Set-Oriented Logical Connectives: Syntax and Semantics

Stuart C. Shapiro
Department of Computer Science and Engineering and Center for Cognitive Science
The State University of New York at Buffalo
Buffalo, NY 14260-2000
shapiro@cse.buffalo.edu

Abstract

Of the common commutative binary logical connectives, only `and` and `or` may be used as operators that take arbitrary numbers of arguments with order and multiplicity being irrelevant, that is, as connectives that take sets of arguments. This is especially evident in the Common Logic Interchange Format, in which it is easy for operators to be given arbitrary numbers of arguments. The reason is that `and` and `or` are associative and idempotent, as well as commutative. We extend the ability of taking sets of arguments to the other common commutative connectives by defining generalized versions of `nand`, `nor`, `xor`, and `iff`, as well as the additional, parameterized connectives `andor` and `thresh`. We prove that `andor` is expressively complete—all the other connectives may be considered abbreviations of it.

1. Introduction

A commonly used syntax for formulas of Propositional Logic is the Common Logic Interchange Format (CLIF) (ISO/IEC 2007). Assuming that $a, b, c, p_1, \ldots, p_n$ are CLIF well-formed formulas (wffs), examples of non-atomic expressions in CLIF are: (not a), (and $p_1 \ldots p_n$), (or $p_1 \ldots p_n$), (if a b), and (iff a b).

CLIF uses "Cambridge prefix" notation, the benefits of which are a simple, consistent syntax, and that operators may easily be given arbitrary numbers of arguments. We may well then ask why `and` and `or` are the only two logical connectives in CLIF that can take an arbitrary number of arguments. One quick answer is that `not` only takes one argument, and `if` is not commutative. However, these reasons do not apply to `iff`, nor to the other common connectives, `nor`, `nand`, and `xor`.

Not only do `and` and `or` take arbitrary numbers of arguments, they take sets of arguments. (By definition, (and) is T and (or) is F.) That is, order and multiplicity are irrelevant among the arguments: (and a b c) is the same as (or, at least, is equivalent to) (and c b a); and (or a a b c b c a) is the same as (or, at least, is equivalent to) (or a b c).[1]

[1] The two examples of each pair might literally be the same if the KR system gives them the same internal representation.

What is special about `and` and `or` is that they are associative and idempotent, as well as being commutative. Consider a fully parenthesized expression all of whose operators are the same associative, commutative, idempotent, binary operator. Because the operator is associative, inner parentheses may be removed; because it is commutative, the order of the operands may be permuted; because it is idempotent, multiple occurrences of any one operand may be exchanged for just a single occurrence. Because `and` and `or` are associative, commutative, and idempotent, they may be given arbitrary numbers of arguments, with the order and multiplicity being irrelevant. That is, they may be considered connectives that take sets of arguments. However, `xor` and `iff` are associative, but not idempotent, and `nor` and `nand` are neither associative nor idempotent.

Even more surprising is that wffs consisting of one of these connectives multiple times do not have the semantics most people assume. For example: for `nor`, $(F \downarrow T \downarrow F) \equiv T$; for `nand`, $(T \mid T \mid T) \equiv T$ for `xor`, $(T \oplus T \oplus T) \equiv T$ and for `iff`, $(F \Leftrightarrow F \Leftrightarrow T) \equiv T$.

2. Generalizing

We now introduce generalized versions of `nor`, `nand`, `xor`, and `iff` that have the semantics we want. From here on, we use `nor`, `nand`, `xor`, and `iff` to mean these generalized versions. We will use i, j, and n as metalinguistic variables ranging over nonnegative integers, a, b, c, p_1, ..., p_n as metalinguistic variables ranging over wffs, and the metalinguistic relations $\equiv$ for logical equivalence and $\models$ for logical entailment. Except where otherwise noted, we will assume that p_1, and ..., and p_n are not necessarily distinct wffs.

Syntax: (con $p_1 \ldots p_n$), $n \geq 0$, where con is either `nor`, `nand`, `xor`, or `iff`.

Semantics: For each connecitve, the general case and the base case of an empty set of arguments are shown.

- (nor $p_1 \ldots p_n$) is True if p_1, and ..., and p_n are all False; otherwise it is False.
  (nor) $\equiv$ True.
- (nand $p_1 \ldots p_n$) is False if p_1, and ..., and p_n are all True; otherwise it is True.
  (nand) $\equiv$ False.

