Hx \to Gx)$ is true relative to this interpretation, since $\underline{\text{Ext}}(Hx)$ is the empty set.

Problem. Give a formal counterexample to the argument:

$$\forall x(Fx \vee Gx) \vdash (\forall x Fx \vee \forall x Gx).$$

Solution: Let $X = \{1, 2\}$, let $\underline{\text{Ext}}(Fx) = \{1\}$ and $\underline{\text{Ext}}(Gx) = \{2\}$. Since all elements of X are either in $\underline{\text{Ext}}(Fx)$ or $\underline{\text{Ext}}(Fx)$, $\forall x(Fx \vee Gx)$ is true relative to this interpretation. But since $\underline{\text{Ext}}(Fx)$ is not equal to X, $\forall x Fx$ is not true relative to this interpretation. Similarly, $\forall x Gx$ is not true relative to this interpretation, and so $\forall x Fx \vee \forall x Gx$ is not true relative to this interpretation.

Problem. Give a formal counterexample to the argument with no premises and conclusion $\forall x \forall y(Fx \to Fy)$.

Solution: Let $X = \{1, 2\}$, and let $\underline{\text{Ext}}(Fx) = \{1\}$. Then it is not true that for all x, y in the DoQ, if Fx then Fy. In particular, $1 \in \underline{\text{Ext}}(Fx)$ but $2 \notin \underline{\text{Ext}}(Fx)$.

2.1 Specifying interpretations with tables

It can be boring to write out the interpretations of a bunch of predicates. So, in order to spice up life, we give here an alternative way to specify the interpretation of predicate letters. (This method only works when the domain X is a finite set.)

First we list the predicate letters in the top row of the table, and we list the elements of the domain in the first column of the table. We then write "+" in cell (i, j) if the predicate in column j applies to the object on row i. Otherwise, we write "−" in cell (i, j).

Problem. Give an interpretation for a problem involving the predicate letters F, G, H, and the names m, n, o.

Solution:

$$X = \{1, 2, 3\}, \qquad \text{Ref}(m) = 1, \quad \text{Ref}(n) = 1, \quad \text{Ref}(o) = 2.$$

	F	G	H
1	+	−	−
2	+	+	−
3	+	−	−

Instead of drawing the table, we could have just written:

$$\underline{\text{Ext}}(Fx) = \{1, 2, 3\}, \quad \underline{\text{Ext}}(Gx) = \{2\}, \quad \underline{\text{Ext}}(Hx) = \emptyset.$$

But tables are fun.

When giving an interpretation, you must be clear and explicit about which subsets are assigned to which predicate letters. But given that you satisfy the requirements of clarity and explicitness, you can use whatever method you want to specify your interpretation.

Note. An interpretation must assign the same subset of the domain to both Fx and Fy, even though they have different variables. In other words, the interpretation gives the extension of the predicate letter F, and doesn't care about what variable we put after F. We will deal later with the tricky case of sentences that involve the same predicate letter with different variables — e.g., $\forall x \forall y(Fx \to Fy)$ and $\exists x(Fx \to \forall y Fy)$.

3 Semantic Properties and Relations

Recall that within propositional logic, we defined a bunch of special kinds of sentences (e.g., tautologies, inconsistencies), and a bunch of special logical relationships between sentences (e.g., implies, is subcontrary to). Actually, these concepts are defined in terms of interpretations, and so they extend naturally to predicate logic. For example, we would define a predicate logic sentence as *tautologous* just in case it is true relative to every predicate logic interpretation.

Definition.

- If ϕ is false relative to some interpretation, then ϕ is said to be *falsifiable.*
- If ϕ is false relative to every interpretation, then ϕ is said to be *inconsistent.*
- If ϕ is true relative to some interpretation, then ϕ is said to be *consistent.*
- Let Γ be a set of sentences. If there is an interpretation relative to which every sentence in Γ is true, then Γ is said to be *consistent.* Otherwise, Γ is said to be *inconsistent.*
- A set Γ of sentences *logically implies* a sentence ψ if there is no interpretation that makes all of Γ true while making ψ false. That is, there is no *formal counterexample.* In that case, the argument with premises Γ and conclusion ψ is *valid.*
- Two sentences ϕ, ψ are *logically equivalent* if they have the same truth value relative to every interpretation.

