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 graphs and programming language theories to develop the theoretical foundations of natural language programming (NLP) that has in recent years been used to produce natural language documents for intelligent agents and human readers. The analysis given reveals three benefits of NLP. First, it is “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, i.e. English language documents on control theory and procedures, on the Internet or in printed documents. This theoretical paper also defines a large class of intelligent agents that can read such documents. This enables human users and agents to have 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 for autonomously operating robotic agents where engineers can use natural language programming (NLP,[1]) in sEnglish documents to define their behaviour. After their operational launch, the agents can read new published documents to acquire knowledge such as various physical skills, facts about the environment, behaviour policies and also 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.systemenglish.com). Although sEnglish has been in use for some time this is the first paper providing a theoretical background to it.

Use of NLP documents have various impacts on robot programming:

(1) sEnglish documents can shorten 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. Other engineers can also read the published papers and learn the details of how the vehicle operates, how decisions are reached and how skills are performed.

(2) Users of the autonomous vehicle/robot will clearly understand how it operates and will also know its limitations to avoid misuse or misunderstanding. Users can modify English sentences in the controlling document to influence the vehicles behaviour or how its skills are performed. Also the legal responsibility for misuse can be shifted to the user.

(3) The use of sEnglish provides the rational agent, controlling a vehicle or robot, with abstractions that it can use in reasoning.

(4) The abstractions of sensing and actions through sEnglish can provide a hierarchical abstraction system that can allow the formal verification of the hybrid system of robot/environment interactions.

(5) There are also consequences for team based computer programming, [2] studies hierarchical abstractions for software systems. The need of supporting knowledge collaboration, in software development environments, is highlighted in [3], based on the conceptualization of software development as knowledge-intensive and distributed cognitive activity. This principle is extended in [1] for the context of shared knowledge and collaboration between human users, i.e. scientists, engineers, and machines or intelligent agents. [1] defines natural language programming (NLP) and uses it to link software code with concepts from an ontology via sentences.

Since publishing of the book [1], that introduced the meaning of NLP and its societal/technological context, it has found applications in a number of areas. The paper [4] contained the description of an, at the time new, sliding mode controller for a 6DOF holonomic control of a satellite that allows precise position and attitude control, with thrusters only, within a satellite cluster in formation for interferometry. The control scheme has been fully defined using NLP and an sEnglish paper, that publishes the control scheme in a webpage on the world wide web, is available at www.systemenglish.com and [5].

[6] illustrated another application of NLP for system verification of autonomous underwater vehicles (AUV) by model checking. Here NLP, in the form of an sEnglish documents, defined abstractions for the events and control operations of subsystems of an AUV. These abstractions were first compiled into Satelllow[7] and ISPL [8], the interpretation language for the Multi-Agent Model Checker (MCMAS) used for system verification of the AUV in [9].

The publication [10] reported about the use of sEnglish/NLP for describing all the control operations of an experimental autonomous ground vehicle (AGV) so that the manual equalled the program of the vehicle. The sEnglish
document that was the program of the AGV, contained sections on initialisation of the hardware, preparations for a mission, sensor systems operations, collision avoidance and realtime replanning of path while the vehicle was moving in a model urban environment. The sEnglish document controlling the AGV and video are available at [5].

[11] reviewed the area of decision making in autonomous vehicle systems and concluded that there is a technological gap between sensing devices and intelligent agent programming through logic that handles a discrete world. The transition from sensing to discretisation of events, skills, actions, plans and world models are underdeveloped. It is identified that sEnglish/NLP can play a crucial role in filling this technological gap between sensing and cognition.

[12] is a software environment in which authors can create sEnglish documents in NLP and publish them on the web or a local network in a factory or offices. The sEnglish reader agent [13] provides the client end of the authoring tools that anyone can use to read the sEnglish documents and use its sentences to invoke control actions on her/his PC or application device. In this role the reader agent acts like an intelligent personal assistant (IPA).

The Cognitive Agent Toolbox [14] (CAT), extends the usefulness of [13] from the role of a IPA to that of an autonomous agent that can control a vehicle, manufacturing machine or an appliance. CAT is based on the use of sEnglish/Jason(AgentSpeak) in a MATLAB/Java environment. This application of sEnglish clarifies the actual role of sEnglish and natural language programming: it is not a new agent programming language to compete with robot programming languages, it is a conceptualisation tool of programming that can be used in association with high level programming languages to associate programming steps with human concepts and build up a hybrid automaton that easily lends itself to formal verification and to support legal certification for safety [6], [9].

