Some Notes on Computer Generation of Counterexamples in Topology

by

A. Michael Ballantyne

ATP 24 February 1977

ATP-24

Some Notes on Computer Generation of Counterexamples in Topology

1975

A. Michael Ballantyne

In [1] and [2], efforts at producing a topology theorem prover are described. In [2] we describe a method of representing some topological theorems which seems to have enough power of expression to be of real value to a mechanical topologist. With some reflection it also appears that this representation can be used as the basis for a program which finds finite (finite topological spaces) counterexamples to some statements which are not theorems.

Almost all the work in computational logic has been geared toward the development of effective proof procedures for the computer. Hopefully when such programs are developed, they will afford the working mathematician a mechanical helper. When considering the activities of the research mathematician, it becomes apparent that some routine which can refute false conjectures, can in many ways be a more useful aid than some program which can only verify valid conjectures (see [3] in which Bledsoe argues the need for research in automating the hunt for counterexamples).

When a student attacks a homework exercise from a textbook, he usually is assured that what he is trying to prove is, in fact, valid. When he is forced to step our of this rather narrow circle of confidence, he must take a different attack at the problem. Now, instead of driving for a proof, he must in addition, look rather carefully for a counterexample. What follows are some observations on what such a search entails.

First, let me comment briefly on Resolution. What Resolution amounts to is a search for a counterexample. If the algorithm produces $\square$ then we know that no counterexample is to be had. If, on the other hand, the program grinds forever then we have a counterexample. If the program halts in some finite time without producing $\square$, then we also have a counterexample. A resolution program will

usually terminate without producing $\square$ only when the Herbrand universe is finite. In this case the Herbrand universe with the correct interpretation is a counterexample. Most interesting statements have an infinite Herbrand universe. One could describe Resolution as a brute force enumerative approach to finding a counterexample. In passing we mention that Roach and Siklossy [4] have implemented a program which refutes conjectures which arise in simulated robot tasks. This program does not explicitly construct a counterexample, but instead disproves a conjecture by showing its absence from a finite enumeration of the consequences of the hypotheses.

In general, mathematicians are quite a bit more clever than either of these two approaches. Effective human problem solvers encode the problem well and bring a lot of knowledge to bear. From now on we will be using very elementary point set topology as a basis for our discussions.

Suppose we had the following definition:

Definition. Let (X,τ) be a topology. Then a set $A \subset X$ is said to be regularly open if $\overline{A^0} = A$.

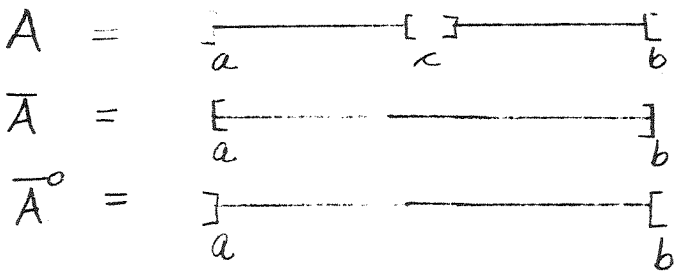
In other words, a point set is regularly open if when we take its closure and then from that take its interior, we get A back.[1] Clearly any regularly open set is open. The natural question to ask is whether the terms open and regularly open are equivalent. More exactly, is

(I) $\quad \forall A[OPEN(A) \longrightarrow REGOPEN(A)]$.

One's first inclination is to look to the standard topology E^1 (the real numbers with the usual topology), and within this context to look at the most familiar open set, the open interval $]a,b[$ for some $a, b \in \mathbb{R}$. Unfortunately

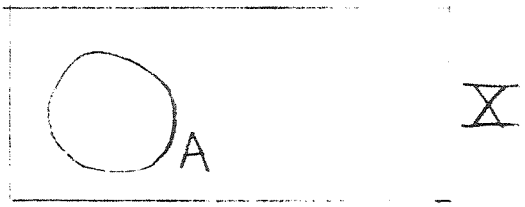
[1] For any set A, A^0 denotes the interior of A (the largest open subset of A), $\overline{A}$ denotes the closure of A, and A' denotes the complement of A, $X-A$.

we have $\overline{]a,b[}^{0} = [a,b]^{0} =]a,b[$. That is, when we close $]a,b[$ we pick up the points a,b but then promptly lose them when we take the interior. So we need to be a little more clever. We need to find an open set that picks up some other set of points when we take its closure and doesn't let go of all those points when we apply the interior operator. After some fiddling the natural choice of a set A is made by taking an open interval $]a,b[$ and deleting from it some point c to get the two intervals $]a,c[$ and $]c,b[$. Now when we close up A we pick up the points a,b,c and when we take the interior we drop a,b and keep c. Diagramatically we have

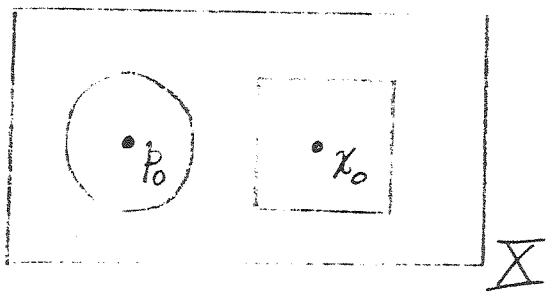


In a certain sense we could have picked a simpler $A^{*} \subseteq \mathbb{R}$, namely $\mathbb{R} \cap \{x_0\}'$ for any $x_0 \in \mathbb{R}$. This set A^{*} is a simpler counterexample in the sense that its closure picks up one point and loses none when we then take the interior. Although both choices A and A' are perfectly satisfactory, the second is preferable for the reason that it is the meagerest counterexample obtainable, in the sense mentioned above, once we have fixed our topological space to be E'.

