Classical Logic with Truth Value Gaps and without Supervaluations

Classical logic along with our practices of regimenting natural language with that logic is one of modern philosophy's great success stories. Almost all of us find almost all of the judgments of validity that regimenting induces to agree with our own. True, there is some grumbling about the relation between the indicative and material conditionals. True, most of us think there is more validity than that captured by our practices of regimentation –intensional and higher order validities, for example. But all of us think that baby logic gets a lot right; none of us (some odd ducks from the seventh continent excepted) anticipate ceasing to reason as we teach our baby logic students they ought to.

Alas, the classical semantics of classical logic, based as it is on the assumption of bivalence, is not quite as clearly one of our greater success stories. For natural language doesn't obviously have a bivalent semantics. Indeed, it seems rather obvious that it doesn't have one. Applications of vague predicates to borderline instances, sentences written on the blackboard in Room 317 that say that whatever is written on the blackboard in Room 317 isn't true –these sentences seem perfectly meaningful; typically they say something that isn't true but could have been. What they say, though, strikes many of us neither true nor false. If so, the semantics of our language isn't bivalent.

If we accept the judgments about validity that our practices of regimentation into classical logic underwrite but don't think classical semantics reflects the semantics of our language, we're in a bit of a bind. For we then don't have an explanation of why regimenting our (non-bivalent) language in the (bivalent) language of logic gives us an accurate story about what is and what is not a valid argument. Supervaluationism –the best extant attempt to reconcile logic's classifications of validity with the existence of truth value gaps –is less than a complete success.¹ For the supervaluationist says things that run at variance with our understanding of what our words mean –for instance, that disjunctions can be true though none of their disjuncts are; that (determinately) there is a number n between 2 and 10 such that n grains of sand make a heap, but no number n between 2 and 10 is such that (determinately) n grains of sand make a heap.

My goal in this paper is to make a start on getting us out of this bind. I provide a (sentential) logic and semantics that: (1) is in line with the judgments about validity that make logic one of philosophy's success stories; (2) embraces the idea that sentences can be meaningful though truth valueless, and; (3) provides a simpler and I think much more satisfying account of the meaning and compositional semantics of sentence compounding devices than is provided by supervaluationism. I have sketched the ideas behind this system elsewhere.\(^2\) What is new here is a proof procedure (a version of the tree method) and completeness proof.

The proposal is motivated by the idea that sentence compounding devices have two roles in natural language. Consider 'not'. Sometimes it contributes to sense. When it does, a use of

(1) It's not the case that S

expresses the negation of whatever claim S expresses, with (1)'s utterance asserting the negated claim. But sometimes use of the idioms of negation contributes to force. In this case, an utterance of (1) will amount to the denial of the claim expressed by S. Denial is a distinctive speech act, not to be defined in terms of or otherwise reduced to assertion. It differs from assertion of the negation in terms of appropriateness conditions: asserting p's negation is apt if and only if p is false; denying p is apt iff p is either false or truth valueless.\(^3\)

This dual function is present in all of the connectives. There are, for instance, two ways to understand a use of

(2) Either Bob is bald or it's not the case that Bob is bald.

On one, it is the assertion of a disjunction regimented in baby logic with \(p \lor \neg p\); this is apt just in case \(p\) is truth valued. But there is a use of 'or' on which (put a bit roughly) \(A \lor B\) is appropriate just in case use of \(A\) is appropriate or use of \(B\) is. Suppose that 'or' is used with this meaning in an utterance of (2) and 'not', in the use, is used with the meaning it has when it is used in denial. Since a use of 'Bob is bald' is apt just in case it's true, and a

\(^2\) Richard 2008, 2010. I don't here discuss the project of giving an account of validity for a non-bivalent language that can ascribe semantic properties to its sentences. I say something about this in Appendix A to Richard 2008 and in Richard, forthcoming.

\(^3\) The biconditional here is not the material conditional but one defined in terms of the "force connectives" introduced below.

