Propositions

John F. Sowa

Abstract. Informally, sentences in different languages may mean "the same thing." Formally, that "thing," called a *proposition*, represents abstract, language-independent, semantic content. As an abstraction, a proposition has no physical embodiment that can be written or spoken. Only its statements in particular languages can be expressed as strings of symbols. To bring the informal notion of proposition within the scope of formal treatment, this paper proposes a formal definition: a proposition *p* shall be defined as an equivalence class of sentences in some formal language **L** under some *meaning-preserving translation* (MPT) defined over the sentences of **L**. For any equivalence class *p* and model **M**, all the sentences in *p* shall have the same vocabulary, structure, and truth value. This paper defines a series of six MPTs $f_0,...,f_5$. For most purposes, f_4 and f_5 are the most useful.

Note: This paper is a revised and extended excerpt from Section 4 of Worlds, Models, and Descriptions. See that article for further discussion.

A proposition should represent the language-independent meaning of a sentence, but "language independence" and "meaning" are two notoriously difficult notions to formalize. Stalnaker (1976) proposed that the proposition expressed by a sentence should be defined as the set of possible worlds in which the sentence is true. That definition, however, is too coarse grained: it causes all mathematical theorems to collapse into a single proposition. Fermat's last theorem, for example, doesn't mean the same as 2+2=4, and it's much harder to prove. Peirce had a better definition: a proposition corresponds to a collection of equivalent sentences with their partial interpretants:

> To say that a proposition is true is to say that every interpretation of it is true. Two propositions are equivalent when either might have been an interpretant of the other. This equivalence, like others, is by an act of abstraction (in the sense in which forming an abstract noun is abstraction) conceived as identity. And we speak of believing in a proposition, having in mind an entire collection of equivalent propositions with their partial interpretants. Thus, two persons are said to have the same proposition in mind. The interpretant of a proposition is itself a proposition. Any necessary inference from a proposition is an interpretant of it. (CP 5.569).

To formalize Peirce's theory of interpretants, a *proposition* shall be defined as an equivalence class of sentences under some *meaning-preserving translation* (MPT). Preservation of truth is one requirement for an MPT, but other properties, such as vocabulary and structure, should also be preserved. For simplicity, this paper will restrict the definition to some version of first-order logic. An MPT from the sentences of a first-order language $\mathbf{L}_1$ to the sentences of a first-order language $\mathbf{L}_2$ shall be a function *f* that satisfies the following four constraints:

1. *Invertible*. The translation function *f* must have an inverse function *g* that maps sentences from $\mathbf{L}_2$ back to $\mathbf{L}_1$. For any sentence *s* in $\mathbf{L}_1$, $f(s)$ is a sentence in $\mathbf{L}_2$, and $g(f(s))$ is a sentence in $\mathbf{L}_1$ but not necessarily *s* itself. To ensure that *f* is defined for all sentences in $\mathbf{L}_1$, the language $\mathbf{L}_2$ must be at least as expressive as $\mathbf{L}_1$. If $\mathbf{L}_2$ is more expressive than $\mathbf{L}_1$, then the inverse *g* might be undefined for some sentences in $\mathbf{L}_2$. In that case, the language $\mathbf{L}_2$ would express a superset of the propositions of $\mathbf{L}_1$.

2. *Truth preserving*. Although the sentences s and $g(f(s))$ might not be identical, both must have exactly the same truth conditions: for any model **M** of the language $\mathbf{L}_1$, M⊨s if and only if **M**⊨$g(f(s))$. Preserving truth is necessary, but not sufficient: the condition is too coarse to distinguish 2+2=4 from Fermat's last theorem; and its computation may be intractable or even undecidable. Ideally, the test to determine whether two sentences "mean the same" should be "obvious." Formally, it should be computable by an efficient algorithm — one whose computation time is linearly or at worst polynomially proportional to the length of the sentence.
3. *Vocabulary preserving*. When s is translated from $\mathbf{L}_1$ to $\mathbf{L}_2$ and back to $g(f(s))$, the syntax might be rearranged, but the words or symbols that represent proper names and ontology must appear in both sentences s and $g(f(s))$. For example, the sentence *Every cat is a cat* should not be considered to have the "same meaning" as *Every dog is a dog* or *Every unicorn is a unicorn*, since the first is about cats, the second is about dogs, and the the third is about nonexistent things. This criterion could be relaxed to allow terms to be replaced by synonyms or definitions, but arbitrary content words or predicates must not be added or deleted by the translations. An admissible MPT might replace the word *cat* with *domestic feline*, but it should not replace the word *cat* with *dog* or *unicorn*.
4. *Structure preserving*. When s and $g(f(s))$ are mapped to a *canonical form* that uses only negation ~, conjunction ∧, and the existential quantifier ∃ as the logical operators, they must contain computationally equivalent patterns of negations and quantifiers: i.e., the decidability, provability, and computational complexity of both sentences must be the same. (Formally, the functions f_1 through f_4, which are defined below, specify what patterns are considered computationally equivalent.) The short justification for this approach is that conjunction is the simplest and least controversial of all the Boolean operators, while negation introduces serious philosophical and computational problems, which are inherited by other operators that require a negation in their definition. Intuitionists, for example, deny that ~~p should be considered provably equivalent to p. Computationally, ~~p and p have different effects on the binding of values to variables in Prolog, SQL, and other systems. For *relevance logic*, Anderson and Belnap (1975) disallowed the *disjunctive syllogism*, which is based on ∨ and ~, because it can introduce extraneous information into a proof.

