Does Everything Follow from Contradiction? Yes and No:

A *RE*-introduction to the Epistemology of Logic or

What Traditional Logicians Should Have Said About Modern Logic (and Vice Versa)

The issue I am raising belongs to the epistemology of logic rather than to logic proper: how do we know that a computational procedure establishing valid logical consequence is correct according to the rules of the computational method? (Except where more specificity is relevant, for the sake of brevity I will use 'rules' as a generic term covering any device computational methods can use as a means of establishing validity: axioms, laws, inference principles, definitions, matrices, trees, etc.) How do we know that in a decision procedure, for example, or calculational proof we have correctly applied the rules of the method? Knowledge of computational correctness may not seem to present much of a mystery. But I will argue that an analysis of how we know that even just one step in a computational process is correct requires a positing of two distinct epistemic types: (1) knowledge that certain statements are necessarily true and (2) knowledge of certain contingent truths (or at least truths that need not be epistemically necessary).

Recognizing that a step correctly instantiates a rule is not the same as recognizing that the rule or any formula we arrive at by means of it expresses a necessary truth or is connected in some other interesting way to necessary truth. To this extent, those who hold that modern logic does not require the existence or knowledge of necessary truths are right. But rules do not apply themselves. Knowledge that a computational procedure correctly applies a rule involves making an inference whose validity we recognize, and we can recognize the validity of an inference if and only if we recognize the inference as an instance of a necessary logical truth, a truth about inferential validity. So we need knowledge of necessary logical truths to account for our knowledge of the correctness of any computational procedure. And it does not matter whether the computational procedure we recognize to be correct is at the metatheorem or theorem level, or whether the procedure bears on formulas that are schemata or sentences. The epistemic questions I am asking are

the same for any use of computational methods. (I will also discuss why we sometimes need epistemically necessary truths of a mathematical, not logical, kind.)

Of course, we can express the rules to be followed in computations by means of directions that are neither true nor false, like 'Delete a true formula as a component of a conjunction.' Being neither true nor false, such rules telling us steps allowed in computational procedures do not express something logically necessary. But we construct our methods by choosing rules on the basis of our awareness of the necessity of truths like 'Deleting a truth as a component of a conjunction preserves truth value.'[1]

The computational methods I have in mind are those using formal languages and formal definitions of logical consequence, whether semantic or proof-theoretic. I will not need a complete account, in terms of necessary and sufficient conditions, of what makes a method a computational method. I only need to characterize computational methods sufficiently to show that, whatever else may be true of them, knowledge of computational correctness presupposes knowledge of necessary truth GAINED NON-COMPUTATIONALLY.

The concept of necessary truth that will be pertinent to this inquiry is that of a statement whose denial is contradictory, a statement whose denial either is or implies a contradiction. I will not define a necessary truth as true in all possible worlds because the computational use of possible world semantics is an instance of the kind of knowledge whose epistemic pre-conditions I am investigating. Knowledge of the correctness of a computational step in possible world semantics requires a grasp of necessary truth gained non-computationally just as much as any knowledge of computational correctness does. To understand our knowledge of it we need an account of necessary truth that is philosophically deeper — I do not say logically deeper — than the possible worlds account. I will offer such an account.

In addition to the recognition that some statements of logic are true on penalty of contradiction, knowledge of computational correctness requires the recognition that those statements are bivalently true. So those truths will belong both to traditional logic and to

classical modern logic (which I will collectively call standard logic; 'standard logic' will not be synonymous with classical modern logic). But knowledge of those statements is required for recognizing the computational correctness of procedures in logics that are multivalued, paraconsistent, dialethic, or any other nonstandard form of logic.

To establish the distinction between the two epistemic types needed for knowing computational correctness, I begin with a refutation of procedures claiming to show that everything follows from contradiction. We can indeed reach any *q* from any (*p* -*p*) using rules of inference, such as disjunctive syllogism and material implication, whose validity seem necessarily true. But we can also know that such principles no longer have logical necessity if negating *p* no longer does the logical work of taking away *p*. Thus, knowledge that a formula results from a correct application of computational rules is one thing; knowledge that rules and what follows from them express necessary truths is another. I will call the first kind of knowledge computational knowledge. Since the science of logic has traditionally (and accurately) been thought to know necessary truths, I will call the second kind of knowledge logical knowledge.

I intend to show that computational knowledge is not only different from logical knowledge but relies on logical knowledge, logical knowledge that, therefore, cannot be achieved by the use of computational methods. It does not rely on logical knowledge as knowledge of a conclusion relies on knowledge of the premises. It relies on logical knowledge that way any science relies on recognizing that its inferences are valid. To show that, I next argue that recognizing computationally that a rule is correctly applied requires recognizing the validity of an inference, and therefore the recognition that a statement about inferential validity is necessarily true.

The fact that the grasp of inferential validity presupposed by the knowledge of computational correctness is the grasp of a necessary truth that an inference form is valid does not imply that, at the computational level, either proof-theoretic or modal definitions of logical consequence are more useful, more powerful or more in accord than other definitions

with the "correct" concept of logical consequence at the epistemic level, the level of the grasp of the validity of an inference form that is presupposed by the grasp that a procedure conforms to computational rules. Such questions mix apples and oranges.[2] Computational methods cannot provide entirely accurate models of how we precomputationally achieve knowledge of necessary truths of logic, since, by hypothesis, that knowledge not only is not but cannot be computational. The rock bottom metaphysical explanation of what causes truths to be necessary and/or epistemic explanation of what causes our knowledge that a truth is necessary might resemble semantic computational approaches to logical consequence more, proof-theoretic ones more, or not resemble one more than the other. And the fact that the grasp of inferential validity is the grasp of a necessary truth of logic does not imply that what valid inferences preserve is modal truth. What is grasped is that the denial of a statement about the preservation of truth, whatever the modality of that truth, is contradictory.

Traditional logic and classical modern logic both presuppose a grasp of necessary logical truths that is implicit and seek to make our understanding of inferential validity explicit, but modern logic does this by means of an incomparably more powerful tool, computational methods. (I will explain the contrast between implicit and explicit knowledge sufficiently for the purposes of this discussion.) Since computational knowledge and logical knowledge are distinct epistemic types, however, the relation of computational knowledge to logical knowledge is like the relation of mathematics to physics, which are also distinct epistemic types: computational knowledge is not the same as logical knowledge but is as much an indispensable tool for advancing our logical knowledge as mathematics is for physics.

To show that knowledge of inferential validity requires recognizing that some statements of logic are necessarily true, I will use an overlooked implication of Lewis Carroll's Achilles-Tortoise paradox. And a refutation of Quine's critique of the "analytic" will show that grasping the necessary does not require occult mental acts but some of the same

acts, if any, required for grasping contingent empirical truths; whatever it may take to grasp contingent truths, it takes no more to grasp necessary truths. In fact, knowledge of logical necessity bespeaks limited mental powers, not extraordinary ones. (Finally, I will discuss bivalence and the Law of Excluded Middle. ??)

Since we can be aware of logical necessity, the success of nonstandard logics at the computational level (the fact that we can construct computationally verifiable definitions of logical consequence, truth conditions and/or truth values for them) does not justify pragmatism in the epistemology of logic.[3] It has always been recognized that we must use logic to do logic. I am just adding that the logic we must use to do any kind of logic, including many-valued, paraconsistent, and dialethic logics, is logic that is standard in the sense that it does not permit contradiction and is bivalent. Unless we can know that principles of standard logic are necessarily true, we could not even know the truth of statements like "This nonstandard system succeeds computationally" or "This step is correct according to these non-standard rules."

> To put this epistemic inquiry in context, consider the following statements:
>
> A distinction between propositions (or statements, or sentential contexts) which are de dicto and propositions (and so on) which are de re originates in medieval philosophy. But only contemporary modal logic affords the tools for a precise characterization of this distinction, although it must be granted that the distinction remains a puzzle in *epistemic* contexts. (Emphasis in original.)[4]

And as Hilary Putnam said about Tarski's treatment of truth:

> The critics (of William James account of truth) also claim the problem of giving us a satisfactory account of truth was solved in this century by the work of the great logician Alfred Tarski. I myself believe that Tarski's great technical contribution notwithstanding, his work does *nothing* to explicate the notion of truth. (Emphasis in original.)[5]

Elsewhere, Putnam has even demonstrated that Tarski's technical achievement tells us

nothing about the philosophical problems concerning truth.[6] Similarly, I am arguing that the technical success of nonstandard logics tells philosophers nothing about the epistemic status of consistency, bivalence, or necessary truth.

This thesis should be considered very minor and unimportant by whose chief interest is doing modern logic for its own sake, or for the sake of its power and fecundity; for there just is no way other to get as far in logic as we do by using computational methods. But it is of more than minor importance for those of us interested in the epistemic pre-conditions for knowing what we are doing when we are doing modern logic. It is also more than a minor point for those who want to draw philosophical conclusions from what modern logic can do. Now that we have nonstandard logics, the impression can be that they eliminate the epistemic need for necessary truth, consistency and bivalence. Just as Tarski's great technical achievement tells philosophers nothing about the notion of truth, the technical success of nonstandard logics tells philosophers nothing about the epistemic status of consistency, bivalence, or necessary truth.

The problem, of course, is that those whose teaching concerns epistemic issues about logic in one class are very often the same people who were teaching logic in the pervious class. It is very difficult to shift from the point of view where it is hardly worth noting that you are depending on implicit knowledge of necessary logical truth, since you couldn't even begin to do what you are doing otherwise, to a point of view where such issues create problems to be solved.

1.

The technical success of non-standard logics can cast doubt on whether at least some of logic's formulas express necessary truths (or are epistemically connected with necessary truths in some other important way). For example, some philosophers and logicians are bothered by the apparent soundness of procedures, in traditional and classical modern logic, showing that everything follows from contradiction, ex contradictione quodlibet (ECQ). Logicians and philosophers of logic interested in the "entailment" of a specific conclusion by

a specific finite set of premises want to avoid the problem that any contradiction entails any statement whatsoever. Paraconsistent and relevance logics prevent ECQ from holding by treating other inference principles, disjunctive syllogism in particular, traditionally considered necessarily true as if they were not.

But from an epistemic point of view, we don't have to worry about ECQ. We don't need non-standard logics to avoid the problems that ECQ seems to get us into. For we are capable of recognizing that attempts to justify ECQ do not and cannot succeed as we are capable of recognizing, for the sake of determining inferential validity, that a computational proof does succeed. Methods of showing that '$(p \barwedge p) \supset q$' is valid do illustrate that we can recognize that each step in the process is justified by a rule or a rule together with preceding steps that satisfy rules. From that viewpoint the answer to the question, 'Is everything true, if contradiction is,' can be yes. But from the viewpoint of our ability to recognize a rule's association with logically necessary truth, the answer must be no.[7] The reason is that permitting contradiction prevents there being any necessary truth for the rules the procedure uses to be associated with. (Do not confuse ECQ with the fact that a truth-functionally inconsistent antecedent implies everything. Inconsistent antecedents make their conditionals valid because the antecedents resolve to *falsehood*; ECQ is the claim that if both parts of a contradiction are simultaneously *true*, and so the compound sentence '$p \sim p$' is true, everything is true.)

A traditional proof that '$(p \barwedge p) \supset q$', where '$q$' can be anything, is valid is

(1) $p \barwedge p$	Premise
(2) p	1, Law of simplification
(3) $p \vee q$	2, Law of addition
(4) $\neg p$	1, Law of simplification
(5) q	3, 4, Law of disjunctive syllogism
(6) $(p \barwedge p) \supset q$	1, 5, Conditional proof

But if we allow contradiction, the law of disjunctive syllogism (step 5) cannot do the

logical work that the argument relies on. We can call disjunctive syllogism a 'law of subtraction.' By denying that '*p*' is true, we subtract it from '*p* or *q*', and '*q*' is the remainder. But how do we deny the truth of '*p*'? By the truth of '¯*p*'? Normally, that is how we accomplish denying '*p*'; for that is the job we normally give 'not' and other negation signs, the job of denying, subtracting, canceling, deleting, taking away.[8] But the truth of '¯*p*' does not mean that '*p*' is false, if '*p* ~*p*' is also true. If '¯*p*' does not make '*p*' false, '*p* or *q*' does not license us to conclude '*q*' from '¯*p*'. (Or more technically, disjunctive syllogism is no longer 'truth-functionally valid' — or true on every bivalent evaluation, or true under every possible assignment of T or F to atomic propositions, etc., whatever you consider the most appropriate technical concept; for it is not true in the case where '*p*¯*p*' is true. (In addition to technical terms, I will not only continue to also use nontechnical terms, but will use them in preference where possible; for the issue is the epistemic status of our technical methods, especially the epistemic conditions *presupposed* by the technical clarity and rigor of computational methods. Since I am explaining the epistemic conditions required for achieving the clarity and rigor associated with computational technical vocabulary, the explanation cannot presuppose that clarity and rigor except as the fact given to be explained, not as the means of explanation. The epistemic conditions that give us the ability to design and use computational methods are causally *prior to* the existence of their technical vocabularies, and so to clarity and rigor with which we endow those vocabularies. Still, any such explanation may have to introduce technical terms of another kind, philosophical terms, endowed with whatever clarity and rigor is appropriate for the purpose at hand. And as with technical terms of any kind, their introduction must rely, ultimately, on the use of nontechnical vocabulary.)

Epistemically, disjunctive syllogism's (or *modus tollendo ponens*'s) function depends on excluding contradiction. The dependence is not that of a conclusion on a premise; rather, disjunctive syllogism's function depends on a use that negation signs sometimes — and in fact, normally — have. The logical work disjunctive syllogism does depends on the value that

we happen to, but need not, use "not" and "-" for, the relation *other-than*.[9] The existence of multiple kinds of negation is not an issue here. The contradictions about which ECQ is supposedly demonstrated are the simultaneous affirmation and denial (or positing and taking away, or whatever) of the same thing by means of a negation sign used as we happen to ordinarily use it in two-valued contexts; so that is the kind of "negation" in play when we are talking about ECQ. Traditional principles of noncontradiction – whether for sentences or predicates, schema or schematic letters, in an object language or a metalanguage – just express the job we normally give to negation signs: Sentence or schema '^{-}p' is a denial of '*p*', and vice versa; predicate 'F' cancels 'non-F', and vice versa.

