Some Strong Conditionals for Sentential Logics

Jason Zarri

[11/18/2012]

[Note: This is a rough draft intended to be circulated to get feedback. Links are welcome, *but please do not cite*.

Comments can be sent to: jlzarri@scholardarity.com]

1. *Introduction*

In this article I define a strong conditional for classical sentential logic, and then extend it to three non-classical sentential logics. It is stronger than the material conditional and is not subject to the standard paradoxes of material implication, nor is it subject to some of the standard paradoxes of C. I. Lewis's strict implication. My conditional has some counterintuitive consequences of its own, but I think its pros outweigh its cons. In any case, one can always augment one's language with more than one conditional, and it may be that no single conditional will satisfy all of our intuitions about how a conditional should behave. Finally, I suspect the strong conditional will be of more use for logic rather than the philosophy of language, and I will make no claim that the strong conditional is a good model for any particular use of the indicative conditional in English or other natural languages. Still, it would certainly be a nice bonus if some modified version of the strong conditional could serve as one.

I begin by exploring some of the disadvantages of the material conditional, the strict conditional, and some relevant conditionals. I proceed to define a strong conditional for classical sentential logic. I go on to adapt this account to Graham Priest's Logic of Paradox, to S. C. Kleene's logic K_3, and then to J. Łukasiewicz's logic Ł, a standard version of fuzzy logic.

2. *Extant Conditionals: Material, Strict, and Relevant*

In classical sentential logic, all conditionals are treated as *material conditionals*, also called *material implications*. The truth table for the material conditional, which is expressed by the symbol '⊃', is:

p	q	p ⊃ q
1	1	1
1	0	0
0	1	1
0	0	1

Though this account of conditionals has some puzzling consequences, the truth table is not itself hard to interpret.[1] Since the properties of the material conditional are something of a mixed bag, I will divide its assessment into three subsections.

[1] For those who aren't familiar with truth tables or the material conditional, to put things as simply as I can: p ⊃ q is read as "If p then q", or "p implies q," where one can substitute arbitrary sentences for 'p' and 'q'. The sentence substituted for 'p' is called the *antecedent* of the conditional, and the one substituted for 'q' is called the conditional's *consequent*. One reads the rows of the truth table from left to right, starting with the first row and working one's way down until one reaches the last row. Each row represents the "truth values" that sentences can take, with '1' representing truth and '0' representing falsity on a given evaluation. In the first row, a '1' occurs in the column under p, which means that p is true on the evaluation being considered, and the '1' in the column under q in that row means that q is also true on that evaluation. The '1' in the column under p ⊃ q on that same row means that it too is true on that evaluation. Thus, what this row tells us is that if p and q are both true then p ⊃ q must be true too. Once one has understood how to evaluate this row, one can apply the same method to evaluate the other rows. In sum, what the truth table says is that p ⊃ q is *false* when p is *true* and q is *false*; and is true in all other cases.

1. **The Good**: The material conditional satisfies the rule of inference called *Modus Ponens*: If we let '$\rightarrow$' represent an arbitrary type of conditional, Modus Ponens tells us that from p and p $\rightarrow$ q one can infer q. One can tell by inspecting the truth table for the material conditional that if p and p ⊃ q are both true then so is q. In the second and fourth rows, the only cases where q is false, either p is false or p ⊃ q is false. Since Modus Ponens is valid, inferring q from p and p ⊃ q will never take one from truth to falsity.

2. **The Bad**: Because the material conditional is a truth functional operator—one whose truth value is determined solely by the truth values of its component sentences—one cannot tell whether it is true without knowing the truth values of p and/or q. If you know that p is false then you know that p ⊃ q is true, whatever the truth value of q may be. And if you know that q is true then you know that p ⊃ q is true, whatever the truth value of p may be. If all you know is that p is true, or if all you know is that q is false, you cannot determine the truth value of p ⊃ q. Among other things, this means you cannot use Modus Ponens without knowing whether q is true or false. You can only apply Modus Ponens if you know that p and p ⊃ q are both true, and if you know that p is true you can't know whether p ⊃ q is true without knowing whether q is true. If you already know that q is true, you don't need to infer its truth from anything**.** And if you already know that q is false, you know that you *cannot* infer its truth from anything—if you had a valid argument with q as its conclusion, you would know that that argument must be unsound. So while the material conditional satisfies Modus Ponens, the fact that is does is not enough to make reasoning with it useful in practice.

