Specifying Essential Features of Street Networks

Simon Scheider[1], Daniel Schulz[1]
[1] Fraunhofer Institut für Intelligente Analyse und Informationssysteme (IAIS), Schloss Birlinghoven,
53754 Sankt Augustin, Germany
{Simon.Scheider, Daniel.Schulz}@iais.Fraunhofer.de

Abstract. In order to apply advanced high-level concepts for transportation networks, like hypergraphs, multi-level wayfinding and traffic forecasting, to commercially available street network datasets, it is often necessary to generalise from network primitives. However, the appropriate method of generalisation strongly depends on the complex street network feature they belong to. In this paper, we develop formal expressions for road segments and some essential types of roads, like roundabouts, dual carriageways and freeways. For this purpose, a formal network language is developed, which allows a clear distinction among the geometrical network, its embedding into the Euclidian plane, as well as navigational constraints for a traffic mode.

Keywords: spatial network features, road typology, formal specification, network generalisation, embedded graphs.

1 Introduction

Nowadays digital street network data plays an important role in all kinds of businesses related to transport and navigation. One prominent example are navigation devices, which use network graphs together with map matching and shortest path algorithms to navigate a vehicle on a physical network. Another example are topographical maps showing the street infrastructure at different scales, which have to be adapted to different generalisation levels. A third example are traffic forecasting models, which can be used for planning purposes or to assess traffic frequency or traffic quality on a given location in the street network.

There exist only a small number of data providers for digital street data. The data model usually consists of an embedded (usually non-planar) graph, which is a graph whose edges and nodes have some representation in the Euclidian plane and which may be overlapping, and a set of additional street attributes, like e.g. road category and feature identifiers like street name. Of course, each vendor has its own format, and each format has its own set of street attributes, whose quality is often unknown.

However, for probably all kinds of transport network applications, network attributes and feature identifiers are crucial bits of information. This is because transportation modelling obviously is only feasible with complicated relational structures built on the usual network primitives [4]. This is documented by the big amount of research on high-level logical network structures [11]. For example in order to solve the task of wayfinding on a highway network, Timpf et al. [8] [9] distinguished among several task levels (route planning, wayfinding instructions, driving instructions), whose logical primitives are complex network features like e.g. freeway exits. In the research area related to transport GIS, Mainguenaud [5] introduced multi-level hypergraph networks that allow master nodes and master edges to represent sub-networks. Additionally, the fact that a given physical network infrastructure can be used by more than one tansport mode, leads to the concept of feature-based virtual networks [10], in which a second graph is built on the primitives of a street network graph. This graph abstracts from unnecessary geometrical and logical details for a certain transport mode.

However, in order to build these high-level data structures from a commercial dataset, given attributes are often not sufficient. Therefore it would be useful to have formal network generalisation techniques available, which could be used to automatically derive certain high-level features from a dataset of primitives. There have been important conceptual contributions to this problem, e.g. Stell and Worboys [7] and Stell [6] introduced a formalisation of simplification methods for spatial networks and graphs in general. As we will illustrate by two examples in section 2 of this paper, the appropriate method of generalisation for a given subset of the network is nevertheless strongly dependent on the type of complex network feature this subset belongs to. These can be different types of roads and different types of junctions.

In this paper, we develop formal expressions for road segments and some essential types of roads, like roundabouts, dual carriageways and freeways. In this scenario it is not sufficient to account only for logical characteristics of a street network, it is also necessary to take into account geometrical properties of its Euclidian embedding, as well as navigational constraints for vehicle traffic.

In the following sections, we will first consider two common examples of generalisation tasks for a street network, and discuss the main challenge they pose to a formal treatment using the existing concept of simplification by Stell and Worboys. Then we will introduce a formal specification of a street network, integrating physical network characteristics with navigational constraints for vehicle traffic on this network. On this basis we introduce definitions for road segment types and types of roads. We conclude with an outlook to future work.

2 Two examples of generalisation tasks

The data model of a spatial street network dataset commonly consists of street segments and their intersections. Their logical equivalents are edges and nodes of a graph. The central question is, if this graph representation is enough for a formal treatment of common generalisation tasks.

Stell and Worboys [7] introduced a formal concept for the simplification of general graphs, which is considered as a combination of “amalgamation” and “selection” operations. A “selection” of a graph $G = (N, E)$ is an operation that takes a subset of nodes $N' \subset N$ and produces a derived graph $G' = (N', E')$, with each edge of E' being a possible path between two nodes of N' in the original graph G. In this way, nodes of the original graph can be removed, but the connecting paths between all remaining nodes still exist as an edge of the derived graph. Amalgamation means to combine nodes of the original graph by a surjective function, so that these nodes and their connecting edges can collapse into a new node, and edges being connected to a remaining node collapse into one new edge.

The first example we consider is the network data model of a roundabout junction linking 3 roads together (figure 1 on the right). If we try to apply the simplification concept to the generalisation of a roundabout junction, it would make sense to have a single amalgamation operation that combines all nodes m, q, r, n, p, o that are part of the loop into exactly one new node. All edges a, b, c, d, e, f belonging to this loop would disappear, and the remaining node would be connected to all three remaining edges entering the junction.

The second example is a dual carriageway road having an intersection with a second road (figure 1 on the left and figure 2). Here, we would rather apply first a selection operation, which removes the nodes o and q, leaving the nodes m, n, p and r together their two connecting paths [m a o b p] , [p c n] and [m f q e r] , [r d n]. In a second step, we would like to amalgamate the two nodes p and r to the new node u, so that the graph consists of the three nodes m, u and n and a new linear feature evolves consisting of the new edges j and k. In this case, the generalisation operation translates a dual carriageway to a bidirectional road, consisting of two segments, and one intersection at the new node u.

