Working paper for ICA Generalization Working Group Workshop, 1999

Title (workshop theme):

Practical Solutions for Specific Generalization Tasks

Author: **Dan Lee**
Keywords: **building simplification, centerline, enriching data, interactive editing**

Given the "classic" ARC/INFO GIS software environment, our effort to develop generalization tools has been based on the existing data model and software structure of the system. Although the result of generalization is a new set of data, it is usually not obtained by well-defined straightforward queries as in typical data extraction, but requires more complicated decision-making The rules guiding generalization vary from case to case and ways to avoid and resolve conflicts dynamically remain under research. The practical solutions for generalizing ARC/INFO data, therefore, have become the combinations of automated processes, Macro-program procedures, and interactive editing. This paper discusses our experience in deriving generalization solutions and presents the new tools for simplifying buildings and collapsing road casings to centerlines, along with post processes.

Strategies of deriving generalization solutions

To provide computer-assisted solutions for generalization, we need to pursue the use of advanced technology in the mapping industry, reducing or replacing the tedious manual work in production environment. Our success is measured by how much time is saved and how much consistency is generated in comparison to the traditional manual generalization on the user's end, and by the efficiency in development. Generalization tasks are comprehensive and interrelated. The following strategies are very important in defining doable tasks and producing practical solutions: breaking complicated problems down to simpler solvable problems, automating the process as much as possible and enriching the output with marked areas and process status, supporting interactive editing, and defining an overall workflow.

Defining solvable problems

In manual generalization, a cartographer draws the generalized result in context with all mapped features. Many actions, such as elimination, simplification, aggregation, displacement, and so on, can happen at the same time as drawing a feature. Attempting such comprehensive human decisions and actions in full automation would be unrealistic in today's digital cartography and drag the development into an endless effort.

Defining a solvable problem in our development means to find a particular area in a bigger problem in which certain rules can be applied and a decent result can be accomplished automatically. The automation would result in significant time saving in real production. Also quite importantly, the implementation must be constrained to a reachable time frame, for example, a product release cycle.

To distinguish the components of generalization and define solvable problems, we have divided generalization into nine categories of operation, including simplification, aggregation, collapse, exaggeration, displacement (Lee, 1996). Each may then be solved automatically by unique operators along with human interaction. An operator is a program that utilizes a special technique or an algorithm to produce the best possible results for a particular generalization task, for example, the operator BENDSIMPLIFY for simplifying lines (ARC/INFO 7.2.1 release). Unresolved issues are anticipated and will be revisited and resolved later. Examples of new generalization operators will be given in later sections.

Automating the process and enriching the output data

A computer program can do no more than what human being tells it to do. Since we lack full knowledge about generalization cases and rules, we can only program what can be clearly defined and is technically solvable. The generalization operators usually work according to user-input parameters and a set of rules. In most cases these globally set parameters and rules can be adjusted to suit local circumstances, and the generalized results are satisfactory. However, there can be some areas where the global parameters don't apply and the user's inspections and decisions are necessary or where the computation and development become too expensive to do. The way of handling these situations is either making the program interactive for user's instructions at the question areas during the process, or marking these areas for post-process.

Enriching the output data is a new concept that comes along with computer-assisted generalization. It means flagging the remaining problems and reporting information about the automated process that would help further editing. This approach has the advantage of not interrupting the automated process, not requiring user's attention during the automated process, and giving flexibility to post-processing. The examples in later sections demonstrate the use of this approach in more detail.

Supporting interactive editing

As part of our current solution to generalization tasks, interactive editing programs, such as AML (the ARC/INFO Macro Language) scripts, are being made available to help solve the remaining problems and refine the results. Each of the editing programs supports certain types of editing associated with a generalization operator and its enriched output. It invokes the ARCEDIT-based editing environment with a menu-driven interface and allows the user to queue through each flagged area and apply a unique method to modify features. Some of these methods are made by putting a series of existing commands into one menu choice, which simplifies and minimizes user's involvement, others by AML programs that compensate what is not offered directly by the existing tools.

The interactive environment also allows other features to be displayed as references to help deal with spatial conflicts, topology, and the overall balance of mapped information. All other editing tools are accessible as well, if needed.