3. **The Ugly**: Ordinarily, when one asserts a conditional one implies that there is some sort of connection between its antecedent and its consequent. As an example, take the sentence "If that ball is red, then that ball has a color". (We shall suppose this sentence is uttered in a context where it is clear which ball is meant.) I'd wager that most people would say that the truth of the antecedent guarantees the truth of the consequent; if the ball under discussion is red, it just *has* to be true that it has a color. However, a material conditional can be true when its antecedent and consequent have nothing to do with each other. For instance, from the first row of the truth table it follows that every true sentence materially implies every other true sentence. Thus, "Electrons are negatively charged" materially implies "Neil Armstrong was the first man to walk on the Moon". Yet no matter how true it may be, "If electrons are negatively charged then Neil Armstrong was the first man to walk on the Moon" would be an odd thing to say, to say the least. Also, it follows from the third and fourth rows of the truth table that a false sentence materially implies every true sentence as well as every false sentence. If we interpret the English 'if...then…' construction as a material conditional, the sentences "If bananas are vegetables then the sky is

> blue" and "If George W. Bush is articulate then the Pope isn't Catholic" are both true because their antecedents are both false. (In the author's opinion the antecedent of the last material conditional is pretty clearly false. But however that may be, I'll ask the reader to be kind enough to suppose that it is false for the sake of the ensuing discussion.) The fact that the consequent of the first sentence is true and that the consequent of the second sentence is false makes no difference. The strangeness of these and similar sentences gives intuitive support to the view that '⊃' and the English 'if…then…' construction do not quite mean the same.

These and other odd results are often referred to as *the paradoxes of material implication*. It is these "paradoxes" that drove C.I. Lewis to define a different conditional, called the *strict conditional* or *strict implication*, for which he used the symbol '⥽' , which would not suffer from these and similar problems. In effect, Lewis defined strict implications as material implications that are necessarily true. Hence, to understand his account fully we will have to say a word about necessity.

One definition of necessary truth would be that a necessary truth is a statement which could not possibly be false. (By saying that so-and-so is possible, I refer how things *could have* turned out, *whether they actually turned out that way or not*, *not* to my or anyone else's knowledge—or lack thereof—of how things *actually did* turn out. The notion of possibility at issue here is thus metaphysical rather than epistemological; it deals with the world as opposed to our knowledge of it.) Necessary truths hold "come what may," as the philosopher W. V. O. Quine put it in his article "Two Dogmas of Empiricism" [ref]. Take an elementary statement of arithmetic, say "7 + 5 = 12". This statement doesn't just *happen* to be true. Irrespective of how things actually did turn out, they could have turned out differently in many different ways; dinosaurs might never have become extinct, Immanuel Kant could have spent his entire career in a dogmatic slumber, and George W. Bush could have been ever so slightly more eloquent but no matter how different the world could have been, "7 + 5 = 12" would still have been true. Let us use the diamond symbol, '◊', to mean 'it is possibly the case that', and the box symbol, '□', to mean 'it is necessarily the case that. Writhing "◊ 7 + 5 = 12" then has the effect of saying that "7 + 5 = 12" is possibly true, and writing "□ 7 + 5 = 12" has the effect of saying that "7 + 5 = 12" is necessarily true. We can give the meaning of □ p, in general as being equivalent to ¬ ◊ ¬ p, where '¬' is the negation sign. (This is just a symbolic translation of the above definition of necessary truth.)

Now that we are a little clearer on what $\Box p$ means, we can understand Lewis's account of strict implication. He defines a strict conditional of the form $p \prec q$ as being equivalent to $\Box(p \supset q)$ [ref]. On this definition, $p \prec q$ holds only if $p \supset q$ holds no matter what; in other worlds, it holds only if there is no possible situation in which $p \supset q$ is false. In still other words, in order for $p \prec q$ to be true, there cannot be any possible situation in which p is true while q is false.

