Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 4 (2011) 678–687

Procedia Computer Science

International Conference on Computational Science, ICCS 2011

A natural language programming solution for executable papers

Sandor M Veres[a]* and J. Patrik Adolfsson

Faculty of Engineering and the Environment
University of Southampton, Highfield, Southamton, SO17 1BJ, UK

Abstract

The paper describes a system for executable papers for publishers enabling them to reuse content and to generate further advances of science and engineering. The executable algorithmic descriptions within a paper are presented in natural language sentences and basic code, thereby making long term compatibility absolute. Authors are required to use publicly numerical libraries on the Internet or references to publications with executable papers. As used by authors the system automatically creates a web of algorithmic knowledge on the Internet. Novelty of new algorithms in publications can be evaluated by automated tools available to authors, reviewers and readers of scientific papers published.
Keywords: interactive publications, data driven computing, natural language programming

1. Objectives of the system proposed

The system described here is derived from machine readable and executable papers [1-5] in HTML/LaTex formats, for which a Reader Agent [5] is currently available. The system proposed has the following features:

1.1 It provides tools for creating natural language descriptions of executable algorithms (NLDEA) that can be included in PDF and HTML publications and which can be unambiguously interpreted/executed by a web-browser plug-in. Long term compatibility is absolute by use of natural language and self contained interpretation within the paper.

1.2 The NLDEA are formal to such a degree that they enable reviewers to use automated tools to compare past results published in this format. Past published and currently used ontologies and procedural steps can be compared for similarity. A similarity score is computed with 0 being totally dissimilar and 1 meaning essentially identical algorithms.

1.3 No grammar needs to be learnt by authors or readers of NLDEA. Sentence meanings are expressed by other sentences and sentence interpretation is by template fitting. High level computer language is only used for sentence meanings defined by algorithms in publicly available numerical libraries on the Internet.

1.4 The executable paper proposed is to be part of a wider system for machine intelligence where machines directly

* Corresponding author. Tel.: +44-023-80597754; fax: +44-2380-433122.
E-mail address: s.m.veres@soton.ac.uk.

 Selection and/or peer-review under responsibility of Prof. Mitsuhisa Sato and Prof. Satoshi Matsuoka
doi:10.1016/j.procs.2011.04.071

read publications [6] to develop their intelligence. The proposed executable solution is however uch more than a solution to "executable" papers as asked by Elsevier. It leads to publications for machines where all the text of the paper is interpretable by intelligent agents. The system also allows full definition of intelligent agents in English sentences. Natural language programming (NLP) and machine readable sEnglish™ documents and editing tools were first publicly exhibited at the July 2008 IFAC World Congress in Seoul by SysBrain Ltd., UK.

This paper is limited to describing an executable paper system where the papers are either in PDF or in HTML formats and can contain one or several NLDEA sections. In this paper we identify an NLDEA section with an NLP section. Apart from NLDEA sections the rest of the paper is presented in the format of traditional human readable papers as specified by the respective publisher. The system described does not interfere with publisher's current paper presentational requirements. Executable sections can for instance be required by the publisher to be presented either in special fonts, boldface or italics.

2. Outline of the system

The system proposed and depicted within Fig. 1 is based on the inclusion of NLDEA in the form of natural language programming (NLP) text within a published paper, which may be interacted with to gain experience with published results. sEnglish™ is one type of NLP that uses English (www.system-english.com). A web-browser is used to view an electronic version of a published paper. Through the use of a browser plug-in, a reader can rerun experiments of the author or modify editable natural language descriptions of a numerical demonstration that is subsequently reinterpreted and demonstrated. The reader can also modify some natural language sentences to alter display modes of the results.

