Evaluating and Minimizing Ambiguities in Qualitative Route Instructions

Matthias Westphal
Department of Computer Science
University of Freiburg
Freiburg, Germany
westpham@informatik.uni-freiburg.de

Jochen Renz
Research School of Computer Science
The Australian National University
Canberra ACT 0200, Australia
jochen.renz@anu.edu.au

ABSTRACT

Route navigation is a widely studied subject from both cognitive and practical points of view. A particular aspect is the generation of (verbal) route instructions that are robust with respect to ambiguous verbal terms. Work in this area usually builds on counting the number of ambiguous turn options along a route. Simple graph search can then be used to derive a route whose description is the most fault-tolerant according to this measure.

In this paper we contrast this approach with a probabilistic planning one that estimates the probability of reaching the destination given a probabilistic model of an agent interpreting the route instruction. To this end, we discuss different models of agents, the evaluation of route instructions and derive optimal and approximate approaches for the planning problem.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Spatial databases and GIS; E.1 [**Data Structures**]: Graphs and networks

General Terms

Algorithms

Keywords

qualitative reasoning, route instructions

1. INTRODUCTION

Getting lost or losing the way is a major nuisance when navigating in an unknown environment. Even today's sophisticated navigation systems do not prevent us from taking the wrong exit or missing a turn. GPS-based navigation systems usually notice our mistakes very quickly and can then re-plan the best route from the current location to the destination. However, in such situations, it often takes significantly longer to get to the destination and, depending on the situation, it can have other more severe side effects. When traveling without a navigation system, matters are even worse, as it usually takes considerable time to recognize that we are lost. Once we notice it, we have to find out where we are and find an alternative route to the destination, or ask someone for new route instructions. In both cases, with or without having a navigation system, route descriptions should be formed in a robust way that takes into account plausible mistakes of users. Such descriptions ideally still lead to the destination even if a mistake is made. In other words, they should reduce the probability of accidentally losing one's way.

Whether or not wrong turns are made depends on the accuracy of the route description, but also on the accuracy of the terms used therein. If terms and description are perfectly accurate, and if we accurately follow every instruction as intended, then we should not make a wrong turn. However, such a level of accuracy is not common, neither in natural language, which is what is generally used for making route descriptions, nor in human behavior. Consider, for example, the route instruction "Turn left at the next intersection". There can be different situations where this might lead to a wrong turn. Firstly, there could be two streets that go left at the next intersection. Secondly, there could be no street that goes left at the next intersection. Thirdly, there could be confusion about what is the next intersection. The instructor might have forgotten about a small intersection that comes before the intended instruction, or the driver might overlook the next intersection and only turn left at the second intersection. A possible solution to the first situation could be to introduce finer grained direction terms such as "turn sharp left", "turn slightly left", etc.. A possible solution to the second and third situation might be to give an instruction for absolutely every possible turn, which requires a long, unnatural and probably unnecessary list of "continue straight" instructions. An alternative could be to use landmarks or street names in the instructions, or to use distance measures, but often there are no visible street names and no unique landmarks, and distances can be hard to judge.

In this paper we focus on pure directional route instructions, without using landmarks or street names, and without using distance descriptors, and analyse how accurately we can reach our destination using such instructions alone. Our goal is to identify a route instruction in a given street network that leads us from the origin to the destination with the highest probability. To this end, we represent several of the above discussed issues with regard to ambiguity in a simple probabilistic agent model. The agent model specifies how and with what probability an instruction primitive (e.g., a turn or "continue straight" instruction) is executed. Given a route instruction, this model then follows the description and only stops if either the next instruction cannot be executed, or all instructions have been executed. Based on this principle, we can calculate the probability with which an agent reaches the destination. The route instruction with the highest probability of reaching the goal is called the *most robust instruction*. It is clear that this description will most likely not coincide with shortest routes, but our assumption is that accurately reaching the destination is more important than a shortest route and that it might bring us to the destination fastest on average as we do not waste time and effort with trying to recover from taking the wrong turn.

We develop a novel probabilistic method of computing the most robust route instruction and compare it with the best previous approach, the work by Haque et al. [6]. Their approach is to count the number of ambiguous turns, and thus avoid ambiguous situations wherever possible. By reducing ambiguities, this technique increases the chance of reaching the desired destination, but it does not explicitly take into account probabilities. This entails that it cannot differentiate between route instructions with the same number of ambiguous intersections, but different associated probabilities. Further, it avoids ambiguous turns even if they all lead to the destination.

For our empirical comparison we use different models of direction relations, one that is very coarse and uses only "front", "left", "right", and "back". The second model is more fine-grained and corresponds to the directions that were found most useful for navigation in cognitive studies by Klippel and Montello [7]. We also use two different models for interpreting and generating route descriptions. One that gives an instruction for every intersection and one where we go straight by default until the next instruction can be executed. We empirically compare our approach with the work by Haque et al. for the different models using street networks from parts of Paris and Canberra. Canberra is a city that has been planned from scratch and where all streets follow certain patterns. Paris, instead, is a city where streets seem to be almost random and where intersections occur in all angles. For each city we randomly generate $10,000$ pairs of origin and destination points and compute the robust route instructions. It turns out that for most pairs all approaches considered in our evaluation can find a route instruction that leads to the destination with 100% probability. This is not surprising, because intersections with multiple turns that match the same turn description are relatively rare. However, even though these intersections occur infrequently, they are usually the ones that create problems and that make us take a wrong turn. We, therefore, concentrate our analysis on these cases. It turns out that for those pairs where the approach of Haque et al. does not find a robust route instruction, our approach leads to better results in many cases.

The paper is structured as follows. In the next section we briefly discuss previous work on the subject of generating route instructions. Section 3 then gives background on qualitative representations of turn instructions and the used graph representations. The following Section 4 details the considered agent models and the relation between instructions and sets of paths, as well as success probabilities. Section 5 gives our algorithms for optimizing route instructions and theoretically compares it to the work of Haque et al.. Finally, Section 6 contains our empirical evaluation and findings. Directions for future work and the conclusion complete the paper.

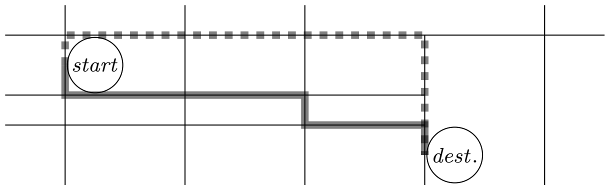

Figure 1: An example of two routes from the same start to the same destination. The dashed route is easier to describe as it contains fewer turn instructions.

