MEDWIN PUBLISHERS
Committed to Create Value for Researchers

Philosophy International Journal
ISSN: 2641-9130

Using C-Tables to Teach Class Logic

Jason GJ*

Department of Philosophy, California State University, USA

***Corresponding author:** Gary James Jason, Department of Philosophy, California State University, USA, Email: drgaryjjason@gmail.com

Conceptual Article
Volume 9 Issue 2
Received Date: March 20, 2026
Published Date: April 02, 2026
DOI: 10.23880/phij-16000363

Abstract

In this article, I lay out a new type of logic graph that can replace Euler and Venn Diagrams. This new form of logic graphs I call "class-tables" (or "c-tables" for short). I show how c-tables can be used easily to assess standard syllogisms. I then show how c-tables can be used to directly assess class arguments involving named individuals, as well as to assess extended syllogisms, i.e., syllogisms of more than three classes. I end by illustrating how c-tables can also easily handle syllogisms with compound classes.

Keywords: Class Logic; C-Tables; Syllogisms; Venn Diagrams; Euler Diagrams

Introduction

A highly general view of the history of formal logic is that it falls into three periods: traditional logic (founded by Aristotle in the 4th century BCE), algebraic logic (founded by Boole in the first half of the 19th century), and symbolic logic (founded by Frege and Peirce in the late 19th century). Russell, Whitehead, and others gave First-order Logic its current form in the early 20th century. Class logic has been an important part of deductive logic throughout its history [1][1]. Class logic deals with arguments using properties (i.e., one-place or monadic predicates).

One notable achievement of algebraic and symbolic logicians is their development of logic graphs or diagrams. In the 18th century, mathematician Leonhard Euler (1707-1783) developed what appears to be the earliest form of logic graphs, called accordingly "Euler diagrams." In the 19th century, several logicians developed a variety of logic diagrams. Mathematician John Venn (1834-1923) developed the most widely used graphs, "Venn Diagrams." Author and mathematician Charles Dodgson (1832-1898) developed "Carroll diagrams" (derived from his pen-name Lewis Carroll). In the 20th century, the computer scientists Edward Veitch (1924-2013) and Maurice Karnaugh (1924-2022) devised other logic graphs (called respectively "Veitch diagrams" and "Karnaugh maps").

This article aims to show how a tabular method—somewhat similar to Karnaugh maps—can be used instead of Venn (and Euler) Diagrams as a tool in class logic[2]. I will call these **Class Tables** (hereafter "C-tables"). In effect, c-tables combine logic graphs with Boolean operators.

Class Statements and C-tables

Expressing class statements as C-tables is easy. Start with the layout of a C-table. A one-class C-table is a 2-by-2

---

[1] We can define class logic as the part of set theory useful for representing statements and evaluating arguments about groups expressed in ordinary natural language. Richard Purtill suggests this definition in Logic for Philosophers, NY: Harper & Row Publishers (1971), p. 123.

[2] The diagrams are simpler than Karnaugh maps, developed and used to simplify circuits in electronic engineering.

table with two cells or cells (Table 1).

A	−A
1	2

Table 1: One-class C-table showing a single class A and its complement −A, with cells representing A and non-A individuals.

Here, 'A' is any arbitrary (simple) class, and '−A' is its complement[3]. Cell 1 represents the things that are A, and cell 2 the things that are not A.

A two-class C-table is a 3-by-3 table with 4 cells (Table 2):

	A	−A
B	1	2
−B	3	4

Table 2: Two-class C-table for classes A and B, with four cells corresponding to all possible combinations of A/−A and B/−B

Here, cell 1 represents the things that are both A and B, cell 2 the things that are not A but are B, cell 3 the things that are A but not B, and cell 4 the things that are neither A nor B.

A three-class C-table is a 5-by-3 table with 8 cells (Table 3):

	A ∩ B	A ∩ −B	−A ∩ B	−A ∩ −B
C	1	2	3	4
−C	5	6	7	8

Table 3: Three-class C-table for classes A, B, and C, with eight cells representing all conjunctions of A/−A, B/−B, and C/−C.

Cell 1 represents the class of things that are A and B and C, i.e., (A ∩ B) ∩ C. Cell 2 represents the class of things that are A and not B and C, i.e., (A ∩ −B) ∩ C. The other cells are easy to identify.