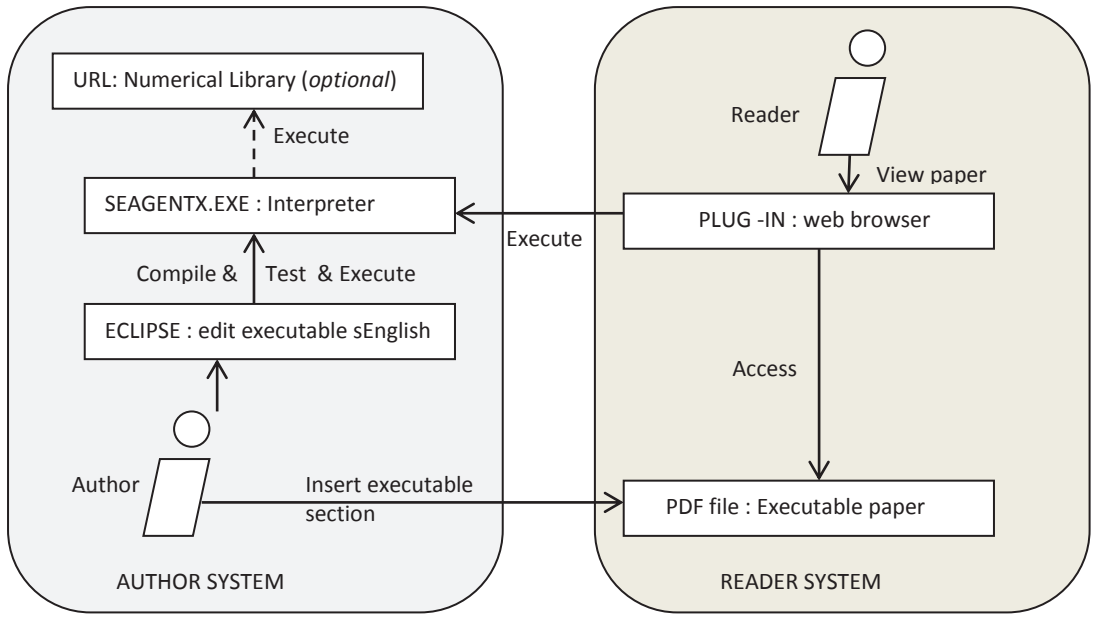


Fig. 1. Functionality of the executable paper system. Difference in [5] is the use of standalone programs not browser plug-ins

2.1 The appearance of executable sections in ordinary pdf publications

Fig. 3 illustrates an example of an NLP text in a paper. Our parser can unambiguously compiles these NLP sentences into executable MATLAB™ code. In a similar fashion, we have parser for Java based sentence meanings. There is the potential to expand NLP for the use of most popular computer languages: C++, Python and ADA, etc. Currently our Reader Agent [5] can use the full power of MATLAB's graphical data displays: the web-browser

plug-in of a journal paper's reader can not only interpret but execute these sentences after suitable modification. Due to the use of precise sentences, the reader of the journal paper can clearly understand the algorithmic steps.

Data classes: **Main classes are: signal, dynamical model, ar model, error. Subclasses of signal are: speech signal white noise. Subclasses of dynamical model are ar model. Subclasses of error are mean square modelling error. Subclasses of speech signal are noisy speech signal. Subclasses of white noise are scalar white noise. Attributes of signal are sampling frequency, sample length, data. Attributes of ar model are ar order, error variance, ar parameters.**

Demonstrating ar filtering of speech signal : Demonstrate ar filtering of speech signal with white noise.

Let Dist be 'scalar white noise' with 'sample length' 100000. Let Sp be a 'speech signal' with 'sample length' 100000. Read 'data' of Sp from file 'http://www.speechsamples.com/sample34521.mat' . Add signal Dist and signal Sp to obtain signal Nsp . Fit ar model Arm to Nsp using 'my method'. Filter speech signal Nsp using ar model Arm to obtain speech signal Spest. Display signal Nsp and Spest in new figure titles 'noisy speech' and 'filtered speech' between ~0.12s and ~0.26s.

Fitting ar model : Fit ar model Arm to Sp using 'my method'.

If M_ is 'my method' , then do the following. Estimate ar model set Ms from Sp up to order 10 . Minimise "(M.ar_order)*M.error_variance" over all entries of Ms to obtain best ar model M. Finish conditional actions.

Filtering speech signal using ar model : Filter speech signal Nsp using ar model Arm to obtain speech signal Spest.

Let K be the 'ar order' of Arm. Let L be the 'sample length' of Spest. Run cycle for "S=K+1:L". Filter Spest at sample S using Arm. Finish cycle for 'S'.

Estimating ar models up to orders : Estimate ar model set Ms from Sp up to order 10 .

Define K as 'integer'. Start empty set Ms. Run cycle for "K=1:Q_". Compute ar model M by ls estimation for order K from Sp. Add M to set Ms. Finish cycle for 'K'.

Filtering speech signal using ar model : Filter speech signal Nsp using ar model Arm to obtain speech signal Spest.

Let K be the 'ar order' of Arm. Let L be the 'sample length' of Spest. Run cycle for "S=K+1:L". Filter Spest at sample S using Arm. Finish cycle for 'S'.

Fig. 2 Example of the main part of a self-contained executable section of a journal paper written in NLP . Some basic operations can either be placed in the Appendix, as we did that in this paper, or placed into a readily available URL of the publisher or the author .

The reader of the paper can execute the sentence ***'Demonstrate ar filtering of speech signal with white noise.'*** with an easy to follow interpretation of the meaning of this sentence as defined within the text in Fig 3. Before execution, modifications can be made by the reader, for instance the maximum model order can be changed from 10 to another number or the least squares estimation done by the sentence '***Compute ar model M by ls estimation for order K from Sp.***' can be replaced by another method, e.g. maximum likelihood method. The user can also change the information criterion used for model order selection in '***Minimise "(M.ar_order)*M.error_variance" over all entries of Ms to obtain best ar model M.'.*** Some of this text is recognizable in the screen dump of the sEnglish™ perspective under Eclipse in Fig. 8 and in the executor window in Fig. 6. When the last sentence of the demonstration is interpreted then the browser plug-in displays a figure as shown in Fig. 2.

