Spatial Reasoning about Flow Directions: Towards an Ontology for River Networks*

(Extended Abstract)

João Argemiro de Carvalho Paiva† , Max J. Egenhofer, and Andrew U. Frank
National Center for Geographic Information and Analysis
University of Maine
Orono, ME 04469
paiva@thrush.umesve.maine.edu
{max,frank}@mecan1.maine.edu

Abstract

Humans are very good in deriving the flow directions of a river network from such representations are aerial photographs or remotely sensed images. Apparently, the 2-dimensional geometry of the network is in most cases sufficient to derive its sources and destinations by reasoning about the flow directions of river network, and they need no additional information about slopes or heights. Formalizing the problem such that it can be automatically performed, however, has proven to be an extremely difficult problem. Within the realm of reasoning about flow directions in river networks, a particularly important problem is the analysis of relevant hydrological features. This paper describes initial results of the development of an ontology for river networks and formalizes the features in terms of a graph. It is shown how certain reasoning processes, simplifying the complexity of a river network, can be expressed as graph operations.

1 Introduction

River or drainage networks are fundamental concepts used for various analyses in geo-sciences. Geologists, for instance, derive original slope and original structure from drainage patterns, or transportation engineers examine river networks to determine how to access undeveloped land via waterways. A common problem in analyzing river networks is that these studies frequently have to be based on "incomplete" spatial information, i.e., information that lacks some clues that are crucial for certain decisions. Remotely sensed images or aerial photographs, for example, are data sources that contain only the necessary information about the location and extent of rivers, but unlike *in situ* observations, they lack explicit information about the flow direction. While humans have a distinct ability to derive the flow directions of such a planimetric representation of a river network, it is a difficult problem to infer them automatically.

Usually, additional information from a digital elevation model is used to complete the inference of the flow directions (O'Callaghen and Mark, 1984; Band, 1986; Frank *et al.*, 1986). While such an approach may be appropriate for steep terrain with significant elevation differences, it is infeasible in flat terrain. The Amazon region, located in northern Brazil, is a prototypical area of the latter type. It extends over 5,000,000 km^2 with approximately 100 m elevation differences along large parts of the Amazon and Solimoes rivers. Current efforts in building a geographic information system of this area to monitor deforestation (Souza, 1992) face the difficulties of covering a very large area with no existing maps and many temporal changes, e.g., due to high-water and erosion (Larovere and Goodman, 1992). In order to apply remotely sensed images as a means to monitor environmental changes in this area, it is necessary to explore alternative approaches to derive the flow direction in a river network.

This paper is part of a larger effort investigating different human reasoning mechanisms in geographic space (NCGIA, 1992). Reasoning in geographic space is typically based on inference, rather than direct observations (Chase and Chi, 1981). Different types of geographic spaces may be conceptualized such as complete partitions in 2-D as used to model such political subdivisions as countries; and networks to represent highway systems (Egenhofer and Herring, 1991; Frank and Mark, 1991). The geographic (large-scale) space that is made up by a river network is, in a first approximation, best modeled as a directed graph, in which the flow of the water determines the direction of the edges in the graph. (This model may be too simplistic for some situations such as tidal changes or human-regulated dams, which may periodically reverse the flow of some channels in a river network.)

The first quantitative studies of river networks and drainage basins (Horton, 1945) introduced the idea of ordering channel networks. Further work (Strahler, 1952) simplified the Horton ordering scheme, making it purely topological (Melton, 1959). Geologists recognized early that the angles at which stream segments join contain crucial information for the inference of the flow directions in drainage networks (see Serres and Roy (1990) for a review). Previous work in this area uses remotely sensed images. These images frequently provide only a partial view of a river network. Flow directions have been inferred by skelettonizing the water channels and applying a set of constraint rules about the junctions (angles and channel lengths) of river channels (Wand *et al.*, 1983; Haralick *et al.*, 1985). A simplified set of rules uses only on the angle geometry at each junction of three channels and is based on the assumption that the two consecutive channels that bound the most acute angle are the upstream channels (Serres and Roy, 1990).