Now, here we see one advantage of using C-tables. It is hard to construct Venn diagrams for more than three classes because one has to overlap ovals, making distinguishing cells difficult. However, with C-tables, we can increase the number of classes in a straightforward way. For example, for four classes or terms, we need a 5x4 table (Table 4):

	A ∩ B	A ∩ −B	−A ∩ B	−A ∩ −B
C ∩ D	1	2	3	4
C ∩ −D	5	6	7	8
−C ∩ D	9	10	11	12
−C∩−D	13	14	15	16

Table 4: Four-class C-table for A, B, C, and D, illustrating how the tabular method extends systematically to additional classes.

It is again easy to identify the cells—cell 11, for example, is the class (−A ∩ B) ∩ (−C ∩ D). A five-term C-table requires a 9-by-4 table, a six-term C-table requires a 9-by-9 table, a seven-term C-table a 17-by-9, and so on.

Moving next to statements, we can represent singular statements (i.e., statements with named individuals. We can represent the statement 'x has a property A' (or 'x is a member of class A') as Table 5.

A	−A
x	

Table 5: C-table representation of the singular statement "x is A," locating a named individual x in the A cell.

The statement that individual x is not A is represented by Table 6.

A	−A
	x

Table 6: C-table representation of the singular statement "x is not A," placing x in the −A cell.

We can express general statements (i.e., universal and particular ones) by using the null symbol '∅' and the existence symbol '√.' We can represent the statement that everything is A by Table 7.

A	−A
	∅

Table 7: C-table for the universal statement "Everything is A," using the null symbol ∅ in the −A cell.

We can represent the statement that nothing is A as the C-table (Table 8):

A	−A
∅	

Table 8: C-table for the universal negative "Nothing is A," marking A as null with ∅.

---

[3] I treat A, B, C, … as meta-linguistic variables ranging over classes.

We can represent the statement that something is A by the C-table (Table 9):

A	−A
√	

Table 9: C-table for the particular affirmative “Something is A,” using the existence symbol √ in the A cell.

And the statement that something is not A by the C-table (Table 10):

A	−A
	√

Table 10: C-table for the particular negative “Something is not A,” with √ in the −A cell.

The extension is easy for two-class statements with named individuals. We can represent the statement that individual x is both A and B, as in Table 11.

	A	−A
B	x	
−B		

Table 11: Two-class C-table showing the singular statement “x is both A and B” by placing x in the A∩B cell.

The representations of the statements that x is A but not B, not A but B, and neither A nor B are obvious.

The null and existence symbols readily express the four transitional categorical statements. For example, we can represent the statement that all A’s are B’s in Table 12.

	A	−A
B		
−B	∅	

Table 12: Two-class C-table for “All A are B,” nulling the A∩−B cell with ∅.

We can express the statement that some A’s are B’s, as shown in Table 13.

	A	−A
B	√	
−B		

Table 13: Two-class C-table for “Some A are B,” placing √ in the A∩B cell.

We can express the statement that no A’s are B’s as Table 14.

	A	−A
B	∅	
−B		

Table 14: Two-class C-table for “No A are B,” marking the A∩B cell as null.

Finally, we can express the statement that some A’s are not B’s as Table 15.

	A	−A
B		
−B	√	

Table 15: Two-class C-table for “Some A are not B,” with √ in the A∩−B cell.

Assessing Categorical Syllogisms

Moving next to assessing categorical syllogisms, again, the job is straightforward, using the null and existence symbols and (as usual) symbolizing universal before particular premises.

Example:
1. All A are B
2. All B are C /∴ All A are C

First, we set up the C-table (Table 16).

	A ∩ B	A ∩ −B	−A ∩ B	−A ∩ −B
C				
−C				

Table 16: Initial three-class C-table layout for testing the syllogism “All A are B; All B are C; therefore, All A are C”.

Second, since both the premises are universal, we can diagram them in any order (Table 17).

	A ∩ B	A ∩ −B	−A ∩ B	−A ∩ −B
C		∅		
−C	∅	∅	∅	

Table 17: C-table with universal premises diagrammed, showing that “All A are B; All B are C” validates the conclusion “All A are C”.

