Theoretical foundations of natural language programming and publishing for intelligent agents and robots

Sandor M Veres

***Abstract*— This paper is an application of ontology theory, conceptual graph and programming languages theories to develop the foundations of natural language programming (NLP) that can be used to produce natural language documents for intelligent agents and human readers. The analysis given reveals three benefits of NLP. First, it is a "conceptualized" programming that enables developers to write less bug prone programs due to clarity of code presentation and enforced structuring of data. Secondly, NLP can aid programming of the all important abstractions for robots: event, action and world model abstractions can be created by sentences. Thirdly, NLP can be used to publish natural language documents by researchers, e.g. English language documents, on the Internet or also in print that is actually code. The paper introduces a large class of intelligent agents that can read such documents. This enables human users and agents a shared understanding of how application systems work.**

***Index Terms*— Computer programming, intelligent physical agents, robots, conceptual graphs, machine knowledge.**

I. INTRODUCTION

The paper presents the theoretical background to an information processing system [1] for autonomously operating robotic agents where engineers can use NLP (natural language programming) documents to define their behaviour and later the agents can read new published documents to acquire knowledge such as various physical skills, facts about the environment and also behaviour policies, goal priorities. The NLP documents concerned are similar to engineering booklets with contents, sections, subsections in English, that can appear in HTML, LaTex(pdf) formats (see numerous examples at www.system-english.com).

Use of NLP documents shortens the route to robotic software update. There is no need for an application engineer to read a journal publication and implement it by programming. The research engineers methodological work is directly utilized by the autonomous vehicle or robot reading NLP documents. Engineers whom are different from the author of the publication can also read the papers and learn the details of how the vehicle operates, how decisions are reached and how skills are performed. Not only will users of the autonomous vehicle clearly understand how it operates but will also know its limitations to avoid misuse or misunderstanding. Users can modify English sentences in the publication to influence the vehicles behaviour or how its skills are performed. Also the legal responsibility for misuse can be shifted to the user.

This work has been supported by EPSRC Grant No. EP/E02677X/1

S M Veres are with the School of Engineering Sciences, Southampton, Highfield, SO17 1BJ, UK, Email: s.m.veres@soton.ac.uk

The process of knowledge distribution among scientists and engineers is by way of publishing at conferences, in journals and books. They describe in technical papers how a control method works, how a navigation method proposed by an author works. Also information processing and decision making procedures are published by research engineers (REs) [1]. Other engineers, lets call them application development engineers (ADEs), read these technical papers, conceptualize the meanings and may opt to implement the methods in practice. There are thousands of control and signal processing engineering papers written every year worldwide. Their impact on industry is patchy as many of the papers are not read or chosen to be ignored due to financial necessities of companies. Knowledge transfer to industry (KTI) is slow as is well known. Cynics may even ask: why do we need to speed up KTI?... we are fine as we are. Others may say: this is a topic for our knowledge engineers ...but artificial intelligence is far from being able to help yet. Reference [2] describes principles and reviews systems developed to support software developers, that also act as mediating tools to provide a platform facilitating knowledge sharing. The need of supporting knowledge collaboration in software development environments, is highlighted in [2], based on the conceptualization of software development as knowledge-intensive and distributed cognitive activity. This principle is extended in [3] for the context of shared knowledge and collaboration between human users, i.e. scientists, engineers, and machines or intelligent agents. It is proposed that documents written in English are distributed to networked devices that human users can also read and hence develop shared understanding with the devices. Hence a system is created that allows humans and machines to communicate intuitively (i.e., by using human language) through intelligent cooperation. This is achieved by creating a coherent meaning-system understandable by both the devices and people. Is it however possible to make a machine to read a technical book written in English sentences to formulate methodologies at the conceptual level? Some would say: natural language interfaces have been developed and you can use various controlled English languages. These need a priori knowledge and definition of meanings .. and you need to know the grammar and what phrases to use .... the problem is a complexity nightmare! And yet this paper is describing an existing system whereby engineers can write documents in English that can also contain equations, figures and images, quotes, numbers, physical quantities, etc. that suitable intelligent agents can read and apply it in their daily work [1,3]. This paper describes a system with the following practical features: (1) The authors and readers of these machine readable documents (called system English papers, for short called sEnglish papers) do not need to learn grammar, just to apply common sense. The most important thing for an author is to have conceptual clarity in their area and express that in a suitable style: writing skills are needed. It only takes a few hours to be trained as an author. (2) The autonomous systems can read these special engineering publications in sEnglish. sEnglish publications are of special format with contents, sections and subsections. They may contain images, mathematical formulae to support understanding both to humans and to autonomous systems who read them. (3) The autonomous systems who read these documents do not need access to a central dictionary of meanings. There is no need for building up an infrastructure so that autonomous systems can read sEnglish books. The system can be used immediately, without any investment. Autonomous vehicles (AUVs, UAVs, AGVs and spacecraft), robots, toys can be built that understand natural language documents without a supporting infrastructure to be set up first. We hold the view that software downloads for intelligent autonomous systems are old-fashioned and outdated now. Autonomous systems and robots should read books on mental and physical skills and facts about the world from documents that their user can also read with ease and hence they can have a shared understanding of a selected part of the world.

In this publication system the manufacturer of a vehicle platform or robot can invite publications from research engineers to enrich the knowledge base of agents controlling these platforms in a format that also merits current standards of scientific and engineering publications. This could for instance be the case for the use of autonomous vehicles, robotic pets and intelligent toys, gardening robots or agricultural unmanned autonomous aerial vehicles. Publications for autonomous systems is a logical next step of information systems in a historical perspective [3]: (a) There was first verbal communication between people. (b) Writing was developed to record events and knowledge. (c) Later book was invented, i.e. multiple copies of written material were printed that masses of people could read. (d) Journal and newspaper publishing and the WWW exists to distribute knowledge. In the computer science community the idea of natural language programming has been largely considered impossible and impractical due to ambiguity. This paper reports about a complete system for autonomous systems unambiguously understanding human readable books to enhance their skills and knowledge about the world. There is a complete software system available today for NLP in sEnglish ('system English') An complete authoring tool (sEAT) [2] and sEnglish reader agent installation system (sERA) [3] is available. When installed on a PC, the reader agent sERA can read web pages that are sEnglish documents and execute sentences and complex actions sequences as instructed by a human instructor of the reader agent. sERA is simple reactive agent (sERA) that understands these papers and can execute commands in sentences.

