Computational comprehension of spatial directions expressed in natural language

Pavlo Kapustin

Master thesis

Institute of Information Science and Media Studies

University of Bergen

June 2015

Table of contents

Disclaimer

Supervisor: B. Tessem

Chapters 1, 2, 3, 5, 6 and 7 are independent work of master project author. Parts of text in chapter 4 are taken from the article [1]. Text taken from the article is mostly written by the project author.

Software implementation of the intelligent navigation assistant prototype is independent work of the project author.

Software implementation of the framework is work of the project author with smaller contributions by M. Kapustin[1] (specifically, in the area of comprehension assessment heuristics).

Most of the original ideas of approach [1] are work of M. Kapustin, with contributions from the project author.

Design of the approach [1], and design of the framework based on this approach, is joint research by M. Kapustin and the project author.

Acknowledgements

The author would like to thank B. Tessem for guidance, assisting with literature and many great ideas on using the framework in the spatial reasoning domain. The author would also like to thank C. Veres[2] and A. Bagge[3] for agreeing to align my assignments in their courses with my master project.

[1] Moscow Institute of Physics and Technology, Department of Applied Mathematics
[2] University of Bergen, Department of Information Science and Media Studies
[3] University of Bergen, Department of Computer Science

1 Introduction

> *"The meaning of meaning and how to deal with meaning in formal and natural systems has been one of the great mysteries of intelligence - artificial or otherwise. It has been an issue from the earliest days of philosophy and logic, and it has become an engineering issue with the advent of computerized question answering systems, information retrieval systems, machine translation, speech understanding, intelligent agents, and other applications of natural language processing, knowledge representation, and artificial intelligence in general".*

(W. Woods [2:75]).

There are many different approaches to processing natural language. Many of them are statistical, based on the analysis of word combination frequencies in the texts. Some approaches are grammar- and syntax-aware. Though, virtually no approaches have substantial focus on the actual *meaning* of the text. This is unfortunate, because without focus on meaning there is little understanding, and good understanding of a request, question, or a problem is extremely helpful when it comes to finding the answer or a solution.

Natural language understanding (NLU) is a relatively small subtopic of natural language processing, dealing with machine reading comprehension [3]. Even though NLU technologies often work with *concepts* (rather than just words), most of the traditional ways of representing knowledge do not contain any information on the "internal structure"[1] of the concepts. For example, a traditional semantic network can tell whether two concepts are in a specific relation, but there is no (*quantitative*) information on the concept and relation definitions, or even information on to which degree these concepts are in the relation. This makes it nearly impossible to work on the level of the concept *meanings*, because computers need numbers, and if they are to work with the meaning, it must be represented quantitatively.

But is it really important to work on the concept meaning level? Here it may be appropriate to cite the words of T. Winograd [4:1]: *"We assume that a computer cannot deal reasonably*

[1] By internal structure here is meant something defining the essence of a concept or a relation, defining how it is different (or similar) to the other concepts and relations.

with language unless it can understand the subject it is discussing". Of course, one needs to be more realistic than AI researchers were in the early seventies, but some steps in the direction of modeling of the meaning were taken, e.g. starting from L. Zadeh [5].

Who hasn't experienced a voice interface getting one of the words totally wrong, so that the sentence doesn't make any sense? If the software knew that some of the words were "completely incompatible", it could have tried some other candidate words, hopefully with a better result.

It seems obvious that every achievement in the field of computer understanding of the meanings of the words and phrases can yield benefits when it comes to the possibilities of the software. It is likely to be true even if the "meaning" of the words to the computer would only be a fraction, a tiny projection of what the words actually mean to us humans.

This project does not attempt to solve the great mystery of meaning. Instead, it focuses on a tiny subset: modeling meaning of simple spatial directions expressed in natural language, and analyzing their comprehensibility computationally.

2 Research problem

2.1 Problem statement

The problem statement for this project could be formulated approximately like the following.

There is a need for a model allowing to quantitatively work with meaning (or its approximation) of simple words from spatial domain, and computationally interpret and analyze comprehensibility of short phrases made of these words. The model should be able to support development of a research prototype of an intelligent navigation assistant.

2.2 Use case: intelligent navigation assistant

Let's start by describing a use case for computational understanding of spatial directions expressed in natural language: an intelligent navigation assistant.

Within this project, intelligent navigation assistant is seen as software integrated into a car GPS or a mobile phone that communicates with the user using natural language interface and assists her with navigation-related tasks.

Let's imagine a situation: one is driving a car in the middle of a town, and her hands are tied.

> Driver: "Street to the left, can I go there?" GPS understands that you mean second to the left, because the first left will only get you into the water.
>
> Navigator/GPS: "No, that's one-way". GPS highlights the street with red so you make sure you're talking about the same street.
>
> Driver: "Okay, I'd like to check that cafe on the other side of the river, can we then get back?"
>
> Navigator/GPS: "No, we'll need to drive around the whole town".
>
> Driver: "Okay, let's park then, I thought there was a garage here somewhere on the right".
>
> Navigator/GPS: Highlights the garage and marks the directions.
>
> Driver: Only this expensive one?
>
> Navigator/GPS: Highlights a cheaper garage, but that is quite far away.
>
> Driver: "Can I maybe park somewhere on this hill?"

Navigator/GPS: "First street to the right, it's free until 9 tomorrow morning". The route and a free parking spot are displayed.

Even in 2015, when self-driving cars have been around some towns for a while, this dialog reads as science fiction. Obviously, this science fiction would be very useful in reality, because in this example the navigator software seems to *understand* the driver. Indeed, we can see that navigator's questions and answers are sharply focused on the real *goals* of the driver and the *meaning* of her questions and requests (rather than on matching words spoken by the user to a standard list of predefined commands). It is fairly easy to see that the system having similar capabilities would be of much higher value to the user, compared to the systems available today (like navigation software for cars and mobile phones). This is because the software would be able to answer her questions and help solve her problems with incomparably higher accuracy. It is obvious that understanding the questions greatly increases quality of the answers, and understanding the problems highly improves the ability to finding correct solutions.

Below is a very approximate list of research fields relevant for developing a system like this.

Table 2-1. Relevant research fields for the "intelligent navigation assistant" use case.

Functionality	Research areas
Communication with natural language	Natural language understanding (NLU), natural language processing (NLP), voice recognition, speech synthesis
Reasoning abilities for solving problems and answering questions	Automated reasoning, commonsense reasoning
Navigation skills	GPS, navigation, spatial reasoning
Learning and acquiring new concepts	Semantic technologies, machine learning

2.3 Project goals

This project focuses on a tiny subset of the mentioned research fields, namely attempting to model the meaning of phrases and individual words from the spatial domain, and interpreting and understanding phrases computationally. It is somewhat difficult to place this focus area

within one research field. It would be more or less correct to say that it mostly lies within natural language understanding, and touches up on commonsense reasoning and spatial reasoning.

The goals for the project are:

- Perform analysis of related work, with focus on different knowledge representations, and analyze their strengths and weaknesses when it comes to working with concepts on the level of their meanings.
- Design and develop natural language understanding framework, capable of interpreting and computationally analyzing the meaning of spatial directions expressed in natural language.
- Implement a research prototype of an intelligent navigation assistant with some functionality like in the one described above (2.2), illustrating capabilities of the framework.

3 Analysis of related work

When one speaks of understanding natural language, commonsense reasoning, and working on the level of concept meanings, one of the questions that arises immediately is the question of knowledge representation.

This chapter is a review of significant works in the fields of natural language understanding and commonsense reasoning with knowledge representation in focus.

3.1 Dawn of NLU and micro-worlds

One of the first natural language understanding programs, Weizenbaum's "Eliza" [6], was a relatively simple program configured by scripts of which the most famous was "doctor" simulating a Rogerian psychotherapist. "Doctor" was sometimes able to provide startlingly human-like interactions, even if the question was outside of the small knowledge base:

```
- My head hearts
- Why do you say your head hearts?
- My mother hates me
- Who else in your family hates you?
```

The program was implemented using relatively simple *pattern matching* techniques against *keywords*.

D. Bobrow's "STUDENT" [7] is another early NLU program that could solve word problems found in high school algebra books.

```
"If the number of customers Tom gets is twice the square of 20% of the number
of advertisements he runs, and the number of advertisements is 45, then what
is the number of customers Tom gets?"
```

D. Bobrow comments that "STUDENT" considers words as symbols, and "*makes do with as little knowledge about the meaning of the words as is compatible with the goal of finding the solution*". He explains that the semantic model in "STUDENT" is based on equality and five arithmetic functions [7:155].

T. Winograd's SHRDLU [4] is a system for understanding English, aiming farther than "Eliza" and "STUDENT". T. Winograd writes [4:1]: *"We assume that a computer cannot deal reasonably with language unless it can understand the subject it is discussing"*.

T. Winograd argues that "Eliza" and "Student" give impressive performances, because they carefully and severely limit the type of understanding they are trying to achieve. He means that the solutions they provide are too special-purpose [8:167].

SHRDLU has a reasoning engine allowing it to remember the goals, discuss and "carry out" its actions and plans. It can reply in English sentences, and ask for clarifications, when it cannot interpret the sentence with the knowledge already available.

```
- pick up a big red block.
- OK.
- grasp the pyramid.
- I DON'T UNDERSTAND WHICH PYRAMID YOU MEAN.
- the blue pyramid is mine.
- I UNDERSTAND.
- does a steeple
- SORRY, I DON'T KNOW THE WORD "STEEPLE".
- a "steeple" is a stack which contains two green cubes and a pyramid.
- I UNDERSTAND.
- are there any steeples now?
- NO.
- why did you do that?
- BECAUSE YOU ASKED ME TO.
```

The demonstration of SHRDLU at the time was very successful and led to a lot of subsequent AI research.

However, the research optimism was soon lost, and SHRDLU seems to never have made it out of its "micro-world". H. Dreyfus mentions in [9:12] that, for example, SHRDLU's concept of owning cannot be easily expanded beyond SHRDLU because it doesn't understand owning and cannot deal with meanings.

H. Simon argues that SHRDLU was not really meant to be expanded beyond its "block world" because of a "fixed representation". He means that the system solved "pick up a big red block" simply by finding the procedure "pick up" and invoking it with the parameter, corresponding to "big red block" [10:1062].

Indeed, the procedural knowledge is not easily expandable, partly because the procedures have to be written by humans. T. Winograd mentions himself in one of his later emails [11]: *"There are fundamental gulfs between the way that SHRDLU and its kin operate, and whatever it is that goes on in our brains. I don't think that current research has made much progress in crossing that gulf, and the relevant science may take decades or more to get to the point where the initial ambitions become realistic"*.

T. Winograd continues to work on knowledge representation in the later years. He departs from "Procedures as representation for data" [12], and together with D. Bobrow they design frame-inspired KRL, a "Knowledge Representation Language" [13] aiming to *"integrate procedural knowledge with a broad base of declarative forms"*. Much later, in an interview with A. Norberg [14], T. Winograd mentions that KRL was an overcomplicated attempt to come up with a universal language easy for reading and writing programs, with rich semantics of logic languages, and that could be "*processed like human memory, so you could have realistic AI programs"*. KRL itself didn't have much future.

In any case, SHRDLU was a major achievement at the time. This project considers SHRDLU as an inspiration and aims to implement a sort of computational understanding in the micro-world of spatial directions. Unlike SHRDLU, though, this project aims to use knowledge representation allowing to easily extend beyond this domain.

3.2 Commonsense knowledge and taxonomies

The language subset worked on in this project includes both general purpose words and simple spatial concepts describing directions, movements, and general purpose words. These naturally fall in the category of commonsense knowledge: *"collection of facts and information that an ordinary person is expected to know"* [15].

A wide group of approaches to describing commonsense knowledge in particular (and categorize semantic information in general) are based on some sort of taxonomical representations, like semantic networks, frames, and description logics.

3.2.1 ConceptNet and semantic networks

"The greek philosopher Porphyry (c. 234-305 A.D.), commenting on Aristotle's Categories, drew what might qualify as the first semantic network" [16:471]. According to [17], modern semantic networks were developed by R. Simmons and R. Quillian in the early sixties.

Let's turn our attention to ConceptNet, a semantic network, *"containing lots of things computers should know about the world, especially when understanding text written by people"* [18].

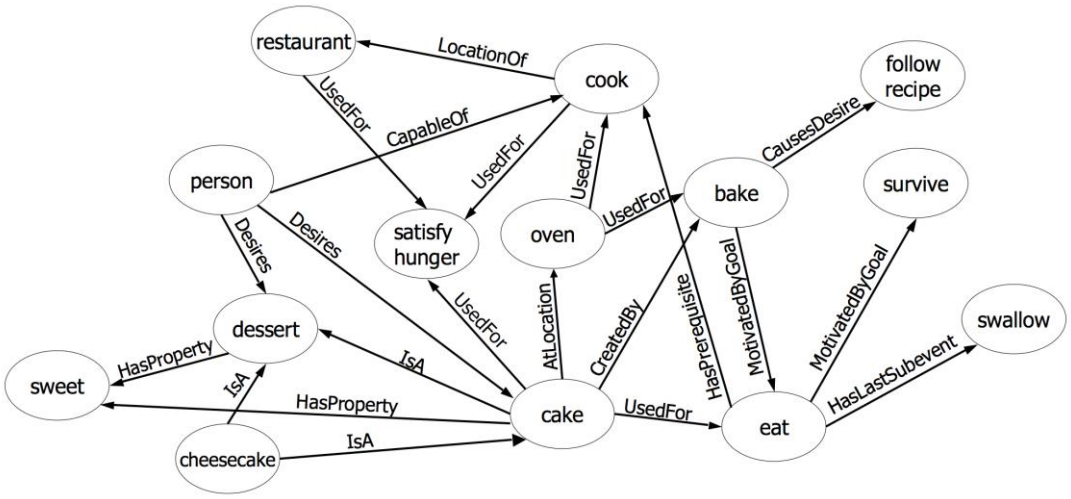

Fig. 3-1. An illustration of a small section of ConceptNet. R. Speer et al [21].

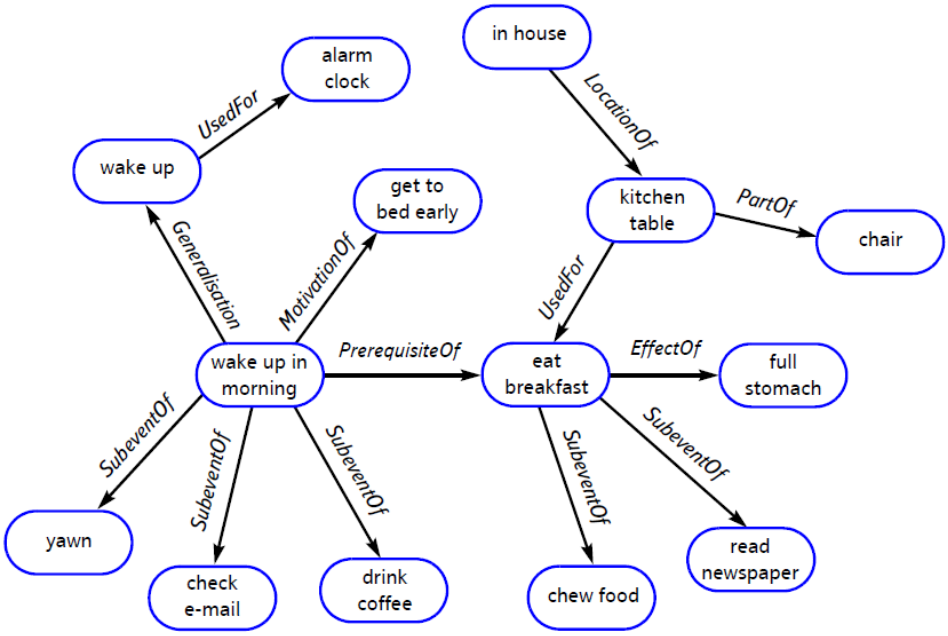

Fig. 3-2. Excerpt from ConceptNet. H. Liu and P. Singh [19]

H. Liu and P. Singh [20:10-12] discuss various graph reasoning methods that can be performed over ConceptNet data:

- Computing conceptual similarity:

```
"apple" ~ "red apple" (76%)
"buy food" ~ "purchase groceries" (69%)
"big dog" ~ "animal" (53%)
```

- Context finding:

```
"go to bed" yields "take off clothes," "go to sleep," "sleep," "lie down,"
"lay down," "close eye," "turn off light," "dream," "brush tooth," and
"snore."
```

- Inference chaining:

```
"buy food" -> "have food" -> "eat food" -> "feel full" -> "feel sleepy" ->
"fall asleep."
```

- Conceptual analogy (structural):

```
"couch" -> "sofa," "chair," "bed," "seat"
```

There are other projects implementing reasoning engines on top of ConceptNet. R. Speer et al. [21] describe AnalogySpace. Pursuing the goal of drawing conclusions from

analogies, it utilizes singular value decomposition technique to reduce the dimensionality of knowledge in ConceptNet. According to the authors, AnalogySpace allows to generalize ConceptNet knowledge and make it more available to applications, particularly in the area of natural language.

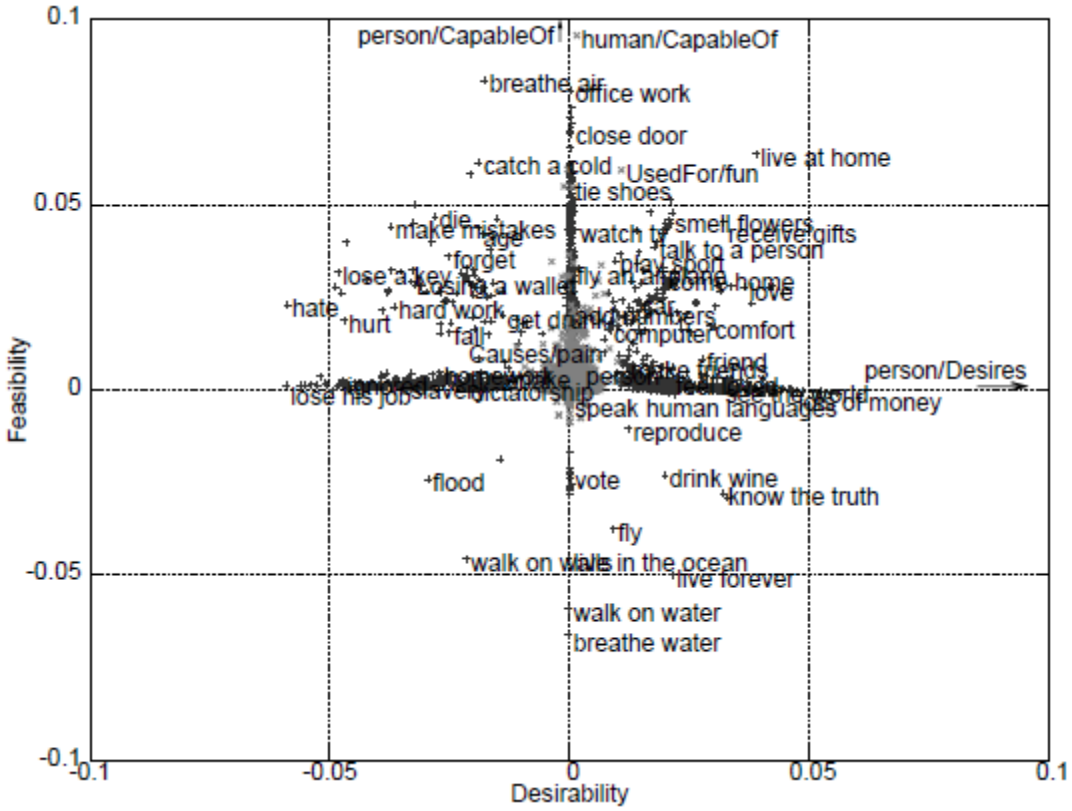

Fig. 3-3. AnalogySpace projected onto its first two components, with some concepts and features labeled. Robert Speer et al [21].

3.2.2 Some challenging and meaning-related problems

Let's take a look at some of the challenges with the traditional semantic network representation when it comes to working with meaning and reasoning with natural language concepts. Fig. 3-1 and Fig. 3-2 are used as illustrations:

1. Issues with predicate quantifiers. Are all cakes sweet? Or only some cakes?
2. Equivalence of the meanings in a given context. Does "satisfy hunger" mean the same as "eat" in a given context?
3. Fuzziness degrees. To what extent are cakes sweet? Is one's stomach going to be equally full after "eat breakfast" and "eat dinner"?

4. Uncertainty handling. What's the possibility that one would yawn after waking up in the morning?
5. Relation between basic and composed concepts. How does concept "early" relate to concept "get to bed early"?
6. Composition of concepts into phrases. Does "sweet person" make sense in a given context, or not? Can an "oven" "follow a recipe" to "bake" something or not?
7. Concept definitions. How is "early" defined? How is "morning" defined? Can a computer easily decide whether it is "early" or "morning" at a given time?
8. Exact interpretation of the relations. Many of the relations in semantic networks computers cannot really handle. People often go to bed early so they easier wake up in the morning (relation "MotivationOf"). But what can a computer infer from this? Given one goes to bed late, does it mean that one doesn't want (need) to wake up in the morning? That one won't wake up in the morning? That one won't eat breakfast?

It is quite difficult to answer questions like this based on the information available in the network. Many things are too vague or ambiguous for a computer, many are lacking in this representation, especially from a *quantitative* standpoint.