But in any proof of ECQ, the sign by which we indicate that one premise is the contradictory of another must have recognizably the same use in that premise as in any rule (or axiom or whatever, depending on how the computational method is structured), such as disjunctive syllogism or material implication, that allegedly makes any '*q*' a logical consequence of a contradiction. If not, we will recognizably commit a fallacy of equivocation. So however the sign indicating contradiction operates in the rule to justify our concluding to '*q*', the sign must operate in the same way in one of the contradictory premises. If the rule does not use that sign or an equivalent sign, or if it uses the same sign but in a different way, the contradictory premise and the rule are irrelevant to each other. This is another reason why the existence of multiple kinds of negation is not an issue, as long as the use negation signs have in contradictions is the same as whatever use they have in disjunctive syllogism. Contradictory premises are irrelevant to deriving '*q*' from '^{-}p' via disjunctive syllogism if the negation signs do not have the same use. All we would have accomplished would be to change the subject, as Quine said about attempts to deny principles of non-contradiction.[10] And the subject(s) originally under discussion are contradiction, and so negation, as they are understood in the context of standard bivalent logic, whether traditional or classical modern).

That disjunctive syllogism's work 'depends on' the use we normally give negation signs does not mean that it depends on contingent lexicographical facts such as that we

happen to use noises like 'not' and shapes like '-' as signs for negation. Disjunctive syllogism depends on that which we happen to, but need not, use 'not' and '-' for. But disjunctive syllogism does not depend on the relation between 'not' or '-' and standard negation being what it happens to be; disjunctive syllogism depends on standard negation being what it happens to be.

The distinction between the lexicographical sense of 'use' and its sense in 'disjunctive syllogism depends on the use that we normally give negation signs,' does not require some special mental state, different from whatever mental states (if any) may explain our awareness of how shapes, noises or actions are used as signs, to account for our awareness of what negation is. Knowledge of how we use negation signs includes a knowledge of what negation is. And I will argue that any mental states required to explain our awareness that contradictions cannot be true must be of the same kind as whatever mental states may be required to explain our empirical awareness that, for example, the letter-token, *T*, is *not* the letter-token, *T*. Also, someone could confuse the way we use 'is not' with the way we use 'is.' If they were to say 'A is not A,' we would not have to conclude that they were employing a nonstandard logic. We could have *behavioral* evidence that the way they were using 'is not' is the way the rest of us use 'is.' That behavioral evidence would tell us that they were lexicographically mistaken about the uses of 'is' and 'is not,' not logically mistaken about the work that standard negation does.[11]

No inference that requires negating something in the standard way is valid if the truth of '^{-}p' does not succeed in making 'p' false; so no valid inference can show that if a contradiction is true, everything is. If an inference relies on a principle using a noise or shape as a negation sign, to allow contradiction would be to prevent the principle from doing the work the inference would need. If an inference has no negation signs or if it does not use them in the standard way, '$p{\sim}p$', is not a contradiction in the normal sense; so the inference cannot show that such contradiction in the normal sense makes everything true.

Having looked at an alleged proof of ECQ, let us examine a decision procedure alleged

to show that '$(p \neg p) \supset q$' is valid. '$\neg p \supset (p \supset q)$' is truth-functionally valid under normal negation, since '$\neg p$' excludes the only case where '$p \supset q$' is false, the case where 'p' and '$\neg q$' are both true. So '$p \supset q$' evaluates as true as a logical consequence of the truth of premise '$\neg p$'. And if '$p \supset q$' and 'p' are true, 'q' evaluates to true as a logical consequence. But 'p' is also given as a premise. So the standard truth-table definitions of the operators appear to require ECQ.

But if we accept '$p \neg p$', '$\neg p$' does not exclude the only case where '$p \supset q$' is false, since 'p' is still a premise. So '$p \supset q$' is no longer a logical consequence of '$\neg p$'; and the truth of is not established according to the rules of the decision procedure. If '$p \supset q$' need not be true, the fact that 'q' is a logical consequence of '$p \supset q$' and 'p' does not imply that 'q' is true. Of course, if we permit contradiction, we can change some other rule or rules of the procedure to make 'q' is a logical consequence of '$p \supset q$'. But that is my point; the bivalent truth-functional rules will no longer have whatever relation with necessary truths they have now. (On the relation of truth-functions to necessary truth, see Section ??).

Moreover, any method of showing the bivalent truth-functional validity of '$(p \neg p) \supset q$' under normal negation will make use of other bivalent truth-functional operators. No other operator can do the work that the truth values of formulas using '$\supset$' requires (and vice versa) if we allow contradiction. Should we say that '$p \supset q$' is never true, since the truth of '$\text{-}(p \neg q)$' does not mean that '$p \neg q$' is false and the truth of '$\neg p \vee q$' does not mean that '$\neg(\neg p \vee q)$' is false; or should we say that '$p \supset q$' is always true, since '$\neg q$' does not falsify 'q'? There are no grounds for saying either, or equal grounds for saying both, since a unary operator, the negation sign, is not doing the job that the employment of binary operators requires. Allowing contradiction strips the gears of our logical machinery, truth-functional operators in this case, of teeth.

Also, *modus ponens* (or the law of detachment) is not valid if we permit contradiction; for '$(p \supset q)p \supset q$' is true if and only if '$(\neg p \vee q)p \supset q$' is true. But the latter is not true if the truth of 'p' does not mean that '$\neg p$' is false. Also, ' $p \supset q$' is true only if '$p \neg q$' is not. But '$p \supset q$'

and '$p^{-}q$' can both be true, if '^{-}q' does not falsify 'q'. (And what epistemic goal have we achieved in knowing that '$p \supset q$' is true, if we also know that 'p^{-}q' is true — or is even able to be true — since '$q^{-}q$' is true?)

Traditionally, the science of logic was thought to achieve knowledge of the necessary truth of logical principles. Assuming we can have such knowledge, computational procedures establishing that ECQ is valid illustrate the difference between the two epistemic types that I am arguing are needed for knowing computational correctness. Does ECQ express a valid logical consequence? We can be aware that each step in a process establishing ECQ is justified by a rule ordinarily epistemically associated with, or a definition that ordinarily grounds, a logical truth traditionally recognized to be necessary. In that sense, the answer to our question about ECQ can be yes. But with respect to our ability to know whether those formulas still express valid logical consequences in the sense of necessary truths of logic, if contradiction is permitted, the answer has to be no.[12] The reason is that permitting contradiction prevents any statement from being a logically necessary truth, to the extent that the statement makes use of negation signs. So, if there is logical knowledge in the sense of knowledge of necessary truths of logic, as I will argue there is, knowledge that a computational process for showing validity correctly obeys rules or instantiates definitions is not the same as logical knowledge, since it is not the same as knowledge of any necessary truth of logic.

The fact that we can define the bivalent operators by the matrix method without making negation primitive tells us something important about the epistemic type represented by that computational method itself, but not anything (directly) pertinent to the epistemic type I am calling knowledge of necessary truths of logic. For if we cannot know that what is expressed by 'This attempt to define operators succeeds' excludes its contradictory opposite, or that there is no third choice between excluding or not excluding its contradictory opposite, we achieve no epistemic goal when we think we learn that what 'This attempt to define operators succeeds' expresses, or any other noises express, is true.

Could we have purely *pragmatic* reasons for taking '$p \supset q$', for example, as true under contradiction in some assignments of truth and falsity to '*p*' and '*q*' and false in others? Not if the process of arriving at such a pragmatic decision relies on even one inference thought to be deductively valid; for in making an inference, we rely on knowledge that some argument form has logical validity. And the latter knowledge is not purely pragmatic since, as I argue in the following sections, it is knowledge of the necessary truth of a principle in a logic that does not allow contradiction; for if we allow contradiction, nothing is logically necessary.

A logic with only positive principles, where no consequence depends on negation signs, would not avoid the *epistemic* problem. Again, without a negation sign we cannot claim that a contradiction implies everything. Also, if contradiction is allowed, when we are proceeding as if positive formulas like '$p \supset (p \vee q)$' or '$(pq) \supset p$' were true, we would do so knowing that they might also not be true. We could decide to use a 'true' positive principle until such time that we found that the principle was also not true. But that decision would not be purely pragmatic, if we were limiting ourselves to positive consequences because of our knowledge that a necessary result of using negation signs while allowing contradiction is that the wheels of our logical machinery would spin but do no work.[13] Nor would it be purely pragmatic if the process of reaching that decision relied on even one inference's being thought to be valid.

As paraconsistent and dialethic logicians are correct to remind us, we have theories, databases, documents and other linguistic structures that contain contradictions and yet do not imply that all sentences are true. The reason why is not just that ECG is false, but that NOTHING follows from contradiction. So in order to prevent the explosion, we do not need a nonstandard logic. To say that is not to say, however, that paraconsistent and dialethic systems cannot be interesting and important for other reasons. (Henceforth, I will not speak of nonstandard "logics" but of nonstandard "systems," to confine "logical" knowledge to knowledge of the necessary. Systems in this sense, of course, can use either a proof-theoretic or semantic approach.)

2.

Having established the difference between the two epistemic types, knowing that a computational proof or decision procedure conforms to rules and knowing that the rules or the formulas arrived at by following them express necessary truth, I will now argue that recognizing that a step in a computational process conforms to (is an instance of) a rule requires both kinds of knowledge. Recognizing the correctness of a computational procedure requires a grasp of what the rules and the formulas justified by the use of the rules, both which may or may not have any connection with necessary truths, are. And recognizing the correctness of even just one step in a computational process, requires an inference whose validity we recognize because we know it to be an instance of a necessary truth concerning logical consequence. Here, I will focus on the fact that knowing that computational processes conform to rules requires a grasp of the validity of an inference and assume and assume that the grasp requires knowledge that a logical truth is necessary. Subsequently, I will show why the grasp of validity requires knowledge of logical necessity.

Since knowledge of computational correctness presupposes (as a kind of epistemic cause other than knowledge of a logical premise ??have I introduced this idea yet??) a grasp of necessary truth concerning logical consequence, that presupposed knowledge must be acquired other than by computational methods. Without a non-computational grasp of the validity of the inference (and hence of necessity) used to recognize that a computational procedure conforms to a rule, we could not know that the procedure conforms to a rule. This applies as much to procedures in nonstandard systems as it does to standard. The grasp of the validity of an inference required to recognize that procedures conform to the rules of nonstandard — paraconsistent, dialethic, multivalued or whatever — systems relies on a grasp that the opposite of a bivalent truth would be contradictory.

The epistemic inevitability of standard logic does not mean merely that we must start with standard logic, can then use it to build a metalinguistic ladder we can climb to reach a computational method that has nonstandard rules, can kick away the ladder of standard logic

after we have constructed the rules of our nonstandard computational method, and can then go merrily on our way relying on nonstandard logic to recognize that computational steps in our nonstandard method correctly conform to our nonstandard rules. Epistemically we can never stop relying on noncomputationally achieved knowledge of truths of standard logic. To recognize that any computational procedure in a nonstandard system conforms to the nonstandard rules, we must continue to make inferences whose validity is always known noncomputationally by knowing necessary truths of standard logic. We cannot kick the ladder of standard logic away. If we ceased using standard logic, we could know that any step in our nonstandard system conforms to the rules.

That noncomputational knowledge of standard logic is epistemically inevitable in this way is a truth that should be of almost no interest to the practicing modern logician; for it is about as close to a trivial truth as one can get without being trivial. But that very closeness-but-no-cigar to triviality makes it important to the philosopher of logic who in addition to recognizing the prodigious success of computational logic, both internally and various of its applications, wants to know the epistemic preconditions for that success.

Even knowledge of the correctness of an operation so fundamental to computational methods as substitution saving truth or validity requires our grasp of a necessary truth about logical consequence, usually, modus ponens. Consider this example from Quine:

> Substitution of {*w*: *Gw* v -*Hw*} for '*F*' and '*Gy*' for '*p*' in
>
> (4) $\forall x(Fx \rightarrow p) \leftrightarrow . \exists x\, Fx \rightarrow p$
>
> yields:
>
> (5) $\forall x(Gx \vee \neg Hx . \rightarrow Gy) \leftrightarrow . \exists x(Gx \vee \neg Hx) \rightarrow Gy$
>
> The utility of substitution . . . is as a means of generating valid schemata from valid schemata. E.g., since (3) [not shown here] and (5) were got by substitution in schemata which were seen . . . to be valid, we conclude that (3) and (5) are valid. (My emphasis)[14]

"We conclude," that is, knowledge that the newly generated formula is valid according to

computationally verifiable rules is a result of an INFERENCE. As Quine had said earlier about truth-functional schemata "From the validity of a schema we may infer, without separate test, the validity of any schema which is formed from it by (uniform) substitution . . . . of schemata for letters. " (My emphasis)[15]

The rule whose correct application we need to be able to recognize might be something like:

> The result of uniform substitution of schemata for letters in a valid formula is a valid formula.

The inference to the validity of a new schema would be an instance of modus ponens and could go like this:

> (A) If *p q* v ~(*p q*) is a formula gotten from a valid formula by uniform substitution of schemata for letters, *p q* v ~(*p q*) is a valid formula.
>
> (B) *p q* v ~(*p q*) is a formula gotten from a valid formula by uniform substitution of schemata for letters.
>
> (C) Therefore, *p q* v ~(*p q*) is a valid formula.

Or it could go like this:

> (1) If '(*p* ⊃ *q*) v -(*p* ⊃ *q*)' is the result of substituting '*p* ⊃ *q*' for each '*p*' in '*p* v⁻*p*', '(*p* ⊃ *q*) v -(*p* ⊃ *q*)' is a valid formula if '*p* v⁻*p*' is.
>
> (2) '(*p* ⊃ *q*) v -(*p* ⊃ *q*)' is the result of substituting '*p* ⊃ *q*' for each instance of '*p*' in '*p* v⁻*p*'.
>
> (3) '(*p* ⊃ *q*) v -(*p* ⊃ *q*)' is a valid formula if '*p* v⁻*p*' is.
>
> (4) If '*p* v⁻*p*' is a valid formula, '(*p* ⊃ *q*) v -(*p* ⊃ *q*)' is a valid formula.
>
> (5) '*p* v⁻*p*' is a valid formula.
>
> (6) Therefore, '(*p* ⊃ *q*) v -(*p* ⊃ *q*)' is a valid formula.