2. PREVIOUS WORK

Previous work has mainly focused on generating route instructions via shortest-path algorithms. Given a specific cost metric the shortest-path algorithm identifies the route that is best used for the instruction. The qualitative description of this route then serves as an instruction. Such an approach is certainly practical as it is computationally easy to handle. However, it limits the possible criteria for the selection of routes to those expressible as a cost metric for the search. Certain criteria are dependent on the actual set of paths that corresponds to it, e.g., the probability of reaching the desired destination.

There has been a lot of work on generating suitable routes (or route instructions) for various purposes. A common point is the focus on finding routes that are "simple" or "easy to describe". The typical approach is to apply graph search on a route network to obtain one suitable path (in terms of desired properties) and then to describe this path. Originally, Mark [11] proposed to use a measure of "description complexity" in conjunction with metric route length to obtain routes suitable for navigation. Such routes would for example avoid many turns as these are more complex to describe than simply continuing straight on a road. Following work by Duckham and Kulik [4] has studied the problem using only the complexity of route instructions without regard for metric route length. Here, a shortest-path search is used in combination with a cost metric that assigns a cost value to each turn description at an intersection taking into account the number of outgoing arcs. Figure 1 illustrates the concept of simple routes.

An adaption of this work by Haque et al. [6] uses the same shortest-path search idea, but models the weights according to how many outgoing arcs are considered "instruction equivalent". This leads to their measure of *unreliability* as the number of instruction equivalent options minus one. The *unreliability value* of an entire route is then simply the sum of individual unreliability values at all intersections along the route. The unreliability value is thus zero if the instruction has no ambiguous turn instruction. Further, the authors suggest to break ties with the metric path length. Finally, their work considers a weighted sum of metric distance and unreliability value to balance reliability and route length. This paper will be referenced often in this work, since their unreliability measure is connected to the probability of reaching a desired destination. We will formally define and analyse their unreliability measure in Section 5.

There has also been work on incorporating landmarks into such schemes, e.g., by Richter and Duckham [16], and Duckham et al. [5]. This extended version of the earlier graph search algorithm computes the shortest route description according to landmarks, i.e., it exploits knowledge about landmarks in the vicinity. Ambiguities, however, are ignored. Since the visibility, saliency and availability of landmarks is a problem in its own right, we do not take into account landmark information in our work, but note that an extension would be a useful endeavor. The contribution and principal issues presented in this work should be useful to this end.

Generating route instructions has also been studied from a natural language processing point of view. Marciniak and Strube [9, 10] used machine learning to derive natural language route instructions from a previously learned corpus of instruction expressions. In a similar context, the idea of a probabilistic approach to route generation appeared in a reinforcement learning approach for route instructions. Cuayáhuitl et al. [2] use hierarchical Markov Decision Processes (MDP) [1, 12] to adaptively derive instructions. While the problem formulation we use in this work is very close to MDPs, it should be noted that it corresponds to an unobservable (or partially observable) MDP as route instructions, i.e., policies, are only history-dependent (or depend on the belief state, respectively).

We also want to mention that there has been important work on cognitive studies detailing the quality of verbal route instructions e.g., by Lovelace et al. [8] and the vocabulary used in such instruction, e.g., by Klippel and Montello [7]. While these aspects are not the focus of our work, we use the turn descriptions proposed by Klippel and Montello in our empirical evaluation.

3. PRELIMINARIES

To present our ideas and findings, we briefly recall a few concepts about qualitative turn labels, graph representations of route networks and directed graphs with multiple arcs.

3.1 Qualitative Turn Labels and Route Instructions

As mentioned previously, we work with simple descriptions of routes, that is, each description is a sequence of *turn labels*. A qualitative turn label is a qualitative relation between street segments. Such a qualitative relation establishes the relation of an agent at a street intersection to possible outgoing streets. Note that several streets might map to the same qualitative relation, i.e., outgoing streets are not necessarily uniquely determined by a qualitative relation. Haque et al. [6] refer to these as “instruction equivalent”. Naturally, different conceptualizations of relations between street segments exist. In this paper, we consider two different ones based on the $\mathcal{STAR}$ family of qualitative calculi [14] used in the field of qualitative spatial reasoning. While $\mathcal{STAR}$ calculi are often used to represent and reason with one unique global reference direction (cf. [14]), we do not require a unique global reference for our representation purposes. Instead, we use the incoming street of an intersection as the reference direction to describe the relation between this segment and the outgoing segments, and write $\mathcal{STAR}^r$ for such representations. The first one we use, $\mathcal{STAR}_2^r$, is a naïve representation that divides the plane into four sectors of equal size “left”, “straight”, “right” and “back”. The second representation we use, $\mathcal{STAR}_4^r$, adapts to the findings of Klippel and Montello [7] and uses six sectors of different size (the same as in $\mathcal{STAR}_2^r$ plus “slightly left”, “sharp left”, etc.). Figure 2 shows these two qualitative representations.

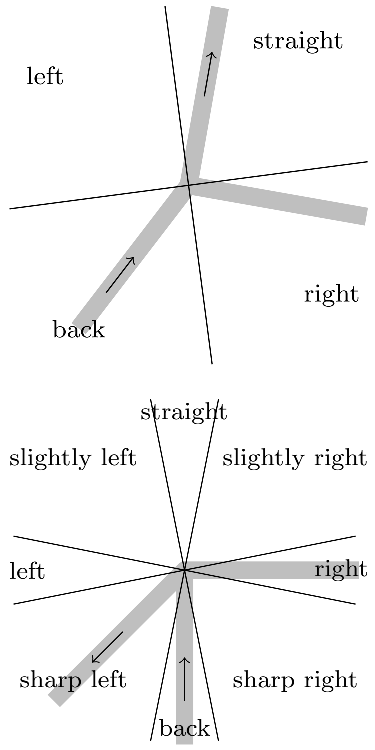

Figure 2: Two distinct schemata used to define qualitative turn descriptions for street intersections. $\mathcal{STAR}_2^r$ is illustrated at the top and $\mathcal{STAR}_4^r$ at the bottom.

Given a qualitative representation of turn labels, a qualitative route instruction is simply a sequence of qualitative relations from one fixed qualitative representation $\Delta = \langle r_1, \ldots, r_n \rangle$ (e.g., $\Delta = \langle$“left”, “right”, “straight”, “left” $\rangle$).