Actually E' is a much more complicated topology than we need. If we abstract from our second counterexample, what we really have is a set A which is not all of X but whose closure is X. Since X is both open and closed this assures us that $X \neq A$. Diagramatically we have



In our second counterexample we knew that A^* was not equal to X since we put x_0 in $A^{*\prime}$. In our abstracted counterexample we will do the same thing -- that is, we will start off with some arbitrary open set A and put an arbitrary point, call it x_0, in A'. Since we don't really care what A is, we will assign it the simplest value possible. Since $A = \phi$ will not do, we give A the value $\{p_0\}$ for some point p_0. These assignments are consistent with the axioms of topology, hence we have created for a counterexample, the following small topology and an assignment to A within that topology.



$$X = \{x_0, p_0\}$$

$$\tau = [\{x_0, p_0\}, \{p_0\}]$$

We denote open sets with a circle and closed sets with a rectangle. Note that the set $\{p_0\}$ had to be an open set since from our statement (I) we know that A must be open.

Counterexample number 3's structural simplicity and apparent ease of construction would offhand indicate that we could write a program which would have to know only the rudiments of topology to be able to construct such simple counterexamples.

In what follows we will assume that the reader knows how the program

described in [2] attempts to prove a theorem. Essentially that program (let's call it GRAPHER) uses the statement of a theorem to draw a picture from which it tries to prove the theorem. If GRAPHER were trying to prove

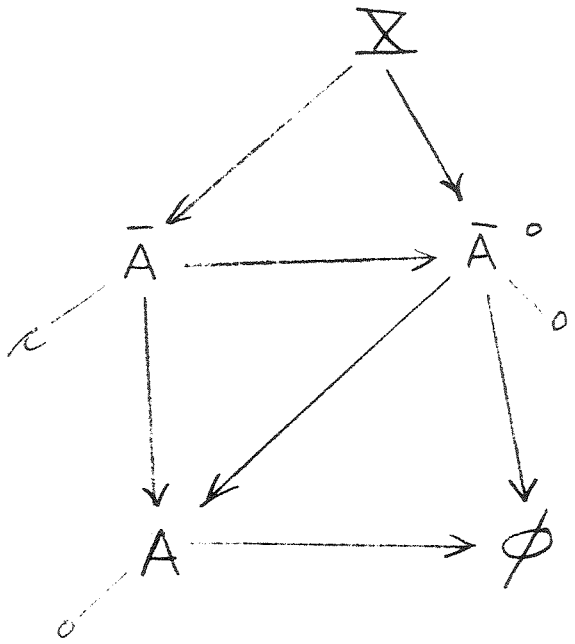
(I) $\quad OPEN(A) \longrightarrow REGOPEN(A)$

where A is a Skolem constant, it would rewrite the theorem in the following way.

(i) $\quad OPEN(A) \longrightarrow \bar{A}^{0} = A$

(ii) $\quad OPEN(A) \longrightarrow \bar{A}^{0} \subset A \wedge A \subset \bar{A}^{0}$. $\quad A \subset \bar{A}^{0} \wedge \bar{A}^{0} \subset A$.

The first half of the conclusion is proved immediately from the graph which the program constructs.