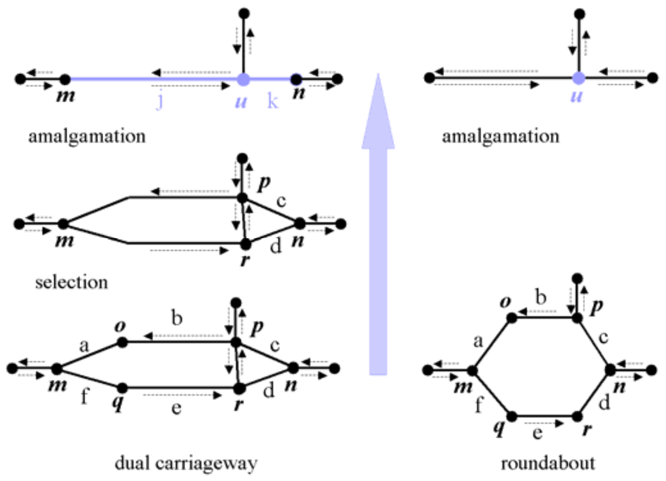

Fig. 1. Simplification of two embedded graphs, one being a common data model of a dual carriageway, one being a common data model of a roundabout. The simplification method and its result are dependent on the complex feature type: The simplified feature is 1-dimensional in the case of a dual carriageway, and 0-dimensional in the case of a roundabout.

For a formal treatment, the problem now is how to distinguish between these cases? We could apply the appropriate amalgamation and selection operations if we could identify the sub graphs. Unfortunately, there are no appropriate attributes in the original dataset of the vendor which describe such complex types of roads and intersections. Additionally, note that the logical graph in both cases is nearly equal, so a distinction of nodes and edges on this logical level seems to be impossible. For example how can we distinguish nodes m and n from nodes p and r in the case of a dual carriageway, which is necessary in order to apply the proposed amalgamation operation?

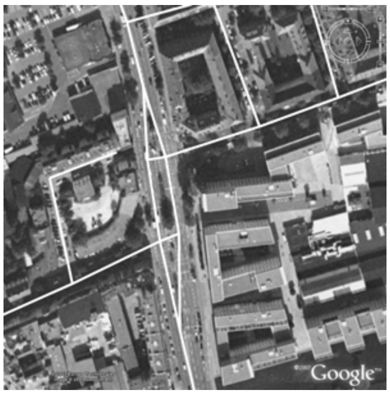

Fig. 2. A dual carriageway road intersecting with two normal roads. Geometries (white lines) of a commercial network dataset are shown against the background of an aerial view.

This example underlines our principal hypothesis that a formal specification of complex network features is not possible only on the logical graph level commonly used to describe network data models. The formal apparatus introduced in the next section is an attempt to provide an integrated specification language for this purpose. In this language we can clearly distinguish among 3 aspects: the geometrical network graph, its embedding into the Euclidian space, and a second graph reflecting navigational constraints for a type of traffic.

3 Street networks

3.1 Some notes on the formal notation

In the following, data types are expressed by upper case symbols, denoting types of a typed higher order logic. Data structures and operations are algebraic expressions of this language. Syntax and semantics are equivalent to the "Basic Extended Simple Type Theory" (BESTT) approach (compare Farmer [2] and Farmer and Mohrenschmidt [3]), so that each type stands for a set. There are three atomic types with an obvious semantic: "Bool" for boolean values, "Rat" for rational numbers and "INT" for integers, and other atomic types are introduced in section 3.2. Derived types can be constructed from atomic ones by combining them to product, function, set and list types, meaning their appropriate set theoretic combinations (compare table 1). Let A and B be any two types. In this formal approach, every expression “a” has a type “A”, which means they can always be written together like this (a:A).

Expressions can be formed by typed variables, constants, function abstraction (written as “λ[variable].[expression]“) and function application (written as “[function]([expression])“), so that function arguments are of the appropriate type. Some language constants with obvious meaning are used, like:

$\downarrow$: A→Bool (:=definedness), toset: A×…×A→set[A] (:=tuple to set projection), ||:set[A]→Rat (:=set cardinality), ||: list[A]→Rat (:=list size), #i: $A_0\times\ldots, A_i,\ldots\times A_j\rightarrow A_i$ (:= extracts the i^{th} element of a tuple), [i]: list[A] →A (:=extracts the i^{th} element of a list), if: Bool×A×A→A (:=conditional expression), I: (A→Bool)→A (:= finds a unique object of type A by a predicate (definite description)), as well as well known logical and arithmetic symbols. The function “I“ can be used to select unique objects with the help of its predicate argument.

Table 1. The four possible type constructors of a BESTT language. Let A and B be any types.

Type constructors	Comments
A×B	Product type
A→B	Function type
set[A]	Set type
list[A]	List type

The semantics of a BESTT language is strict: Every domain has an error element for undefined functions, and errors are automatically propagated. Therefore expressions are always defined in a classical sense. If an expression cannot be evaluated, it returns the error element, and the application of operator “$\downarrow$“ will return false.

3.2 Specifying the geometrical street network

In this section, we introduce an abstract specification of a typical data model of the physical street network infrastructure, as implemented in commercially available navigational maps like “Navteq” and “Teleatlas”. In table 2, we introduce street segments (lines) of type L and nodes of type N together with their Euclidian embedding of type set[P] as atomic elements of road geometries, which can be seen as equivalents to “point” and “line” feature types of ISO-GDF.

Table 2. The three basic types of a spatial street network.

Basic Types	Comments
N	Network nodes
L	Street segments (lines)
P	2-dim Euclidian space := Rat×Rat

Axiom 1. A street line (type L) has an embedded linear curve (type set[P]) that is not geometrically self-intersecting.

Let P be the type of a tuple of spatial coordinates in the 2-dimensional Euclidian space. According to axiom 1, a geometric representation of a line is a set of points (type set[P]), meaning a geometrical curve that does not cross itself in the 2-D space, equivalent to a “simple line” in [1]. In table 3, we give some basic operations for these types. Their axiomatic specification is omitted in this paper for simplicity reasons.

Table 3. Some basic operations for a spatial street network.