An important part of NLP documents is their definition of an ontology that is the bases of statements in sentences. As contributive factors to the development and study of ontologies for knowledge sharing and reuse are reviewed in [4] . The following definition is generally accepted by the research community: "Ontologies are explicit formal specification of a shared conceptualization" [5], [6] where: "explicit" means that the type of concepts used, and the constraints on their use are explicitly defined [7]; "formal" means that the ontology should be machine readable; "shared" reflects the notion that an ontology captures consensual knowledge, that is, it is not private to some individual, but accepted by a group. Conceptualization emphasizes the abstract model of some phenomenon in the world by having identified the relevant concept of that phenomenon" [7]. Conceptualization refers to the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them [8]; i.e. addresses the abstract, simplified view of the world that is required for representation [9].

Reference [4] describes the hierarchical abstraction layers of programming and points out that block diagram based abstractions of programs are ill placed, they are two low resolution. Eventually the author advocates the development of "Developmental Mathematics" which can be a complex process. In this paper we neatly fit NLP into the abstractions layers of programming. As shown above NLP provides simultaneous human (through the use of natural language) as well as machine interpretation (translated code) of an NLP sentence. Note that code unambiguity (of NLP sentences) preserves the cornerstone of digital computing: the usefulness of digital computation is its determinism, i.e. its reliability in banking, accounting and computer control of aircraft, machines and robots, etc.

This paper is organized as follows. The next section provides definitions of ontologies. This is followed by a model of human thinking in terms of conceptual graphs. Section IV introduces the NLP document and Section V provides examples. Section VI briefly discusses levels of abstractions for sentences. Section VII is concerned with procedural knowledge sharing among a set of agents. Shared knowledge description is concluded with the involvement of the human operator of the agents that is followed by conclusions.

II. Fundamentals

A natural language program document (NLPD) consists of three components: an ontology, a set of sentence templates and an underlying instructional (imperative) programming language. The process of creating and making NLPDs is described in a flow diagram in Figure 1. An expert author of an algorithmic field (for instance in subfields of signal processing, control, science and logic inference) produces an ontology and provides hierarchical definitions of sentence meanings using a programming language for low abstraction level semantics. The sentence meanings, ontology definitions and meta sentences to define the subject area composed into an HMTL document by a document creator algorithm.

Some of the web pages of this document are illustrated in Figures 2,3. The NLPD is utilized either by engineers who can read the document and borrow from it algorithm for their own work, or by an intelligent agent, typically controlling a vehicle, machine, appliance or a robot in general. Machines reading publications directly can shorten the route how results are made useful in intelligent machines - there are no upgrades needed by engineers. The agents seek out their knowledge themselves. This section will first provide formal definitions for the ontologies used and also for the relationship with the underlying programming language. Next formal definitions of human interpretations are given in terms of conceptual graphs. Finally NLP documents are defined and illustrated.

First a formal definition of a traditional ontology is provided here that is short of using the description logic (DL,OWL, see W3C.org) framework as that will not strictly be required for natural language programming.

Definition 2.1: An *agent's modelling ontology* is a description logic $M_{DL} = \{\Gamma, R, =, , I, \cup, \cap, \neg, \exists, \forall\}$ with atomic classes Γ, roles R so that satisfy the following axioms:

(a) Subclass relationship $A \supset B$ is defined by $A \cup B = B$.
(b) There are atomic classes *perceived object, imagined object*, *mathematical object* that are disjoint and satisfy *modelling object = (perceived object* $\cup$ *imagined object* $\cup$ *mathematical object)*
(c) There is an atomic role *has attribute*.

In NLP the individuals created from classes of O will become examples of human concepts as represented on a digital computer. To achieve this the O will be linked to some (popular) imperative programming language: the ontology O is required to have a subset of classes that are classes in the programming language and any class in O are subclasses of some basic types (classes) in the programming language. This will be formally described as follows. First we restrict the definition of an ontology that NLP will use.

Definition 2.2: A *restricted ontology* $O = \langle \Gamma \,|\prec|\, @ |\Lambda\rangle$ consists of a lattice $\langle \Gamma \,|\prec\rangle$ over the class set Γ, an attribute label set Λ and an attribute mapping $@ : \Gamma \to 2^{\Gamma(\Lambda)}$ where $\Gamma(\Lambda) : \Lambda \to \Gamma$ is a mapping from the attribute label set to the class set Γ . The class set has a universal sup class ξ_0 that is the *modelling object* so that $\forall \xi \in \Gamma$: $\xi \prec \xi_0$ and a *universal sub model* class ξ_∞ such that $\forall \xi \in \Gamma$: $\xi_\infty \prec \xi$. The @ is also required to satisfy the inheritance condition $\forall \xi, \zeta \in \Gamma$: $\xi \prec \zeta \;\Rightarrow\; @(\xi) \supseteq @(\zeta)$.

Let $N = \langle B, L, D, H, P\rangle$ be an instructional (imperative) programming language (can be either object oriented or not) that has a set of basic types B, a set L of syntactically correct sets of instructions, declarations D, subroutines H and programs P. The ontology O is required to be such that any class of O is a subclass of a type (class) in B, i.e. of a basic type in N, as given in the following definition.

Definition 2.3: Let $N = \langle B_N, L_N, D_N, H_N, P_N\rangle$ be an instructional computer language that has a set of variable types B_N. An ontology $O = \langle \Gamma | \prec | @ | \Lambda\rangle$ is said to be *supported by* N if $B_N \subset \Gamma$ and $\forall \xi \in \Gamma\; \exists \zeta \in B_N : \xi \prec \zeta$.