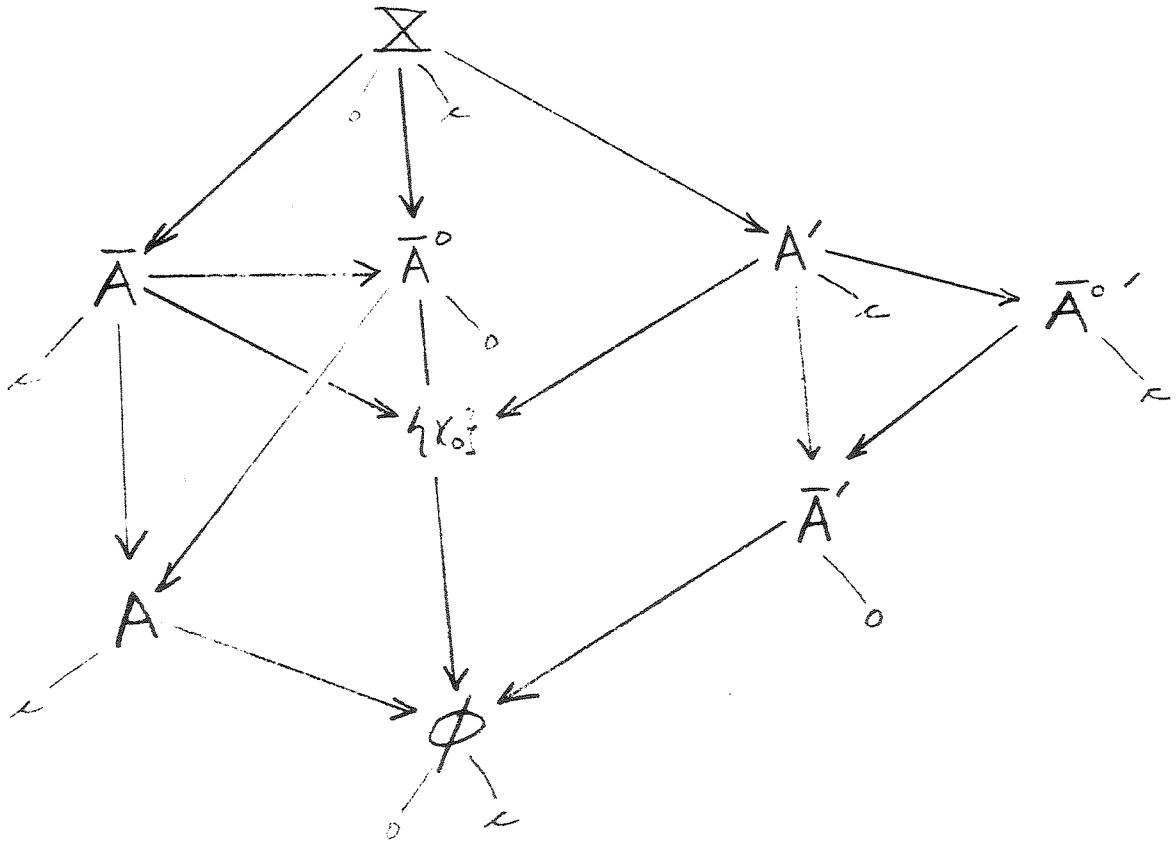


NOTE: The little o's indicates that the set is open; c means the set is closed

The GRAPHER tries to prove the second conclusion of (ii) and rewrites it as follows

(iii) $\quad OPEN(A_0) \wedge x_0 \in \bar{A}^{0} \longrightarrow x_0 \in A$.

The arc $\bar{A}^0 \longrightarrow \{x_0\}$ is added to the current graph but to no avail. The program makes a few other unsuccessful probes and then gets suspicious. Maybe (I) is not true! So now GRAPHER tries to assert the negation of the conclusion; i.e. $x_0 \notin A_0$ which is immediately rewritten as $x_0 \in A_0'$. All these facts are incorporated into the picture until we finally have



There do not appear to be any contradictions in the diagram so the program tries to assign values to the various objects in the diagram subject to the restraints imposed by the relations depicted in the graph and the axioms of topology. Basically what the algorithm does is to start from the bottom of the diagram and work upwards -- assigning the smallest possible value to each object. For the

time being we will informally step through the program's actions. Later we will give a more precise specification of the algorithm.

When we view the graph we see that the two smallest objects, excluding ϕ, & $\{x_0\}$ are A and $\overline{A}'$. The GRAPHER always starts to work on the uncomplimented set for reasons which, if not clear now, should become clear later.

1. The algorithm tries to assign A the smallest possible value which is ϕ.

2. This choice for A is substituted for A throughout the graph. All possible reductions are performed. This means that $\overline{A} = \overline{\phi} = \phi$. This causes trouble though for we have $x_0 \in \overline{A}$; in other words $x_0 \in \phi$. So $A = \phi$ is not a viable choice.

3. The program realizes that it must put a point in A. It uses the only point so far discussed. It therefore tries to assign the value $\{x_0\}$ to A. But the program immediately detects x_0's simultaneous presence in A and A'. Hence this too is an impossible assignment.

4. The program now discards x_0, "activates" another point p_0 and gives A the value $\{p_0\}$. So far, this choice appears to be okay. At the time our partial assignment to X is $\{x_0, p_0\}$.

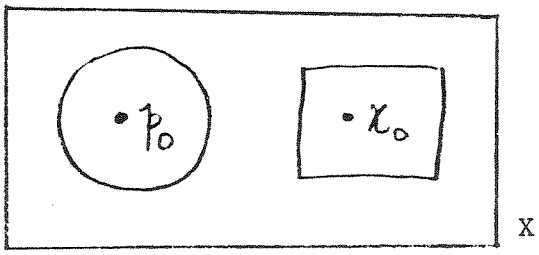
5. We now move "up" the graph and find the smallest superset of A. Since from the graph we have both $A \subset \overline{A}$ and $A \subset \overline{A}^0$ and also $\overline{A}^0 \subset \overline{A}$ we operate on $\overline{A}^0$ next. Since we see that $\overline{A}^0$ is constrained to contain at least x_0 and p_0, $\{x_0, p_0\}$ is our first assignment to $\overline{A}^0$. At this point our open sets are $\{p_0\}$ and $\{p_0, x_0\}$.

6. The program continues to work up the graph. The set which is immediately above $\overline{A}^0$ is $\overline{A}$. We assign to $\overline{A}$ the minimal assignment, that is, $\{x_0, p_0\}$.

7. At last the program works its way up to X. X also gets the value $\{x_0, p_0\}$. The values of the other sets now fall out immediately. We are finally left with

$$A = \{p_0\}$$
$$X = \{x_0, p_0\}$$
$$\tau = \{\{x_0, p_0\}, \{p_0\}\} .$$

Diagramatically we have



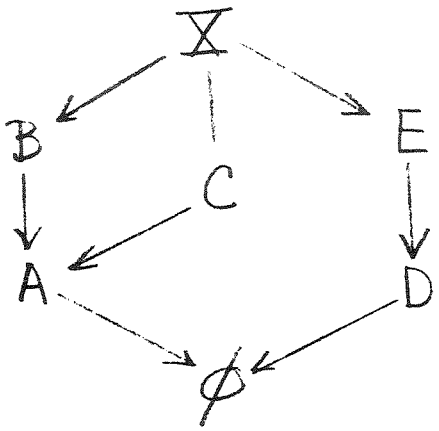
which was what we constructed as counterexample 3. It appears that the human and the program are doing pretty much the same sort of thing.