Equations can also be included in NLP sentences as MathML or LaTex code that we call sentence-embedded objects (SEOs) in the NLP terminology. Other SEO types are quotes, code scripts, physical quantities, pure numbers and images. An NLP sentence's meaning can determine what it does with the equations. For instance either of the NLP sentences can be used as in Fig 3.

Define "x=1.7892;y=-4.901; t=5". Evaluate "f=x^(-2*y)+e^(-x)".

Evaluate $f(x,y) = x^{-2y} + e^{-x}$ for = 1.7892, $y = -4.901$.

Fig. 3 Illustration of numerically evaluating symbolic expressions.

The former example uses MATLAB™ code SEOs that are between double quotes " ". The second solution is a MathML representation of formulae between $ delimiters that the browser plug-in can interpret in terms of mathematical formulae as shown above. The former can be edited by any basic text editor and the latter also by a graphical HTML editor such as Amaya from the W3C. Mathematical formulae can also be used to state symbolic derivation steps, for instance the NLP sentence in Fig. 5

Note that the function $f(x) = \frac{(x+a)^2}{(a-b)(a-c)} + \frac{(x+b)^2}{(b-a)(b-c)} + \frac{(x+c)^2}{(c-a)(c-b)}$ **evaluates to** $f(x) = 0$ **.**

Fig. 5 Illustration symbolic computation represented by an sEnglish sentence.

is in fact executable and has the code meaning that compares the two SEOs by symbolic computation (not included here).

2.2 Executing sEnglish based NLP sections in ordinary PDF journal papers

The principle of executing NLP sections in publication in PDF or HMTL format, as downloaded by a reader on the Internet, is that they install (only once) a browser plug-in that can be used to interpret any executable sEnglish™ text in PDF or HTML publications. This is actually a special case of reading fully machine understandable publications [1,2,3] that can be used by intelligent agents. Fig. 6 shows an execution window that is activated by the plug-in after the reader selects a text to execute. This window permits modifications of the selected sEnglish™ text

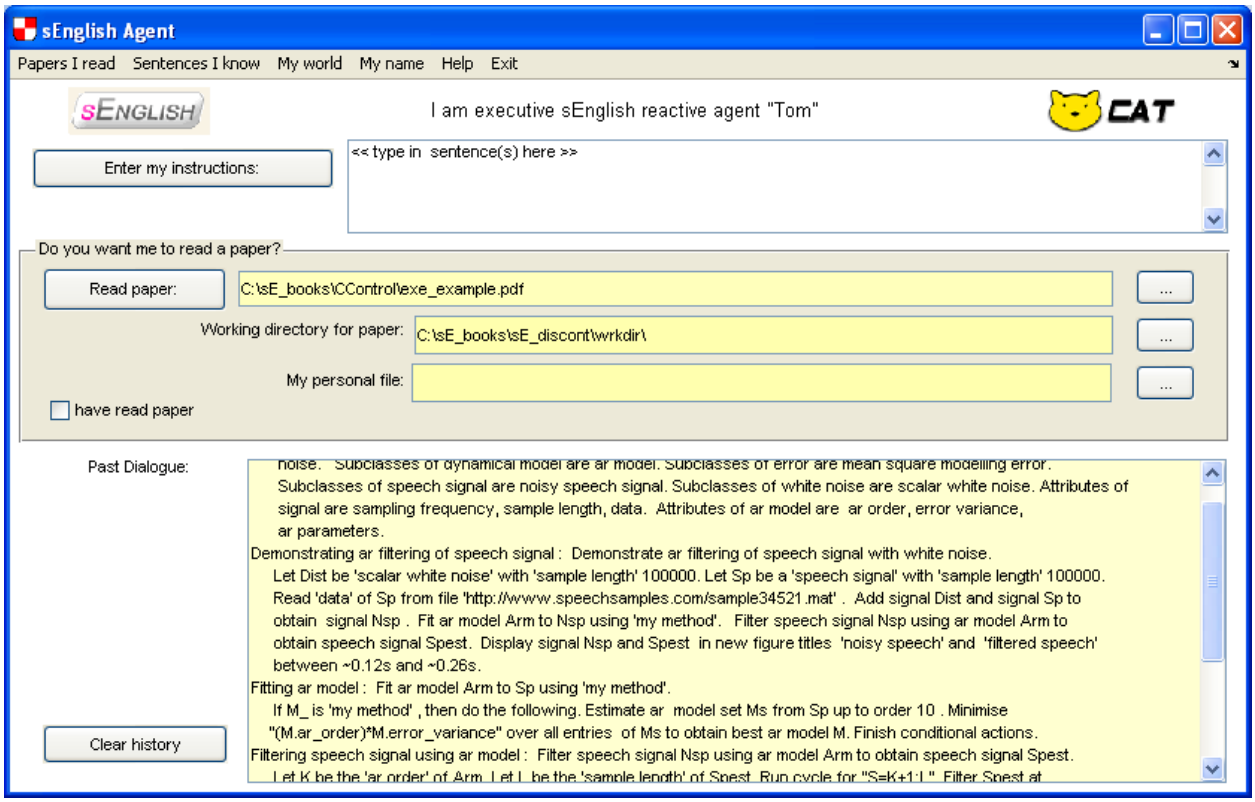


Fig. 6 Illustration of the "execution window" created by an internet browser plug-in.

and the “Enter my instructions” field can be used to enter sentences to execute. The “My World” menu point in the horizontal menu bar can be used to define numerical objects of classes declared in the ontology associated with the sEnglish™ text selected (see next section about creating executable texts). To complete our illustration of executable paper section, the figure displayed in Fig. 7 is displayed when the sentence “Demonstrate ar filtering of speech signal with white noise” is instructed to be executed.