3.2 Abstractions of Route Networks

Route networks can be considered as common directed binary graphs with positional information for each vertex, such that they can accurately describe the contour and metric length of route segments. However, for our purposes we do not need this level of detail. In order to generate or evaluate route instructions, the only relevant parts of such a graph are street intersections where a decision has to be made by an agent. Westphal et al. [17] introduce a suitable succinct representation for this purpose with the notion of *decision frames*. Decision frames give information about the set of *states* of an agent (position and orientation at street intersections) and the possible *decisions* that can be made by the agent at each state and where they lead to. Such decision frames can be cast as directed graphs if one considers multiple arcs between any two nodes. To this end, we define directed graphs similar to Diestel [3]:

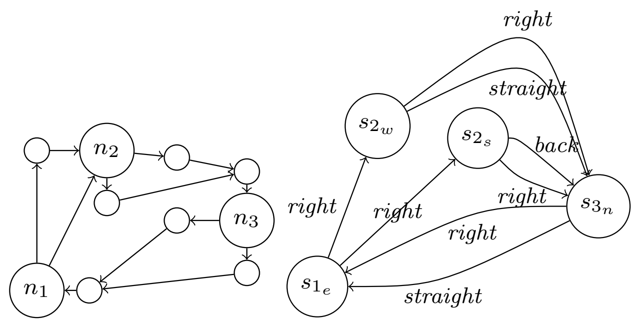

Figure 3: Representation of the state space. The left side shows a route network (decision nodes labeled with n), and the right side a corresponding decision frame with qualitative turn labels.

Definition 1. A *directed binary graph* is a tuple $G = (V, E)$, where

- V is a set of nodes (or vertices), and
- E is a set of arcs (or edges) with a map $E \to V^2$ that assigns to each arc a start and end node.

We write $e = v_1v_2$, for $e \in E$, $v_1, v_2 \in V$, to refer to some arc from v_1 to v_2. Note, this does not uniquely determine e as there might be *multiple arcs* from v_1 to v_2.

It is indeed sufficient to treat decision frames as directed graphs and to consider standard search algorithms on them. In the following we give a general description of the decision frame representation and refer to the work by Renz and Wölfl [15], and Westphal et al. [17] for a more formal discussion and definition of the involved reductions.

The purpose of the decision frame representation is to provide an overlay to a route network that makes the decisions of an agent more accessible and allows us to consider (graph altering) precomputations in Section 5. Essentially, the representation abstracts from a route network by only considering nodes where decisions take place, and summarising other nodes as links between them. It is therefore reminiscent of a topological map that contains places and links between them. However, the crucial difference is that we choose to explicitly represent the orientation of agents at each such node to make algorithms simpler and the approach more general. Orientation is represented by duplicating nodes whenever there are multiple ways of approaching them.

We can generate a decision frame from accurate metric input, i.e., a directed graph G and a position in the plane for each of its vertices (e.g., OpenStreetMap data). The first abstraction step is to identify a set of *decision nodes*. Namely, those nodes which have more than one successor node, and are thus points where a decision has to be made. It is further necessary to also consider sources and sinks (i.e., nodes without incoming or outgoing arcs, respectively) as decision nodes, as otherwise parts of G might be completely ignored. For each such decision node, we create a new state (node) in the decision frame for each of its incoming arcs in G. These represent the current orientation of an agent traveling along the incoming arcs of the decision node in G. Usually, the corresponding representation for this is the *directed line graph* [3], the variant of the graph where arcs are interpreted as vertices and each connecting node as an arc. This has been used previously for the shortest-path search approach to route descriptions [4, 6]. In our case we use the same basic idea but maintain a reduction to decision nodes. Finally, we add arcs in the decision frame that correspond to the paths in G that directly link decision nodes.

As an example, Figure 3 depicts a route graph and a corresponding decision frame. Note that whenever two paths enter a decision node through a shared arc in G, we do not have two different states (e.g., the two paths from n_2 to n_3 enter at the same arc in Figure 3), while two paths with different entrances are represented with two states (e.g., the two paths entering n_2 translates to s_{2_w} and s_{2_s}).

The following definition gives a formal summary:

Definition 2. A *decision frame* is a directed binary graph given by a tuple $\mathcal{S} := (S, A)$ based on a directed binary graph G of a route network, where

- S is a set of states, where each state $s \in S$ corresponds to an incoming arc of a node in G that is considered a decision node, and
- A is a set of arcs, where each arc $a = s_1s_2$, $s_1, s_2 \in S$, corresponds to a path in G that links the two decision nodes s_1, s_2 without visiting any other decision nodes.

As all possible entries into an intersection are represented by a different state, we can label all arcs in a decision frame $\mathcal{S} = (S, A)$ using a given set R of qualitative turn labels. In our case labels from the sets $\mathcal{STAR}_2^r$ or $\mathcal{STAR}_4^r$. To this end, we assume in the following that there is a labeling function l that assigns to each arc its qualitative turn label. Formally, $l\colon A \to R$ assigns to each arc $a = ss'$ the corresponding qualitative turn label that matches the direction of the outgoing path from s to s'.

We further note that additional labels can represent extra information on the paths, e.g., path length, visited street addresses, etc..

4. INTERPRETATION OF ROUTE INSTRUCTIONS

Qualitative decision frames provide the necessary fundamental information to interpret route instructions. However, the question remains *how* agents (especially humans) interpret a given route instruction. To this end we follow the work by Haque et al. [6] and define a simple yet sufficient agent model by formalizing underlying assumptions.

4.1 The Agent Model

First of all, we assume that the qualitative vocabulary is fixed, that is, an agent has the same interpretation of qualitative relations as used in the route graph representation. Thus, an agent does not confuse different relations. Secondly, instructions are interpreted step-wise (from beginning to end). We assume here that agents do not prefer any particular outgoing arc among those with the same turn label. Thirdly, if in a state the current turn label does not correspond to any outgoing arc, the agent stops. Similarly, if there are no more turn labels remaining the agent stops as well. This corresponds to the model used by Haque et al. who further consider replanning in cases where agents do not reach the desired destination. We do not consider replanning in this paper, since our focus is on evaluation and optimization of the robustness of initial route instructions.