[…]

3. *CSL and its Meta-Language*

Now it is time to make things a bit more formal. I shall begin my account of strong implication by giving a standard language for classical sentential logic (CSL). For atomic sentences, we will use capital letters from the entire alphabet, with numerical subscripts appended to them if necessary. As for connectives, we will use the symbols $\neg$, $\wedge$, $\vee$, $\supset$, and $\equiv$, respectively, for negation, conjunction, disjunction, material implication, and material equivalence. For brackets one can use parentheses, (,); square brackets, [,]; or curly brackets, {, }. It makes no difference which brackets are used where. I will use 1 and 0 as truth values, 1 for *true* and 0 for *false*. Finally, the usual recursive clauses for constructing *well-formed formulas*—*wffs*, pronounced "woofs," for short—from atomic sentences will be adopted. All atomic sentences of CSL are wffs, and where p and q are arbitrary sentences of CSL:

1. If p is a wff, so is $\neg(p)$.[2]

2. If p and q are wffs, so is $(p \wedge q)$.

3. If p and q are wffs, so is $(p \vee q)$.

4. If p and q are wffs, so is $(p \supset q)$.

5. If p and q are wffs, so is $(p \equiv q)$.

6. Nothing else is a wff of CSL.

Our meta-language[3] for CSL, MCSL, will include the lowercase letters p, q, r, etc, all the way to the end of the alphabet, as meta-variables[4]; these have the same function as the letters we

[2] However, if p is an atomic sentence, one doesn't have to put brackets around it.

just used in giving the recursive clauses for wffs of CSL. They too may have subscripts appended to them if necessary. Any sentence of CSL also belongs to MCSL, though the reverse is not true. Additionally, MCSL includes quote-names for sentences of the object-language. Thus the meta-linguistic term 'A' names the atomic sentence A of the object-language; the term 'B ∧ C' names the compound sentence B ∧ C of the object language, and so on. The meta-language will also have corner quotes, ⌜ and ⌝, which selectively quote logical connectives and brackets but not the meta-variables p, q, r, (etc.), so that it still makes sense to substitute sentences of CSL for them.[5] Finally, MCSL will contain a novel item, the meta-linguistic designator ⌜(q|p)⌝, a "conditional designator" which designates the truth value that q takes given that p is true, i.e., given that p has the value 1. It is to be read as ⌜the value of q given p⌝ or ⌜the value of q conditional on p⌝. The stroke, |, is not a connective; it merely serves to separate the letter q from the letter p.

In order to be permissibly substituted into the designator ⌜(q|p)⌝, q and p must either be wffs of CSL, or else sentences of MCSL itself that have been formed from such a designator, or sentences which have been formed from a designator which was formed from a sentence which was formed from such a designator… . Ultimately, any sentence that can appear inside the conditional designator will have been derived from a sentence of CSL, and no meta-variable will appear inside a conditional designator when it is actually being used. (We will see how novel sentences can be formed from designators in MCSL in the next section.) If all these conditions are met, we will say that ⌜(q|p)⌝ is a *well-formed designator*—a *wfd,* pronounced "*woofed*," for short—and if not, ⌜(q|p)⌝ is a *non-well-formed designator*. The conditional designator works like this: If p never takes the value 1, then ⌜(q|p)⌝ designates nothing—for q cannot take a value *given that p is true* if p can never be true—and is said to be *empty*. It is also empty if the value of

[3] A *meta-language* is a language that one uses to talk about another language, which is called the *object-language*. These notions are relational: The same language can be a meta-language of one language and an object language of another. In logic, formalized meta-languages are employed to talk about various formal logical systems in order to avoid the paradoxes and/or viciously circular definitions that would ensue if one tried to use a formal language to make statements about itself.

[4] A meta-variable is a variable that belongs to the meta-language of a given language. Here we use the letters p, q, r, and so on as meta-variables in MCSL, for which any sentence of CSL can be substituted.

[5] If we used ordinary quotation marks instead, the meta-variables would be included in the quoted material. For example, the sentence A ∧ B is an *instance* of ⌜p ∧ q⌝; that is, it is a sentence of CSL that is obtained from ⌜p ∧ q⌝ by substituting the atomic letters A and B, respectively, for p and q—but the string of symbols 'p ∧ q'—notice the lack of corner-quotes!—belongs, *not* to CSL, but rather to its meta-language MCSL, and A ∧ B is not an instance of 'p ∧ q', for that is a string of symbols: 'p' and 'q' are placeholders for sentences, and they themselves assert nothing. .

q varies when the value of p is 1, for in that case q doesn't take a *unique* value given that p is true. If q always takes the value 1 when p takes the value 1, then ⌜(q|p)⌝ designates 1, and in our meta-language we can say that ⌜(q|p)⌝ = 1, which is another way of saying that ⌜(q|p)⌝ designates 1. Similarly, if q always takes the value 0 when p takes the value 1, then ⌜(q|p)⌝ designates 0, and in our meta-language we can say that ⌜(q|p)⌝ = 0.