Epistemically, then, the use of formal methods relies on our ability to recognize that a deductive inference is valid. Rules do not apply themselves. Applying them requires an at least minimal implicit deduction, usually one so obvious it is difficult to notice that it is a

deduction. Even when we are doing dialethic logic, knowledge of the legitimacy of the result of a substitution presupposes knowledge of the validity of inferences in standard logic.

N.B. The remainder of this section consists of a hopefully sufficient variety of other examples that those still skeptical will be convinced. If you do not need further convincing that recognizing that a step correctly instantiates a computational rule requires knowledge of the validity of an inference, you can safely skip the rest of this section. Section 3 begins the discussion of why knowledge of inferential validity requires a grasp of logically necessary truth.

Or, consider this set theoretical example adapted from Stephen Pollard[16]:

(6) $(\forall x\ x\in x \rightarrow \forall y\ y=y)$ Assumption

(7) $(\forall x\ x\in x \rightarrow \sim\forall y\ y=y)$ Assumption

(8) $\sim\forall x\ x\in x$ 6, 7 Rule of Negation Introduction

The Rule of Negation Introduction could be:

(D) If $(\varphi \rightarrow \psi)$ and $(\varphi \rightarrow \sim\psi)$ appear on earlier lines, $\sim\varphi$ may be written on a later line.

Recognizing that the formula on line 8 is a computationally correct consequence of the formulas on lines 6 and 7 and the Rule of Negation Introduction, requires an inference that would be an instance of modus ponens such as:

(E) If $(\forall x\ x\in x \rightarrow \forall y\ y=y)$ and $(\forall x\ x\in x \rightarrow \sim\forall y\ y=y)$ appear on earlier lines, $\sim\forall x\ x\in x$ may be written on a later line.

(F) $(\forall x\ x\in x \rightarrow \forall y\ y=y)$ and $(\forall x\ x\in x \rightarrow \sim\forall y\ y=y)$ appear on earlier lines.

(G) $\sim\forall x\ x\in x$ may be written on a later line.

But even to get from (D) to (E) requires inferences that are an instance of modus ponens at a more fundamental level, inferences such as:

(H) If ($\forall x\ x\in x \rightarrow \forall y\ y=y$) is a substitution instance for φ in (D) and ($\forall x\ x\in x \rightarrow \sim\forall y\ y=y$) is a substitution instance for ψ in (D) , then if ($\forall x\ x\in x \rightarrow \forall y\ y=y$) and ($\forall x\ x\in x \rightarrow \sim\forall y\ y=y$) appear on earlier lines, $\sim\forall x\ x\in x$ may be written on a later line.

($\forall x\ x\in x \rightarrow \forall y\ y=y$) is a substitution instance for φ in (D) and ($\forall x\ x\in x \rightarrow \sim\forall y\ y=y$) is a substitution instance for ψ in (D)

If ($\forall x\ x\in x \rightarrow \forall y\ y=y$) and ($\forall x\ x\in x \rightarrow \sim\forall y\ y=y$) appear on earlier lines,

$\sim\forall x\ x\in x$ may be written on a later line.

(I) If ($\forall x\ x\in x \rightarrow \forall y\ y=y$) and ($\forall x\ x\in x \rightarrow \sim\forall y\ y=y$) appear on earlier lines,

$\sim\forall x\ x\in x$ may be written on a later line.

($\forall x\ x\in x \rightarrow \forall y\ y=y$) and ($\forall x\ x\in x \rightarrow \sim\forall y\ y=y$) appear on earlier lines.

$\sim\forall x\ x\in x$ may be written on a later line.

(H) If ($\forall x\ x\in x \rightarrow \forall y\ y=y$) and ($\forall x\ x\in x \rightarrow \sim\forall y\ y=y$) are wffs, AND

"If ($\forall x\ x\in x \rightarrow \forall y\ \ \ y=y$) and ($\forall x\ x\in x \rightarrow \sim\forall y\ y=y$) appear on earlier lines, $\sim\forall x\ x\in x$ may be written on a later line" is the result of a

uniform substitution of ($\forall x\ x\in x \rightarrow \forall y\ y=y$) and ($\forall x\ x\in x \rightarrow \sim\forall y\ y=y$) for φ and ψ, respectively in (D), THEN

if ($\forall x\ x\in x \rightarrow \forall y\ y=y$) and ($\forall x\ x\in x \rightarrow \sim\forall y\ y=y$) appear on earlier lines, $\sim\forall x\ x\in x$ may be written on a later line.

[If A and B are wffs, AND "If A and B appear on earlier lines, *xxx* may be written on a later line" is the result of uniform substitution of A for φ and B for ψ in (D), THEN if A and B appear on earlier lines, *xxx* may be written on a later line.]

(I) ($\forall x\ x\in x \rightarrow \forall y\ y=y$) and ($\forall x\ x\in x \rightarrow \sim\forall y\ y=y$) are wffs, AND

"If ($\forall x\ x\in x \rightarrow \forall y\ y=y$) and ($\forall x\ x\in x \rightarrow \sim\forall y\ y=y$) appear on earlier lines, $\sim\forall x\ x\in x$ may be written on a later line" is the result of a uniform substitution of ($\forall x\ x\in x \rightarrow \forall y\ y=y$) and ($\forall x\ x\in x \rightarrow \sim\forall y\ y=y$) for φ and ψ, respectively in (D).

[A and B are wffs, AND "If A and B appear on earlier lines, *xxx* may be written on a later line" is the result of uniform substitution of A for φ and B for ψ in (D).]

(J) If ($\forall x\ x\in x \rightarrow \forall y\ y=y$) and ($\forall x\ x\in x \rightarrow \sim\forall y\ y=y$) appear on earlier lines, $\sim\forall x\ x\in x$ may be written on a later line.

[If A and B appear on earlier lines, *xxx* may be written on a later line.]

The fact that (B), (F) and (I) are computationally verifiable does not dispense with the need to grasp the validity of an inference to know that a rule concerning computational procedures, such as making a substitution or introducing a formula beginning with a particular sign, here, ~, has been applied correctly. What these examples illustrate is that recognizing that a rule has been applied correctly requires, in addition to a grasp of what the rule and any formulas to which it is applied are, a recognition of the validity of an inference form like modus ponens.

That recognizing computational correctness depends on grasping of the validity of inferences may be even clearer in the case applying semantic definitions of logical constants. The following examples, adapted from Matthew McKeon,[17] illustrate just some of the kinds of inferences that can be required for applying semantic definitions in establishing logical consequence. For the sake of abbreviating a lengthy series of inferences and for perspicuousness, I will use ordinary English where possible, and I will assume that the interpretation of formal sentences uses a non-empty domain, the empty variable assignment and a variable assignment that is an extension of it.

To be shown is that a formula represents a logical truth if [AND ONLY IF ??] it is a model-theoretic consequence of the empty set of sentences.

(A) If any arbitrary structure is a model of $\forall x\mathrm{M}(x) \rightarrow \exists x\mathrm{M}(x)$, $\forall x\mathrm{M}(x) \rightarrow \exists x\mathrm{M}(x)$ is a model theoretic consequence of the empty set of premises.

(B) Any arbitrary structure is a model of $\forall x\mathrm{M}(x) \rightarrow \exists x\mathrm{M}(x)$.

(C) $\forall x\mathrm{M}(x) \rightarrow \exists x\mathrm{M}(x)$ is a model theoretic consequence of the empty set of premises.

(K) If sentence X is a consequence of the empty set of sentences, every structure for X's language is a model of X.

(L) X is a model-theoretic consequence of the empty set of sentences.

(M) Every structure for X's language is a model of X.

To be shown is that any arbitrary structure U is a model of $\forall x$M(x) -> $\exists x$M(x).

(N) Every structure has a non-empty domain D.

(O) U is a structure.

(P) U has a non-empty domain D

Or

(Q) If U is a structure whose domain is D, D is non-empty.

R) U is a structure whose domain is D.

(S) D is non-empty.

And

(T) Every non-empty domain has at least one element d.

(U) D is a non-empty domain.

(V) D has at least one element d.

And

(D) If (U is a model of $\forall x$M(x)) is a result of substituting $\forall x$M(x) for φ in the definition of $\forall$, then if U is a model of $\forall x$M(x), U satisfies M(x).

(E) (U is a model of $\forall x$M(x)) is a result of substituting $\forall x$M(x) for φ in the definition of $\forall$.

(F) If U is a model of $\forall x$M(x), U satisfies M(x).

And

(G) If U is a model of $\forall x$M(x), U satisfies M(x).

(H) U is a model of $\forall x$M(x). [By hypothesis; $\forall x$M(x) is the antecedent of the conditional we are testing.]

(I) U satisfies M(x).

And

(J) If (U satisfies M(x)) is a result of substituting M(x) for φ in the definition of $\exists$, then if U satisfies M(x), U is a model of $\exists x$M(x).

(K) (U satisfies M(x)) is a result of substituting M(x) for φ in the definition of $\exists$.

(L) If U satisfies M(x), U is a model of $\exists x$M(x).

And

() If (if U satisfies M(x), U is a model of $\exists x$M(x)), then if (U is a model of $\forall x$M(x) -> U satisfies M(x)), then U is a model of $\forall x$M(x) -> U is a model of $\exists x$M(x)).

() If U satisfies M(x), U is a model of $\exists x$M(x)).

() If (U is a model of $\forall x$M(x) -> U satisfies M(x)), then U is a model of $\forall x$M(x) -> U is a model of $\exists x$M(x).

() If (U is a model of $\forall x$M(x) -> U satisfies M(x)), then U is a model of $\forall x$M(x) -> U is a model of $\exists x$M(x).

() If U is a model of $\forall x$M(x), U satisfies M(x).

() U is a model of $\forall x$M(x) -> U is a model of $\exists x$M(x).

() If (U is a model of $\forall x$M(x) -> U is a model of $\exists x$M(x)), then either ~(U is a model of $\forall x$M(x)) or U is a model of $\exists x$M(x).

() U is a model of $\forall x$M(x) -> U is a model of $\exists x$M(x).

() Either ~(U is a model of $\forall x$M(x)) or U is a model of $\exists x$M(x).

() If (U is a model of $\forall x$M(x) -> $\exists x$M(x)) is a result of uniform substitution of $\forall x$M(x) for φ and $\exists x$M(x) for ψ in the definition of -> then if either ~(U is a model of $\forall x$M(x)) or U is a model of $\exists x$M(x) then U is a model of $\forall x$M(x) -> $\exists x$M(x)).

() U is a model of $\forall x$M(x) -> $\exists x$M(x) is a result of uniform substitution of $\forall x$M(x) for φ and $\exists x$M(x) for ψ in the definition of ->.

() If either ~(U is a model of $\forall x$M(x)) or U is a model of $\exists x$M(x) then U is a model of $\forall x$M(x) -> $\exists x$M(x).

() If either ~(U is a model of $\forall x$M(x)) or U is a model of $\exists x$M(x) then U is a model of

$\forall x$M(x) -> $\exists x$M(x).

() Either ~(U is a model of $\forall x$M(x)) or U is a model of $\exists x$M(x).

() U is a model of $\forall x$M(x) -> $\exists x$M(x). [Q.E.D.]

if (U satisfies M(x)) is a result of substituting M(x) for φ in the definition of ->, then if (U satisfies M(x)) holds, U satisfies $\forall x$M(x) -> $\exists x$M(x)

(W) If U satisfies $\forall x$M(x) then for any element d of D U satisfies $\forall d$M(d).

(X) U satisfies $\forall x$M(x) [by hypothesis].

(Y) For any element d of D, U satisfies $\forall d$M(d).

(X) If for any element d, U satisfies $\forall d$M(d), U satisfies $\exists d$M(d).

(Y) For any element d, U satisfies $\forall d$M(d).

(Z) U satisfies $\exists d$M(d).

As the use of dictum de omni in examples ?? and ?? shows modus ponens is not the only necessary logical truth the grasp of which may be required for recognizing computational correctness.

Again, recognition of computational correctness requires knowledge of the computational rules that establish what is correct in the method. Such knowledge need not be of anything necessarily true or epistemically connected with necessary truth. The rules in a card game like bridge are not necessarily true, but same kind of inference required to know that a computational procedure is a correct application of a rule is required to know that a play in bridge is a correct application of a rule:

If I have a card in the suit that was led, my not playing a card in that suit will cause us to lose the hand.

I have a card in the suit that was led.

Therefore, my not playing a card in that suit will cause us to lose the hand.

As this example shows, the point I am making is not arcane; the point is so obvious it is difficult not to overlook it. The rules of bridge necessarily imply that we lose the hand if I do not play a card in a particular suit, but the necessity comes from the implicitly known validity of an inference form, not from the rules themselves.

3.

The grasp of necessary logical truth that recognizing the validity of an inference requires must be implicit. By "implicit" I mean that we grasp the truth of, for example, modus ponens without formulating it in statements distinct from the concrete instance in which we use it. We do not grasp the truth of a statement like "If statement 1 implies statement 2, and statement 1 is true, then statement 2 is true," apart from and as opposed to grasping the validity of an inference like "If the road is wet, it's slippery; and the road is wet; therefore, it is slippery." A knowledge of modus ponens as expressed in a sentence distinct from the sentences making up such an inference would have to be irrelevant to our grasp of the inference's validity; for if modus ponens needed to be expressed separately for us to recognize the validity of such an inference, an infinite regress would result. Quine, using Lewis Carroll's 'Achilles-Tortoise Paradox' showed this in "Truth by Convention";[18] I will show a further consequence of that paradox in Section ??. (Quine would probably want to add that we cannot avoid the regress by distinguishing between the premises of an argument and its inference rules, since we can no more make a hard and fast distinction between rules and premises than we can, according to him, between analytic and synthetic statements. I will show how to avoid any problem pertinent to logical knowledge with either kind of distinction.)