3.2.3 From semantic networks to description logics

Some of these challenges have been researched for a long time. As long as 40 years ago, W. Woods writes about some of the issues with semantic network representation in his famous paper "What's in a Link" [22]. In particular, he discusses issues with link interpretation, issues with predicate quantifiers, and mentions problems with handling probabilistic information and uncertainty degrees. He is challenging the basic of semantic network methodology for attempting to represent in a single mechanism both the ability to model the associative connections, and the ability to store factual knowledge (by assemblies of pointers to other facts). He writes [22:15]: *"One should keep in mind that the assumption that such a representation is possible is merely an item of faith, an unproven hypothesis used as the basis of the methodology. It is entirely conceivable that no such single representation is possible".*

In his "Meaning and Links" [2] W. Woods explains: *In fact, there was generally nothing really semantic in "semantic networks", and I felt the term itself was a misnomer. It was in response to these observations that I wrote the paper "What's in a Link."* He discusses, in particular,

semantic tags and link tags, extensions that could help to solve some of the issues raised in "What's in a Link".

In the late seventies - early eighties the issues raised on imprecisions and ambiguities in semantic networks trigger a lot of research around knowledge representation frameworks. KRL [13] was mentioned earlier. R. Brachman analyzes taxonomic links in his paper "What IS-A is and Isn't" [23], and argues that *"there are almost as many meanings for the IS-A link as there are knowledge-representation systems".* He works on KL-ONE framework [24], "*designed to overcome semantic indistinctness in semantic network representations and to explicitly represent conceptual information as a structured inheritance network"* [25]. This research triggered a whole family of new systems that came to be "KL-ONE family", and eventually gave rise to a new field now referred to as description logics [2:82].

3.2.4 Logic-based commonsense knowledge

Description logics are a family of formal knowledge representation languages, designed to overcome lack of formal logic-based semantics of frames and semantic networks [26]. Description logics play an important role as a theoretical foundation for ontology languages and the Semantic Web. Various formal reasoners exist for different description logic languages [27].

Using logic-based representations for representing and reasoning with commonsense knowledge was first described in J. McCarthy's classical paper "Programs with Common Sense" [28]. But how well-suited are logic-based approaches for this?

Cyc is a project assembling a comprehensive ontology and knowledge base of commonsense knowledge formulated in the language CycL, that is based on predicate calculus" [29]. It contains definitions of 239,000 concepts and a whole 2,093,000 facts [30]. The project has been described as *"one of the most controversial endeavors of the artificial intelligence history"* [31:275].

H. Liu and P. Singh [20] argue for why representing commonsense knowledge using formal logic may be not such a good idea:

- Precise definition of terms and interrelationships require a lot of additional *"logical scaffolding",* a task of *"daunting complexity"* [20:5]

- Commonsense knowledge is defeasible and very context sensitive. "*People have no problem believing that "birds can fly," even though they know that "penguins are birds who cannot fly", and that "birds with broken wings cannot fly."* [20:6]

  The way commonsense knowledge is categorized is largely defined by humans, and they categorize concepts in a very special ways, drawing fuzzy boundaries and using resemblance for grouping [20:6]
- While logical reasoning is deductive, commonsense (human-induced) reasoning is largely *"inductive, abductive, and empirical, where (over-)generalizations from known experiences plays a prominent role"* [20:6]
- Authoring knowledge for logic-based systems like Cyc is manual work that needs to be done by logicians. [20:7]

The last argument by H. Liu and P. Singh is of ultimate importance, and can be expanded. If the system is to *learn* (acquire new knowledge) in any way, the formal logic-based representation seems problematic: how would the system itself generate strict logical rules, even theoretically? This problem is very similar to the problem with procedural knowledge representation used by SHRDLU: the system works well as long as it has the procedures written by humans, but how can we teach the system to write new procedures for itself?

Let's revise the problems mentioned in 3.2.2 and see whether a description logic-based representation would help answering them. Well, it appears that we would have managed the question 1 "Are all cakes sweet? Or only some cakes?" Description logics have predicate quantifiers to deal with this. We could have also used circumscription to make the reasoner treat all cakes as sweet, unless specifically stated otherwise. When it comes to the rest of the questions, description logic representation would have probably not helped us a lot.

3.2.5 Other interesting studies

J. Pustejovsky in his "Generative lexicon" [32] challenges the standard approach to the lexical word meaning, where each lexeme is associated with a number of word senses. He argues, for example, that the concept "*fast motorway*" cannot be adequately accounted on the basis of the 3 meanings that are given two a word "*fast*" as in "fast typist", "fast car", and "fast waltz". He introduces *qualia* structure of words including:

- The relation between an object and its constituent parts (*Constitutive* role)

- That which distinguishes it within a larger domain (*Formal* role)
- Its purpose and function (*Telic* role)
- Factors involved in its origin or "bringing it about" (*Agentive* role).

For example, *Telic* role of school would be “educational institution”, while its *Formal* role would be “building”. Generative Lexicon is discussed in the light of issues like “IS-A” and concept inheritance in [33].

It should also be noted that there is a certain amount of study on adding capabilities like fuzziness and uncertainty handling to semantic network and description logic-based representations [34, 35, 36, 37, 38], can be mentioned here. However, each suggested extension attempts to solve one little incapability of the generally accepted representation, leaving out many other open questions.

3.2.6 Summary

Let’s turn our attention to the problems mentioned in 3.2.2. So far we have briefly touched upon problems 1-4. Not that there was a conventional solution to all of those, but some approaches were discussed. When it comes to problems 5-8, we haven’t even come close to approaching them, and this is not accidental. This has to do with the fact that problems 5-8 are concerned with concept and relation meanings, or, in other words, concept and relation *internal structure* and concept composition, while none of the representations we have looked at so far possess information like this. This will be further discussed in 3.4.

3.3 Latest advances of NLU in spatial reasoning

This section is a sort of a digress listing some of the most interesting cutting edge works on applications of NLU to spatial reasoning problems.

T. Kollar et al. [39] describe a very interesting approach for directing robots by natural language. Their system first parses input phrases into a sequence of SDCs (*spatial description clauses*), containing a *figure* (the subject of the sentence), a *verb* (an action to take), a *landmark* (an object in the environment), and a *spatial relation* (a geometric relation between the landmark and the figure). Then, given the information about the environmental geometry and detected visible objects, it maps these SDCs to places, objects or paths (*groundings*), using

different techniques, and constructs probabilistic graphical model (*grounding graph)* to infer the most probable path through the environment.

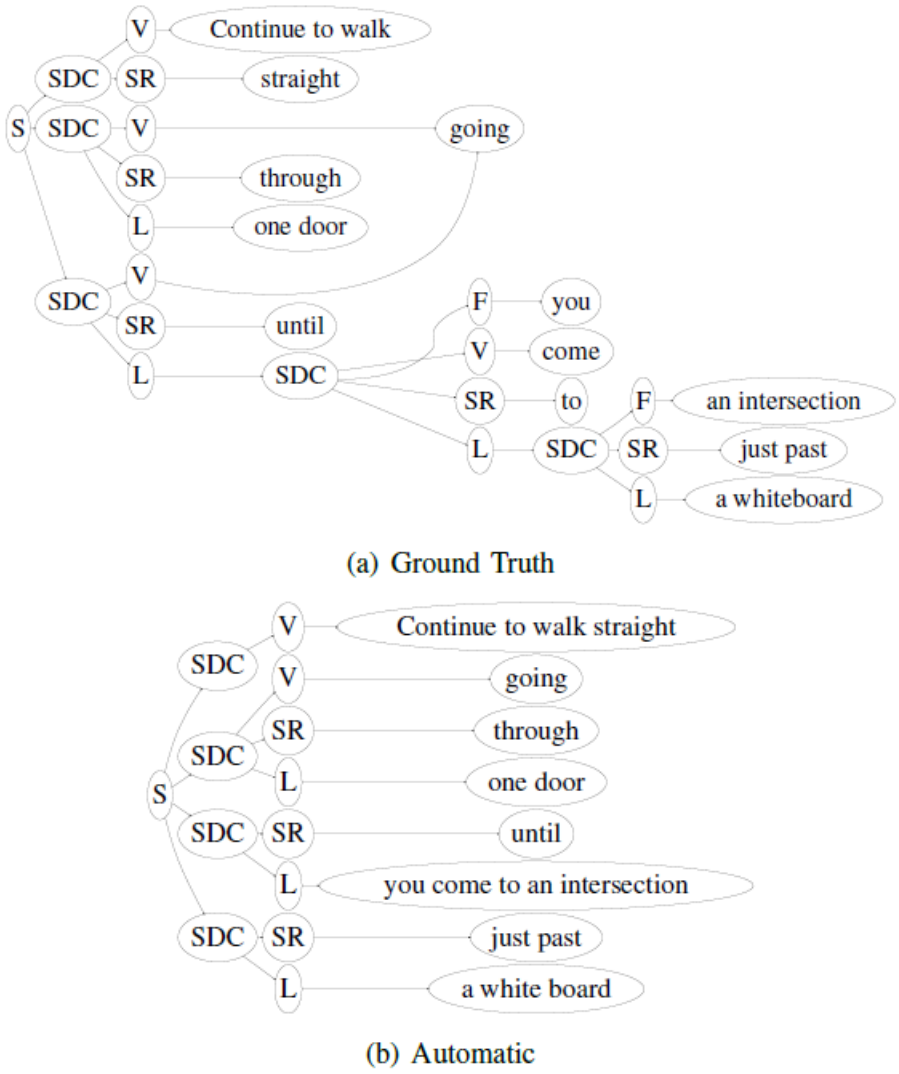

Fig. 3-4. Example of spatial description clauses. a) ground truth b) Automatically generated from natural language input T. Kollar et al [39]

It is interesting how the semantics of spatial prepositions like “to”, “through”, “past” is modeled. They use features that are functions of the geometry of the path and landmark. Using these features, the prepositions are learned using Naïve Bayes classifier. The dataset used contains hand-drawn positive and negative examples:

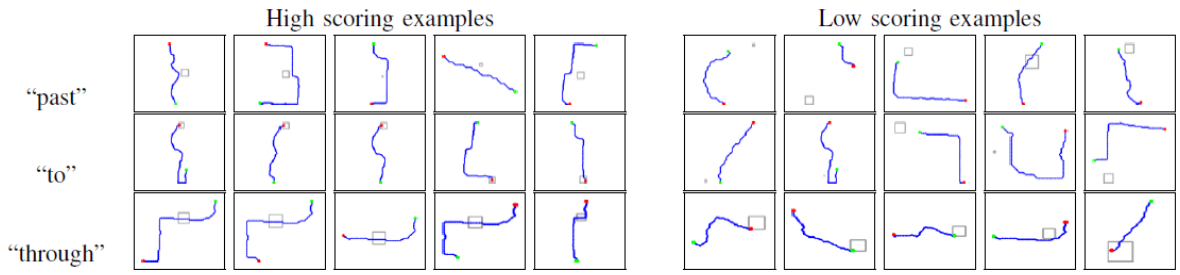

Fig. 3-5. Positive and negative examples to learn the spatial prepositions. T. Kollar et al. [39]

S. Tellex and D. Roy [40] discuss another application of NLU to spatial reasoning: video search. This paper also discusses the features used to learning spatial prepositions in more details.

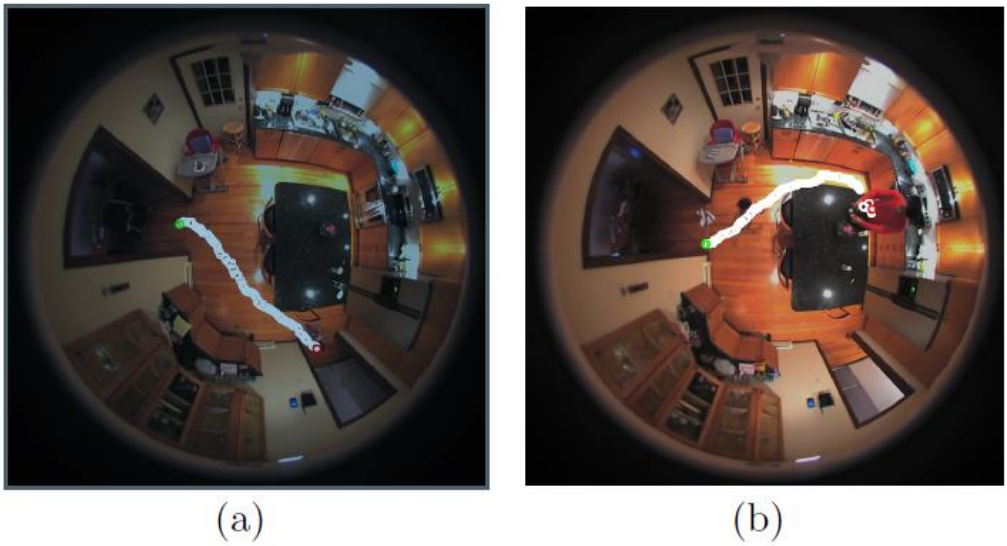

Fig. 3-6. Frames from two clips returned for the query "across the kitchen." S. Tellex and D. Roy [40]

S. Hemachandra et al [41] present an approach allowing the robots to follow natural language directions without any previous knowledge of the environment.

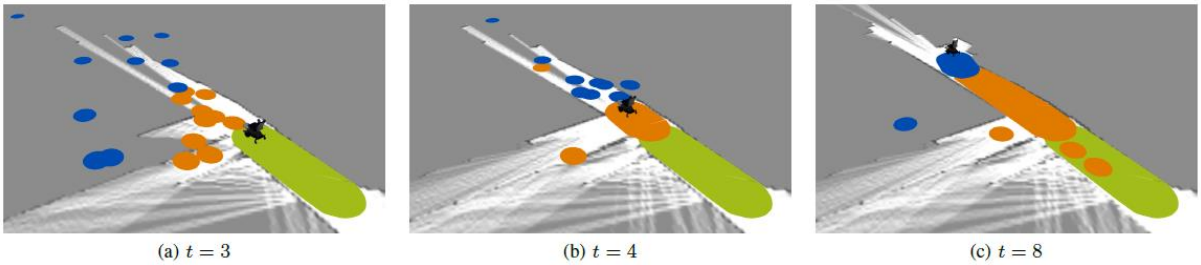

Fig. 3-7. Visualization of the evolution of the semantic map over time as the robot follows the command "go to the kitchen that is down the hallway." Small circles and large filled-in areas denote sampled and visited regions, respectively, each colored according to its type (lab: green, hallway: yellow, kitchen: blue). The robot (a) first samples possible locations of the kitchen and moves towards them, (b) then observes the hallway and refines its estimate using the "down" relation provided by the user. Finally, the robot (c) reaches the actual kitchen and declares it has finished following the direction. S. Hemachandra et al. [41]

The researchers use a hierarchical framework based on a DCG (*distributed correspondence graph)* to convert natural language information to a probabilistic graphical model expressing the correspondence between linguistic elements from the command and their corresponding groundings. This learned model is later used by 1) the semantic map inference algorithm hypothesizing the existence and location of coherent regions, and 2) belief space planner reasoning directly over the map and behavior distributions to solve for a policy using *imitation learning*.

The works mentioned in this section show some of the latest advances of NLU applications to spatial reasoning problems. To achieve these impressive results, a large number of advanced techniques very different from formal logic are put together. State-of-the-art machine learning algorithms, probabilistic graphical models and particle filters are heavily used. Everything handles about learning the probabilities, because this works great when there is:

a) Little information (about robot environment)

b) Little knowledge (about what user commands *actually* mean)

Our project, focusing on the modeling of the word and phrase meaning, takes a very different approach (understanding in full that, in foreseeable future this will yield much less impressive results when it comes to what the computer can do, but hoping that this will change at some point when the computer will be able to know more).

3.4 Fuzzy semantics

This chapter is a review of the theories that the rest of the project will be building upon. To start, it is necessary to go back to the beginning of the seventies.

3.4.1 Quantitative fuzzy semantics

In 1971, in his paper on Quantitative Fuzzy Semantics [5], L. Zadeh asks: *"Can the fuzziness of meaning be treated quantitatively, at least in principle?"*

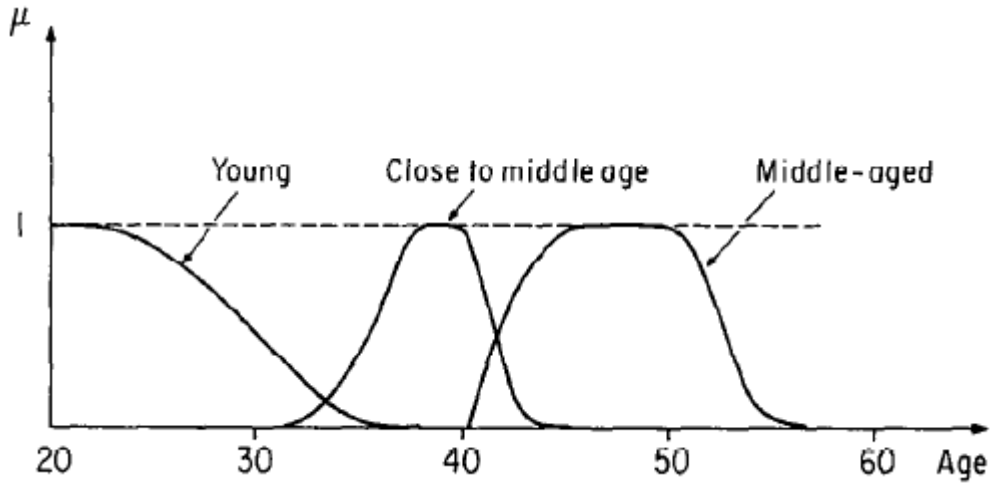

Fig. 3-8. Characterization of "young", "close to middle-age" and "middle-aged" as fuzzy sets. L. Zadeh [5]

Let's consider Fig. 3-8. For the first time in our research review, there is a graph that visualizes a certain quantitative relation between concepts "Young", "Close to middle age", "Middle-aged", and "Age".

In 1972, L. Zadeh [42] suggests possibility of using fuzzy sets for modeling linguistic hedges (e.g. "very", "more or less", "much", "essentially", "slightly"). He suggests that hedges can be viewed as *operators* that act on the fuzzy set representing the *meaning* of its *operand* (e.g. operator "very" acting on a meaning of "tall man"). He also introduces several operations for manipulating the fuzzy sets (complementation, intersection, normalization, concentration, dilation, fuzzification etc.).

G. Lakoff provides a useful illustration on how the fuzzy hedges work [43]:

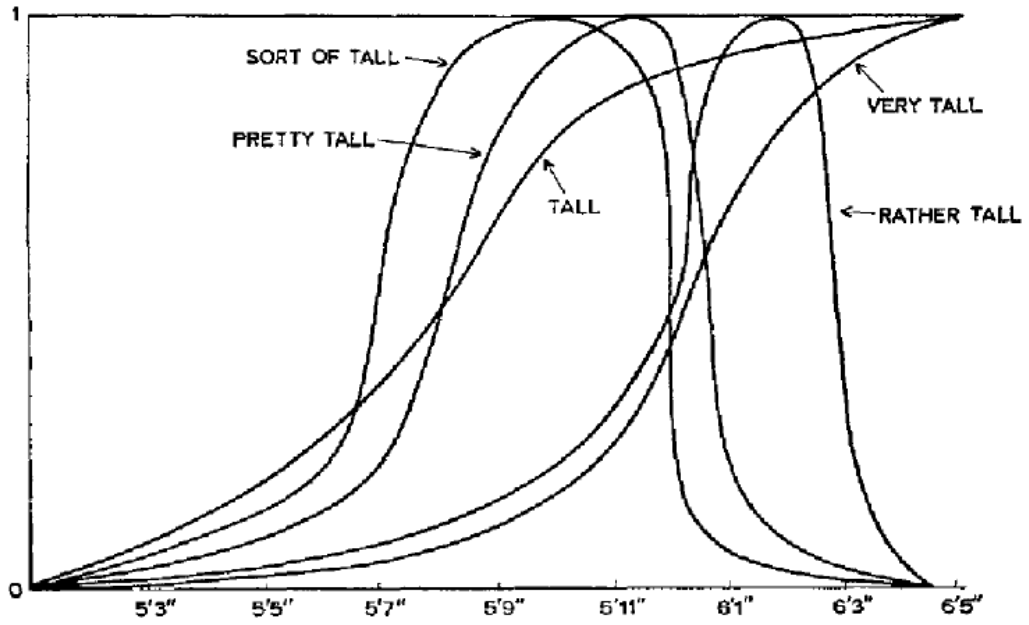

Fig. 3-9. Some examples of modifier valuations using Zadeh functions. G. Lakoff [43]

In 1975, L. Zadeh introduces a concept of linguistic variables [44]. Values these variables take are words or sentences in a natural or artificial language, e.g. “age” is a linguistic variable if its values are “young”, “not young”, “not very old and not very young” etc.

In 1978, L. Zadeh presents a meaning representation language “PRUF” [45]. In [45, 46] he suggests a computational approach to fuzzy quantifiers (like “several”, “few”, “many”, “approximately five” etc.).

