Theme One
A Program of Inquiry

by

Jon Awbrey
Susan Awbrey

Chapter 2

Study. The Universe Modeler

The notation we use for propositional calculus is derived from the system of Logical Graphs developed by Charles S. Peirce around the turn of the century [P1, P2, P3]. These ideas were extended by the work of G. Spencer Brown in his book Laws of Form [SpB]. The present notation develops this system of Logical Graphs only by an operation of reflection that reapplies its own founding principles to the received formulation.

These founding principles, or esthetic imperatives, present throughout Peirce's logical work, are:

> Try to reduce the number of primitive notions.
>
> Try to vary what has been held to be constant.

Our present aim draws us quickly past these reflections, but see if you can recognize for yourself how they issue in the system that follows.

In propositional calculus, as used here, our purpose will be to describe some universe of discourse in terms of a set of features or propositions that are interpreted as primitive. We will want the capacity to define terms in terms of each other, and more generally to impose constraints on the logical compatibility of terms. Given a set of such definitions and constraints we would like the facility of deducing their logical equivalents and consequences.

The fastest way to the point will be to assume the reader already knows some version of propositional calculus, that domain of reasoning depicted in Venn diagrams and in Truth tables, and to present what follows as nothing more than a variant notation for it. In this way we only need to show how to represent an adequate set of basic expressions in the new notation.

We take as terms the words in some set W. These are usually interpreted as referring to features of or propositions about elements in the universe of discourse, whether objects, events, situations, or whatever category is under discussion. Terms correspond to the labeled circles of Venn diagrams or the sentence letters of Truth tables.

We take such terms to be the simplest expressions of the calculus for W, and we form more complex expressions by combining any expressions in basically two ways:

Any finite sequence of N expressions (think of starting with terms) may be connected by means of N–1 blanks to form a new expression.

Any finite sequence of N expressions may be connected by means of an operator (, , , ...) with N–1 commas, putting one expression into each of the N slots between commas, to form a new expression.

In spirit, the System of Logical Graphs does not specify an interpretation for its connectives, i.e., does not assign them a fixed meaning in relation to logical notions like conjunction (*and*), disjunction (*or*), and so on, the idea being to find laws of logic that are invariant over variations in both operands and operators. In practice, however, it is convenient to choose either one of two common interpretations for the connectives, corresponding to what Peirce called the *Existential* and the *Entitative* versions of Logical Graphs.

We will immediately specialize to the Existential interpretation, which Peirce himself eventually came to favor, but mention this for those readers of Laws of Form who will find the Entitative alternative emphasized there. In the interpretation adopted here, we assign to the blank connective, or concatenation, the meaning of conjunction. This forces a number of other interpretive choices:

The blank term " " by itself has the value of true.

The operator (, , , ...) says "just one false" of its contents, that is to say, the expression (E1, E2, E3) has the value true just in case exactly one of the expressions E1, E2, E3 is false.

Table 2. The Existential Interpretation

Expression	Interpretation
` '	True.
()	False.
A	A.
(A)	Not A.
A B C	A and B and C.
((A)(B)(C))	A or B or C.
(A (B))	A implies B. If A then B.
(A , B)	A not equal to B. A exclusive-or B.
((A , B))	A equals B.
(A , B , C)	Just one of A, B, C is false.
((A),(B),(C))	Partition into A, B, C. Just one of A, B, C is true.
(X ,(A),(B),(C))	Partition X into A, B, C. Genus X of species A, B, C.

From these first attachments is developed the whole array of meanings for expressions, a sample of which is displayed in Table 2. The unfamiliar compound connectives that appear in the Figure will be explained later on, in the discussion of examples. In what follows, the general form of expression founded on Logical Graphs will be referred to as LG. The Existential interpretation of these expressions will be indicated as Ex.