Does this capture the conclusion? Yes, because the conclusion requires the cells (A ∩ B) ∩ −C and (A ∩ −B) ∩ −C to be null, and they are. So the argument is valid.

Example:

1. No A are B
2. No C are A /∴ No C are B

Set up the C-table and diagram the premises (Table 18).

	A ∩ B	A ∩ −B	−A ∩ B	−A ∩ −B
C	∅	∅		
−C	∅			

Table 18: C-table for “No A are B; No C are A; therefore, No C are B,” illustrating why the argument fails to force the conclusion.

Does this table capture the conclusion? No, because to represent it, both the cells (A ∩ B) ∩ C and (−A ∩ B) ∩ C need to be null, but the latter cell isn’t. I have my students show what is missing by putting it in brackets (Table 19).

	A ∩ B	A ∩ −B	−A ∩ B	−A ∩ −B
C	∅	∅	[∅]	
−C	∅			

Table 19: C-table with bracketed null cell added to indicate what is missing to validate “No C are B” in the previous example.

So, the argument is shown to be invalid.

Example:

1. Some A are B
2. All B are C /∴ Some A are C

	A ∩ B	A ∩ −B	−A ∩ B	−A ∩ −B
C	√			
−C	∅		∅	

Table 20: C-table for “Some A are B; All B are C; therefore, Some A are C,” showing a valid particular–universal syllogism.

The argument is shown to be valid. To capture the conclusion, a checkmark must be in cell (A ∩ B) ∩ C, and there is.

Now, one of the trickier skills the student needs to learn in doing any logic graph for testing syllogisms is representing a particular claim when diagramming the universal premises (if any) does not force it into a specific cell. In Venn diagrams, the student does this by placing the check mark on a boundary line—which can be awkward and hard to read. With C-tables, it is fairly easy.

Example:

1. Some A are C
2. Some not B are not C /∴ Some A are not B

Since both premises are particular, we can do them in any order. Let’s start with premise 1, shown in Table 21.

	A ∩ B	A ∩ −B	−A ∩ B	−A ∩ −B
C	√1	√1		
−C				

Table 21: Use of numbered existence marks in a C-table to represent “Some A are C” without fixing the location in B/−B cells.

The student should read the numbered checkmarks: “There is at least one individual in the cells marked by √1, but we don’t know in which one or ones.” That is, we know that there is at least one thing in A and C, but we don’t know whether it is in B or not. Similarly, diagramming the second premise yields Table 22.

	A ∩ B	A ∩ −B	−A ∩ B	−A ∩ −B
C	√1	√1		
−C		√2		√2

Table 22: Combined C-table for “Some A are C; Some not B are not C; therefore, Some A are not B,” demonstrating why the conclusion is not forced.

Does this capture the conclusion? No, because that would require check marks *with the same number* in the cells (A ∩ −B) ∩ C and (A ∩ −B) ∩ −C, and we don’t have that. For all the premises tell us, both of those cells might be empty. We signal this with check marks marked with an index ‘i’ indicating that they have to have the *same* number and again bracketed to show what is needed to capture the conclusion, shown in Table 23.

	A ∩ B	A ∩ −B	−A ∩ B	−A ∩ −B
C	√1	√1 [√i]		
−C		√2 [√i]		√2

Table 23: C-table with indexed bracketed checkmarks indicating what would be required to validate “Some A are not B” in the previous example.

Example:

1. No A is C
2. Some not B are not C /∴ Some A are not B

This is invalid, as Table 24 shows.

	A ∩ B	A ∩ −B	−A ∩ B	−A ∩ −B
C	∅	∅		
−C		√1 [√]		√1

Table 24: C-table for "No A is C; Some not B are not C; therefore, Some A are not B," exhibiting the invalidity via qualified existence marks.

The premises require a checkmark in the cell of (A ∩ −B) ∩ −C (since the cell (A ∩ −B) ∩ C is null). And all we have is '√1'—which signals that it could, in fact, be empty. Again, we represent this with brackets.

One last example:

1. Some A are B.
2. Some A are not C. /∴ Some B are not C.

Table 25 shows that this argument is invalid.

	A ∩ B	A ∩ −B	−A ∩ B	−A ∩ −B
C	√1			
−C	√1 √2 [√i]	√2	[√i]	

Table 25: C-table for "Some A are B; Some A are not C; therefore, Some B are not C," illustrating another invalid pattern.