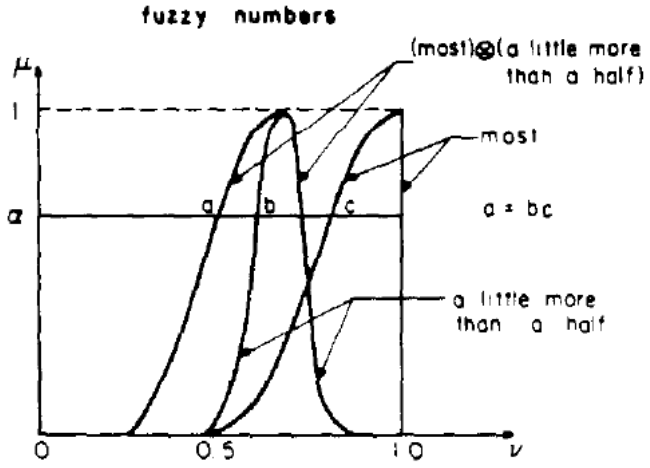

Fig. 3-10. The intersection/product syllogism with fuzzy quantifiers. L. Zadeh [46]

3.4.2 Test-score semantics

In 1983, L. Zadeh is expanding the idea of fuzzy word meaning onto propositions, introducing *test-score semantics* [47], and, later in [48], knowledge representation based on fuzzy logic. He argues that fuzzy logic representation is more appropriate for commonsense reasoning, compared to more traditional representations (semantic networks, frames, predicate calculus etc.), because of its inherent ability to handle uncertainty and imprecision.

In this model, a proposition is viewed as a system of implicitly defined elastic (fuzzy) constraints, whose domain is the collection of fuzzy relations in so-called *explanatory database*. For example, proposition "Joan is young and attractive" has two variables: *age* (implicit in the proposition) and *attractiveness,* that need to be constrained. The explanatory database would contain these relations (where μ is membership degree):

```
POPULATION [Name; Age; µAttractive]

YOUNG [Age; µ]
```

According to L. Zadeh [48], to represent the meaning of the proposition, it is necessary to construct a *test procedure* that tests, scores, and aggregates the elastic constraints, yielding an overall test score. This test score serves as a measure of compatibility between the proposition and the explanatory database, and the *meaning* of the proposition is represented by the test procedure itself.

L. Zadeh mentions in [47] that there is no way to automatically construct the test procedure and explanatory database, that it would require *"substantially better understanding of natural languages and knowledge representation than we have at this juncture".*

3.4.3 Computing With Words

In 1996, L. Zadeh mentions a new methodology "Computing With Words" (CWW) [49]. The idea is that words are used instead of numbers for computing and reasoning. For example, one problem for CWW could be this one:

```
I have to be at the airport about an hour before departure. Usually it takes
about forty five minutes to get to the airport from my home. I would like to
be pretty sure that I arrive at the airport in time. At what time should I
leave my home?
```

CWW methodology is based on a *Generalized Constraint Language* (GCL) [50, 51], and is rooted in [47]. CWW is further discussed in [50, 51, 52, 53, 54]. CWW process is performed in 2 steps: 1) precisiation of natural language and 2) computing with precisiated language.

In CWW, proposition is viewed as a *generalized constraint* of form *X isr R*, where *X* is the constrained variable, *r* is constraint type, and *R* is the constraining relation. The meaning of the proposition is then defined (precisiated) by specifying *X*, *R* and *r* [54:66-67]. For example, proposition "Robert is tall" can be viewed as a *possibilistic* constraint with variable *X* being Robert's height, and relation *R* being "tall".

Generalized constraints are a powerful way of conveying fuzzy information like human perceptions. L. Zadeh defines 3 *primary* constraints [53, 54]:

- Possibilistic: *X is R. Poss(X=u) =* $\mu_R(u)$, where R is a fuzzy set in a space *U={u}*, and μ_R is the membership function of *R*. *R* is the possibility distribution of *X*
- Probabilistic: *X isp R. X* is a random variable, and R is its probability distribution
- Veristic: *X isv R. Ver(X=u) =* $\mu_R(u)$, where *Ver(X=u)* is the truth-value of *(X=u)*

Fig. 3-11 visualizes different types of precisiations of "approximately a" [51].

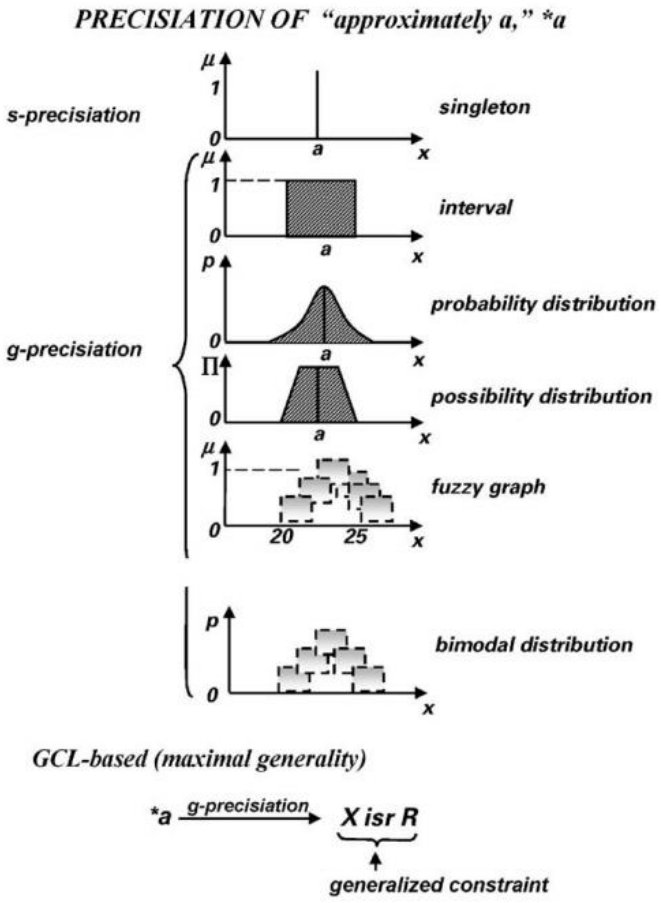

*Fig. 3-11. Hierarchy of precisiations of *a (approximately a). L. Zadeh [51]*

CWW is a very interesting methodology focusing on enabling word-based computations. However, it does leave many open questions, for example:

- The precisiation step is completely dependent on humans for choosing *X*, *R*, and constructing the explanatory database [54:185].
- There is no attempt made to somehow formalize, or automatically relate implicit variables (like *"age"*) to words in natural language (like *"young"*).
- Proposition meaning is not defined in any quantifiable way (which makes it impossible for a system to analyze whether it is comprehensible).
- Nothing is said about how to handle the knowledge that is evolving, changing, expanding over time in this representation.

4 Method and framework design

This master project is based on joint research [1] focusing on modeling concept meanings and computational comprehension of phrases. The approach relies on some of the techniques described by L. Zadeh in [5, 42, 46], but is very different when it comes to overall knowledge representation, modeling phrase meaning, etc. In particular, the approach is an attempt to challenge the issues mentioned in 3.2.2, and at the end of 3.4.3.

This chapter describes central ideas of the approach and the parts relevant for this master project.

4.1 Approach overview

The approach is based on fuzzy logic, as a good fit for working with different levels of truthness and concepts with unclear boundaries, phenomena commonly occurring in knowledge coming from natural language, and commonsense knowledge in particular [48].

All knowledge is encoded in *fuzzy properties* (with values ranging from zero to one), each of them encoding an independent piece of information.

Structuring knowledge and dealing with its context sensitivity is modeled with *contexts*. Context is defined as a coordinate system with one axis encoding one independent property. If the context has *N* properties, then the knowledge in this context is described as a *fuzzy region* in an *N*-dimensional unit hypercube.

Phrases, words and other natural language constructs are modeled as region transforms, or *meaning-operators*. For example, interpretation of a phrase is a transformation of the *source region* (the region before interpretation) by the corresponding phrase operator, yielding the *resulting region* (the region after interpretation).

Computational interpreting is viewed as choosing the interpretation that makes the most sense, using different heuristics.

Meaning of an operator can be assessed during its composition. Overall phrase comprehension can be evaluated when the phrase operator is being applied. Then both source and resulting regions are considered along with other factors (e.g. phrase mood).

Simplified schematics of the phrase interpretation is given on Fig. 4-1.

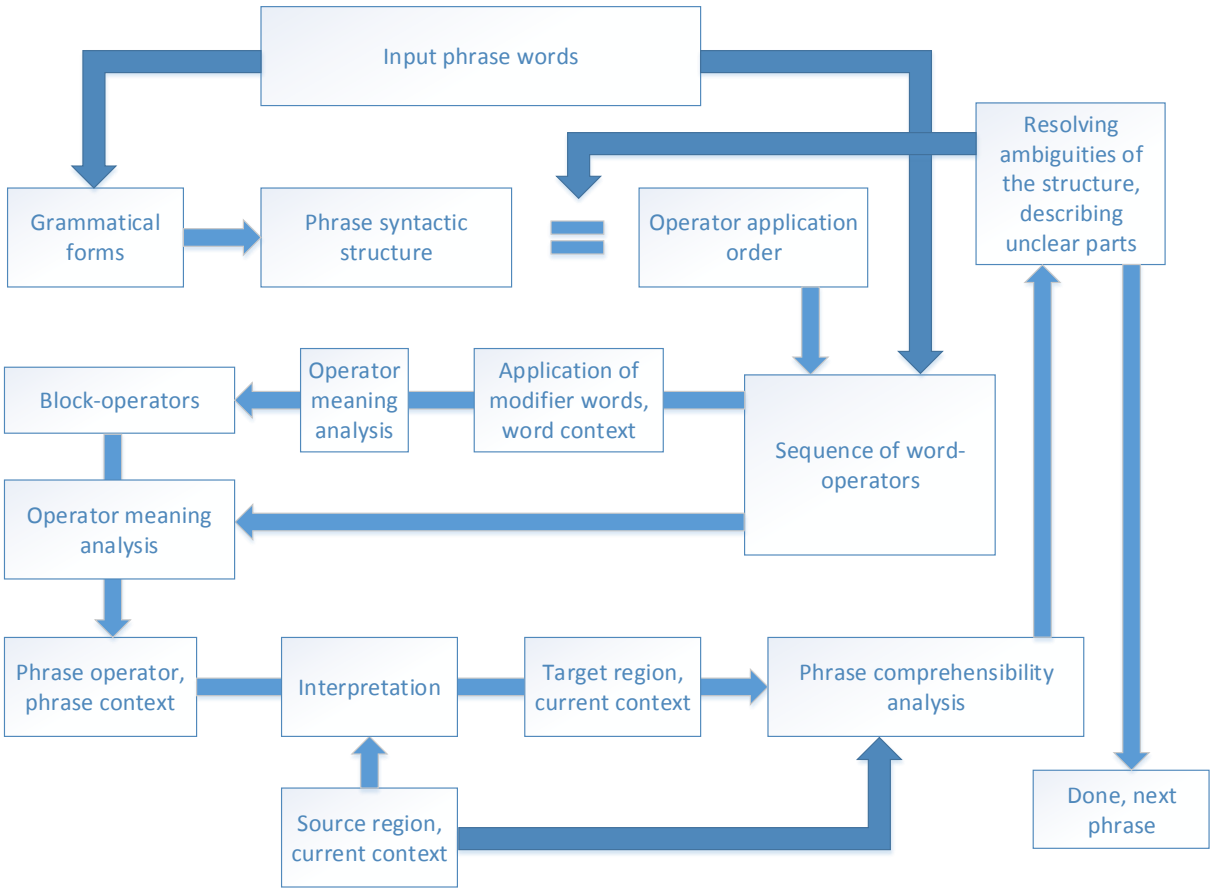

Fig. 4-1. Simplified schematics of phrase interpretation

Please note: this is a model drawing. Only some parts of this model are implemented in this master project. This is discussed in more detail in section 4.4.3.

4.2 Representation basics

4.2.1 Properties, context and regions

Let's assume that the state, or knowledge of our system can be completely described with a finite number of independent real parameters (properties). Values of each property range from zero to one, so that one corresponds to maximal presence of the property, and zero

corresponds to its complete absence. For example, if we would like to model car speed, we could use property "quickness"[1]. "Zero" would mean "not fast at all" and "one" would mean "as fast as it gets").

We are going to call some of the properties *basic properties*. Many of the basic properties will include values that can be directly "perceived" by the software. For example, for a robot or a car with a built-in rangefinder, property "relative distance" can be perceived directly. The same is true for the property "relative time", as long as the system has a built-in clock.

The rest of the properties we are going to call *derived*. They can be defined via other concepts, e.g. using the model described later in this section.

Context is a coordinate system consisting of *axes* that represent values of currently relevant properties, one axis per property. For example, when talking about movement speed, two of the relevant properties would be "relative distance" and "relative time.

In the context's coordinate system we can define a fuzzy shape: a region. Each point of the region is assigned a value from zero to one, describing this point's degree of membership.

Regions can be used to express the meaning of different concepts[2]. For example, using axes *t* and *s* ("relative time" and "relative distance") we can express regions, describing concepts "fast" and "slow" (Fig. 4-2, membership degree is depicted with color intensity).

[1] For simplicity of the examples, direct use of "speed" is avoided, as it is a more complex concept.

[2] It is interesting to compare the way regions are organized with L. Zadeh's CWW [53, 54]. In CWW figuring out that "young" has to do with the implicit variable "age" would be a part of manual precisiation step. In the proposed approach concept definition (e.g. "fast") already specifies how exactly it is "constraining" the variable(s) (e.g. "distance", "time" (Fig. 4-2), or "quickness" (Fig. 4-3).

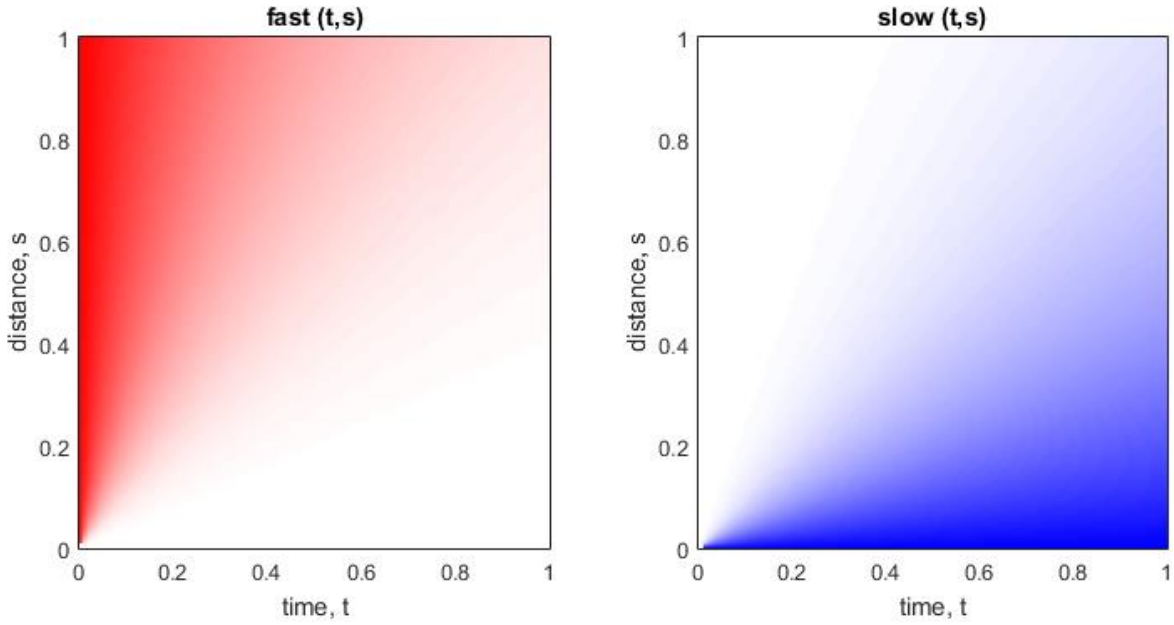

Fig. 4-2. Concepts "fast" and "slow"

Based on existing regions, new (derived) properties can be introduced. For example, based on the region corresponding to the concept "fast" (Fig. 4-2), we can introduce a new property "quickness". It is natural to assume the values of "quickness" (*x*-axis on Fig. 4-3) be equal to the degrees of membership of its corresponding region's points (color intensity on Fig. 4-2, left).

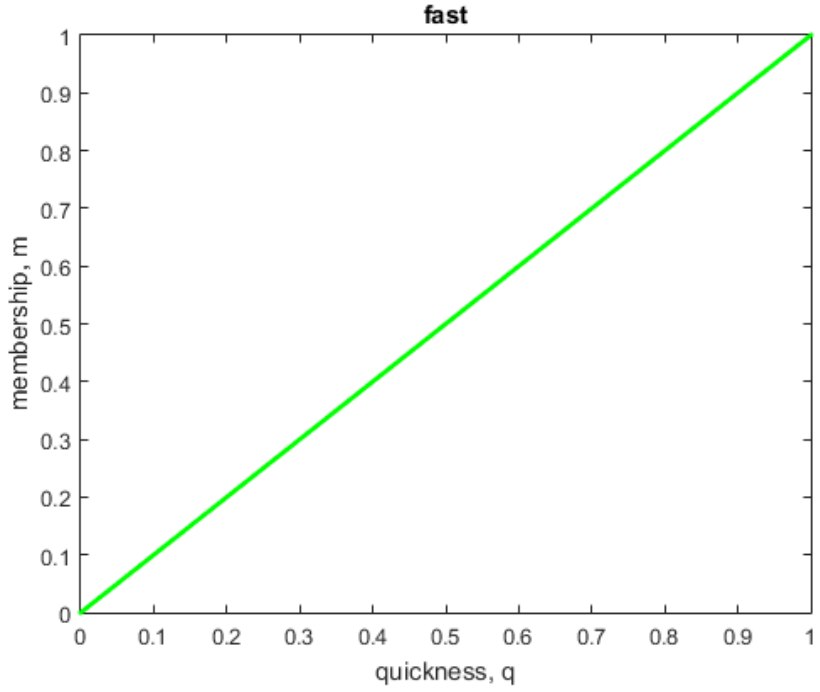

Fig. 4-3. Concept "fast" in context with one property "quickness"

As long as this relation holds, we are going to call such region *reference* region of a property.

Let's now introduce a new concept "moderatelyPaced" containing only one axis "quickness" (q), and define a region in this context, described by the function *moderatelyPaced(q)* (Fig. 4-4).

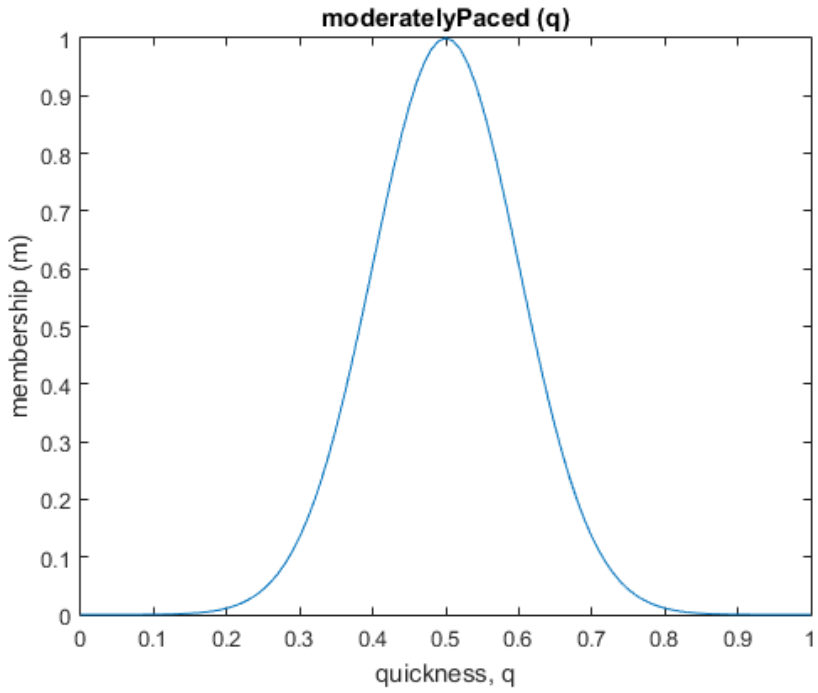

Fig. 4-4. Concept "moderately paced"

Please note that as long as we only have one property in this context, we are using *Y*-axis for the degree of membership (instead of using color intensity, as in the previous example).

Let's now see how this region looks in coordinates *s, t.* We previously assumed values of property "quickness" (*x*-axis on Fig. 4-3) to be drawn from the degrees of membership of the concept "fast" (color intensity on Fig. 4-2, left). Because of this, $moderatelyPaced(fast(s,t))$ will be function composition of $fast(s,t)$ and $moderatelyPaced(q)$, yielding a function like "$moderatelyPaced(s,t)$". This function will transform membership degree of each point of the concept $fast(s,t)$ in accordance with the rule given by $moderatelyPaced(q)$, and this corresponds to the concept "moderately paced" in the context with axes s,t (Fig. 4-5).