- $(\texttt{xor}\ p_1 \ldots p_n)$ is True if exactly one $p_i \in \{p_1, \ldots, p_n\}$ is True; otherwise it is False.
  $(\texttt{xor}) \equiv$ False.
- $(\texttt{iff}\ p_1 \ldots p_n)$ is True if p_1, and $\ldots$, and p_n are all True or are all False; otherwise it is False.
  $(\texttt{iff}) \equiv$ True.

Theorem 1. *When restricted to two arguments, the generalized* `nor`*,* `nand`*,* `xor`*, and* `iff` *are equivalent to the respective standard binary connectives.*

Proof. By inspection of the syntax and semantics. □

Theorem 2. *For any wff,* a, $(\texttt{nor}\ a) \equiv (\texttt{not}\ a)$.

Proof. In every model in which a is True, both $(\texttt{nor}\ a)$ and $(\texttt{not}\ a)$ are False; in every model in which a is False, $(\texttt{nor}\ a)$ and $(\texttt{not}\ a)$ are True. □

3. andor

The connectives `and`, `or`, `not`, `nor`, `xor`, and `nand` are all special cases of one parameterized connective, `andor`.

Syntax: $(\texttt{andor}\ (i\ j)\ p_1 \ldots p_n), 0 \leq i \leq j \leq n$.

Semantics: $(\texttt{andor}\ (i\ j)\ p_1 \ldots p_n)$ is True if at least $min(i, |\{p_1 \ldots p_n\}|)$ and at most $min(j, |\{p_1 \ldots p_n\}|)$ of $p_i \in \{p_1, \ldots, p_n\}$ are True; otherwise it is False.
$(\texttt{andor}\ (0\ 0)) \equiv$ True.

Theorem 3 shows that `and`, `or`, `not`, `nor`, `xor`, and `nand` are special cases of `andor`.

Theorem 3.

1. $(\mathit{and}\ p_1 \ldots p_n) \equiv (\mathit{andor}\ (n\ n)\ p_1 \ldots p_n)$
2. $(\mathit{or}\ p_1 \ldots p_n) \equiv (\mathit{andor}\ (1\ n)\ p_1 \ldots p_n)$
3. $(\mathit{not}\ a) \equiv (\mathit{andor}\ (0\,0)\ a)$
4. $(\mathit{nor}\ p_1 \ldots p_n) \equiv (\mathit{andor}\ (0\,0)\ p_1 \ldots p_n)$
5. $(\mathit{nand}\ p_1 \ldots p_n) \equiv (\mathit{andor}\ (0\ n-1)\ p_1 \ldots p_n)$
6. $(\mathit{xor}\ p_1 \ldots p_n) \equiv (\mathit{andor}\ (1\,1)\ p_1 \ldots p_n)$

Proof. Straightforward from the semantics. □

Not only can every wff using `and`, `or`, `not`, `nor`, `nand`, and `xor` be translated, preserving semantics, into a wff using only `andor`, but also every wff using only `andor` can be translated, preserving semantics, into a wff using only `and`, `or`, and `not`.

Theorem 4. [2] *For any integers,* j, k, n *such that* $0 \leq j \leq k \leq n$*, and any distinct wffs,* $p_1, \ldots, p_n$*,*
$(\mathit{andor}\ (j\ k)\ p_1 \ldots p_n) \equiv \bigvee\{\bigwedge(p \cup \{\neg \bigvee(\mathcal{P} - p)\}) \mid p \in \bigcup_{i=j}^{k} \mathit{choose}(i, \mathcal{P})\}$,
where:

- $\mathcal{P} = \{p_1 \ldots p_n\}$;
- *for any positive integer,* i*, and set of wffs,* P*, choose(i, P) is the set of all the subsets of* P *of size* i*;*
- *for any wff,* $p, \neg p = (\mathit{not}\ p)$*;*
- *and for any set of wffs,* $P = \{q_1, \ldots q_n\}$*,*
  $\bigvee P = (\mathit{or}\ q_1, \ldots q_n)$
  $\bigwedge P = (\mathit{and}\ q_1, \ldots q_n)$

