# MARCO: A Modular Architecture for Following Route Instructions

**Matt MacMahon**
adastra@mail.utexas.edu
Department of Electrical and Computer Engineering
Intelligent Robotics Laboratory
University of Texas at Austin *

## Abstract

Understanding and following verbal instructions is a crucial human ability, allowing quick transfer of knowledge about acting in the world. This paper presents the MARCO modular architecture for reasoning about and following route instructions. The *Syntactic Parser* parses route instruction texts. The *Content Framer* interprets the surface meaning of each sentence. The *Instruction Modeler* combines information across phrases and sentences: imperatively, as a representation of what to do under which circumstances, and declaratively, as a model of what is stated and implied about the world. The *Executor* reactively interleaves action and perception, executing the instructions in the context of the environment. The *Robot Controller* adapts low-level actions and local view descriptions to the particular follower's motor and sensory capabilities. When the follower can interact with a director, the follower can plan dialog moves that increase its understanding and likelihood of reaching the destination.

## Introduction

People use verbal instructions to guide one another through a task, even a task that the follower does not have the knowledge or ability to accomplish alone. Instructions convey both immediately practical imperative knowledge – what to do – and declarative knowledge – what exists in the world and to interact with it. A *route instruction* is an instruction intended to guide a mobile agent toward a spatial destination.

Route instructions are limited enough to be tractable, are applicable to useful real-world tasks with clear criteria for evaluation, and can draw on existing research into spatial reasoning. Small and simple language models and ontologies can cover many common and useful examples. For instance, only four core actions are essential to following basic instructions: turning at a place, traveling between places, verifying a view description against an observation, and terminating the instruction following.

Route instructions are useful: everyday, people use route instructions to travel along previously unknown routes and there is a large industry devoted to generating driving instructions. The criteria for success are clear and easily measured: when the follower has reached the destination knowingly, with certainty, and with little effort. Finally, route instructions can build on years of research about spatial reasoning in robotics, artificial intelligence, and cognitive psychology.

A *route instruction set* is the collection of route instructions that describes a route. A route instruction set is also referred to as "route directions", "route guidance", or "directions"[1]. Route instructions are a special case of verbal instructions, which include recipes, assembly instructions, and usage manuals. While route instruction sets may include verbal, gestural, and pictorial components, (respectively using words, body gestures, and map elements) this work focuses on verbal route instructions. The route instruction provider is referred to here as the *director* and the agent following a route instruction set as the *follower*.

This paper introduces the MARCO architecture for understanding and executing natural language route instructions. The first two modules of the architecture, the *Syntactic Parser* and *Content Framer*, are applicable to any natural language understanding problem. They interpret an unstructured text string into a representation of word meaning and phrase and sentence structure. The *Instruction Modeler* integrates meaning across phrases and sentences and can utilize prior domain knowledge. The *Executor* module reactively sequences discrete low-level actions, while the *Robot Controller* realizes those commands in a continuous perceptual and motor space. Both of these modules are task-independent at the architecture level. An architecture diagram with implemented modules is shown in Figure 2.

Two Instruction Modelers are described within: the *Compound Action Specifier* models the imperative content of general instructions, while the *SSH Route Topology Modeler* models the declarative content of spatial descriptions. Each modeler can process sentences which have either imperative and declarative surface forms. Finally, I present related work on autonomously understanding and following route instructions.

*This work was funded under ONR work order N0001405WX30001 for the NRL Research Option, Coordinated Teams of Autonomous Systems. Thanks to the Intelligent Robotics Laboratory and the Shape & Space Laboratory at University of Texas at Austin and the Naval Research Laboratory's NCARAI.

[1]The term "route instruction" avoids confusion with the terms "cardinal direction" (north, west, etc.) and "relative direction" (left, up, etc.), components of route instructions.