(C) Mark Richard 2011. Do not cite or reproduce without permission.
use of 'Bob is not bald', with 'not' used with its "force meaning", is apt just in case 'Bob is bald' is false or truth valueless, using the sentence in this way is guaranteed to be apt.

Obviously the trick here is to find meanings for the "force signaling" uses of the connectives that will (for example) allow a stand alone use of 'Bob is not bald' to be a vehicle for denial while also interacting with the force signaling meaning of 'or' in such a way as to deliver the just mentioned appropriateness conditions for (2). The way to achieve this is simple. A semantics for the language needs to do is assign appropriateness conditions (which for semantic purposes will either be sets of models or things that determine such sets) to sentences and functions from appropriateness conditions to appropriateness conditions to sentence connectives that are functioning in 'force signaling' mode. Then, as we are about to see, everything comes out just fine.

The propositional language we will discuss –call it ST– lexicalizes the distinction between using a connective to contribute to sense and using it to signal force. In addition to atomic sentences and devices of punctuation, it has a set of truth functional connectives; we take binary 'v' and unary '¬' as primitive and assume the others introduced by standard definitions. It also has a set of "force connectives"; we take unary 'not' and binary 'or' as primitive and assume other force connectives introduced by definitions corresponding to the standard ones for the truth functions. The set of unforced sentences of ST, the set US, is the result of closing the set of atomic sentences under truth functional negation and disjunction. The set S of sentences of ST is the result of closing US under "force negation" and "force disjunction". Sentences of the language that are not unforced are forced sentences. Note that this definition has the upshot that truth functional connectives cannot outscope force connectives: While each of

\[(p \lor \neg p)\]
\[(p \lor \neg p)\]
\[(p \lor \neg p)\]

are sentences of the languages,

\[(p \lor \neg p)\]

is not a sentence of the language. In what follows we use lower case \(p, q,\) and \(r\) as variables ranging over atoms; \(A, B,\) and \(C\) as variables ranging over unforced sentences; \(R, S, T\) as variables ranging over arbitrary sentences.
A *model for ST* is any member \( v \) of \( V \), the set of all partial assignments of the truth values \( t \) and \( f \) to atoms. Though lacking a truth value is not having some third truth value, we occasionally write such things as \( v(p) = \# \) to indicate that \( p \) is truth valueless in \( v \). Members of \( V \) determine truth values for molecular unforced sentences of ST in accord with Kleene's strong truth tables. Unforced sentences express propositions and are vehicles for asserting what they express. Models of the language, by making (partial) assignments of truth values to such sentences, determine appropriateness conditions for all the unforced sentences of the language. Indeed, given the (intended) meanings of the force connectives, such models determine appropriateness conditions for all the sentences of the language. The primary task in giving a semantics for ST is defining \( S \) is appropriate in the model \( v \) inductively. This we do by assigning to each sentence of the language a semantic value that characterizes the models in which it is appropriate.

An unforced sentence \( A \) is appropriate in those \( v \)'s that make \( A \) true; on a first pass, we can represent its appropriateness conditions with the pair \( \langle \{ A \}, \phi \rangle \), the idea being that when \( s \) and \( s' \) are sets of unforced sentences, \( \langle s, s' \rangle \) represents the appropriateness conditions of \( S \) provided that use of \( S \) is apt in \( v \) iff all the members of \( s \) are true in \( v \) and none of the members of \( s' \) are. Call pairs \( \langle s, s' \rangle \) of sets of sentences conditions. We use sets of conditions (SOCs) as the semantic values of sentences; a condition of the form \( \{ s, s' \} \) is said to be *basic*.