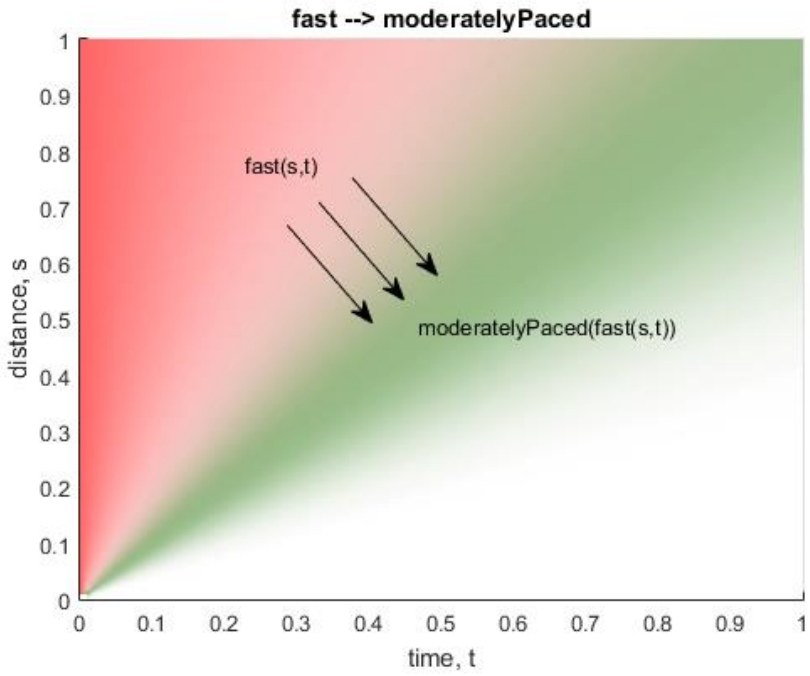

Fig. 4-5. Mapping "moderatelyPaced" back to reference context.

This kind of mapping is general operation, allowing us to "expand" any axis via its reference axes, thus, mapping different pieces of information into the same (reference) context for processing.

4.2.2 Operators

Region transforms, or *operators*, play key role in the proposed approach. This section is an example of modeling two function words as operators and using them to create new concepts.

We can define the operator corresponding to the word "not" like this: $not(x) = 1 - x$, where *x* is the value of a property (Fig. 4-6). This is similar to L. Zadeh's "complementation" [42:10].

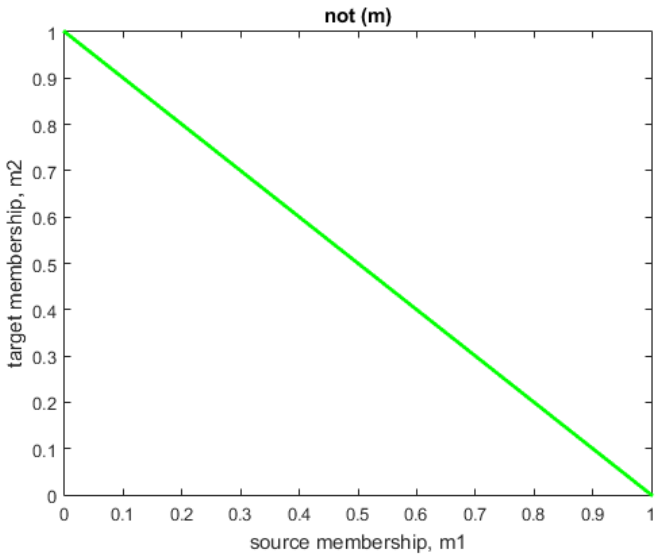

Fig. 4-6. Operator "not"

Using operator "not", we can define a new concept "slow" as *not(fast)* (Fig. 4-7)

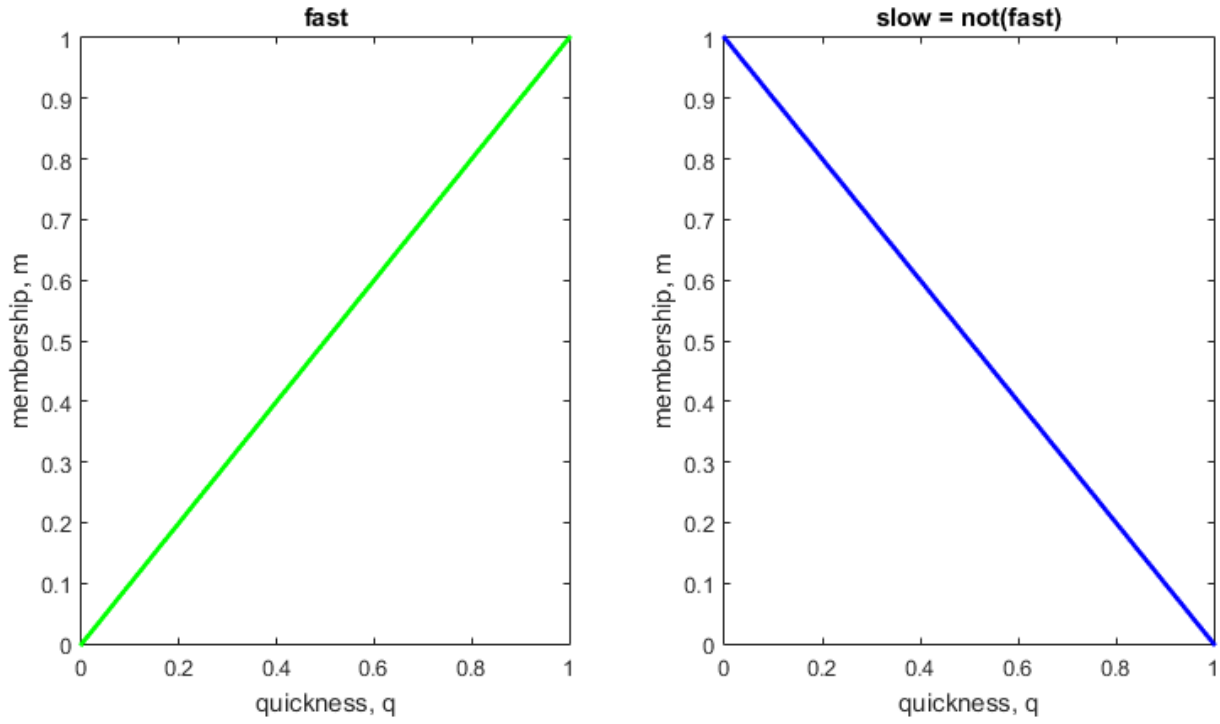

Fig. 4-7. slow = not(fast)

Mapping this back to our reference context, we get: $not(fast(s,t))$ (Fig. 4-8)

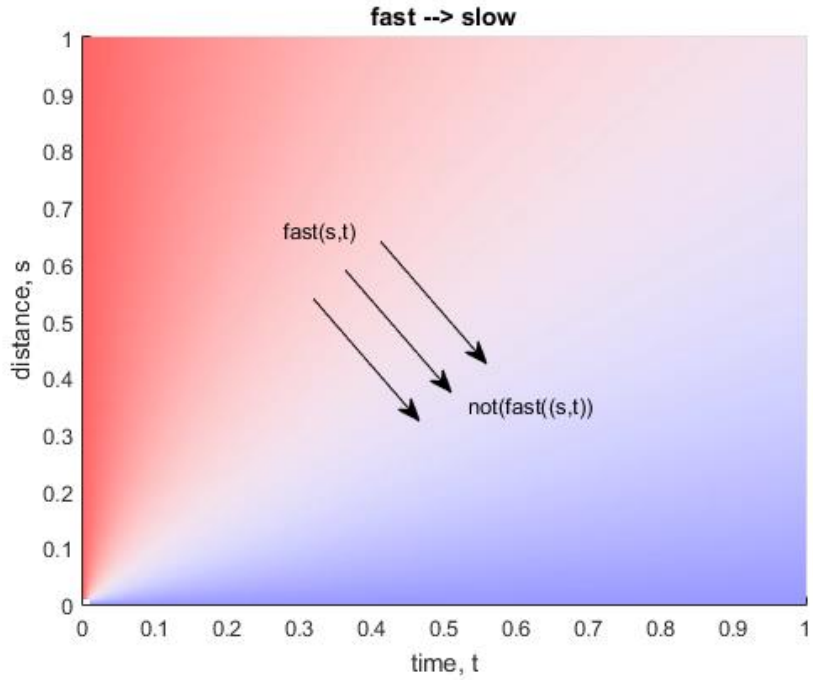

Fig. 4-8. Mapping slow = not(fast) back to reference context

We can define "Very" approximately like this: $very(x) = x^2$ (Fig. 4-9). This is similar to L. Zadeh's "very" [42:23].

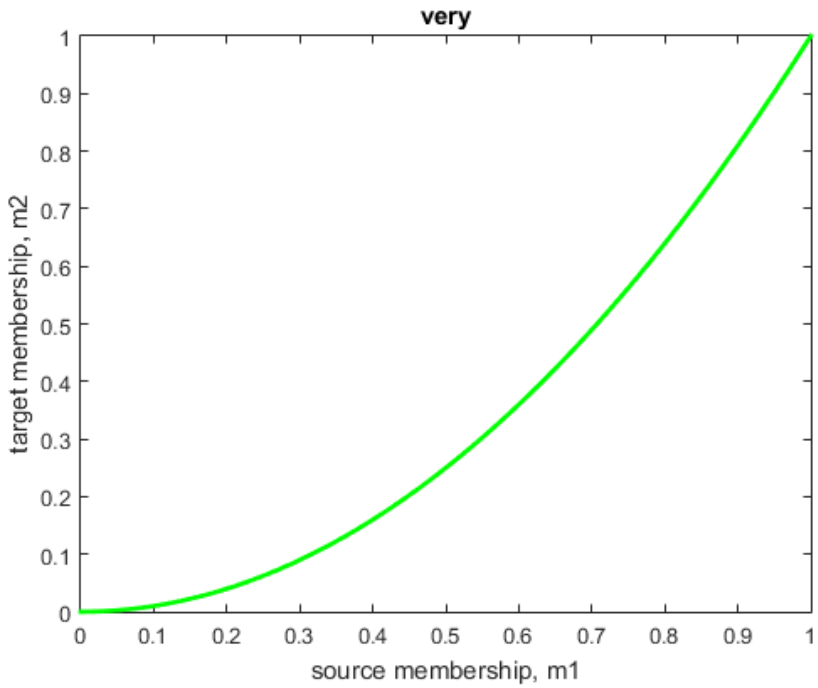

Fig. 4-9. Operator "very"

Applied to region "fast", we'll get region corresponding to "very fast".

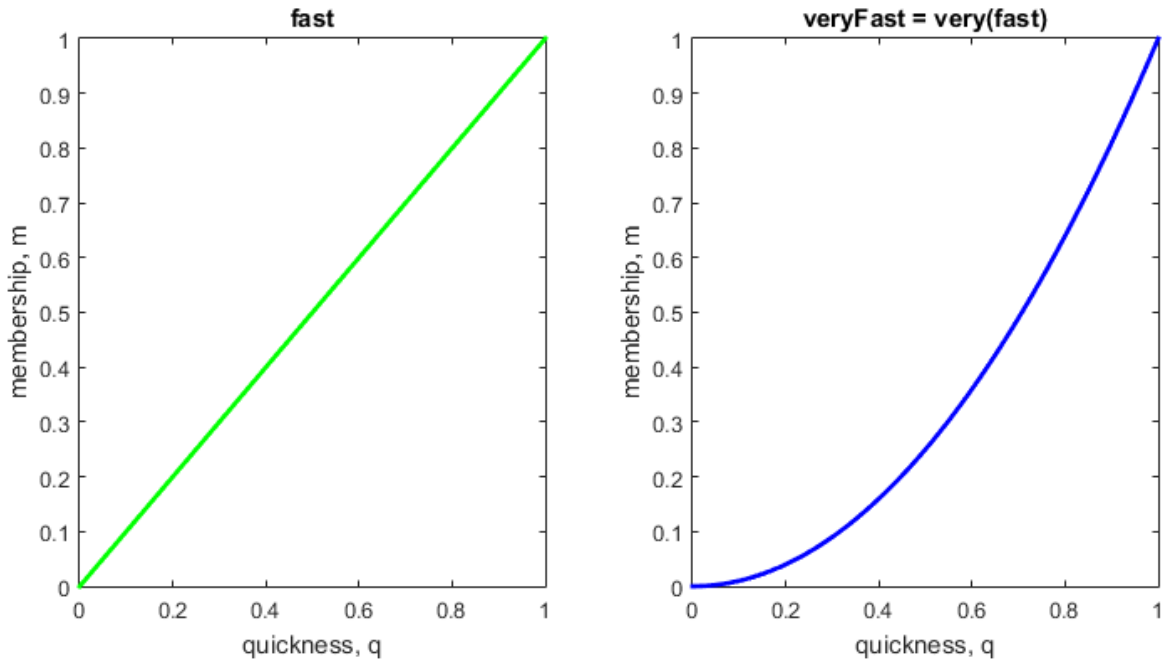

Fig. 4-10. veryFast = very (fast).

Mapping this back to our reference context, we get: $very(fast(s,t))$

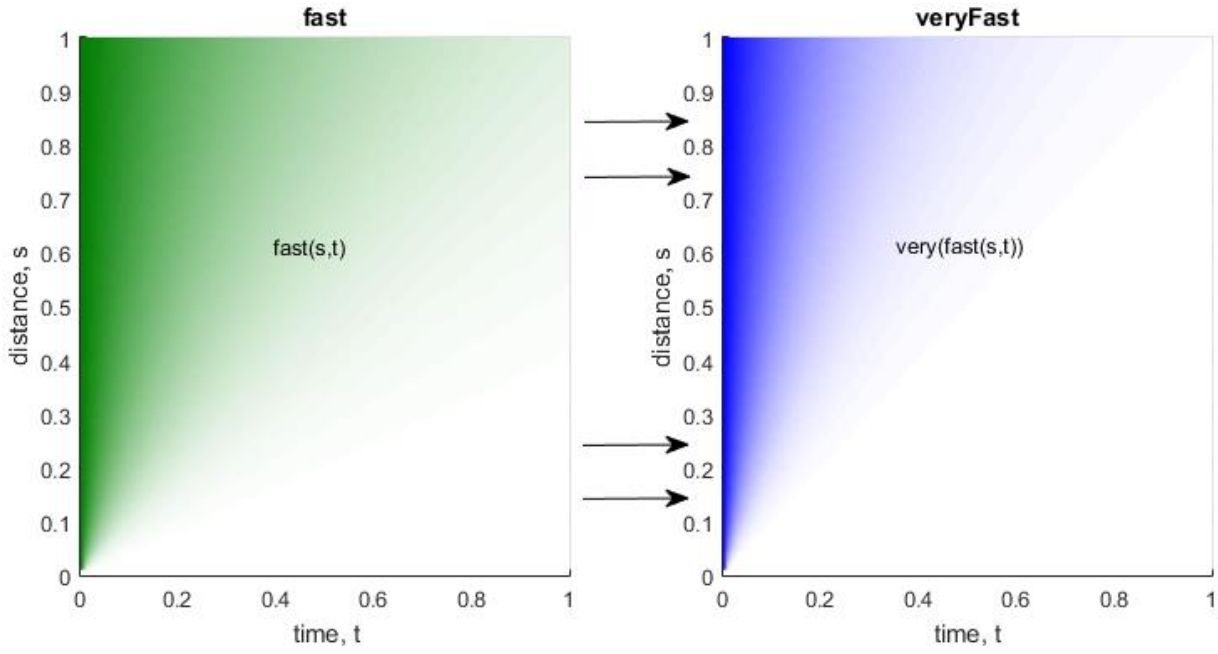

Fig. 4-11. Mapping veryFast = very(fast) back to reference context

4.3 Modeling meaning

This section builds on top of the ideas described above to create models of meanings of natural language constructs.

4.3.1 Meaning of phrases and words

As it was mentioned earlier, we assume that the knowledge of the system at any time can be described with a number of independent parameters, and that phrases are modeled as operators transforming this knowledge.

Then, *before* interpreting a new phrase we have a certain projection of the system state onto a given context: let us call it *source region* S_1. *After* interpreting, the state will change, and so will its projection onto that context. Let us call this new projection *resulting region* R_1. Now, if the source state was S_2, the resulting state after interpretation would also be different, say, R_2. Thus, each set of possible source regions $\{S_i\}$ has a corresponding set of resulting regions $\{R_i\}$.

Then *phrase meaning in a given context* can be defined as an operator transforming every source region to a corresponding resulting region. In other words, phrase meaning can be seen as mapping $\{S_i \rightarrow R_i\}$.

We are going to call the subspace containing region modifications resulting from the phrase interpretation *phrase context*. In other words, this is the subspace that the phrase operator works in. To contrast with phrase context, *current context* is the context that describes the knowledge of the system.

Phrase operators can be defined as a composition of smaller operators that correspond to words and word combinations. Let's take a look at how these can be modeled.

Many of the words, unlike phrases, can only be used together with other words. They certainly have some meaning, but this meaning is in a way incomplete until the word is put together with the other words. This fact can be modeled using parameterization: words can be seen as parameterized operators. They have a default behavior described by a certain region in word's internal context. When put together with other words (like modifiers), this region may be adjusted by those. For example, a car receiving command "drive" starts moving with some average speed. If it received command "drive fast", the speed would be different: word "fast" modifies the region of the operator "drive". In case of the command "drive very fast", word "very" modifies the region of the operator "fast" that, in its turn, modifies the region of the operator "drive" (Fig. 4-12).

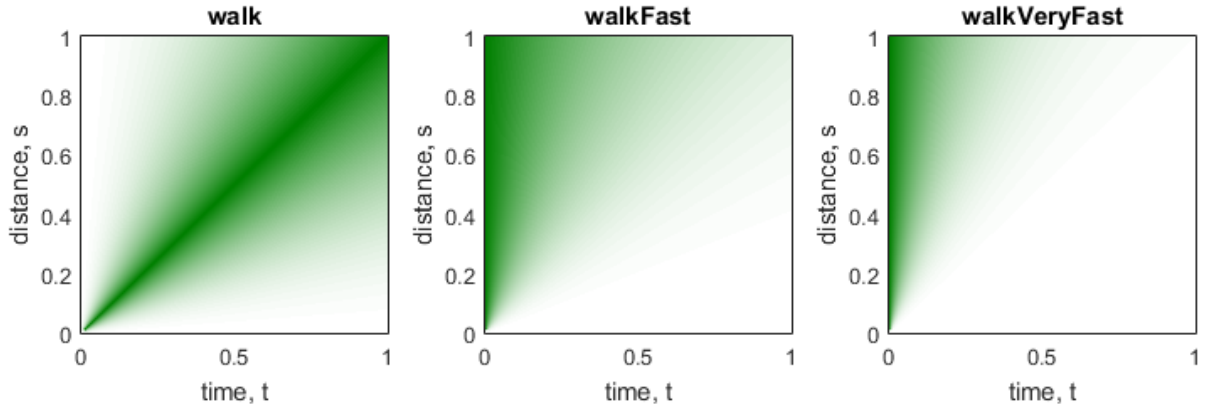

Fig. 4-12. "Walk" -> "walk fast" -> "walk very fast"

We will call such composite operators resulting from interaction of two or several modifying operators *block-operators*.

4.3.2 Parts of speech

The problem of selecting suitable and universal "building blocks" for constructing all possible kinds of meanings is extremely complex. Though, every natural language is one solution to this problem. Let's consider how some of the "building blocks" used in natural language can be modeled[1].

Qualitative adjectives ("large", "tall", "simple")[2]. Operator, corresponding to a qualitative adjective, may be modeled as a projection operator. More concretely, $I(X) \oplus P(Y)$, where $I(X)$ is identity operator, $P(Y)$ is projection operator, and $X(x)$ and $Y(y)$ are the membership functions in corresponding region subspaces. $P(Y)$ substitutes $Y(y)$ for some particular membership $Y^*(y)$. If the context has an axis directly corresponding to the adjective, then dimensionality of *y* will be one, otherwise – more than one. Here it is assumed that the source region is given as a set of independent regions with membership functions X and Y (instead of one multidimensional region[3]), and this is how the contexts are implemented within this master project.

[1] This topic is discussed in more details in [1:15-16].

[2] To some extent, this can also be applied to some adverbs.

[3] These representations are to some extent interchangeable (some more details on this are provided in [1:15].

Some examples of modeling qualitative adjectives as operators were given in section 4.2.1. In particular, the images for operators “fast” and “slow” (the regions that the adjective operators project), are shown on Fig. 4-2.

Conceptually, *nouns* can be modeled as objects that have a set of qualities. If the object did not exist from before, then the axes specific for it would be added to the context, immediately initialized with the “default” for this concept membership functions (as if a set of qualitative adjectives has been applied to the context).

This project models nouns in a simplified way. For example, nouns “crossing”, “corner” and “dead end”[1] are modeled using just one axis denoting the number of roads leading from a specific location on a map (this property is “built-in” in Google Street View API that is used in the software prototype).

Verbs can be modeled as words that work with time axis, often with its small part, and also create a special action context.

This project models verbs in a simplified way: they are seen as a collection of axes they work with. For example, “walk” is defined as a verb working with axes “quickness” (speed) and “direction”. The exact way the verb is affecting these axes depends on the adverbs[2] it is used with (“walk much slower”, “walk southeast”[3] etc.)

Adverbs. General discussion on modeling adverbs is out of scope of this project. This project models adverbs “slower”, “faster”, and adverb-like constructs “further left” and “further right” in a simplified way: they are seen as a combination of operator “more” + “qualitative adjective”, where “more” is currently implemented like a simple shift of the membership function along the *X*-axis (Fig. 4-13 and Fig. 4-14).

[1] These nouns are not implemented in the software yet, though, the axis reflecting the number of possible directions is updated according to the location on the map.

[2] Current implementation only supports combining verbs with adverbs

[3] There is one more simplification to the current version of the software: to work with directions, another verb is currently used instead of “walk” (this avoids a subtle problem with unwanted axis overwriting that is not yet fixed).