C-Tables with Named Individuals

For those of us who teach class logic to provide students with an intuitive basis for monadic predicate logic, it is useful to consider arguments with named individuals. One way to do this is to treat a named individual as a class, as some texts do [2][4]. For example, we could view 'Aristotle' as the class of things that are identical to Aristotle, but that is hardly intuitive to many students.

We can handle such arguments with C-tables easily. Example:

1. All A are B
2. x is A /∴ x is B

Table 26 shows that this is a valid argument.

	A	−A
B	x	
−B	∅	

Table 26: C-table with named individual x showing the valid argument "All A are B; x is A; therefore, x is B".

Again, we diagram universal premises before particular ones, including particular statements with named individuals.

Example:

Some A are B
x is A /∴ x is B

Table 27 shows that this argument is valid.

	A	−A
B	√ x_1	[x_i]
−B	x_1	

Table 27: C-table with x for "Some A are B; x is A; therefore, x is B," using indexed markers to show the argument's invalidity.

Again, by using x_1 we signal that we know that x is A, but we don't know whether it is B or not. However, the conclusion requires that there be an indication that x is B, and we don't know whether it is A or not.

Example:

1. x is A.
2. All A are B.
3. All B are C. /∴ x is C.

Table 28 shows that the argument is valid.

	A ∩ B	A ∩ −B	−A ∩ B	−A ∩ −B
C	x	∅		
−C	∅	∅	∅	

Table 28: Three-class C-table with x for the valid argument "x is A; All A are B; All B are C; therefore, x is C".

Slightly more complicated:

1. x is A.
2. All A are B.
3. Some B are C. /∴ x is C.

Start by symbolizing the premises, the universal one symbolized first (Table 29).

	A ∩ B	A ∩ −B	−A ∩ B	−A ∩ −B
C	√1 x_1	∅	√1	
−C	x_1	∅		

Table 29: C-table setup for "x is A; All A are B; Some B are C; therefore, x is C," showing the distribution of x and √ marks.

We use 'x_i' since premise 1 tells us that x is A, but not whether it is C or not. The argument is invalid since, to capture the conclusion, 'x_1' must also be in the cells (−A ∩ B) ∩ C and (−A ∩ −B) ∩ C. We indicate this with brackets in Table 30.

---

[4] See, for example, Stan Baronett, *Logic (2nd ed.)*, Oxford: Oxford University Press (2013), pp. 203-204.

	A ∩ B	A ∩ −B	−A ∩ B	−A ∩ −B
C	√1 x_1	∅	√1 [x_1]	[x_1]
−C	x_1	∅		

Table 30: C-table with bracketed positions of x indicating what would be needed to make "x is C" follow in the previous argument.

Extended Syllogisms

Another advantage of using C-tables is that we can use them to check extended syllogisms mechanically. Example:

1. All A are B.
2. All B are C.
3. Some A are D. /∴ Some C are D.

Table 31 shows that representing the premises also captures the conclusion, showing that the argument is valid.

	A ∩ B	A ∩ −B	−A ∩ B	−A ∩ −B
C ∩ D	√	∅		
C ∩ −D		∅		
−C ∩ D	∅	∅	∅	
−C ∩ −D	∅	∅	∅	

Table 31: Four-term C-table for "All A are B; All B are C; Some A are D; therefore, Some C are D," showing a valid extended syllogism.

A more interesting example:

1. All A are B.
2. No B are C.
3. Some C are D. /∴ Some A are D.

Table 32 shows that this argument is invalid.

	A ∩ B	A ∩ −B	−A ∩ B	−A ∩ −B
C ∩ D	∅	∅	∅	√
C ∩ −D	∅	∅	∅	
−C ∩ D	[√]	∅		
−C ∩ −D		∅		

Table 32: Four-term C-table for "All A are B; No B are C; Some C are D; therefore, Some A are D," demonstrating invalidity via possible models.

We can extend this further. Example:

1. All A are B.
2. All B are C.
3. No C are D.
4. All D are E. /∴ No A are E.

This argument is invalid, as shown by Table 33.