Defining a workflow

A generalization workflow puts all automated or interactive steps into a logical sequence to accomplish a result and is usually data and scale dependent. For a particular data set, our experience in supporting a generalization benchmark has set an example of using such a workflow. The project required that we derive a 1:5000 scale topographic mapping database from a 1:1000 scale database. Each of the 17 feature layers (building, road, boundary, relief, and so on) was processed separately. Within each layer a specific sequence was set to generalize point, linear, and areal features. The results were then combined into 14 output layers. The post-editing needed was described to the client. With some modifications, this framework can be applied to other data sets.

A generalization workflow will also need to be tailored to fit various scales. The more the scale reduces, the more the generalization technique shifts from making local modifications to making global representations. For example, buildings are represented individually at large scales. They can be simplified or exaggerated up to certain extent, but still remain individual. As scale reduces, more and more buildings will be excluded, collapsed, and displaced; some will become part of urban areas. At very small scales, buildings, even urban areas will completely disappear; they can only be represented by location symbols. As a map producer (Pla, 1998) described (summarized by the author):

> In generalizing buildings from 1:5000 to 1:10,000 or 1:25,000 scales, the following options are used:
>
> > Extraction of small buildings and replacing them with a minimum symbol
> > Simplification of the rest of buildings
> > Exaggeration of special buildings
> > Other actions, including elimination, and aggregation
>
> When generalizing buildings to smaller scales, 1:50,000, 1:100,000 or 1:250,000, typically in urban areas, more streets disappear and aggregation of buildings is more used, besides other options.

A generalization workflow, therefore, is a result of analyzing and understanding the overall requirements (the theme, resolution, feature relationships and priorities, and so on), and putting a cartographer's thinking into the most efficient and logical operations. It is critical in the completion of a comprehensive generalization task.

In the next two sections, the two newly developed generalization operators, BUILDINGSIMPLIFY (that reduces details from building boundaries) and CENTERLINE (that collapses road casings to centerlines) are introduced. The discussion is focused on defining the solvable parts of the tasks, the enriched output, and the necessary post-processing.

Simplifying Buildings

Simplifying buildings (footprints) means finding a simpler representation of the original buildings by reducing details in their boundaries, while maintaining the essential shape and size of the buildings (Figure. 1). Buildings are generally orthogonal areas, therefore simplification will preserve and enhance orthogonality (Lee, 1998).

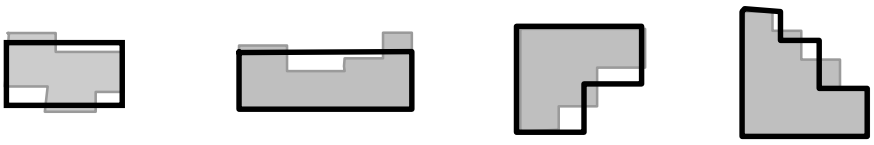

Figure 1 Buildings simplified to simpler forms

Simplifying buildings is for applications at large scales, where buildings are still represented individually. This section discusses the complications related to this task and their solution.

Identifying the solvable task among the complications

Without worrying about spatial relationships, each building can be easily reduced to a simpler form based on a set of rules. An iterative approach can be implemented to evaluate the pre-defined conditions and rules one at a time and make modifications to the geometry accordingly (Lee, 1998a).

However, it is unavoidable that any changes in geometry will cause spatial conflicts, overlapping or crossing with neighbor features, or being too close to them. With the existing data model, it is quite difficult to detect and resolve conflicts during the process to guarantee a conflict-free result. Although in theory a conflict can be resolved within its solution space, or local region (Peng, 1997), and the displacement of involved features should gradually descend outwards from the center of the conflict, the implementation has not been proved to be easy with the current data model. Therefore, we had to define the limitations and rely on some post-editing.

Additional complications are caused by groups of adjacent buildings. If each building connected in a group is simplified separately, the shared boundaries may end up mismatched in the results. If only the outer boundary of a building group is simplified as a single building, recovering the interior walls is technically quite challenging.

It then became clear that the automatically solvable task is the simplification, with limited conflict detection, of individual buildings and buildings connected in the simplest ways. The simplification status will be recorded for later examination and post-processing.