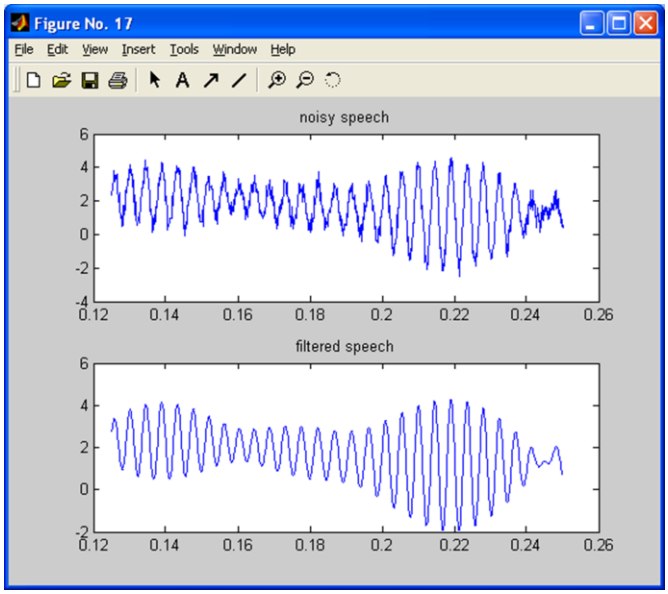


Fig 7. The graphical displays of the web browser are based on the rich capabilities of executable MATLAB™. The reader of the journal paper can further manipulate the 2D and 3D displays for inspection. Note that the reader does not need separate MATLAB™ installation on their computer; installation of the sEnglish™ executable paper interpreter is sufficient.

Modified LaTex tables and graphs, when part of an NLP text in the paper, can be re-evaluated through the GUI of the NLP plug-in to the web-browser. Operations on tables and graphs can be invoked by NLP sentences in the paper. The reader can modify these sentences and redisplay them in a different format. The reader can edit these to redefine data views. Image analysis in 2D and 3D can also be formally described through NLP sections of published executable papers.

Due to the simplicity of this approach, executable papers using sEnglish™ do not have interactive visual tools to edit data and redisplay results, apart from the use of any an ASCII editor. Instead of visual data manipulation tools, the reader can modify sEnglish™ sentences that compile into executable code. This weakness is at the same time the power of this approach to executable papers: difficult to think of any numerical or symbolic scientific results that could not be described in terms of sentences and formulae and data sets.

3. Creating NLDEA sections for ordinary publications

Authors of publications can create natural language descriptions of algorithms using our user friendly Eclipse Java editor plug-in for creating so called “sEnglish documents”. This system is normally used to create fully machine understandable publications [8-10] . We converted this to produce only a section for an ordinary publication in a so called “compact document description” (CDDs) that is illustrated in Fig. 2, in conjunction with Appendix that contains the details for some trivial sentences. Trivial sentences are such that the reader does not normally need to look up their meanings while reading how the executable NLP section works. The author can use an “Appendix to keep the paper self contained and thereby long term compatible for execution; it may also contain a formal

description of the used data ontology. The appearance of the Eclipse plugin, while creating the example in Section 2, is displayed in Fig. 2. On the left the "sEnglish Project View" displays the document structure that can be exported into a pure CDD. The project contains an "Ontology File" (see Appendix), a "DOC Info" file that records the document title, the section titles document title, the section titles. There are also agent logic definitions in NLP that are for future use that goes way beyond the requirements of the executable paper challenge. Its origins are that the authoring system presented