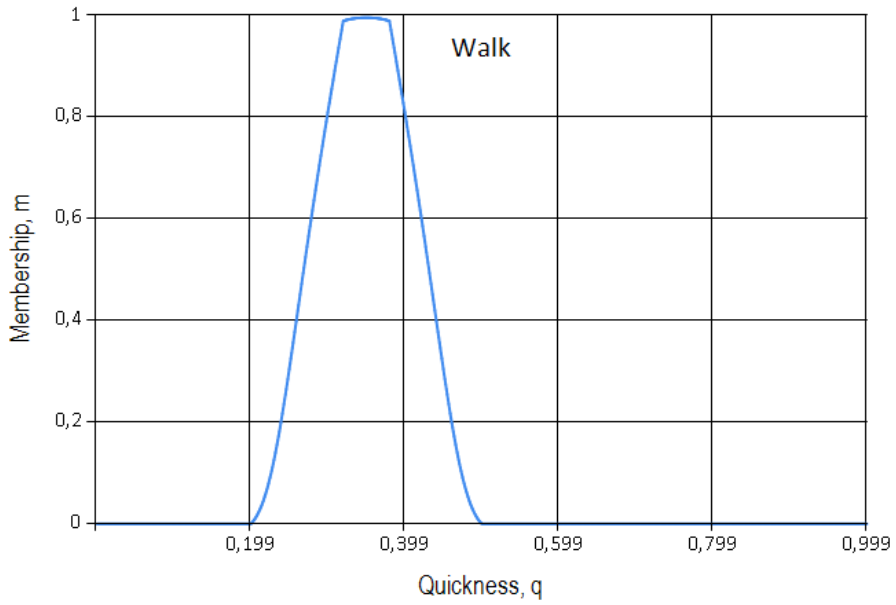

Fig. 4-13. Walk (before application of "faster")

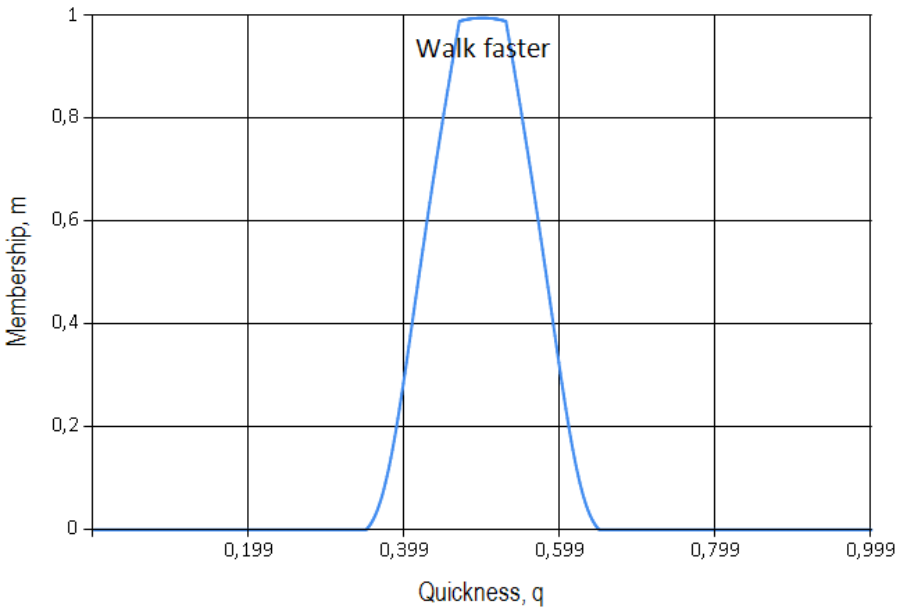

Fig. 4-14. Walk faster

Function words. General discussion on modeling function words is out of scope of this project. Modeling conjunctions is discussed in more details in [1:15]. Word "more" was mentioned earlier in this section, and concepts "not" and "very" were discussed in section 4.2.2 (Fig. 4-6 and Fig. 4-9).

4.4 Interpreting phrases

4.4.1 Overall process

This section briefly discusses the overall process[1] of phrase interpretation. Please consider the schematics (Fig. 4-15). This is the same drawing as Fig. 4-1, provided here again for reader's convenience.

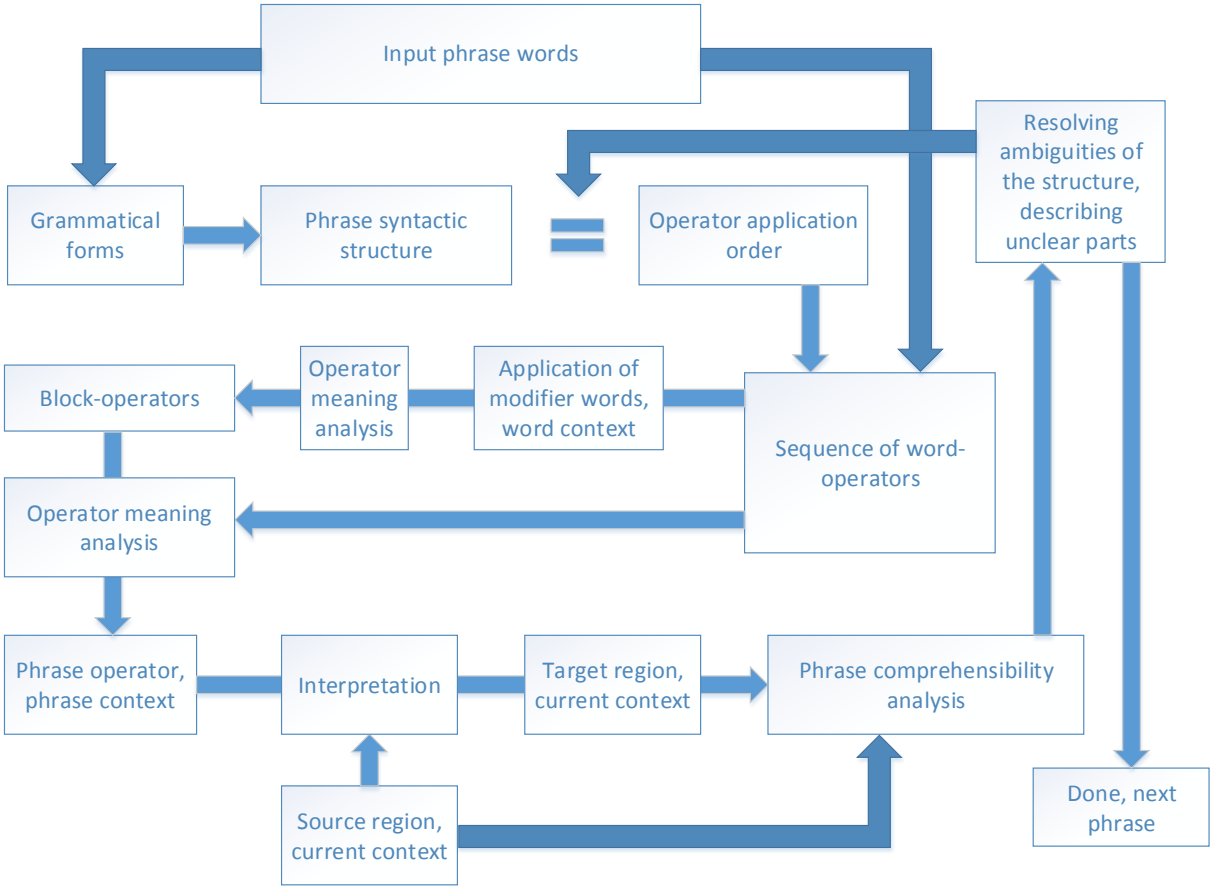

Fig. 4-15. Simplified schematics of phrase interpretation.

First, when a new phrase comes in, the syntactic structure of the phrase should be obtained from the grammatical forms of words as much as it is possible. In other words, this means determining the order of application of operators and block-operators to one another. In case when the syntactic structure is ambiguous, the possible options need to be remembered so

[1] Please note: Only some parts of the process described here are implemented in this master project. The details are described in section 4.4.3.

that the corresponding operators can later be compared based on their clarity (comprehensibility) score. Determining phrase comprehensibility is discussed in section 4.4.2.

After, the sequential application of operators to one another is done. More precisely, the operators that modify the internal context of another operator, do that (yielding block operators), while the operators working with the external context of the narrative, form a line, in the order of application. Result is the phrase operator: composition of several word and block operators. Construction of the phrase operator includes its analysis for meaningfulness.

After that, the phrase operator is applied to the source region in the current context (by sequential application of word and block operators from the line). The result of this application is the resulting region. Given both source and resulting regions, overall phrase comprehensibility is assessed (section 4.4.2). If there are several possible interpretations of the phrase (several phrase operators), the best option is chosen according to the comprehensibility score. If none of the interpretations reach the threshold, the phrase may be considered unclear. Other options of handling this are discussed in [1:16-17].

The ability to analyze different interpretations from comprehensibility point of view allows to deal with situations when syntactic structure of the sentence cannot be completely restored from word order and grammatical forms (like when using a voice interface), and when for restoring it people would also need to resort to semantics.

4.4.2 Assessing comprehensibility

This section describes some techniques and ideas that are used to determine whether a statement or a phrase makes sense, and that can be used to decide which of the possible interpretations is the right one[1].

The most basic type of sensibility assessment can be checking whether the words used to assemble a phrase operator are actually compatible, in particular, in terms of their axes. For example, “fast vehicle” makes sense, because both adjective “fast” and noun “vehicle” would include “quickness” axis as part of their definition areas, as long as both concepts have to do with this property. On the other hand, “fast blanket” does not really make sense, because

[1] Choosing between several interpretations is not implemented in the current version of the software.

noun “blanket” has really nothing to do with “quickness”, and therefore would not have this axis, something that can easily be detected by the software.

Let’s take a look at detecting other types of incomprehension and meaningless phrases.

A region that has no points with high enough degree of membership (say, 0.95) can correspond to a *contradiction*. Indeed, this situation would mean that there is no property combination in the context that definitely corresponds to our concept. For example, if we consider hypothetical concept *and(slow, fast)*, with *and* defined similar to Zadeh’s “product” [42:11], a typical contradiction can be seen (Fig. 4-16).

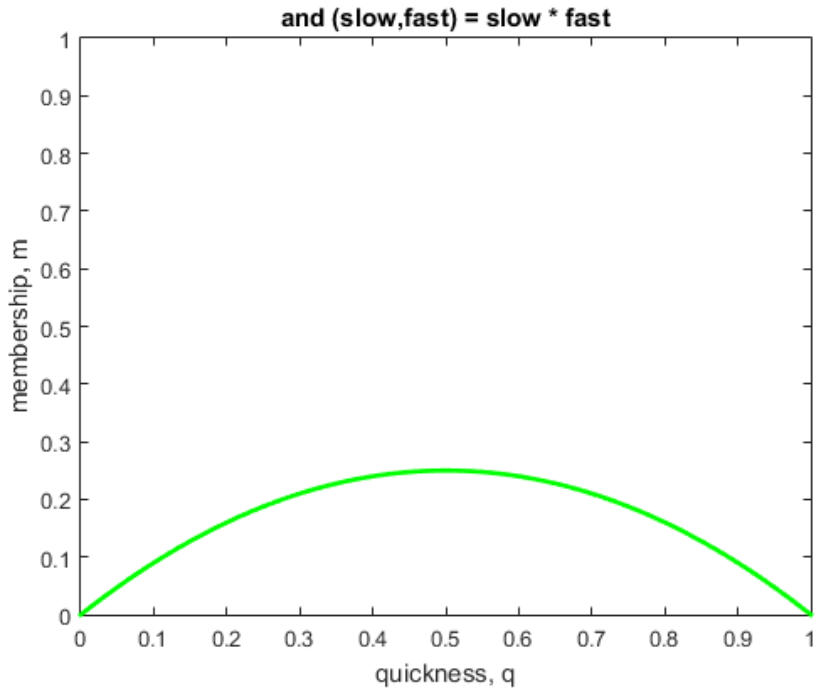

Fig. 4-16. Contradiction

On the other hand, if almost all the points of the context belong to a region, this usually does not make much sense either, as it provides no information (“it can be anything”). For example, if we consider a hypothetical concept *or(slow, fast)*, with *or* defined as $or(m1, m2) = not(not(m2) * not(m2))$, this is the situation (Fig. 4-17).

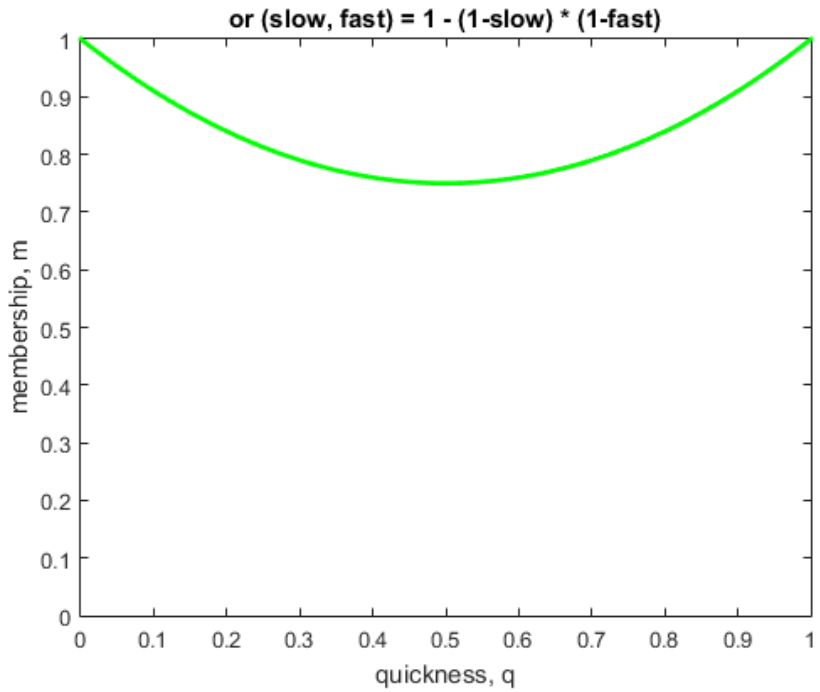

Fig. 4-17. Lack of information

Another potentially useful heuristics has to do with the fact that normally new phrases provide new information, in other words, change something in our knowledge. Let's compare these projections of verb-like concepts "walk" and "stand still" on a context with axes "time" and "distance" (Fig. 4-18).

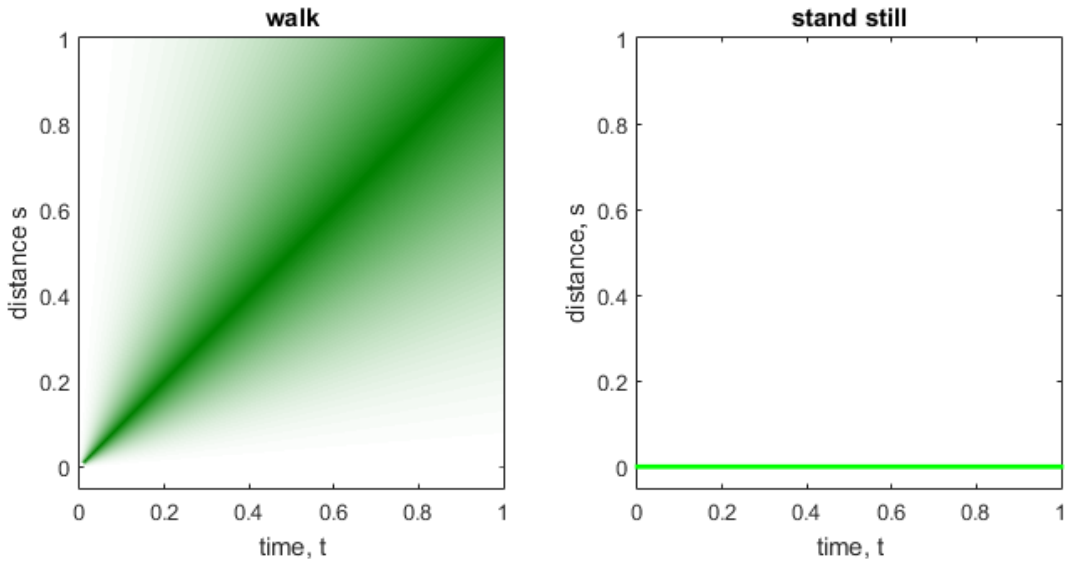

Fig. 4-18. "Walk" and "stand still"

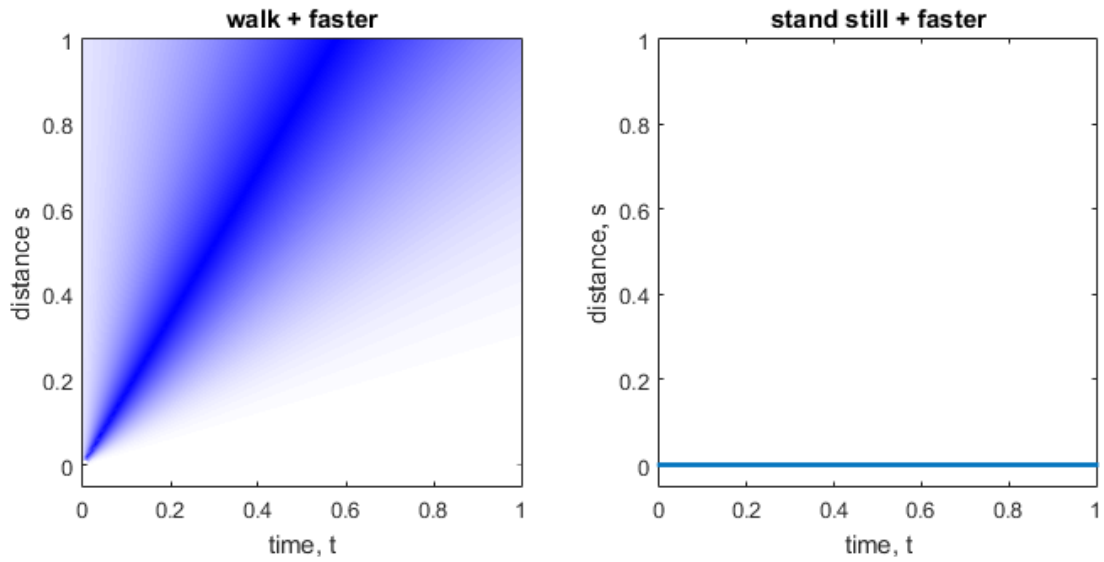

Fig. 4-19. "Walk" + "faster" and "stand still" + "faster"

If we apply operator "faster" to these concepts, "walk faster" will result in a different region, while "stand still faster" will stay the same, because "compression" over time axis performed by operator "faster" will have no effect (Fig. 4-19). So, after comparing the regions[1], we can conclude that phrase "stand still faster" makes no sense, indeed.

In many cases, new information not only changes our knowledge, but often is expected to precisiate it rather than making it more vague. This is especially the case when a system is receiving some instructions.

For example, if we are trying to direct a robot, and the region describing it is transformed from "Somewhere NE" to "Anywhere except SW", it may be a sign of misunderstanding (Fig. 4-20). An exception to this could be a special phrase "forget it, I will explain from the beginning".

[1] In the current version of the software this comparison of membership functions is implemented over one-dimensional axes-properties (like "quickness").

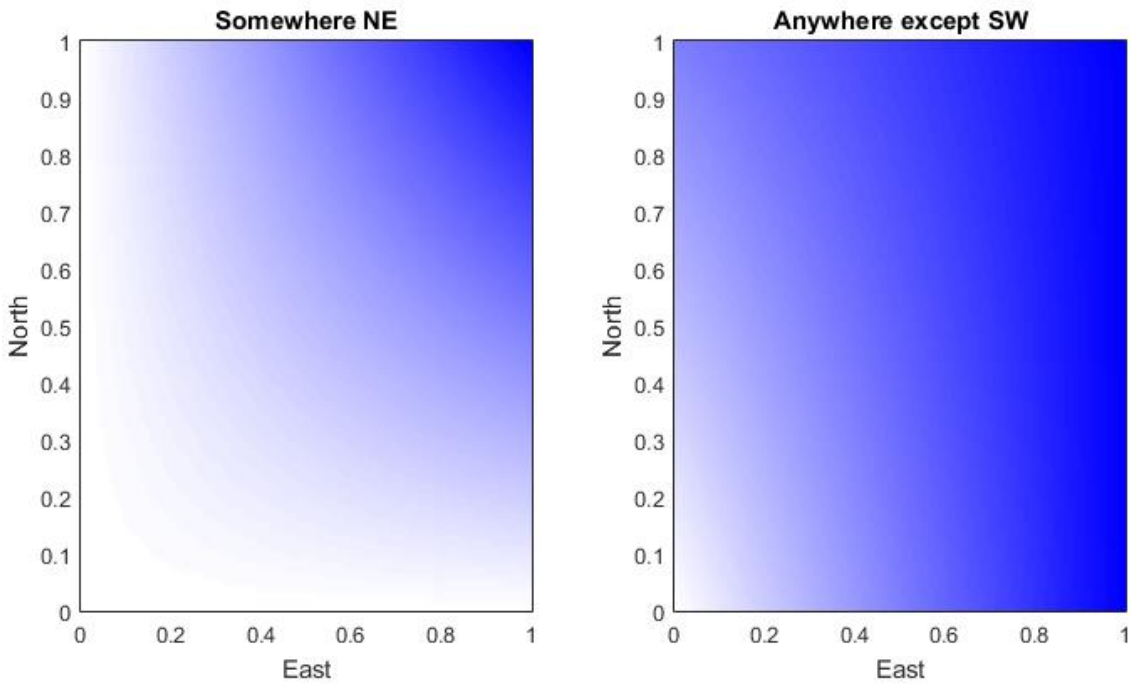

Fig. 4-20. Description getting more vague

Another interesting class of sensibility assessments is analysis of the correspondence between statement's assumed goal, and the information it actually conveys. For example, it is possible to check whether the grammatical mood of the message (realis, imperative, conditional etc.) corresponds to the resulting region.