An ontology O supported by a programming language N

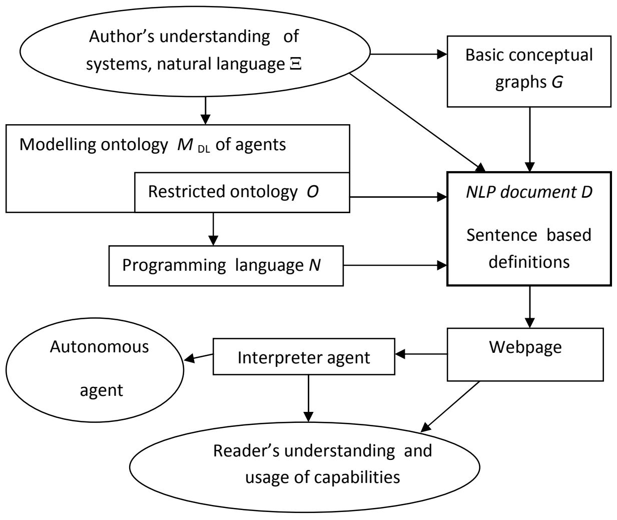

Author's understanding of systems, natural language Ξ
Basic conceptual graphs G
Modelling ontology M DL of agents
Restricted ontology O
NLP document D
Sentence based definitions
Programming language N
Autonomous agent
Interpreter agent
Webpage
Reader's understanding and usage of capabilities

Fig. 1. Flow diagram of NLP document creation and usage.

will often be indexed as O_N to express this relationship. For instance if the underlying programming language is MATLAB, we will use the notation O_M, if it is C or C++, then we can use O_c or O_{cpp}, respectively. Next a possible relationship of an ontology O_N to conceptual graphs (CG) will be described.

III. Conceptual graphs of human thinking

Formalizing human thinking will not be attempted here, CGs will be used instead to represent human thought as it is generally accepted [10], [5] that CGs are powerful representations of human thought. For completeness a simple and general enough definition of conceptual graphs will be provided here.

Definition 3.1: Let C and R be sets of classes from two disjoint ontologies O_c and O_r respectively. A *basic conceptual graph* $G = \langle G_v, G_e, L_v\rangle$ is a vertex-labelled directed-bigraph that satisfies the following conditions:

(a) the vertices G_v are decomposed into two sets of *concepts* C and *conceptual relations* R so that $G_v = C \cup R$;
(b) the directed edges of G occur in pairs so that $G_e \subset C \times R \times C$.
(c) Vertices G_v are labelled by a *referent* function L_v : $G_v \to R$ where R contains of existential, individual, set, universal and question labels as listed in Table 1.

The set of basic conceptual graphs is denoted by $B_{cg}(O_c, O_r)$.

Some basic CGs are based on perception of the surrounding physical world and are direct abstractions from perception processes of an agent. Other basic CGs can be obtained through communication by other agents or by operations on conceptual graphs the agent has in memory. Classical work [10] on CG defined four basic operations on conceptual graphs to obtain new ones: (1) *copying*, i.e. creating and identical copy; (2) *restricting* a CG where concepts are replaced by conceptual subclasses or by individual

Referent	Formats	Examples
Existential	[*con*: *]	DOG:*
unique ex.	[*con*: @1]	DOG:@1
anaphoric	[*con*: #]	DOG:#
Individual	[*con*: # *num*]	DOG:453
	[*con*: *name*]	DOG:Vak
Set	[*con*: $\{i_1,..,i_n\}$]	DOG:{Vak,Spot}
generic	[*con*: {*}]	DOG:{*}
counted set	[*con*: {*}@*num*]	DOG:{*}@4
anaph. set	[*con*: {*}#]	DOG:{*}#
Universal	[*con*: ∀]	DOG:∀
negative	[*con*: ∀¬]	DOG:∀¬
plural set	[*con*: {*}∀]	DOG:{*}∀
fuzzy set	[*con*: {*}@many]	DOG:{*}@many
Question	[*con*: ?]	DOG:?
plural	[*con*: {*}?]	DOG:{*}

TABLE I

DEFINITIONS OF REFERENT TYPES IN R. *con* STANDS FOR *concept* THAT IS AN ENTRY FROM O_v AND *num* STANDS FOR *number* THAT IS A NONNEGATIVE INTEGER. THE L_v MAPS A *concept* IN O_v TO ON OF THE LABELS IN COLUMN 2 OF THIS TABLE.

markers; (3) *joining* where identical concepts of individuals are used in two graphs to create a third one that shares the individuals and keeps all other relations from the two graphs. (4) *simplifying*, that means removing duplicate conceptual relations from a graph.

Basic conceptual graphs are timeless and do not contain references to the context how the “thought” was generated, e.g. communication, belief, desire or whether it occurred in the past or will be valid in the future. These modalities are clearly important for human and also for artificial agents such as the belief-desire-intention agents. Context boxes [10], [5] are used to represent negation, modality and propositional attitudes like “know” or “believe” and indirect speech. Conceptual graphs have been extended by agent’s perspectives and temporal localization [6] where agent’s perspectives and temporal localizations are also described. Using these references we provide a sufficiently general definition of compound conceptual graphs (CCGs) to provide background for the concepts of natural language programming.

Definition 3.2: Let O_c and O_r be two disjoint ontologies for concepts and conceptual relations with associated basic conceptual graphs $B_{cg}(O_c,O_r)$. The set of compound conceptual graphs $C_{cg}(O_c,O_r)$ is the smallest set that satisfies the following conditions:

(a) $B_{cg}(O_c,O_r) \subseteq C_{cg}(O_c,O_r)$.

(b) Let T_l be a set of temporal labels. For any $x \in T_l,\ g \in C_{cg}(O_c,O_r)$ also $T(x,g) \in C_{cg}(O_c,O_r)$.

(c) Let T_r be a set of temporal relation labels on the real time-line. For any $x \in T_r,\ G,H \in C_{cg}(O_c,O_r)$ also $T(G,x,H) \in C_{cg}(O_c,O_r)$.

(d) Let A_l be a set of agent aspect labels. For any $x \in A_l,\ G \in C_{cg}(O_c,O_r)$ also $A(x,G) \in C_{cg}(O_c,O_r)$. The set of A-enveloped aspect graphs will be denoted by $\mathbf{A}(O_c,O_r) \subset C_{cg}(O_c,O_r)$.

(e) Let $G \in C_{cg}(O_c,O_r)$ and c a concept in G. The compound graph G' obtained from G by replacing all occurrences of c by a $A \in \mathbf{A}(O_c,O_r)$ is also a compound graph: $G' \in C_{cg}(O_c,O_r)$.

Large parts of human knowledge can be modelled by sets of conceptual graphs (CGs) that can also have modalities of time, perception and communication, etc.

Here we will aim to formalise human thought processes but make the following assumption.