Some notes on the algorithm

The program starts at the bottom of the graph and works its way upward. The notion of upward is well defined since the graph can contain no loops. If there ever arises a chain of inclusions of the form $A_0 \subseteq A_1 \subseteq \ldots \subseteq A_n \subseteq A_0$ then the links corresponding to those relations are removed and the diagram notes the fact that $A_0 = A_1 = \ldots = A_n$.

If the sets A and A' appear in the graph, the algorithm will only attempt to assign values to A. The reason being that we would have to know what X is before we could give A' a meaningful value. The assignment to X is essentially

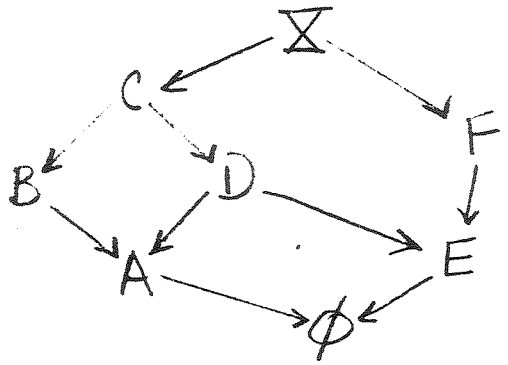
the last important assignment. The assignments occur primarily in a depth-first fashion. For example, if our graph were



the algorithm would traverse the graph in the order

A B C D E X .

The exception occurs when some set A has several subsets, say B,C mentioned in the graph. Then both B,C must be assigned values before A. To illustrate this, suppose our graph looked as follows:



then our order of assignment goes

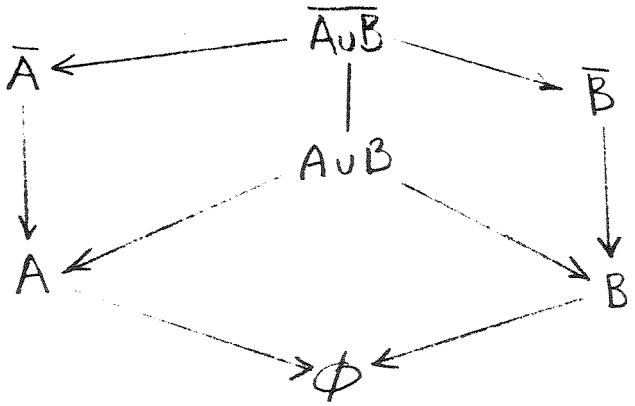
A, B, E, D, C, F, X .

The reason behind this funny order is that at each choice point we assign the smallest value to a set with respect to the restrictions implied by the graph. One of the most crucial restrictions is the values of a set's subsets.

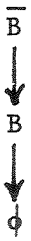
Let A be a set to which we want to assign a value, call it $v(A)$. Let $A_1,\ldots,A_n$ be the immediate subsets of A. Then our initial choice of a value for A is

$$v(A) = \bigcup_{i=1}^{n} v(A_i) .$$

whenever a choice $v(A)$ for A is made, a new copy of the graph is made with that $v(A)$ substituted for A throughout. The REDUCE operations described in [2] are applied to the nodes of this modified graph and the search begins from where we left off. For example, suppose we have



and we assign to A the value ϕ. Then ϕ is substituted for A throughout the graph and we wind up with



since $\overline{\phi} = \phi$ and $B \cup \phi = B$.

If an assignment causes a contradiction then we back up (a la MICROPLANNER) to our last assignment and try to add another point. We use the next available

point in POINTLIST (the list of points available to GRAPHER) that does not cause a contradiction. For example, the program never tries to add p to A if $p \in A'$. If the program cannot add a new point, it backs up to the next choice point and tries again. The program never tries to add two new points to a node. It appears to be the case that if <u>one</u> new point is not sufficient, then no finite number of new points will work.

At all times the program is keeping track of the topology τ. Whenever open sets are assigned values, they are put into the topology and all finite unions and intersections of the new set with previous sets are added also.

Having described these basic principles, we can look at what the algorithm does with some other examples.

<u>Theorem</u> ?? $\quad \forall A \ \forall B \left[\overline{A \cap B} = \overline{A} \cap \overline{B}\right]$.

The program rewrites this as

$$\overline{A \cap B} \subseteq \overline{A} \cap \overline{B} \wedge \overline{A} \cap \overline{B} \subseteq \overline{A \cap B} \quad \text{where} \quad A,B \quad \text{are Skolem constants .}$$

The first conjecture is proved immediately from the graph, leaving only the second. The program rewrites this as

$$x_0 \in \overline{A} \cap \overline{B} \longrightarrow x_0 \in \overline{A_0 \cap B} \quad \text{for} \quad x_0 \quad \text{another Skolem constant.}$$