Example. $\forall x(Fx \vee \neg Fx)$ is tautologous, because in any interpretation, each object in the domain is either in $\underline{\text{Ext}}(Fx)$, or is in the complement of $\underline{\text{Ext}}(Fx)$.

Example. $\exists x(Fx \wedge \neg Fx) \to Gm$ is tautologous, because $\exists x(Fx \wedge \neg Fx)$ is inconsistent.

4 Semantic Problems and Answers

An answer to a problem requiring the presentation of an interpretation is best seen as having two parts, as follows: (1) state the domain of your proposed interpretation, and present (using one of the above methods) the interpretation of the names and predicate symbols; (2) state the truth values of the various sentences, defend your claim that they have those truth values, and (most importantly) be explicit as to how this information solves the problem you began with. Let's consider a couple of examples of solved problems.

Problem. Show that $\exists x Fx \to \forall x Fx$ is not tautologous.

Solution:

1. Let $X = \{1, 2\}$, and let $\underline{\text{Ext}}(Fx) = \{1\}$.

2. The sentence is false relative to this interpretation: Since 1 is in $\underline{\text{Ext}}(Fx)$, $\exists xFx$ is true relative to this interpretation. However, since 2 is not in $\underline{\text{Ext}}(Fx)$, $\forall xFx$ is false relative to this interpretation. Therefore (by truth tables) $\exists xFx \to \forall xFx$ is false relative to this interpretation. Since the sentence is false relative to *some* interpretation, it is not tautologous. □

Problem. Show that the sentence from the previous problem is consistent (true relative to some interpretation).

Solution:

1. Let $X = \{1\}$, and let $\underline{\text{Ext}}(Fx) = \{1\}$.
2. Since $\underline{\text{Ext}}(Fx) = X$, it follows that $\forall xFx$ is true relative to this interpretation, and therefore (by truth tables) $\exists xFx \to \forall xFx$ is true relative to this interpretation. Since the sentence is true relative to some interpretation, it is consistent.

5 Systematic Searches for Interpretations

Solving a semantic problem (e.g. "is the argument with premises Γ and conclusion ϕ valid?") requires one to check all possible interpretations. Now in propositional logic, there are only finitely many interpretations of a finite set of sentences (namely, 2^n interpretations, where n is the number of propositional variables). But in predicate logic, there are infinitely many interpretations of any sentence. So, it is simply impossible to search through all interpretations of a predicate logic sentence.

In fact, it has been proven that the task of deciding if a predicate logic argument is valid cannot be "automated" — i.e. there is no algorithm for solving the validity problem for predicate logic arguments. (This interesting result would be discussed in detail in a more advanced logic course, such as PHI 312 or PHI 321.) However, in the special case of sentences containing only predicates with a single variable (i.e. no relation symbols), there are algorithms for solving the validity problem. These algorithms are the topic of the current section.

We will consider three successively stronger algorithms for testing the consistency of a finite set of sentences. If you feed a finite set Δ of sentences into the algorithm then either it will return the verdict "Inconsistent", or it will return an interpretation that makes all the sentences true. Of course, if we can solve the consistency problem then we can solve the validity problem. So, these algorithms solve the validity problem for arguments with certain types of sentences.

5.1 Algorithm A

The first algorithm works for sentences that contain no quantifiers.

1. Perform a truth table test on the sentences in Δ. If there is no truth assignment that makes them all true, then return "Inconsistent." If there is a truth assignment j that makes them all true, then proceed to Step 2.

2. Put the names that occur in the sentences in Δ in a list. Let the DoQ be $\{1, \ldots, n\}$ where n is the number of names. For each predicate letter F, and for each $i = 1, \ldots, n$, let $i \in \underline{\text{Ext}}(Fx)$ just in case $\mathsf{j}(Fa_i) = T$, where a_i is the i-th item on the list of names.