Assumption 3.1: Human knowledge base $H_{KB} = \{G_i \mid i \in J\}$ about the present and the past of the world is represented by a time varying set of conceptual graphs with abstraction modalities of time, intention, desirability, likelihood and communication.

Now we come to the definition of a NLP text that are related to CGs.

IV. NLP DOCUMENTS

In the following an NLP *text* is understood as a sequence of ASCII characters. A *word* is a sequence of characters without space. A *proper name* is a word starting with a capital letter but not all words starting with capital letters are proper names, for instance all sentences start with capital letters.

Definition 4.1: Let Ξ be a natural language with word set Ξ_w. A *word* is a sequence of characters without space. A *proper name* is a single word that starts with a capital letter. An NLP *class name* is a finite sequence of words from Ξ_w. An *attribute name* is a finite sequence of words from Ξ_w.

1) An ontology O is called Ξ compliant if all of its class names and attributes are finite sequences from Ξ_w.
2) An *NLP sentence* is a sequence of words that starts with an initial capital and ends with ., ! or ?. A sentence may contain proper names, class names and attribute names as sub-sequences of the sentence. The set of all feasible, but not necessarily plausible, sentences is named $\overline{S}(\Xi,O,S)$.
3) An *NLP text* is a finite sequence of NLP sentences. The set of all feasible, but not necessarily plausible, sentences is named $\overline{X}(\Xi,O,S)$.

This definition with minimalist conditions sets constrains to form words and on sequences of words to form sentences and NLP text. The semantics of NLP meanings is however defined in terms of a broader data set to reflect context sensitivity of meanings. Semantics will be defined in terms of procedures run on a digital computer with input-output devices such as sensors and actuators.

To secure conceptual graph correspondence for each sentence $s \in S$ defined by m, we also need a conceptual description function as defined.

Definition 4.2: For each $s \in S$ the *conceptual description* $d(s) \in \overline{X}(\Xi,O,S)$ is a formalized NLP text description.

While we are not going into the details of an algorithm that parses NLP sentences to obtain conceptual graphs, we are making the following assumption.

Assumption 4.1: There is conceptual resolvent function $F_{CG} : \overline{X}(\Xi,O,S) \to B_{cg}(O_c,O_r)$ that uniquely identifies a $g(s) = F_{CG}(d(s),s) \in B_{cg}(O_c,O_r)$ in terms of a CG with referenced vertices belonging to conceptual classes in O and conceptual relations in O_r as permitted by $B_{cg}(O_c, O_r)$ of $\in H_{KB}$.

For any set S the notation S^* will be used for the set of its sub-sequences.

Definition 4.3: Let N be a computer programming language with syntactically correct code set L_N and let O be an ontology compliant with natural language Ξ. A *meaning function* $m : S \to S^* \cup L_N \cup \{\emptyset\}$ maps any sentence to a sequence of sentences or to a code in L_N or to the empty set to express that there is no meaning . An *interpreter* is a tuple $I = \langle S, m, L_N \rangle$. An interpreter I is called *semantically complete* if there is a *translator* $T : S \to L_N$ so that

1) For all $s \in S$ the $m(s) \neq \emptyset$.
2) $T(m(s)) = concatenation(T(s_1),\ T(s_2),\ \ldots\ T(s_k))$ if $m(s) = \{s_1,\ s_2,\ \ldots\ s_k\}$

Practically useful are sets of semantically complete sentences that use some ontology that is compliant with a natural language. As meaning of natural language sentences is context dependent and can be ambiguous, a most desirable property for sentence meaning is to retain the most important characteristics of *von Neumann architectures* that is determinism of code meanings. This is the property that makes today's computers reliable for bank transactions or airplane ticket search and reservation systems, etc. Determinism is however not defined globally, that would be untenable in the global world of knowledge, but locally in the context of a single NLP document.

Definition 4.4: NLP *document* is a tuple $D = \langle \Xi, O, S, m, d, L_N, N \rangle$ of natural language Ξ, ontology O, sentence set S, meaning function m, conceptualization descriptions d and syntactically correct codes L_N (in terms of programming language N), if the following are satisfied:

- O and S are Ξ compliant.
- The interpreter $I = \langle S, m, L_N \rangle$ is semantically complete.
- There is a unique translator T such that $T(s) \in L_N, \forall s \in S$ and all $s \in S$ are acceptable in Ξ.

The property of unique compilation is to preserve the main benefit of digital computers in that they are deterministic.

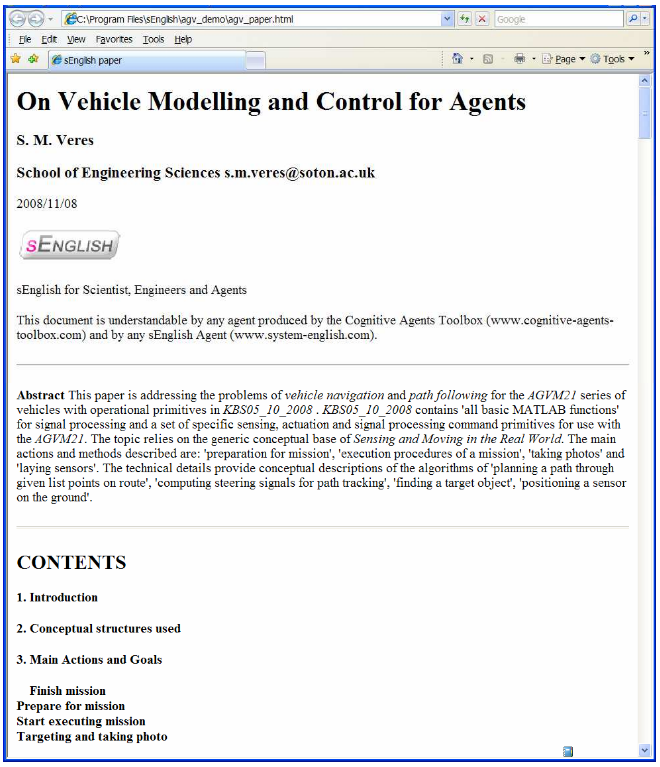
V. Examples of NLP documents

Figures 2,3 illustrate the web pages of an NLP document, sentences and document format for modelling of the environment and goals. Figure 4 shows the graphical user interface of the sEnglish authoring tools. The reader can find and download a number of NLP documents from www.system-english.com .