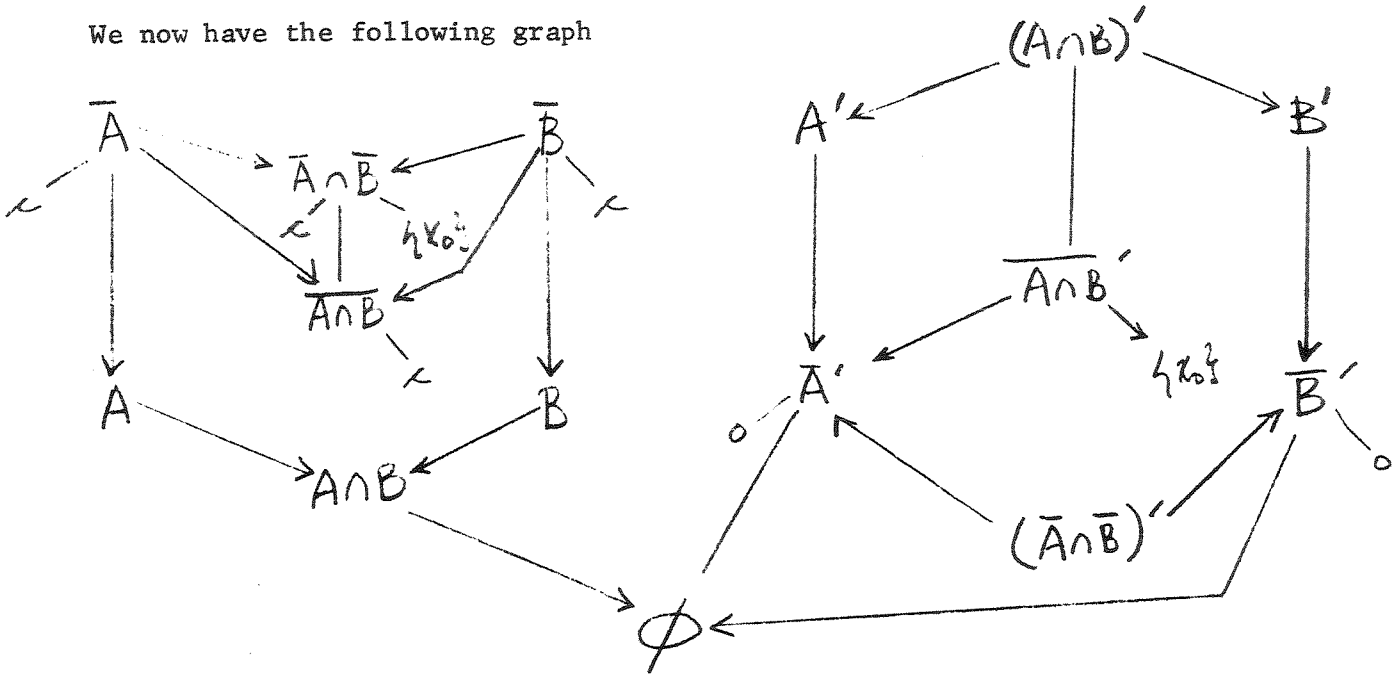
The hypothesis is asserted and the conclusion attempted. A failure here evokes an attempt at a counterexample. Now the program asserts

$$x_0 \notin \overline{A \cap B} \ .$$

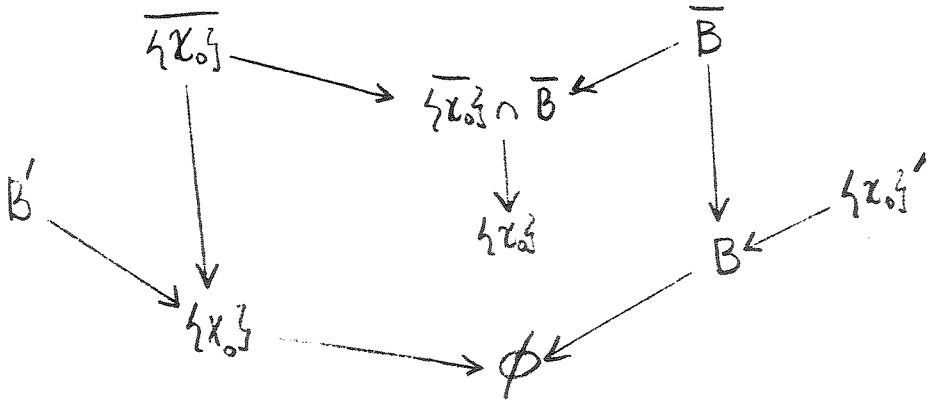
In other words,

$$x_0 \in \overline{A \cap B}' .$$

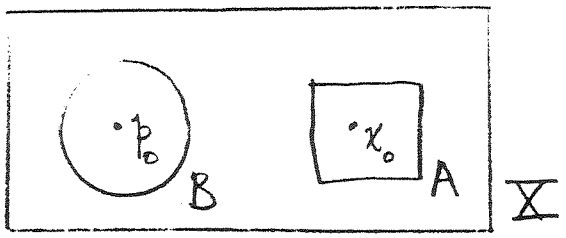
We now have the following graph



The program's first step is to assign $A \cap B$ the value ϕ. By the reductions mentioned in [2] this forces us to assert $A \subseteq B'$ and $B \subseteq A'$. The program now tries to assign A the value ϕ. This produces a contradiction since $\overline{A} = \overline{\phi} = \phi$ and we have $x_0 \in \overline{A}$. So the program tries the next point, x_0. At this point the graph contains the following substructure.



Before the program can assing $\overline{A} = \{\overline{x_0}\}$ a value, it must first assign $\{\overline{x_0}\} \cap \overline{B}$ a value. Obviously the minimal value is $\{x_0\}$. The next set to which the program tries to assign a value is $\overline{B}$. But before it can do so B must have a value. Since B can have neither the value ϕ nor $\{x_0\}$ (Why?) it is given the value $\{p_0\}$. Now $\overline{B}$ is assigned the value $\{x_0, p_0\}$ and X is assigned the same value. So we have the, by now familiar, topological space



We will do one more example in some detail to show that not all our counterexamples are the same.