Performing simplification and recording status

A building simplification operator, BUILDINGSIMPLIFY as an ARC command, has been implemented for the upcoming ARC/INFO release. It reduces extraneous details

from building footprints according to two user-input parameters, simplification_tolerance and minimum_area. All attributes on building polygons are transferred to the output.

This operator recognizes buildings as topologically disjoint, connected with straight lines near parallel to each other, and connected in more complicated ways. Each separate building is simplified by itself. Buildings connected with straight lines are simplified as a group. Buildings connected in more complicated ways are not simplified (Figure. 2).

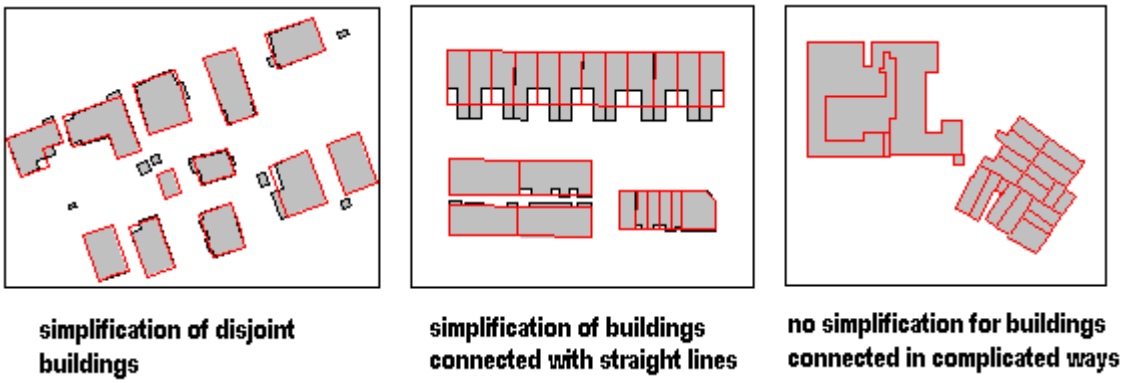

Figure 2 Buildings in three types of appearances

The boundaries of disjoint buildings or buildings connected with straight lines will be enhanced so that all near-90-degree angles become exactly 90 degrees. Based on the simplification_tolerance, a building can be simplified by, for example, filling up, cutting off, or widening isolated small spaces (intrusions or extrusions), straightening a side, but keeping the measured area roughly the same as the original (Swiss Society of Cartography, 1987). Any building or a group of connected buildings with a total area smaller than the minimum_area will be excluded. The maximum degree of simplification is reached when a building is reduced to a rectangle (Figure. 3). Since this operator does not detect and avoid all spatial conflicts, the special REGION feature class is used as the output that allows overlapping topology in buildings.

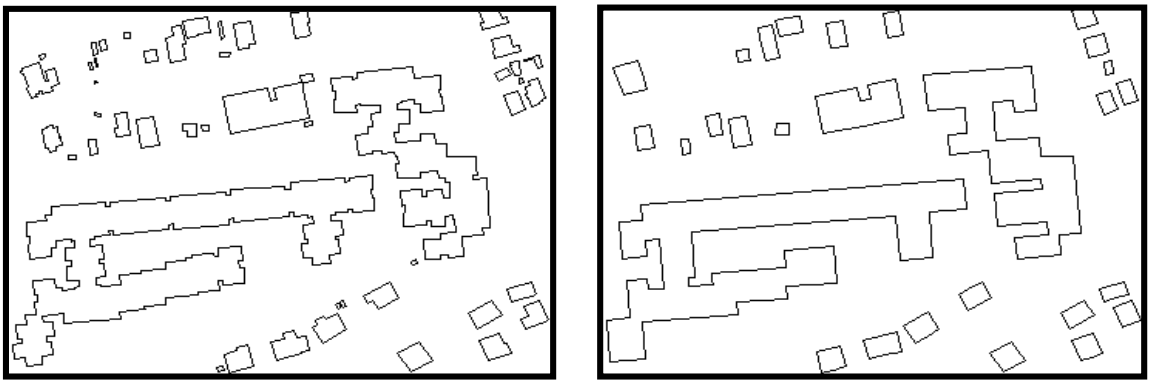

Figure 3 Before (left) and after (right) simplification

The program records the status of each building in the output by an attribute item, STATUS. A separate building will have a STATUS value of 1 if it is completely simplified. If a spatial conflict is found during the iterative process, the building will not be simplified further and will receive a STATUS value of 2.