[15] reports on the applicability of sEnglish in the space technologies where rational agent programming is complemented with external calls to sEnglish sentences that define the well structured connections to sensing and control processes. sEnglish documents, that are NLP routine libraries, can also contain equations, figures, images, quotes, numbers, physical quantities that humans can read as technical papers and can also be read by suitable rational agents. A number of these documents can be found at [5].

Publications for autonomous systems is a logical next step of information systems in a historical perspective [1]:

(a) There was first verbal communication between people.
(b) Writing was developed to record events and knowledge manually for top layers in society.
(c) Printed book was invented, multiple copies of written material could reach masses of people.
(d) Journals, newspapers and the WWW has been used to distribute information and knowledge.
(e) This paper reports about foundation steps of a system of publishing for autonomous machines, the next revolutionary step in information technology.

Fort his a complete software system is available:
(1) authoring tool (sEA T) of sEnglish human/machine readable documents [12];
(2) sEnglish reader agent (sERA) [13] as an IPA;
(3) programming environment to create autonomous agents that can read the sEnglish documents [14].

In this paper we fit NLP into the theoretical abstraction layers of programming as described in [3]. NLP is accompanied by simultaneous human and machine interpretation of NLP sentences. Code inambiguity of NLP sentences preserves the cornerstone of digital computing: the usefulness of digital computation largely originates from 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 illustrations. 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. Note however that this is a paper about the theory, application examples can be found elsewhere as listed in the references.

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 HTML 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
The \( \text{@} \) is also required to satisfy the inheritance condition \( (DL,\text{OWL}, \text{see W3C.org}) \) framework as that will not strictly
modelling object \( = \{ \text{perceived object} \cup \text{imagined object} \cup \text{mathematical object} \} \) that are disjoint and satisfy
(c) There is an atomic role \( \text{has attribute} \).

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 \( O = (\Gamma \mid \Gamma \ni \text{@} \ni \Lambda) \)
consists of a lattice \( (\Gamma \ni \otimes) \) over the class set \( \Gamma \), an attribute
label set \( \Lambda \) and an attribute mapping \( \text{@} : \Gamma \rightarrow 2^{\Gamma(\Lambda)} \) where \(\Gamma(\Lambda) : \Lambda \rightarrow \Gamma \) is a mapping from the attribute label set to the class set \( \Gamma \). The class set has a universal sup-class \( \xi_0 \) that is the \text{modelling object} so that \( \forall \xi \in \Gamma : \xi \prec \xi_0 \) and a
universal sub model class \( \xi_\infty \) such that \( \forall \xi \in \Gamma : \xi_\infty \prec \xi \). The \( \text{@} \) is also required to satisfy the inheritance condition
\( \forall \xi, \zeta \in \Gamma : \xi \prec \zeta \Rightarrow \text{@}(\xi) \ni \text{@}(\zeta) \).

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

Let \( N = (B, L, D, H, P) \) 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 \( N = (B_N, L_N, D_N, H_N, P_N) \) be an instructional
computer language that has a set of variable types \( B_N \). An ontology \( O = (\Gamma \mid \text{@} \ni \Lambda) \) is said to be supported by
\( N \) if \( B_N \subset \Gamma \) and \( \forall \xi \in \Gamma : \xi \ni \text{@} \ni \xi \).

An ontology \( O \) supported by a programming language \( N \) 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 [16], [17] 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 \( G = (G_c, G_r, L_c) \) is a vertex-labelled
digraph that satisfies the following conditions:
(a) the vertices \( G_c \) are decomposed into two sets of \text{concepts}
\( C \) and \text{con} \( \text{relations} \) \( R \) so that \( G_c \subset C \times R \times C \);
(b) the directed edges of \( G \) occur in pairs so that \( G_c \subset C \times R \times C \);
(c) Vertices \( G_c \) are labelled by a \text{referent} function \( L_c : G_c \rightarrow R \) where \( R \) contains of existential, individual, set,
universal and question labels as listed in Table 1.
The set of basic conceptual graph is denoted by \( B_{cg}(O_c, O_r) \).