I have called knowledge that a formula results from a correct application of a computational rule computational knowledge, as opposed to logical knowledge of a necessary truth that a rule may be epistemically connected to. In fact, computational

knowledge presupposes knowledge of two kinds of contingent truths: (1) that a rule is what it is and (2) that a state of affairs that the rule concerns occurs [for example, that a formula of a certain form appears on a line of a proof — as in (F) — or that a formula [or schema] is a substitution instance of a schema or schematic letter that appears in the rule to represent the formulas which the rules concern — as in (B) and (I) — or that I have a card in the suit that was led — as in the bridge example]. If we are using a conditional to express the rule, (2) would be the contingent knowledge that the antecedent of the conditional is satisfied.

Since rules do not apply themselves, the other kind of knowledge presupposed by the recognition of computational correctness is knowledge of the validity of an inference applying a rule to a case. Knowledge of inferential validity includes implicit knowledge of a necessary truth of logic stating the validity of the inference made from rules and facts to which the inference applies the rules, for example, the implicit recognition of what we can subsequently explicitly express by "if '*p*' implies '*q*' is true and '*p*' is true, '*q*' must be true."

This kind of required knowledge belongs to what is traditionally called logical knowledge (as opposed to the knowledge of what may be contingent truths that I am calling computational knowledge).

??

The goal of traditional (and classical modern) logic was justified explicit knowledge of the necessary truth of inferences principles that are only implicitly known to begin with. Why does the recognition that an inference is valid require the implicit grasp of a necessary truth? That logical knowledge is of the necessary does not mean that the truth preserved by valid inferences must be modal. (Of course, since the opposites of all truths are either contradictory or non-contradictory, all truths are either necessary or contingent as defined by the contradictoriness of the opposite, and so all truths are modally characterized in one way or another whether or not they use modal terms and whether or not we know their

modality, implicitly or explicitly, when we know their truth.) That the implicit grasp of inferential validity is the grasp of a necessary truth merely means that at the epistemic level the difference between knowing that three propositions like (i) "If it is raining, the road is slippery," (ii) "It is raining," and (iii) "The road is slippery" happen to be simultaneously true, on the one hand, and knowing that (iii) is a logical consequence of (i) and (ii), on the other, is that even when we don't know that (i) and (ii) are true, we know that on the hypothesis that (i) and (ii) are true, (iii) is true. But the latter knowledge, logical knowledge, is knowledge of the necessary. That it is knowledge of the necessary follows from the fact that it is caused simply by knowing that for which "if" and "then" happen to be used, namely, a certain kind of relation between the truths of propositions, a relation I will call "C". And the reason it is caused simply by our understanding what C is is that this relation's holding between the truths of (i), (ii) and (iii) is what we are hypothesizing when we hypothesize the truth of "If it is raining, the road is slippery", as opposed to hypothesizing the truth of "It is raining" and "The road is slippery."

Since our knowledge of the validity of modus ponens is caused only by a recognition (not in any way a perfect recognition, but a recognition sufficient for us to make it that for which "if . . . , then" constructions are used) of what C is, the evidence for the validity of modus ponens consists only of what C is. Since the evidence consists only of what C is, it consists of knowing that, if modus ponens is not valid, what C is is not what C is; C is not C. In other words, C, that for which we happen to be using 'if . . . , then', both is C and is not C, since what C is causes modus ponens to be valid, but modus ponens is assumed not to be valid. That is why knowledge of the validity of modus ponens is knowledge of the necessary: it is knowledge that its denial is contradictory. If the evidence for modus ponens consisted of more than the fact that, if modus ponens is not valid, C is not C, then the evidence does not consist solely of the fact that C is what it is. And if the evidence does not consist solely of C's being what it is, that relation C holds between the truths of (i) and (ii) and the truth of (iii) is not what we are hypothesizing, as opposed to hypothesizing the actual truth of (ii)

and (iii), when we hypothesize the truth of (i).

"If modus ponens is not valid, C is not C" does not mean that something that was previously C would have changed and become something that is now not C, while retaining the rest of the identity it has other than that of being C. On that assumption, nothing would be simultaneously C and not C. Rather, the assumptions are that what we call "modus ponens" includes what C is and that modus ponens remains what it is throughout the discussion. Therefore, if what C is is the evidence for the validity of modus ponens, modus ponens, because it includes what C is, can fail to be valid only if what C is is simultaneously not what C is; if it can fail to be valid for any other reason, our knowledge of what C is is not the sole cause of our knowledge that modus ponens is valid. So if we know that modus ponens is valid, we know it only by knowing that the opposite is contradictory, and if we do not have implicit knowledge that the inference form made explicit by modus ponens is valid when we are using that form to make an inference, we do not know that inference is valid. So we can recognize that an inference is valid if and only if we can recognize, implicitly at first and then explicitly, that a statement that it is valid is true on penalty of contradiction, necessarily true.

What C is causes "Modus ponens is valid" to be true in such a way that (1) knowing what C is and (2) knowing that what C is is that for which we use "if . . . , then" is sufficient for us to know that modus ponens is valid. And so in recognizing the truth of "Modus ponens is valid," we are recognizing that if it is not true, the evidence for its truth both is and is not what it is: what C is, which happens to be the evidence for the truth of "Modus ponens is valid," is not what C is at the same time that it is what C is.

Notice that, as in the case of negation and disjunctive syllogism, the necessity of modus ponens does not depend on the relation between "if . . ., then" and C being what it happens to be; it depends on C being what it happens to be. It does not depend on the lexicographical relation between "if . . ., then" and C being what it is; it depends on the

logical relation hypothesized between antecedent and consequent, the relation C, being what it is. Quine might find a "fallacy of subtraction" in making the distinction between knowledge of what C is and knowledge of what the relation between "if . . ., then" and C is. Finding this fallacy might be occasioned by the worry, typical of Quine, that the distinction would could require us to open a Pandora's box of all sorts of illegitimate mental states and acts. In particular, it might lead us back into a psychology of thought functioning independently of language. We need not worry. When we hear and understand 'Your mother has died', our shock and grief is not caused by our relation to what the noise 'died' is; it is caused by the fact that we have a relation to what death is. But we need not conclude that the relation to what death is is causally independent of our ability to, and our acts of, using language.

The pre-computational grasp of that for which "if . . . , then" is used might include more than C, the relation between the truth values of propositions that makes modus ponens necessarily true. I have called that for which we use "if . . . , then" C to avoid using words like "implication" in this immediate context, because that for which we use such words might include other relations, like relevance. If so, I here am only interested in one aspect of what we hypothesize to be the case when we use "if. . . , then . . .": the fact that it is contradictory to simultaneously hypothesize it and that the antecedent is true but the consequent not true.

rules are not chosen just so that we can know by computation that the result follows necessarily from the rules. In any successful use of a computational method for any purpose, whether or not the purpose is verifying inferential validity, we know that a result follows necessarily from the rules because we know that if a procedure is not correct according to the rules then either the rules or the procedure are not what they are. That is how we construct computational methods; otherwise, they could not give us knowledge of computational correctness. Still, a rule and a result need have no connection with necessary

truth other than being a premise and a conclusion, respectively, of a valid inference.

For example, we know that conclusions follow necessarily from the rules of bridge; so we know that if a particular conclusion does not follow, some rule is and is not what it is. But all the epistemic necessity comes from the association with necessary truths about valid inferences that the rules of bridge acquire by being expressed in sentences that use words like "if," "then," "all," "some," "or," etc. The conceptual objects for which we use the vocabulary that is more specific to the topic of card games do not make the opposite of any of bridge's rules contradictory; those conceptual objects, like suits, denominations, trump, card led, etc., are what they are whether or not the rules of bridge are what they are. On the contrary, C, that for which we use "if . . . , then," is not what it is if modus ponens does not always hold.

We so choose computationally applyable rules that we know that if a correct computational result does not express a necessary truth, some of the rules are not what they are, because of their connection with necessary truth.

Definitions stipulated for the sake of acquiring knowledge of computational correctness will make necessary some truths that are not included in the necessary truths we must know pre-computationally in order to recognize computational correctness. If -A -> (A -> B) is not true, the truth conditions for which "-" and "->" are used are and what they are and are not what they are. But "-A -> (A -> B)" is not rendered true by that for which we use "if . . . , then" precomputationally.

[Is the validity of such anomalies "logical," that is, does it result solely from the way we are using "logical" vocabulary, if it depends on definitions, like that of ->, that have no direct parallel in the precomputational vocabulary we call "logical"? Yes, though -> as defined truth-conditionally is not in our precomputational vocabulary, direct parallels to "&"

and "v" and "-" are in the precomputational vocabulary we call "logical," and using "&," "v" and "-", we can give -> its truth-functional definition.]

4.

Without attempting a complete definition of the science of logic, we can say that both traditional logic (TL) and classical modern logic (CML) aimed to achieve explicit knowledge of the kind of necessary truths of which we have implicit knowledge whenever we knowingly make a valid inference. But if logic is the science that seeks explicit knowledge of the necessary truth of principles of valid inference, recognition that a computational step conforms to rules of computation is not enough to give us logical knowledge. Merely recognizing, for example, that a formula expressing modus ponens results from using rules correctly does not tell that the formula expresses a necessary truth. We can design rules, like the rules of bridge, by the use of which we can arrive at a result that need have no connection with necessary truth. Since computationally verifiable rules need not express necessary truths, recognition of computational correctness is useful for achieving that common goal of TL and CML only if we can also non-computationally recognize a connection between some rule or rules specifying the correctness of computations and some necessary truth or truths, such that we can also non-computationally recognize that if a formula resulting from using the rules correctly does not express necessary truth, the rules do not have their known connection with necessary truth.

In order to have explicit knowledge that a computational result expresses a necessary truth, we must begin from explicit knowledge of the connection with necessary truth of some computational rule or set of rules. When we have a computational process, in which either a formula like one reflecting the necessary truth of modus ponens is given as a rule, or a formula expressing modus ponens is verified by following the rules, we can have two distinct kinds of knowledge of the necessary truth of modus ponens: (1) the implicit knowledge of the validity modus ponens that a grasp of the correctness of a step in the process would require

and (2) the explicit knowledge of the validity of the formula expressing modus ponens that is either a rule of computation or results from using the rules.

Only when (2) is knowledge of a resulting formula are computational methods responsible for giving us the kind of knowledge that is the goal in both TL and CML, explicit knowledge of what was known only implicitly to begin with: necessary logical truth about the validity of inferences. The presupposed explicit knowledge of the connection between rules and necessary truth is not the knowledge I am calling computational, that is, knowledge that something is the result of a correct application of the rules. For in the final analysis the source of our explicit knowledge of the connection between a rule and necessary truth cannot be the merely computational grasp that a process correctly instantiates rules. The ability of a computational process to result in knowledge of the necessity of a truth presupposes that we already have an explicit grasp of the connection between a rule or rules of the computational method and necessary truth. Computational methods can give us knowledge of necessary truth only in dependence on some other way of knowing necessary truth. So, computational methods are useful in logic only because there is such a thing as explicit noncomputational knowledge of necessary truth. Without explicit noncomputational knowledge of necessary truth we could not recognize that the result of a computationally correct procedure expresses a necessary truth, just as without implicit noncomputational knowledge of necessary logical truth we could not recognize the a step is a correct application of computational rules.

Of course, we can express the rules to be followed in computations by means of directions that are neither true nor false, like 'Delete a true formula as a component of a conjunction.' Being neither true nor false, such rules telling us steps to take in computational procedures do not express something logically necessary. But we construct our methods by choosing rules on the basis of our awareness of the necessity of truths like 'Deleting a truth as a component of a conjunction preserves truth value', whose necessity derives from at least one of the ways we use "and,"[19]

The necessary truths with which the presuppositions of a computational method are

connected need not all be truths about forms of inference. A matrix definition of an operator is a useful tool for logical verification only because we know that the matrix's assignment of signs for a finite number of values, like "T" and "F" or "0" and "1", to signs for a finite number of formulas, like "p" and "q", EXHAUSTS ALL THE POSSIBILITIES, that is, because we know that additional assignments different from, rather than just repeating, those shown in the matrix are im-possible. If the rules permit only two values, say, "0" and "1," and permit only one value to be assigned to any formula at one time, we can know that there can be four (2^2) and only four different combinations of values for any two formulas: *p*/0, *q*/0; *p*/1, *q*/0; *p*/1, q/1; p/0, q/1. And so we can also know that corresponding to each of the latter combinations there can be 16 (4^2) and only 16 different combinations of four 0/1 values in the matrix: (1, 0, 0, 0; 1, 1, 0, 0; 1, 1, 1, 0; etc.).

If we did not have that kind of knowledge, matrices would be useless for logical verification. For if we did not know that the sixteen possible combinations of four 0/1 assignments exhaust all the possible combinations of different 0/1 assignments to two formulas, we could not use matrices to learn, by a finite number of steps, that a formula using operators defined by this method was assigned 1 by all possible combinations or 0 by all possible combinations. And we would not consider such formulas computationally verified if we could not recognize that they are true on all possible assignment of truth values to their atomic formulas. Recognizing that amounts to knowing that if a matrix did not assign all possible values to the component sentences of a sentence using an operator, either the matrix or the truth-conditions defining the operator both are and are not what they are. (Again, if these remarks are true, they SHOULD be all but trivial for those whose interest is doing logic by computational methods. But that all-but-triviality is precisely why they those whose interest is the epistemic presuppositions of doing logic by computational methods should find them important.)

In these cases, we need to know necessary mathematical truths in order to verify necessary truths of logic by computational methods. Using mathematical induction in

computational proofs also allows us to grasp in a finite number of steps that all the possibilities for integers are covered; for knowing the definitions of 'n', '+', and '1' can cause knowledge of the necessary truth that n + 1 can be any of the infinite possible integers greater than 1. Like modern logic, most of mathematics uses computational methods. So our recognition that all the mathematical possibilities are covered can never eliminate the need for the implicit recognition of necessary logical truths that is presupposed to using computational methods in mathematics. To grasp the correctness of any step in a computational procedure for establishing a mathematical truth, for example, that there can only be 2^2 combinations of assignments of two values to two formulas, we must have an implicit awareness of the necessary truth of whatever logical truths about valid inference the procedure would instantiate, just as we must for computational procedures in logic.