VI. Levels of abstractions

Assume $\Xi_p = \{O, S\}$ is a conceptual program. Any $s \in S$ has an associated set of subroutine calls denoted by $H(s) = \{h_i \in L_N,\ i = 1, \ldots, n_s\}$. Each h_i has a set of input object classes corresponding to ontology classes in Γ of O. An individual object under a class $\gamma \in \Gamma$ is simply a structure with field names $\{\lambda \mid \exists \chi \in \Gamma : \langle \lambda, \chi \rangle \in @(\gamma)\}$.

An $s \in S$ within a $\Xi_p = \{O, S\}$ is called to have an *elementary meaning* if $m(s) \in L_N$ and it is called to have


C:\Program Files\sEnglish\agv_demo\agv_paper.html
Google
File Edit View Favorites Tools Help
sEnglish paper
Page
Tools

On Vehicle Modelling and Control for Agents

S. M. Veres

School of Engineering Sciences s.m.veres@soton.ac.uk

2008/11/08

sENGLISH

sEnglish for Scientist, Engineers and Agents

This document is understandable by any agent produced by the Cognitive Agents Toolbox (www.cognitive-agents-toolbox.com) and by any sEnglish Agent (www.system-english.com).

Abstract This paper is addressing the problems of *vehicle navigation* and *path following* for the *AGVM21* series of vehicles with operational primitives in *KBS05_10_2008* . *KBS05_10_2008* contains 'all basic MATLAB functions' for signal processing and a set of specific sensing, actuation and signal processing command primitives for use with the *AGVM21*. The topic relies on the generic conceptual base of *Sensing and Moving in the Real World*. The main actions and methods described are: 'preparation for mission', 'execution procedures of a mission', 'taking photos' and 'laying sensors'. The technical details provide conceptual descriptions of the algorithms of 'planning a path through given list points on route', 'computing steering signals for path tracking', 'finding a target object', 'positioning a sensor on the ground'.

CONTENTS

1. Introduction

2. Conceptual structures used

3. Main Actions and Goals

Finish mission
Prepare for mission
Start executing mission
Targeting and taking photo

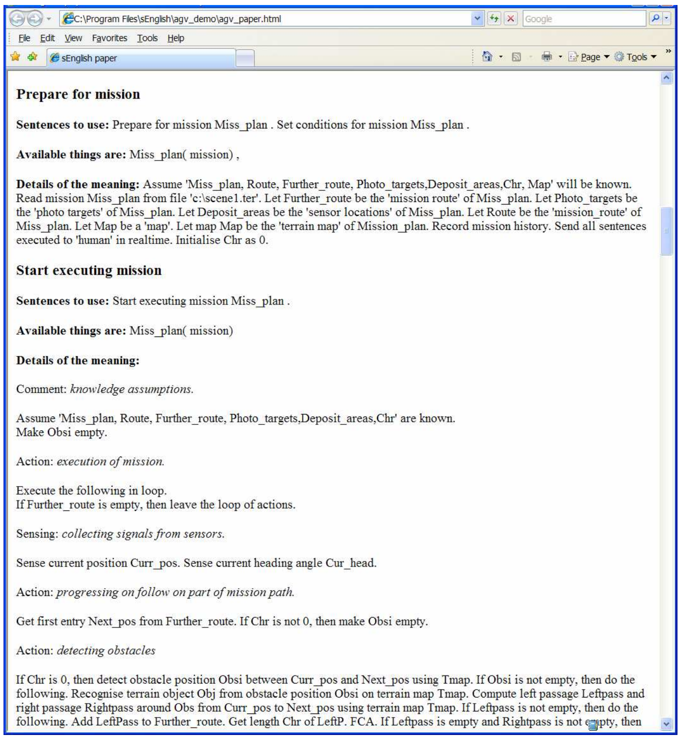
Fig. 2. Title page of an NLP document in "system English" that is short for sEnglish. See numerous examples of machine generated sEnglish documents at www.system-english.com.

an *explained meaning* if $m(s) = \{s_1,\ s_2,\ \ldots\ s_k\},\ s_i \in S, i = 1, \ldots, k$.

Definition 6.1: A sentence $s \in S$, that has elementary meaning only, is called *zero-level abstraction D-abstract ion*. An $s \in S$ is called to be of a *level k D-abstraction* if k is the smallest positive integer such that there is a sequence of $s_i \in S, i = 0, 1, \ldots, k-1$ such that $m(s_0) \in L_N, s_i \in m(s_{i+1}), i = 1, \ldots, k-1,\ s_k = s$.

Zero level D-abstractions are also called *subconscientious* abstractions. The term "D-abstraction" is short for "deterministic abstractions". Although D-abstractions are unlike human abstractions in flexibility, they are abstractions as they abstract away the details into their developed meaning in terms of other sentences and eventually code in L_N. Note however that even zero-level abstractions abstract away from the individuals the meaning refers to as available objects (inputs): a zero level level sentence $s \in S$ is defined by the set of classes of its outputs and inputs in the corresponding calls $H(s) = \{h_i \in L_N,\ i = 1, \ldots, n_s\}$. Hence zero level abstractions are defined by classes and not individuals.

Any D-abstraction at any level is well defined through a series of meaning definitions and hence deterministic. This property is a bit unlike human concepts where definitions are not as well determined. The connections between abstractions in an NLP $\Xi_p = \{O, S\}$ and between human concepts, in terms of human ontologies $\langle O_c, O_r \rangle$, are best defined via the


C:\Program Files\sEnglish\agv_demo\agv_paper.html
Google
File Edit View Favorites Tools Help
sEnglish paper
Page Tools

Prepare for mission

Sentences to use: Prepare for mission Miss_plan . Set conditions for mission Miss_plan .

Available things are: Miss_plan(mission) ,

Details of the meaning: Assume 'Miss_plan, Route, Further_route, Photo_targets,Deposit_areas,Chr, Map' will be known. Read mission Miss_plan from file 'c:\scene1.ter'. Let Further_route be the 'mission route' of Miss_plan. Let Photo_targets be the 'photo targets' of Miss_plan. Let Deposit_areas be the 'sensor locations' of Miss_plan. Let Route be the 'mission_route' of Miss_plan. Let Map be a 'map'. Let map Map be the 'terrain map' of Mission_plan. Record mission history. Send all sentences executed to 'human' in realtime. Initialise Chr as 0.