	(A ∩ B) ∩ C	(A ∩ B) ∩ −C	(A ∩ −B) ∩ C	(A ∩ −B) ∩ −C	(−A ∩ B) ∩ C	(−A ∩ B) ∩ −C	(−A ∩ −B) ∩ C	(−A ∩ −B) ∩ −C
D ∩ E	∅	∅	∅	∅	∅	∅	∅	
D ∩− E	∅	∅	∅	∅	∅	∅	∅	∅
−D ∩ E	[∅]	∅	∅	∅		∅		
−D ∩ −E		∅	∅	∅		∅		

Table 33: Five-term C-table for "All A are B; All B are C; No C are D; All D are E; therefore, No A are E," illustrating an invalid chain of universals.

A more complicated example:

1. All are B.
2. Some B are C.
3. No C are D.
4. Some D are not E. /∴ Some A are not E.

Table 34 shows that this argument is invalid.

	(A ∩ B) ∩ C	(A ∩ B) ∩ −C	(A ∩ −B) ∩ C	(A ∩ −B) ∩ −C	(−A ∩ B) ∩ C	(−A ∩ B) ∩ −C	(−A ∩ −B) ∩ C	(−A ∩ −B) ∩ −C
D ∩ E	∅		∅	∅	∅		∅	
D ∩−E	∅	√2 [√i]	∅	∅	∅	√2	∅	√2
−D ∩ E	√1		∅	∅	√1			
−D ∩ −E	√1[√i]	[√i]	∅	∅	√1			

Table 34: Five-term C-table for "All A are B; Some B are C; No C are D; Some D are not E; therefore, Some A are not E," showing failure to capture the conclusion.

Finally, an example with six terms:

1. No A are B.
2. No B are C.
3. No C are D.
4. No D are E.
5. Some E are F. /∴ Some A are not F.

Table 35 shows that this argument is invalid.

	(A ∩ B) ∩ C	(A ∩ B) ∩ −C	(A ∩ −B) ∩ C	(A ∩ −B) ∩ −C	(−A ∩ B) ∩ C	(−A ∩ B) ∩ −C	(−A ∩ −B) ∩ C	(−A ∩ −B) ∩ −C
(D ∩ E) ∩ F	∅	∅	∅	∅	∅	∅	∅	∅
(D ∩ E) ∩ −F	∅	∅	∅	∅	∅	∅	∅	∅
(D ∩− E) ∩ F	∅	∅	∅		∅		∅	
(D ∩ −E) ∩ −F	∅	∅	∅	[√i]	∅		∅	
(−D ∩ E) ∩ F	∅	∅	√1	√1	∅	√1	√1	√1
(−D ∩ E) ∩ −F	∅	∅	[√i]	[√i]	∅			
(−D ∩−E) ∩ F	∅	∅			∅			
(−D ∩−E) ∩ −F	∅	∅	[√i]	[√i]	∅			

Table 35: Six-term C-table for "No A are B; No B are C; No C are D; No D are E; Some E are F; therefore, Some A are not F," exhibiting invalidity with many terms.

We can also examine extended syllogisms with named individuals. Example:

1. x is A.
2. Some A are B.
3. All B are C.
4. No C are D. /∴ x is not D.

Table 36 shows that this argument is invalid.

	A ∩ B	A ∩ −B	−A ∩ B	−A ∩ −B
C ∩ D	∅	∅	∅	∅
C ∩ −D	√ x_1	x_1	[x_1]	[x_1]
−C ∩ D	∅	x_1	∅	
−C ∩ −D	∅	x_1	∅	[x_1]

Table 36: Four-term C-table with x for "x is A; Some A are B; All B are C; No C are D; therefore, x is not D," showing that the conclusion does not follow.

Syllogisms with Compound Classes

Some texts extend the power of syllogistic by using "compound classes." A **compound class** is a class defined by two properties conjoined. We can represent this by concatenating the property constants. For example, we might express "All friendly dogs are happy" as "All FD are A."

We can use expanded C-tables to express statements with compound classes. For example, "All AB are C" can be represented by Table 37.

	A ∩ B	A ∩ −B	−A ∩ B	−A ∩ −B
C				
−C	∅			

Table 37: Expanded C-table representing the statement "All AB are C" by treating A and B separately within a three-class layout.

However, the table becomes simpler if we treat 'AB' as if it were an atom (Table 38).