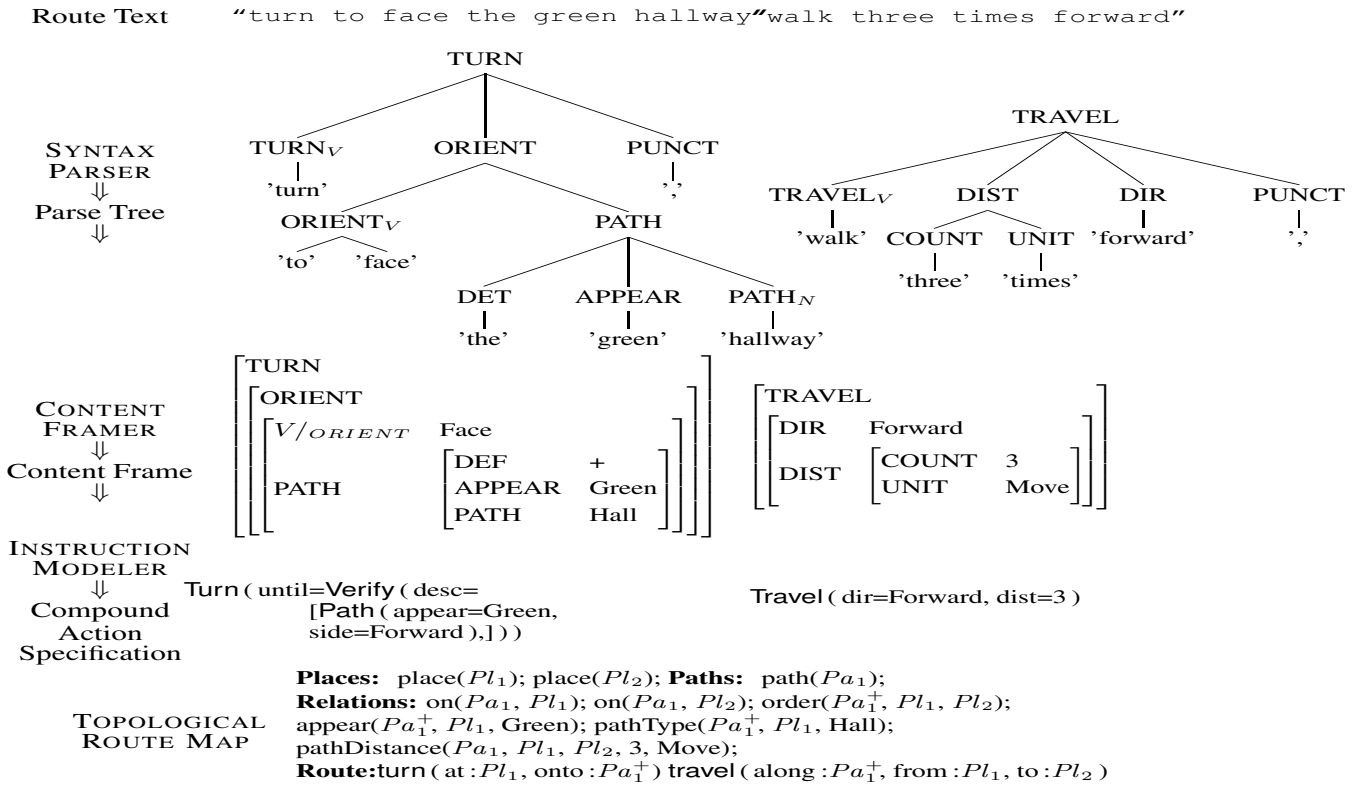


Figure 1: Representations used in understanding two example sentences by the MARCO framework

## Understanding and Following Instructions

MARCO is an architecture for understanding and following natural language route instructions. It builds on work in computational linguistics and intelligent architectures. The language understanding part of the architecture builds on ideas from the Nautilus natural language understanding system (Wauchope *et al.* 1997; Perzanowski *et al.* 2001; Simmons *et al.* 2003) and from Reiter and Dale's natural language generation architecture (1997). The execution and control modules build on top of work in intelligent robot architectures (Bonnasso *et al.* 1997; Simmons & Apfelbaum 1998). The architecture draws ideas from multi-tiered intelligent architectures like 3T and an early implementation was programmed in TDL.

Figure 1 shows the intermediate representations used by the MARCO framework to understand a route instruction text. Figure 2 shows the relation of the natural language understanding and robot control parts of the architecture.

The parser, here from a Probabilistic Context-Free Grammar, parses the unstructured route instruction text. This tree can be transformed into an *content frame* representation, Content frames are a recursive representation of the surface meaning of an utterance, after taking out incidental features, such as phrase order and word selection. Content frames can represent the information captured in verbs and verb arguments. Content frames are analyzed to extract *compound action specifications*, which are an imperative model of the relations between *low-level actions* and *view descriptions*. The compound action specifications guide the follower when navigating the route.

MARCO also integrates individual utterances to model the declarative semantics of the route instructions. The instructions imply a model of the spatial structure and appearance of the route: its topology of paths and places and their perceptual attributes. A powerful representation of the route can be found in the *Spatial Semantic Hierarchy*, as explained in the next section. A sketch of converting content frames into a topological route representation follows the brief introduction to the SSH.

## Syntax Parser

The *Syntax Parser* parses the raw route instruction text. A portion of route instructions from a human-subject study were hand-labeled for semantic features. (See Table 1 for examples). The texts were cleaned by hand to remove capitalization, standardize punctuation, and fix typos that split or merge words (affect the parse structure). A *referring phrase* is a noun phrase that refers to some entity or attribute being described, analyzed on its semantic content instead of its syntactic makeup (Kuipers & Kassirer 1987). By tagging the referring phrases and verbs in a set of route instructions with semantic annotations to the phrase and argument types, this analysis characterized the surface meaning of route instruction utterances.