We may include ⌜(q|p)⌝ in a truth table, as long as we remember that the values entered under it are the values it *designates*, not the values it *has,* i.e. *none*. (If it is not clear why I have characterized ⌜(q|p)⌝ as a designator, and not as an operator which forms sentences that have truth values of their own, consider what value(s) (A | A ∧ ¬A) would have to have on such an approach. In classical logic, the assignment of multiple truth values to the same sentence is a Very Bad Thing. In section 5 we'll explore a logic in which it isn't.) In addition, I will enter the symbol '%' under ⌜(q|p)⌝ to mark the cases where it is empty. The following truth tables will illustrate some of the conditional designator's logical properties:

A	¬A	A ∧ ¬A	(A | A ∧ ¬A)	(A ∧ ¬A | A)	(A ∧ ¬A | A ∧ ¬A)
1	0	0	%	0	%
0	1	0	%	0	%

A	¬A	A ∨ ¬A	(A | A ∨ ¬A)	(A ∨ ¬A | A)	(A ∨ ¬A | A ∨ ¬A)
1	0	1	%	1	1
0	1	1	%	1	1

A	B	A ∧ B	(A | A)	(B | A)	(B | A ∧ B)
1	1	1	1	%	1
1	0	0	1	%	1
0	1	0	1	%	1
0	0	0	1	%	1

4. *The Strong Conditional Defined*

With our meta-linguistic conditional designator ready to hand, we can now define what I call the *strong conditional*, or *strong implication*, for which I will use the symbol '→'. Its definition is (where 'v( )' is the valuation function, which gives the semantic value of an expression):

If $\ulcorner(q|p)\urcorner = 1$, then $v(p \rightarrow q) = 1$

If $\ulcorner(q|p)\urcorner = 0$, then $v(p \rightarrow q) = 0$

If $\ulcorner(q|p)\urcorner$ is empty, then $v(p \rightarrow q) = 0$

'→' is introduced in MCSL. A sentence of the form $\ulcorner p \rightarrow q\urcorner$ is a wff of MCSL if and only if the corresponding designator $\ulcorner(q|p)\urcorner$ is a wfd of MCSL.

We can then extend CSL by adding in the strong conditional—as well as the corresponding *strong equivalence* connective, '↔', defined as $\ulcorner(p \rightarrow q) \wedge (q \rightarrow p)\urcorner$— obtaining the language I call CSL+. Its recursive clauses for constructing wffs from atomic sentences are the same as those for CSL, with two additions to govern our two new connectives:

$\ulcorner p \rightarrow q\urcorner$ is a wff of CSL+ if and only if $\ulcorner p \rightarrow q\urcorner$ is a wff of MCSL.

$\ulcorner p \leftrightarrow q\urcorner$ is a wff of CSL+ if and only if $\ulcorner p \rightarrow q\urcorner$ and $\ulcorner q \rightarrow p\urcorner$ are wffs of MCSL.

We can obtain the following truth tables for '→,' which are counterparts of the ones for $\ulcorner(q|p)\urcorner$ given above:

A	¬A	A ∧ ¬A	(A ∧ ¬A) → A	A → (A ∧ ¬A)	(A ∧ ¬A) → (A ∧ ¬A)
1	0	0	0	0	0
0	1	0	0	0	0

A	¬A	A ∨ ¬A	(A ∨ ¬A) → A	A → (A ∨ ¬A)	(A ∨ ¬A) → (A ∨ ¬A)
1	0	1	0	1	1
0	1	1	0	1	1

A	B	A ∧ B	A → A	A → B	(A ∧ B) → B
1	1	1	1	0	1
1	0	0	1	0	1
0	1	0	1	0	1
0	0	0	1	0	1

Furthermore, we have:

A	¬A	B	A ∧ ¬A	(A ∧ ¬A) → B	B → (A ∧ ¬A)
1	0	1	0	0	0
1	0	0	0	0	0
0	1	1	0	0	0
0	1	0	0	0	0

This last table shows that *explosion,* understood as the principle that contradictions imply everything, does not hold for '→'. However, because CSL+ is (quasi-) classical, contradictions still *entail* everything. This has the consequence that ⌜p → q⌝ can be false even if p entails q. Strong implication is thus, for lack of a better term, more *discriminating* than entailment.