Most work on river network topology takes into consideration the existence of channels and their junctions, while disregarding other hydrological features such as lakes, islands, or river deltas. An exception is Mark's and Goodchild's (1982) extension of Shreve's (1966; 1967) "probabilistic-topological model" for channel networks by lakes. This paper develops a more comprehensive model of features in a river network and

* João Paiva's work has been supported by the Brazilian National Research Council (CNPq) under grant number 260226/91.2. Additional support from Intergraph Corporation and NSF for the NCGIA under grant No. SES 88-10917 is gratefully acknowledged.

† On a leave of absence from the National Institute for Space Research (INPE), Image Processing Division, São José dos Campos, SP, Brazil.

describes an ontology for reasoning about flow directions in river networks. The river network features will be formalized in terms of a graph data model. Such an approach is a prerequisite for developing robust methods to infer the flow direction of a (partial) river network. For example, it is necessary to identify islands in a river network, because for them the "most acute angle" rule infers contradicting directions for the "start" and the "end" of an island.

The remainder of this paper is structured as follows: Section 2 introduces some notions from graph theory. Section 3 describes fundamental river patterns and shows their abstract representations as graphs. Section 4 analyzes these graphs and draws some conclusions about simplifications of river patterns, without losing information necessary for a formal reasoning process about flow directions. The conclusions in Section 5 discuss future work.

2 Graphs

The formal basis for modeling river networks is the mathematical structure of a *graph*. Graphs have been investigated extensively in computer science and applied mathematics. The following fundamental definitions are based on Knuth (1973) and Gill (1976).

A *directed graph*, or *di-graph*, is a set of vertices and a set of arcs where each arc leads form a vertex V to a vertex V'. V and V' are also called respectively the *initial* and *final* vertex of an arc e. The *orientation* of an arc is an equivalence class, defined as a positive value *from* the initial vertex *to* the final vertex, and negative in the reverse direction. The *out-degree* of a vertex V is the number of arcs leading out from it, i.e., the number of arcs e, whose initial vertex is V. Conversely, the *in-degree* of V is the number of arcs whose final vertex is V. The *degree* of a vertex V is then the sum of the in-degree and out-degree of V.

A directed graph will be depicted as a sequence of nodes (for the vertices) and edges between the nodes (for the arcs). The orientation of each arc will be represented by an arrow pointing from the node for the initial vertex to the node for the final vertex (Figure 1).

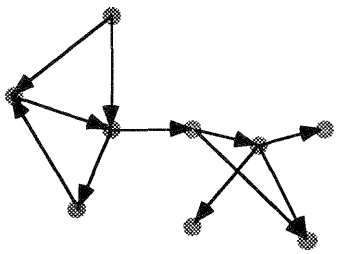

Figure 1: A directed graph.

Given a set of arcs $\{e_1, e_2, \ldots, e_n\}$, $<e_1, e_2, \ldots, e_n>$ is an *oriented path* of length n from V to V' if (1) V is the initial vertex of e1, (2) V' is the final vertex of e_n, and (3) the final vertex of any e_k $(1 \leq k < n)$ is equal to the initial vertex of e_k's adjacent arc $e_{(k+1)}$. The *length* l of an oriented path is the number of arcs along the path. A path $<e_1, e_2, \ldots, e_n>$ is a *loop* if the initial vertex of e_1 coincides with the final vertex of e_n.. A loop is called a *cycle* if the initial vertices of all arcs in the path are distinct. Based on the concept of a cycle, two specific kinds of graphs are defined: (1) A *multi-graph* is a graph in which cycles of length 2 are permitted, i.e., two vertices can be linked by more than one edge, and (2) a di-graph without cycles is a *directed acyclic graph* or *dag*.

An ordinary graph G_0 abstracts from a directed graph G the orientations of the arcs. Thus, G_0 has an edge between V and V' if G has an arc either from V to V' or from V' to V. Two vertices, V and V' in an ordinary graph are *connected* if there exists a path between V and V'. Reversely, two vertices, V and V' are *disconnected* if there exists no path that connects V with V'. If V_1 " G_{0i} and V_2 " G_{0j} are disconnected, then the two graphs G_{0i} and G_{0j} are disconnected as well.

3 River Junction Patterns

The identification of possible node configurations in a river network is an important step in the chain of reasoning about the network's flow direction. An example of a drainage pattern is shown in Figure 2.