From the hand-labeled text, a Probabilistic Context Free Grammar (PCFG) was trained to parse and semantically tag

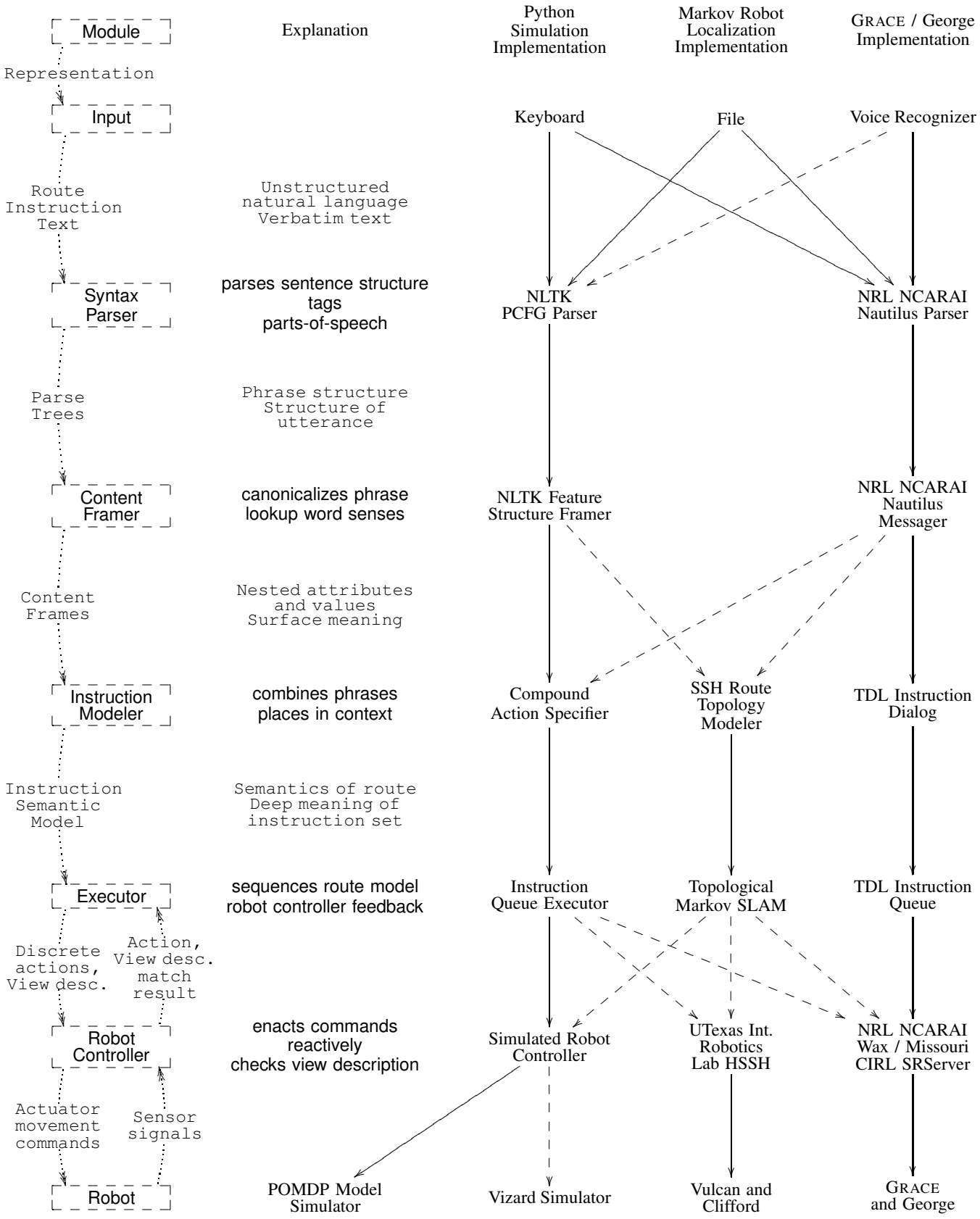


Figure 2: Module interactions of the MARCO framework. Solid arrows are implemented, dashed arrows are planned.

| | |
|---|---|
| *EDA* | turn to face the green halllway, walk three times forward, turn left, walk forward six times, turn left, walk forward once |
| *EMWC* | Follow the grassy hall three segments to the blue-tiled hall. Turn left. Follow the blue-tiled hall six segments, passing the chair, stool and bench, to the intersection containing the hatrack. Turn left. Go one segment forward to the corner. This is Position 5. |
| *KLS* | take the green path to the red brick intersection. go left towards the lamp to the very end of the hall. at the chair, take a right. at the blue path intersection, take a left onto the blue path. at the coat rack, take another left onto the plain cement. at the end of this hall at the corner, you are at position 5 |
| *KXP* | head all the way toward the butterfly hallway, keep going down it until you reach a dead end square area. pos 5 is in the corner to the left as you enter the square block. |
| *TJS* | go all the way down the grassy hall, take a left, go all the way down the blue hall until you see a coat rack, take another immediate left. |
| *WLH* | from four face the grass carpet and move to the hat rack, turn left and move onto the blue carpet, walk past two chairs and to the lamp, turn left, move into the corner such that the lamp is behind you and to your right you see a gray carpeted alley |