A crucial assumption of the above model is that agents try to match a turn label in the instruction to an option at the current decision node. However, we argue that it is not immediately clear whether agents interpret each turn label as an "immediate instruction", i.e., a request to turn at this intersection. As discussed in Westphal et al. [17], we note that this *strict* interpretation can be contrasted with a *weaker* notion of turn labels as "at the next possibility", i.e., agents would continue straight as long as possible until an outgoing arc with the turn label becomes available. This difference is especially important when considering the question whether agents would belief to have made a wrong turn if the next instruction is not immediately applicable. It can easily be argued that agents would continue straight ahead in such cases (unless this is not possible). We will thus treat this assumption separately in our discussion and evaluation, referring to *strict* and *weak* interpretations.

4.2 Interplay Between Route Instructions and Paths

The agent model easily turns into a (non-deterministic) recursive expression which yields for a given start state and description a valid path that could be traversed by an agent. Let $\mathcal{S} = (S, A)$ be the decision frame as usual and $s[r] := \{ s' \mid a \in A, a = ss', l(a) = r \}$ the set of successor states of s reachable with turn label r. For the strict interpretation we can define the $trace(s, \langle r_1, \dots, r_n \rangle)$ expression as

$$\begin{cases} \langle s \rangle + trace(s', \langle r_2, \dots, r_n \rangle) & \text{for any } s' \in s[r_1] \\ \langle s \rangle & s[r_1] = \emptyset \text{ or } n = 0 \end{cases}$$

Let st be the turn label for *straight*, then, for the weak interpretation, we get the definition

$$\begin{cases} \langle s \rangle + trace(s', \langle r_2, \dots, r_n \rangle) & \text{for any } s' \in s[r_1] \\ \langle s \rangle + trace(s', \langle r_1, \dots, r_n \rangle) & s[r_1]{=}\emptyset, \text{for any } s' \in s[st] \\ \langle s \rangle & s[r_1] \cup s[st]{=}\emptyset \text{ or } n = 0 \end{cases}$$

The weak approach raises several interesting points. For one, the weaker interpretation sounds more reasonable, as it can be expected that agents assume that a "continue straight" instruction has been left out (accidentally or on purpose). On the other hand, this kind of "default action" raises some issues about shortest route instructions, since exploiting them can sometimes lead to very succinct descriptions of routes that contain loops, and thus are far from shortest path considering metric length or visited states. Figure 4 shows an example of a short description that induces a path with a loop. Even worse, weak interpretations of instructions may lead to revisiting a state only because of default straight actions and thus allow for arbitrarily long paths, although it is implausible any human agent would get caught in such loops. Note that explicitly generating shortest instruction without such loops is NP-hard [17].

As long as we do not mind such loops, we can reduce the weak interpretation to the strict one by augmenting the decision frame with additional arcs that represent "shortcuts": for each state s_0 without a turn label r, we add an arc a, $a = s_0 s'$ for each distinct path without a loop $s_0 s_1 \dots s_n s'$ in $\mathcal{S}$, where arcs $s_i s_{i+1}$ are traversed because of a straight action, and $s_n s'$ because of the turn label r. Thus, for such augmented graphs it is not necessary to search for the next state where r is applicable.

4.3 Probabilistic Transitions

Assuming we are given a decision frame, we can compile its set of arcs to transitions with associated probabilities. To this end, we unify arcs with the same turn label between the same states and assign the correct probability to this transition. The multiplicity of arcs in the original decision frame is merely required to accurately reflect the universe of arcs in order to compute probabilities. Formally, for each pair of states s, s' and turn label r we replace all arcs with $a' = ss', l(a') = r$ with one arc a labelled with r and its *transition probability*

$$prob(a) = \frac{|\{ a' \mid a' \in A, a' = ss', l(a') = r \}|}{|\{ a' \mid a' \in A, x \in S, a' = sx, l(a') = r \}|} \quad .$$

Algorithm 1 Compiling arcs in decision frames to probabilistic transitions.

```
Input: Decision frame $\mathcal{S}$
Output: Decision frame $\mathcal{S}'$ with transition probability $prob$
    for each arc.
 1: function COMPILE($\mathcal{S}$)
 2:     Let $\mathcal{S} = (S, A)$
 3:     $A' \leftarrow \emptyset$
 4:     $prob \leftarrow \emptyset$
 5:     for $s \in \mathcal{S}$ do
 6:         for each qualitative turn labels $r$ do
 7:             $successors \leftarrow \emptyset$
 8:             for each arc $a$ from $s$, where $l(a) = r$ do
 9:                 Let $a = ss'$
10:                 if not defined $successors[s']$ then
11:                     $successors[s'] \leftarrow 0$
12:                 $successors[s'] \leftarrow successors[s'] + 1$
13:             $od \leftarrow |\{ a \mid a \in A, x \in S, a = sx, l(a) = r \}|$
14:             for $s' \in \mathcal{S}$ where $successors[s']$ is defined do
15:                 Define $a' = ss'$ in $A'$ with $l(a') = r$
16:                 $prob(a') \leftarrow \frac{successors[s']}{od}$
17:     return $(S, A')$, $prob$
```

This procedure is given in Algorithm 1. It should be noted, that although we have disregarded loops in case of the weak interpretation, this does not influence our transition probabilities. We further note that the resulting graph can be cast as an (unobservable) MDP, but we refrain from explicitly defining it as such here, since we just focus on success probabilities. This compilation step is easy to perform and must only be applied to networks once.

Finally, given such a decision frame with transition probabilities, the probability of following one path through the decision frame is simply the product of the transition probabilities. Given a route instruction $\Delta = \langle r_1, \dots, r_n \rangle$, the probability of reaching a destination state is the sum of the probabilities of each path to the destination under this description. This can be calculated in time polynomial in the size of the graph and description by using dynamic programming as seen in Algorithm 2. In terms of an MDP this is the evaluation of the policy Δ.

In the following we use the term "decision frame" in the sense of such a pre-compiled graph with transition probabilities.