5.2 Algorithm B

The second algorithm works for simple quantified sentences.

Definition. A *simple monadic sentence* is one that contains no relation symbols, and a single quantifier whose scope is the entire sentence.

For example, $\forall x((Fx \wedge Gx) \rightarrow \neg Hx)$ is a simple monadic sentence.

Definition. If ϕ is a simple monadic sentence, we let ϕ^a denote the instance of ϕ that is obtained by taking off the initial quantifier and replacing all instances of the variable with the name a.

For example, if ϕ is the sentence $\exists x(Fx \wedge Gx)$ then ϕ^a is the sentence $Fa \wedge Ga$.

We now show how to determine if a collection $\phi_1, \ldots, \phi_n$ of simple monadic sentences is consistent.

1. Reorder the sentences if necessary so that the first m sentences begin with an existential quantifier, and the remaining sentences begin with a universal quantifier.

2. If none of the sentences begins with an existential quantifier, then choose one arbitrary name a. Now put the sentences $\phi_1^a, \ldots, \phi_n^a$ into Algorithm A.

3. If there is at least one sentence beginning with an existential quantifier, then choose m arbitrary names $a_1, \ldots, a_m$ (one per existential sentence). Now put the following sentences into Algorithm A:

$$\phi_1^{a_1}, \ldots, \phi_m^{a_m} \qquad \text{(instances of existential sentences)}$$

$$\begin{array}{ll} \phi_{m+1}^{a_1}, \ldots, \phi_{m+1}^{a_m} & \\ \vdots & \text{(instances of universal sentences)} \\ \phi_n^{a_1}, \ldots, \phi_n^{a_m} & \end{array}$$

That last step could use some explanation: for each distinct existential sentence ϕ_i, we choose a distinct name a_i and construct the corresponding instance $\phi_i^{a_i}$. Then for each universal sentence ϕ_j, we construct m instances $\phi_j^{a_1}, \dots, \phi_j^{a_m}$, one for each name that was introduced for an existential sentence.

5.3 Algorithm C

Before you read this section, you need to know how to transform a sentence into an equivalent sentence in disjunctive normal form. For this, see Appendix A of Lemmon's book.

The third algorithm is a simple extension of Algorithm B; it works for truth functional combinations of simple monadic sentences.

Definition. A *pure monadic sentence* is a sentence that is a truth-functional combination of simple monadic sentences.

For example,

$$\exists x Fx \wedge \forall y(Gy \to Fy),$$

is a pure monadic sentence. On the other hand,

$$\forall x(Fx \to (existsyFy)$$

is not simple monadic, since it has nested quantifiers.

1. Transform $\phi_1 \wedge \cdots \wedge \phi_n$ into disjunctive normal form. The result will be of the form $\psi_1 \vee \cdots \vee \psi_m$, where each ψ_i is a conjunction of simple monadic sentences and negated simple monadic sentences.
2. Within each disjunct $\psi_1, \dots, \psi_n$, change each negated simple monadic sentence into an equivalent simple monadic sentence using the quantifier-negation equivalences.
3. For $i = 1, \dots, n$ put the modified conjuncts in ψ_i into Algorithm B. If Algorithm B yields an interpretation, then output this interpretation. If Algorithm B says Inconsistent, then repeat this step until all disjuncts have been checked, and if all disjuncts fail the test, output Inconsistent.

5.4 The small domain method

Although Algorithm C always gives the correct answer, it has the drawback that it does not match the way that we normally reason when we try to decide if some sentences are consistent. The method in this section is perhaps less "clean" than Algorithm C (e.g. it might be more difficult to implement it in a computer program), but it might be somewhat more intuitive.

The "small domain method" is based on the following fact that has been proven by paid logicians:

Fact: If a pure monadic sentence is consistent, then it is true relative to some "small" domain. (In fact, the size of the domain needed is a function of the number of predicates and variables in the sentences.)