**Table 1**

<table>
<thead>
<tr>
<th>Referent</th>
<th>Formats</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Existential</td>
<td>[con: *]</td>
<td>DOG:*</td>
</tr>
<tr>
<td>[con: @1]</td>
<td>DOG:@1</td>
<td></td>
</tr>
<tr>
<td>Individual</td>
<td>[con: # num]</td>
<td>DOG:53</td>
</tr>
<tr>
<td>[con: @ num]</td>
<td>DOG:vak</td>
<td></td>
</tr>
<tr>
<td>Set</td>
<td>[con: {*}, .., *]</td>
<td>DOG:{vak, Spot}</td>
</tr>
<tr>
<td>[con: *]</td>
<td>DOG:*@4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[con: *]@#</td>
<td>DOG:*@#</td>
</tr>
<tr>
<td>Universal</td>
<td>[con: ∀]</td>
<td>DOG:∀</td>
</tr>
<tr>
<td>negative</td>
<td>[con: ∼]</td>
<td>DOG:¬</td>
</tr>
<tr>
<td>plural set</td>
<td>[con: *]@many</td>
<td>DOG:*@many</td>
</tr>
<tr>
<td>fuzzy set</td>
<td>[con: *]@many</td>
<td>DOG:*@many</td>
</tr>
<tr>
<td>Question</td>
<td>[con: ?]</td>
<td>DOG:?</td>
</tr>
<tr>
<td>plural</td>
<td>[con: *?]</td>
<td>DOG:*?</td>
</tr>
</tbody>
</table>

**Definitions of referent types in \( R \), \text{con} stands for concept that
is an entry from \( O_c \), and \text{num} stands for number that is a
nonnegative integer. The \( L \) maps a concept in \( O_c \) to on of the
labels in column 2 of this table.**

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 [16] 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 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 [16], [17] are used to represent modalities, modality and propositional attitudes like “know” or “believe” and indirect speech. Conceptual graphs have been extended by agent’s perspectives and temporal localizations [18] 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 \( O_t \) be two disjoint ontologies for concepts and conceptual relations with associated basic conceptual graphs \( B_{cg}(O_c, O_t) \). The set of compound conceptual graphs \( C_{cg}(O_c, O_t) \) is the smallest set that satisfies the following conditions:

(a) \( B_{cg}(O_c, O_t) \subseteq C_{cg}(O_c, O_t) \).
(b) Let \( T_l \) be a set of temporal labels. For any \( x \in T_l \), \( g \in C_{cg}(O_c, O_t) \) also \( T(x, g) \in C_{cg}(O_c, O_t) \).
(c) Let \( T_r \) be a set of temporal relation labels on the real time-line. For any \( x \in T_r \), \( G, H \in C_{cg}(O_c, O_t) \) also \( T(G, x, H) \in C_{cg}(O_c, O_t) \).
(d) Let \( A_l \) be a set of aspect labels. For any \( x \in A_l \), \( G \in C_{cg}(O_c, O_t) \) also \( A(x, G) \in C_{cg}(O_c, O_t) \). The set of A-enveloped aspect graphs will be denoted by \( A(O_c, O_t) \).
(e) Let \( G \in C_{cg}(O_c, O_t) \) and \( c \) a concept in \( G \). The compound graph \( G' \) obtained from \( G \) by replacing all occurrences of \( c \) by an \( A \in A(O_c, O_t) \) is also a compound graph: \( G' \in C_{cg}(O_c, O_t) \).

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 \( \Xi \) be a natural language with word set \( \Xi_w \). 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 \( \Xi_w \). An attribute name is a finite sequence of words from \( \Xi_w \).

1) An ontology \( O \) is called \( \Xi \) compliant if all of its class names and attributes are finite sequences from \( \Xi_w \).
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 \( S(\Xi, O, S) \).
3) An NLP text is a finite sequence of NLP sentences. The set of all feasible, but not necessarily plausible, sentences is named \( \bar{X}(\Xi, O, S) \).

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 \( d(s) \in \bar{X}(\Xi, O, S) \) 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 \( F_{CG} : \bar{X}(\Xi, O, S) \rightarrow B_{cg}(O_c, O_t) \) that uniquely identifies a \( g(s) = F_{CG}(d(s), s) \in B_{cg}(O_c, O_t) \) in terms of a CG with referenced vertices belonging to conceptual classes in \( O \) and conceptual relations in \( O_r \) as permitted by \( B_{cg}(O_c, O_t) \) 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 \( \Xi \). A meaning function \( m : S \rightarrow S^* \cup L_N \cup \{ \emptyset \} \) 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 \( I = (S, m, L_N) \). An interpreter \( I \) is called semantically complete if there is a translator \( T : S \rightarrow L_N \) so that

1) For all \( s \in S \) the \( m(s) \neq \emptyset \).
In terms of other sentences and eventually code in human abstractions, they abstract away the details into their developed meaning. 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 $L_N$.