These four conditions impose strong constraints on translations that are said to preserve meaning. They ensure that the content words or predicates remain identical or synonymous, they preserve the logical structure, and they prevent irrelevant content from being inserted. If s is the sentence *Every farmer who owns a donkey beats it*, then the sentence $g(f(s))$ might be *If a farmer x owns a donkey y, then x beats y.* Those sentences use different logical and syntactical symbols, but they have the same truth conditions, they have the same content words, and they have the same structure when expressed with only ∧, ~, and ∃.

In general, an MPT is defined as a function that translates sentences from one language to another, but for many applications $\mathbf{L}_1$ and $\mathbf{L}_2$ may be the same language. Any MPT f defined over a single language **L** partitions the sentences of **L** into a set of equivalence classes called *propositions*. A series of six MPTs f_0, ..., f_5 are defined below, which have the following properties:

- Each MPT f_i is its own inverse: $g_i = f_i$.
- Every equivalence class p in the partition of **L** determined by f_i contains exactly one *canonical sentence c*.
- For every sentence s in p, $f_i(s) = c$ and $g_i(f_i(s)) = c$.

- Any two sentences s_1 and s_2 of **L** are contained in the same equivalence class if and only if f_i maps both of them to the same canonical sentence c: $f_i(s_1) = c$ and $f_i(s_2) = c$.

As an abbreviation, the sentence s is said to *state* the proposition p if and only if s is contained in the equivalence class p.

For any language **L**, the simplest MPT f_0 is the identity function: for every sentence s in **L**, $f_0(s)=s$. This translation satisfies all four criteria for an MPT, since f_0 is its own inverse, it preserves truth, vocabulary, and structure, and it is very easy to compute. Its primary drawback is that it fails to group sentences into useful classes: every sentence in **L** is the canonical sentence of a proposition in which there are no other sentences.

In order to group sentences into larger equivalence classes, the MPT f_1 translates every sentence to a canonical sentence that uses only the three logical symbols $\wedge$, $\sim$, and $\exists$. The inverse of f_1 is f_0 itself. The details for defining f_1 will vary with the syntax of every version of FOL, but to illustrate the method, let **L** be a conventional infix notation for FOL with the two quantifiers represented by the symbols $\exists$ and $\forall$, with negation represented by $\sim$, and with conjunction, disjunction, material implication, and equivalence represented by the symbols $\wedge$, $\vee$, $\supset$, and $\equiv$. The translation of any sentence or subsentence s of **L** to its canonical form $f_1(s)$ is defined recursively:

1. If s is an *atom* (i.e., any relation symbol applied to its arguments), the result is s unchanged:

   $f_1(s) \Rightarrow s$

2. If s consists of a prefix operator (negation or quantifier) applied to a subsentence u, the result is the transformed operator applied to the translation of u:

   $f_1(\sim u) \Rightarrow \sim f_1(u)$

   $f_1((\exists x)u) \Rightarrow (\exists x) f_1(u)$

   $f_1((\forall x)u) \Rightarrow \sim(\exists x)\sim f_1(u)$

3. If s consists of a dyadic operator ($\wedge$, $\vee$, $\supset$, or $\equiv$) applied to two subsentences u and v, the result is a transformation of the operator applied to the translations of u and v:

   $f_1(u \wedge v) \Rightarrow f_1(u) \wedge f_1(v)$

   $f_1(u \vee v) \Rightarrow \sim(\sim f_1(u) \wedge \sim f_1(v))$

   $f_1(u \supset v) \Rightarrow \sim(f_1(u) \wedge \sim f_1(v))$

   $f_1(u \equiv v) \Rightarrow (\sim(f_1(u) \wedge \sim f_1(v))) \wedge (\sim(f_1(v) \wedge \sim f_1(u)))$

The function f_1 satisfies the criteria for an MPT: it has an inverse f_1; it preserves truth because every step of the translation expands one operator according to its definition in terms of $\wedge$, $\sim$, and $\exists$; it leaves the content words and symbols unchanged; and it preserves the structure of a sentence when mapped to $\wedge$, $\sim$, and $\exists$. For any sentence s, $f_1(s)$ is a unique canonical sentence of the equivalence class defined by f_1: every sentence in **L** that is mapped to that canonical sentence states the same proposition. For all logical operators except $\equiv$, the time to convert any sentence to the canonical sentence is linear. If a sentence contains a large number of equivalences, the translation time may be exponentially proportional to the number of equivalences. Such sentences, however, are extremely rare; when people write $p \equiv q \equiv r$, they usually intend it as the conjunction $p \equiv q$ and $q \equiv r$, which merely doubles the translation

time.