Table 1: Example directions by different route directors. All directions describe travel from the same starting and ending positions in the same environment. Instructions are as typed by the directors, including typos (e.g. "halllway").

route instruction texts. This parser labeled additional route instructions. The automatically tagged route instructions are then hand-corrected. The tagged trees help bootstrap the process of learning the full language model.

## Content Framer

The *Content Framer* extracts the local surface semantics of words and phrase structure from the syntactical parse tree. These *content frames* model the phrasal structure and word meaning of what was said by dropping text token ordering info, labeling parts of speech, and dropping punctuation. The Content Framer looks up nouns, verbs, adjectives and adverbs in WordNet (WordNet 2005; Fellbaum 1998), an ontology for English. The output of this processing stage, the content frame, is the structure of each instruction phrase and the words' surface meaning.

## Instruction Modeler

The *Instruction Modeler* integrates content frames into a model of the entire instruction set. *Imperative* models, such as Compound Action Specifications, model what the follower should do. *Declarative* models, such as Topological Route Map, model the world.

### Compound Actions Specifier

The *Compound Actions Specifier* builds an imperative model of the instructions from (1) general knowledge of the verbs and prepositions of route instructions and (2) domain- and robot-specific knowledge of perceiving and acting in the environment. Prepositions and verbs of motion in route instructions are relatively independent of where the instructions are followed (the environment) and who is following them (the agent). On the other hand, perceiving and acting on objects and object attributes are dependent on the agent and environment.

Route instructions require at least four low-level action types. To *turn* is to change the agent's orientation (pose) while remaining in the same location. To *travel* is to change the agent's location without changing orientation along a path. To *verify* is to check an observation against a description of an expected view. Finally, to *declare-goal* is to terminate instruction following by declaring the agent is at the destination. Additionally, route instructions contain *view descriptions*, which partially describe what the follower will see in certain poses along or near the route.

Some route instructions may contain other action types, such as "`open the door`" or "`take the elevator to the 2nd floor`". However the four action types above are both necessary and sufficient for following many route instructions.

The *compound action specification* captures the transitions between low-level actions on perceiving observations matching view descriptions. Content frames model the local surface semantics of the phrase structure of the instruction utterances. Compound action specifications model the structure of what the follower is told to do by each utterance, pulling together information across phrase boundaries. Evaluating the full meaning of each utterance is deferred until the utterance's environmental context comes into play as the follower proceeds along the route.

Prepositional phrases and verb objects translate to pre-conditions, while conditions, and post-conditions in compound action specifications. For instance, phrases may describe which path to take, how far to travel, or the view that will be seen when the action is accomplished. Other action verbs have some arguments implicit, such as "`face`" implies turn until the description is matched. Conditional actions are modeled by embedding actions. For instance, note the implicit travel action before the turn in "`At the corner, turn left.`"

Nouns, adverbs, adjectives are grounded in an environment- and follower-specific concept knowledge base. Each known concept is grounded in knowledge of how it affects the agent's interaction with the world. For instance, the concept of an intersection is linked to the code that segments intersections from the observation stream and classifies their type. By examining the local path

configuration, the code can classify dead end, "T", and corner intersections (Kuipers *et al.* 2004).

The WordNet ontology is used to look up the nearest synonym or more abstract hypernym in the knowledge base to each mentioned word. This allows the agent to look for a more general object, such as "couch", when it does not have knowledge of a specific sub-type, such as "futon."

## Executor: Interleaving Action and Perception

The *executor* breaks down the model of the instructions into low-level turn, travel, verify, and declare-goal actions. The executor interprets each compound action to execute low-level actions, including verifying view descriptions to find the appropriate context for an action. For instance, the executor continues to turn until a verify returns a view description has matched.

The executor is equivalent to, and can be implemented by, the reactive task sequencing tier of the standard three-tiered intelligent architecture (Bonnasso *et al.* 1997). One implementation of the MARCO executor is written as tasks in TDL, CMU's Task Description Language (Simmons & Apfelbaum 1998). The executor reasons at the Causal and Topological levels of the SSH (Kuipers 2000).

The executor algorithm need not know anything about how the follower moves through the environment or how the view descriptions are verified. Those actions and observations may be opaque to the executor stage, in an ontology it can pass through but not comprehend. One executor module can be run with different robot controllers.

