De Morgan’s Laws Revisited: To Be AND/OR NOT To Be
Raoul A. Bernal, Amgen, Inc., Thousand Oaks, CA

ABSTRACT
De Morgan's Laws, named for the nineteenth century British mathematician and logician Augustus De Morgan (1806-1871), are powerful rules of Boolean algebra and set theory that relate the three basic set operations (union, intersection and complement) to each other.

If A and B are subsets of a universal set U, de Morgan's laws state that

\( (A \cup B)' = A' \cap B' \)

\( (A \cap B)' = A' \cup B' \)

where \( \cup \) denotes the union (OR), \( \cap \) denotes the intersection (AND) and \( A' \) denotes the set complement (NOT) of \( A \) in \( U \), i.e., \( A' = U \setminus A \). The first law simply states that an element not in \( A \cup B \) is not in \( A' \) and not in \( B' \). Conversely, it also states that an element not in \( A' \) and not in \( B' \) is not in \( A \cup B \). The second law simply states that an element not in \( A \cap B \) is not in \( A' \) or not in \( B' \). Conversely, it also states that an element not in \( A' \) or not in \( B' \) is not in \( A \cap B \).

This paper will demonstrate how the de Morgan's Laws can be used to simplify complicated Boolean IF and WHERE expressions in SAS code. Using a specific example, the correctness of the simplified SAS code is verified using direct proof and tautology table. An actual SAS example with simple clinical data will be executed to show the equivalence and correctness of the results.

INTRODUCTION
In general, for any collection of subsets, de Morgan’s Laws are as follows:

Theorem. Let \( U \) be a set of subsets \( A_i \subset U \), \( i \in I \), where \( I \) is an index set which could be countable, uncountable or finite. Then,

\( (\bigcap_{i \in I} A_i)' = \bigcup_{i \in I} (A_i)' \)

i.e., the complement of the intersection of any number of sets equals the union of their complements.

\( (\bigcup_{i \in I} A_i)' = \bigcap_{i \in I} (A_i)' \)

i.e., the complement of the union of any number of sets equals the intersection of their complements.

These laws are tautologies in and of themselves, and are generally proved using truth tables. They can be proven simply by using Venn diagrams. These results, which prove that the intersection and union of sets are dual under complementation, are used extensively in digital circuit design for manipulating and simplifying the types of logic gates used. However, these two results are not well known among SAS users for simplifying IF and WHERE conditions in SAS code. The following example using a SAS dataset will demonstrate how these laws can be used to simplify conditions in clinical SAS programs.

EXAMPLE
Simplifying a SAS condition statement

It is not obvious that the SAS statement

\( (1) \) where not (A or B) and not (B and C);

reduces to

\( (2) \) where not (A or B);
Proof using set notation and de Morgan's laws:

\[(A \cup B) ' \cap (B \cap C) ' = (A' \cap B' ) \cup (B' \cup C ')
\]

\[= (A' \cap B' ) \cup (A' \cap B' \cap C') \quad \text{applying DM Laws}
\]

\[= (A' \cap B') \cup (A' \cap B' \cap C') \quad \text{applying distributive property}
\]

\[= (A' \cap B') \cup (A' \cap B' \cap C') \quad \text{applying associative property}
\]

\[= (A' \cap B') \cup (A' \cap B' \cap C') \quad \text{simplifying (B' \cap B')}
\]

\[= (A' \cap B') \cup (A' \cap B' \cap C') \quad \text{applying DM Law # 2}
\]

\[= (A' \cap B') \cup (U) ' \quad \text{complement (U) = \{null set\}}
\]

\[= (A' \cap B') \quad \text{trivial}
\]

\[= (A' \cap B') \quad \text{DM Law # 2}
\]

\[QED\]

Proof using tautology table:

<table>
<thead>
<tr>
<th></th>
<th>(1)</th>
<th>(2)</th>
<th>(3)</th>
<th>(4)</th>
<th>(5)</th>
<th>(6)</th>
<th>(7)</th>
<th>(8)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>A or B</td>
<td>not (A or B)</td>
<td>B and C</td>
<td>not (B and C)</td>
<td>not (A or B) and not (B and C)</td>
</tr>
<tr>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>FALSE</td>
<td>FALSE</td>
</tr>
<tr>
<td>TRUE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>TRUE</td>
<td>FALSE</td>
</tr>
<tr>
<td>TRUE</td>
<td>FALSE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>FALSE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>FALSE</td>
</tr>
<tr>
<td>FALSE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>FALSE</td>
<td>FALSE</td>
</tr>
<tr>
<td>FALSE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>FALSE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>FALSE</td>
</tr>
<tr>
<td>FALSE</td>
<td>FALSE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
<tr>
<td>FALSE</td>
<td>FALSE</td>
<td>FALSE</td>
<td>FALSE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

It is clear that the equivalent results in columns (5) and (8) of the tautology table validate the earlier proof.