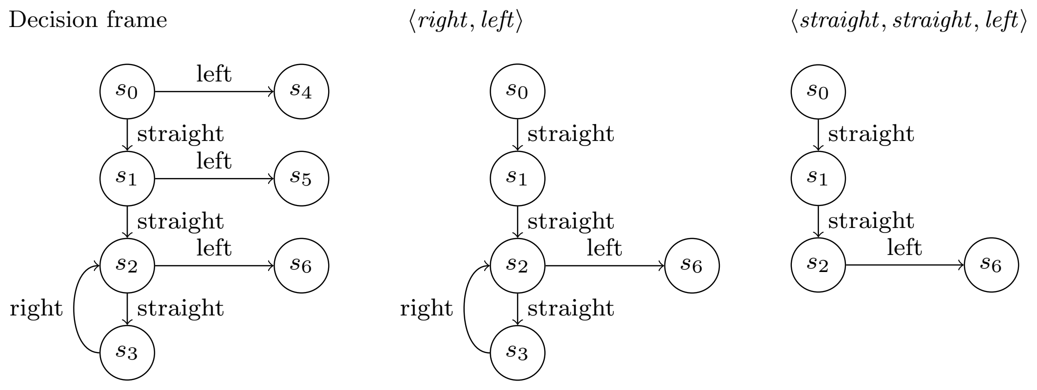

Figure 4: A decision frame (left), where a path from s_0 to s_6 can be succinctly described by $\langle \mathit{right}, \mathit{left} \rangle$, but this forces a loop in the path (middle). The shortest description without a loop is $\langle \mathit{straight}, \mathit{straight}, \mathit{left} \rangle$ (right).

Algorithm 2 Evaluating qualitative route instructions.

```
Input: Decision frame $\mathcal{S}$ with transition probability $prob$
    for each arc, start/destination pair $s, s_\star$, route instruc-
    tion $\Delta$. Further, let $P$ be a global associative array.
Output: Success probability for current arguments.
 1: function EVALUATE($\mathcal{S}$, $s$, $s_\star$, $\Delta$)
 2:     if $P[s, \Delta]$ defined then return $P[s, \Delta]$
 3:     if $\Delta = \emptyset$ then
 4:         if $s = s_\star$ then return 1.0 else return 0.0
 5:     $p \leftarrow 0.0$
 6:     $count \leftarrow 0$
 7:     $r \leftarrow \Delta.pop\_front()$
 8:     for all outgoing arcs of $s$, $a = ss'$ where $l(a) = r$ do
 9:         $p \leftarrow p + prob(a) \cdot$ EVALUATE($\mathcal{S}$, $s'$, $s_\star$, $\Delta$)
10:         $count \leftarrow count + 1$
11:     if $count = 0$ then return 0.0
12:     $p \leftarrow \frac{p}{count}$
13:     $P[s, \Delta] \leftarrow p$
14:     return $p$
```

5. ALGORITHMS FOR ROUTE INSTRUCTIONS

In this section we contrast the work by Haque et al. with our approach with respect to the probability that a description leads us from the origin to the destination. We further describe how graph search can be applied to compute lower bounds on success probabilities.

5.1 Reliable Routes

In the work by Haque et al. their unreliability measure accounts for the total number of instruction equivalent alternatives encountered along the chosen path. We sketch their work here based on decision frames.

Their algorithm to obtain the "most reliable route" performs a shortest-path search on the decision frame, where the cost of the initial state is 0 and that of a succeeding state s' of s is the cost of s plus the number of all successor states under the same action minus one.[1] Formally, the cost of s' is computed from the used arc a and the cost of s, $C[s]$, by the function $cost(C[s], a)$, defined as

$$C[s] + |\{ s'' \in S \mid a' \in A, a' = ss'', l(a') = l(a) \}| - 1 \quad (1)$$

This cost function can be used in a simple graph search algorithm, for example, the one shown in Algorithm 3.

Unfortunately, this unreliability cost function only roughly corresponds to some bound on the probability of reaching the desired destination, since it does not account for the ambiguities' depths along the path. For example, the unreliability values of 0 and 1 correspond to lower bounds on the probability of 1.0 and 0.5, respectively. That is, if the unreliability cost is 0 then there is no ambiguity, if it is 1 then there is one intersection with two outgoing arcs with the same turn label. However, in general, an unreliability value of u can only be seen as a lower bound of 2^{-u} on the probability. This is because the worst case occurs when

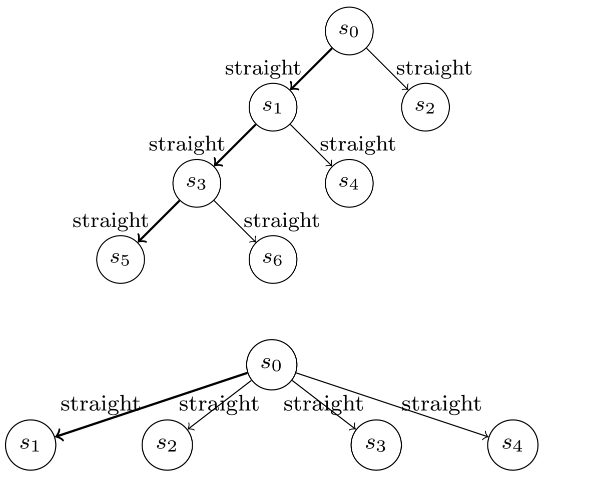

Figure 5: Example of two paths with the same "unreliability", but different success probabilities. The path from s_0 to s_5 has an unreliability value of 3 in both cases, but the success probability is $\frac{1}{8}$ and $\frac{1}{4}$ respectively.

[1]Originally, it is the number of instruction equivalent outgoing arcs minus one in the underlying route network. Using decision frames this is already improved, because we do not count arcs that lead to the same successor state, since such information is already reflected in the precomputed graph.

there are u states with two ambiguous successors each. The probability is then only $0.5 \cdot 0.5 \cdot \ldots \cdot 0.5$, u-times. Figure 5 gives an example.

Algorithm 3 Shortest-path algorithm for route instructions with cost function and initial cost of s_0.

```
Input: Decision frame $\mathcal{S}$, start/destination pair $s_0, s_\star$, cost
    function $cost$, and initial cost of $s_0$, $c_0$.
Output: Instruction of least-cost path, cost of path;
    false if no path exists.
 1: function SHORTEST-PATH($\mathcal{S}$, $s_0$, $s_\star$, $cost$, $c_0$)
 2:     $queue \leftarrow \{s_0\}$
 3:     $link \leftarrow \emptyset$
 4:     for all states $s$ in $\mathcal{S}$ do $C[s] \leftarrow \infty$
 5:     $C[s_0] \leftarrow c_0$
 6:     while $queue \neq \emptyset$ do
 7:         $s \leftarrow$ pick state $s$ from $queue$ with minimal $C[s]$
 8:         if $s = s_\star$ then                ▷ Construct instruction
 9:             $\Delta \leftarrow$ empty list
10:             $s' \leftarrow s$
11:             while $link[s']$ defined do
12:                 $\Delta.push\_front(link[s'][1])$
13:                 $s' \leftarrow link[s'][0]$
14:             return $C[s], \Delta$
15:         for each outgoing arc $a$ of $s$, $a = ss'$ do
16:             if $cost(C[s], a) < C[s']$ then
17:                 $C[s'] \leftarrow cost(C[s], a)$
18:                 $link[s'] \leftarrow (s, l(a))$
19:                 $queue \leftarrow queue \cup \{s'\}$
20:     return false
```