Example 4. Molly's World **Files: molly.***

Example 4 is taken from the literature on computational learning theory, adapted with some changes from the example called "Molly's Problem" in the paper *Learning with Hints* by Dana Angluin [Ang]. We quote Angluin's motivational description to set up the problem:

> Imagine that you have become acquainted with an alien named Molly from the planet Ornot, who is currently employed in a day-care center. She is quite good at propositional logic, but a bit weak on knowledge of Earth. So you decide to formulate the beginnings of a propositional theory to help her label things in her immediate environment.

The purpose of this quaint pretext is, of course, to make sure the reader appreciates the constraints of the problem: that no extra savvy is fair, all facts must be presumed or deduced upon the immediate premises.

Our use of this example is not directly relevant to the purposes of the discussion from which it is borrowed, so we simply give our version of it without comment on those issues.

Here is our rendition of the initial knowledge base delimiting Molly's World:

File: molly.log

```
( object ,( toy ),( vehicle ))
(( small_size ),( medium_size ),( large_size ))
(( two_wheels ),( three_wheels ),( four_wheels ))
(( no_seat ),( one_seat ),( few_seats ),( many_seats ))
( object ,( scooter ),( bike ),( trike ),( car ),( bus ),( wagon ))
( two_wheels    no_seat            ,( scooter ))
( two_wheels    one_seat    pedals ,( bike ))
( three_wheels  one_seat    pedals ,( trike ))
( four_wheels   few_seats   doors  ,( car ))
( four_wheels   many_seats  doors  ,( bus ))
( four_wheels   no_seat     handle ,( wagon ))
( scooter           ( toy  small_size ))
( wagon             ( toy  small_size ))
( trike             ( toy  small_size ))
( bike  small_size  ( toy ))
( bike  medium_size ( vehicle ))
( bike  large_size  )
( car               ( vehicle  large_size ))
( bus               ( vehicle  large_size ))
( toy               ( object ))
( vehicle           ( object ))
```

All the forms used in the preceding log file will probably be familiar from earlier discussions. The purpose of one or two constructions may, however, be a little obscure.

> The rule "(bike large_size)" , for example, merely says that nothing can be both a bike and large_size.
>
> The rule "(three_wheels one_seat pedals , (trike))" says that anything with all the features of three_wheels, one_seat, and pedals is excluded from being anything but a trike. In short, anything with just those three features is equivalent to a trike. Recall that the form "(A , B)" may be interpreted to assert either the exclusive disjunction or the inequality of A and B.

The rules have been stated in this particular way simply to imitate the style of rules in the reference example.

This last point brings up an important issue, the question of *rhetorical* differences in expression and their potential impact on the *pragmatics* of computation. Unfortunately, we must abbreviate our discussion of this topic for now, and only mention the following facts.

Logically equivalent expressions, even though they must lead to logically equivalent normal forms, may have very different characteristics with regard to efficiency of processing.

Thus, all four of the forms

"((A , B))" "(A , (B))" "((A) , B)" "((B , A))"

are equally succinct ways of saying that A is logically equivalent to B, yet each can have different effects on the route that Model takes to arrive at its answer. Apparently, some equalities are more equal than others.

These effects occur partly because the algorithm chooses to make cases of variables on a basis of *leftmost shallowest first*, but their impact can be complicated by interactions each expression has with the context it occupies. The lesson is that it is probably best not to bother too much about these problems, but just experiment with different ways of expressing equivalent information until you get a sense of what works best in various situations.

We think you will be happy to see only the ultimate Sense of Molly's World, so here it is:

Sense outline
File: molly.sen

```
object
 two_wheels
  no_seat
   scooter
    toy
     small_size
  one_seat
   pedals
    bike
     small_size
      toy
     medium_size
      vehicle
 three_wheels
  one_seat
   pedals
    trike
     toy
      small_size
 four_wheels
  few_seats
   doors
    car
     vehicle
      large_size
  many_seats
   doors
    bus
     vehicle
      large_size
  no_seat
   handle
    wagon
     toy
      small_size
```

This outline is not the Sense of the unconstrained log file, but a result of running Model with a Query on the single feature “object”. Using this focus helps to make more relevant Sense of Molly’s World.