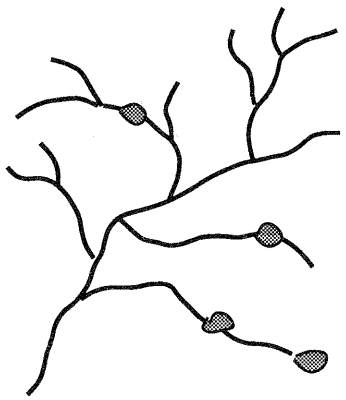

Figure 2: A drainage pattern.

Such channel patterns can be constructed from a small set of some basic *river junction patterns*. This paper focuses on the formalization of the most common river junction patterns as they are formed by channels, islands, and lakes. For the time being, we exclude such river features as waterfalls or dams, although their recognition and inclusion into out model may be an additional source of information for the inference of flow directions.

3.1 Channels

Channels are ways along which fluvial processes act to transport water and minerals out of a local region. In our model of river networks, a channel is a connected segment of a river between two distinct nodes. These nodes may be metrically significant points, such as sharp turns, or topologically significant landmarks, such as a source, a destination, or a junction. Each channel has a flow direction "upstream" or "downstream," which corresponds to the natural flow direction of the water. It is assumed that this flow direction is constant for each channel, i.e., that it does not change periodically.

Each channel maps onto an arc of a di-graph, with the flow direction of the channel being represented by the orientation of the arc. This mapping abstracts quantitative differences in length and shape of a channel. On the other hand, it preserves such qualitative information as the connectivity and the flow

direction, i.e., which arcs are linked by which vertices. Figure 3 shows examples of channels that map onto the same directed graph.

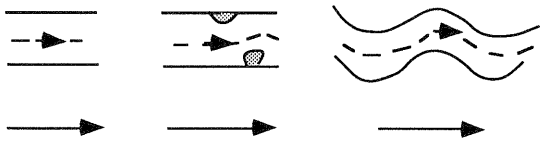

Figure 3: Channel patterns that map onto the same graph.

The following algebraic specification describes formally the behavior of channels. It is based on the definition of a *vertex*, which is a simple type with operation to *make* a vertex from a unique id, to *get* the id from a vertex, and to test whether or not two vertices are equal (*isEqual*), i.e., whether they have the same id's.

sort **channel**
operations make vertex × vertex → channel
initialVertex: channel → vertex
finalVertex: channel → vertex
isEqual channel × channel → boolean
upStream: channel × vertex → boolean
downStream: channel × vertex → boolean
variables c1: channel; v1, v2, v3, v4: vertex;
axioms initialVertex (make (v1, v2)) == v1
finalVertex (make (v1, v2)) == v2
upStream (c1, v1) == finalVertex (c1) = v1
downStream (c1, v1) == initialVertex (c1) = v1
isEqual (make (v1, v2), make (v3, v4)) ==
vertex.isEqual (v1, v3) and vertex.isEqual (v2, v4)

3.2 Channel Connections

In a river network, channels can be connected in various ways. The constraint between two consecutive channels is that their flow direction is the same. In terms of the graph model, two arcs, e_1 and e_2, may join if they have complementary vertices, i.e., either *initial* (e_1) = *final* (e_2) or *final* (e_1) = *initial* (e_2).

In order to have connections between channels, channels must be part of a *network*. Such a network is built iteratively by adding channels. Channels are connected through common vertices. These vertices can then be classified according the number of its upstream and downstream channels.

sort **network**
operations create: → network
addChannel: network × channel → network
isIn: network × channel → boolean
inDegree: network × vertex → integer
outDegree: network × vertex → integer
variables n1: network; c1: channel; v1: vertex
axioms isIn (create, c1) == false
isIn (addChannel (n1, c1), c2) ==
if isEqual (c1, c2) then return true
else isIn (n1, c2)
inDegree (create, v1) == 0
inDegree (addChannel (n1, c1), v1) ==
if upStream (c1, v1) then 1 + inDegree (n1, v1)
else 0 + inDegree (n1, v1)
outDegree (create, v1) == 0
outDegree (addChannel (n1, c1), v1) ==
if downStream (c1, v1)
then 1+ outDegree (n1, v1)
else 0 + outDegree (n1, v1)

The specification will be extended throughout this paper as new features are introduced.

3.2.1 Source and Destination

The *source* is the origin of the water flow. Each river network has at least one source. In terms of the graph model, a source is a vertex of out-degree 1 and in-degree 0.