The simplest route instruction executor is the *Instruction Queue Executor*. It steps through a list of instructions, attempting to execute each fully before moving to the next. Executing each instruction may consist of ensuring various preconditions, distance estimates, and postconditions are met. Each condition may entail moving then verifying the resulting view matches a view description.

This instruction following algorithm may be replaced with more sophisticated algorithms that leverage previous knowledge of the environment map, or knowledge of how directors describe spatial route navigation. An executor that reasons about the possible route topologies of a route instruction set is described later.

## Robot Controller

The *Robot Controller* module executes the low-level turn, travel, verify, and declare-goal actions. Robot controllers present a common interface to the executor, with domain- and agent- dependent implementations. This stage moves the agent for turns and travels and can verify whether or not a view description matches an observation.

The robot controller acts at the level of the control tier of the 3T intelligent architecture. The robot controller in GRACE is written as commands and monitors in TDL. The ontology of the robot controller is the Control level of the SSH ontology: continuous sensori-motor experience and control laws.

For instance, consider the instruction "`Move to the blue hall.`" The executor executes the robot's travel action and then the robot's verify action. A controller for a robot with peripheral vision would execute the verify by analyzing the periphery of the view while facing along the travel direction. A controller for a robot without peripheral vision would turn to face each hallway, verify if it is blue, and turn back, all as part of the verify action.

## SSH Route Topology Modeler

Compound action specifications and the Instruction Queue Executor model the route instructions as a list of imperative commands. The follower concentrates on inferring what the director intended the follower to do. However, route instructions also state or imply a spatial route layout. Another approach to following route instructions is to extract the implied map of the route from the route instructions. The follower infers what the director intended the follower to know about the route map.

Route instructions can be represented naturally in the *Spatial Semantic Hierarchy* (*SSH*) (Kuipers 2000). Route instruction texts describe causal and topological structures annotated with metrical and rich view (object and landmark) information. A *view* abstracts the sensory image to a symbolic representation. In the SSH, the *causal* level discretizes continuous control motions into reliable, high-level actions. At the causal level, motions are abstracted to either *turn* or *travel* actions. A turn action changes orientation within a place, while a travel moves the agent from one place to another.

The *topological* level of the Spatial Semantic Hierarchy represents the environment as *places*, *paths*, and *regions* and the topological relations of *connectivity*, *path order*, *boundary relations*, and *regional containment* (Remolina & Kuipers 2004).

Route instructions can be represented by the SSH causal and topological ontologies, with the actions annotated with metrical and view attributes. Route instructions are expressed in both declarative and procedural language, e.g. both "`As you walk down the hall, you will see a lamp,`" and "`Walk down the hall past the lamp.`" Route instructions include both causal actions ("`Walk forward three steps.`") and topological actions ("`Face along the blue path and follow it to the intersection with the brick hall.`").

The Instruction Queue Executor acts at the causal level, although the robot may recognize topological entities such as paths and intersections. Each turn and travel moves the follower to the next pose and each verify compares the view against the view description. However, moving up the ontology, an executor can reason about the spatial layout of the route. Explicitly reasoning at the topological level can help handle ambiguity in the language, interpretation, observation, execution, and map learning.

## Inferring an SSH Topological Route Map

The *SSH Route Topology Modeler* builds a declarative model of the spatial layout of the route from the content frames.

It reasons about the semantics (meaning), anaphora (co-reference resolution), and discourse properties (inferring the conversational intent of an utterance) of route instruction texts.

Both Grice's conversational maxims (1975) and Sperber & Wilson's Relevance Theory (2004) are linguistic theories of discourse – how sentences are strung together to form broader meaning. Each theory assumes that a cooperative speaker conveys meaning by crafting the discourse to clearly and concisely carry across the necessary concepts.

I propose sets of axioms and heuristics for a follower to infer the topological map. Once the makeup and structure of the topological map is known, the map can be displayed graphically as a user or developer interface.

One set of axioms instantiates the entities, or *resources*, required by complex concepts. For instance, "the corner" in these instructions usually refers to the intersection of two paths, each terminating at the corner. Other *resource axioms* include "each path connects at least two places, "each intersection is the meet point of at least two paths", and "a left or right turn action implies changing paths".

Another class of axioms tracks when information is explicitly mentioned and when it can be inferred. These *conversational axioms* include "When two consecutive turn commands specify their location, these are distinct places separated by an unstated travel action." and "When a turn is immediately followed by a travel without a location mentioned, the travel starts where the turn results." These help resolve anaphora issues: for each mention of an entity, is it new or previously mentioned?

## Why Model Route Instructions in the SSH?

The Spatial Semantic Hierarchy is a rich, but complex model for reasoning about space. Is the SSH excessive or just complex enough for the needs of following route instructions?