If \( A \) is an unforced sentence \( A \), the semantic value of \( A \) is \( \{ \{ A \}, \phi \} \). In order to assign semantic values to forced sentences, we introduce three operations on SOCs. \( c, c' \), and subscripted \( c \)'s range over SOCs:

\[
\text{Disj} (c, c') = c \cup c'.
\]

\[
\text{Disj} (c_1, \ldots, c_i, c_{i+1}) = \text{Disj} (\text{Disj} (c_1, \ldots, c_i), c_{i+1}).
\]

\[
\text{Conj} (c, c') = \{ \langle \alpha, \beta \rangle \mid \exists o \in c \exists o' \in c' \exists a, a', g, g' (o = <g, d>. o' = <g', d'>. \alpha = g \cup g'. \beta = d \cup d')).
\]

\[
\text{Conj} (c_1, \ldots, c_i, c_{i+1}) = \text{Conj} (\text{Conj} (c_1, \ldots, c_i), c_{i+1})^4
\]

---

\(^4\) This corrects a mischaracterization of Conj in Richard 2008, Chapter 2, n. 29, which unfortunately is carried over to Richard 2010.

(C) Mark Richard 2011. Do not cite or reproduce without permission.
Conv ( {<α,β>} ) = { <α',β' > | ∃ p ∈ α (α' = φ . β' = {p}) v ∃ p ∈ β (β' = φ . α' = {p}) } 

Conv ( {c₁,...,cᵢ} ) = Conj ( Conv( c₁ ), … Conv( cᵢ ) )

Let v be in V. We say that a basic condition {<α,β>} is appropriate in v provided all of α is true in v and none of β is; where c = {c₁,...,cᵢ}, c is appropriate in v provided at least one member of it is. We next prove that Disj, Conj, and Conv have properties appropriate to be represented by 'or', 'and', and 'not'.

(1) Disj( c, c' ) is appropriate in v iff (c is appropriate in v or c' is appropriate in v).
Proof: Follows directly from the definition of c is appropriate in v. (2) Conj (c, c') is appropriate in v iff (c is appropriate in v and c' is appropriate in v). Proof: Suppose Conj (c,c') is appropriate in v. Then for some <α,β> in Conj(c,c'), its unit set is appropriate in v, so that v makes all of α true, none of β true. Any such <α,β> is obtained by merging the first member of some <α',β'> in c with the first member of some <α'',β''> in c' to obtain α, and likewise merging β' with β'' to obtain β. So all of α' and α'' are true in v, none of β' or β'' is, and thus the unit sets of the last two mentioned pairs are appropriate in v, and thus c and c' are. The converse is proved in analogous fashion. (3) Conv (c) is appropriate in v iff c is not appropriate in v. Proof: Suppose first that c = {<α,β>}. c is appropriate in v iff all of α is true there and none of β is, and thus is not appropriate in v iff some member of α is not true in v or some member of β is true in v. But this is the condition under which Conv (c) is appropriate in v. Now let c = {c₁,...,cᵢ}. c is not appropriate in v iff no member of it is appropriate therein. Since each member of c is of the form <α,β>, this means (given the first part of the proof) that c is not appropriate in v iff for each c' in c, Conv(c') is appropriate in v. And thus, by (2) c is not appropriate in v iff Conj(Conv(c₁), … Conv( cᵢ )) is appropriate in v.

We may now complete the semantics for ST. The semantic value of an unforced wff A is { <\{A\} , φ > }. The semantic value of 'not' is the function Conv; the semantic value of 'or' is the function Disj; the semantic value of a forced sentence is the result of applying the semantic value of its main operator to those of its operands. A sentence is appropriate in v iff its semantic value is appropriate in v. A sentence is valid iff
appropriate in all models; an argument is valid provided that its conclusion is appropriate in all models in which all of its premises are.