Basic operation	Of type	Comments
line	$L \rightarrow set[P]$	A function embedding street segments (lines)
point	$N \rightarrow P$	A function embedding street nodes
border	$L \rightarrow N \times N$	A function returning the two end nodes of a street segment
path	$list[L] \rightarrow Bool$	Predicate to identify paths of node-connected street segments
cycle	$list[L] \rightarrow Bool$	Predicate to identify closed paths of node-connected street segments
nodes	$list[L] \rightarrow list[N]$	A function returning the list of nodes of a path. If the path is a cycle, all nodes will be returned, if the path is unclosed, all nodes without start and end node are returned
next	$list[L] \times INT \rightarrow INT$	Iterator for looping in a cycle
$region_{inner/outer}$	$list[L] \rightarrow set[P]$	A function returning the inner/outer region for a list of lines being a simple cycle

Let “line”: $L \rightarrow set[P]$ be the total function embedding every line as a curve into the 2-dimensional Euclidian plane. Let “border“: $L \rightarrow N \times N$ be the total function that produces the tuple of two end nodes for a line, so that their embeddings point(#1(border(h:L))) and point(#2(border(h:L))) are the two bordering points on the line's curve.
Lines can be seen as the atomic spatial entities of street networks, being logically and geometrically connected exclusively at their bordering nodes. They can be considered as edges of an embedded street network graph. The “path” predicate ensures that a given list of lines is a path of this graph. A street network graph must comply with the following axioms:

Axiom 2. A street network is a connected graph of street segments of type L (edges) and nodes of type N (vertices). This means that every pair of vertices of the street network is connected through a path.

Axiom 3. Two lines of the street network geometrically intersect either in the point of their common end node (planar “border” intersection), or by crossing each other non-planarly.

For our purpose, in the remainder we will introduce special operations as typed expressions (numbered formulas) and specify them by definitions. In order to increase legibility, we will sometimes add some comments to the defining expressions in italic.

Axiom 3 has the consequence that the embedding of the street network graph is not necessarily planar. This means there can be "road intersections" as well as "bridge-tunnel" relationships between lines. A "planar" predicate is given by:

$$\text{planar} : \text{set}[L] \rightarrow \text{Bool} . \tag{1}$$

Definition 1.

$\forall$ k:set[L]. planar(k) =

```
    ∀(e_1:L), (e_2:L)∈k. e_1 ≠ e_2 ⇒
        line(e_1) ∩ line(e_2) =
        if(     toset(border(e_1)) ∩ toset(border(e_2))≠∅,
                point(I(n:N).toset(border(e_1))∩toset(border(e_2)) = {n}),
                ∅
        )
```

Additionally, there are two special kinds of paths: simple paths and simple cycles. Simple paths are not closed, which means the end nodes of the path are not equal, and they are "simple", which means not self-intersecting. This is ensured, if they have a planar embedding and their list of nodes is unique, which means there is not any node occurring more than once in the list.

$$\text{simplepath: list}[L] \rightarrow \text{Bool} . \tag{2}$$

Definition 2.

$\forall$ k:list[L]. simplepath(k) =

```
    path(k) ∧ planar(toset(k)) ∧
    toset(border(k[|k|-1])) ∩ toset(border(k[0])) = ∅  ∧   //unclosed path//
    |toset(nodes(k))| = |nodes(k)|          //each node occurs just once in the path//
```

A simple cycle is a closed simple path.

$$\text{simplecycle : list}[L] \rightarrow \text{Bool} . \tag{3}$$

Definition 3.

$\forall$ k:list[L]. simplecycle(k) =

```
    cycle(k) ∧ planar(toset(k)) ∧
    |toset(nodes(k))| = |nodes(k)|          //each node occurs just once in the path//
```

For a simple cycle, the function $\text{region}_{\text{inner/outer}} : \text{list}[L] \rightarrow \text{set}[P]$ (compare table 3) is always defined. It returns the expected 2-dimensional region without holes and without bordering lines, equivalent to the interior/exterior of a cell complex [1].

3.3 Traffic on a street network

The embedded graph introduced in the last section is undirected. This is because different kinds of traffic on the same geometrical network infrastructure can have multiple directional constraints. In other words, the street network graph provides an infrastructure for different kinds of traffic networks, vehicular, pedestrian or bus transport, with its specific kinds of navigational constraints.

Navigation is a sequence of directional decisions. A directional decision in the proposed street network can only be taken at a common end-node (n:N) of two or more lines (k_1:L, k_2:L), $n \in \text{toset}(\text{border}(k_1)) \wedge n \in \text{toset}(\text{border}(k_2))$. This is because junctions by definition exist only at border nodes (axiom 3).

A transport network on the proposed street network infrastructure therefore can be expressed as a second, directed graph, whose edges (“navigation edges”) express navigational rules. They connect one line to another iff turn off at a street network node is allowed. Normally, in navigation applications, it is convenient to specify a single “navigation node” for either side of a street, which means for each direction of driving. For reasons of simplicity, we specify navigation nodes of such a graph to be just street lines of type L, with the consequence that for navigation paths, we ignore the risk of possibly unwanted u-turns at bidirectional lines. Further on, the first line of a navigation edge will be called the “from” line, the second one the “to” line.

In order to express a navigation graph, first a “maximal directed navigation graph” can be extracted from the street network graph. This is the special navigation graph for the case that all geometrically possible turn offs at navigation nodes are allowed.

$$\begin{aligned}&\text{navigraph}_{\text{max}}:\text{set}[L\times L\times N]\\ &:=\{t : L\times L\times N \mid \text{toset}(\text{border}(\#1(t))) \cap \text{toset}(\text{border}(\#2(t)))= \{\#3(t)\}\} .\end{aligned} \tag{4}$$

In the scope of this paper, we will focus on vehicle traffic. A navigation graph for vehicle traffic is simply the subset of “navigraph_max” for which vehicle turn-off is not prohibited.

$$\begin{aligned}&\text{navigraph} :\text{set}[L\times L\times N]\\ &:= \{t : L\times L\times N \mid t\in \text{navigraph}_{\text{max}} \wedge \neg\text{prohibited}(t)\} .\end{aligned} \tag{5}$$

In the remainder, if we refer to a navigation graph of type set[L×L×N], we always mean the navigation graph for vehicle traffic in formula 5. Let us call two lines (vehicle-) “navigable”, if their tuple occurs in the (vehicle-) navigation graph. For navigation graphs, a single axiom is necessary. It ensures that for every line of it, there must be at least two navigation edges, one for incoming and one for outgoing traffic. This means the line is always a “from”-line as well as a “to”-line:

Axiom 4.