A similarly distinct river landmark is the *destination* of a network. Rivers flow either into a lake or into the sea. Each river has one destination, though there are river networks with multiple destinations such as in river deltas. In terms of a graph, a destination is a vertex that has in-degree 1 and out-degree 0.

sort **network** (cont.)
operations source: network × vertex → boolean
destination: network × vertex → boolean
axioms source (n1, v1) == inDegree (n1, v1) = 0 and
outDegree (n1, v1) = 1
destination (n1, v1) == inDegree (n1, v1) = 1 and
outDegree (n1, v1) = 0

3.2.2 Auxiliary Nodes

Auxiliary nodes are nodes whose in-degree and out-degree are 1 (Figure 4).

Figure 4: An auxiliary node and its graph representation.

Unless such nodes are specific features, such as lakes, they contain no topologically significant information.

sort **network** (cont.)
operation auxNode: network × vertex → boolean
axiom auxNode (n1, v1) == inDegree (n1, v1) = 1 and
outDegree (n1, v1) = 1

3.2.3 Junction

Two or more channels may join at a junction and merge into a single channel. In the di-graph model, a junction corresponds to a vertex of out-degree 1 and in-degree 2 or higher. The in-degree represents the upstream channels, while the out-degree is a measure for the number of downstream channels. Figure 5 shows an example of a junction and its mapping onto a di-graph.

Figure 5: Junction pattern and its graph representation.

sort **network** (cont.)
operation junction: network × vertex → boolean
axiom junction (n1, v1) == inDegree (n1, v1) ≤ 2 and
outDegree (n1, v1) = 1

3.2.4 Split

Reverse to the junction, a single channel may split into two or more separate channels. Such a situation occurs usually in a river delta or when islands are formed. In terms of the di-graph model, a split corresponds to a vertex with an in-degree of 1 and an out-degree of at least 2.

sort **network** (cont.)
operation split: network × vertex → boolean
axiom split (n1, v1) == inDegree (n1, v1) = 1 and
outDegree (n1, v1) ≥ 2

Figure 6 depicts an example of a splits and its mappings onto a di-graph.

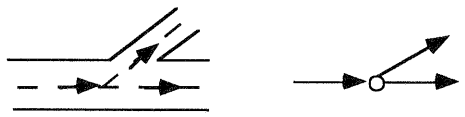

Figure 6: Split pattern and its graph representation.

3.3 Lakes

A lake is a waterbody without a flow direction. River channels carry water into and out of a lake; occasionally, lakes may have no (observable) channels associated with them.

For lakes the same classification of vertices applies, i.e., a lake may be a source, destination, junction, split, or simply an auxiliary node of a river network. In addition to these configurations, a lake may be the destination of several channels, i.e., the lake vertex is of in-degree > 1 and out-degree 0. Likewise, a lake may be the source of multiple river branches, sometimes even of different river networks. Thus, a lake may also have out-degree >1 and in-degree 0. Even the combination of in-degree 0 and out-degree 0 is feasible for a lake (without any visible channels); however, such a vertex would be an isolated vertex and, therefore, would not be part of any river network (Figure 7).

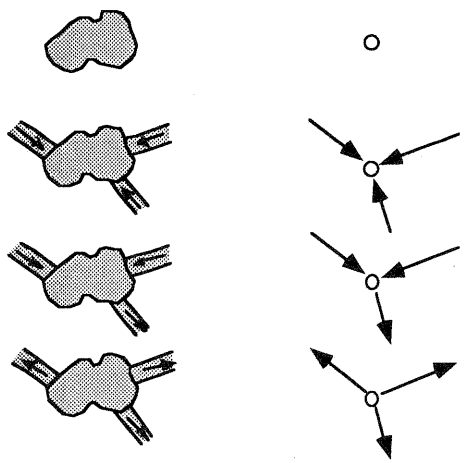

Figure 7: Configurations with a lake and their representations as graphs.

In terms of the graph representation for a river network, the degree of a vertex does not provide any clue whether the vertex is a lake or not, because a lake can have any combination of in-degree *n (0 ≤ n)* and out-degree *m (0 ≤ m)*.

