Fun with Algorithms

(by Brett Sherman)

Algorithm A

Use: To test for the consistency of a set of sentences containing only propositional logical vocabulary (i.e. no quantifiers; only $\&$, $\vee$, $\rightarrow$, etc.).

Algorithm: For any set S of sentences containing only propositional logical vocabulary, try to assign truth-values to all of the elementary sentences that make up the sentences in S in such a way as to make all of the sentences in S true. If there is such an assignment, then the sentences in S are consistent. If not, then they are inconsistent.

Note: In propositional logic, the elementary sentences are of the form P, Q, R, etc. In predicate logic, the elementary sentences are of the form Fa, Fb, Ga, Gb, etc.

Example: Test for the consistency of the following two sentences:

$$Fa \rightarrow -(Gb \vee Fb), \qquad Ga \,\&\, -Fb$$

Elementary sentence	Possible assignment of truth-values
Fa	F
Ga	T
Fb	F
Gb	F

Since there is a possible assignment (in fact, there are several), the two sentences are consistent.

Algorithm B

Use: To test for the consistency of a set of simple monadic sentences (note: a sentence is simple monadic just in case the main operator of the sentence is a quantifier and it doesn't contain any other quantifiers or names within the scope of the main quantifier).

Algorithm: (1) Take each sentence beginning with an existential quantifier and give an instance of the sentence such that each sentence contains a different arbitrary name. (2) Then take each sentence beginning with a universal quantifier and produce an instance of the quantifier for each name used in step (1). If there are no sentences beginning with existential quantifiers, then you need only one instance of each universal sentence. (3) Take the list of instances and plug that set of sentences into Algorithm A. If those sentences are consistent then the set of simple monadic sentences is consistent. If not, then they aren't.

Example: Test for the consistency of the following three sentences:

$$(\exists x)(Fx \& Gx), \qquad (\exists x)(Gx \vee Hx), \qquad (x)(Fx \rightarrow Hx)$$

Step (1): instantiate each existential sentence using a different name:

$$Fa \& Ga, \qquad Gb \vee Hb$$

Step (2): instantiate each universal sentence using all of the names from Step (1):

$$Fa \rightarrow Ha, \qquad Fb \rightarrow Hb$$

Step (3): plug the new list of instances into Algorithm A.

Elementary sentence	Truth-assignment
Fa	T
Ga	T
Ha	T
Fb	T
Gb	T
Hb	T

Since there is an interpretation that makes all of the instances true, then the three simple monadic sentences are consistent.

Bonus: You can build an interpretation that makes the simple monadic sentences true just from the information in Step (3). First, for each name used in the list of elementary sentences, put an object in the domain. In the above example, the only names used are 'a' and 'b', so we only need two objects in the domain. So we'll let the domain = $\{1, 2\}$. Second, assign each name to one of the objects in the domain.

So we'll say:

$$a \rightarrow 1, \qquad b \rightarrow 2$$

Finally, to determine the extension of the predicates, just look at the truth-value of the elementary sentences. If a given sentence is true, then put the object named into the extension of the predicate used in the sentence. So, for example, if Fa is false, then leave 1 out of the extension of F. In the above example, since every elementary sentence is true, the extensions would look like this:

$$\text{Ext}(F) = \{1, 2\} \qquad \text{Ext}(G) = \{1, 2\} \qquad \text{Ext}(H) = \{1, 2\}$$

You can verify that this interpretation—the specification of the domain, the assignment of names to objects, and the specification of the extensions of predicates—will make the original set of simple monadic sentences true.

Algorithm C

Use: To test for the consistency of pure monadic sentences (note: a pure monadic sentence is a sentence that is a truth-functional combination of simple monadic sentences).

Algorithm: (1) Take your pure monadic sentences and conjoin them into one giant sentence of the form $P\&Q\&R$, etc. (2) Treating the simple monadic sentences as elementary, put the entire giant sentence into Disjunctive Normal Form. (3) Drive in any negations that are on the outside of quantifiers. (4) You've now got a big disjunction of conjunctions—that is, a sentence of the form $(P\&Q) \vee (R\&S)$, etc, where each sentence letter is a simple monadic sentence. Take each disjunct one at a time and plug the simple monadic sentences into Algorithm B. If any one disjunct is consistent, then the whole giant sentence is consistent, and the original set of pure monadic sentences is consistent. So if the first disjunct is consistent, you're done. If not, you have to move to the next one. If none of them are consistent, then the whole thing is inconsistent.

Example: Test for the consistency of the following two sentences:

$$(x)(Fx) \vee (\exists x)(Gx), \qquad -(x)(Fx \rightarrow Gx) \rightarrow (x)(-Fx)$$

Step (1): Conjoin the sentences:

$$((x)(Fx) \vee (\exists x)(Gx))\&(-(x)(Fx \to Gx) \to (x)(-Fx))$$

Step (2): To put this into DNF, I'll first assign each simple monadic sentence an elementary sentence letter:

$$(P \vee Q)\&(-R \to S)$$

Now I'll put this into DNF:

$$(P \vee Q)\&(R \vee S)$$

$$((P \vee Q)\&R) \vee ((P \vee Q)\,\&\,S)$$

$$(P\&R) \vee (Q\&R) \vee (P\,\&\,S) \vee (Q\,\&\,S)$$

Now I'll substitute the simple monadic sentences back in:

$$\begin{aligned}&((x)(Fx)\&(x)(Fx \to Gx)) \vee ((Ex)(Gx)\&(x)(Fx \to Gx))\\ &\quad \vee((x)(Fx)\& -(x)(-Fx)) \vee ((Ex)(Gx)\,\&\,(x)(-Fx))\end{aligned}$$

Step (3): Drive in any negations outside of quantifiers:

$$\begin{aligned}&((x)(Fx)\&(x)(Fx \to Gx)) \vee ((\exists x)(Gx)\&(x)(Fx \to Gx))\\ &\quad \vee((x)(Fx)\&(Ex)(Fx)) \vee ((\exists x)(Gx)\&(x)(-Fx))\end{aligned}$$

Step (4): Plug in each disjunct into Algorithm B until you get a winner.

So we'll first test for the consistency of the following two sentences:

$$(x)(Fx), \qquad (x)(Fx \to Gx)$$

There are no existential sentences, so we only need one name to instantiate these:

$$Fa, \qquad Fa \to Ga$$

And if we plug this list into Algorithm A, we can easily see that it is consistent by making both Fa and Ga true. So the first disjunct is consistent, and so the entire disjunction is consistent. Therefore, the two pure monadic sentences are consistent.

Testing for Validity

An argument from premises $\phi_1, \ldots, \phi_n$ to conclusion ψ is valid iff the sentences $\phi_1, \ldots, \phi_n, -\psi$ are inconsistent. So to determine whether an argument from premises $\phi_1, \ldots, \phi_n$ to conclusion ψ is valid, just make a list of sentences $\phi_1, \ldots, \phi_n, -\psi$. Drive in any negations. Look to see whether you've got simple monadic sentences or pure monadic sentences, and test for the consistency of the set using the appropriate algorithm.

If the sentences are consistent, then the original argument is invalid

If the sentences are inconsistent, then the original argument is valid.