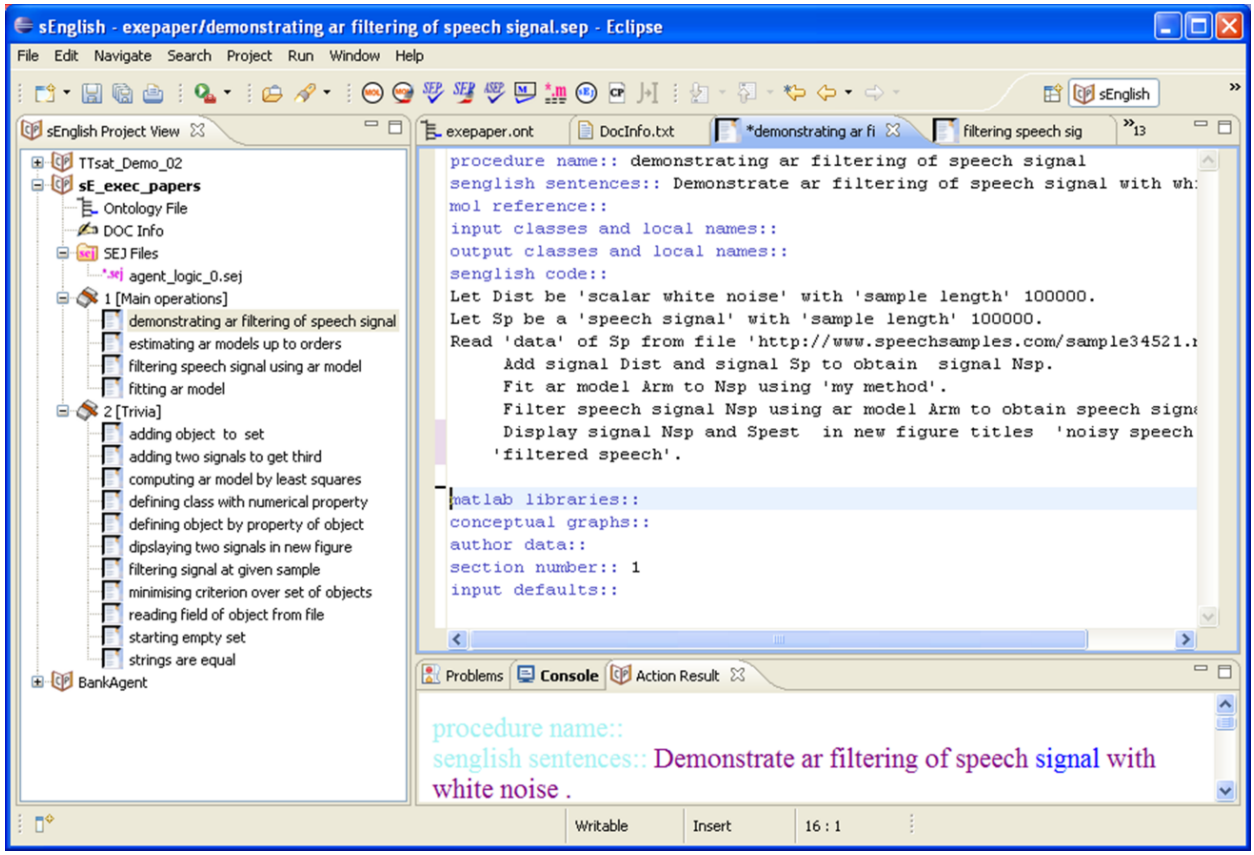
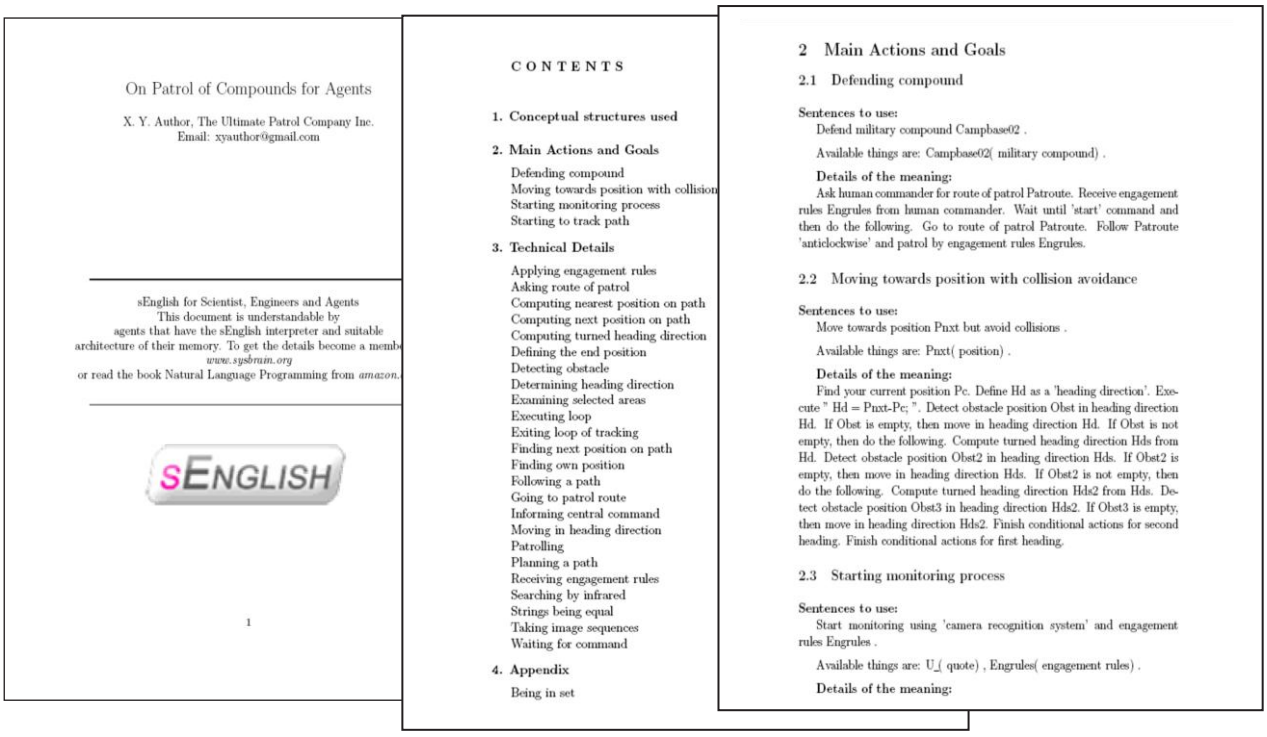