First, the Spatial Semantic Hierarchy is a well-developed, theoretically grounded cognitive spatial model. The SSH is supported by having working implementations on mobile robots (Kuipers *et al.* 2004), and by modeling human navigation and spatial learning performance (Kuipers, Tecuci, & Stankiewicz 2003).

### Partial knowledge of the environment

The SSH is a lattice of representations that can model states of partial spatial knowledge, including partial maps as the environment is learned. For instance, the SSH can easily model knowing only some routes instead of the overall spatial network; knowing that two paths meet a corner, but not the relative turn direction between the two; or knowing some path segment distances but not others.

Route instructions must select certain aspects of the route to describe, so the texts by nature underspecify the route. Instruction texts do not often mention the absolute location of everything in the environment. The SSH can model the director eliding a turn direction, occasionally mentioning a distance, and selectively noting landmarks and other perceptual features.

### Mapping while route following

The Spatial Semantic Hierarchy enables navigating despite ambiguity in the map. In the domain of Simultaneous Localization and Mapping (SLAM), the agent is attempting to learn the map of the environment while navigating it. Given an experience trace, more than one map is always possible in an environment with *place aliasing*, or perceptually identical places. For instance, each familiar place encountered, may be a new place that merely looks similar. The SSH can explicitly reason about the complete set of possible maps to navigate or produce the most likely map (Remolina & Kuipers 2004; Kuipers *et al.* 2004).

Resolving linguistic and perceptual ambiguity while following route instructions is analogous to resolving perceptual aliasing. Route instructions do not completely specify the route, leaving spatial ambiguity. For instance, a turn direction may be unspecified, leaving topological ambiguity.

The SSH can handle the ambiguous maps derived from underspecified or linguistically ambiguous route instructions using the same reasoning mechanism that handles the spatial ambiguity in SLAM. That is, the partial, ambiguous map of the environment derived from language understanding and the partial, ambiguous map learned from exploration can be represented and reasoned about in the same way by the same cognitive processes.

### Comparison with other spatial representations

The SSH has several advantages over global metrical maps, such as occupancy grids. With a known global metrical map, a robot can follow a set of route instructions, although finding the route in the metrical map will require processing or a hybrid representation. However, since the global metrical map is a precise representation of what has been observed, it is difficult to reason about unseen, unknown places. To navigate with a global occupancy grid alone, the agent must know the exact dimensions of the hallways and rooms, which are never described in natural language route instructions. Generating route instructions using a metrical map is possible, but using a representation that more closely follows people's cognitive maps should produce instructions which are more robust and easy to follow.

The advantage of this representation is cognitive parsimony. Topological maps represent not only routes, but also networks of places and paths. The reasoning skills needed to create, integrate, manipulate, and utilize topological maps are already documented in people and derived in the SSH (Remolina & Kuipers 2004). Additionally, a topological route map eases integrating the route instructions with known maps and primes learning the environment through experience.

## Planning the Route Instruction Dialog

When the follower interacts with the route director, it can also optimize dialog. The follower decides when to interrupt the route director with a question, trading off the cost of asking a question against the likelihood of missing the destination. Speech acts or dialog moves can clarify,

disambiguate, or fill in needed knowledge. Nursebot reasoned about when and how to extend the dialog to increase understanding (Roy, Pineau, & Thrun 2000). Rather than exploring the world to fill in gaps and ambiguities in the instructions, the follower can simply ask for route details and clarification.

The follower can decide when to interact and ask for route instructions, balancing the benefit of more likely reaching the destination against the cost of interrupting and querying someone. This adds actions of finding a knowledgeable director and initiating dialog, as well as the state the location of directors. Thus, the most capable agent must optimally chose when to seek help by asking for route instructions, optimally query for route instructions, and optimally navigate using the route instructions.

## Related Work in Automated Instruction Following and Analysis

Several software systems analyze or follow route instructions. Riesbeck's system evaluated route instructions by high-level characteristics, independent of the environment (1980).

In examining plans-as-communication, Agre & Chapman considered the inverse: communications as underspecified plans (1990). They showed how route instructions, such as a set from Agre's flat to the metro station, do not uniquely specify action sequences, but instead constrain navigation by providing a plan skeleton, with exploration sub-goals the follower must accomplish. Chapman followed up on this theoretical paper in implementing the "Sonja" system, which interpreted spoken advice and instructions to better fight the monsters in her virtual dungeon (1990).

Alterman, Zito-Wolf, & Carpenter implemented a system which reactively replans to read the instructions when its naïve plan proves inadequate (1991). The system, FLOABN, operated in a discrete event simulation. Example instructions focused on different ways of paying for phone calls.

Webber *et al.* looked at the broader question of inferring an intended plan from any instructions (1995). This work examined the linguistic and domain knowledge needed to get a virtual agent to follow instructions from various domains. Di Eugenio reports on the language system of this work (1998). Her software analyzes general instructions, such as craft guides, matching the text against a plan library using plan recognition. The system was integrated with the AnimNL system, which is a VR animation able to simulate several tasks. The big lack in the system is an inability to synthesize meaning across the discourse, instead, it interprets each sentence in its own context.