The genius of CML has been to formulate rules that allow us to know, as a result of recognizing the contingent truths that procedures are in fact instantiations of the rules, that if a formula arrived at does not express a necessary truth concerning inferential validity, some already explicitly known necessary truth(s) would be false. For the purposes of logical verification we select computational rules that we recognize to be so related to necessary logical truths that unless formulas arrived at by their means express necessary logical truths, the rules would both be and not be what we know them to be.

[SHOULD THE "An objection might be" PARAGRAPH COME NEXT??]

For example, we can semantically define some non-individual constants, such as '&' and 'v', so that we know the use of each is identical with (at least) a one of the ways we precomputationally use certain words, such as "and" and "or," respectively. Given what we know about that use of '&'/'and' and/or 'v'/'or', we can then recognize that while "A & B" is true, 'A' cannot not be true, on penalty of contradicting what we assume in assuming that "A & B" is true. And since the other terms in those definitions are uninterpreted symbols, we can know that, under penalty of contradiction, certain wffs using '&' and 'v' according to the

definitions are true on any interpretation of the other symbols in these wffs; for we can know that these wffs must be true solely because that for which we are using the symbols '&' and 'v' are what they respectively are.

An objection might be that while it is appropriate to speak of connections between a rule like modus ponens, or an introduction or elimination rule in a natural deduction system, and a logically necessary truth, it is not appropriate to speak in the same sense of connections between definitions and necessary truths. Definitions, not being propositions, are neither true nor false and so neither necessarily true or false or contingently true or false.

But we can so choose semantic definitions that we can know, for reasons to be discussed below, that a computational method verifiyies "$\forall x$M(x) -> $\exists x$M(x)", for example, only if that formula expresses a necessary truth; for if it does not express a truth, the following satisfaction conditions used in defining the non-individual constants "$\exists$", "$\forall$", and "->", respectively, are not what they, at the same time, are (a non-empty domain, the empty variable assignment and an extension of it are assumed):

A structure satisfies $\exists x$M iff at least one element of the domain satisfies $\exists x$M.

A structure satisfies $\forall x$M iff all the elements of the domain satisfy $\forall x$M

An structure satisfies A -> B iff either it does not satisfy A or does satisfy B.

Such definitions of logical operators by satisfaction conditions do not express necessary truths but contingent stipulations for the use of signs, though stipulations that are intended to remain constant throughout the use of a particular computational method. But once we have stipulated those definitions, relations between the satisfaction conditions of two or more formulas using those constants either hold necessarily or do not hold necessarily. And for standard systems, we so choose the conditions defining non-individual constants that we can recognize computationally that at least some formulas using them

express truths made necessary by relations between those truth conditions. That is, we so define non-individual constants that if certain formulas employing them are not true, the conditions defining those constants both are and are not what they are. They are what they are, since in stipulating them we assume, at least implicitly, that the stipulations do not change, and are not what they are, since we are assuming for the sake of argument that formulas they cause to be true are not true.

In nonstandard systems, at least some of the rules or definitions are not chosen because of their connection to precomputationally known necessary truths. But all rules are chosen so that we can recognize that if a computational result is not what it is, some rule is not what it is. Stipulated definitions are not necessary truths but can be chosen so that we know, by means of an inference relying on implicit awareness of some necessary truth, that if a result is not what it is, the defining condition (not the relation of the definition to that condition, but the condition) is not what it is at the same time that it is what it is. In this way, we construct rules and definitions such that some wffs cannot fail to preserve some designated value in multivalued systems, or to preserve truth on some intensional possible world semantics.

Of course, computational methods also require rules which, like syntactical formation rules, may have no more connection with necessary truth than do the rules of bridge. But applying even syntactical rules requires an implicit grasp of necessary inferential validity, just as applying the rules of bridge do. Even an operation as simple and basic as substitution provides a sufficient example, as we saw in the Introduction ??, and to use definitions like the semantic definitions above we need to know that letters like "A" and "B" are acting as placeholders for which wffs are to be substituted. So even rules with no independent connection to necessary truth are chosen to be useful for knowing that when a wff like "$\forall x$M(x) -> $\exists x$M(x)" is verified by the use of the method, it expresses something made true solely by the fact that the definitions of its non-individual constants (which definitions presuppose the rules being what they are) are what they are.

5.

One important upshot of the distinction between knowledge of logical necessity and computational knowledge is that the following relation holds between logic in the traditional sense (TL) and classical modern logic (CML): Computational methods are indispensable *tools* for determining the validity of forms of inference, that is, for achieving TL's goal of explicitly knowing the validity of forms of inference. These tools are indispensable because they are incomparably clear, precise, comprehensive and progressive. By "clear," I mean the rules and definitions are neither vague nor ambiguous. By "precise," I mean rigorous; CML reduces the verification of validity to the recognition that each step in a procedure satisfies one or more of a finite set of rules mechanically specifying the formation, positing, combining and detachment of finite sets of marks. By "comprehensive," I mean that CML's methods can verify the validity of some inference forms we pre-computationally know to be valid but that the methods of TL could not show to be valid (apparently[20]). By "progressive," I mean that the ways in which CML can expand our knowledge of the validity of inference forms may be limited only by our ingenuity in constructing computational methods.

The preceding paragraph is meant to express facts about the relation between TL and CML that everyone already knows. Not so the following statements which, I am arguing, are also true. To the extent that the recognition of neither the correctness of computational procedures nor of what the rules are that make them correct need amount by itself to the recognition of any necessary logical truth, the tool that CML provides for achieving the goal of TL is a different epistemic type from TL. Here, I am not just repeating the commonplace that, relative to other kinds of knowledge, logic is knowledge of a tool that those other kinds of knowledge use: inferential validity. That is true; logical knowledge is knowledge of that tool. But relative to logical knowledge, therefore, computational knowledge is a tool for knowledge of a tool, a tool once removed. In class, Hilary Putnam called Quine's requirement that, for referring, sentences be regimented in Fregean fashion "the sacrilization of logic." It does not lessen the importance of Putnam's insight to note that,

strictly speaking, Quine was not sacrilizing logic but logic's use of a particular tool.

Epistemically speaking, the role of computational methods for achieving knowledge of necessary truths of logic is analogous to the role of mathematics for physics. The difference between knowledge of computational correctness and knowledge of necessary truth does not prevent computational knowledge from being an indispensable tool for achieving knowledge of necessary truth any more than the difference between mathematical knowledge and physical knowledge prevents mathematics from being an indispensable tool for physics. Knowledge of mathematical truths is not the same as knowledge of truths of physics, but we can hardly get anywhere in physics without mathematics. Likewise, knowledge that steps in a computational process satisfy rules is not the same as knowledge that any rule or formula reached by means of the steps expresses something logically necessary, but computational methods of inference are just as much indispensable tools in logic as mathematics is in physics. Logic can start without them but can hardly get anywhere without them in comparison to where it can get only with them. CML is a way of achieving the goal of TL, but CML does it by means of an incomparably more powerful tool, a tool without which you cannot get any further in logic than you can get in physics without mathematics.

To describe mathematics as a tool that serves something other than itself is not to imply that mathematical knowledge is not intrinsically valuable. Not only is mathematics an entirely legitimate kind of knowledge on its own, but if it was not entirely legitimate on its own, it would not be a useful tool for physics. Relative to the knowledge physics acquires, mathematical truths are "only" a tool (an indispensable and incomparably powerful tool), a means, not the end; mathematical truths are not the kind of truths that physics seeks to discover and verify. But for mathematics itself mathematical truths are not only a tool; they are the truths that mathematics seeks to discover and verify.

Just as mathematics is entirely legitimate subject on its own, not merely as an aid to physics, so the study of computational methods is an entirely legitimate subject on its own, not merely as an aid for verifying the validity of inferences in standard logic. And just as the

mathematical truths physics uses are truths that mathematics studies by mathematical methods, so CML not only uses computational methods to verify logical validity, but it studies computational methods by computational methods themselves. In fact logicians have for some time been rightfully more interested in the study of computational methods by computational methods than in simply using computational methods to verify the logical validity of formulas. Similarly mathematicians are more interested in studying mathematical truths for their own sake than as aids to other things. So to describe knowledge of computational correctness as a tool that serves something other than itself, knowledge of necessary truths about forms of inference, is not to imply that CML is just of a servant of TL.

Among the topics mathematicians study for their own sake are things that have no foreseeable applications, scientific or practical. But it is unlikely that mathematics would have advanced as far as it has had it been proven so useful in fields other than mathematics. Likewise, it is unlikely that modern logic's interest in non-standard logics and in using computational methods to study computational methods themselves would have advanced as far as it has if computational methods had not first proven so powerful in verifying the validity of inferences in consistent and bivalent logic. Why would anyone have tried to prove the consistency and completeness of computational systems, for example, if those systems were not already known to be such a powerful method of establishing inferential validity?

But there is this difference between mathematics and physics, on the one hand, and computational methods and logical knowledge, on the other: With respect to knowing that truths are necessary, the roles of the tool and the field using the tool are reversed. Mathematics, the tool, knows truths to be necessary while the truths of physics, the field using the tool, are not epistemically necessary. But knowledge of computational correctness, the tool, is not knowledge of the epistemically necessary, while logic, the field using the tool, knows necessary truths as such. Still, since logical knowledge is of the necessary while computational knowledge is not, computational methods are to logic as mathematics is to

physics: indispensable tools epistemically different from that of which they are the tools. So nothing here demeans the role of computational methods in logic.[21] And although non-standard logics are not interested in the validity of traditional inference forms, still we could not know that their theorems are validated by their rules without implicit knowledge of the validity of the standard inference principles that we must use to draw reach that conclusion.[22]

Although computational methods are an indispensable tool for logic, like any tool, they have limitations. I am not just thinking of the fact that each computational method has limits in comparison with other computational methods; for example, Sommers developed a computational system that has advantages over Frege's, while Frege's has some advantages over it. I am thinking of a much more basic limit: Every time we construct a new computational method for verifying logical validity we find that it has characteristics that are, at a minimum, anomalous by the standard of some knowledge about what is or is not logical necessary that we already possess. But the occurrence of such anomalies need not affect either the technical success of the method as a computational system or it usefulness as a tool for verifying necessary logical validity.

For example, given the standard truth-functional definitions of the operators, wffs like B -> (A -> B) and -A -> (A -> B) express necessary truths; for if they do not express truths, the truth conditions defining the operators both are and are not what they are. But we know that, as their terms are defined, these formulas do not and cannot express any precomputationally known necessary truth. (In other words, we can explicitly recognize that not all truth-functionally defined validity is "logical" validity in the sense that its necessity derives from that for which we use any of the vocabulary we call "logical" in the necessary truths whose knowledge is presupposed by computational knowledge.)

Likewise, standard computational methods show the necessary truth-functional validity of a formula like '$\exists x(Rax \supset Rxb)$'. Because of that computational validity, Quine calls "For any two people, John and Mary, there is someone who, if admired by John, admires Mary" *logically true*.[23] But we are able to recognize that this is only an artifact of an

(indispensable) *tool* of logic; for we know '$\exists x$(R$ax \supset$ Rxb)' is necessarily true only relative to definitions stipulated by and for the use of that tool, and that under those definitions it does not express any precomputationally known necessary truth. For on the basis of that for which we use the vocabulary we call "logical" in the necessary truths whose knowledge is presupposed by computational knowledge, we can know that the following are both possibly true at the same time:

(i) John admires everyone but Mary, and no one admires Mary.

(ii) Necessarily, if John admires Mary, John does not admire John.

If it is possible for (i) and (ii) to both be true, '$\exists x$(R$ax \supset$ Rxb)' cannot express a necessary truth. And as far as we know, nothing that is made necessary by the way we precomputationally use "logical" vocabulary makes it impossible for (i) and (ii) to both be true.

(Where could the necessity in (ii) come from? The above definition of necessity in terms of contradiction did not stipulate that every candidate for the what-it-is in 'would be what it is and the same time that it is not what it is' must be something of a logical nature, such as implication, conjunction or alternation. Before children are sophisticated enough to perform the abstraction needed to believe that there are universal laws behind each of the radically individual and unrepeatable concatenations of elements that constitute the contexts of their lives, and so long before they are confused by reading Hume, they believe in causal connections between what one real thing or event is and some distinct real thing or event is. That is, they believe that what existent A is has something do to with distinct existent B is. The prephilosophical existence of such beliefs is reflected in this paragraph in the use of the worlds 'could' and 'come from' [in the first sentence], 'perform the abstraction needed' and 'confused by reading Hume' [in the third sentence], and 'reflected' [in this sentence]. And they believe that there are necessary causal connections, since they believe that no change would exist without the existence of something other than itself. We, who are just those children at a later age, can express that necessity by saying that if A exists and B does not, then at least one of A and B both is what it is and is not what it is. If

postulating such a causal connection between John's admiring Mary and John's not admiring John violatee any precomputationally known logical necessity, we would not have any precomputational beliefs about causal necessity.)

C. I. Lewis was right that the necessity of B's being true if A -> B and A are true is included in our precomputational understanding that for which "if . . . , then" is used (whether or not we explicitly express that necessity with modal vocabulary). But it does not follow that a computational modal logic can capture that for which we use "if . . . , then" in a way that does not generate anomalies by the standard of precomputationally known necessary truth. Because the prior grasp of necessary truth on which the grasp of computational correctness depends is non-computational, computational methods will always be anomalous if considered to be models of how we know necessary truths of logic. Since knowledge of computational correctness presupposes knowledge gained noncomputationally, there must always be something lacking in computational methods *if they are considered as models of what they PRESUPPOSE* and which they therefore cannot contain. But it does not follow that they lack anything considered in themselves.