$\forall(k{:}L)\in\{k \mid \exists t \in \text{navigraph}.\ k=\#1(t) \vee k=\#2(t)\}.$
$\quad \exists t_1, t_2\in\text{navigraph}.\ t_1\neq t_2 \wedge$
$\qquad k\in\{\#1(t_1), \#2(t_1)\} \wedge k\in\{\#1(t_2), \#2(t_2)\} \wedge$
$\qquad (\#1(t_1) = \#2(t_2) \vee \#2(t_1) = \#1(t_2))$

The axiom prevents so called "graveyards" and "factories", which are lines allowing traffic to enter without giving it a chance to leave them anymore, and vice versa. These anomalies can cause several undesired effects in navigation devices. Graveyards and factories can exist e.g. in the context of spatially separate entering and exiting roads of a parking garage. In these graveyards, traffic exits implicitly exist inside of the garage, but they are not explicitly modelled in the network. These exceptions usually need to be dealt with explicitly. For the sake of simplicity we will not consider them in our specification.

The essential operation for navigation graphs is a shortest path operation. It can easily be formalised without an algorithmic definition. Let "cost": list[L]$\rightarrow$Rat be a total function assigning a sum of costs to every path of lines. Let "min": list[L]$\times$ set[L]$\times$set[L$\times$L$\times$N]$\times$ (list[L]$\rightarrow$Rat) $\rightarrow$ Bool be a predicate that ensures a given path on a given line subset and for a given navigation graph to be minimal according to a given cost function. Then a shortest path operation for two lines k_1 and k_2 and an arbitrary subset "sn" of street network lines can be specified like this:

$$\text{shortest\_path: set}[L\times L\times N]\times\text{set}[L]\times L\times L \rightarrow \text{list}[L]\ . \qquad (\mathbf{6})$$

Definition 4.

$\forall\ ng{:}\text{set}[L\times L\times N], sn{:}\text{set}[L], k_1{:}L, k_2{:}L.\ \text{shortest\_path}(ng, sn, k_1, k_2) =$
$\quad I(o{:}\text{list}[L]).\ (\forall u\in \text{toset}(o).u\in sn) \wedge k_1 = o[0] \wedge k_2 = o[|o|-1] \wedge$
$\qquad (\forall i\in\{0,\ldots,|o|-2\}.\exists t\in ng.\#1(t) = o[i] \wedge \#2(t)=o[i+1]) \wedge$
$\qquad \min(o, sn, ng, cost)$

An operator complying to definition 4 will return the "undefined" expression, if there does not exist any navigable path between the two lines k_1 and k_2 in the subset "sn", and otherwise, it will return a list of lines being the shortest possible one.

4 Features of a street network

4.1 Types of road segments

Now, important subtypes of lines can be specified according to their navigational characteristics by introducing predicates accordingly.

Dead line. Lines of the street network that are not part of the tuple of at least one navigation edge do not belong to the transport network of the respective traffic type. These lines can be specified with the predicate “dead”:

$$\text{dead} : \text{set}[L \times L \times N] \times L \rightarrow \text{Bool} . \qquad (7)$$

Definition 5.
$\forall ng{:}set[L \times L \times N], k{:}L.\ dead(ng, k) = \neg\exists t \in ng.\ k = \#1(t) \vee k = \#2(t)$

Dead end. Lines are called “deadend”, if at least one of the two end nodes are not part of any navigation edge. Consequently, “dead” lines are special kinds of dead ends, which means $dead(ng, k) \rightarrow deadend(ng, k)$:

$$\text{deadend} : \text{set}[L \times L \times N] \times L \rightarrow \text{Bool} . \qquad (8)$$

Definition 6.
$\forall ng{:}set[L \times L \times N], k{:}L.\ deadend(ng, k) = \exists n \in toset(border(k)).\neg\exists t \in ng.\ n = \#3(t)$

One-way. If a line is exclusively a “from”- or a “to”- line for all navigation edges at one end node, then it is a “one-way” street segment:

$$\text{oneway} : \text{set}[L \times L \times N] \times L \rightarrow \text{Bool} . \qquad (9)$$

Definition 7.
```
$\forall ng{:}set[L \times L \times N], k{:}L.\ oneway(ng, k) =$
    $\neg deadend(ng, k) \wedge$
    $(\forall n \in toset(border(k)).$
        $(\exists t \in ng.\ k = \#1(t) \wedge n = \#3(t) \rightarrow \neg\exists\ t \in ng.\ k = \#2(t) \wedge n = \#3(t)) \vee$
        $(\exists t \in ng.\ k = \#2(t) \wedge n = \#3(t) \rightarrow \neg\exists\ t \in ng.\ k = \#1(t) \wedge n = \#3(t))$
    )
    //both end nodes of a line can be passed in only one direction//
```

In a common sense, a line is a one-way road segment if it can be passed through in only one direction. This means that one node is only an exit, and the other node is only an entrance. This is the case, if the line is not a dead end, so both nodes can be passed, and if both nodes can be passed in only one direction. Then one node must be exit and the other one must be entrance because of axiom 4, therefore definition 7 is correct.

Bidirectional line. Bidirectional road segments are lines that are not “dead” or “one-ways”.