Let us try to extend our knowledge of regularly open sets. As Bledsoe mentions in [3], one of the natural steps to undertake is to test for closure with respect to some function. Specifically, if P is a n-ary predicate and f is some defined n-ary function then P is closed with respect to f if

$$\forall A_1, \ldots, A_n [P(A_1) \wedge \ldots \wedge P(A_n) \longrightarrow P(f(A_1, \ldots, A_n))] \quad .$$

When dealing with sets, some of the most obvious functions to consider are $\cup, \cap$. This would seem especially true in topology since topology is nothing more than a family of sets closed with respect to union and intersection [with some cardinality restrictions]. So, going back to our notion of regularly open set, two possible theorems pop into mind

1. $\text{Regopen}(A) \wedge \text{Regopen}(B) \longrightarrow \text{Regopen}(A \cap B)$
2. $\text{Regopen}(A) \wedge \text{Regopen}(B) \longrightarrow \text{Regopen}(A \cup B)$.

Indeed, the GRAPHER establishes the truth of (1) immediately. The second conjecture is more difficult. After the program has defined the main concepts, we have (2) in the restated form

$$2'.\quad \mathrm{OPEN}(A) \wedge \overline{A}^{0} = A \wedge \mathrm{OPEN}(B) \wedge \overline{B}^{0} = B \longrightarrow \mathrm{OPEN}(A \cup B) \wedge \overline{A \cup B}^{0} = A \cup B \ .$$

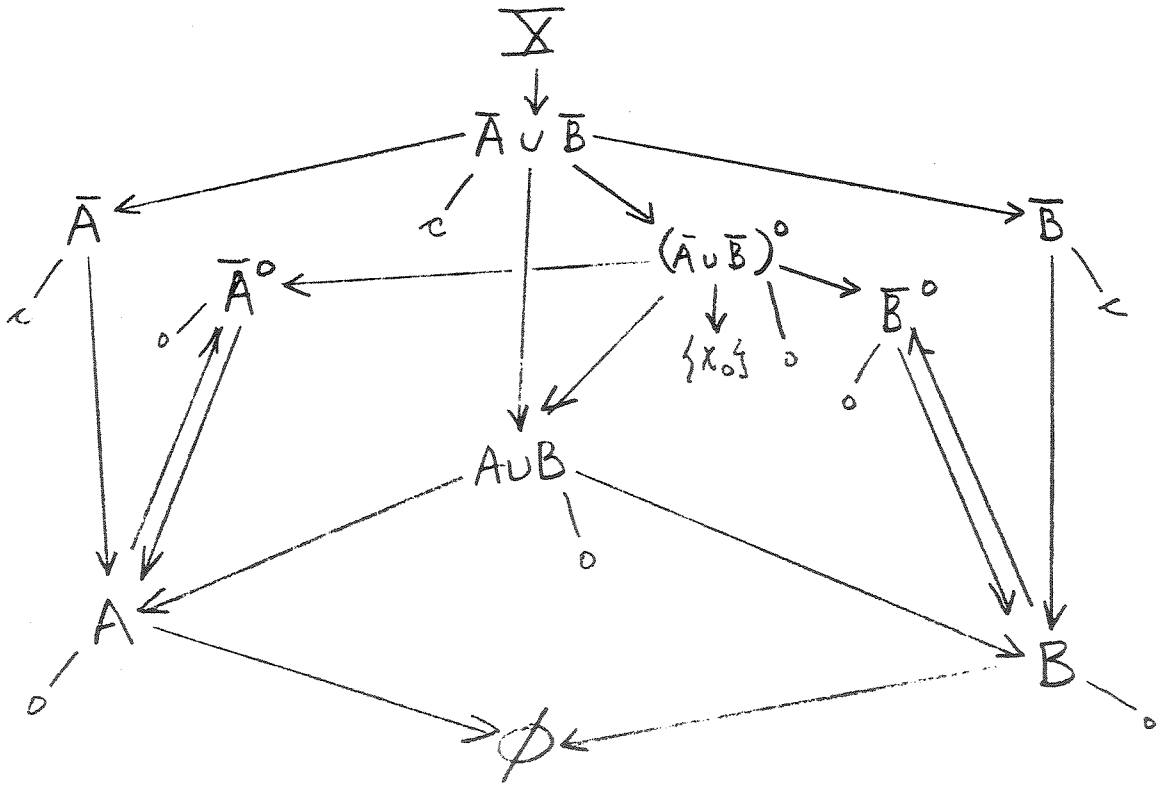
The first conclusion is immediately verified and the second is redefined as

$$A \cup B \subseteq \overline{A \cup B}^{0} \wedge \overline{A \cup B}^{0} \subseteq A \cup B \ .$$

The first inclusion relation is also immediately verified. The program grinds on the second conclusion, redefining it as:

$$x_0 \in \overline{A \cup B}^{0} \longrightarrow x_0 \in A \cup B \ .$$

The hypothesis is asserted but we have failure on the conclusion. Now the program tries for a counterexample and it asserts $x_0 \in (A \cup B)'$, or, as the GRAPHER prefers to say, $x_0 \in A' \cap B'$. At this point our computer-produced graph contains the following subgraph, (using the fact that $\overline{A \cup B} = \overline{A} \cup \overline{B}$ to deduce that $x_0 \in (\overline{A} \cup \overline{B})^{0}$):



Note that the fact $x_0 \in A' \cap B'$ dictates that $x_0 \in A'$ and $x_0 \in B'$. The graph that the machine constructs does contain these facts. We do not draw them in here for the graph is already complicated enough. The complinentary pair of arcs between A and $\overline{A}^0$ indicate equality. Now the program begins by assigning to A the value ϕ. When ϕ is substituted for A throughout and the reductions are made, then the graph reduces to a graph containing nodes pertaining to B only. This reduced-graph contains the fact that $x_0 \in B$ and $x_0 \in B'$ [try to see how this happens] hence $A = \phi$ is not possible.