Fig. 8 Eclipse plug-in to create and debug NLP executable text for journal publications in LaTex(PDF) and HTML formats.

is suitable to create not only fully machine understandable papers but to provide the full definition of a belief-desire-intention agent in a single sEnglish document. The special buttons of "**MOL, SEP, M, .m, sEj**" and "**CP**" are provided in the sEnglish perspective to compile, debug document components and finally export it as an HTML or LaTex document. To create a CDD, we augmented the functionality of the "Compile Paper" (**CP**) button with a compiler that produced the text content of Fig. 2 and the Appendix, which together contain all the code to run any of the sentences in Fig. 2. Note however that we have also retained "activity tags" that existed in the original sEnglish documents. Activity tags become subsection titles in fully machine understandable sEnglish papers . Illustration is on Fig. 9 of a document that defines the functionality of an intelligent rover in English, so that engineers operating the rover can read it.

Creating an sEnglish document is relatively easy: apart from the above mentioned files, the author needs to define an ontology for data objects or a "world model". Activity definition is illustrated on right hand side editor window in Fig. 8. Templates of the sentences, that can be used invoke the activity, are defined under "senglish sentences::". Classes and names of available and resulting object classes, as a result of invoking the sentence, also need to be declared with classes defined in the ontology (see Appendix for an ontology example). The section with "senglish code::" is to list the sentences that form the meaning of the activity and the sentences that invoke it.


On Patrol of Compounds for Agents

X. Y. Author, The Ultimate Patrol Company Inc.
Email: xyauthor@gmail.com

sEnglish for Scientist, Engineers and Agents
This document is understandable by
agents that have the sEnglish interpreter and suitable
architecture of their memory. To get the details become a memb
www.sysbrain.org
or read the book Natural Language Programming from amazon.

sENGLISH

1

CONTENTS

1. Conceptual structures used
2. Main Actions and Goals
   Defending compound
   Moving towards position with collision
   Starting monitoring process
   Starting to track path
3. Technical Details
   Applying engagement rules
   Asking route of patrol
   Computing nearest position on path
   Computing next position on path
   Computing turned heading direction
   Defining the end position
   Detecting obstacle
   Determining heading direction
   Examining selected areas
   Executing loop
   Exiting loop of tracking
   Finding next position on path
   Finding own position
   Following a path
   Going to patrol route
   Informing central command
   Moving in heading direction
   Patrolling
   Planning a path
   Receiving engagement rules
   Searching by infrared
   Strings being equal
   Taking image sequences
   Waiting for command
4. Appendix
   Being in set

2 Main Actions and Goals

2.1 Defending compound

Sentences to use:
Defend military compound Campbase02 .

Available things are: Campbase02(military compound) .

Details of the meaning:
Ask human commander for route of patrol Patroute. Receive engagement rules Engrules from human commander. Wait until 'start' command and then do the following. Go to route of patrol Patroute. Follow Patroute 'anticlockwise' and patrol by engagement rules Engrules.

2.2 Moving towards position with collision avoidance

Sentences to use:
Move towards position Pnxt but avoid collisions .

Available things are: Pnxt(position) .

Details of the meaning:
Find your current position Pc. Define Hd as a 'heading direction'. Execute " Hd = Pnxt-Pc; ". Detect obstacle position Obst in heading direction Hd. If Obst is empty, then move in heading direction Hd. If Obst is not empty, then do the following. Compute turned heading direction Hds from Hd. Detect obstacle position Obst2 in heading direction Hds. If Obst2 is empty, then move in heading direction Hds. If Obst2 is not empty, then do the following. Compute turned heading direction Hds2 from Hds. Detect obstacle position Obst3 in heading direction Hds2. If Obst3 is empty, then move in heading direction Hds2. Finish conditional actions for second heading. Finish conditional actions for first heading.

2.3 Starting monitoring process

Sentences to use:
Start monitoring using 'camera recognition system' and engagement rules Engrules .

Available things are: U_(quote) , Engrules(engagement rules) .

Details of the meaning:

Fig. 9. Illustration of pages of a fully executable sEnglish™ paper that uses activity tags as subsection titles.