3.4 Islands

An island separates a channel temporarily into two separate channels that must join later on. In terms of the di-graph model, an island is an ordered sequence of two vertices, the first of in-degree 1 and out-degree 2, and the second of in-degree 2 and out-degree 1. Figure 8 show examples of islands and their graph representations.

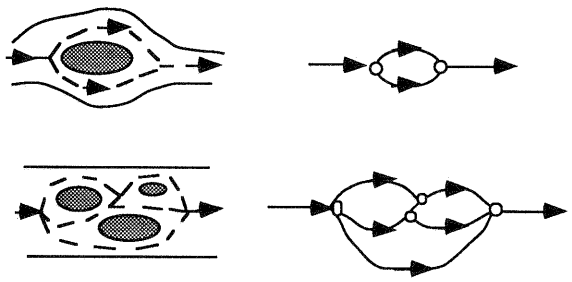

Figure 8: Configurations with islands and their representations as graphs.

A formal analysis whether two vertices, *V1* and *V2*, form an island or not has to consider the following issues (for simplicity, only nodes of degree 3 are considered, but the idea generalizes to vertices of higher degrees):

- *V1* must be a split vertex in the network *n*;
- *V2* must be a junction vertex in *n*;
- the downstream paths from the final vertices of the two downstream channels of the split *V1* must have a junction in *V2*; and
- the upstream paths from the initial vertices of the two upstream channels of the junction *V2* must have a split in *V1*.

This operation can be easily expressed as an operation on a *partially ordered set* (Birkhoff, 1967), made up by the set of vertices and the orientation of the edges between the vertices (such that downstream is ≤ and upstream ≥). In a partially ordered set, an element *u* is an upper bound of a set *A* if $a \leq u$ for all *a* " *A*. The *least upper bound* (lub) is then the smallest element in the set of upper bounds of a given set. Reversely, an element *u* is a lower bound of a set *A* if $u \leq a$ for all *a* " *A* and the *greater lower bound* (glb) is the largest element in the set of lower bounds of a given set. Applied to the identification of an island in a channel graph, the glb and lub define an island as follows:

sort **network** (cont.)
operation island: network × vertex × vertex → boolean
axiom island (n1, v1, v2) ==
split (n1, v2) and junction (n1, v2) and
(glb (finalVertex (downStreamChannel1 (n1, v1)),
finalVertex (downStreamChannel2 (n1, v1))) = v2)
and
(lub (initialVertex (upStreamChannel1 (n1, v2)),
initialVertex (upStreamChannel2 (n1, v2))) = v1)

3.5 Deltas

A delta is a split that is not followed by a junction of the downstream channels or their subsequent channels (Figure 9).

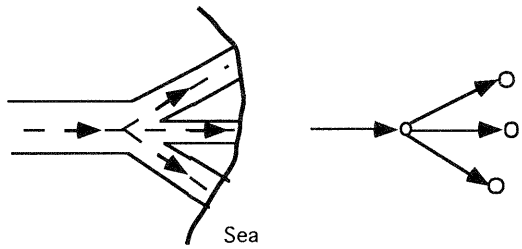

Figure 9: A river delta and its graph representation.

The analysis is similar to the process of identifying islands; however, in lieu of searching for the least upper bound, it is the goal for a delta vertex that its downstream nodes do not have a common least upper bound.

sort **network** (cont.)
operation delta: network × vertex → boolean
axiom delta (n1, v1) ==
split (n1, v1) and
(glb (finalVertex (downStreamChannel1 (n1, v1)),
finalVertex (downStreamChannel2 (n1, v1))) = {})

3.6 Channel Pattern Analysis

Table 1 shows the compilation of the features and their corresponding graph representations.

	inDegree	outDegree
source	0	1
destination	1	0
auxiliary node	1	1
junction	>1	1
split	1	>1

Table 1: Summary of channel features and their vertex degrees.

InDegree	OutDegree	Feature
0	0	lake with no inlet or outlet
1	0	destination lake with inlet
0	1	source lake with outlet
1	1	auxiliary node lake with inlet and outlet
2	0	lake with 2 inlets
0	2	lake with 2 outlets
3	0	lake with 3 inlets
2	1	junction of 2 rivers lake with 2 inlets and 1 outlet
1	2	split lake with 1 inlet and 2 outlets
0	3	lake with 3 outlets

Table 2: Classification of vertices according to their degrees.