The program tries to put the point x_0 into A but this does not work since $x_0 \in A'$. A is now given the value $\{p_0\}$. Of course $\overline{A}^0$ gets the same value and the program tentatively assigns $\overline{A}$ the same value.

The GRAPHER now tries to give B a value. The possibilities ϕ, $\{x_0\}$, $\{p_0\}$ are easily eliminated. [Note that if $B = \{p_0\}$ then $B = A$ and we are back to the case where the graph talks only about A.] Now B gets the value $\{p_1\}$. Hence $\overline{B}^0$ gets the same value. $\overline{B}$ also gets the same value. This choice for $\overline{B}$ fails since it means that $\overline{A \cup B}$ (which is the same as $\overline{A} \cup \overline{B}$) is equal to $\{p_0, p_1\}$. But from the graph we know that $x_0 \in \overline{A} \cup \overline{B}$. So the program realizes that it must give another point. The first point on the list to try is x_0 and so for the time being $\overline{B} = \{p_1, x_0\}$ looks good. Let us recap our assignments so far

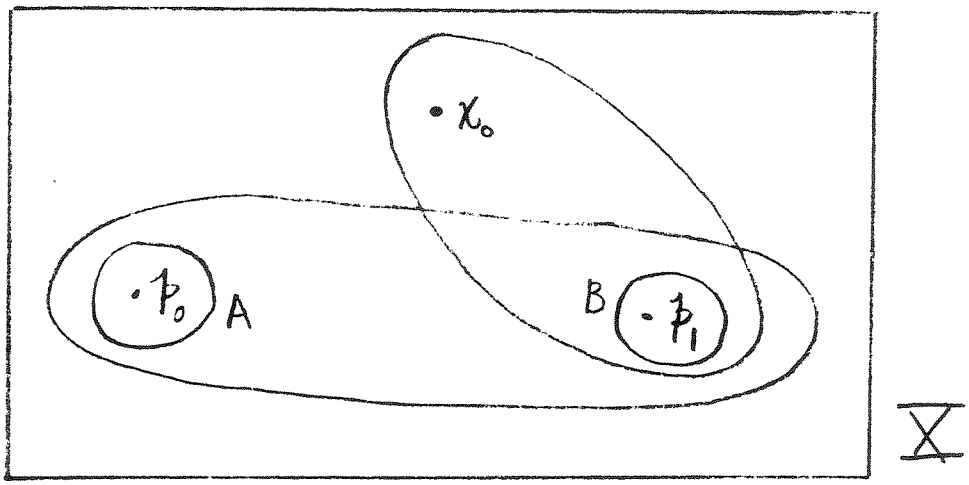
$$A = \overline{A}^0 = \{p_0\}$$
$$B = \overline{B}^0 = \{p_1, p_0\}$$
$$\overline{A} = \{p_0\}$$
$$\overline{B} = \{p_1, x_0\} \ .$$

At this point our space $X = \{x_0, p_0, p_1\}$, and our open sets are $A, B, \overline{A}', \overline{B}'$ and all

possible unions. In other words

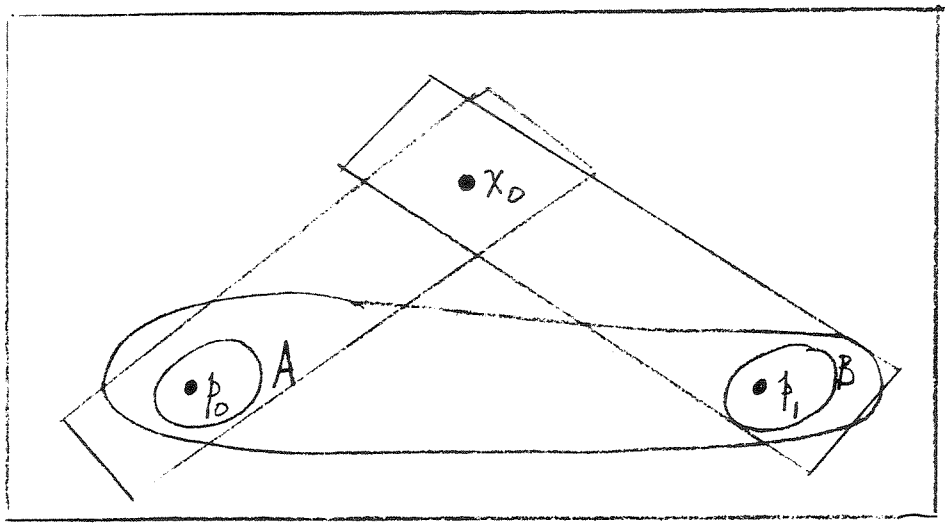
$$\tau = \{\{p_0\},\{p_1\},\{p_0,p_1\}\},\{p_1,x_0\}\} \ .$$

Hence the closed sets, call them τ' are $\{\{p_1,x_0\}\{p_0,x_0\},\{x_0\},\{p_1\}\}$. Hence we have the following topology