For example, parameter "movement speed" could be connected to a controller that can alter the speed. Then, if the region with membership value above certain threshold is small and simply connected (like region corresponding to a command "drive fast"), it is possible to find the center *C* of this region and output command "apply value *C*" to the controller's effector. However, in case of the command "drive very fast or very slowly" it would be natural for the system to ask for clarifications, even though the phrase itself is clear (unlike phrase "drive very fast AND very slowly").

On the other hand, when the phrase is in realis or conditional mood ("but if I was driving very fast or very slowly"), such region shape is perfectly acceptable and does not require any clarifications.

Another somewhat similar heuristics can analyze whether the words used in the phrase agree with each other when it comes to transforming the source region. For example, let's consider a robot that can "walk" and "run", and now it is running. Robot receives command "walk

faster". We can examine that the verb "walk" is moving the region in one direction, trying to decrease the speed (compared to "run"), while adverb "faster" is trying to move the region in the opposite direction. This can be seen as an indication of misinterpretation.

4.4.3 Implementation note

This project only implement some parts of the proposed implementation process and comprehensibility heuristics.

Regarding the interpretation process, no grammatical forms are analyzed as of yet. After the parsing, a simplified method is used to compose phrase operators from word-operators (including modifiers). Both analysis of operator meaning and overall phrase comprehension is performed as described, but currently no distinction between phrase and current contexts is made, and only one interpretation is taken into account.

Regarding comprehensibility heuristics, the first five methods described are implemented so far (axis mismatch, detection of contradiction and tautology, and situations that have to do with no information change and no information gain).

More concrete description of the software is included in chapter 5.

4.5 Comparative summary

This section attempts to show some of the differences of this approach compared to the others (chapter 3).

Let's start by discussing how the challenges mentioned in 3.2.2 and 3.4.3 could be solved (or attempted solved) with the proposed approach. In the cases when there is not an answer ready, some hypothesizing is done, informal but hopefully still useful for comparing the models.

Questions from section 3.2.2:

1. Issues with predicate quantifiers. Are all cakes sweet? Or only some cakes?
   In the proposed approach, it is modeled with truth degrees. Definition of the noun "cake" would probably have axis "sweetness", and its membership function would indicate that the truthness of a random cake being "ultimately sweet" is not very high,

but higher, than the truthness of a cake being “not sweet at all”. The membership function would have its highest values around sweetness degrees between, say, 0.4 and 0.8, corresponding to the range from “a bit sweet” to “very sweet”.

2. *Equivalence of the meanings in a given context. Does “satisfy hunger” mean the same as “eat” in a given context?* This will depend on the way the concepts are defined. Both of these verb-like concepts can be defined as decreasing the value of axis “hunger level” over a short period of time. In this case, the corresponding operators would give similar transformations in a given context, and thus, it can be concluded they mean the same.
3. *Fuzziness degrees. To what extent are cakes sweet? Is one’s stomach going to be equally full after “eat breakfast” and “eat dinner”?* For the first part, please see the answer to question one. Concepts “breakfast” and “dinner” could include an axis that defines the amount of food, and would act like a parameter for the verb “eat”, resulting in different block-operators for “eat breakfast” and “eat dinner”.
4. *Uncertainty handling. What’s the possibility that one would yawn after waking up in the morning?* Some uncertainty can be modeled with membership functions. This case is more complicated. Just the definition of the concepts (“wake up”, “morning”, “often”, and “yawn”) does not contain any knowledge on the matter, it has to be provided. Though, it is interesting that the system could learn this knowledge on its own by reading and interpreting phrase “people often yawn after waking up in the morning”. The operator corresponding to this phrase would add a special causality axis to the context, and the word “often” would be a modifier affecting this axis, by this defining the probability of the event.
5. *Relation between basic and composed concepts. How does concept “early” relate to concept “get to bed early”?* In the proposed approach, there won’t be an explicit definition of the concept “get to bed early”. However, when “get to bed early” is met in a text, appropriate axes would be added to the context, describing this idea. Word “early” would alter the membership function of the time axis, making its membership function less fuzzy (than it was as a result of “get to bed”). The updated membership function would have its highest truth degrees around the part of the axis corresponding to, say, 22:00.

6. *Composition of concepts into phrases. Does "sweet person" make sense in a given context, or not? Can an "oven" "follow a recipe" to "bake" something or not?* "Sweet person" would make sense if "sweet" adjective definition contains axis "person character", and would not make sense otherwise. Interpretation and assessing comprehension was covered in section 4.4 as much as the scope of this master project allows. Some more details, including synonyms and homonyms, are provided in [1].
7. *Concept definitions. How is "early" defined? How is "morning" defined? Can a computer easily decide whether it is "early" or "morning" at a given time?* These concepts can be defined as regions, and when it comes to the time axis, the regions would look fairly similar. It is easy for the software to detect whether it is "early" or "morning" at any point of time by checking the membership degree of the point corresponding to the hour in question.
8. *Exact interpretation of the relations. Many of the relations in semantic networks computers cannot really handle. People often go to bed early so they easier wake up in the morning (relation "MotivationOf"). But what can a computer infer from this? Given one goes to bed late, does it mean that one doesn't want (need) to wake up in the morning? That one won't wake up in the morning? That one won't eat breakfast?* In the proposed approach, this one is bit similar to question 4. For the system to learn this knowledge, it would need to read and interpret the phrase "People often go to bed early (A) so they easier wake up in the morning (B)". This phrase-operator would add a special "motivation" axis to the context, connecting part A with part B. Motivation "A –> B" means causality "A –> B", but the probability of that causality is unknown. The reasoning engine can detect it and ask "what is the probability of B given A"? Before that is known, no further assumptions can be made in this case.

Let's now compare the proposed approach with CWW [53, 54]. As it was mentioned previously, in CWW, figuring out that "young" has to do with the implicit variable "age" is a part of manual precisiation step. In the proposed approach these implicit variables are explicit from the beginning (as they are made into axes and are part of concept definitions). For example, concept "fast" specifies how exactly it is "constraining" the variable(s), e.g. "distance", "time" (Fig. 4-2), or "quickness" (Fig. 4-3).

Both CWW and the discussed approach pursue goal of working quantitatively with natural language concepts. However, as it was already mentioned in section 3.4.3, their focus is very different. CWW aims to perform correct computations when dealing with words and perceptions rather than numbers. CWW is completely dependent on the precisiation step rewriting natural language into GCL [50, 51], before it can start working with natural language concepts quantitatively. CWW does not include quantitative models of meaning, or provide ways to assess comprehensibility of natural language phrases in software.

The discussed approach does exactly this, as it is focuses on interpreting natural language "as it is", attempting to analyze the meaning and extract the knowledge encoded in phrases without the need for language to be precisiated first.

Table 4-1 is a somewhat subjective attempt to summarize the differences between different knowledge representations.

Table 4-1. Comparison of different knowledge representations.

	Procedural knowledge	Semantic networks	Logic-based approaches	Computing with words	Proposed approach
Creating/learning new knowledge	No	Easy [19]	Difficult [19]	Difficult	Easy
Generalizing and scaling	No [9, 10]	Yes [19]	Yes	Yes	Yes
Focus on quantitative representation of meaning	No	No	No	Yes, but only after precisiation	Yes
Exact specification of details	Yes	No [21]	Yes	Yes	Yes
Extracting knowledge directly from natural language	Yes, but only for the hardcoded part	No	No	No (needs precisiation) [53, 54]	Yes
Exact reasoning	Varies	Poor [21]	Good	Poor	Poor
Approximate reasoning	Varies	Varies [48, 19]	Poor [48]	Good	Good
Uncertainty handling	Varies	Poor [48]	Poor [48]	Good	Good
Model and implementation complexity	Varies	Low	Medium	High	Very high

5 Description of the software

This chapter describes the software that was implemented:

- Natural language understanding framework (later: framework).
- Research prototype of the intelligent navigation assistant, illustrating abilities of the framework, in particular, computational analysis of the meaning of spatial directions (later: prototype).

It should be noted that both framework and assistant were developed together. On one hand, framework features were implemented to allow for specific prototype features. On the other hand, prototype features were planned to illustrate framework capabilities. This is why it also makes more sense to describe the software as a whole.

5.1 Functionality

The software is implemented as a desktop application (Fig. 5-2), as it is the most feasible option for a research prototype[1]. The user can dialog with a system using either voice interface[2] or text input. Every phrase of the user is interpreted by first being analyzed for comprehensibility, and then being either accepted or rejected. If the phrase is accepted, the system reacts with appropriate actions.

The system implements two features of the intelligent navigation assistant: mode “interactive fly-over” and mode “I am telling you that”.

5.1.1 Mode “Interactive fly-over”

This is a mode when the user quickly previews the area before deciding whether she actually wishes to get (drive or walk) into that area. The user sees the location being previewed both on a map and in a Google Street View simultaneously. She may issue commands like “run faster”, “look further left”, “steer northeast”. Using these commands, she can control:

[1] The software can be fairly easily adapted to run, for example, on a mobile phone, using e.g. Xamarin.

[2] The software uses a standard voice recognition library (System.Speech in .NET).

- Heading direction (e.g. “steer north”).
- Camera movement speed (e.g. “walk”, “walk faster”, “slower”, “crawl”, “stop”).
- Camera look direction (e.g. “look back”, “look further right”, “further left”, “look forward”.

The camera moves with requested speed along Street View “links” (available directions on the map), and the link that matches requested heading direction best is chosen. The look direction in Street View updates with user commands. As the camera is moving, system’s state is updated with the number of available directions on the map (according to the location). Currently, this parameter (“DirectionMultitude”) is only visualized in the user interface, and not directly used. The plan is to use it to define words “crossing”, “corner”, and “dead end”, and supporting commands like “stop at next crossing”, “turn left” etc.

The “interactive fly-over” feature makes full use of the fuzzy logic framework.

5.1.2 Mode “I am telling you that…”

In this mode, the user tells system about something, and the system reacts depending on the information. For example, if the user says “I need coffee”, the system will highlight all coffee places nearby (Fig. 5-1).

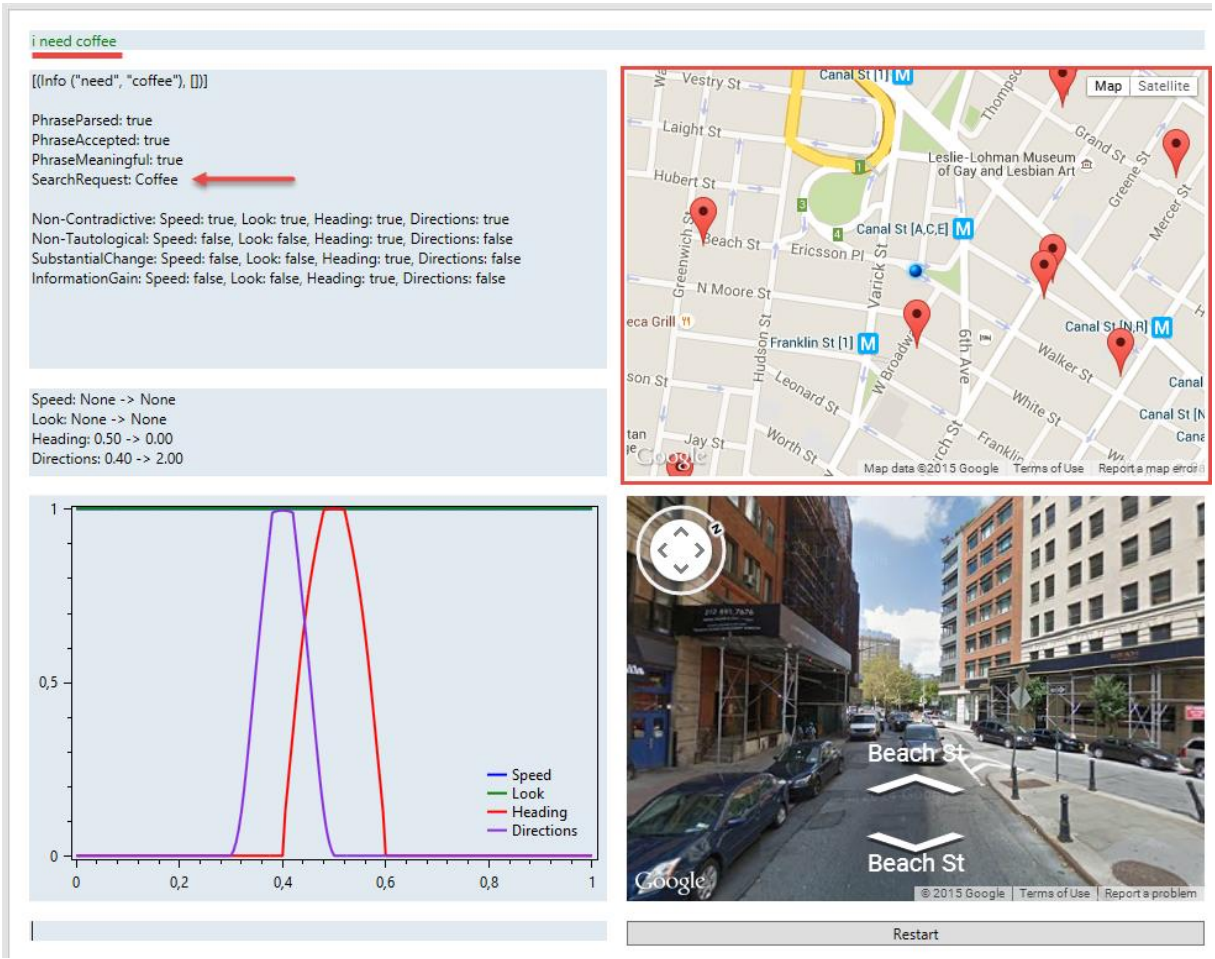

Fig. 5-1. The user says she wants coffee

This feature is not making use of the fuzzy logic framework, and is mentioned here mostly for completeness.

5.1.3 User interface parts

The user interface of the system contains of the following parts (see numbers on Fig. 5-2).

1. *Last phrase status.* Displays the last phrase entered by the user, and shows (with color) its comprehension status (see number 3 for details).
2. *Parse tree.* Prints out the abstract syntax tree for the phrase.
3. *Parse and comprehension results.* Displays whether the phrase was:
   a. Parsed,
   b. accepted (meaningful enough to compose phrase operator),
   c. meaningful (all comprehension tests passed).

Last item in this section shows where any specific "search request" is set, based on the feature "I am telling you that…". For example, if the user says "I want food", the search request will be set to "restaurant".

4. *Comprehension heuristics.* Displays the status for all of the implemented heuristics, per parameter / axis: "contradiction", "lack of information" (tautology), "substantial change" and "information gain". These heuristics were described in more details in section 4.4.2.
5. *System state (defuzzified[1])* in form "defuzzified value -> value in sensor/effector units". For example, consider Fig. 5-2. "Look: 0.65 -> 54.00" means that after defuzzifying the region for axis "LookingDirection" (corresponds to the green line on the chart), the most probably value is 0.65. Converted to effector units (camera look direction), in this case it corresponds to 54 degrees.
6. *System state (graphically).* Displays membership functions for all of the four axes, corresponding to parameters speed, look direction, heading direction, and direction multitude (number of Street View links) at the current location.
7. *Input field for new phrases.* This is the field where the user can input new phrases. Alternatively, she can hold "Shift" button and say the phrase ("Shift" activates the voice interface).
8. *Map.* Displays current location on a map with a blue marker.
9. *StreetView.* Displays current location in Google Street View, taking look direction into account.
10. *Restart button.* Resets all parameters and brings the camera to the initial location.

[1] Here the term "defuzzification" is used in the normal for fuzzy logic sense: converting from a fuzzy region / membership function to one specific value. In this project defuzzification results are always between zero and one.

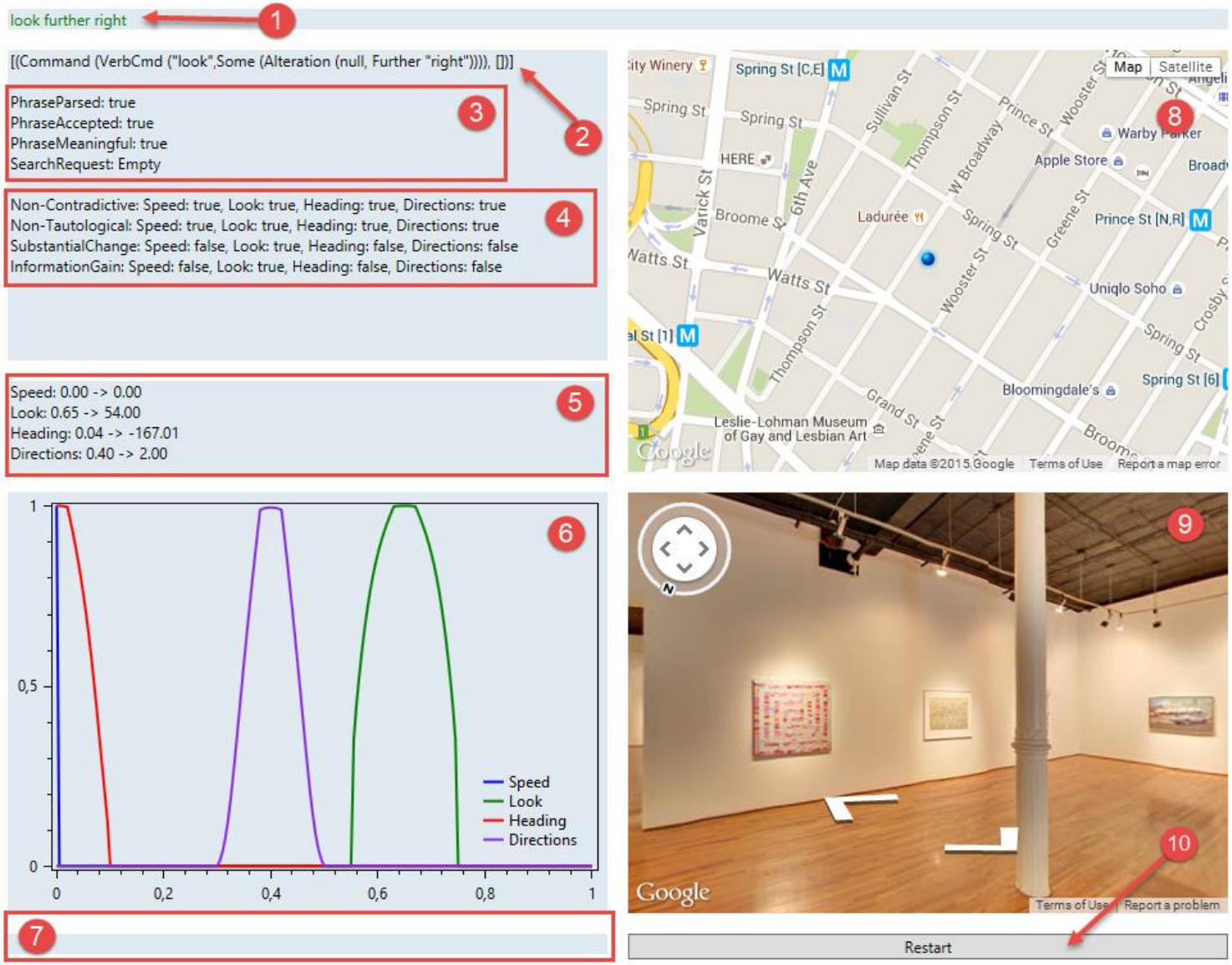

Fig. 5-2. Running research prototype. 1. Last phrase status. 2. Parse tree. 3. Parse and comprehension results.4. Comprehension heuristics. 5. System state (defuzzified). 6. System state (graphically) 7. Input field for new phrases. 8. Map. 9. StreetView. 10. Restart button.

5.2 Phrase interpretation

This section describes phrase interpretation process, the central part of the software. As it was mentioned earlier, its implemented part is much smaller, than the process described in 4.4. The implemented part is illustrated on Fig. 5-3. Contrasted to Fig. 4-1, it shows a much smaller part, but provides more details.