Baby logic tells us that the form *Either A or B; it's not the case that A; thus B* can be used to give a valid argument. It tells us that the form *Either A or not A* can be understood as expressing a logical truth. Natural practices of regimenting English with ST tells us the same. Suppose that an argument (or sentence) of English can be regimented as valid in SL. It is not hard to verify (as we will below) that there is then a valid ST regimentation of the argument (sentence): simply change all the SL connectives in the SL representation to the corresponding force connectives. For example, our practices of regimenting in SL render

Either Fred is not gone or he left his wallet. Fred didn't leave his wallet. So Fred is not gone.

as

\[(3) \quad \neg F \lor W; \neg W; \text{so, } \neg F;\]

changing the connectives to force connectives yields

\[(4) \quad \neg F \text{ or } W; \neg W; \text{so, not } F,\]

which is ST-valid. (This is immediate from (1) and (3) above.) Likewise for any instance of excluded middle, which, following the above procedure, is regimented in ST as \( P \text{ or not } P \).

Very often, but not invariably, regimentations that are SL-valid are ST-valid as well. (3), for example, is itself ST-valid: appropriateness for unforced sentences is truth; the truth of both (3)'s premises guarantees that of its conclusion. This is usually true of *arguments*, though there are exceptions –most notably ones where the truth of the premises has no bearing on whether the atoms in the conclusion take on truth values, as in

\[(C) \quad \text{Mark Richard 2011. Do not cite or reproduce without permission. Version 3. Posted 30 July 2011.}\]
(5) \(-F \vee W; -W; \text{so, } G \vee \neg G.\)

More generally, tautologies are not ST-valid: while \(P \text{ or not } P\) is ST-valid (as \(P\) is either appropriate or not), \(P \vee \neg P\) is not ST-valid, as \(P\) is not required to have a truth value.

However: let \(p_1, p_2, \ldots, p_i\) be the atoms in an SL tautology \(T\); the sentence

(6) If \((p_1 \vee \neg p_1) \& (p_2 \vee \neg p_2) \& \ldots \& (p_i \vee \neg p_i)\), then \(T\)

is ST-valid. (This since the antecedent is apt only if every constituent of \(T\) has a truth value.). Adding an analogous premises to an ST-valid argument always yields an SL-valid argument.

It is not difficult to extend the tree method to a system that is sound and complete with respect to the semantics of ST. The method's rules are all of the form

(7) \(S / T_1; T_2; \ldots; T_i\)

where \(S\) and the \(T_i\)'s are sets of formula schemata (supplemented with certain extra symbols noted below). To apply a rule of the form of (7), one: locates unchecked instances of all the schemata in \(S\) (all on a single open branch); extends each open branch \(b\) on which those instances occur with \(i\) paths, path \(p_j\) (\(j\) between 1 and \(i\)) containing (appropriate) instances of the schemata in \(T_j\); checks the occurrences of instances of members \(S\). The rules employ both '' (to indicate closure of a path) as well as strings of the form \(\lor p, \) \(p\) any atomic sentence. These last strings are understood as indicating that \(p\) is truth valueless.

The rules are as follows.

R1. \(A \lor B / A; B\)
R2. \(- (A \lor B) / -A, -B\)
R3. \(-- A / A\)
R4. \(A, -A / \Box\)

(C) Mark Richard 2011. Do not cite or reproduce without permission.
A branch is open unless and until it contains a token of '\textcolor{red}{X}'\textquotesingle, at which point it is closed. A literal is any atomic, negated atomic, or atomic or negated atomic prefixed by the down arrow. A branch is exhausted when it is open and there is no unchecked formula on it save for literals. To apply the method to a set of sentences, one lists them in any order and then applies rules, in any order, to the list and continues to apply rules to what results from applying rules until every branch is closed or exhausted. The method classifies an argument A as valid if at some point every branch in a tree that begins with a list of the argument's premises and the (force) denial of its conclusion closes; it classifies the argument as invalid provided such a tree eventually terminates with at least one exhausted branch. Valid sentences are ones that are the conclusions of arguments with null premises.