Consider also another kind of anomaly. T. H. Irwin finds Aristotle's account of valid deduction "relatively narrow" as compared to what is now "a more familiar" account: "An argument may be valid even if it is redundant, or a premise is identical to the conclusion, or it has only one premise . . ."[24] But no one would consider 'p -> p', for example, a deduction — and much less a valid deduction — on the basis of their implicit pre-computational grasp of what valid inference is. It is only after learning modern propositional logic and being justifiably impressed with the unsurpassable power of computational methods of verifying validity, that we conclude that 'p -> p' should be considered an inference because it meets the tests for being a wff and a tautology in the best method we have for determining validity. And there is nothing wrong with the propositional calculus because it puts 'p -> p' on a par with '((p -> q)p) -> q'. That is not an error any more than material implication is an error. It is just another limitation, from the point of view of our pre-computational grasp

of valid inference, characterizing this tool for achieving explicit knowledge of logical necessity. Such anomalies are a small price to pay indeed considering (1) that any tool in any kind of endeavor will have limitations and (2) the power that this tool gives us

To illustrate the point that any computational tool for verifying noncomputationally known necessary truths will have limitations, consider the system by which Fred Sommers has shown, against the previously all but universal view, that the traditional "two term" analysis of propositions can validate all the inference forms that Frege's function/argument analysis can validate. Sommer's method succeeds only by introducing its own anomalies. And, again, there is nothing wrong with that. For example, he shows that two-term categorical syllogisms can handle the relational propositions that only Fregean polyadic predicates are supposed to be able to handle. To do so he, following Leibniz, uses redundant premises like "Every R to an X is R to an X."[25] But by the standards of precomputational knowledge of inferential validity, using a redundant premise is no more anomalous than counting 'p -> p' as an inference.

Sometimes we describe our precomputational grasp that a logical truth is necessary as "intuitive." So the anomalies we always find in computational methods are judged to be so by the standard of pre-computational intuitions. But a common implication of the word "intuition" is that intuitions can result in false beliefs. Computational methods can replace reliance on intuition in the sense of something that can be wrong. They cannot replace but depend on actual pre-computational knowledge that certain logical truths are necessarily true. So somethings that are anomalous by the standard of logical truths whose necessity we must know , rather than just believe, will always characterize computational methods for logical verification. (More on how we precomputationally know necessary truths in Part II.)

One of the most well known limitations of computational tools is, of course, that no (consistent and bivalent) computational system adequate for expressing arithmetic can prove all the truths expressible in the system. But my point about the inevitable limitations

of computational methods comes from a perspective more epistemically fundamental perspective than Godel's. My point is from the pre-computational perspective of the epistemic conditions for the knowledge of necessary logical truth needed for grasping computational correctness, while Godels point is from a perspective internal to computational methods, the methods whose use for acquiring knowledge presupposes the conditions I am examining. As such, my point about the limits of computational methods concerns computationally proven and provable, not unproven or unprovable, truths. To know that a computational process proves anything, we must know necessary truths of logic otherwise than by computational methods.

This point is closer to Putnam's demonstration that how we know a proof to be 'sound' (here meaning logically valid with true premises) cannot be formalized in a system, including a formal system of inductive logic, that we could recognize, computationally, as itself sound without running up against Godel. From there, Putnam concludes that "reason can go beyond whatever reason can formalize" (his emphasis).[26] I am trying to be more specific, as well as more fundamental, by showing that one way reason *must* go beyond what it can formalize is by noncomputationally knowing necessary truths presupposed by our ability to recognize success in formalization.

Also, machines can arrive at correct computational results. But since current (and all future?) machines are so designed that their abilities are restricted to achieving what can be achieved by computational methods, machines are not capable of the kind of recognition we achieve in the noncomputational knowledge on which knowledge of computational correctness depends. The inability of machines to achieve the kind of knowledge of the necessary that we have is itself a (causally) necessary consequence the fact that knowledge of computational correctness presupposes knowledge gained precomputationally.

Certainly, we have sufficient evidence that some animals have a type of knowledge that a rule they have learned has been violated. They demonstrate that by their negative reactions if a behavioral expectation we have given them in teaching them a rule, for

example, that such and such behavior will get such and such a reward, is frustrated. But that is not evidence that they grasp the logical relation of valid inference. Expecting that an event will occur is one thing; knowing (certitude caused by awareness of evidence sufficient to exclude the opposite is true) that a statement asserting the occurrence is true is another. Likewise, many higher animals can acquire an expectation based on the repeated connection of two kinds of events; that is not the same as inductively concluding the truth that past evidence makes it unreasonable to believe the opposite of a universal statement asserting that connection. And animals can use words with universal meanings, but that gives us no more evidence that they can recognize the logical relation of universality than we have that they can recognize the logical relation of implication or the epistemological relation of justifiable belief. To show that animals can have an even *implicit* knowledge that an inference is valid, we would need evidence that they can have either implicit or explicit knowledge of the truth of the premises of the inference. For it is in knowing or hypothesizing the truth of sentences using words like 'if . . ., then' and 'all', that we have implicit knowledge of the necessary validity of inference forms.

Interestingly, these caveats about machines and animals do not directly disprove materialism in the philosophy of mind; they just disprove certain arguments for materialism. But what if it turned out to be the case that the methods of empirical science are restricted to giving information about processes that can be represented by computational models? As we know, the methods of science are restricted to giving us information about certain fundamental things only statistically, not deterministically. Likewise, those methods might be restricted to only giving us information expressible in algorithms of physical processes. In that case, scientific theory could cover every scientifically knowable piece of empirical data about how our brain functions in recognizing logical necessity while telling us nothing about how we noncomputationally acquire the grasp of inferential validity that judging inferential validity by computational methods presupposes. Would this scientific inability disprove materialism? No, it would only be

evidence for the limitations of scientific methods, something we know already from science's inability to know the speed and location of a particle at the same time or the simultaneity (which we know to hold; see Chapter 8 of *Causal Realism*) of distant events.

6.

Having begun above to discuss the epistemic dependence of logical knowledge on the exclusion of contradiction, I will briefly illustrate its dependence on bivalence. Peter Rutz has shown, computationally, that any formula of a multivalent logic is equivalent to an *n*-tuple set of bivalent formulas such that all multivalent theorems are deducible from laws of bivalent logic.[27] But, again, my interest is in pre-computational epistemic conditions for doing logic computationally.

Consider the trivalued '$\supset$' which is defined to have value .5 when '*p*' has value 1 and '*q*' value .5, or '*p*' has value .5 and '*q*' value 0. If so, a formula like

(1) $((p \supset q) \supset p) \supset p$

would have value .5. But when we are aware that the trivalued definition assigns (1) the value .5, we are also aware that the definition either assigns a formula the value .5 or does not assign it .5, and that if a formula does not not have the value .5, it has that value. If we were not aware of these bivalent truths, we could not be aware of how any multivalued truth-functional operator is used. Nor is this a case where we can metalinguistically climb a bivalent ladder and then kick the ladder away once we have constructed the multivalent object language. If rules expressed by means of a bivalent metalanguage add the value *X* to the standard 1/0 truth tables for evaluating formulas in an object language, the object language will be trivalent. But that does not free us from needing to know the bivalent truth that formulas in the object language either have value *X* or do not have value *X*; and if they have value *X*, they do not not have value *X*. So it remains the case that the logic involved in knowing that a formula in the object language has been assigned a certain value must be standard logic, as must be the logic used in concluding that steps in the object language conform to the rules stated in the metalanguage. Any truth we know about multivalent

formulas is bivalent.

So the issue here is more basic than whether a metalanguage defining a multivalent object language must be bivalent. To invoke the language/metalanguage distinction as a way to avoid the epistemic dependence of nonstandard logic on standard would be to beg the question of whether knowledge of computational correctness is the same epistemic type as knowledge of logical validity; for the language/metalanguage distinction is relevant to logic only to the extent that it is needed for the use of computational methods, whose relation to knowledge of logical validity is the question at issue. The same kind of epistemic questions arise about an object language, a metalanguage or a meta-metalanguage, questions such as 'How can we be aware that step 2 in this procedure is necessarily justified by step 1 and rule A,' or 'How do we know that rule B necessarily preserves truth.' These are epistemic questions about our knowledge of logical truth. They and their answers are not in a metalanguage, if a 'metalanguage' is defined by the *purpose* (or *function,* if 'purpose' sounds too psychologistic) of expressing syntactical or semantical rules for another language. From the viewpoint of our precomputational knowledge of necessary truths, it is only incidental that the language in which they are expressed can also be used for that purpose, although it is hardly incidental to our knowledge of computational correctness that precomputational language can also be used for that purpose. But we knew necessary truths expressible in precomputational language long before computational systems useful for establishing inferential validity even existed.

And if we use a meta-metalanguage or a meta-meta-metalanguage, the same kind of questions, with the same kind of answers, arise at each level. If the rules expressed in a bivalent meta-metalanguage add the value *Y* to the standard 1/0 truth-tables for evaluating formulas in the metalanguage, the formulas of the metalanguage will be trivalent, and so the rules expressed in the metalanguage for the object language will be trivalent. Assume that the rules expressed in the trivalent metalanguage make the formulas of the object language similarly trivalent by adding value X. Then, we can know the bivalent truth that formulas in

both the metalanguage and the object language either have value *X* or do not have value *X*, respectively; for if they do not not have value *X*, they have value *X*. So, it is not the case that although (a) the logic involved in knowing that steps in a metalanguage conform to the rules expressed in its meta-metalanguage must be standard, still (b) the logic involved in knowing that steps in the subsequent object language conform to the rules of its metalanguage need be no more standard than is the logic expressed by the rules of its metalanguage.

I do not intend to discuss intuitionism (or anti-realism) any more than to note that identifying value 1, say, as "proven" and value 2 as "disproven," and even adding value X identified as "neither proven nor disproven," would not affect the epistemic bivalence that comes from having the use of signs for negation in the ordinary sense. Of any statement we can know either that it is either proven or not proven (which is not to say that it is either proven or disproven), and either disproven or not disproven (which is not to say that it is either disproven or proven). And we can know that it is neither proven nor disproven OR that it is NOT neither proven nor disproven, and we can know that IF it is NOT neither proven or disproven, THEN it is either proven or disproven.

The reply may be that I have not really engaged the opponent because I have changed the rules, that on the basis of the intuitionist's definitions, he agrees that we can know that something is either proven or not proven, since not being proven differs from being disproven. Yes, but in so agreeing the intuitionist is really granting my EPISTEMIC point, independently of whatever kind of point, perhaps metaphysical, he may be trying to make. It is the intuitionist or anti-realist who, from the epistemic point of view, can only succeed in changing the subject, namely, negation as ordinarily understood and as understood in the context of standard logic, whether traditional or classical modern).

Epistemically, our knowledge of the law of excluded middle is caused by our knowledge of the use that negation signs ordinarily have, the work that we happen to, but need not, give "not" and "-", the work of applying the relation *other-than* and communicating it.[28] Again, the existence of other kinds of negation is not an issue. Once we have achieved

the grasp of ordinary negation, once ordinary negation is part of our conceptual equipment, we are unable not to know, when fully conscious and attentive to the relevant factors, linguistic and/or other than linguistic, that a statement is either proven or not proven, disproven or not disproven, etc. That just happens to be the job that "not" currently has.

And it does not matter if the use of negation for statements is logically or psychologically prior to its use for predicates. That is a different kind of question. However and whenever we acquire it, once we have negation, as exemplified by the way "not" is ordinarily used in constructions like "proven or not proven," in our conceptual equipment, we are stuck with it (fortunately!). And if we have not yet acquired it, we are capable of acquiring it at any time. We can then fail to grasp the truth of certain statements only by not being attentive to something we know, namely, what negation is, since the truth of those statements is caused by what negation is. In other words, we can fail to know those truths only by ignoring the relevant factor. There can be many reasons for ignoring a relevant factor, especially the influence of philosophical problems, questions, mind-sets, cultures and/or theories that appear to be relevant, due to the causal complexity of ALL philosophical issues, while causing us to focus elsewhere than on the relevant factors.

Nothing here detracts from the potential value of studying nonstandard computational systems either for their own sake, for their applications, or for what they can tell us, by way of contrast, about standard logic. But we could not be aware of that value, or of the truth of sentences stating that such systems have a value, if we were not at the same time implicitly aware of the necessary truth of principles of standard logic. Nonstandard systems have value for us only to the extent that our knowledge about them and their applications is consistent and bivalent. Likewise, philosophical critiques of principles of noncontradiction or excluded middle.[29] must rely, explicitly or implicitly, on recognition of the validity of deductive inferences of standard logic. The paraconsistent logician wants to get a conclusion like 'We can avoid the result, "$(p \neg p) \supset q$"' from the premises 'If disjunctive syllogism is dispensable, we can avoid the result, "$(p \neg p) \supset q$"' and 'Disjunctive syllogism is dispensable.' But he could not

grasp the validity of that inference if he did not rely on implicit knowledge of necessary truths whose necessity derives from the same precomputationally known meaning, namely, negation, that makes disjunctive syllogism necessarily true.

??

How, then, do we precomputationally know necessary truths of logic? I have said that we know the validity of modus ponens solely by knowing that for which "if . . . , then" constructions happen to be used. Why should we believe that is our sole evidence for modus ponens? That is all we are hypothesizing when

5.

An instantiation proof procedure may have a rule like the following for a prenex formula:

(B) An initial existential quantifier may be dropped.[30]

If so, awareness that dropping an '$\exists x$' is legitimate because it is in accord with rule (B) is an implicit awareness that an argument like the following is valid:

(1) If the '$\exists x$' in formula (III) is an initial existential quantifier, it may be dropped.

(2) The '$\exists x$' in formula (III) is an initial existential quantifier.

(3) The '$\exists x$' in formula (III) may be dropped.

We cannot get any place requiring deductive inferences if we permit contradiction. Another obvious example will reinforce the point. Using our metalanguage, we may posit a rule like

(B) Any wff can appear as a premise on any line in a proof.

If so, awareness that positing premise *p* is legitimate because it is in accord with (B) amounts to an implicit awareness of the validity of

(1) If *p* is a wff, *p* can appear as a premise on any line in a proof.

(2) *p* is a wff.

(3) *p* can appear as a premise on any line in a proof.

We cannot even be aware of the legitimacy of introducing a premise, if we are not aware of the validity of an inference belonging to standard logic.

??

[My approach to bivalence will be better appreciated if my approach to non-contradictory necessity is explained first; better appreciated after my discussion of that epistemic value for the sake of which negation signs are ordinarily used.]

xxxx

The reason they are a useful methods is that we can perceive some sort of "connection," "correlation," "link," "similarity," "translation," etc. between the rules and premises of formal methods and the "laws of logic," whatever that might mean. I do not need to know what that means; nor do I need to be able to make more specific what "correlation," etc. mean here. For all I need to know is that some sort of link between the rules and something else (which I happen to call "laws of logic") is broken when I substitute contradiction. When I do that, something that was there all along is no longer there. I do not have to know completely what that something, a relation to X, is. Rather, I now have sufficient knowledge to motivate me to wonder further what that something is. But I am not guaranteed, nor need I be, of any success in finding out further what X and this relation to it are.