	AB	−(AB)
C		
−C	∅	

Table 38: Simplified C-table for "All AB are C," treating the compound class AB as a single atomic class.

It becomes simple because we are interested only in two classes: A ∩ B and −(A ∩ B), that is, individuals with both A and B and those without both A and B.

Syllogisms using categorical statements with compound classes are easily assessed with C-tables. Example:

1. All AB are C.
2. No C are D. /∴ No AB are D.

Table 39 shows that this argument is valid.

	(AB) ∩ C	(AB) ∩ −C	−(AB) ∩ C	−(AB) ∩ −C
D	∅	∅	∅	
−D		∅		

Table 39: C-table with compound subject showing that "All AB are C; No C are D; therefore, No AB are D" is valid.

Example:

1. All AB are C.
2. Some C are D. /∴ Some AB are D.

Table 40 shows that this argument is invalid.

	(AB) ∩ C	(AB) ∩ −C	−(AB) ∩ C	−(AB) ∩ −C
D	√1 [√]	∅	√1	
−D		∅		

Table 40: C-table with compound subject for “All AB are C; Some C are D; therefore, Some AB are D,” demonstrating invalidity.

The fact that the cell [(AB) ∩ −C] ∩ D is null means that to capture the conclusion, we need a checkmark squarely in the cell [(AB) ∩ C] ∩ D. But all we have from premise 2 is a qualified checkmark indicating that something is either in that cell or another (or both).

We can extend this approach to handle syllogisms with categorical statements with compound subjects and predicates. Example:

1. No AB are CD.
2. No CD are EF. /∴ No EF are AB.

Table 41 shows that the argument is invalid.

	(AB) ∩ (CD)	(AB) ∩ −(CD)	−(AB) ∩ (CD)	−(AB) ∩ −(CD)
EF	∅	[∅]	∅	
−(EF)	∅			

Table 41: C-table with compound subjects and predicates for “No AB are CD; No CD are EF; therefore, No EF are AB,” revealing the argument’s invalidity.

And, of course, we can have extended syllogisms with compound classes. Example:

1. All AB are CD.
2. No CD are EF.
3. Some EF are GH.
4. No GH are IJ. /∴ Some IJ are not AB.

Table 42 shows that this argument is invalid.

	[(A B) ∩ (CD)] ∩ (EF)	[(A B) ∩ (CD)] ∩ −(EF)	[(A B) ∩ −(CD)] ∩ (EF)	[(A B) ∩ −(CD)] ∩ −(EF)	[−(A B) ∩ (CD)] ∩ (EF)	[−(A B) ∩ (CD)] ∩ −(EF)	[−(A B) ∩ −(CD)] ∩ (EF)	[−(A B) ∩ −(CD)] ∩ −(EF)
(GH) ∩ (IJ)	∅	∅	∅	∅	∅	∅	∅	∅
(GH) ∩ −(IJ)	∅		∅	∅	∅		√	
−(GH) ∩ (IJ)	∅		∅	∅	∅	[√i]	[√i]	[√i]
−(GH) ∩ −(IJ)	∅		∅	∅	∅			

Table 42: Extended C-table with multiple compound classes illustrating the invalid argument “All AB are CD; No CD are EF; Some EF are GH; No GH are IJ; therefore, Some IJ are not AB”.

Conclusion

To summarize, using C-tables instead of Venn (or Euler) diagrams has several advantages. First, they are easy to construct. They require no talent for drawing circles on the board or special templates for word-processing the diagrams.

More importantly, C-tables more easily extend logical assessment to syllogisms of more than three terms. You can represent an extended syllogism with six terms in a C-table more easily than in a Venn diagram. Of course, C-tables (like truth tables, Venn diagrams, Euler diagrams, Carroll diagrams, and Karnaugh maps) are mechanical (i.e., effective) methods of deciding validity. So, they grow exponentially in complexity with adding each new atom. A C-table for an argument with 10 terms would require 1,024 cells. One had better be able to write in a very tiny print. However, syllogisms with (say) six terms are easy to construct, especially when compared with Venn diagrams.

Finally, C-tables easily handle arguments with named individuals and compound terms.

References

1. Richard P (1971) Logic for Philosophers. Harper & Row Publishers, New York, USA.

2. Stan B (2013) Logic. 2nd (Edn.), Oxford University Press, Oxford, USA.