If the simplification_tolerance is relatively large compared to the size of the building, the building will be simplified directly to a rectangle centered at its own center of gravity. The area will remain the same. The sides of the resulting rectangle will be the same ratio as the sides of the bounding box aligned to the longest side of the original building (Figure. 4). If the resulting rectangle contains a side smaller than the simplification tolerance, the building will have a STATUS value of 3.

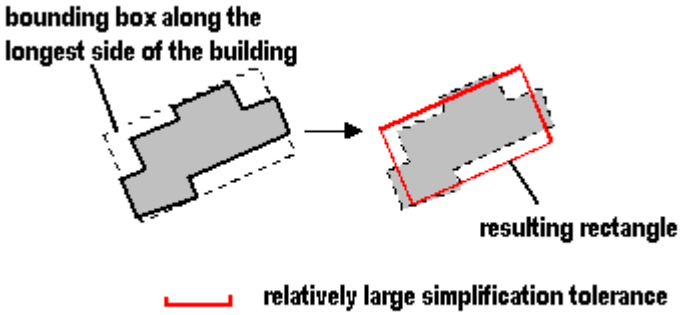

Figure 4 Building simplified directly to a rectangle

For buildings connected with straight lines, simplification will be limited to simple rules only. These buildings will have a STATUS value of 4. And finally, buildings connected in a complicated way will have a STATUS value of 5.

The output will contain another new item, BLDGGROUP. This item stores a unique value for each group of connected buildings. A single building will receive a BLDGGROUP value of zero. This item is used in checking conflicts among buildings and groups of buildings.

Locating spatial conflicts

Once buildings are simplified, an overall conflict detection among them can be done automatically by another new ARC command, FINDCONFLICTS. This program takes the simplified buildings as input and finds where they overlap or are too close to each other based on a specified distance and on the BLDGGROUP information.

To find the spatial conflicts, region-buffers are created around each building or group of connected buildings. Overlapping buffers indicate a conflict. An output will then be produced, storing these region-buffers with an item FREQUENCY for polygons. A polygon gets a FREQUENCY value of 2 or more according to how many region-buffers overlap (Figure 5). All non-conflicting areas receive a FREQUENCY value of 1. Buildings connected in one group are not considered as conflicting with each other. Only

the outer boundary of such a group is checked with neighboring buildings or groups of buildings.

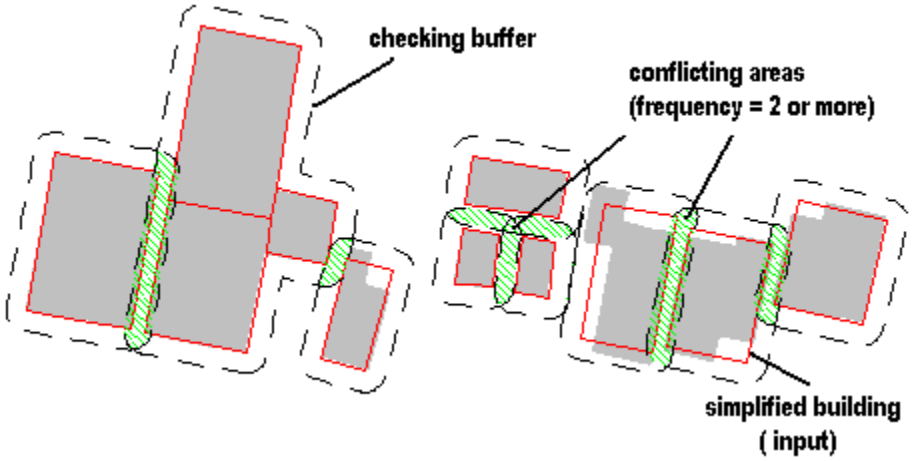

Figure 5 Finding spatial conflicts by buffering

Interactively resolving conflicts and refining the results

Given the simplified buildings with information about the status and conflicting areas, the post-editing is easier. An interactive program has been developed to invoke the ARCEDIT environment with a menu-driven interface. It allows the user to queue through each conflicting area and resolve it interactively. Two unique tools, move and delete as menu choices, were created that enable moving and deleting buildings as regions and maintain the region topology. The user can also easily select buildings according to their STATUS values and refine the result.