Thus, to test a pure monadic sentence for consistency, do the following: Find a quantifier-free sentence that is equivalent to the original sentence relative to a domain with one object; test this resulting sentence for consistency using ordinary truth tables. If the resulting sentence is consistent, you are done — the original sentence is true in that domain. If the resulting sentence is inconsistent, then start over again with a domain with two individuals. If the resulting sentence is consistent, you are done — the original sentence is true in that domain. If the resulting sentence is inconsistent, then repeat the procedure in a domain with three individuals, etc., until either you find an interpretation relative to which the sentence is true, or you conclude that there is no such interpretation. For when you are entitled to draw the latter conclusion, see the last section.

5.4.1 Finding equivalent quantifier-free sentences

Suppose that the domain X has only three objects $1, 2, 3$. In this case, the universal statement "$\forall x Fx$" is equivalent to a conjunction: "1 is an F, 2 is an F, and 3 is an F." Similarly, the existential statement "Something is a F" is equivalent to the disjunction "Either 1 is an F, or 2 is an F, or 3 is an F." In sum, when there are only finitely many things, we can (by naming each object) translate every simple monadic sentence into a sentence without quantifiers.

The same sort of equivalences also hold for truth-functional combinations of simple monadic sentences (i.e., pure monadic sentences). For example, if we take the standard association of names with numbers:

$$\mathrm{Ref}(a) = 1 \quad \mathrm{Ref}(b) = 2 \quad \mathrm{Ref}(c) = 3,$$

then $\forall x Fx \to \exists x Fx$ is equivalent to:

$$(Fa \wedge Fb \wedge Fc) \to (Fa \vee Fb \vee Fc).$$

Generally, in order to obtain a quantifier-free sentence that is equivalent (relative to some finite domain) to a sentence ϕ, you should do the following:

1. Disassemble ϕ into truth-functionally simple components;
2. If any of these components are quantified statements, expand them into equivalent conjunctions or disjunctions;
3. Put the expanded statements back together again using the original truth-functional connectives.

Problem. Find a quantifier-free sentence that is equivalent in a domain with three objects to $\forall x(Fx \rightarrow Gx) \vee \exists x(Hx \wedge Mx)$.

Solution: Since the main operator is a disjunction "$\vee$", we separate the original sentence into $\forall x(Fx \rightarrow Gx)$ and $(\exists x)(Hx \wedge Mx)$. Since these two statements are quantified statements, we expand them as follows:

$$\begin{aligned}
\forall x(Fx \rightarrow Gx) &\equiv (Fa \rightarrow Ga) \wedge (Fb \rightarrow Gb) \wedge (Fc \rightarrow Gc) \\
\exists x(Hx \wedge Mx) &\equiv (Ha \wedge Ma) \vee (Hb \wedge Mb) \vee (Hc \wedge Mc)
\end{aligned}$$

Finally, we put these back together again using $\vee$ to obtain:

$$\begin{aligned}
&[(Fa \rightarrow Ga) \wedge (Fb \rightarrow Gb) \wedge (Fc \rightarrow Gc)] \\
&\quad \vee \quad [(Ha \wedge Ma) \vee (Hb \wedge Mb) \vee (Hc \wedge Mc)].
\end{aligned}$$

□

Problem. For each of the sentences $\forall xFx, \exists xFx \rightarrow \exists xGx, \forall x(Gx \vee Fx)$, find a quantifier-free sentence that is equivalent relative to a domain containing two individuals.

Solution:

$$\begin{aligned}
\forall xFx &\equiv Fa \wedge Fb \\
\exists xFx \rightarrow \exists xGx &\equiv (Fa \vee Fb) \rightarrow (Ga \vee Gb) \\
\forall x(Gx \vee Fx) &\equiv (Ga \vee Fa) \wedge (Gb \vee Fb)
\end{aligned}$$

5.4.2 Testing for consistency

Relative to domains with a finite number of individuals, we can test the consistency of a sentence of monadic PC using ordinary truth tables: Just translate the sentence into an equivalent quantifier-free sentence.