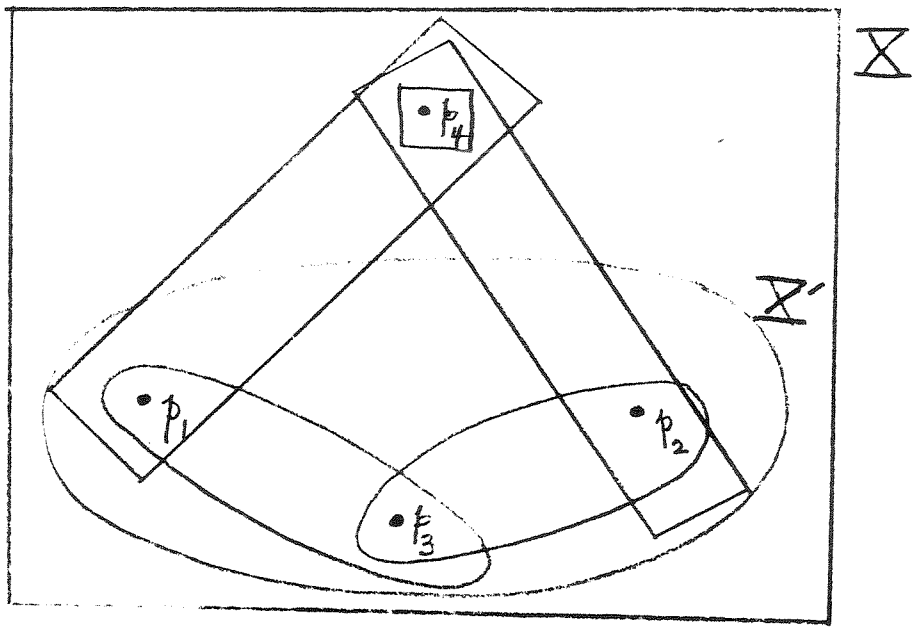
We have $A \cup B = \{p_0\} \cup \{p_1\} = \{p_0,p_1\}$. Hence $\overline{A \cup B} = \bar{A} \cup \bar{B} = \{p_0\} \cup \{p_1,x_0\} = \{p_0,p_1,x_0\}$ and $\overline{A \cup B} = (\bar{A} \cup \bar{B})^0 = \{p_0,p_1,x_0\}_0 = \{p_0,p_1,p_0\}$.

But notice that in this topology, $\bar{B}^0 = \{x_0,p_1\}$ violates the restriction in the graph that $B \subset \bar{B}^0$. So GRAPHER looks at its last assignment, which was the assignment of $\{x_0,p_1\}$ to $\bar{B}$ and tries to redo it. It finds that it cannot redo $\bar{B}$ so it looks at the assignment of $\{p_1\}$ to B. There is nothing GRAPHER can do to this assignment since by its rules, if one new point won't work, two new points won't work. The next assignment that GRAPHER can undo is the assignment of $\{p_0\}$ to $\bar{A}$. GRAPHER makes the tentative assignment $\{p_0,x_0\}$ to $\bar{A}$ and proceeds as before. This time everything works and we have the following space and topology



We would like to present one more counterexample that was generated after the previous work was done. The algorithm had evolved some in the meantime, but is sufficiently like the algorithm mentioned to make the following example meaningful.

The computer was asked if every subspace of a normal space is normal. For those who have forgotten, a topological space is normal if every two disjoint closed sets can be separated by open sets. In response to the query, the machine printed the following four point space as a counterexample.



Note that X with the indicated topology has no pairs of disjoint closed sets and hence is normal. However, we drop the point p_4 to form the subspace X', we now have the two disjoint closed sets $\{p_1\}$ and $\{p_2\}$. But every open set that contains p_1 also contains p_3 and every open set that contains p_2 also contains p_3 hence $\{p_1\}$ and $\{p_2\}$ cannot be separated by open sets. We were pleasantly surprised by the appearance of this example since the standard textbook counterexamples are quite complicated. For example, if Ω is the first uncountable ordinal and ω the first infinite ordinal, then the space $X = [0,\Omega] \times [0,\omega]$ is normal. If we form X' by dropping off the corner point (Ω,ω), then X' is not normal (X' is called the Tychonoff plank). Other examples are formed by embedding some non-normal space in a cube (which is always normal).

Conclusion

We have presented an algorithm for constructing small counterexamples to false topological theorems. We make no claims for any sort of generality -- even within the realm of point set topology. The algorithm has been applied to quite a number of simple false assertions and it did succeed in generating simple counterexamples. There are a lot of interesting complications associated with generalizing the procedure. One of the most apparent is what to do with the tantalizing problem of cardinality: that is, when we step into the domain of the infinite. To a large degree, though, the only way these topologies differ is that they are infinite. What I mean is that there are, I feel, very few facts about the concepts finite and infinite, countable and uncountable that people have as a working part of their mathematical knowledge. I feel it is a challenging, but not an overwhelming task to isolate many of the facts and incorporate them into a mechanical mathematician.

REFERENCES

[1] Bledsoe, W.W. and Peter Bruell. A man-machine theorem proving system. A.I. Jour., 5(1974), 51-72.

[2] Ballantyne, Mike and Bill Bennett. Graphing Methods for topological proofs. Univ. of Texas, Math. Dept. Memo ATP 7, 1973.

[3] Bledsoe, W.W. Discussions on theorem proving. Univ. of Texas, Math. Dept. Memo ATP 10, Nov. 1973.

[4] L. Siklossy and J. Roach. Proving the impossible is impossible is possible: disproofs based on hereditary partitions. IJCAI-73, 383-387.