Collapsing Road Casings to Centerlines

Collapsing road casings to centerlines means finding the single-line representation of the casings based on specified width tolerances, while preserving proper connectivity (Lee, 1998b). It is another typical generalization problem that different algorithms (Christensen, 1999; Thomas, 1998) can be applied to solve the essential task, but a fully automated solution for dealing with real world data can be a great challenge. This section describes our approach, again including an automated process and post-editing.

Analyzing the nature of the problem and defining the solvable task

Road casings are collected and extracted in many different ways depending on user's data model and standards. What make the centerline problem more complicated than it seems include some confusing situations in road casings, difficult decisions on complex intersections, and single-line roads and wide areas mixed in data.

The first thing to do in deriving centerlines is to differentiate the inside of the casings, where the centerlines should be, from the spaces between them. If the casings representing a connected road network form a closed polygon, then it would be easy to use the inside of the polygon to guide the centerlines. But casings are not normally collected that way and it is not easy to either make such polygons or find the inside areas without such polygons or other references automatically. It is especially confusing when a number of casings run parallel with their widths within the specified width range or when casings have similar widths as the spaces between them.

Simple road intersections, such as a "T"-shaped or a "+"-shaped intersection, are easy to recognize and to solve automatically, while arbitrarily shaped intersections (Figure 6) need more rules to guide the way to connect them. The derived centerlines may not always connect nicely at one point. They need to follow priorities and connect in certain order. For example, at a three-way intersection, the two centerlines that are almost on a straight line or near perpendicular are connected first and then the third centerline can be projected or extended to the first connecting line. Applying the similar logic, it is possible that at a multiple-way intersection, all centerlines get connected, but the connection can be unsatisfactory because some lines can be projected too far or too many connecting points are produced. To certain extent, rules can be made to further refine and adjust these details, but some extremely complicated intersections may need human inspection and decision to help solve them.

Figure 6 Arbitrarily shaped intersections

Ideally, casings represent the edges of roads. But quite often the casings also follow the edges of wide areas, such as parking lots, open areas, or unusually shaped cul-de-sacs, and single-line roads may be included in the same data as well. These features may not be attributed differently and it is not a trivial job to automatically recognize them so that centerlines can be produced and connected to them properly.

Our goal was to provide a generic solution without restrictions on the input data. We defined an approach that uses an automated program to create centerlines and simple intersections (up to four-way intersections), and yet requires post-editing to resolve the remaining intersections, to remove unintended centerlines, and to make corrections in other areas marked by the program.

Creating centerlines and necessary attributes

A new operator, CENTERLINE as an ARC command, has been implemented also for the upcoming ARC/INFO release. It produces centerlines according to two user-input parameters, maximum_width and minimum_width.

The CENTERLINE program scans and separates the data in two directions, horizontal and vertical, and creates centerlines where casing width is within the specified range. It then evaluates each intersection and connects those that fit the rules. Casings with the width beyond the specified range and single lines not used to create centerlines are copied to the output data such that the road network won't be broken.

The output data contains an item LTYPE (line-type) that differentiates centerlines from unresolved areas. If the width of the input data is relatively constant and the intersections are simple, a complete centerline result can be produced, that is, LTYPE = 1 for all resulting lines. Otherwise, unused lines (such as a single casing or casings with a width beyond the specified range) and outlines around complicated intersections will be flagged with a LTYPE value of 2 for editing them further (Figure 7).