4.2 Types of roads and their intersections

According to a common understanding, a road is an identifiable route between two or more places. In our formalisation, a road is a complex linear geometrical feature that is a path of lines. Furthermore a road is considered to be a simple path, because it normally does not cross itself. Another essential characteristic of a road is that it must be "identifiable". This means each street segment must uniquely belong to only one road, and therefore roads logically cannot overlap. Definition 8 states that a "road system" is a set of simple paths of lines (roads), and each line must belong to only one road:

$$\text{roadsystem} : \text{set}[\text{list}[L]] \rightarrow \text{Bool} . \qquad \textbf{(10)}$$

Definition 8.
$\forall$ rs:set[list[L]] . roadsystem(rs) =
$(\forall(r:\text{list}[L])\in rs.\text{simplepath}(r))$
$\wedge$
$(\forall(k:L).\ \exists r\in rs.\ k\in \text{toset}(r) \Rightarrow (\neg\exists r_2\in rs.\ r\neq r_2 \wedge k\in \text{toset}(r_2)))$

Normal roads. A normal road can be any simple path of lines in the street network graph that is not a ring. In contrary to a street, it is necessary that one can drive on a road from one start to one end. A normal road allows for every kind of intersection with other roads:

$$\text{normal\_road} : \text{set}[L\times L\times N], \text{list}[L] \rightarrow \text{Bool} . \qquad \textbf{(11)}$$

Definition 9.
$\forall$ ng:set[L×L×N], k:list[L]. normal_road(ng, k) =
simplepath(k) $\wedge$
$\forall i\in\{0,\ldots,|k|-2\}.\exists t\in ng.\#1(t)=k[i] \wedge \#2(t)= k[i+1]$
//the normal road can be passed through in at least one direction//

Linear rings. Linear rings are just simple geometrical loops. A simple loop is a simple path of street lines whose end nodes are connected. On a linear ring, it must be possible to drive one way around. Linear rings also allow for every kind of intersection:

$$\text{linear\_ring} : \text{set}[L\times L\times N], \text{list}[L] \rightarrow \text{Bool} . \qquad \textbf{(12)}$$

Definition 10.
$\forall$ ng:set[L×L×N], k:list[L]. linear_ring(ng, k) =
simplecycle(k) $\wedge$

$\forall i \in \{0,\ldots,|k|-1\}.\exists t \in ng.\#1(t)=k[i] \wedge \#2(t)= k[next(k,i)]$
//the linear ring can be passed through in at least one direction//

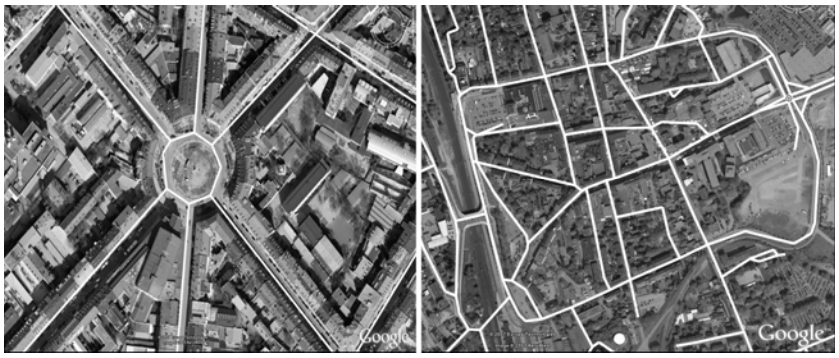

Fig. 3. A roundabout (left) and a linear ring (right) enclosing a housing area.

Roundabouts. For a roundabout, further navigational and geometrical constraints have to be considered. A linear ring is a roundabout, if it is a simple loop of one-way lines that can only be passed through in one (e.g. anticlockwise) direction. Whether a roundabout can be passed through clockwise or anticlockwise is dependent on the directionality rule.

So far, a roundabout could not be distinguished from a city block having an appropriate configuration of surrounding one-way streets, or from an appropriate linear ring surrounding a city center. In order to distinguish them, geometrical properties have to be taken into account. The principal idea of a roundabout is that of a special kind of road junction. Therefore the inner region formed by the loop is not usable for street infrastructure or housing infrastructure, because this would complicate the traffic situation inside of the junction. This means that street lines lying "inside" of the loop and being logically connected with the loop are not allowed, as well as buildings lying inside of the loop.

We specify a roundabout to be a linear ring that does not enclose any connected lines inside of the region formed by the loop. Buildings are not considered for simplicity reasons:

$$\text{roundabout} : \text{set}[L\times L\times N]\times \text{list}[L] \rightarrow \text{Bool} . \qquad (13)$$

Definition 11.
$\forall$ ng:set[L×L×N], k:list[L]. roundabout(ng, k) =
linear_ring (k) $\wedge$ ($\forall i \in$toset(k).oneway(ng,i)) $\wedge$
//the linear ring can be passed through in exactly one (counter clockwise or clockwise) direction//
$\neg\exists$(o:list[L]). path(o) $\wedge$
($\exists$(m:L)$\in$toset(o), (n:L)$\in$toset(k).

$\text{toset}(\text{border}(m)) \cap \text{toset}(\text{border}(n)) \neq \emptyset) \wedge$
$(\forall(m{:}L) \in \text{toset}(o).\ \text{line}(m) \cap \text{region}_{\text{inner}}(k) \neq \emptyset)$
//there is no logically connected path of lines lying inside of the loop//

A roundabout must have at least one external access point because of axiom 2. A roundabout with exactly one access point can be called a “turnaround circle” (German: “Wendehammer”).

Dual carriageways (closed). If a path is a “closed dual carriageway”, then it consists of two parallel, non-crossing one-way passages, each of them carrying traffic in one direction, and ultimately being connected to one node at both ends of the carriageway. The nodes provide exit from as well as entrance to it. This means that all lines of the road must be one-way lines and must form a simple loop. But in contrary to a roundabout, there are exactly two special nodes with pairs of connected lines in the loop where drive through is not permitted. We call this intersection feature “diametrical bifurcation” and it has some characteristics which can be used to distinguish a dual carriageway from a roundabout. A diametrical bifurcation (compare figure 5) consists of two one-way lines and their common end node. Navigation is not allowed from one line to the other crossing their common node, although the node is an entrance for traffic to one line and an exit for the other one:

$$\text{diabifurcation} : \text{set}[L\times L\times N]\times N\times L\times L \rightarrow \text{Bool}\ . \tag{14}$$

Definition 12.
$\forall\ ng{:}\text{set}[L\times L\times N], (n{:}N), (k_1{:}L), (k_2{:}L).\ \text{diabifurcation}(ng, n, k_1, k_2) =$
$k_1 \neq k_2 \wedge \text{oneway}(ng, k_1) \wedge \text{oneway}(ng, k_2) \wedge$
$(\neg\exists t \in ng.\#3(t)=n \wedge \{\#1(t),\#2(t)\} = \{k_1, k_2\}) \wedge$
$(\exists t_1, t_2 \in ng.\ \#3(t_1)=n \wedge \#3(t_2)=n\ \wedge$
$((\#1(t_1)= k_1\ \wedge \#2(t_2) = k_2) \vee (\#1(t_1)= k_2\ \wedge \#2(t_2) = k_1))$
$)$
//at their common node, driving through both one-way lines is not allowed, but exit from one line and entrance to the other line is possible//

Because of axiom 4, a bifurcation node must be connected to at least one third line, allowing for entrance as well as exit to the dual carriageway. In consequence, passage in a dual carriageway is possible from one bifurcation node to the other on all line tuples in between them. These two separately navigable passages are called “lanes” and provide exactly opposite driving directions.