Proof. $(\texttt{andor}\ (j\ k)\ p_1 \ldots p_n)$ is True just in case for some $i, j \leq i \leq k$, the wffs in some subset of $\mathcal{P}$ of size i are True, and all the rest are False. $\bigcup_{i=j}^{k} \mathit{choose}(i, \mathcal{P})$ is the set of all subsets of $\mathcal{P}$ of size between j and k, inclusive. So $(\texttt{andor}\ (j\ k)\ p_1 \ldots p_n)$ is True just in case any $p \in \bigcup_{i=j}^{k} \mathit{choose}(i, \mathcal{P})$ is a set of True wffs and all the wffs in $p - \mathcal{P}$ are False. All the wffs in $p - \mathcal{P}$ are False iff the wff $\neg \bigvee(\mathcal{P} - p)$ is True. So the given p is True and all the wffs in $p - \mathcal{P}$ are False iff the wff $\bigwedge(p \cup \{\neg \bigvee(\mathcal{P} - p)\})$ is True. Therefore, $(\texttt{andor}\ (j\ k)\ p_1 \ldots p_n)$ is True just in case $\bigvee\{\bigwedge(p \cup \{\neg \bigvee(\mathcal{P} - p)\}) \mid p \in \bigcup_{i=j}^{k} \mathit{choose}(i, \mathcal{P})\}$ is True. □

`andor` is expressively complete, in the sense that any wff of Propositional Logic is equivalent to one that uses `andor` as its only connective.

Theorem 5. `andor` *is expressively complete.*

Proof. By Theorem 3, any formula containing any of the connectives `not`, `and`, `or`, `nor`, or `nand` can be replaced by a logically equivalent formula using only `andor`. Since `not` and `and`; `not` and `or`; `nor`; and `nand` each form an expressively complete set of connectives, `andor` is expressively complete. □

4. thresh

Just as `(nand ...)` $\equiv$ `(not (and ...))` and `(nor ...)` $\equiv$ `(not (or ...))`, we can define a connective equivalent to $(\texttt{not}\ (\texttt{andor}\ (i\ j) \ldots))$, which, for historical reasons, we call `thresh`.

Syntax: $(\texttt{thresh}\ (i\ j)\ p_1 \ldots p_n), 0 \leq i \leq j \leq n$.

Semantics: $(\texttt{thresh}\ (i\ j)\ p_1 \ldots p_n)$ is True if either fewer than $min(i, |\{p_1 \ldots p_n\}|)$ or more than $min(j, |\{p_1 \ldots p_n\}|)$ of $p_i \in \{p_1, \ldots, p_n\}$ are True; otherwise it is False.
$(\texttt{thresh}\ (0\ 0)) \equiv$ False.

Theorem 6. $(\mathit{thresh}\ (i\ j)\ p_1 \ldots p_n) \equiv (\mathit{not}\ (\mathit{andor}\ (i\ j)\ p_1 \ldots p_n))$

Proof. $(\texttt{andor}\ (i\ j)\ p_1 \ldots p_n)$ is True just in case at least $min(i, |\{p_1 \ldots p_n\}|)$ and at most $min(j, |\{p_1 \ldots p_n\}|)$ of $p_i \in \{p_1, \ldots, p_n\}$ are True. So $(\texttt{not}\ (\texttt{andor}\ (i\ j)\ p_1 \ldots p_n))$ is True if either fewer than $min(i, |\{p_1 \ldots p_n\}|)$ or more than $min(j, |\{p_1 \ldots p_n\}|)$ of $p_i \in \{p_1, \ldots, p_n\}$ are True, which are just the situations in which $(\texttt{thresh}\ (i\ j)\ p_1 \ldots p_n)$ is True. □

Corollary 1. $(\mathit{andor}\ (i\ j)\ p_1 \ldots p_n) \equiv (\mathit{not}\ (\mathit{thresh}\ (i\ j)\ p_1 \ldots p_n))$

Proof. Follows immediately from Theorem 6, and the fact that $(a \equiv \neg b) \models (\neg a \equiv b)$. □

It is easy to show that `thresh` generalizes `and`, `or`, `nand`, `nor`, and the identity function. However, `thresh` also generalizes `iff`.

[2] Without loss of generality, this theorem considers the distinct wffs, $p_1, \ldots, p_n$, among the set of arguments.

Theorem 7. [3] *For any integer, $n \geq 2$, and any distinct wffs, $p_1, \ldots, p_n$,*
$(\texttt{iff}\ p_1 \ldots p_n) \equiv (\texttt{thresh}\ (1\ n-1)\ p_1 \ldots p_n)$.

Proof. $(\texttt{thresh}\ (1\ n-1)\ p_1 \ldots p_n)$ is True just in case either fewer than 1 or more than $n-1$ of p_1, and ..., and p_n are True. That means that it is True just in the situations in which either p_1, and ..., and p_n are all False, or they are all True, which are just the situations in which $(\texttt{iff}\ p_1 \ldots p_n)$ is True. □

5. Syntactic Sugar?

Since, by Theorems 3, 4, and 6, any formula containing any of these set-oriented connectives may be translated into one containing only `and`, `or`, and `not`, it may be felt that the set-oriented connectives are "only" syntactic sugar. Indeed, Theorem 4 shows a combinatorial increase in formula length when `andor` is removed. However, that is precisely the point. KRR systems should provide these connectives to their users, allowing humans to express information concisely, and leave it to the program to expand the wffs into long ones that use the less expressive connectives.