The principle of noncontradiction is the most basic Law because all propositions would follow logically from contradiction, assuming that all other logical laws were true. But for that very reason, the others are not logical laws if contradictions are true. Because everything would

follow indiscriminately. That is, it is not that this proposition, q, must follow because *linked logically* to this other, p.

Negation in is the only operation on atomic propositions that does not form a molecular proposition. Hence negation is the more fundamental operator. And the exclusion of contradiction simply expresses the effect of negation on a proposition negated; the intended effect of negation on the proposition negated.

Prior may say that anything follows logically from contradiction and from the other rules of the system. But those other rules no longer map to logical truth if they do not exclude their contradictory opposites from truth. For then I can introduce, for example, "((p -> q) & p) -> -q" as a premise.

An argument showing that everything follows from contradiction in logic is like a proof in mathematics that uses division by zero.

Somebody, perhaps it was prior, defines negation by using the implication sign. But how do we know the truth of modus tollens? By knowing the truth of the principle of noncontradiction and, here, knowing what negative signs express.

Rutz's point is within logic itself; in addition to that, there is the following epistemic point.

I ask Putnam if he is saying that science will and will not reject the PNC. He replies that science will reject it only in selected areas. The areas, in effect, are a sub-lanaguage under a metalanguage. But that only amounts to saying that in this sublanguage or this area, what are

negation signs in the metalanguage will not have the function of negation signs in the sublanguage, nor will any other signs in the sublanguage have the function that negation signs have in the metalanguage. Pena must face the same problem.

Another thing. How can a sublanguage not obey the same logical, as opposed to linguistic, rules that govern the metalanguage? If the sublanguage really is spawned by the metalanguage, then it really is an extension of that language, a part of it. Certainly, artificial languages are extensions of natural languages, since we can only create artificial languages by using the tools of natural language.

This is not a language/metalanguage issue. It is a matter of recognizing, in the case of multi-valued logics, that if p has a value, e.g., assigned to it, p does not not have that value assigned to it, or p does not both have that value assigned and not have that value assigned. Or it is not even a case of "recognizing...". First of all, it is a case of p having that value assigned and so not not having that value assigned, or a case of p having that value assigned or not having that value assigned. And if it has one or the other, it then does not have the opposite. Then there is our recognition of those facts as expressed in language. What is important in order to be able to express that in language is that the words of the language has certain functions, not that they relate to other words as if they were a metalanguage relative to a distinct language constituted by those other words.

The latter relationship is important for constructing systems of marks for achieving certain ends, e.g., modelling logical relations. But that relationship is not important or is irrelevant to the end of expressing the fact that a certain value is assigned to p and the recognition that therefore it is not the case that p does not have that value.

In the case of knowing necessary truth, the goal attained is supposed to be linguistic (or in earlier times, conceptual, or logical) in a way that they goal attained in knowing contingent truths is not linguistic. The reason why the value achieved by a necessary truth is supposed to be linguistic is a causal argument. The cause of our knowledge of the truth is linguistic, an understanding of the meaning of its terms. Since there can be no more in the effect than the cause can put there, the knowledge produced by this cause must be linguistic in some reductionist sense.

That argument would be perfectly valid, if the cause of our knowledge were linguistic. But the knowledge of meaning required to know necessary truth is not lexicological knowledge of meaning. A person can be linguistically mistaken about the meanings of "and" and "or" in English. It does not follow that if he affirms a sentence like "p and not-p", he is adopting a non-standard logic. And there can be behaviroal evidence that he is confusing "and" and "or."

To claim that we have no right to say that the science of the future will not causes us to revise the principle of noncontradiction (or the principle that a change happening to something would not exist without the thing to which it happens) is to say this: that which the science of the future may tell us about what things are will be that things are not what they are; that which the science of the future will discover about what things are will require that things are not what they are. In other words, to claim that now is to imply a contradiction now; so we must give up noncontradiction now, i.e., believe that negation is and is not negation now.

We must use logic to do logic, and the logic we must use is bivalent and consistent, even if the logic that we do by its means is multivalent and/or dialectical.

The paragraph on "not not .5" Of course, this assumes a meaning for "not", but it would be irrelevant to argue whether this meaning is in the metalanguage as opposed to the language. The important thing is that for which "not" is used; that which it communicates, at whatever level it is communicated. The important thing is what we know, by implication, if we know that it is true that p does not not have value .5.

To really deny the PNC, a principle would have to allow a proposition to have value M and not have value M.

The validity and constraint imposed by the P of NC has nothing to do with whether a formula corresponding to it appears in a particular linguistic construct.

The language/meta-language distinction functions in explaining how we are aware of validity in formal methods. So that distinction is NOT of use in explaining logical awareness. We need some language, of course, but that is all.

The usual reply to Carol's paradox is that rules are not premises but there is more to it. For we must grasp the truth of the rules by knowing the words of the premises; otherwise he cannot be aware of the validity of the argument. Therefore we must now investigate self-evidence (or "the analytic"). Doing so will illuminate the relevant issues such as synonymy and Quine's critique of the analytic. (If they are known just by knowing meanings, we know them by [knowing them amounts to] knowing that if they are not true, some meaning would both be what it is and not what it is [meaning in the objective sense would not be what it is, in contrast to meaning in the sense of a relation between a word and that for which the word is used would not be what it is].)

We cannot *arrive at* a pragmatic view of which logic to use without using inferences, and we cannot *apply* the pragmatic view once arrived at without using inferences. For the reason we accept a theory T on pragmatic grounds are the implications of T, in contrast to the implications of competing theories. E.g. the implications of T may be simpler than those of other theories.

For any awareness of the validity of an inference is an implicit awareness of the necessary truth of a rule stating that any inference based on the linguistically constituted relations this inference exhibits is valid.

Rather, I so design, by conscious awareness, the rules for marks in a formal method, that I am aware that they can do at least some of the work I want logic to do, i.e., that I know logical relations like entailment do.

proving logical truths by using formal languages and abstract formulas on which you operate according to rules, rules which save the truths of logic.

Is logical knowledge knowledge of the correctness of the steps in a proof as opposed to knowledge of the necessary truth of inference principles? Perhaps we want to say this. But if so there is another question. In addition to knowledge of the correctness of steps in a proof, an algorithm, is there such a thing as knowledge of necessary truths that are or that correspond to logical inference principles? I am arguing not only that there is such a thing but if there were not such a thing, we could not have knowledge of the correctness of the steps in a proof.

To draw a conclusion from two statements you need another statement saying that the connectionbetween the first two statements justifies the conclusion. But then you are drawing a conclusionfrom three statements and need a fourth statement saying that the connection between the firstthree statements justifies the conclusion.

On page 96 of "truth by convention," Quine says that to use a general rule to derive a specificinstance justified by the rule, a logical inference is required. But this requires another general rulecovering such derivations, and to derive the first derivation from this other general rule is anotherlogical inference requiring another general rule.Why is this required? Because every move has to be justified by deriving this specific case froma general rule.

So knowing that a specific instance follows from a general rule is not itself a matter of applying ageneral rule to this specific instance of deriving an inference from another general rule. Knowingof the validity of the derivation is not an algorithmic process, that is, a process of steps performedby deriving a specific step from a general rule about steps. Simply making a series of marks thathappen to conform to rules is not the same as knowing that they conform to rules. So knowledgeis not an algorithmic process.

Either the algorithmic derivation is not a logical inference (Quine's term), or logical inference isnot an algorithmic process. If it is an inference, inference is not algorithmic. If it is not aninference and inferences are algorithmic, knowing that the inference is valid is other than makingthe inference.

The standard reply to Carol's Paradox is that he does not distinguish between a rule and apremise. And yes, he does not make that distinction. But that answer would make knowledge of validity in matter of recognizing that two premises are an instance of the kind of case covered by the rule. And Quine's critique of analytic truth comes in here. How do we know that this is an instance of the kind of case covered by the rule, if not by the same process over again?

Carroll's Paradox shows that knowing that the validity of an inference is not knowing another premise, specifically, a premise that would express why the inference is valid. And so we

do not need to explicitly formulate the rule to see a logical necessity. If we need to explicitly formulate it, it would work like another premise.

But in formal methods we must explicitly formulate the rule that a step satisfies; that's the wholepoint of using formal methods. So that explicit formulation is not the same as awareness of the logical necessity of a step. If that were how we knew the logical necessity, that is, if we know logical necessity by knowing that the step satisfies an explicit rule, the knowledge of logical necessity would require an infinite regress.

If logical relations between premises necessitate the drawing of certain conclusions, then alogical law stating that fact need not enter the reasoning as a premise from which the conclusionfollows. To derive the conclusion, one need not enunciate the law expressing what follows fromthe logical relations. One need only grasp the logical relations themselves.

If a law was correct, it would tell us that someone who knows the relations, not someone whoenunciate the law, nor was that the conclusion follows. And someone knows the relations byknowing both of the premises and observing their logical relations to each other. By observingthe relations, you see that if the premises are true, the conclusions must be true, that is, that *thetruth of the conclusion differs from the truth of the joint premises (or the joint truth of thepremises) only by logical relations*. That is what the law would enunciate. But you see this notby enunciate in the law but by enunciate in the premises.

What is a valid argument? It is a set of statements with certain logical relations between them.

What is a logical law? It is an objectification, a statement expressing, the logical relationsbetween statements that make a set of statements a valid argument. When we know thestatements, we are aware of the logical relations between them whether we objectify thoserelations in a logical law or not. That is what logical relations are, relations we are aware of whenwe are aware of other things.

So the logical law is not needed as a premise in the argument; it is superfluous as a premise.Even more, if it were a premise, we would need to know the logical relation between it and theother premises without expressing these logical relations in a law. In grasping any validargument, we need to know the logical relations otherwise than by expressing them in a logicallaw (the same with knowing a self-evident truth other than by applying criteria for self evidence).

1)awareness of what negation (the relation other-than) is is not lexicological awareness of the happenstance that that relation is what a certain mark is used for; so awareness of meaning required for logical truth is not lexicological awareness. We can be lexicologically mistaken (e.g., by thinking "not" is used the way we use "or" -- and their can be behavioral evidence for this), and logically correct. 2) in non-lexicololgical awareness of meaning, the awareness is something "mental" is a psyghological sense, but that of which we are aware, the "meaning" need not be mental in that sense. Logical meanings may be mental in the sense that they are only objects of awareness, but they are not mental in the sense of . ...

The atomistic/holistic problem is a false dichotomy. Sure, to understand the meanings involved in a self-evident truth, we have to know the entire background language. That is a statement concerning the necessary causes of

the discovery or pedagogical learning of those meanings, not a statement of how statements using those meanings are verified.

Aquinas in the summa, part one, question 85, article six and in the commentary on the ninth book of the metaphysics, lecture 11, strongly implies that "knowledge of terms" in knowledge of self-evident truths is knowledge of extramental essences, not linguistic relations.

I will show that by showing why arguments believed to prove that ex contradictione quodlibet (ECQ) do not and cannot succeed in doing that; for we can be as explicitly aware as you please that though the conclusion that ECQ results from the correct application of rules in an argument, the rules themselves lose their force as expressing or "corresponding to" any necessary truth of logic once we permit contradiction.

As Graeme Forbes said on a related matter:

If the necessary/non-necessary distinction is, or can be made, sufficiently clear for the purposes of this discussion, still the implicit/explicit distinction might not be. I will now show that the distinction, in the grasp of computational correctness, between the grasp of conformity to a rule and the grasp that some truth is necessary does not depend on the implicit/explicit distinction.

We can, of course, make distinctions any way we want; we could divide logic, for instance, into that done by blue eyed people and that done by non-blue eyed people. By distinct epistemic types here I mean a difference between

belonging to standarof the validity of inferences requires awareness of necessary truths of

standard logic.

in logics that are multivalent or permit contradiction,

epistemic types must be present in.

But recognizing that something can be inferred from a rule requires a grasp of the necessary validity of an inference principle justifying the transition from rule to result.

the validity of an inference is a grasp that the inference is an instance of a necessary truth.

I am trying to drive a wedge between two epistemic types both present in the grasp of the correctness of a step in a computation. One is the explicit grasp that a step conforms to (is an instance of) a rule; neither the rule nor the step need be or correspond to a necessary truth. The other is the implicit grasp of the validity of an inference principle used to accomplish the first grasp, and I will argue that the second grasp is a recognition that some statement is necessarily true.

Carroll's paradox occurs only if recognition of the validity of the inference requires making modus ponens explicit. The recognition cannot require that.

for achieving TL's goal of explicit knowledge of the validity of forms of inference

Again, if the latter recognition required an explicit awareness, we would be in an infinite regress, as Carroll's paradox shows. When this third kind of recognition is of a derived formula for modus ponens, This is another explicit awareness of necessary truth of a formula expressing modus ponens.

either express or "correspond to"

necessarily true about inference principles like modus ponens or dictum de omni such that

we know

Computational methods are indispensable tools

And we may want to say that in some cases, we have a third kind of knowledge of modus ponens.

justifying validity of the transition from rule to result, knowledge of computational correctness, even in non-standard logics, requires knowledge of the necessary truth of inference principles of standard logic.

Since it is the case that, in a addition to a grasp of what the rules and steps of a computational process are, knowledge of computational correctness requires an implicit awareness of the necessary validity of an inferential relation, then ML deserves to be considered a distinct epistemic type from the explicit recognition of necessary truths about inference that is the goal of TL. For since there is such a thing as the grasp that a truth is necessary, the difference between that knowledge and knowledge of contingent truths constitutes an important and interesting epistemic distinction if anything does.

Just as Putnam says Tarski is technically successful and philosophically irrelevant, I am saying that computational methods are irrelevant to the nature of logical knowledge. That is,

The method is a tool for checking the validity of inferences, for determining the validity of inferences computationally. But is that the way we know logical truth?