A proof that this method provides a decision procedure for ST requires only a routine extension of standard proofs that the tree method for SL provides a decision procedure for SL. Let B be a tree. Say that the significance of a formula on B is zero if it is a literal or is checked; otherwise, its significance is equal to the number occurrences of forced and unforced connectives in the formula. Say that applying a rule to B reduces the tree's significance provided that for some n, applying the rule to B reduces the number of formulas on B of significance n and doesn't increase the number of formulas on B whose
significance is greater than n. One can verify by inspection that each of the above rules reduces the significance of any tree to which it is applied. Since each rule adds only finitely many new formulas to a tree to which it is applied, it follows that any application of the tree method will eventually terminate.

Given the ST’s semantics and (1) through (3) above, we have (4) (a) an unforced wff is appropriate in v iff it is true in v; (b) S or T is appropriate in v iff (S is appropriate in v or T is); (c) not S is appropriate in v iff S is not appropriate in v. Let us say that '∃x' is appropriate in no model, and that ∨p is appropriate in v just when p is truth valueless in v. One can establish by inspection of the rules that (5) for each instance of a rule S / T₁, …, Tᵢ in the above list and for any model v, the instances of members of S are appropriate in v iff the instances of at least one Tⱼ are appropriate in v. (For example, for R2: A sentence of the form –(AvB) is appropriate in v iff true iff (AvB) is false iff each of A, B is false iff each of –A, –B is true iff each is appropriate.) This establishes that the method is sound with respect to ST’s semantics.

As for completeness: Suppose a finished tree with a exhausted branch b. Since b is exhausted there can't be more than one of p, -p, ∨p on b, p any atom. Thus there will be a model v in which an atom is true iff it occurs on b, false iff its negation occurs on b, and leaves it truth valueless otherwise. M must make every formula on b appropriate, as can be seen by an induction on number of connectives in a formula that appeals to (1) through (5).

Consider now an SL argument A. Let B be a completed tree in a standard version of the tree method for SL for A (i.e., B begins with A's premises and the negation of its conclusion). The only rules employed in such a tree (in a standard version of the tree method for SL) will be rules R1 through R4 above. Consider the ST argument obtained from A by replacing each occurrence of an unforced connective in A with its forced equivalent. Clearly the result of replacing each unforced connective in B with its forced equivalent will be a tree of the system for which we just proved completeness. In this tree only rules R5, R8, R9, and R10 are used. (This is so because of the, as we might put it, structural isomorphism between the original tree and the new one and between the two sets of rules.) Suppose that all the paths on the original tree B closed. Then so do all the paths on the transformed tree. Suppose that some path on B, b, remained open. b's
transformation may not be exhausted in the system for ST, but only because b may contain occurrences of formulas of the form \( \neg p \). Extend b by applying rule 12 as many times as possible to such formulas. Since b was not closed, it did not contain occurrence of the atom p. Thus, eventually b will be exhausted, which means, given completeness, that the transformation of A is SL-valid, as claimed above.

These results, it seems to me, show that we can accept that meaningful sentences may lack truth values while continuing to reason in natural language in the way in which we teach our students to reason in logic classes. We may do this without appealing to the (in my opinion) somewhat strained account of the truth conditions of disjunctions and existential statements required by supervaluation. We do, of course, have to bear in mind that in some cases the sentences with which we reason may be truth valueless, and thus sometimes the conclusions we draw from our sentences cannot be understood as conclusions to be asserted. But this, it seems to me, is something anyone who takes the idea that meaningful sentences may be truth valueless should be willing to accept. For such sentences, being meaningful, will say something. But what they say will not be true, nor will it be false. In drawing this last conclusion, though, I am certainly not asserting that they are not true or false.\(^5\)

Mark Richard

Philosophy, Harvard University

richard4@fas.harvard.edu

\(^5\) In this discussion I have not addressed issues raised by reasoning in languages in which we can assess sentences (or claims) for such properties as truth and appropriateness. The story we need to tell about the logic of such languages will of course be more complicated than the story told here. But the upshot, as far as whether we need to invoke supervaluation in giving a semantics, I think is very much the same. I have discussed this in a preliminary way in Richard 2008's Appendix A.