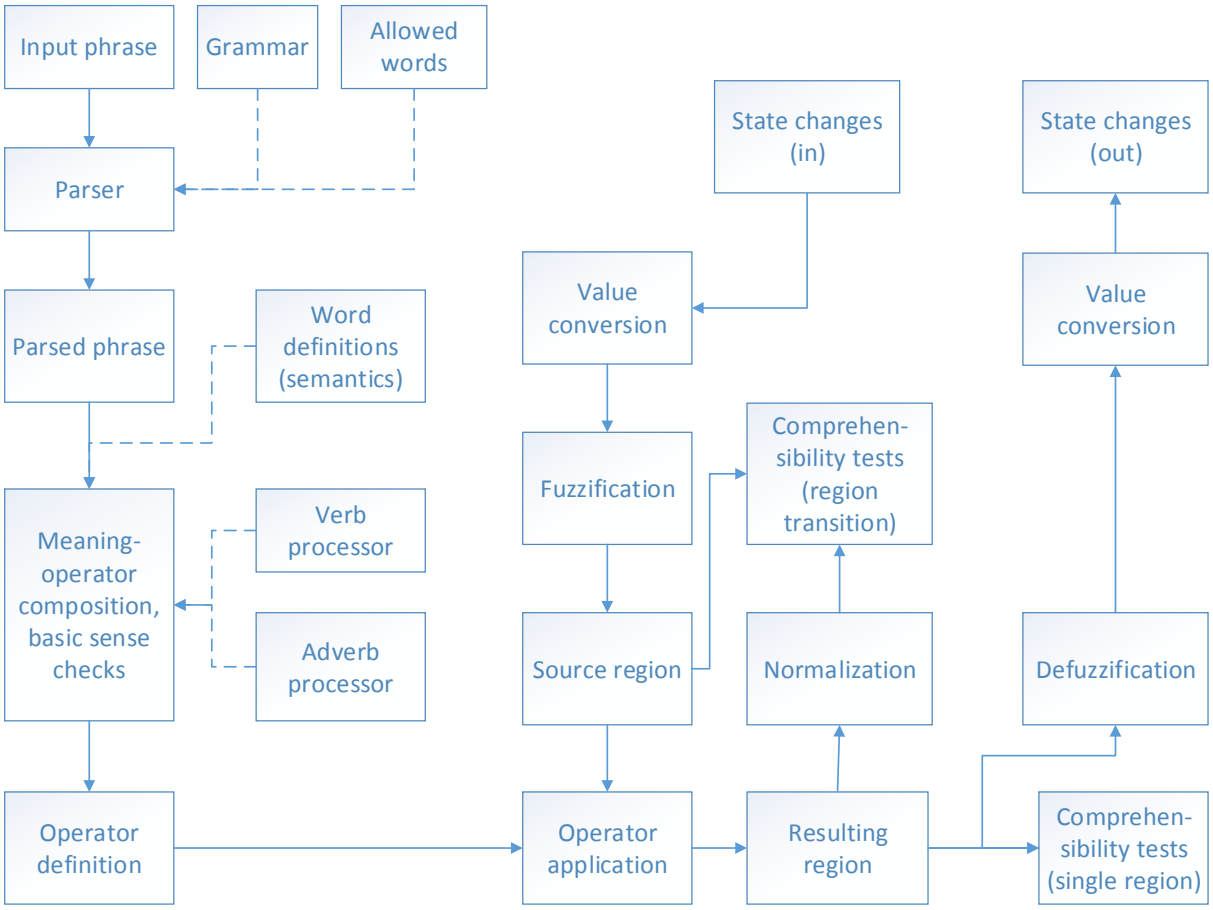

Fig. 5-3. Schematics of phrase interpretation as implemented. Solid line shows main workflow. Dashed line means "uses".

First, the input phrase is parsed. In future versions, the framework should be able to work with a very relaxed grammar, but current version uses a very simple and fairly restrictive grammar (Code Fragment 5-1) [1].

[1] This and other code fragments are in F#. They are simplified and incomplete to fit illustration purposes.

```
type Very = Very

type DirectionSpecifier = Very option * string //very far
type Direction = DirectionSpecifier option * string //very far left
type Quality = Very option * string //very fast

type Definition =
    | Direction of Direction //right, very slightly left
    | Quality of Quality     //very fast

type BasicAlteration =
    | QualityAlteration of string //faster
    | Further of string //further left

type Alteration = string option * BasicAlteration //much faster, slightly
further left

type Adverb =
    | Definition of Definition
    | Alteration of Alteration

type Command =
    | VerbCmd of string * Adverb option //look left, move slower
    | AlterationCmd of Alteration //slower, much faster

```

Code Fragment 5-1. Grammar.

For example, the grammar says that a command is either a “verb command” (like “walk slower”) or an “alteration command” without a verb (like “further left”). Then, in its turn, and “adverb” is either a “defining adverb” (like “slowly”) or an “altering averb” (like “slower”), etc.

Current grammar version relies on the list of allowed words (Code Fragment 5-2).

```
let _allowedWords = {
    Pronouns = ["I"; "i"; "we"];
    InfoVerbs = ["need"; "want"];
    Objects = ["coffee"; "food"];

    CommandVerbs = ["walk"; "run"; "crawl"; "look"; "face"; "steer";
 "stop"];

    DirectionVectors = ["left"; "right"; "forward"; "back"; "north";
 "south"; "east"; "west"; "northeast"; "northwest"; "southeast";
 "southwest"];

    DirectionSpecifiers = ["far"; "slightly"];

    QualityWords = ["fast"; "slowly"];
    QualityAlterations = ["faster"; "slower"];

    Very = ["very"];
    Further = ["further"];

   AlterationModifiers = ["much"; "slightly"];
}
```

Code Fragment 5-2. Allowed words.

Given the parsed phrase and using semantic word definitions (section 5.3), the phrase operator is being composed. Phrases like “look slowly” will be rejected already at this stage.

Later, the phrase operator is applied to the source region, yielding the resulting region. If the resulting region passes single-region heuristics, its membership function is normalized, and comprehensibility tests for region transition are run.

Lastly, the resulting region is defuzzified, the obtained value is converted to effector units, and the state is updated accordingly (e.g. adjusting the speed, heading or look direction of the camera).

5.3 Word definitions

This section describes how the words are currently defined in the system, using some examples.

Two things should be noted here. First, current implementation models contexts in a simplified way, compared to what is described in the most of chapter 4. Instead of being stored

as one multidimensional region, it is stored as a set of independent one-dimensional regions, one region per axis[1]).

Secondly, membership functions are implemented using analytic functions and their combinations[2].

Let's take a look how some of the words are defined in the software (Code Fragment 5-3).

```
1  let fastAdverb = {
2      Axes = [MovementSpeed]
3      Membership = id
4  }
5
6  let fasterAdverb = {
7      Axes = [MovementSpeed]
8      Transformation = more
9  }
10
11 let furtherRight = {
12     Axes = [LookingDirection; HeadingDirection]
13     Transformation = more
14 }
15
16 let lookAction = {
17     Axis = LookingDirection
18     Membership = defaultLook
19 }
20
21 let steerAction = {
22     Axis = HeadingDirection
23     Membership = noInformation
24 }
25
26 let getDefinitionAdverb adverb : DefinitionAdverb option =
27     match adverb with
28     | "fast" -> Some fastAdverb
29     | "right" -> Some <| makeLook right
```

Code Fragment 5-3. Word definitions (semantics).

Let's start with "fast". First, line 2 says that it is compatible with only one axis "MovementSpeed". This means, that if it is used with a verb, the verb should also be compatible with this axis. For example, construction "look fast" will be rejected during

[1] These representations are to some extent interchangeable (some more details on this are provided in [1:15]

[2] Instead of e.g. using interpolation over control points, that is a much more generic approach

composition of the phrase operator. Membership function for “fast” (line 3) is simply identity function (this is similar to Fig. 4-3).

When it comes to “faster”, it works with the same axis as “fast”, “MovementSpeed”, and is defined in a simplified (non-general) way, using transformation function *more* (line 8), that “shifts” the membership function to the right along the *x*-axis. This was illustrated on Fig. 4-13 and Fig. 4-14. A more general way of implementing comparative adjectives, that would also apply to adverb “faster”, is briefly mentioned in [1:15].

The same transformation function more is used by the adverb-like construct “further right”, that works with axes “LookingDirection” and “HeadingDirection”. This construct is now implemented as one concept as a simplification[1]. Translation from words “further left” to this concept is done after parsing.

Let’s consider “look” (lines 17-18). It is defined using membership function like shown on Fig. 5-4. This means that it is normal to look more or less straight unless something more specific is said.

[1] It is interesting to note that, for example, Russian has a special word for this concept, and it is not possible to say “further left” in two words in Russian.

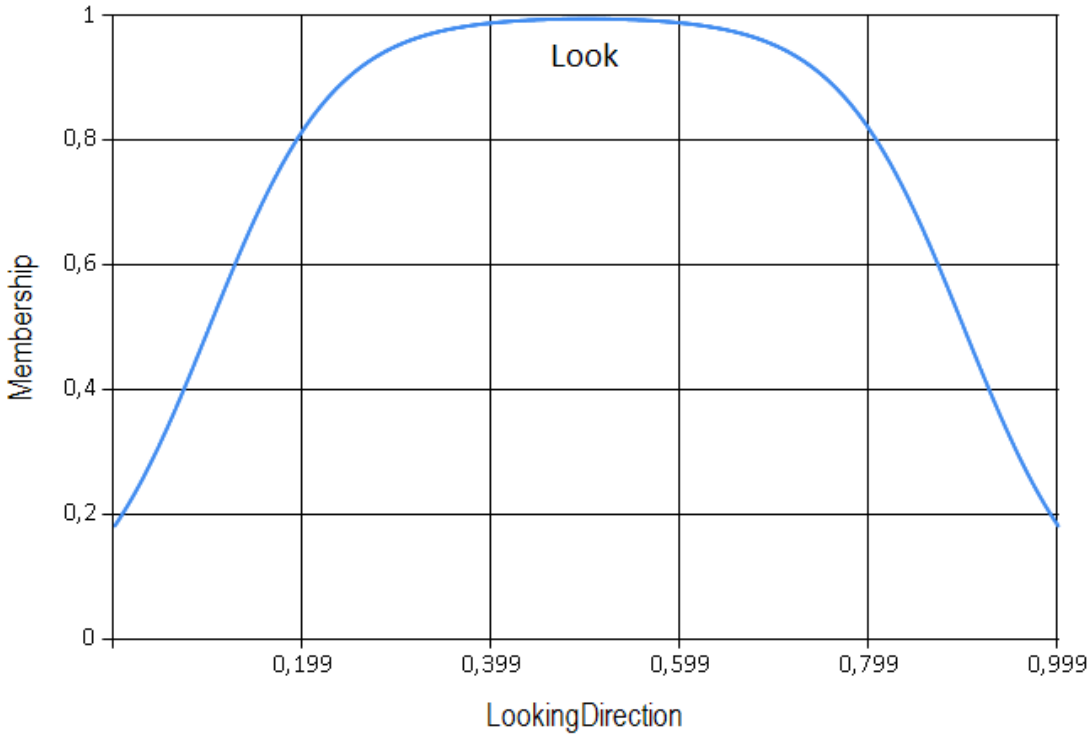

Fig. 5-4. Membership function for "look"

Word "right" (line 29), is implemented with a function like shown on Fig. 5-5. If the user says "look right", membership functions for "look" and "right" are blended, resulting in a function similar to one shown in green on Fig. 5-12, left.

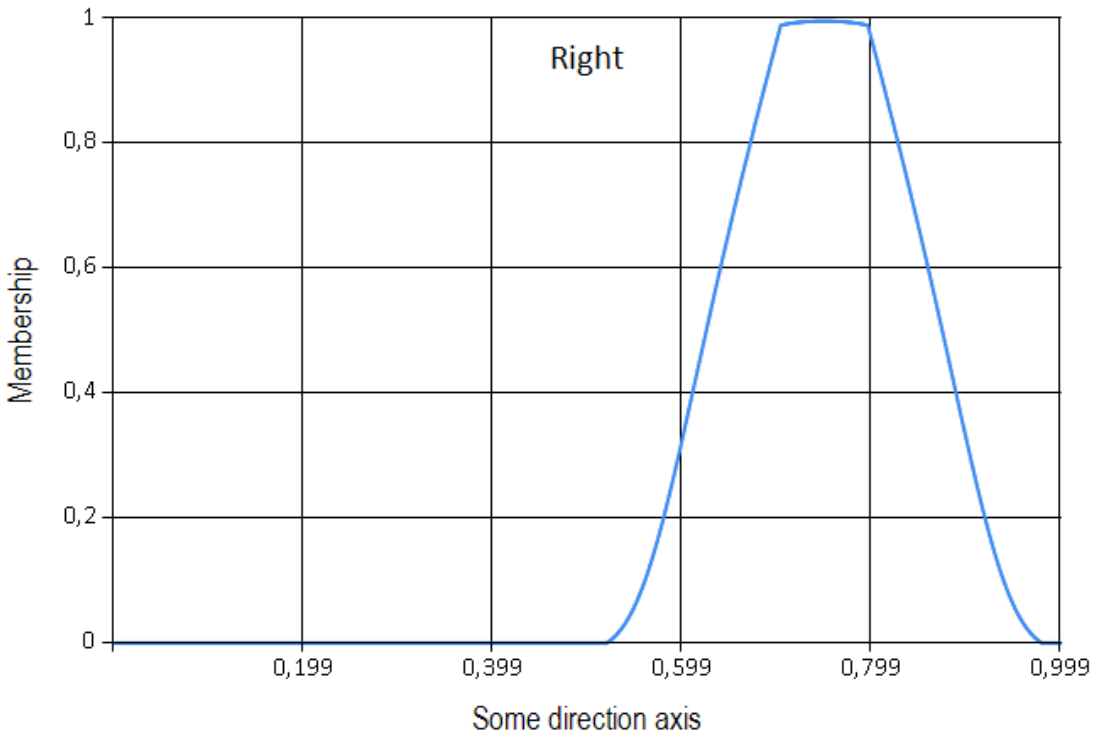

Fig. 5-5. Membership function for "right"

Let's now take a look at "steer". It works with only one axis: "HeadingDirection", and its membership function is defined as "noInformation". This function is defined as equal to one in all the points (Fig. 5-6). This represents the fact that, unless it is said specifically where to steer, the system doesn't know.

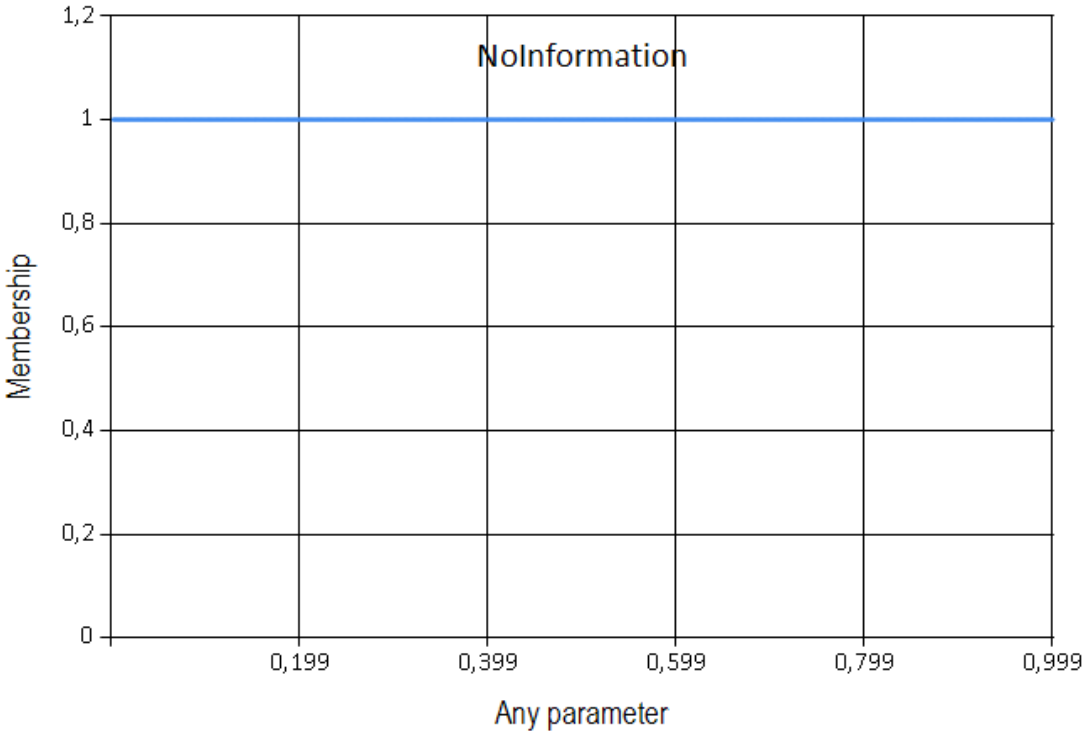

Fig. 5-6. No information (for any parameter)

5.4 Examples

This section presents some commented examples of using the software.

1. Missing semantic word definition. The command is shown in pink, this means that it is parsed, but not accepted (this is because the word “much” is lacking semantic definition).

```
walk much faster

[(Command
   (VerbCmd
     ("walk",Some (Alteration (Some "much", QualityAlteration "faster")))), [])]
```

Fig. 5-7. “Walk much faster”

2. Command “look further right”. Before (Fig. 5-8) and after (Fig. 5-9). Membership function for “look direction” is shown in green on the figures. Defizzified value changes from 0.5 (corresponding to 0 degrees) to 0.65 (corresponding to 54 degrees).

Fig. 5-8. "Look further right". Before.

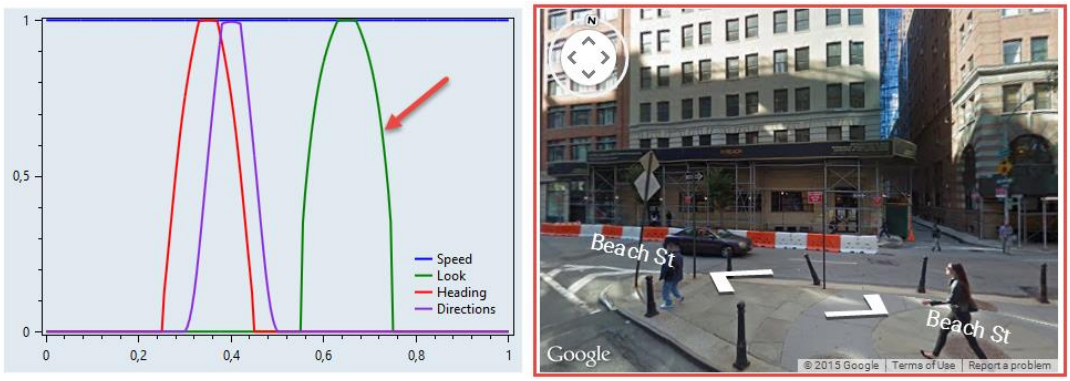

Fig. 5-9. "Look further right". After.

3. Command "walk faster" after command "walk fast" (Fig. 5-10). In the current implementation, "walk faster" after "walk fast" cannot shift speed membership function much further right, it is already closer to "running" than to "walking". That's why the change in the membership function is very little, and the "substantial change" heuristics fails, classifying the phrase as unclear.

Non-Contradictive: Speed: true, Look: true, Heading: true, Directions: true
Non-Tautological: Speed: true, Look: true, Heading: true, Directions: true
SubstantialChange: Speed: false, Look: false, Heading: false, Directions: false
InformationGain: Speed: false, Look: false, Heading: false, Directions: false

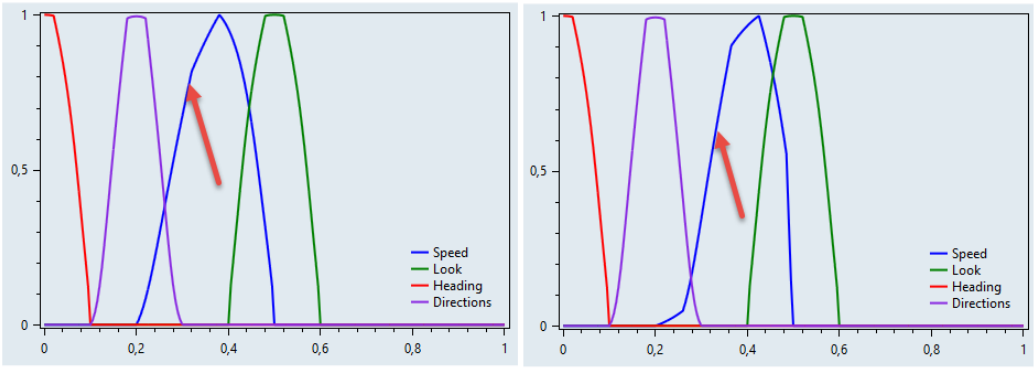

Fig. 5-10. "walk fast" (left) and "walk fast" followed by "walk faster" (right).

4. Command "look fast". Although the command is parsed and all the words used have semantic definitions, the phrase is rejected, because "look" and "fast" do not have compatible axes (parameter they act upon).

look fast

[(Command (VerbCmd ("look",Some (Definition (Quality (null, "fast"))))), [])]

PhraseParsed: true
PhraseAccepted: false
PhraseMeaningful: false
SearchRequest: Empty

Fig. 5-11. "look fast".

5. Command "look" after command "look right" (Fig. 5-12). Although, command "look" changes the membership function substantially, it is less specific, than "look right", and does not provide any new information, because none of the points of the new membership function decrease their truth degrees compared to the old function.