It is a noteworthy fact that in CSL+, the "Law of Identity" is not valid in general. As one can tell by inspecting the truth tables I have given, nothing strongly implies a contradiction and contradictions strongly imply nothing, and so *a fortiori* they do not strongly imply themselves. We can call these facts *the paradoxes of strong implication*. In spite of them, every non-contradictory sentence strongly implies itself, so the "Restricted Law of Identity"—if p is not a contradiction, then ⌜p → p⌝—*is* valid.

However, most common "paradoxes of material implication" do not have counterparts that hold for strong implication. For one thing, ¬A fails to entail A → B; so in general, a false sentence does not strongly imply every sentence. For another, B fails to entail A → B; so in general not every sentence strongly implies a true sentence. This means that, unlike the material conditional, the joint truth of A and B does not entail A → B; in fact, no distinct atomic sentences strongly imply each other. However, we do have the result that every non-contradictory sentence implies a tautology. *Modus Ponens* also holds for '→', as does *Modus Tollens*.[6] [Show this] Since the truth values of sentences of the forms ⌜p → q⌝ and ⌜p ↔ q⌝ are sensitive both to the identity of p and q and their truth-functional structure, I will call them *structure-sensitive* connectives.

The examples of formulas that we have given up to this point have all been fairly simple. We will now turn to sentences involving nested strong conditionals or strong equivalences, which are slightly more complex. To form such sentences we can start with this truth table of the meta-language MCSL:

A	B	C	A ∧ B	(A ∧ B) ∧ C	(A ∧ B | (A ∧ B) ∧ C)	(A | A ∧ B)	(A | (A ∧ B) ∧ C)
1	1	1	1	1	1	1	1
1	1	0	1	0	1	1	1
1	0	1	0	0	1	1	1
1	0	0	0	0	1	1	1
0	1	1	0	0	1	1	1
0	1	0	0	0	1	1	1
0	0	1	0	0	1	1	1
0	0	0	0	0	1	1	1

Thus, in CSL+ we can have [(A ∧ B) ∧ C] → (A ∧ B) and also (A ∧ B) → A, as well as [(A ∧ B) ∧ C] → A. But what about [(A ∧ B) ∧ C] → [(A ∧ B) → A]? In the meta-language MCSL we can have this truth table:

---

[6] *Modus Tollens* is a rule of inference which says, from p → q and ¬q, one can infer ¬p.

A	B	C	A ∧ B	(A ∧ B) ∧ C	(A ∧ B) → A	((A ∧ B) → A | (A ∧ B) ∧ C)
1	1	1	1	1	1	1
1	1	0	1	0	1	1
1	0	1	0	0	1	1
1	0	0	0	0	1	1
0	1	1	0	0	1	1
0	1	0	0	0	1	1
0	0	1	0	0	1	1
0	0	0	0	0	1	1

Here (A ∧ B) → A takes the value 1 in the only case where (A ∧ B) ∧ C takes the value 1, so ((A ∧ B) → A | (A ∧ B) ∧ C) designates 1, which means the corresponding conditional, [(A ∧ B) ∧ C] → [(A ∧ B) → A], is true.

Can we have more deeply nested strong conditionals and equivalences in CSL+, or would we have to extend CSL+ to get a richer language to obtain them? Consider this sentence:

A → (A → [(A ∧ A) → (A ∨ A)])

Is this a wff of CSL+? We can begin its formulation by constructing a meta-linguistic truth table of MCSL, like so:

A	A ∧ A	A ∨ A	(A ∨ A | A ∧ A)	(A ∧ A) → (A ∨ A)	((A ∧ A) → (A ∨ A) | A)
1	1	1	1	1	1
0	0	0	1	1	1

Starting with the sentence of the fourth column, i.e. (A ∨ A | A ∧ A), and working to the right, each formula is either a designator of MCSL formed from some of the sentences of the preceding columns, or else a strong conditional formed from such a designator of a preceding column. Continuing the table, we obtain:

A → [(A ∧ A) → (A ∨ A)]	(A → [(A ∧ A) → (A ∨ A)] | A)	A → (A → [(A ∧ A) → (A ∨ A)])
1	1	1
1	1	1

So far, it seems that the answer to our question of whether we have to extend our language beyond CSL+ is "No": Given that it can handle some nested strong conditionals, it can handle them all. […]

5. *A Strong Conditional for the Logic of Paradox*

In this section I define a modified form of the strong conditional that will apply to our first example of a non-classical logic, which is an extension of Graham Priest's Logic of Paradox, or LP for short. LP is a "dialetheic" logic—one in which sentences may admissibly be deemed both true and false. It is motivated as a way to accept that some paradoxes, like the infamous Liar Paradox, are really what they appear to be: genuine contradictions. Whatever one may think of its motivation, it is, *qua* logic, an interesting system which merits investigation.