5.2 Success Probabilities

Because decision frames based on our agent model are very close to unobservable MDPs, it is not surprising that decision theoretic variants of finding robust route instructions can be shown to be NP-hard (as in the case of unobservable MDPs [12]). One option to accurately optimize route instructions is to consider the set of all traces for potential route instructions, to see which traces lead to the destination. This could be done by performing a search over route instruction prefixes and tracking the "frontier", i.e., the set of current states for each such prefix. While this is possible, it comes at a very high computational cost – in the worst case exponential in the number of states. The computational cost of tracking sets of states under some instruction(-prefix) is particularly severe when compared to simple shortest-path search, which has been used in previous approaches. For this reason, we follow a more practical alternative in this paper and do not further consider optimal computations.

Instead of the completely accurate computation, we approximate the probability of reaching the destination by again using shortest-path search. Using the same search as the previous approach (Algorithm 3), we simply replace the cost function with

$$cost(C[s], a) := C[s] \cdot prob(a) \tag{2}$$

where $prob(a)$ is the transition probability of the chosen arc, and the cost of the initial state is -1.0.[2] The shortest-path

[2]The cost of the initial state has to be negative since the shortest-path algorithm minimizes cost.

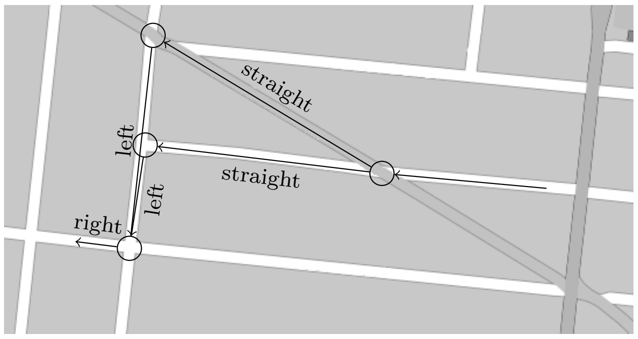

Figure 6: An example of merging paths under the same instruction in case of $\mathcal{STAR}_2^r$ and weak interpretations. The instructions is ⟨"left", "right"⟩.

found with cost q has an associated route instruction with a success probability of *at least* $-q$, i.e., this probability value is only a lower bound.

While the found path (and thus instruction) is optimal with regard to the lower bounds on all other paths, it is in general not the optimal instruction. In fact, the lower bound corresponds to the actual probability if and only if all alternative traces under the given instruction do not lead to the destination state. This assumption is pessimistic, but in many cases correct though. Still, we can improve the lower bound, by mixing the shortest-path search based on cost function (2) with tracking sets of traces by simply adding arcs to the decision frame.

The idea behind this approach is that there are cases where taking a wrong turn has no negative effect as both lead to the same state using the same instructions. This will affect the probability of reaching this state. Whenever there are multiple traces that lead to the same state using the same instructions, then the probability of reaching that state increases compared to only considering each trace individually. To this end, we track sets of traces only up to a certain number k of subsequent instructions and record instruction sequences where at least two traces lead to the same state. We can precompute this kind of look-ahead for a given decision frame independent of the origin and destination, that is we have to perform the precomputation only once for any given network. For each state s, we test all traces with at most k subsequent instructions. If two traces meet at the same state s' after executing the same instructions, we explicitly store them as a new arc $a = ss'$ in the decision frame. Instead of just one label, we associate with those arcs the sequence of turn labels that leads from s to s', and the sum of each traces' probability as the transition probability. Obviously, this precomputation is exponential in k, but if we set k to a reasonably small number then this precomputation is feasible.

In the experiments in the next section, we use $k = 5$. Since we only look ahead k instructions, it is clear that we might miss those traces that lead to the same states after more than k instructions. Therefore, the probability we get is still only a lower bound of the actual probability – but a better lower bound than what we get without the look-ahead computation. Figure 6 shows an example for $k = 2$.

Finally, we note that as suggested by Haque et al., we break ties during shortest-path search based on the metric

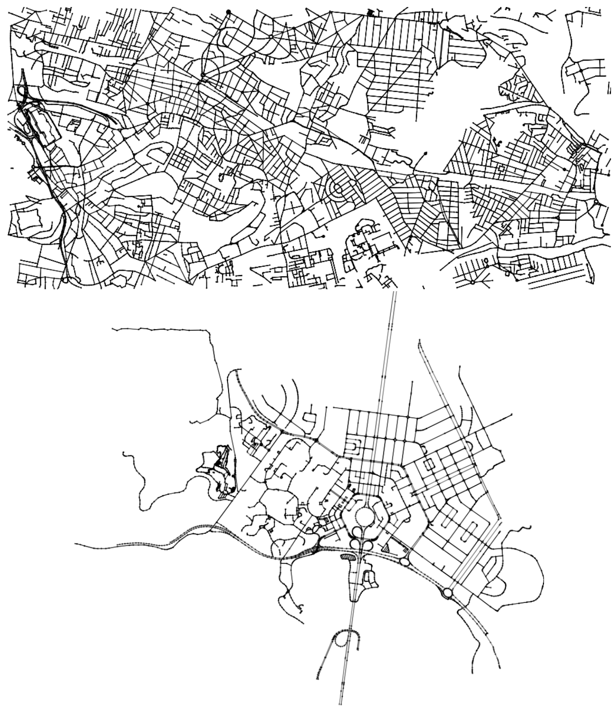

Figure 7: The used road networks: Part of the Paris area and Canberra.

route length of paths [6] (independent of the used cost function).

6. EXPERIMENTAL EVALUATION

We performed an evaluation of the proposed algorithms and compared them with the previous unreliability approach of Haque et al.. As input data we used OpenStreetMap data of parts of Paris and Canberra (the networks can be seen in Figure 7). We compare the following shortest-path algorithms:

1. the unreliability cost function by Haque et al. (1) on the decision frame,

2. the probability cost function (2) on the decision frame, and

3. the probability cost function (2) on the decision frame with a look-ahead of $k = 5$.

Further, we ran 2., and 3. on decision frames with weak and strict assumption, respectively.