Start executing mission

Sentences to use: Start executing mission Miss_plan .

Available things are: Miss_plan(mission)

Details of the meaning:

Comment: *knowledge assumptions.*

Assume 'Miss_plan, Route, Further_route, Photo_targets,Deposit_areas,Chr' are known.
Make Obsi empty.

Action: *execution of mission.*

Execute the following in loop.
If Further_route is empty, then leave the loop of actions.

Sensing: *collecting signals from sensors.*

Sense current position Curr_pos. Sense current heading angle Cur_head.

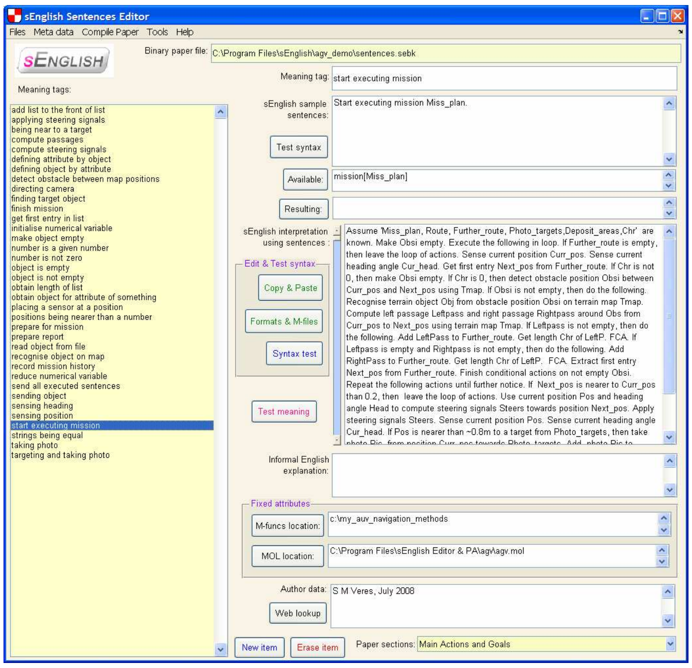
Action: *progressing on follow on part of mission path.*

Get first entry Next_pos from Further_route. If Chr is not 0, then make Obsi empty.

Action: *detecting obstacles*

If Chr is 0, then detect obstacle position Obsi between Curr_pos and Next_pos using Tmap. If Obsi is not empty, then do the following. Recognise terrain object Obj from obstacle position Obsi on terrain map Tmap. Compute left passage Leftpass and right passage Rightpass around Obs from Curr_pos to Next_pos using terrain map Tmap. If Leftpass is not empty, then do the following. Add LeftPass to Further_route. Get length Chr of LeftP. FCA. If Leftpass is empty and Rightpass is not empty, then

Fig. 3. Example section of an NLP document



Binary paper file: C:\Program Files\sEnglish\agv_demo\sentences.sebk

Meaning tags:

add list to the front of list
applying steering signals
being near to a target
compute passages
compute steering signals
defining attribute by object
defining object by attribute
detect obstacle between map positions
directing camera
finding target object
finish mission
get first entry in list
initialise numerical variable
make object empty
number is a given number
number is not zero
object is empty
object is not empty
obtain length of list
obtain object for attribute of something
placing a sensor at a position
positions being nearer than a number
prepare for mission
prepare report
read object from file
recognise object on map
record mission history
reduce numerical variable
send all executed sentences
sending object
sensing heading
sensing position
start executing mission
strings being equal
taking photo
targeting and taking photo

Meaning tag: start executing mission

sEnglish sample sentences: Start executing mission Miss_plan.

Available: mission[Miss_plan]

Resulting:

sEnglish interpretation using sentences : Assume 'Miss_plan, Route, Further_route, Photo_targets,Deposit_areas,Chr' are known. Make Obsi empty. Execute the following in loop. If Further_route is empty, then leave the loop of actions. Sense current position Curr_pos. Sense current heading angle Cur_head. Get first entry Next_pos from Further_route. If Chr is not 0, then make Obsi empty. If Chr is 0, then detect obstacle position Obsi between Curr_pos and Next_pos using Tmap. If Obsi is not empty, then do the following. Recognise terrain object Obj from obstacle position Obsi on terrain map Tmap. Compute left passage Leftpass and right passage Rightpass around Obs from Curr_pos to Next_pos using terrain map Tmap. If Leftpass is not empty, then do the following. Add LeftPass to Further_route. Get length Chr of LeftP. FCA. If Leftpass is empty and Rightpass is not empty, then do the following. Add RightPass to Further_route. Get length Chr of LeftP. FCA. Extract first entry Next_pos from Further_route. Finish conditional actions on not empty Obsi. Repeat the following actions until further notice. If Next_pos is nearer to Curr_pos than 0.2, then leave the loop of actions. Use current position Pos and heading angle Head to compute steering signals Steers towards position Next_pos. Apply steering signals Steers. Sense current position Pos. Sense current heading angle Cur_head. If Pos is nearer than ~0.8m to a target from Photo_targets, then take

Informal English explanation:

M-funcs location: c:\my_auv_navigation_methods

MOL location: C:\Program Files\sEnglish Editor & PA\agv\agv.mol

Author data: S M Veres, July 2008

Paper sections: Main Actions and Goals

Fig. 4. Sentences editor for an NLP document that is part of the *Authoring Tools* for sEnglish documents.

human interpretation function Ψ. Ψ, as defined in Definition 8.1, associates a sentence with a set of conceptual graphs in the human knowledge base H_{KB}.

VII. Procedural knowledge sharing among agents

To be able to define shared skills and knowledge among agents via NLP documents we provide formal definition of a very broad class agents [11], [12]. The term ‘”skill” is used for procedural capabilities such as interaction with the environment and “knowledge” also includes models of relations in the current environment the agent is in. 2^S will denote the set of subsets of a set S and S^* will denote the set of all finite sequences by elements of S.