The language of LP, as I present it here, is much the same as the language of CSL. We will retain the same atomic sentences, connectives, brackets, and recursive clauses for well-formed formulas. The main difference lies in the assignments of truth values to sentences. In LP there are three truth values, the sets {1}, {0}, and {1, 0}; {1} being interpreted as *(exclusively) true*, {0} being interpreted as *(exclusively) false*, and {1, 0} being interpreted as *both true and false*. LP is a first-order language, but I will omit the details pertaining to the quantifiers, predicates, etc., because the issues we will consider concern only the propositional aspects of the language. Priest gives the following clauses for negation and conjunction, respectively [ref]:

$1 \in v(\neg A)$ iff $0 \in v(A)$

$0 \in v(\neg A)$ iff $1 \in v(A)$

$1 \in v(A \wedge B)$ iff $1 \in v(A)$ and $1 \in v(B)$

$0 \in v(A \wedge B)$ iff $0 \in v(A)$ or $0 \in v(B)$

(Where 'iff' means 'if and only if'). That is, 1 (true) is a member of the value of "¬A" iff 0 (false) is a member of the value of A, and 0 is a member of the value of "¬A" iff 1 is a member of the value of A. Similarly, 1 is a member of the value of "A ∧ B" iff 1 is both a member of the value of A and the value of B, and 0 is a member of the value of "A ∧ B" iff 0 is either a member of the value of A or the value of B (or both). Similar clauses may be given for the other connectives, the construction of which I will leave as an exercise for the reader.

Let us proceed to the definition of the strong conditional. We start with our meta-language for LP, i.e. MLP. MLP is much like MCSL. It includes the lowercase letters p, q, r, etc, as meta-variables, the sentences of LP itself, quote-names for both atomic and compound sentences of LP, corner quotes, and its own meta-linguistic designator $\ulcorner(q|p)\urcorner$. Since some sentences can actually take the value $\{1, 0\}$ in this logic, our account of MLP's designator will have to differ from the one we gave of MCSL's. To determine what value $\ulcorner(q|p)\urcorner$ designates, we construct a truth table and check to see if there are any rows where $1 \in v(p)$. If not, $\ulcorner(q|p)\urcorner$ is empty. If there are, we check to see if $1 \in v(q)$ in *every* row where $1 \in v(p)$. If so, $1 \in \ulcorner(q|p)\urcorner$. In addition, we check to see if $0 \in v(q)$ in every row where $1 \in v(p)$. If so, we also have $0 \in \ulcorner(q|p)\urcorner$. If neither 1 nor 0 is a member of $v(q)$ on every evaluation where $1 \in v(p)$, $\ulcorner(q|p)\urcorner$ is empty.

We'll begin our definition of strong implication for LP+ by appealing to the following truth table of MLP:

A	¬A	B	(A ∧ ¬A)	(A | A ∧ ¬A)	(A ∧ ¬A | A)	(B | A ∧ ¬A)	(A ∧ ¬A | B)
1	0	1	0	1, 0	0	%	0
1	0	0	0	1, 0	0	%	0
0	1	1	0	1, 0	0	%	0
0	1	0	0	1, 0	0	%	0
1	0	1, 0	0	1, 0	0	%	0
1, 0	1, 0	1	1, 0	1, 0	0	%	0
0	1	1, 0	0	1, 0	0	%	0
1, 0	1, 0	0	1, 0	1, 0	0	%	0
1, 0	1, 0	1, 0	1, 0	1, 0	0	%	0

The definition of strong implication for the extended language LP+ is then:

If ⌜(q|p)⌝ = {1}, then v(p → q) = {1}

If ⌜(q|p)⌝ = {0}, then v(p → q) = {0}

If ⌜(q|p)⌝ = {1, 0}, then v(p → q) = {1, 0}

If ⌜(q|p)⌝ is empty, then v(p → q) = {0}.