Problem. Could $(\exists x)Fx \wedge (\exists x)\neg Fx$ be true relative to a domain with only one object?

Solution: If there were only one thing in the universe, then $(\exists x)Fx \wedge (\exists x)\neg Fx$ would be equivalent to $Fa \wedge \neg Fa$, which is a contradiction. So, no; this sentence cannot be true if there is only one object in the domain. □

Problem. Could $(\exists x)Fx \wedge (\exists x)\neg Fx$ be true if there were exactly two objects in the domain?

Solution: In this case $(\exists x)Fx \wedge (\exists x)\neg Fx$ would be equivalent to:

$$(Fa \vee Fb) \wedge (\neg Fa \vee \neg Fb).$$

A truth table test shows that this sentence is consistent, e.g. the truth assignment j such that $\mathsf{j}(Fa) = \mathrm{T}$ and $\mathsf{j}(Fb) = \mathrm{F}$. So, yes; $\exists xFx \wedge \exists x\neg Fx$ could be true if there were two things in the domain. □

5.4.3 Putting it all together

Problem. Use the small domain method to show that $\forall x(Fx \to \neg Fx)$ is consistent.

Solution: In a domain with one individual, the original statement becomes $Fa \to \neg Fa$. This is true when Fa is false. So, $\forall x(Fx \to \neg Fx)$ can be true in a domain with one individual. In particular, here is an interpretation which shows explicitly that $\forall x(Fx \to \neg Fx)$ is consistent:

$$X = \{1\}, \qquad \underline{\text{Ext}}(Fx) = \emptyset.$$

□

Problem. Use the small domain method to determine if the following argument is valid:

1. $\exists x Hx \to \forall x(Fx \to Gx)$
2. $\exists x Fx \qquad\qquad // \exists x Hx \to \forall x Gx$

Solution: Relative to a domain with one member, we get the argument:

1. $Ha \to (Fa \to Ga)$
2. $Fa \qquad\qquad // Ha \to Ga$

A truth table test shows that if the two premises are true, then the conclusion must also be true. So, we now try a domain with two members. Relative to a domain with two members, we get the argument:

1. $(Ha \vee Hb) \to ((Fa \to Ga) \wedge (Fb \to Gb))$
2. $Fa \vee Fb \qquad\qquad // (Ha \vee Hb) \to (Ga \wedge Gb)$

A truth table test shows that the premises can be true while the conclusion is false. For example, we could choose the truth-assignment:

$$\mathsf{j}(Ha) = \text{T} \qquad \mathsf{j}(Hb) = \text{F}$$

$$\mathsf{j}(Ga) = \text{T} \qquad \mathsf{j}(Gb) = \text{F}$$

$$\mathsf{j}(Fa) = \text{T} \qquad \mathsf{j}(Fb) = \text{F}$$

Therefore, the original argument is invalid. □

5.4.4 Decision procedures

Suppose that you need to determine whether a sentence ϕ is consistent. You check a domain with one individual, and ϕ is false. You check a domain with two individuals and ϕ is false. You check a domain with three individuals and ϕ is false. Surely you cannot check domains of all sizes! When, if ever, are you entitled to conclude that there is *no* interpretation relative to which ϕ is true? Amazingly, it has been shown that:

> *Suppose that ϕ is a pure monadic sentence with n predicate letters. If ϕ is consistent, then there is an interpretation $\mathscr{I}$ whose domain has less than or equal to 2^n elements, and ϕ is true relative to $\mathscr{I}$.*[1]

So, to take a specific case, if ϕ has two predicate letters, then for ϕ to be consistent, it must be true in some domain with at most 4 objects! It follows that the small domain method is a "decision procedure" for the consistency of pure monadic sentences: it will answer any question you have about consistency in a finite amount of time.

[1]Compare with Boolos and Jeffrey, *Computability and Logic*, p. 250.