Reasoning about these features will involve the reverse operation, deriving from a graph representation the kind of feature that made up the graph. Table 2 shows an extended "inverted" table, classifying vertices by the number of links and their flow directions, and assigning the corresponding river features. Besides the features from Table 1, the corresponding lake-river patterns are included as well.

4 Simplifications of River Graphs for Flow Inference

The goal of the inference of the flow direction is to derive such a directed graph from an ordinary graph and additional metric information about the junction angles. In order to simplify this process, a few simplifications of the directed graph are possible by removing channels (and corresponding vertices) that are not necessary for the inference process.

Removing a channel puts the network into a state as if the channel had never been inserted.

sort **network** (cont.)
operation remove: network × channel → network
axioms remove (create, c1) == create
remove (addChannel (n1, c1), c2) ==
if equal (c1, c2) then return n1
else addChannel (remove (n1, c2), c1)
isIn (remove (n1, c1), c1) == false

4.1 Elimination of Auxiliary Nodes

Auxiliary nodes, connecting exactly two channels, can be eliminated, because they contain no significant information from which the flow direction can be inferred (Figure 10).

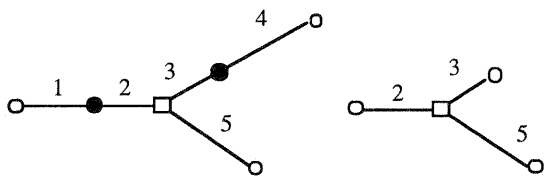

Figure 10: Simplification by eliminating auxiliary nodes.

After removing the upstream and downstream channel from an auxiliary node, the simplified channel, preserving the connectivity and the flow direction, must be inserted.

sort **network** (cont.)
operation merge: network × channel × channel → network
axioms merge (c1, c2) == error if
finalVertex (c1) <> initialVertex (c2)
merge (c1, c2) == error if
(inDegree (finalVertex (c1) + outDegree (c1))) > 2
merge (n1, c1, c2) ==
if not (isIn (n1, c1) and isIn (n1, c2))
then return n1
else addChannel (make
(initialVertex (c1), finalVertex (c2))),
remove (n1, c1), remove (n1, c2).
initialVertex (merge (n1, c1, c2)) ==
initialVertex (c1).
finalVertex (merge (n1, c1, c2)) == finalVertex (c2).

4.2 Elimination of Islands

Islands are features that are irrelevant for the assessment of the flow direction of the river network and their existence would make the reasoning process more difficult. Since an island consists of an ordered sequence of a split and a junction, for a single island between two nodes *V1* and *V2*, the two downstream channels from *V1* to *V2* can be replaced by a *single* channel from *V1* to *V2* (Figure 11).

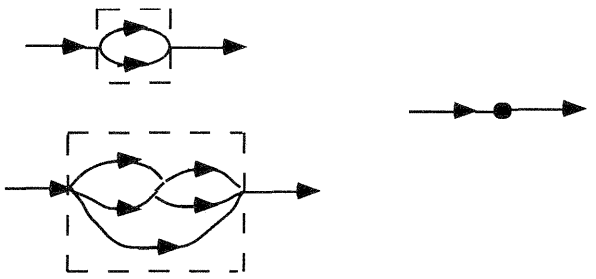

Figure 11: Simplification by eliminating islands.

This assumes that auxiliary nodes between *V1* and *V2* have been eliminated before.

sort **network** (cont.)
operation elimIsland: network × vertex × vertex → network
axioms elimIsland (n1, v1, v2) ==
error if not island (n1, v1, v2)
else remove (n1, downStreamChannel1 (n1, v1))

More complex is the issue if each channel along the island cannot be simplified, because both contain further junctions. In such cases, before eliminating a channel, its junctions have to be incorporated into the other branch. Since junctions on opposite sides of islands are partially ordered, it is impossible to decide which junction should come first and a random choice has to be made. For the inference of the flow directions, such simplifications should not matter.

5 Conclusions

This paper investigated the formalization of river networks. Such a formalization is necessary as a first step in the development of formal reasoning methods about river networks, e.g., to infer the flow direction. We have shown how the junction patterns in a river network can be mapped onto a directed acyclic graph. Irrelevant features, such as auxiliary nodes and islands, can be removed from the graph to make to simplify the inference process.