Given our truth table for ⌜(q|p)⌝ and this definition, we can obtain this truth table for strong implication in LP+:

A	¬A	B	(A ∧ ¬A)	(A ∧ ¬A) → A	(A ∧ ¬A) → B	A → (A ∧ ¬A)	B → (A ∧ ¬A)
1	0	1	0	1, 0	0	0	0
1	0	0	0	1, 0	0	0	0
0	1	1	0	1, 0	0	0	0
0	1	0	0	1, 0	0	0	0
1	0	1, 0	0	1, 0	0	0	0
1, 0	1, 0	1	1, 0	1, 0	0	0	0
0	1	1, 0	0	1, 0	0	0	0
1, 0	1, 0	0	1, 0	1, 0	0	0	0
1, 0	1, 0	1, 0	1, 0	1, 0	0	0	0

In LP+, strong implication satisfies Modus Ponens in this sense: If 1 ∈ v(p), and p → q, 1 ∈ v(q). This is significant because in LP the only candidate conditional is the material conditional, and it does not satisfy Modus Ponens. […]

6. *A Strong Conditional for K_3*

I'll now sketch a strong conditional for the logic K_3, a language which is basically the same as LP except that the third value is interpreted as "neither true nor false" rather than "both true and false." The language has much in common with LP, the only important difference being the notation for the truth values. We will continue to use 1 for truth and 0 for falsity, and introduce the letter n for "neither true nor false".

As usual, we introduce a conditional designator in the meta-language of K_3, MK_3. To evaluate ⌜(q|p)⌝ for a given truth table, we check to see if there is a row of the table where v(p) = 1. If not, ⌜(q|p)⌝ is empty. If there is a row where v(p) = 1, we check to see if v(q) = 1 in every row where v(p) = 1, and if it does, ⌜(q|p)⌝ = 1. If v(q) = 0 in every row where v(p) = 1, ⌜(q|p)⌝ = 0. If v(q) = n in every row where v(p) = 1, then ⌜(q|p)⌝ =n. As always, if v(q) varies when v(p) = 1, ⌜(q|p)⌝ is empty. In light of this, we have these truth tables:

A	¬A	A ∧ ¬A	(A | A ∧ ¬A)	(A ∧ ¬A | A)	(A ∧ ¬A | A ∧ ¬A)
1	0	0	%	0	%
n	n	n	%	0	%
0	1	0	%	0	%

A	¬A	A ∨ ¬A	(A | A ∨ ¬A)	(A ∨ ¬A | A)	(A ∨ ¬A | A ∨ ¬A)
1	0	1	%	1	1
n	n	n	%	1	1
0	1	1	%	1	1

A	¬A	B	(A ∧ ¬A)	(B | A ∧ ¬A)	(A ∧ ¬A | B)	A ∨ ¬A	(B | A ∨ ¬A)	(A ∨ ¬A | B)
1	0	1	0	%	%	1	%	%
1	0	0	0	%	%	1	%	%
0	1	1	0	%	%	1	%	%
0	1	0	0	%	%	1	%	%
1	0	n	0	%	%	1	%	%
n	n	1	n	%	%	n	%	%
0	1	n	0	%	%	1	%	%
n	n	0	n	%	%	n	%	%
n	n	n	n	%	%	n	%	%

In the extended language K_3+ we could then get corresponding truth tables with the strong conditional in place of the conditional designator by a process which is by now familiar enough that I shall spare the reader the details. What is distinctive about K_3+ is that, since sentences which are tautologies in CSL can be neither true nor false in K_3+, and hence not true, sentences of can K_3+ can fail to strongly imply a tautology. B fails to strongly imply $A \lor \neg A$, for B can have the value 1 while $A \lor \neg A$ has the value n. However, A still strongly implies $A \lor \neg A$, since $A \lor \neg A$ cannot be neither true nor false when A is true.