Non-Contradictive: Speed: true, Look: true, Heading: true, Directions: true
Non-Tautological: Speed: true, Look: true, Heading: true, Directions: true
SubstantialChange: Speed: false, Look: true, Heading: false, Directions: false
InformationGain: Speed: false, Look: false, Heading: false, Directions: false

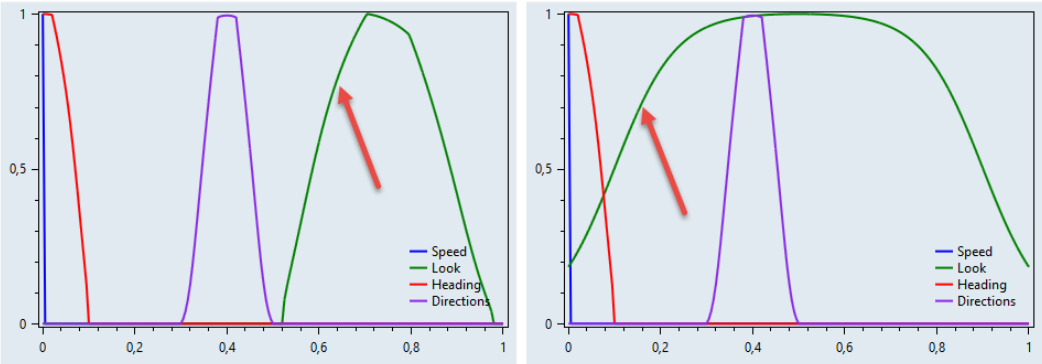

Fig. 5-12. "Look" after "look right"

6. Command "steer". This command results in the membership function that is equal to one in all its points (shown in red on Fig. 5-13). This is because the system does not know where exactly to steer, and this results in the fail of the heuristics checking for the lack of information (tautology).

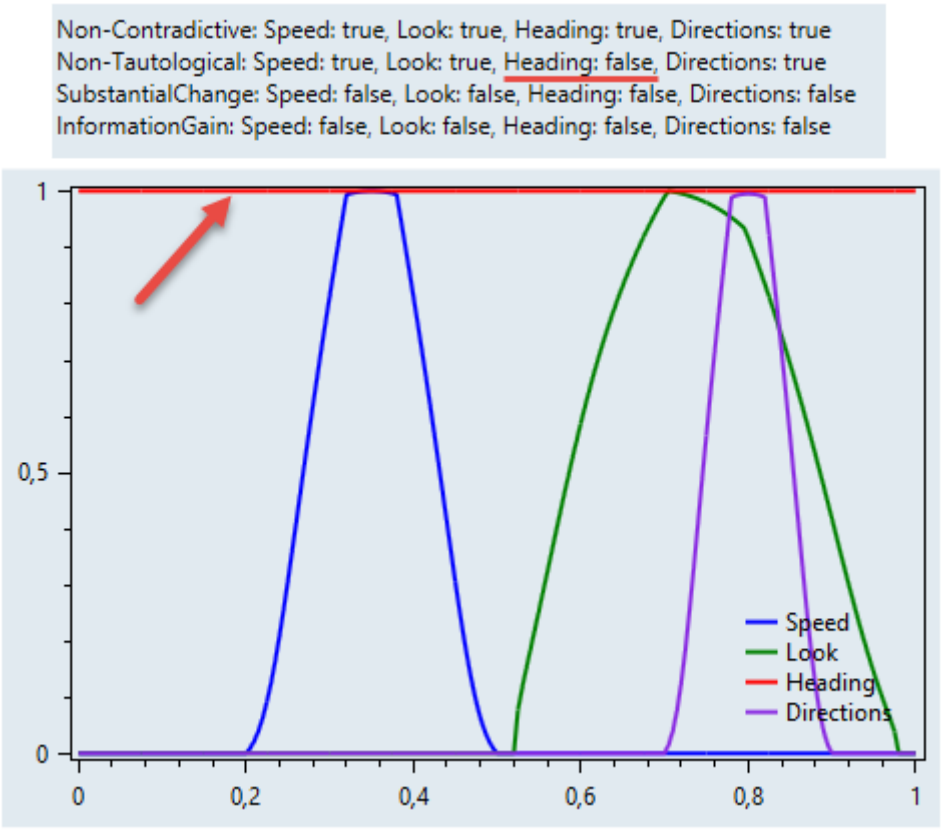

Fig. 5-13. Lack of information

5.5 Supported commands

Currently the system supports simple sentences[1].

This is the list of currently supported phrases.

- `crawl/walk/run [fast/slowly] [faster/slower]`
- `stop`
- `look/face [forward/back/right/left] [further right] [further left]`
- `steer [north/south/east/west/northeast/northwest/southeast/southwest]`
- `steer [further left/further right]`

[1] It did earlier support compound comma-separated phrases, but this functionality was removed as it doesn't pose any conceptual interest (at this stage of prototype development).

```
- faster/slower
- further left/further right
- i need/want coffee/food
```

5.6 Technical details

The software is written in F# (framework and core parts), C# (application parts), and JavaScript (Google Maps and Street View API integration). Please see Fig. 5-14 for details.

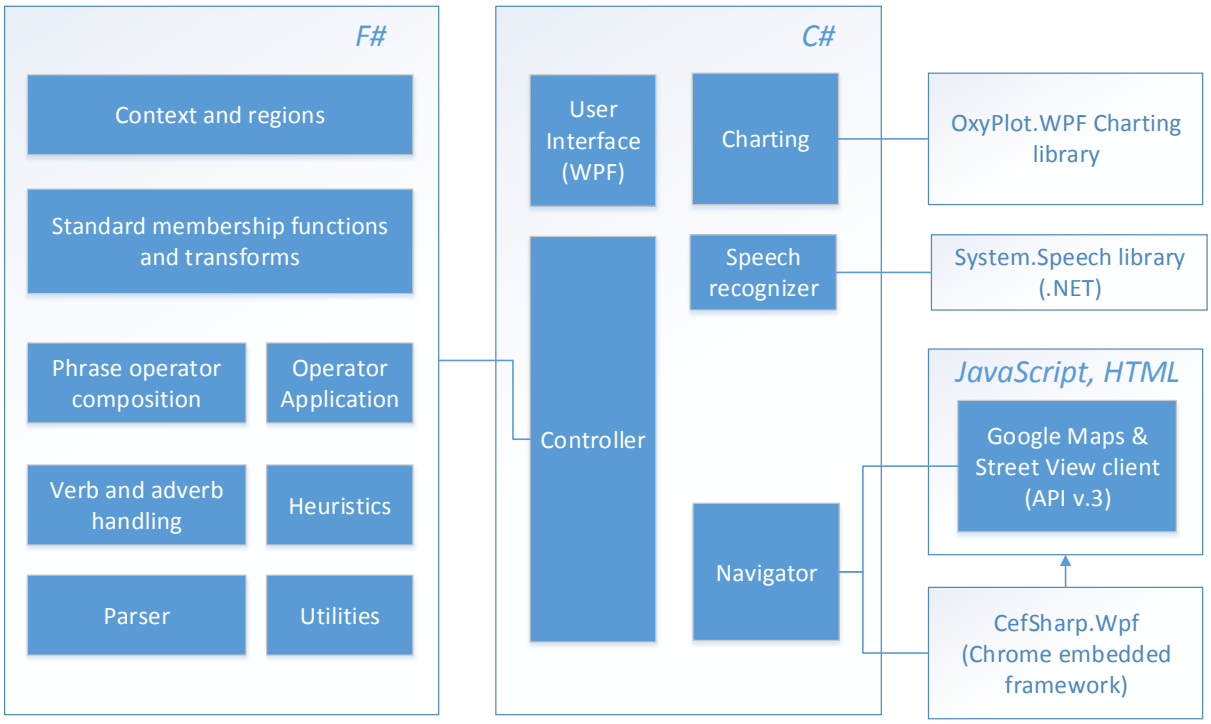

Fig. 5-14. Software structure. Framework and prototype code is in blue. Third party libraries are in white.

The choice of F# is motivated by its convenience of working with functions, and in this framework most of the operations are performed on membership functions, their transformations, tests, etc.

Testing of the project is done by unit tests, and "visual" tests that visualize the results of membership function transformation in F# interactive mode, without the need to run the program. A separate library (F# charting) is used for these "visual tests".

The parser used is hand-written, and similar to a very simple version of monadic combinatory libraries like Haskell's Parsec and F#'s FParsec. The motivation for creating the parser (as opposed to using an existing one) was the need to support ambiguous parsing (when each step may in principle yield several results). This features is implemented, but not yet used in the rest of the framework.

6 Results and future research

6.1 Results

This project attempted to accomplish three goals:

- Perform analysis of related work, with focus on different knowledge representations, and analyze their strengths and weaknesses when it comes to working with concepts on the level of their meanings.
- Design and develop natural language understanding framework, capable of interpreting and computationally analyzing the meaning of spatial directions expressed in natural language.
- Implement a research prototype of an intelligent navigation assistant with some functionality like in the one described in (2.2), illustrating capabilities of the framework.

6.1.1 Analysis of related work

Literature survey was conducted twice. Firstly, at the very start of the project. Secondly, close to the end of the project, to check for significant updates in the field.

Literature survey showed that knowledge representations common for the field are:

- either well suited for accommodation of new knowledge without the need for humans to directly encode it (e.g. semantic networks),
- or fairly detailed representation of knowledge, but with need for it to be manually edited by humans (e.g. logic-based approaches).

It is worth noting that neither semantic network-based or logic based approaches focus on the internal structure of the concepts (or attempt to quantitatively capture some kind of approximation of meaning of these concepts). Generally, there is a very small body of work directly related to representing and working with meaning of natural language constructs, and that the closest works were by L. Zadeh, starting from his Quantitative Fuzzy Semantics [5], and continuing until CWW [53, 54]. L. Zadeh's latest works allow to perform computations with words after the manual precisiation step [54:66-67]. However, before this step, no

quantitative work with natural language construct meanings can be done in CWW. This may not be of ultimate importance for computing with words, but is crucial for working with computational comprehension and understanding of the words and phrases.

6.1.2 Framework features

During the project, following features of the framework were developed:

- Work with contexts represented with a set of independent regions
- Work with one-dimensional regions, represented as an analytic membership function (or a combination of such functions)
- A set of basic analytic membership functions and their transformations
- A simple module for composing phrase operators, supporting some verbs and adverbs
- Operator application module
- Four general heuristics for comprehensibility analysis (4.4.2)
- Support for declarative word definitions
- Parser combinatory library with support for ambiguity
- Utilities for normalization, defuzzification, value conversion, etc.
- Various testing utilities

6.1.3 Prototype features

During the project, two testing modes for the intelligent navigation assistant prototype were developed: “fly-over” and “I am telling you that…”. These included implementation of the following features

- Declarative definitions of spatial direction words (verbs and adverbs)
- Phrase processing cycle, integrating with the framework
- Navigator module, interacting with maps
- Integration with Google Maps and Street View
- Integration with voice recognition engine

6.1.4 Analysis

The prototype software implemented is still very far from intelligent. The navigation assistant prototype cannot do any real assistance yet, except for previewing locations and finding the

restaurants nearby. As of now, the software can only handle very simple sentences, composed from select verbs and adverbs according to a restrictive grammar. No new words can be added without modifying code files.

On the other hand, the software demonstrates the very basics of the ability to computationally analyze comprehensibility of phrases. It is able to work with the quantitative approximation of meaning that is contained within word definitions. It can handle declarative definitions of new words.

6.2 Future research and development

Based on the analysis that is possible to perform at this stage, the framework and the prototype show interesting results. The author would like to (and sees no reason not to do it) call them promising.

Of course, to analyze how viable and feasible the approach is on a larger scale, requires much longer research, development and experimental results.

This sections mentions some of the short-term features of the framework and prototype that are not yet implemented, but planned.

1. Using interpolated functions defined by control points (instead of analytic functions) for defining words[1]. This allows to define new words faster, and without the need to modify the code.
2. Implementing support for nouns.
3. Implementing more heuristics for assessing comprehensibility, and adding support for comprehension level (instead of just a pass / fail flag).
4. Implementing nouns “corner”, “crossing”, “dead end” based on the property / axis “DirectionMultitude”
5. Implementing phrases “stop at next crossing”, “turn left/right” (at the first crossing).
6. Implementing weighing of axis importance and “activation” of verb axes by adverbs. This will make it possible to use verbs working with several axes without “conflicts”. In the

[1] This is partly implemented, but not part of this master project.

current implementation, saying “walk north” would reset the current speed; that’s why a separate verb “steer” is used.

7. Implementing description of problematic situations. For example, when the navigator is in fly-over mode and cannot find its way towards requested direction (encounters a dead-end, a roundabout etc.), the algorithm for describing problematic situations in words [1:21] may be used.

Speaking of the intelligent navigation assistant, implementing some of these features and adapting the software to run on a smartphone seems to be a realistic milestone. During its first versions, the application is most likely to be of limited use for the users. Though, even in the very beginning, it would be interesting to conduct a user experience survey to find out how they perceive a maps-like application that does not do anything before understanding the request, and occasionally says “your last phrase didn’t make any sense”.

7 Conclusions

This project attempted to make some humble steps in still much unexplored field of quantitative representation of meaning of natural language constructs, implementing a framework and software prototype based on the approach [1]. The results are very modest, but interesting. The approach and framework have some principle advantages over the widely used representations like e.g. semantic networks or logic-based approaches. On one hand, the developed framework defines concepts in a quantitative way, trying to capture their essence, approximation of their meaning. On the other hand, the fuzzy logic based representation used in the framework does not restrict the system from learning, or even generating [1:20] new concepts without the need for a person to manually codify this knowledge (unlike in e.g. logic-based representations). This gives systems based on the framework a potential ability to learn from sources meant for humans (like dictionaries), followed by confirmation of this knowledge using integration with systems like WordNet, ConceptNet, etc. For example, if WordNet is saying that two words are synonyms, their meaning-operators should yield similar transformations, and in this way the system can verify it has learned the concept correctly.

Learning from dictionaries and encyclopedia is a somewhat distant future. Currently both the approach and the framework are in a very early research stage, and a lot of work remains to be done before their feasibility can be evaluated. However, the author hopes that they have a future among other systems posing high demands to the level of language understanding, when it comes to both processing and communication.

References

1. M. Kapustin, P. Kapustin. “Modeling of the meaning: computational interpreting and understanding of natural language fragments”, arXiv:1505.08149 [cs.CL]. [Online]. Available: http://arxiv.org/abs/1505.08149
2. W. Woods. “Meaning and Links”, *AI Magazine*, 2007.
3. Wikipedia. “Natural language understanding”, *Wikipedia.org*. [Online]. Available: http://en.wikipedia.org/wiki/Natural_language_understanding
4. T. Winograd. “Understanding Natural Language”, *Cognitive Psychology*, 1972.
5. L. Zadeh. “Quantitative Fuzzy Semantics”, *Information Sciences*, 1971.
6. Wikipedia. “Eliza”, *Wikipedia.org*. [Online]. Available: http://en.wikipedia.org/wiki/ELIZA
7. D. Bobrow. “Natural Language Input for a Computer Problem Solving System”. In: Marvin Minsky, ed., *Semantic Information Processing,* The MIT Press, 2003.
8. T. Winograd. “A procedural model of language understanding”. In: *Computer models of thought and Language*, ed. R. Schank & K. Colby. San Francisco: W. H. Freeman Press, 1973.
9. H. Dreyfus. “What Computers Still Can't Do”. New York: *MIT Press*, 1992.
10. H. Simon. “Artificial Intelligence Systems that Understand”. *IJCAI'77 Proceedings of the 5th international joint conference on Artificial intelligence*, 1977
11. SHRDLU resurrection. [Online]. Available: http://www.semaphorecorp.com/misc/shrdlu.html
12. T. Winograd. “Procedures as a Representation for Data in a Computer Program for Understanding Natural Language”, *DTIC Document*, 1971.
13. D. Bobrow, T. Winograd. “An Overview of KRL, a Knowledge Representation Language”. *Cognitive Science*, 1977.
14. A. Norberg. “Oral history interview with Terry Allen Winograd”. Charles Babbage Institute, University of Minnesota, Minneapolis, 1991.
15. Wikipedia. “Commonsense knowledge (artificial intelligence)”, Wikipedia.org. [Online]. Available: http://en.wikipedia.org/wiki/Commonsense_knowledge_%28artificial_intelligence%29
16. S. Russel, P. Norvig. “Artificial Intelligence: A Modern Approach”. *Pearson*; 3 ed., 2009.

17. Wikipedia. “Semantic network”, Wikipedia.org. [Online]. Available: http://en.wikipedia.org/wiki/Semantic_network
18. ConceptNet. [Online]. Available: http://conceptnet5.media.mit.edu/
19. H Liu, P Singh. “ConceptNet – a practical commonsense reasoning tool-kit”. *BT technology journal*, 2004.
20. H. Liu, P. Singh. “Commonsense Reasoning in and over natural language”. *MIT Media Lab*, 2004.
21. R. Speer, C. Havasi, H. Lieberman. “AnalogySpace: Reducing the dimensionality of common sense knowledge”. *Proceedings of AAAI*, 2008.
22. W. Woods. “What's in a Link: Foundations for Semantic Networks”. *DTIC Document*, 1975.
23. R. Brachman. “What IS-A is and isn't: An analysis of taxonomic links in semantic networks”. *Computer*, 1983.
24. R. Brachman. “An Overview of the KL-ONE Knowledge Representation System”. *Cognitive Science*, 1985.
25. Wikipedia. “KL-One”, Wikipedia.org. [Online]. Available: http://en.wikipedia.org/wiki/KL-ONE
26. Wikipedia. Description logic, Wikipedia.org. [Online]. Available: http://en.wikipedia.org/wiki/Description_logic
27. Description Logic Reasoners. [Online]. Available: http://www.cs.man.ac.uk/~sattler/reasoners.html.
28. J. McCarty. “Programs with common sense”. *Semantic Information Processing*, 1959.
29. Wikipedia. “Cyc”, Wikipedia.org. [Online]. Available: http://en.wikipedia.org/wiki/Cyc
30. Cyc. http://www.cyc.com
31. E. Bertino, G. Piero, B. Zarria. “Intelligent Database Systems”. *Addison-Wesley Professional*, 2001.
32. J. Pustejovsky. “The generative lexicon”. *Computational Linguistics*, 1991.
33. J. Pustejovsky, B. Boguraev. “Lexical knowledge representation and natural language processing”. *Artificial Intelligence*, 1993.
34. L. Duan, Z. Zhou, Y. Qiu. “The Fuzzy Semantic Relations in Semantic Link Network”. *Education Technology and Computer Science, 2010 Second International Workshop on*, 2010.

35. S. Hölldobler, T. Khang, H. Störr. “A Fuzzy Description Logic with Hedges as Concept Modifiers”. In: *Proceedings InTech/VJFuzzy’2002*, 2002.
36. G. Qi, J. Pan, and Q. Ji. “A Possibilistic Extension of Description Logics”. *In: Proceedings of the 20th International Workshop on Description Logics DL’07*, 2007.
37. G. Stoilos, G. Stamou, V.Tzouvaras, J. Pan and I. Horrocks. “Fuzzy OWL: Uncertainty and the Semantic Web”. *Proc. Of the inter. Work on OWL-ED05*, 2005.
38. U. Straccia. “A Fuzzy Description Logic”. *In: Proceedings of AAAI-98,* 1998.
39. T. Kollar, S. Tellex, D. Roy, N. Roy. “Toward understanding natural language directions”. *International Conference on Human-Robot Interaction (HRI)*, 2010 5th ACM/IEEE.
40. S. Tellex, D. Roy. “Grounding Spatial Prepositions for Video Search”. *Proceedings of the 2009 international conference on Multimodal interfaces*, 2009.
41. S. Hemachandra, F. Duvallet, T. Howard, N. Roy, A. Stentz, M. Walter. “Learning Models for Following Natural Language Directions in Unknown Environments”. arXiv:1503.05079 [cs.RO], 2015. [Online]. Available: http://arxiv.org/abs/1503.05079
42. L. Zadeh. “A Fuzzy-Set-Theoretic Interpretation of Linguistic Hedges”. *Journal of Cybernetics*, 1972
43. G. Lakoff. Hedges: “A Study in Meaning Criteria and the Logic of Fuzzy Concepts”. *Journal of philosophical logic*, 1973.
44. L. Zadeh. “The concept of a linguistic variable and its application to approximate reasoning”. *Information Sciences*, 1975.
45. L. Zadeh. “PRUF—a meaning representation language for natural languages”. *International Journal of Man-Machine Studies*, 1978.
46. L. Zadeh. “A computational approach to fuzzy quantifiers in natural languages”. *Computers & Mathematics with Applications*, 1983.
47. L. Zadeh. “Test-Score Semantics as a Basis for a Computational Approach to the Representation of Meaning”. *Literary and Linguistic Computing*, 1986
48. L. Zadeh. “Knowledge representation in fuzzy logic”. *IEEE Trans. on Knowledge and Data Engineering*, 1989
49. L. Zadeh. “Fuzzy logic = computing with words”, *IEEE Transactions on Fuzzy Systems*, 1996.
50. L. Zadeh. “Precisiated Natural Language (PNL)”. *AI Magazine*, 2004.

51. L. Zadeh. “From imprecise to granular probabilities”. *Fuzzy Sets and Systems*, Elsevier, 2005.
52. L. Zadeh. “My life and work – a retrospective view”. *Applied and Computational Mathematics*, 2011.
53. L. Zadeh. “Computing with Words—Principal Concepts and Ideas”, *Studies in Fuzziness and Soft Computing*, 2012.
54. L. Zadeh. “Computing with Words—Principal Concepts and Ideas”. Lecture Notes. [Online]. Available: http://www.cs.berkeley.edu/~zadeh/presentations%202010/CW--Principal%20Concepts%20and%20Ideas-updated%20Jan%2021%202011.pdf
55. Wikipedia. “Membership function (mathematics)”, Wikipedia.org. [Online]. Available: http://en.wikipedia.org/wiki/Membership_function_%28mathematics%29
56. L. Zadeh. “Fuzzy sets and systems”, *Proc. Symp. on System Theory, Polytechnic Institute of Brooklyn*, New York, 1965.