Müller *et al.* implemented a system which can follow a formal route description through an environment, with the intention of adding on a natural language understanding system (2000). Descriptions follow the Tversky and Lee analysis, specifying where to turn or switch paths (Tversky & Lee 1998).

### GRACE and GEORGE in the AAAI Robot Challenge

Perzanowski *et al.* implemented a system that combined a speech recognizer, a deep parser, a dialog model, hand gesture recognition, and a Palm Pilot control interface (1998; 2001). A user could command the robot to move around a mapped, small-scale space by speaking and gesturing.

GRACE extended this architecture, adding the ability to follow a route instruction series through an unmapped, unknown large-scale space (Simmons *et al.* 2003). The right side of Figure 2 shows GRACE's route instruction following architecture. GRACE navigated through a conference center by asking for and following route instructions. GRACE and GEORGE could string together several simple commands, using an instruction queue executor. They also handled implied new interim destinations ("`Take the elevator`").

In 2002, GRACE successfully, though haltingly, completed the Robot Challenge at AAAI 2002 (Simmons *et al.* 2003). The 2003 robots were beset by hardware, software, and communications problems that illustrate the need for more user visibility into the state of the system. Still, the robots were directed down a hallway, up a ramp and through a narrow doorway, and across an exhibition hall.

GRACE and GEORGE had several major limitations. Most debilitating, the commercial speech recognition system was unreliable. The vocabulary and sentence structure were limited so only a trained operator could direct the robots. The navigation planning code relied on having a completed global metrical map, so navigating to unseen, unknown locations was extremely fragile. Crowds of people forming shifting walls further confused the robot.

GRACE and GEORGE did not reason to infer implied actions. They had only one interpretation of the instructions, although this was checked with the director. The robots did not estimate the likelihood of action success, but instead asked the director. The MARCO modules that probabilistically localize along the route were inspired by these difficulties.

## Conclusion

This paper presents MARCO, an architecture for understanding and following natural language route instructions. The MARCO architecture is modular, to apply different parsing, sense tagging, modeling, and execution, and robot control techniques. The first parsing and sense tagging steps of the process can be done without any knowledge of the environment, only a language model. Modeling particular directors will increase the accuracy and lower uncertainty.

Route instructions can be modeled as either an imperative model of what the follower should do, or a declarative model of how the route is spatially configured. In either case, previous knowledge of the environment is helpful, but not necessary. Both the imperative compound action specification and the declarative topological route map can model instructions that only loosely guide the follower towards the goal. Where attributes of the route are

underspecified, the follower relies on being situated in the environment to disambiguate among diverse alternative models. When the follower is receiving the instruction in dialog, however, there is another option: asking for clarification. The follower can plan whether it is likely to reach the goal with the instructions it has received or if it needs more information or different landmarks.

The follower delays full evaluation of the instructions until the environmental context comes into sight. By putting off the difficult decisions, perceiving the environment will often make one choice prominent. The embodied agent can procrastinate exhaustive modeling until the world resolves many ambiguities through inconsistency with the senses, allowing the follower to concentrate its efforts.

## References


[Agre & Chapman 1990] Agre, P. E., and Chapman, D. 1990. What are plans for? *Rob. & Auton. Sys.* 6:17–34.

[Alterman, Zito-Wolf, & Carpenter 1991] Alterman, R.; Zito-Wolf, R.; and Carpenter, T. 1991. Interaction, comprehension, and instruction usage. *J. of the Learning Sciences* 1(3/4):361–398.

[Bonnasso *et al.* 1997] Bonnasso, R. P.; Firby, R. J.; Gat, E.; Kortenkamp, D.; Miller, D. P.; and Slack, M. G. 1997. Experiences with an archtitecture for intelligent, reactive agents. *J. of Exptl. and Theo. AI* 9(1):237–256.

[Chapman 1990] Chapman, D. 1990. *Instruction use in situated activity*. Ph.D. Dissertation, MIT, Dept. of Elec. Eng. & Comp. Sci. , Cambridge, MA. Also Available as MIT, AI Lab. Technical Report 1204.

[Di Eugenio 1998] Di Eugenio, B. 1998. An action representation formalism to interpret natural language instructions. *Compl. Intelligence* 14(1):89–133.

[Fellbaum 1998] Fellbaum, C., ed. 1998. *WordNet: An Electronic Lexical Database*. Cambridge, MA: MIT Press.

[Grice 1975] Grice, H. P. 1975. Logic and conversation. In Cole, P., and Morgan, J. L., eds., *Speech Acts*, volume 3 of *Syntax and Semantics*. New York: Academic Press. 43–58.