[…]

7. *A Strong Conditional for a Fuzzy Logic*

The logic that is the topic of this section is J. Łukasiewicz's logic Ł. Ł is what is known as a fuzzy logic. Fuzzy logics have an infinity of truth values, one for every real number between 0 and 1, which are interpreted as being *degrees of truth*—ways of being more-or-less true, or partially true and partially false. Such logics are often motivated by an ancient puzzle known as the Sorities Paradox. One instance of the paradox is this: Suppose there is a baby girl named 'Tracy'. Over time Tracy grows up, and eventually she will be an adult. A few moments after being born, Tracy is not an adult by any stretch of the imagination. At age fifty, she is an adult if anyone is. Now, it seems very plausible to claim that if Tracy is not an adult at some time t, she will also not be an adult one second later. She is not an adult a second after being born, nor is she an adult one second after that, nor one second after that… but eventually those seconds will add up to fifty years, and if this line of reasoning is correct she will still not be an adult! In classical logic, the principle of bivalence, which holds that every sentence is either true or false seems to demand that there's some precise point in the course of Tracy's life when she goes from being a child to being an adult. That is counterintuitive, to put it mildly. "Fuzzy logicians" propose replace that point with a smooth gradient: As Tracy gets older, that claim that she is an adult gradually goes from being mildly true to moderately true to very true, and eventually to being completely true.

In Łukasiewicz's logic Ł the definition of negation is:

$$v(\neg p) = 1 - v(p)$$

That is, ¬p's degree of truth is 1 minus the degree of truth of p, which seems plausible enough.

The definition of conjunction is:

$v(p \wedge q) = Min(v(p), v(q))$

This means that the truth value of ⌜p ∧ q⌝ is the minimum of the value of p and the value of q—the value of whichever of them happens to be lower (If their values are equal, it has the value they share.) This is a generalization of the definition of conjunction for classical logic: 0 is less than 1, so if either p or q has the value 0 ⌜p ∧ q⌝ must have the value 0 as well. If p and q both have the value 1, ⌜p ∧ q⌝ has the value 1.

It proves to be just as easy to define disjunction as it was to define conjunction:

$v(p \vee q) = Max(v(p), v(q))$

The truth value of ⌜p ∨ q⌝ is the maximum of the values of p and q—the value of whichever of them happens to be greater (As above, if their values are equal, it has the value they share.) This is a generalization of the definition of disjunction for classical logic: 1 is greater than 0, so if either p or q has the value 1 so does ⌜p ∨ q⌝, and if both p and q have the value 0 then ⌜p ∨ q⌝ does too.[…]

MŁ contains an operator which we have not yet encountered in any of the preceding languages. This is the operator [d], which I call 'op', which in a sense bridges the gap between MŁ and Ł+. Op is an operator forming operator: It takes a degree of truth, d, and forms a sentential operator which, when prefixed to a sentence p, has the effect of saying that p has d as its degree of truth. […]

We come now to the definition of MŁ's conditional designator. If ⌜(q|p)⌝ is empty, $v(p \rightarrow q) = 0$. If ⌜(q|p)⌝ is not empty and p's degree of truth, d, is 1, then $v(p \rightarrow q)$ = ⌜(q|p)⌝. If ⌜(q|p)⌝ is not empty and d is distinct from 1, we can "fudge":

⌜(q |[d]p)⌝ = $v(p \rightarrow q)$

The reason we can fudge is that $v([d]p) = 1$ iff $v(p) = d$. If a sentence attributes to p any value other than d, that sentence has the value 0. Thus, any sentence which has an operator formed from op as its main operator either has 1 or 0 as its truth value; there are no intermediate possibilities.

Now, if we wish to know, for example, what value A ∨ ¬A must take when A is true to the degree 0.75, we begin by substituting '0.75' for 'd' in op and by substituting 'A' for 'p' to form the sentence [0.75]A. Then we take both of these sentences and substitute them into MŁ's conditional designator, yielding: (A ∨ ¬A | [0.75]A). To evaluate this sentence, we start (surprise!) by constructing a truth table in MŁ:

A	¬A	A ∨ ¬A	[0.75]A	(A ∨ ¬A | [0.75]A)
1	0	1	0	0.75
0.75	0.25	0.75	1	0.75
0	1	1	0	0.75

If we want to know what value A ∨ ¬A takes when A takes on other values, say 0.50 or 0.25, we proceed similarly:

A	¬A	A ∨ ¬A	[0.50]A	(A ∨ ¬A | [0.50]A)	[0.25]A	(A ∨ ¬A | [0.25]A)
1	0	1	0	0.50	0	0.75
0.50	0.50	0.50	1	0.50	0	0.75
0.25	0.75	0.75	0	0.50	1	0.75
0	1	1	0	0.50	0	0.75

[…]