The function f_1 would classify sentences of the form $p \wedge q$ and $q \wedge p$ in different equivalence classes because its definition does not reorder any of the subsentences. To group such sentences in the same equivalence class, the function f_2 makes the same symbol replacements as f_1, but it also sorts conjunctions in lexical order according to some encoding, such as Unicode. As a result, arbitrary permutations of conjunctions map to the same canonical sentence and are grouped in the same equivalence class. Since disjunctions are defined in terms of of conjunctions, the sentences $p \vee q$ and $q \vee p$ would also be in the same class. Since a list of N terms can be sorted in time proportional to N*log*N, the function f_2 would take just slightly longer than linear time.

Ideally, sentences that differ only in the names assigned to their bound variables, such as $(\exists x)\mathrm{P}(x)$ and $(\exists y)\mathrm{P}(y)$, should also be grouped together. Therefore, the function f_3 should rename the variables to a fixed sequence, such as x_1, x_2 , ... That renaming must be done before conjunctions are sorted; the simplest method would be to move all quantifiers to the front by mapping every sentence to *prenex normal form* and then naming the variables in the order in which they appear. The function f_3 would first convert sentences to prenex normal form and rename variables, then it would perform the translation method of f_1 and sort conjunctions according to the method of f_3.

The function f_3 is not yet an ideal MPT because it would assign sentences of the form u, $u \wedge u$, and $u \vee u$ to different equivalence classes. To group such sentences together, the function f_4 would perform exactly the same translations as f_3 followed by one additional step: deleting duplicate conjunctions. Since f_3 has already sorted conjunctions in lexical order, the deletion can be performed in linear time just by checking whether any conjunct is identical to the one before it; if so, it would be deleted from the result. This process would delete duplicate disjunctions since the method of f_1 would have already mapped disjunctions to combinations of conjunctions and negations.

For most purposes, the function f_4 is a good MPT for grouping sentences that "say the same thing." As an example, consider the following sentence, which Leibniz called the *Praeclarum Theorema* (splendid theorem):

$$((p \supset r) \wedge (q \supset s)) \supset ((p \wedge q) \supset (r \wedge s))$$

This formula may be read *If p implies r and q implies s, then p and q imply r and s*. The function f_3 would translate it to the following canonical sentence:

$$\sim((\sim(p \wedge \sim r) \wedge \sim(q \wedge \sim s)) \wedge \sim(\sim(p \wedge q) \wedge \sim(r \wedge s)))$$

This sentence is not as readable as the original, but it serves as the canonical representative of an equivalence class of f_3 that contains 864 different, but highly similar sentences. The function f_4, which deletes duplicate conjuncts, can relate infinitely many sentences to the same form. Such transformations factor out accidental differences caused by the choice of symbols or syntax.

To account for synonyms and definitions, another function f_5 could be used to replace a word or relation such as *cat* with its definition as *domestic feline*. If recursions are allowed, the replacements and expansions would be equivalent in computing power to a Turing machine; they could take exponential amounts of time or even be undecidable. Therefore, f_5 should only expand definitions without recursions, direct or indirect. Definitions of this form are common in database systems, in which a fixed set of relations are privileged, and *virtual relations* may be defined in terms of the privileged relations. With such restrictions, any sentence that uses virtual relations could always be expanded to a form that only uses relations in the privileged set. Therefore, the function f_5 would first expand all virtual relations to the privileged set before performing the translations defined for f_4. Since

no recursion is involved, the expansions would take at most polynomial time.

In summary, an open-ended number of meaning-preserving translations could be defined. For ease of computing, an MPT that determines a unique canonical form for any proposition in at worst polynomial time is recommended. If no definitional mechanisms are available in the given language **L**, MPT f_4 is recommended. If **L** supports nonrecursive definitions, f_5 could be used.

References

Peirce, Charles Sanders (CP) *Collected Papers of C. S. Peirce*, ed. by C. Hartshorne, P. Weiss, & A. Burks, 8 vols., Cambridge, MA: Harvard University Press, 1931-1958.

Sowa, John F. (2006) Worlds, models, and descriptions, *Studia Logica*, Special Issue on Ways of Worlds II **84:2**, 323-360.

Stalnaker, Robert (1976) Propositions, in A. MacKay & D. Merrill, eds., *Issues in the Philosophy of Language*, New Haven, CT: Yale University Press, pp. 79-91.