Application using a SAS data example:

The following is a made-up, fictitious patient profiles data:

<table>
<thead>
<tr>
<th>Patient ID</th>
<th>Anemia</th>
<th>Blood_Transfu</th>
<th>Caucasian</th>
<th>Race</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>White</td>
</tr>
<tr>
<td>BB</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Black</td>
</tr>
<tr>
<td>CC</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>White</td>
</tr>
<tr>
<td>DD</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Asian</td>
</tr>
<tr>
<td>EE</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>White</td>
</tr>
<tr>
<td>FF</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Japanese</td>
</tr>
<tr>
<td>GG</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>White</td>
</tr>
<tr>
<td>HH</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Black</td>
</tr>
</tbody>
</table>

where the following variables are collected as follows: Anemia is flag indicator for Anemia in patient (1=Yes, 0=No), Blood_Transfu is indicator for any blood transfusion received and Caucasian is flag indicator for white vs. non-white race category.

Substituting codes A for Anemic, B for Blood Transfusion and C for Caucasian, we execute the first half of the WHERE condition in SAS statement (1) above - which is actually SAS statement (2):
SAS Program 1

```sas
title "Not (Anemic or Blood Transfusion Recipient)";
proc print;
  where not(Anemia or Blood_Transfu);
run;
```

which yields

SAS Listing 1

**Not (Anemic or Blood Transfusion Recipient)**

<table>
<thead>
<tr>
<th>Patient</th>
<th>Anemia</th>
<th>Blood_Transfu</th>
<th>Caucasian</th>
<th>Race</th>
</tr>
</thead>
<tbody>
<tr>
<td>GG</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>White</td>
</tr>
<tr>
<td>HH</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Black</td>
</tr>
</tbody>
</table>

Executing separately the second NOT condition from the same SAS statement (1) using

SAS Program 2

```sas
title "Not(Blood Transfusion Recipient and Caucasian)";
proc print;
  where not(Blood_Transfu and Caucasian);
run;
```

yields

SAS Listing 2

**Not (Blood Transfusion Recipient and Caucasian)**

<table>
<thead>
<tr>
<th>Patient</th>
<th>Anemia</th>
<th>Blood_Transfu</th>
<th>Caucasian</th>
<th>Race</th>
</tr>
</thead>
<tbody>
<tr>
<td>BB</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Black</td>
</tr>
<tr>
<td>CC</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>White</td>
</tr>
<tr>
<td>DD</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Asian</td>
</tr>
<tr>
<td>FF</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Japanese</td>
</tr>
<tr>
<td>GG</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>White</td>
</tr>
<tr>
<td>HH</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Black</td>
</tr>
</tbody>
</table>

It is evident from the above two SAS listings that patients GG and HH are the only records in common.

Thus, executing

SAS Program 3

```sas
title "Not (Anemic or Blood Transfusion Recipient) and not (Blood Transfusion Recipient and Caucasian)";
proc print;
  where not(Anemia or Blood_Transfu) and not(Blood_Transfu and Caucasian);
run;
```
yields the following listing

**SAS Listing 3**

<table>
<thead>
<tr>
<th>Patient</th>
<th>Anemia</th>
<th>Blood Transfu</th>
<th>Caucasian</th>
<th>Race</th>
</tr>
</thead>
<tbody>
<tr>
<td>GG</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>White</td>
</tr>
<tr>
<td>HH</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Black</td>
</tr>
</tbody>
</table>

which corresponds to SAS Listing 1 above.

**CONCLUSION**

De Morgan’s Laws are simple, yet powerful tools that can be applied in day-to-day programming - especially when the program has to parse multiple complicated conditions. It should benefit the casual clinical SAS programmer to keep these simple and straightforward laws in mind, to simplify conditions and make the SAS code more efficient. These laws are obviously very helpful in complicated code such as SAS/SQL, where execution time is definitely compounded by the complexity of the subsetting conditions used.

**REFERENCES**


**ACKNOWLEDGMENTS**

I would like to thank my manager Steve Yao and Amgen for the opportunity to present this paper at PharmaSUG 2005.

**CONTACT INFORMATION**

Your comments and questions are valued and encouraged. Please contact the author at:

Raoul Bernal  
Amgen, Inc.  
1 Amgen Center Drive MS 24-2-A  
Thousand Oaks, CA 91320  
Work Phone: 805-447-6649  
E-mail Address: rbernal@amgen.com

**TRADEMARKS**

SAS and all other SAS Institute Inc. product or service names are registered trademarks of SAS Institute Inc in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective countries.