If there were not any junctions between dual carriageways, our specification could be finished now. But of course, dual carriageways can have logical junctions with other ones. In principle, this could have the consequence that a line subset of one dual carriageway could appear also as a part of other dual carriageways (compare figure 4), if they are connected by junctions. This is prevented by the road system definition 8. Nevertheless, one can think of many configurations. In order to ensure the “right” configuration of dual carriageways, it is necessary to formalise the principal types of node intersections at junctions that are allowed for such a road type, taking also into account their geometrical characteristics.

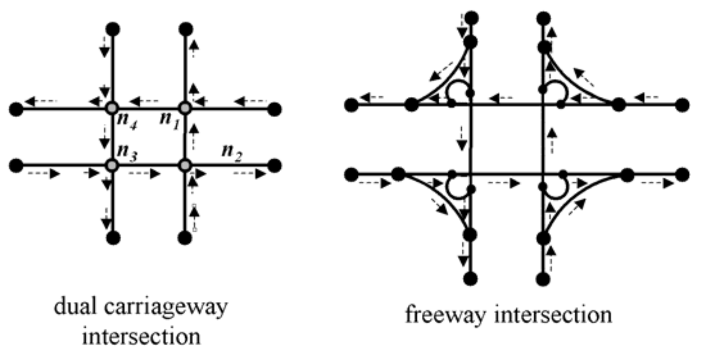

Fig. 4. Two examples of junction types for dual carriageways and freeways.

For this purpose it is first necessary to consider all possible intersection nodes that allow to exit from the loop of one dual carriageway and to enter another one. A junction between two closed dual carriageways always means an intersection between two closed loops of one-way roads, so all intersecting lines must be one-way lines. The points of geometrical intersection can be logical, in which case exactly four 4-valued nodes appear, called “crossing intersections” (see nodes 1-4 in figure 4 and node 1 in figure 5). Or there is no direct logical connection, in which case roads are logically connected by “ramps” (all other nodes in figure 4 and node 3 in figure 5). It turns out that in order to “stay on the same road at intersections”, three rules for three possible intersection types (compare figure 5) have to be followed, two rules for “diverging bifurcations” (rule 2 and 3) and one for “crossing intersections” (rule 1):

$$\text{crossing\_inters} : \text{set}[L\times L\times N]\times N \to \text{Bool}\ . \qquad \textbf{(15)}$$

Definition 14.

$\forall$ ng:set[L×L×N], (n:N). crossing_inters(ng, n) =

$\exists\ (k_{enter1}:L), (k_{enter2}:L), (k_{exit1}:L), (k_{exit2}:L)\ .\ |\{k_{enter1}, k_{enter2}, k_{exit1}, k_{exit2}\}|=4 \wedge$

$(\neg\exists\ (k:L).\ k \notin \{k_{enter1}, k_{enter2}, k_{exit1}, k_{exit2}\} \wedge n\in \text{border}(k)) \wedge$

$\text{oneway}(k_{enter1}) \wedge \text{oneway}(k_{enter2}) \wedge \text{oneway}(k_{exit1}) \wedge \text{oneway}(k_{exit2}) \wedge$

$(\exists t_1, t_2 \in \text{ng}.$

$(\#1(t_1)= k_{enter1} \wedge \#2(t_1) = k_{exit1} \wedge \#3(t_1) = n) \wedge$

$(\#1(t_2)= k_{enter1} \wedge \#2(t_2) = k_{exit2} \wedge \#3(t_2) = n)$

$)$

//a crossing intersection node is a 4-valued node, in which two one-way roads cross each other, so that there are two entrance lines and two exit lines//

***Rule 1**:* At “crossing intersection” nodes, there exists a second entrance to and an exit from the node, which means external traffic can cross the intersection on an external one-way road (compare node 1 in figure 5). Therefore there must be in total 2 distinct entrance lines and two distinct exiting lines. The clear distinction between the crossing one-way road and the loop road is possible, because if external traffic is allowed to “cross” the road, then it is inevitable that either one external entering line must lie outside and one exiting line must lie inside of the loop, or vice versa. If both lines either lie inside or outside, the external traffic cannot cross the dual carriageway, and therefore this is not a valid configuration.

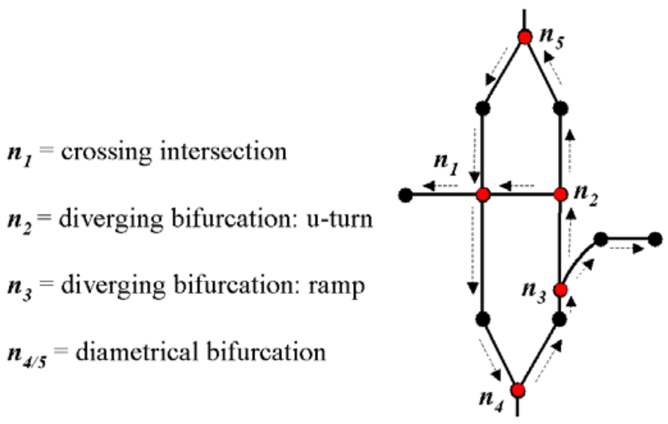
n_1 = crossing intersection
n_2 = diverging bifurcation: u-turn
n_3 = diverging bifurcation: ramp
$n_{4/5}$ = diametrical bifurcation
n_5
n_1
n_2
n_3
n_4

Fig. 5. Types of one-way intersection nodes for dual carriageways.

$$\text{divbifurcation} : \text{set}[L \times L \times N] \times N \rightarrow \text{Bool} . \qquad \textbf{(16)}$$

Definition 13.