[Kuipers & Kassirer 1987] Kuipers, B., and Kassirer, J. 1987. Knowledge acquisition by analysis of verbatim protocols. In Kidd, A., ed., *Knowledge Acquisition for Expert Systems*. New York: Plenum.

[Kuipers *et al.* 2004] Kuipers, B.; Modayil, J.; Beeson, P.; MacMahon, M.; and Savelli, F. 2004. Local metrical and global topological maps in the hybrid Spatial Semantic Hierarchy. In *Proc. of IEEE Intl. Conf. on Rob. & Autom. (ICRA-04)*. New Orleans, LA, USA: IEEE Computer Society Press.

[Kuipers, Tecuci, & Stankiewicz 2003] Kuipers, B. J.; Tecuci, D. G.; and Stankiewicz, B. J. 2003. The skeleton in the cognitive map : A computational and empirical exploration. *Env. & Behavior* 35(1):80–106.

[Kuipers 2000] Kuipers, B. J. 2000. The Spatial Semantic Hierarchy. *AI* 119:191–233.

[Müller *et al.* 2000] Müller, R.; Röfer, T.; Lankenau, A.; Musto, A.; Stein, K.; and Eisenkolb, A. 2000. Coarse qualitative descriptions in robot navigation. In Freksa, C.; Brauer, W.; Habel, C.; and Wender, K. F., eds., *Spatial Cognition*, volume 1849 of *LNCS*, 265–276. Springer.

[Perzanowski *et al.* 2001] Perzanowski, D.; Schultz, A. C.; Adams, W.; Marsh, E.; and Bugajska, M. 2001. Building a multimodal human-robot interface. *IEEE Intelligent Sys.* 16–21.

[Perzanowski, Schultz, & Adams 1998] Perzanowski, D.; Schultz, A. C.; and Adams, W. 1998. Integrating natural language and gesture in a robotics domain. In *Proc. of Intl. Symp. on Intelligent Control*, 247–252. Washington, DC: IEEE Computer Society Press.

[Reiter & Dale 1997] Reiter, E., and Dale, R. 1997. Building applied natural language generation systems. *J. of Natural Lang. Eng.* 1(1):1–32.

[Remolina & Kuipers 2004] Remolina, E., and Kuipers, B. 2004. Towards a general theory of topological maps. *AI* 152(1):47–104.

[Riesbeck 1980] Riesbeck, C. 1980. "You can't miss it!" : Judging the clarity of directions. *Cog. Sci.* 4:285–303.

[Roy, Pineau, & Thrun 2000] Roy, N.; Pineau, J.; and Thrun, S. 2000. Spoken dialog management for robots. In *Proc. of 38th Ann. Meeting of the ACL (ACL-00)*. Hong Kong, China: Morgan Kaufmann.

[Simmons & Apfelbaum 1998] Simmons, R., and Apfelbaum, D. 1998. A task description language for robot control. In *Proc. of IEEE/RSJ Intl. Conf. on Intelligent Robots & Sys. (IROS)*.

[Simmons *et al.* 2003] Simmons, R.; Goldberg, D.; Goode, A.; Montemerlo, M.; Roy, N.; Sellner, B.; Urmson, C.; Schultz, A.; Abramson, M.; Adams, W.; Atrash, A.; Bugajska, M.; Coblenz, M.; MacMahon, M.; Perzanowski, D.; Horswill, I.; Zubek, R.; Kortenkamp, D.; Wolfe, B.; Milam, T.; and Maxwell, B. 2003. GRACE: An autonomous robot for the AAAI Robot Challenge. *AI Magazine* 24(2):51–72.

[Sperber & Wilson 2004] Sperber, D., and Wilson, D. 2004. Relevance Theory. In Horn, L. R., and Ward, G., eds., *The Handbook of Pragmatics*. Oxford: Blackwell. 607–632.

[Tversky & Lee 1998] Tversky, B., and Lee, P. U. 1998. How space structures language. In Freksa, C.; Habel, C.; and Wender, K. F., eds., *Spatial Cognition*, volume 1404 of *LNCS*, 157–176. Berlin: Springer.

[Wauchope *et al.* 1997] Wauchope, K.; Everett, S.; Perzanowski, D.; and Marsh, E. 1997. Natural language in four spatial interfaces. In *Proc. of 5th Applied Nat. Lang. Proc. Conf. on*, 8–11.

[Webber *et al.* 1995] Webber, B.; Badler, N.; Di Eugenio, B.; Geib, C.; Levison, L.; and Moore, M. 1995. Instructions, intentions and expectations. *AI* 73(1–2):253–269. Spec. Issue on "Compl. Res. on Interaction and Agency, Pt. 2".

[WordNet 2005] 2005. Wordnet: a lexical database for the English language. http://wordnet.princeton.edu/.