While the classification of nodes based on their in-degrees and out-degrees is powerful, it may occasionally need user interaction. For example, channels may be hidden so that they do not appear in the data source. Such channels may be running naturally under ground, primarily in karst regions, or they may be hidden from the data collector by such obstacles as overhanging trees. Another possibility for channels being invisible is that the resolution of the data collector is too low to capture narrow waterways. In all cases, the natural flow of water continuos while the observed network is interrupted.

6 Acknowledgments

Thanks to David Mark and Diógenes Alves for their valuable comments.

7 References

L. Band. 1986. Analysis and Representation of Drainage Basin Structure with Digital Elevation Data, in: D. Marble (ed.) Proceedings of the Second International Symposium on Spatial Data Handling, Seattle, WA, pp. 437-450.

G. Birkhoff. 1967. *Lattice Theory.* American Mathematical Society, Providence, RI.

W. Chase and M. Chi. 1981. Cognitive Skill: Implications for Spatial Skill in Large-Scale Environment. in: J. Harvey (ed), *Cognition, Social Behavior, and the Environment.* Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 111-136.

M. Egenhofer and J. Herring. 1991. High-Level Spatial Data Structures for GIS, in: D. Maguire, M. Goodchild, and D. Rhind (eds.), *Geographical Information Systems: Principles and Applications*, Volume 1, pp. 227-237, Longman, London.

A. Frank and D. Mark. 1991. Language Issues for GIS, in: D. Maguire, M. Goodchild, and D. Rhind (eds.), *Geographical Information Systems: Principles and Applications*, Volume 1, pp. 147-163, Longman, London.

A. Frank, B. Palmer, and V. Robinson. 1986. Formal Methods for the Accurate Definition of Some Fundamental Terms in Physical Geography, in: D. Marble (ed.) Proceedings of the Second International Symposium on Spatial Data Handling, Seattle, WA. pp. 583-599.

A. Gill. 1976, *Applied Algebra for the Computer Sciences.* Prentice-Hall, Englewood Cliffs, NJ.

R. Haralick, S. Wang, L. Shapiro, and J. Campbell. 1985. Extraction of Drainage Networks by Using the Consistent Labeling Technique. *Remote Sensing and Environment*, 18:163-175.

R. Horton. 1945. Erosional Development of Streams and their Drainage Basins: Hydrophysical Approach to Quantitative Morphology. *Geological Society of America Bulletin*, 56:275-370.

D. Knuth, 1973, *The Art of Computer Programming*, Vol. 1 Fundamental Algorithms, Addison-Wesley, Reading MA.

R Larovere and S. Goodman. 1992. Computing in the Brazilian Amazon. *Communications of the ACM*, 35(4):21-24.

D. Mark and M. Goodchild. 1982. Topologic Model for Drainage Networks with Lakes. *Water Resources Research*, 18(2):275-280.

M. Melton. 1959. A Derivation of Strahler's Channel-Ordering System. *Journal of Geology*, 67:345-346.

NCGIA, 1992, *NCGIA Update*, 4(1):5-6, National Center for Geographic Information and Analysis, University of California, Santa Barbara.

J. O'Callaghan and D. Mark. 1984. The Extraction of Drainage Networks from Digital Terrain Data. *Computer Graphics and Image Processing*, 13:323-344.

B. De Serres and A. Roy. 1990. Flow Direction and Branching Geometry at Junctions in Dendritic River Networks. *The Professional Geographer*, 42(2):194-201.

R Shreve. 1966. Statistical Law of Stream Numbers. *Journal of Geology*, 74:17-37.

R Shreve. 1967. Infinite Topologically Random Channel Networks. *Journal of Geology*, 75:178-186.

R. Souza *et al.* 1992. Spring: An Object-Oriented Geographic Information System, Technical Report, National Institute for Space Research (INPE), São José dos Campos, SP, Brazil.

A. Strahler. 1952. Hypsometric (Area-Altitude) Analysis of Erosional Topography. *Geological Society of America Bulletin*, 63:1117-1142.

S. Wang, D. Elliott, J. Campbell, R. Erich, and R. Haralick. 1983. Spatial Reasoning in Remotely Sensed Data. *IEEE Transactions on Geoscience and Remote Sensing*, GE-21:94-101.