The author's annotation-free natural language text (see Fig. 3 for illustration) of a numerical procedure is described in their paper in sentences that uniquely compile into computer code . There is no web-based infrastructure needed to make a paper executable . It is an sEnglish browser plug-in that makes NLP sections of a published paper executable. The system has been designed to use minimum amount of information and infrastructure to achieve the capability to make any conceivable algorithm not only executable but naturally understandable by the reader. The power of this approach is based on its simplicity: the author uses human concepts to organise any algorithm in an aesthetic and easily readable way that is then modifiable and executable by the reader. An NLP code is an ontology based, grammar-free, natural-language representation of algorithmic steps on data objects of arbitrary complexity that unambiguously compile into executable code. NLP is not a language but a way to build up meanings of sentences by other sentences until one arrives at basic sentences represented in ordinary computer code. NLP imposes almost no restrictions on sentence structures but requires that the set of sentences must be consistent. Consistency requirements are minimal, relating to two points:

(1) all NLP sentences must be explained by other NLP sentences or computer code,

(2) Modelling objects with proper names in a sentences must belong to an NLP ontology declared within the executable text. All NLP code is based on ASCII characters only, to allow editing in the simplest of editors.

4. Satisfying the Grand Challenge requirements

This section describes how the NLP approach to executable papers satisfies the grand challenge [7] requirements by Elsevier and steps further in one regard, that the call did not require: *the algorithmic descriptions should use professional concepts of the subject area.*

4.1 Executability

The format of a PDF based journal paper is not changing much relative to a conventional paper. The code required to run algorithmic demonstrations of a published paper is wholly contained in NLP descriptions in the paper or in references to other executive papers. The author can refer to other publications for sentence meanings that use shared definitions of the relevant ontology classes with the author' classes. This feature encourages authors to build on each other's work. It reduces the length of their presentation and contributes to its elegance.

4.2. Short and long-term compatibility

The executable algorithmic descriptions within a paper are presented in natural language sentences thereby making long term compatibility absolute. Though professional terms do not change much with time, the system provided here is more lasting than natural language, as all data classes, attributes and procedural names are formally defined in the NLP sections. So even if some terminology changes with time, its past formal meaning can be traced back through executable papers.

Long term compatibility is also supported by that the set of NLP sentences inserted into the paper are self contained and no change of grammar can affect it in the future as *there is no grammar in NLP*: the parser only checks self-referencing within the document. Nevertheless, some sentence meanings can be defined in terms of pure code that depend on stability of programming languages. Most sentences, however, use pre-defined sequences of sentences for their meaning. Long term compatibility is ensured if standard basic MATLAB and Java libraries are used and deposited by the author with their paper with the Publisher. It is the Publisher who should keep the associated libraries as they already keep the paper for perpetuity. Sentence meanings are not to be saved into libraries by the author, they must be explicit in the publication as it belongs to the reader's interpretation space. This ensures long term compatibility. Alternatively, the publisher may store a shared ontology and simple sentences for authors to use. This way long term compatibility can also be ensured.

4.3 Validation

Reviewers can run the examples of the author using the NLP interpreter as a web-browser plug-in. Any NLP code can be highlighted and executed by the plug-in. Most importantly, checking the originality of a published algorithm can be automated if NLP is used to define and demonstrate algorithms in published papers. As NLP is formal, we can create algorithms that can check whether a proposed solution is really new relative to previous publications, or whether it is essentially similar to what has been published before**.** This can help avoid "substantive plagiarism", i.e. the different verbal context will not prevent discovering that an old and a newly submitted algorithm are essentially the same.

4.4 Copyright/licensing

Two types of data can be protected by the author if NLP is used: original data and algorithmic parts that they do not wish to declare in the paper. These can be submitted to the publisher as part of the authors private files in encoded dll format and therefore not visible but available to the reader through the publisher. This can create a new era for the publisher: from a distributor of knowledge to a distributor of technology. Confidentiality and trading agreements can be established between an author and the publisher.

4.5. Systems

The essence of NLP is that algorithms are described in natural language sentences that unambiguously compile into computer code. The same principle can be extended to work that handles large scale computer networks data and it is the author's responsibility to create sentences that handle algorithms and data of that kind. The underlying main programming languages of NLP , that can be either of MATLAB, ADA, Python, Java or even C++, are well capable to access large scale systems over the internet.

4.6 Size

The author can define NLP code for algorithms that handle large file sizes, and access to data depositories over the Internet.

4.7 Provenance

Changes, the reader can make in NLP code of the paper, can affect not only data but also the procedures. Traditional '*do-undo'* facilities and back-stepping to previous NLP code and executions is naturally part of the features of the web browser plug-in.

4.8. Other issues: extensions

It feels natural that the publisher can take on the role as a provider of common knowledge, i.e. author sharable ontologies with some core NLP sentences that can be referenced by an executable paper. This could be done using a collaborative webpage where users can submit and search for algorithms either in executable papers or a common store, each with the confidence of verification of completeness.