For both Canberra and Paris we used 10 000 pairs of distinct origin/destination states, and ran all algorithms twice: once with $\mathcal{STAR}_2^r$ and once with $\mathcal{STAR}_4^r$. The qualitative route instructions computed by these algorithms where then accurately evaluated according to their expected lengths and success probabilities. Route length and success probability are sensible criteria that apply to all the three considered approaches. Although the unreliability cost function by Haque et al. (1) only indirectly optimises success probabilities, it is certainly of interest to compare it with our probabilistic planning approach.

Concerning the exponential-time precomputation for the look-ahead, we note that for $k = 5$, a simple implementation in the Python programming language computed the look-ahead within an hour for each used decision frame on an Intel Pentium Dual-Core with 2.0 GHz. A more sophisticated implementation in a lower-level language should significantly reduce the required time.

Table 1: Percentage of route instructions with 100% success probability generated by the probability cost function without look-ahead.

	$\mathcal{STAR}_2^r$		$\mathcal{STAR}_4^r$	
	Canberra	Paris	Canberra	Paris
strict	64.86%	84.80%	89.50%	95.43%
weak	65.03%	85.12%	89.68%	95.58%

6.1 Approximation Quality

Since the shortest-path search with the probability cost function only computes lower bounds, it is of interest how often the bound reported by the search is the same as the success probability of the generated instruction.

With the strict interpretation of turn labels and without considering a look-ahead, it turns out that in Canberra, for 94.7% of the pairs the search reports the accurate probability, independent of the qualitative turn labels used. For the Paris set, the accuracy is 99.0% when using $\mathcal{STAR}_2^r$ (using $\mathcal{STAR}_4^r$ this improves marginally to 99.2%). When relying on the look-ahead of $k = 5$, the accuracy improves further. Here, for Canberra only the $\mathcal{STAR}_2^r$ representation causes one case where the probability was underestimated. Using $\mathcal{STAR}_4^r$ the values reported by the search always match the evaluation. On the Paris set the search achieves an accuracy of 99.6% for both qualitative turn representations.

These results demonstrate that the probability cost function provides a very good lower bound. Note that this empirical evidence does not prove that the search produces near-optimal results. However, given the structure of real-world networks it would be surprising if more elaborate methods could produce drastically improved results.

6.2 Cost Functions: Unreliability vs. Probability

The empirical evaluation presented by Haque et al. showed that in most cases they found paths with an unreliability value of 0 or 1. We can confirm their findings; the majority of route instructions generated with the probability cost function without precomputed sequences already has a success probability of 100% (cf. Table 1). Moreover, the instructions constructed with the unreliability cost function have the same success probability as those generated with the probability cost function for the strict interpretation and no look-ahead. There is also little difference in the expected route length to the destination between these two approaches independent of the qualitative direction relations used. It should be noted here that Haque et al. originally consider arbitrary graphs as input. Without a reduction to decision nodes it is not obvious whether two instruction equivalent outgoing arcs lead to the same decision node, and thus their cost function would lead to worse results in such cases. Since we apply their cost function to decision frames, we implicitly see an improvement for their method.

As far as the qualitative representation is concerned, we observe that $\mathcal{STAR}_4^r$ results in very robust route descriptions that are clearly superior to those based on $\mathcal{STAR}_2^r$. In

Table 2: Percentage of improved non-perfect instructions through look-ahead. Percentage points of average improvement of the success probability given in brackets.

	$\mathcal{STAR}_2^r$		$\mathcal{STAR}_4^r$	
	Canberra	Paris	Canberra	Paris
strict	0.85%	12.76%	0.00%	4.38%
	(4.32)	(15.86)	(0.00)	(48.75)
weak	18.90%	33.27%	45.64%	1.81%
	(7.07)	(30.24)	(24.36)	(46.88)

Table 3: Average instruction length. Unreliability cost function (ucf), probability cost function (pcf), look-ahead (la).

	$\mathcal{STAR}_2^r$		$\mathcal{STAR}_4^r$	
	Canberra	Paris	Canberra	Paris
strict, ucf	29.24	48.43	25.55	45.47
strict, pcf	29.01	48.44	25.54	45.47
strict, pcf+la	29.02	48.24	25.54	45.40
weak, pcf	24.98	41.95	18.73	31.47
weak, pcf+la	24.59	42.16	18.92	31.47

particular, using $\mathcal{STAR}_4^r$ we obtain route instructions which are close to, or have a success probability over 90%.

In any case, these results raise the question, if the look-ahead approach can improve the results of the probability cost function over those of the unreliability cost function.

6.3 Look-Ahead

The look-ahead has a positive effect on those origin/destination pairs for which originally only route instructions with a non-perfect success probability (i.e., less than 100%) were found (the others cannot be improved anyway). As Table 2 shows, route instruction for many pairs can be improved. It can also be seen that weak interpretations benefit much more from precomputing traces, since it is more likely that wrong turns do not lead to an immediate stop and thus more "look-ahead arcs" can be used.

6.4 Weak Interpretation

It is not surprising that the weak interpretation model leads to shorter route instructions as can be seen in Table 3. Moreover, they do not increase the expected path length to the destination (cf. Table 4). This shows that the weak interpretation of route instructions is a successful approach to shorten route instructions without increasing metric route length. Further, we note that route instruction generated with the strict interpretation model are also valid route instruction for the weak interpretation model, and thus a weak interpretation can never decrease success probabilities.

Table 4: Expected route length to the destination. Unreliability cost function (ucf), probability cost function (pcf), look-ahead (la).

	$\mathcal{STAR}_2^r$		$\mathcal{STAR}_4^r$	
	Canberra	Paris	Canberra	Paris
strict, ucf	1876.52	4643.94	2033.27	4570.45
strict, pcf	1876.99	4644.05	2033.31	4570.45
strict, pfcf+la	1877.74	4640.00	2033.31	4524.97
weak, pcf	1890.28	4922.57	2035.32	4530.08
weak, pcf+la	1802.67	5012.55	2095.97	4525.49

For completeness, we note here that Haque et al. already showed that the length of routes generates with the unreliability cost function is on average about 10% longer than the shortest (metric) path.

7. FUTURE WORK

It is clear that decision frames, or more generally MDPs, are a very powerful framework for modeling and generating route instructions, but within this paper we have only used simplistic agent models similar to previous work [6].