Let the set of states of the agent’s environment be defined by $S = \{s1, s2, \}$. The effector capability of an agent is represented by a set $A = \{a1, a2, \}$ of actions . At the most basic level an agent can then be viewed as a function

$$action : S* \longrightarrow A$$

which maps finite sequences of environment states to actions. Intuitively speaking an agent decides what action to perform on the basis of its history i.e. its experience to date. The non-deterministic behaviour of the environment can be modelled as a function

$$env : S \times A \longrightarrow 2^S$$

which maps the current state of the environment and the agent’s action to a set of environment states that could result from performing the action. If all the sets in the range are singletons then the environment is called deterministic.

An agent is purely reactive if its action only depends on the current state of the environment, so its action can be described by a function $action : S \longrightarrow A$. This definition of reactive agents is a bit simplistic to be useful and the first architectural issue is to split the action mapping into perception and action:

$$see : S \longrightarrow P \quad , \quad act : P* \longrightarrow A$$

where P is a nonempty set of percepts that is composed of relations of objects with classes in a restricted ontology O. To be more precise let M_O be the set of all objects with class in Γ_O. Relations of objects from M_O are unary, binary, tertiary, etc. partial functions with Boolean or real-vector values. For any positive integer i the $R_i(V)$ denotes the set of i-argument partial functions f that are defined as $f : \Pi_{k=1}^{i} M_O \longrightarrow V$. The set of all relation functions over M_O is $R(V) = \cup_{i \geq 0} R_i(V)$, finally the set of all possible relations is denoted by R. We now define the perception P as some subset of R and P^* as sequences of subsets of R.

This means that act now maps sequences of percepts to actions. Let’s now introduce a set of agent states K. Then the next state is defined by a function $next : K \times P \longrightarrow K$ and the action is decided by a new function $action : K \longrightarrow A$ that maps the set of states into actions. State-based agent description assumes that an initial state exists.

Sentences defined in NLP documents can now be used in various ways:

(a) Relational sentences describing the relations in P are created from sensor signals.
(b) Meanings of instructional sentences describing how actions are executed in terms of modelling objects in M_O. These can be "physical" actions such as feedback/feedforward control or actions can also be "mental" actions such as various types of path planning or planning of steps to achieve a more abstract goal. Assessing priorities in a deliberative agent is also type of mental action.

The above described general agent class with the state transition *next* function can cover all kinds of layered and deliberative agent architectures. The details of how state transition is organized in an agent is not the concern of this paper. We are only interested to define a large class of agents that can be fully documented in an NLP document and can make use of new documents read.

Definition 7.1: An agent $A(O, see, next, action)$ is called *NLP documentable* if there is an NLP document D that describes

(i) a agent-world modelling ontology; O
(ii) the execution of its *see* function for perception and communication;
(iii) the execution of its *next* and *action* functions.

A. Agents sharing a conceptual base

We now assume that in a set A of NLP documented agents each agent a is based on the same document $D = \langle \Xi, O, S, m, L_N, N \rangle$. There will be no specific assumption made about the agent architecture and decision making procedures but the following will be shared among the agents.

1) Each agent $a \in A$ uses ontology O to model the set of objects and systems M of the shared physical environment in which they operate.
2) Each agent $a \in A$ uses the same set of sentences M_r to express relationships among objects and systems of the environment in terms of their perception definitions.

In terms of modelling relations the NLP sentences used by agents can be of the following kind:

1) Query sentences that have for meaning the description of a procedure in terms of NLP sentences.
2) Instructional sentences that have meanings described by sequences of NLP sentences describing the steps of the procedure.
3) Goal sentences that have meanings described by a mixed sequence of instructional sentences and other goal sentences (subgoals).
4) Behaviour constraint sentences in terms of relationships among environmental objects (including the agents "body") that describe what is not allowed while achieving goals.

Note that if something is not behaviourally constrained than it is allowed for any agents during its course of actions to achieve a goal.

B. Agents reading new NP documents

It is now clear that if human authors write about new perception or action procedures in an NLP document that is targeted for an agent class than the following holds.

Theorem 7.1: A perception or action sentence definition S by other NLP sentences in a new NLP document D is useful for agents in an agent class A if the following conditions are satisfied:

(I) The ontology used by D is an extension of the base ontology of the agent class A.
(II) S occurs in the base NLP document of agent class A.
(III) Underlying programming language calls used in the meaning definition of S in D are contained in the routine library H of N used by the agent class A.

$\square$

This means that researchers can look up the basic perception, action, planning, decision making and other sentences of a robot family and write new procedures in an HTML published NLP document. The difference relative to currently available automatic software updates (through the Internet) is that in our systems these Intertnet publications are actually readable English (or other) documents that can be the basis of professional knowledge sharing as that is customary today in journal publications. The next section provides a formal description of knowledge sharing with humans.

VIII. Shared knowledge definitions with humans

Apart from clear presentation of knowledge sharing among a set of artificial agents, the most useful feature of NLP documents is that defined meanings also map to conceptual structures that represent human thinking. The relationship of a document $D = \langle \Xi, O, S, m, d, L_N, N \rangle$ to human thinking, and onto knowledge sharing among a set of human users and programmers, is formulated by its human interpretation functions Ψ.

Definition 8.1: A pair of time varying functions $\Psi^t = [\Psi_r^t, \Psi_c^t]$, is called the *human interpretation* of sentences in an NLP document $D = \langle \Xi, O, S, m, L_N, N \rangle$ if the following are satisfied:

- $\Psi_c^r : \Gamma_O^I \to 2^C$ maps current individuals of classes in ontology Γ of O to a set of feasible conceptual interpretation among the concepts O_c of human CGs.
- $\Psi_r^r : S \to 2^{H_{KB}}$ is a mapping consistent with $d : S \to B_{cg}(O_c, O_r)$.

Note that the difference between the basic CGs $B_{cg}(O_c, O_r)$ and H_{KB} is that the latter represents a stream of compound CGs of human interpretation of the current and past state of the world.