$\forall$ ng:set[L×L×N], (n:N). divbifurcation(ng, n) =

$$\begin{aligned}
&\exists (k_{root}:L),(k_{b1}:L),(k_{b2}:L).\ \text{border}(k_{root}) \cap (\text{border}(k_{b1}) \cap \text{border}(k_{b2})) = \{n\} \wedge \\
&\quad (\neg \exists (k_3:L).\ k_3 \notin \{k_{root}, k_{b1}, k_{b2}\} \wedge n \in \text{border}(k_3)) \wedge \\
&\quad \text{oneway}(ng, k_{root}) \wedge \text{oneway}(ng, k_{b2}) \wedge \text{oneway}(ng, k_{b2}) \wedge \\
&\quad (\exists t_1, t_2 \in ng.\ \#3(t_1)=n \wedge \#3(t_2)=n \wedge \\
&\quad \#1(t_1)= k_{root} \wedge \#2(t_1) = k_{b1} \wedge \#1(t_2)= k_{root} \wedge \#2(t_2) = k_{b2})
\end{aligned}$$

//at a diverging bifurcation node, it is only possible to drive from one root line into exactly two branches//

***Rule 2**:* In the case of a “diverging bifurcation”, there is not more than one entrance to the node, and exactly two exits, and therefore there is no crossing traffic possible. These nodes exclusively allow exit to ramps or continuation. Because of international traffic rules aiming to avoid lane crossings, exit to a ramp is only possible to lines lying outside of the loop’s inner region. If the principal directionality is right hand sided, this always means to turn off to one’s right. Turning off to one’s left at such a node, towards the inner region, inevitably means to stay on the same road (see node 3 in figure 5). Therefore if there is a possibility to exit towards the inner region of a dual carriageway loop, this is not a valid configuration.

So far, we have considered all possible intersection nodes at junctions of two dual carriageways. Of course, a bigger variety of intersection nodes are possible in other types of junctions. There is one case that seems to be tricky: “U-turn” roads are paths of planarly embedded one-way lines lying entirely inside of the loop and connecting the two opposite lanes. The tricky thing is that exit nodes to “u-turn” roads are also diverging bifurcations, but they allow turning off towards the inner region of the loop (compare node 2 in figure 5). If we would follow rule 2 at that node, we would not continue on the road, and therefore we have to deal with this special exception:

Rule 3*:* We should break rule 2 at a diverging bifurcation node, if it is an intersection with a one-way “u-turn” road. One-way u-turn roads connect two lanes of the same dual carriageway by a diverging bifurcation. So a dual carriageway should allow to exit towards its inner region at a diverging bifurcation, if this exit can be identified as a u-turn road connecting the opposite lane of the dual carriageway.

It turns out that all other intersections with other types of roads do not pose a problem, because their lines cannot belong to a second dual carriageway. Intersections with bidirectional lines therefore do not have to be taken into account.

$$\text{dual\_carriageway} : \text{set}[L\times L\times N]\times\text{list}[L] \rightarrow \text{Bool}\ . \tag{17}$$

Definition 15.

```
$\forall$ ng:set[$L\times L\times N$], k:list[L]. dual_carriageway(ng, k) =
    simplecycle(k) $\wedge$ ($\forall i\in$toset(k).oneway(ng,i)) $\wedge$
    ($\exists i,j,g,h\in\{1,\ldots,|k|\}$. $i\neq g \wedge j=$next(k,i) $\wedge$ h=next(k,g) $\wedge$
        diabifurcation(ng, I(n:N).toset(border(k[i]))$\cap$toset(border(k[j]))={n},
                k[i], k[j]) $\wedge$
        diabifurcation(ng, I(n:N).toset(border(k[g]))$\cap$toset(border(k[h]))={n},
                k[g], k[h]) $\wedge$
        //there are exactly two diametrical bifurcations in the loop//

        ($\forall f\in\{0,\ldots,|k|-1\}$. $f\notin\{i,g\}\Rightarrow$
                $\exists t\in$ng. #1(t)=k[f] $\wedge$ #2(t)= k[next(k,f)]
        )
        //each pair of connected lines in the loop which is not part of a diabifurca-
        tion can be passed through in exactly one (either counter clockwise or
        clockwise) direction//
    ) $\wedge$
```

```
$(\forall(n{:}N)\in toset(nodes(k)).\ \forall(to{:}L)\notin toset(k).\ \forall t\in ng.$
    $n = \#3(t) \wedge to = \#2(t) \wedge oneway(to) \wedge (\neg\exists(f{:}L).\ diabifurcation(ng, n, f, to))$
    $\Rightarrow$
//rule 1//    $(crossing\_inters(ng,n) \wedge$
                        $\neg\exists\ (to_2{:}L).\ n\in border(to_2) \wedge to\neq to_2 \wedge$
                                    $line(to) \subset region_{inner}(k) \wedge line(to_2) \subset region_{inner}(k))$
             $\vee$
//rule 2//    $(divbifurcation(ng, n) \wedge line(to) \cap region_{inner}(k)=\emptyset)$
             $\vee$
//rule 3//    $(divbifurcation(ng, n) \wedge$
                        $(\exists(d{:}list[L]).\ to=d[0] \wedge planar(toset(k)\cup toset(d)) \wedge$
                                    $(\forall z\in toset(d).\ z\notin toset(k) \wedge$
                                                $(line(z) \cap region_{outer}(k)=\emptyset)$
                                    $) \wedge$
                                    $(\exists(g{:}L)\in k.\ (\exists t_i\in ng.\ \#1(t_i)=d[|d|-1] \wedge \#2(t_i)=g) \wedge$
                                                 $(shortest\_path(ng, d, d[0], d[|d|-1])!) \wedge$
                                                $\neg(shortest\_path(ng, k, \#1(t), g)!)$
                                    $)$
                        $)$
             $)$
$)$
```
//intersection nodes allowing exit to one way lines (and not being diametrical bifurcations), are either diverging bifurcations at ramps (allowing exit towards the outside of the loop), or diverging bifurcations at u-turn roads (allowing exit towards the inner region of the loop), or crossing intersections with another one way road//

Dual carriageways (unclosed). Dual carriageways sometimes can have only one or no diametrical bifurcation node at all. This is the case if the dual carriageway ends at a T-junction. In order to specify unclosed dual carriageways, we have to specify their T-junctions. This is left as a specification task for future efforts.