Note, however, that explicit awareness of the connection to necessary truth of

neither a rule nor a result is the same as the implicit awareness of the necessary truth of an inference principle that is required to grasp the validity of the inference by which we recognize that a computational step is a correct application of a rule. For example,

Still, computational methods do not achieve the goal of knowledge of logical necessity by themselves??

The only "intuition" required is the (for all practical and relevant purposes) uncontroversial intuition that a mechanically specifiable step conforms to a rule of legitimacy mechanically specifying the step. (The discovery or invention of the rules also involves intuition, but since these can take place apart from verification, the intuitions need not be the same.)

(1, 0, 0, 0; 1, 1, 0, 0; 1, 1, 1, 0; 1, 0, 1, 0; 1, 0, 1, 1; 1, 0, 0, 1; 1, 1, 1, 1; 0, 0, 0, 0; 0, 0, 0, 1; 0, 0, 1, 1; 0, 1, 1 ,1; 0, 1, 0, 1; 0, 1, 0, 0; 0, 0, 1, 0; 0, 1, 1, 0; 0, 1, 1, 0).

Still, knowledge of computational correctness is different from the knowledge of logical necessity that is the goal of TL and of much of ML.

Still, the knowledge of logical necessity given by computational methods presupposes knowledge of logical necessity not gained by computational methods.

, which together with many-valued logics I will call non-standard logics,

Deductive inferences of whose validity we are aware can be at the object language or meta-language level; I am asking *epistemic* questions concerning knowledge of logical necessity that are the same in each case.

that what is expressed by the formula or the rules has logical necessity. Having established a

difference between these epistemic types, knowing that a computational proof or decision procedure conforms to rules and knowing that the rules or the formulas arrived at by following them express necessary truth,

; recognition of the conformity of steps to rules cannot be the source of our knowledge of the connection the rules, and therefore the result, with necessary truth

(b) recognition of the correctness of a step in a computational system is not the same as (a); and (c) [without (a) we could not recognize the correctness of step in a computational process. ?? Or,]

Recognition that a step in a computational process is correct requires an instance of (a); it does not explain (a), but (a) explains it. }

the use of matrix methods requires knowledge that after a finite number of steps, we have taken all the steps that are needed to arrive at that conclusion.

Instead of computational proofs, consider a decision procedure such as verification of validity by truth tables. To know that a decision procedure has shown a formula necessarily true or false,

Any computational method must fail to capture what we are capable of knowing by logical necessity, because the grasp of computational correctness depends on ("presupposes" in a causal rather than logical sense) knowledge of necessary truth that has not been acquired by computational methods, short of an infinite regress. This point is not the same as Godel's theorem that no computational system able to include arithmetic can

prove all the truths expressible in the system.

These philosophical issues are outside of logic itself and of no direct interest to the logician.

, until we kick the ladder away after constructing, with the use of standard inference principles, a logic that allows contradiction and deciding to use only our nonstandard logic from then on. To do so

[As both the bridge and computation examples show, knowledge of contingent facts to which the rules are applied, for example the facts that I have a card in that suit or that formula (??) is an instance of uniform substitution in formula (??)) is also required. The genius of]

The epistemic dependence of logical knowledge on excluding contradiction extends to procedures using rules specific to formal methods, rules such as the following:

(A) Uniform substitution in a valid formula produces a valid formula.

Knowledge that the result of substituting '$p \supset q$' for 'p' in '$p \vee \neg p$' is a valid formula, according to this rule, amounts to implicit knowledge that an argument like the following is valid:

I will argue that awareness of this inference's validity is awareness of the necessary validity of an inferential relation expressible by *modus ponens* or the law of detachment, among other ways, but whose necessity is recognizable independently of these ways, *or any* way (see section 7), of expressing it (and for that reason it differs from rules specific to mechanical methods). But that relation need not hold if we permit contradiction; so even in dialethic logical systems, mere awareness of the legitimacy of substitutions would presuppose awareness of the validity of inferences governed by principles of standard logic.

As this example shows, the point I am making is not arcane; the point is so obvious we too easily overlook it.

If we could fail to know that, it would not do us any epistemic good (that is, not achieve any epistemic goal) to know that a statement is proven or that it is unproven.

The contradictions about which ECQ is supposedly demonstrated are the simultaneous affirmation and denial (or positing and taking away, or whatever) by of something by means of a negation sign as ordinarily defined for two-valued contexts; so that is the kind of "negation" in play when we are talking about ECQ. Traditional principles of noncontradiction – whether for sentences or predicates, schema or schematic letters, in an object language or a metalanguage – just express the job we normally give to negation signs: Sentence or schema '$\neg p$' is a denial of '*p*', and vice versa; predicate 'F' cancels 'non-F', and vice versa.

Notes

****

just speak of a method's rules,.) and/or definitions (and/or axioms, or whatever the procedure uses as a way of establishing validity)

, even if (2) the rules and results of the method need have no other connection with necessary truth than being premises and conclusions of valid inferences

, ex contradictione quodlibet (ECQ)

on from having any logical necessity or any definition a procedure relies on from grounding any logical necessity

I will not only continue to also use nontechnical terms, but will use them in preference where possible, since the issue is the epistemic status of our technical methods, especially the epistemic conditions *presupposed* by the technical clarity and rigor of computational methods, the epistemic conditions that give us the ability to design and use, and so are causally *prior*

to, the clarity and rigor with which we endow the technical vocabulary of computational methods. methods endowed with such clarity and rigor.)

I will not only continue to also use nontechnical terms, but will use them in preference where possible; for the issue is the epistemic status of our technical methods, especially the epistemic conditions *presupposed* by the technical clarity and rigor of computational methods. Since I am explaining the epistemic conditions required for achieving the clarity and rigor associated with computational technical vocabulary, the explanation cannot presuppose that clarity and rigor except as the fact given to be explained, not as the means of explanation. The epistemic conditions that give us the ability to design and use computational methods are causally *prior to* the existence of their technical vocabularies, and so to clarity and rigor with which we endow those vocabularies. Still, any such explanation may have to introduce technical terms of another kind, philosophical terms, endowed with whatever clarity and rigor is appropriate for the purpose at hand. And as for technical terms of any kind, their introduction must rely, ultimately, on the use of nontechnical vocabulary.)

In knowing that "'If 'A then B', and 'A' are true, then 'B' is true" is true, the modality of the first two "true"s is irrelevant.

, since what C is is that for which we happen to be using "if . . . , then,"

The use of computational methods for verifying valid inference requires explicit recognition of a connection of the rules or definitions with necessary truth. So to use computational methods as tools for explicitly verifying inferential validity, in addition to (1) implicit logical knowledge of necessary truth and (2) explicit knowledge, which may be of the contingent, that a particular step happens to conform to a particular rule, we need explicit precomputational knowledge of necessary truth. That explicit recognition differs from the implicit recognition of the necessary truth on which the knowledge that a procedure conforms to the rules depends; as the above bridge example shows, implicit knowledge of modus ponens is required even if the rules have no connection with necessary

truth.

[We can also construct definitions and rules such that some wffs cannot not fail to preserve some designated value or to preserve truth on some intensional possible world semantics.]

an understanding of those rules is presupposed by an understanding of what these definitions, definitions to be applied by means of rules, are; for an understanding of how the terms by which these definitions are expressed are being used presupposes an understanding of what those rules are.
precomputationally known inferential validity is truth-functional validity and that not all

in order to know computational correctness. I am saying the opposite concerning some of the anomalies that
(transitive actions, actions resulting in static states of affairs, resulting in physical arrangements other than the actions)

If we did not rely on such recognition, we could not even get the paraconsistent logician's knowledge that '

1. For the way 'cause' is used in this essay, see Cahalan, *Causal Realism*.
2. Though they must be related somehow, they are distinct questions. As with any set of related philosophical questions, we must understand their distinction in order to understand the relations between them. Those relations may not be what they seem to be.
3. For an example of such pragmatism, see Susan Haack, *Deviant Logic, Fuzzy Logic: Beyond the Formalism* (Chicago: University of Chicago Press, 1996). Haack modifies her view in *Evidence and Inquiry* (Oxford: Blackwell, 1995).
4. "Logic, Philosophy of," the Routledge Encyclopedia of Philosophy.
5. Pragmatism, p. 10. For Putnam's discussion of Tarski, see Representation and Reality,

Chapter 4, and see "Does the Disquotational Theory of Truth Solve All Philosophical Problems" in Words and Life.

6. *Representation and Reality*, (Cambridge, MA: Massachusetts Institute of Technology Press, 19??).

7. The 'If contradiction, everything' example differs from others that illustrate this point, such as knowing that a chess move obeys the rules, in the crucial respect that here the rules *are*, or are chosen to reflect something that *is*, logically necessary.

8. 'Deleting,' 'canceling,' etc. do not primarily concern sentences or words but what is expressed by them.

9. This use of "relation" is non-technical with respect to its use, for a multi-place predicate, in the usual computational methods but not with respect to (not in relation to) the philosophical questions concerning relations. One logician who kindly reviewed an earlier draft of this work balked at calling negation a relation, since it is not a predicate and much less a multi-place predicate. This is a clear example of the power of methodological imperialism: "If it is not what my method calls a 'relation' or is not handled the way my method handles it calls 'relations', it is not a relation." It is also a good example of why we must examine the epistemic preconditions of computational methods before making philosophical claims based on them. "Not" means different-from and so must always accompany (or be associated with, be applied to, etc.) an additional intelligible value, Y (as in 'not Y'), which is therefore the *relatum* of the intelligible value, different-from, the "term" of the relation, other-than. That is one way the term 'relation' is precomputationally used, and is a use traditional philosophy has always justifiably had a need for. In that precomputational meaning, negation is a relation. And HOW ELSE WOULD YOU DESCRIBE NEGATION WHEN YOU HAVE A NEED TO DESCRIBE IT PRECOMPUTATIONALLY, as we do? As an additional intelligible value, Y is distinct from the intelligible value of 'not', although Y may be distinct just a reduplication of the original use of 'not' or the occurrence of another token of 'not'. The use of 'relation' that computational logic has derived from its

precomputational use, and that computational logic handles with unprecedented clarity and rigor, is narrower than its precomputational use. And there is absolutely nothing wrong with that. What is wrong is to try to monopolize the philosophical discussion of relation with that use, especially when the philosophical discussion concerns the epistemic conditions that the clarity and rigor of that use presuppose.

10. Philosophy of Logic, p.

11. Necessity and our knowledge of it do not depend on the possibility of translating on the basis of behavior. I make reference to behavioral evidence only in reply to the objection that my account implies an illegitimate theory of mental states. Independent arguments will show that awareness of necessity occurs and that this awareness requires no mental states other than those, if any, required for empirical knowledge. The objection applies, if at all, to an after-the-fact psychological account of knowing necessity, not to those arguments. So it is not circular to invoke necessary truths against the indeterminacy of translation. See John C. Cahalan, *Causal Realism: An Essay on Philosophical Method and the Foundations of Knowledge* (Lanham, Maryland: Rowman and Littlefield, 1985) pp. 42-43.

12. The 'If contradiction, everything' example differs from others that illustrate this point, such as knowing that a chess move obeys the rules, in the crucial respect that here the rules *are*, or are chosen to reflect something that *is*, logically necessary.

13. Some Renaissance logicians appear already to have seen the argument against 'If contradiction, everything,' but perhaps not that all argument fails if we allow contradiction. See E. J. Ashworth, *Language and Logic in the Post-Medieval Period* (Dordrecht: Reidel, 1974) 135.

14. Methods of Logic, 4th ed., p. 177.

15. Methods of Logic, p. 44.

16. A Philosophical Introduction to Set Theory, p. 84.

17. "Logical Consequence: Model-Theoretic Considerations," section d, *Internet Encyclopedia of Philosophy.*

18. The Ways of Paradox and Other Essays.

19. For the way 'cause' is used in this essay, see Cahalan, *Causal Realism*.

20. Fred Sommers, The Logic of Natural Language

21. For the way 'cause' is used in this essay, see Cahalan, *Causal Realism*.

22. And just as computational methods create anomalies for logic, using mathematics as a tool can create anomalies for physics.

23. Quine, *Methods of Logic*, 184.

24. "Aristotle," in *The Routledge Encyclopedia of Philosophy*.

25. *The Logic of Natural Language*, p. 143 ??.

26. *Representation and Reality* (Cambridge, MA: MIT Press, 1988) 117-118.

27. Peter Rutz, *Zweiwertige und mehrwertige Logik* (Münich: Ehrenwirth, 1973).

28. This use of "relation" is non-technical with respect to its use, for a multi-place predicate, in the usual computational methods but not with respect to (not in relation to) the philosophical questions concerning relations. One logician who kindly reviewed an earlier draft of this work balked at calling negation a relation, since it is not a predicate and much less a multi-place predicate. This is a clear example of the power of methodological imperialism: "If it is not what my method calls a relation or is not handled the way my method handles relations, it is not a relation." It is also a good example of why we must examine the epistemic preconditions of computational methods before making philosophical claims based on them. "Not" means different-from and so must be applied to a "term," that is, a *relatum,* some intelligible value, Y, and necessarily implies a reference to a *different relatum*, some X that is different from or considered to be different from Y, something that is non-Y. For example, When negation is applied to sentences (or statements or propositions) it means "not the case that" and implies a reference to an intelligible value other than the sentence, whatever it is that is the case. When negation is applied to the predicate "true," it implies a reference to some X different from that predicate, a sentence or statement or proposition, and describes X as something other than a sentence or statement

or proposition that achieves whatever goal it is by which we measure truth.

29. In brief, my defense of *tertium non datur* is this: That principle should be read, 'Where something has truth value, that value is either truth or falsity.' Any argument that some sentence, say, is neither true nor false must show that a condition needed for sentences to have truth value is missing. 'This sentence is . . .' lacks whatever it may be that lets sentences have truth value. So 'This sentence is true' and 'This sentence is false' are false, and 'This sentence is neither true nor false' is true. Aristotle argued, in effect, that sentences need conditions in the world to have truth value, and wrongly thought that sentences about the future lack one such condition, the future. The premises and conclusions of all critiques of *tertium non datur* lack *any* truth value unless they are reducible to answers to yes-or-no questions.

30. W. V. Quine, *Methods of Logic*, 4th ed. (Cambridge, MA: Harvard University Press, 1982) 213.