6. Previous Literature

The generalized `nor` was defined by (Wittgenstein 1922, 5.502, 5.51). The logic gates `AND`, `OR`, `NAND`, and `NOR` are defined as taking an arbitrary number of inputs in (Weik 1969). The generalized `iff` was defined in (Shapiro 1971), where it was called `MUTIMP`. A logic gate equivalent to `andor` was introduced in (Epstein 1958). Independently, `andor` was introduced in (Bechtel and Shapiro 1976), along with a `thresh` that only has the i parameter. (Shapiro 1979) gives the syntax and semantics of versions of `andor` and the one-parameter `thresh`. The generalized `xor` was defined in (Hayes 1985, p. 72) and (Davis 1990, p. 32). The syntax and semantics of the two-parameter `thresh` was defined in (Choi and Shapiro 1992).

7. Conclusions

Of the common commutative binary logical connectives, only `and` and `or` may be used as connectives that take sets of arguments. This is especially evident in CLIF, a format in which it is particularly easy for operators to be given arbitrary numbers of arguments. This deficit may be overcome by using the generalized versions of `nand`, `nor`, `xor`, and `iff`, and the parameterized connectives, `andor`, and `thresh`, `andor` being expressively complete. The only computational cost in using the set-oriented connectives in existing reasoners is incurred when translating formulas that contain them into wffs containing only the connectives implemented in the reasoners. However, not using them means that the human user incurs the same cost when formalizing information using the less expressive traditional connectives. The burden of formulating long formulas using inexpressive connectives should be borne by programs, not by people. For illustration, `ubprover`, a pedagogical resolution refutation theorem prover using the set-oriented connectives discussed in this paper is available for downloading at `http://www.cse.buffalo.edu/~shapiro/Software/`.

8. Acknowledgments

The author appreciates the comments of Randall Dipert, Albert Goldfain, William J. Rapaport, and A. Patrice Seyed on earlier drafts of this paper, and the contributions of past and present members of the SNePS Research Group in helping to formulate, implement, and use these set-oriented connectives.

References

Bechtel, R., and Shapiro, S. C. 1976. A logic for semantic networks. Technical Report 47, Computer Science Department, Indiana University, Bloomington, IN. Presented at the 1976 Computer Science Conference, Anaheim, CA, February 10–12, 1976.

Choi, J., and Shapiro, S. C. 1992. Efficient implementation of non-standard connectives and quantifiers in deductive reasoning systems. In *Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences*. Los Alamitos, CA: IEEE Computer Society Press. 381–390.

Davis, E. 1990. *Representations of Commonsense Knowledge*. San Mateo, CA: Morgan Kaufmann.

Epstein, G. 1958. Synthesis of electronic circuits for symmetric functions. *IRE Transactions on Electronic Computers*.

Hayes, P. J. 1985. Naive physics I: Ontology for liquids. In Hobbs, J. R., and Moore, R. C., eds., *Formal Theories of the Commonsense World*. Norwood, NJ: Ablex Publishing Corporation. 71–107.

ISO/IEC. 2007. *Information technology — Common Logic (CL): a framework for a family of logic-based languages, ISO/IEC 24707:2007(E)*. ISO/IEC, Switzerland, First edition. available from `http://standards.iso/ittf/license.html`.

Shapiro, S. C. 1971. A net structure for semantic information storage, deduction and retrieval. In *Proceedings of the Second International Joint Conference on Artificial Intelligence*. San Mateo, CA: Morgan Kaufmann. 512–523.

Shapiro, S. C. 1979. The SNePS semantic network processing system. In Findler, N. V., ed., *Associative Networks: The Representation and Use of Knowledge by Computers*. New York: Academic Press. 179–203.

Weik, M. H. 1969. *Standard Dictionary of Computers and Information Processing*. New York: Hayden Book Company.

Wittgenstein, L. 1922. *Tractatus Logico-Philosophicus*. London: Routledge & Kegan Paul, Ltd. Translated by C. K. Ogden.

[3] Without loss of generality, this theorem considers the distinct wffs, $p_1, \ldots, p_n$, among the set of arguments.