The logical paradigm from which this Example was derived is that of *propositional Horn clause* theories. These clauses are of three kinds:

<1> “ A and B and C and ... => Z ”
<2> “ Z ”
<3> “ not { A and B and C and ... } ”

where the proposition letters A, B, C, ... , Z are restricted to what we would call single positive features, not themselves negated or otherwise complex expressions.

For comparison, in the syntax of Ex these forms would look like:

```
<1>   ( A B C ... ( Z ))
<2>     Z
<3>   ( A B C ... )
```

The style of deduction in Horn clause logics is essentially proof-theoretic in character, with the burden of proof falling on implication (“=>”) and inference (*modus ponens* or *resolution*), Cf: [Llo, MaW].

In contrast, the method used here is substantially model-theoretic, the stress being to start from more general forms of expression for laying out the facts (e.g., distinctions, equations, partitions) and to work toward results that maintain logical equivalence with their origins.

What all of this has to do with the output above is this: From the perspective adopted here, almost any theory (e.g., one founded on the postulates of Molly’s World) will have far more models than the implicational and inferential mode of reasoning is designed to discover. We will be forced to confront them, however, if we try to run Model on a large set of implications.

The typical Horn clause interpreter gets around this difficulty only by a stratagem that takes clauses to mean something other than what they say, that is, by distorting the semantics. Our Model, on the other hand, has no such finesse.

This explains why it was necessary to impose the prerequisite “object” constraint on the log file for Molly’s world. It only supplied what we usually take for granted, in order to obtain a set of models we would normally think of as being the import of the definitions.

References

[Ang] Angluin, Dana
(1989) "Learning with Hints", in: *Proceedings of the 1988 Workshop on Computational Learning Theory*, ed: D. Haussler & L. Pitt, Morgan Kaufmann, San Mateo, CA.

[BaC] Ball, W.W. Rouse, & Coxeter, H.S.M.
(1987) *Mathematical Recreations and Essays*, 13th ed., Dover, New York, NY.

[Cha] Chang, Chin-Liang & Lee, Richard Char-Tung
(1973) *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, New York, NY.

[DDQ] Denning, Peter J., Dennis, Jack B., and Qualitz, Joseph E.
(1978) *Machines, Languages, and Computation*, Prentice-Hall, Englewood Cliffs, NJ.

[Ede] Edelman, Gerald M.
(1988) *Topobiology: An Introduction to Molecular Embryology*, Basic Books, New York, NY.

[Llo] Lloyd, J.W.
(1984) *Foundations of Logic Programming*, Springer-Verlag, Berlin.

[MaW] Maier, David & Warren, David S.
(1988) *Computing with Logic: Logic Programming with Prolog*, Benjamin/Cummings, Menlo Park, CA.

[McR] McClelland, James L. and Rumelhart, David E.
(1988) *Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises*, MIT Press, Cambridge, MA.

[P1] Peirce, Charles Sanders
(1931-1960) *Collected Papers of Charles Sanders Peirce*, ed: Charles Hartshorne, Paul Weiss, & Arthur W. Burks, Harvard University Press, Cambridge, MA.

[P2] (1976) *The New Elements of Mathematics*, ed: Carolyn Eisele, Mouton, The Hague.

[P3] (1966) *Charles S. Peirce: Selected Writings; Values in a Universe of Chance*, ed: Philip P. Wiener, Dover, New York, NY.

[SpB] Spencer Brown, George
(1969) *Laws of Form*, George Allen & Unwin, London, UK.

[VaH] Van Hentenryck, Pascal
(1989) *Constraint Satisfaction in Logic Programming*, MIT Press, Cambridge, MA.

[Wil] Wilf, Herbert S.
(1986) *Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, NJ.

[Win] Winston, Patrick Henry
(1984) *Artificial Intelligence*, 2nd ed., Addison-Wesley, Reading, MA.

[Wir] Wirth, Niklaus
(1976) *Algorithms + Data Structures = Programs*, Prentice-Hall, Englewood Cliffs, NJ.