It would be interesting to use our approach to represent a more plausible cognitive agent model. Such a model should ideally cover missing turns and visual saliency of outgoing arcs to produce more accurate probability distributions. In particular, we have assumed that agents have a precise understanding of qualitative terms, whereas in practice we should assume more varied views. Further, it would be beneficial to account for more than just turn instructions, e.g., by referencing landmarks. Another interesting aspect is to deal with the influence of other road users on the behaviour of agents following an instruction, as considered in traffic simulation approaches [13].

8. DISCUSSION AND CONCLUSIONS

No one likes to take the wrong turn and even if it happens only rarely, we still like to avoid it altogether. In this paper we looked at the problem of finding a robust route instruction to go from an origin to a destination in a given street network. As opposed to previous approaches and their evaluations, we propose a probabilistic measure for evaluating the quality of a route description. In particular, we propose to use the probability of successfully reaching the destination using a given route description. The higher the probability, the more robust a route description.

While it is possible to find the most robust route instruction, it is expected to be exponential time in the number of states to compute, and is therefore not practical. Instead, we developed a method for approximating the lower bound of the success probability. Our method is based on a precomputation that performs a look-ahead in the graph to check whether different routes lead to the same states, when executing the same instructions. If we limit the look-ahead to a small number of instructions, it is practical to precompute such instruction-equivalent paths in the decision frame. This pre-processing has to be done only once for a given street network and can then be used for any origin/destination pair.

We evaluated our method and compared it with the previously best method of Haque et al. [6] for computing the most reliable routes, i.e., routes with as few as possible ambiguities. We presented different variants of a simple agent model, one where all instructions have to be "strictly" executed (which is the standard in previous work), and we also proposed a variant where instructions are executed at the next possibility (we call it "weak"). We tested both approaches using different sets of instructions, i.e., different sets of qualitative direction relations. We used a large number of origin/destination pairs in two different cities, Paris and Canberra. It turned out that both methods produced robust route instructions in most cases. This is not surprising, as it usually does not happen regularly that we have ambiguous instructions and take wrong turns. However, in those cases where Haque et al.'s method does not produce a reliable route without ambiguities, it turned out that our look-ahead method produced more robust route instructions. In some settings almost 50% of the unreliable routes could be improved, in some settings the improvement is by almost 50 percentage points on average. Some interesting results of our evaluation are that the weak execution of instruction that we proposed leads to more robust route descriptions with much shorter instruction lengths. It also allows for much more natural route descriptions. The average improvement of the look-ahead method was in most settings significantly better for weak execution than for strict execution. Finally, it is not surprising that we found finer grained direction relations to lead to more reliable routes. The representation based on the findings of Klippel and Montello [7], which only differentiates between eight direction relations, already results in very robust instructions.

We only used route instructions that are based on purely qualitative direction relations. It is clear that route instructions can be further improved by using distance measures, landmarks, or street names, but these are often hard to estimate or might not be clearly visible. Qualitative direction relations form the basis of any route instructions, they are easy to communicate and intuitive to understand. We were therefore interested in what levels of robustness we can achieve by only considering direction relations. As such our results form the baseline for further improvements using more sophisticated instructions. Our results show that this baseline has been increased by the look-ahead method we proposed.

9. ACKNOWLEDGMENTS

The authors thank Bernhard Nebel and Stefan Wölfl for helpful discussions, and the anonymous reviewers for constructive feedback on an earlier draft of this paper.

The map data used in the evaluation was taken from the OpenStreetMap project: `http://www.openstreetmap.org/`.

This work was supported by DFG *SFB/TR 8 Spatial Cognition* project R4-[LogoSpace], an ARC Future Fellowship (FT0990811), and a joint Go8/DAAD project (D/08/13855).

10. REFERENCES

[1] R. Bellman. A markovian decision process. *Indiana University Mathematics Journal*, 6:679–684, 1957.

[2] H. Cuayáhuitl, N. Dethlefs, L. Frommberger, K.-F. Richter, and J. Bateman. Generating adaptive route instructions using hierarchical reinforcement learning. In *Spatial Cognition*, volume 6222 of *LNCS*, pages 319–334, 2010.

[3] R. Diestel. *Graph Theory (Fourth Edition)*. Springer, 2010.

[4] M. Duckham and L. Kulik. "Simplest" paths: Automated route selection for navigation. In *COSIT*, volume 2825 of *LNCS*, pages 169–185. Springer, 2003.

[5] M. Duckham, S. Winter, and M. Robinson. Including landmarks in routing instructions. *Journal of Location Based Services*, 4(1):28–52, 2010.

[6] S. Haque, L. Kulik, and A. Klippel. Algorithms for reliable navigation and wayfinding. In *Spatial Cognition*, volume 4387 of *LNCS*, pages 308–326. Springer, 2006.

[7] A. Klippel and D. R. Montello. Linguistic and nonlinguistic turn direction concepts. In *COSIT*, volume 4736 of *LNCS*, pages 354–372. Springer, 2007.

[8] K. L. Lovelace, M. Hegarty, and D. R. Montello. Elements of good route directions in familiar and unfamiliar environments. In *COSIT*, volume 1661 of *LNCS*, pages 65–82. Springer, 1999.

[9] T. Marciniak and M. Strube. Classification-based generation using TAG. In *Natural Language Generation: Proceedings of INLG-2994*, pages 100–109. Springer, 2004.

[10] T. Marciniak and M. Strube. Modeling and annotating the semantics of route directions. In *Proceedings of the 6th International Workshop on Computational Semantics*, pages 12–14, 2005.

[11] D. M. Mark. Automated route selection for navigation. *Aerospace and Electronic Systems Magazine*, 1(9):2–5, 1986.

[12] C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of markov decision processes. *Mathematics of Operations Research*, 12(3):441–450.

[13] M. Pursula. Simulation of traffic systems - an overview. *Journal of Geographic Information and Decision Analysis*, 3(1):1–8, 1999.

[14] J. Renz and D. Mitra. Qualitative direction calculi with arbitrary granularity. In *PRICAI*, volume 3157 of *LNCS*, pages 65–74. Springer, 2004.

[15] J. Renz and S. Wölfl. A qualitative representation of route networks. In *ECAI*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 1091–1092. IOS Press, 2010.

[16] K.-F. Richter and M. Duckham. Simplest instructions: Finding easy-to-describe routes for navigation. In *GIScience*, volume 5266 of *LNCS*, pages 274–289. Springer, 2008.

[17] M. Westphal, S. Wölfl, B. Nebel, and J. Renz. On qualitative route descriptions: Representation and computational complexity. In *IJCAI*, pages 1120–1125. IJCAI/AAAI, 2011.