**Definition 4.4:** NLP document is a tuple $D = \langle \Xi, O, S, m, d, L_N, N \rangle$ of natural language $\Xi$, 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 $\Xi$ 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 $\Xi$.

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 $\Gamma$ of $O$. An individual object under a class $\chi \in \Gamma$ is simply a structure with field names $\{\lambda_i | \exists \chi \in \Gamma : (\lambda_i, \chi) \in @ (\chi)\}$.

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 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 $s_i \in S, i = 0, 1, \ldots, k-1 \text{ such that } m(s_0) \in L_N, s_i \in m(s_{i+1}), i = 1, \ldots, k-1, \ s_k = s$.

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 human knowledge base

8.1, associates a sentence with a set of conceptual graphs in

A

represented by a set
terms of human ontologies

⟨

by

S

set of all finite sequences by elements of

S

is defined by the set

denote the set of subsets of a set

S

human interpretation function

Ψ

relations in the current environment the agent is in. 2

the environment and "knowledge" also includes models of

is used for procedural capabilities such as interaction with

agents via NLP documents we provide formal definition

tions in an NLP

Ξ

are not as well determined. The connections between abstrac-
tions in series of meaning definitions and hence deterministic. This

series of actions. At the most

basic level an agent can then be viewed as a function

action : S* → 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 × A → 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 → 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 → P , act : P* → 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 : Π_{1≤i≤k} M_O → V . The

set of all relation functions over M_O is R(V) = \bigcup_{i≥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 × P → K and the

action is decided by a new function action : K → 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 “men-
tal” 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

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], [19]. 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 = {s_1.s_2}. The effector capability of an agent is

represented by a set A = {a_1,a_2} of actions . At the most
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 NLP documents**

If human authors write about new perception or action procedures in an NLP document, that is targeted for an agent class, 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 subroutine-calls, used in the meaning definition of $S$ in $D$, are contained in the routine library $H$ of $N$ used by the agent class $A$.

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 Internet publications are actually readable English 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 $\Psi$.

**Definition 8.1:** A pair of time varying functions $\Psi^t = \{\Psi^t, \Psi^d\}$, 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:

- $\Psi^t : \Gamma^t_\Theta \rightarrow 2^c$ maps current individuals of classes in ontology $\Gamma$ of $O$ to a set of feasible conceptual interpretation among the concepts $O_c$ of human CGs.
- $\Psi^d : S \rightarrow 2^{HKB}$ is a mapping consistent with $d : S \rightarrow B_{KG}(O_c, O_r)$.

Note that the difference between the basic CGs $B_{KG}(O_c, O_r)$ 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 $D = \langle \Xi, O.S,m, L_N, N \rangle$ is called a human-machine context $\Xi(A_h, \Xi_p)$. 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 $\Xi_p = \langle O.S \rangle$. 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 $D = (\Sigma, O, S, m, d, L_N, N)$ 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 $\Psi_r(s) \in H_{KB}$ 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.

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

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

(b) A team of humans can share the understanding of natural language sentences definitions and adopt their thinking to the meaning definitions provided. This is easy to do for humans if the NLP sentence definitions and modelling object classes represent generally acceptable examples of human concepts.

**IX. PARTICULARISED sENGLISH DOCUMENTS**

A. The use of a context free language (CFL)

B. Descriptions of agent knowledge

C. Application of context dependent languages

D. Modelling for formal verifiability

**X. Conclusions**

This paper has described the theory of natural language programming and its use by agents. NLP has existed for a couple of years [12], [13] 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. As clarified in this paper, NLP is not another proposal for a robot programming language. Instead, it is a radically new approach that conceptualizes programming and can be used in combination with agent programming techniques to provide abstractions and knowledge sharing [15], [12], [14], [10]. A brief summary is:

- Human robot users can share knowledge with agents through sEnglish documents and this can enhance human understanding of robot behaviour. It can reduce the chance of misunderstanding when complex intelligent behaviour is needed during autonomous missions.
- The published sEnglish papers can be distributed to agents that can extract from them some new skills, sensing procedures, deliberative or reactive behaviour policies. Instead of reprogramming, the autonomous systems updates/learns from publications as humans do.
- The sEnglish publications can be distributed within a design company or on an Intra/Internet to make colleagues aware of results in perception signal processing, navigation, environment modelling or in adaptive/learning control methods.

**References**