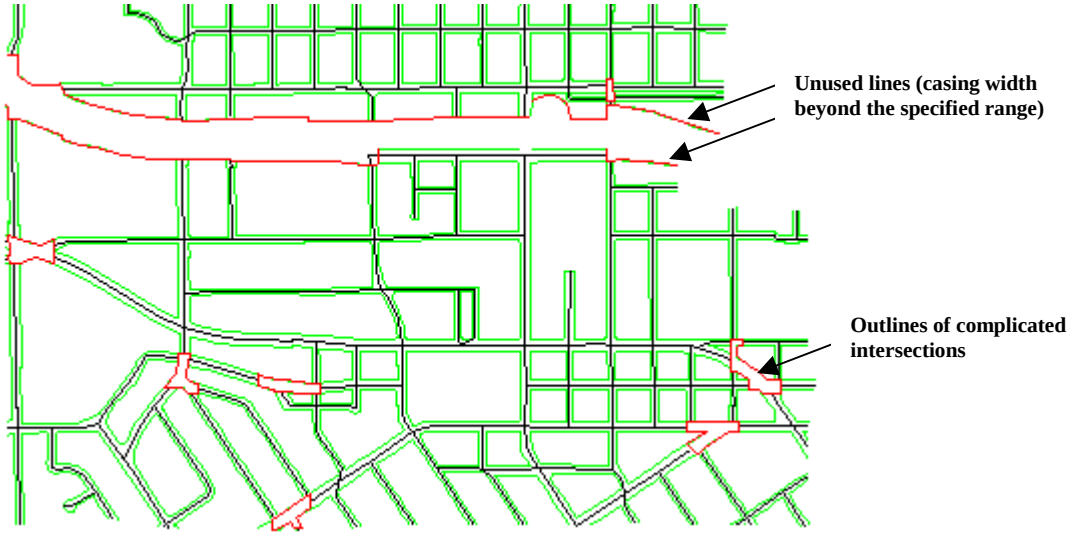

Figure 7 **Results of CENTERLINE with unresolved intersections and wide roads flagged.**

Four other new items, LL# (left casing record number), RL# (right casing record number), L-ID (left casing ID), and R-ID (right casing ID), also come with the output data to relate the centerlines to their source casings for attribute transfer.

Interactively resolving intersections and refining the results

An AML program has been developed to invoke the ARCEDIT environment with a menu-driven interface. It allows the user to queue through each LTYPE 2 area and

resolve the problem interactively. A few most needed editing tools, such as remove an arc (line), extend an arc, and add vertices, are included in the menu choices. A unique tool was created to automatically join a set of selected lines at a user-specified intersecting point. Other special tools may be added in the future to incorporate more advanced rules and make the editing even easier.

Conclusions

Our experience in deriving solutions for building simplification and creating road centerlines has led us to believe that properly balancing the automated process and interactive editing is a practical way of solving complex generalization problems reducing labor work significantly. For example, as already proved by two evaluation tests (Litton, 1998 and Oxenstierna, 1998), the CENTERLINE solution can replace existing procedures and generate 70-80% time saving in production.

We are also defining and implementing a number of other generalization operators, including aggregation of general area features, such as vegetation, and buildings in specific, collapse of buildings to points with orientations, and collapse of area features, such as lake, to single-line presentation. I would like to present our preliminary approaches in solving these problems at the Workshop and share discussions with others. As more and more of these generalization operators become available, the overall generalization workflow will continue being simplified, more labor work replaced, and more consistent results produced. Since the object-oriented technology is being used in defining our future products, it certainly will benefit generalization processes. We are looking forward to developing new tools based on stronger and richer data model and filling up more blanks in the area of digital map generalization.

Reference

Christensen, Albert H. J., 1999, “Cartographic Line Generalization with Waterlines and Medial-Axes”, Cartography and Geographic Information Science, Vol. 26, No. 1, p. 19-32.

Lee, Dan, 1998a, “Simplification of Buildings”, ESRI internal design document.

Lee, Dan, 1998b, “Creating Centerlines from Road Casings”, ESRI internal design document.

Litton, Adrien, March 3, 1998, On CENTERLINE approach, email notes.

Oxenstierna, Andreas, March 2, 1998, On Centerlines, email notes.

Peng, Wanning, 1997, “Automated Generalization in GIS”, ITC Publication Series, No. 50, p. 42.

Pla, Maria, Jan. 7, 1999, On building generalization, email notes.

Swiss Society of Cartography, 1987, “Cartographic Generalization – Topographic Maps”, Second Edition.

Thomas, Federico, 1998, “Generating Street Center-Lines from Inaccurate Vector City Maps”, Cartography and Geographic Information Systems, Vol. 25, No. 4, p. 221-230.