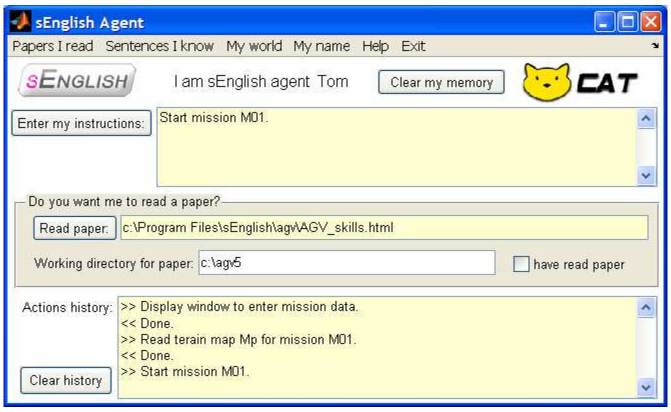
For the human interpreter a sentence $s \in S$ can have varying meanings, i.e. conceptual graphs depending on the context the sentence is uttered in. In general we cannot aim here to formalize human interpretation due to lack if insufficient knowledge about human thinking. There is however a special context for the human mind where meanings of sentences $s \in S$ can be well defined.

Definition 8.2: The human context where a *human agent* A_h have read and remembers the meanings of definitions in $D = \langle \Xi, O, S, m, d, L_N, N \rangle$ is called a *human-machine context* $X(A_h, \Xi_p)$. The human agent A_h is called *nlp-compliant* if it chooses to interpret all $s \in S$ with their unique translator $T(s)$ associated with D.

Note that for humans the way the $\Xi_p = \langle O, S, m \rangle$ interprets a sentence can be only one of the possibilities. Nlp-compliant human behaviour means that the human chooses to interpret sentences according to $\Xi_p = \{O, S\}$. Given a set of humans sharing the knowledge of a $\Xi_p = \langle O, S, m \rangle$ and they are nlp-compliant, then they have shared meaning of any $s \in S$, irrespectively whether they are agent programmers or users of an agent.

Theorem 8.1: Any NLP document $D = \langle \Xi, O, S, m, d, L_N, N \rangle$ provides shared unambiguous understanding of sentence meanings in S for nlp-compliant humans and software agents.

Proof. Any sentence $s \in S$ in an NLP $D = \langle \Xi, O, S, m, d, L_N, N \rangle$ corresponds to a code $T(s) \in L_N$ that defines its meaning for an agent residing on a computer and using D. On the other hand any sentence $s \in S$ also has a unique interpretation $\Psi_r^t(s) \in H_{KB}$ for an nlp-compliant human agent. The correspondence $T(s) \Leftrightarrow \Psi_r(s)$ creates a shared meaning between the software agent and the human as well as a definition of possible meaning for the human. $\square$



I am sEnglish agent Tom

Start mission M01.

Do you want me to read a paper?

c:\Program Files\sEnglish\agv\AGV_skills.html

Working directory for paper: c:\agv5

have read paper

Actions history:

```
>> Display window to enter mission data.
<< Done.
>> Read terain map Mp for mission M01.
<< Done.
>> Start mission M01.
```

Fig. 5. Graphical user interface of an sEnglish reader agent for web pages that can be installed on any PC.

The usefulness of natural language program definitions stems from the fact that

(a) The sentence meanings define the precise meaning that may be ambiguous to the human interpreter

(b) A team of humans can share the understanding of natural language sentences definitions and adopt their high level decision making ability the meaning definitions provided. This is especially so if the NLP sentence definitions and modelling object classes represent generally acceptable examples of human concepts.

IX. Conclusions

This paper has described the theory of natural language programming and theoretical foundations NLP and its use by agents. NLP has existed for a couple of years [2], [3] in software that is now well tested and this paper fills in the gap in terms of precise theoretical definitions. Users of sEnglish can find a formal description that reveals its simplicity and explains its practical strength. This theory clarifies the advantages an NLP publishing system over traditional programming of robots in ADA, Stateflow/Simulink or Python. A brief summary is:

- Through the documents human users can share the knowledge of the agents and that reduces misunderstanding when complex intelligent behaviour is needed during autonomous missions.
- The ”published” papers can be distributed to agents of a specific control program that define their deliberative or reactive behaviour. Instead of reprogramming, the autonomous systems updates/learns from publications as humans do.
- The sEnglish publications can be distributed within a company or on the Internet as ”proper publications” to make colleagues aware of results in sensory signal processing, navigation, environment modelling and adaptive/learning control methods.

The theory also reveals that NLP can be considered as further development of two traditional approaches. On the one hand NLP is an extension of object oriented programming as it uses object classes and its methods are defined by sentences. On the other hand NLP is developing the paradigm of graphical user interfaces (GUIs) further for complex system where GUIs are not sophisticated enough to express internal logic of system operations. In such cases an NLP document can define the conceptual and logical structuring of an application while some NLP sentences can be used to invoke GUIs for data entry and display of data.

References

[1] S M Veres and L Molnar. Documents for intelligent agents in english. In *Proc. of 2010 IASTED Conference on Artificial Intelligence and Applications*, volume CD, Innsbruck, Austria, 2010.

[2] SysBrain Ltd. sEAT - authoring tool for sEnglish documents. *www.system-english.com,*, 2008-2010.

[3] SysBrain Ltd. sERA - reader agent of sEnglish documents. *www.system-english.com,*, 2008-2010.

[4] Y Ye. Supporting software development as knowledge-intensive and collaborative activity. In *Proc. of 2006 Int. Workshop on Interdisciplinary Software Engineering Research*, volume CD, pages 11–23, Shanghai, China, 2006.

[5] E C Way. Conceptual graph overview. *J. Experimental and Theoretical Artificial Intelligence*, 4:75–84, 1992.

[6] B Moulin. Temporal contexts for discours representation: an extension of the conceptual gaph approach. *Applied Intelligence*, 7:227–255, 1997.

[7] L. Molnar and S.M. Veres. System verification of autonomous underwater vehicles by model checking. volume CD, pages 1–10, Bremen, Germany, 2009.

[8] A.R. Willms and S.X. Yang. Real-time robot path planning via a distance-propagating dynamic system with obstacle clearance. *IEEE Transactions on Systems, Man and Cybernetics*, 38(3):884–893, 2007.

[9] S M Veres. *Natural Language Programming of Agents and Robotic Devices*. SysBrain, ISBN 978-0-95584417-0-5, London, June 2008.

[10] J F Sowa. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, Reading, MA, 1984.

[11] S.M. Veres, L. Molnar, N. K. Lincoln, and C. Morice. Autonomous vehicle control systems - a review of decision making. *Journal of Systems and Control*, 78(10):28–69, 2010.

[12] Michael Wooldridge. *Reasoning about Rational Agents*. MIT Press, Cambridge, —2000—.