Freeways. Freeways are dual carriageways with less variety of intersections. Let us first consider the case of one-way intersections. In fact, on a freeway, a crossing intersection is normally not allowed, as well as a u-turn road. This is because at these intersection types, a vehicle has to cross all lanes, which is not a desirable manoeuvre for freeways. It is immediately clear that the only type of one-way exit from a dual carriageway that does not imply lane crossings is the diverging bifurcation at ramps complying with dual carriageway rule 2.

Furthermore, intersections with bidirectional lines are not acceptable for freeways, because each exit or entrance must have an acceleration lane, and the construction of two tangential acceleration lanes for exit and entrance is not possible in one line. In consequence, the specification of freeways is comparably easy: Every dual carriageway that exclusively has got ramp exits (rule 2 exits) or ramp entrances as node intersections is a freeway:

$$\text{freeway} : \text{set}[L\times L\times N]\times\text{list}[L] \rightarrow \text{Bool} . \qquad \textbf{(18)}$$

Definition 16.

```
∀ ng:set[L×L×N], k:list[L]. freeway(ng, k) =
        dual_carriageway(ng, k) ∧
        (∀(n:N)∈toset(nodes(k)). ∀(to:L)∉toset(k). ∀t∈ng.
                n =#3(t) ∧ to =#2(t) ∧ (¬∃(f:L).diabifurcation(ng, n, f, to))
                ⇒
//rule 2//       (divbifurcation(ng, n) ∧ line(to) ∩ region_inner(k)=∅)
        )
```

5 Conclusion

In order to apply advanced high-level concepts for transportation networks, like hyper graphs, multi-level way finding and traffic forecasting, to commercially available street datasets, it is often necessary to generalise from network primitives. The appropriate method of generalisation depends on the complex street network feature these primitives belong to. As was shown in section 2, essential complex features can be e.g. roundabouts and dual carriageways, but also freeways and several types of junctions. In order to specify them, a formal network language was introduced in section 3, which allows a clear distinction among 3 aspects: the geometrical network graph, its embedding into the Euclidian space, and a second graph called “navigation graph”, built on the first one, and reflecting navigational constraints for a traffic mode. On this basis, the road segment types “dead”, “dead end” and “oneway” as well as the road types “normal_road”, “linear_ring”, “roundabout”, “dual_carriageway” and “freeway” could be specified by formal definitions.

Our future work will focus on the implementation of the proposed predicates, considering efficient algorithms and search strategies for the identification of complex network features. Furthermore, the next step should be to consider some essential features that were not discussed in this paper, e.g. complex junction types.

Acknowledgments. The authors would like to thank Dr. Angi Voss for her valuable support concerning the work on formal expressions.

References

1. Egenhofer, M.J., Herring, J.R.: Categorizing binary topological relationships between regions, lines and points in geographic databases. In: Technical Report, Department of Surveying Engineering, University of Maine, Orono (1991)
2. Farmer, W.M., A Basic Extended Simple Type Theory. SQRL Report **14**, (2003)
3. Farmer, W.M., Mohrenschmidt, M.: Simple type theory: Simple steps towards a formal specification. In: 34th ASEE/IEEE Frontiers in Education Conference, Savannah, GA, F1C-6 (2004)
4. Goodchild, M.F.: Geographic Information Systems and Disaggregate Transportation Modelling. Geographical Systems **5**(1-2) (1998) 19–44
5. Mainguenaud, M.: Modelling the Network Component of Geographical Information Systems. Int. J. Geographical Information Systems, **9**(6), (1995) 575–593
6. Stell, J.G.: Granulation for Graphs. In: Freksa, C., Mark, D.M. (eds.): Spatial Information Theory: Cognitive and Computational Foundations of Geographic Information Science, International Conference COSIT'99. Lecture Notes in Computer Science, Vol. 1661. Springer-Verlag, Berlin Heidelberg New York (1999) 417–432
7. Stell, J.G., Worboys, M.: Generalizing Graphs using Amalgamation and Selection. In: Güting, R.H., Papadias, D., Lochovsky, F.H. (eds.): Advances in Spatial Databases, 6th International Symposium, SSD'99. Lecture Notes in Computer Science, Vol. 1651. Springer-Verlag, Berlin Heidelberg New York (1999) 19–32
8. Timpf, S., Volta, G.S., Pollock, D.W., Egenhofer, M.J.: A Conceptual Model of Wayfinding Using Multiple Levels of Abstractions. In: Frank, A.U., Campari, I., Formentini, U. (eds.): Theories and Methods of Spatio-Temporal Reasoning in Geographic Space. Lecture Notes in Computer Science, Vol. 639. Springer-Verlag, Berlin Heidelberg New York (1992) 348–367
9. Timpf, S., Kuhn, W.: Granularity Transformations in Wayfinding. In: Freksa, C., Brauer, W., Habel, C., Wender, K.F. (eds.): Spatial Cognition III. Lecture Notes in Artificial Intelligence, Vol. 2685. Springer-Verlag, Berlin Heidelberg New York (2003) 77–88
10. Zhou, C., Lu, F., Wan, Q.: A Conceptual Model for a Feature-Based Virtual Network. GeoInformatica, **4**(3), (2000) 271–286
11. Tang, A.Y, Adams, T.M., Usery, E.L: A Spatial Data Model Design for Feature-Based Geographical Information Systems. Int. J. Geographical Information Systems, **10**(5), (1996) 643–659