Our approach is akin to the semantic web effort proposed by Tim Berners Lee in the early 2000 where a web site/service can be parsed in a meaningful way by machines. This idea, albeit very powerful when complete, has yet to be materialized despite having a comprehensive structure in place (RDF, OWL). One can argue that the web site/service has difficulty motivating the added cost of providing machine readable content for no apparent benefit.

The executable paper will give the provider significant benefits in terms of quick prototyping and be able to express ideas in a natural language. The publisher will play an increasing role in sharing these ideas amongst professionals in searchable executable papers. The ultimate strength of the NLP approach is its fundamental simplicity, supporting researcher needs to reduce information overload [1], and to serve future needs of machine intelligence [1-6].

References

1. S M Veres, *Natural language programming of agents and robotic devices*. SysBrain Ltd, London. ISBN 978-0-9558417-0-5. www.amazon.co.uk and www.system-english.com (June 2008) .
2. S M Veres, Knowledge of machines – a forward look. *Proc. IMech E, Part I*: 225 (2011) *J. Systems and Control Engineering.* (in print)
3. S M Veres. L Molnar, N K Lincoln and C Morice, Autonomous vehicle control systems – a review of decision making. *IMech E, Part I*: 225 (2011) *J. Systems and Control Engineering.* (in print)
4. L. Dennis, M Fisher, A Lisitsa, N. Lincoln and S M Veres, Satellite control using rational agent reasoning. *IEEE Intelligent Systems Magazine*, 25 (2010), 1541.
5. S.M. Veres, *Reader Agent (sERA) and Authoring Tools (sEAT) for sEnglish papers in Natural Language Programming*. www.system-english.com. (2008-).
6. S.M. Veres, *Embedded hardware systems using NLP, human like reasoning and reading published documents using the Cognitive Agents Toolbox.* www.sysbrain.com/sysbrain. (2010-).
7. Elsevier. *Executable Paper Grand Challenge*. http://executablepapers.com
8. K.L. Lincoln and S.M. Veres (2008). Sliding mode control for agents and humans. Proc. TAROS 2008, Edinburgh.
9. S.M. Veres and L. Molnar (2010). Documents for intelligent agents in sEnglish. Proc. IASTED Conf. on Artificial Intelligence Applications – AIA 2010, Innsbruck, Austria.
10. S.M. Veres (2010). Theoretical foundations of natural language programming and publishing for intelligent agents and robots. Proc. TAROS 2010, Plymouth.

Appendix

Details of meanings for trivial sentences to make the executable paper self-contained. Alternatives to such appendices to papers are a repository of trivial sentences by the publisher or a community website. Another, non-sentence based formal description of the ontology used is also included here.

```
Strings are equal :  The 'st1' is 'st2'.
    Execute " B=strcmp(U1_,U2_);".
Adding object to set :  Add M to set Ms.
    Execute "Ms=[Ms;{M}];".
Adding two signals to get third :  Add signal Dist and signal Sp to obtain  signal Nsp.
    Execute "Nsp.sample_length=min(size(Dist.data,1),size(Dp.data)));
    Nsp.data=Dist.data(Nsp.sample_length,:)+Sp.data(Nsp.sample_length,:);
    Nsp.sampling_frequency=Sp.sampling_frequency;".
Computing ar model by least squares :  Compute ar model M by ls estimation for order K from  Sp.
    Execute " X=[];Y=[]; for k=K+1:length(Sp),
    X=[X;Sp(k-1:-1:k-K+1)];Y=[Y;y(k)];end; P=pinv(X'*X)*X'*Y;E=Y-X*P;
    Err=E'*E/length(E);M.ar_order=K;M.error_variance=Err;M.ar_parameters=P;".
Displaying two signals in new figure :  Display signal Sp and Sp2  in new figure 'overlaid'  with legends 'sig1' and 'sig2'.
    Execute " figure;s1=Sp.data;s2=Sp2.data;
    if size(s1,1)<size(s1,2),s1=s1';end
    if size(s2,1)<size(s2,2),s2=s2';end
    m=min(size(s1,1),size(s2,1));
    plot([s1(1:m,:),s2(1:m,:)]);
    figure;subplot(211),plot((1001:1000+M)'/8000,Nsp(1001:1000+M)');
    title('noisy  speech');subplot(212),plot((1001:1000+M)'/8000,Sp(1001:1000+M)');title('filtered speech');
Filtering signal at given sample :  Filter Spest at sample S using ar model Arm.
    Execute "n=Arm.ar_order; p=Arm.ar_parameters;
    s=0;for k=1:n, s=s+p(k)*Arm(S-k);end;Arm(S)=s;".
Defining object by property of object :  Let K be 'prop' of Arm .
    Execute " K=[];if isfield(Arm,U_), K=getfield(Arm,U_);end".
Defining class with numerical property :  Let C be 'classname' with 'property name' 100000.
    Execute " C=create_object(U1_);setfield(C,U2)=Q_;".
Minimising criterion over set of objects :  Minimise "criterion" over all entries  of Ms to obtain best  M.
    Execute "m=inf;idx=1; for k=1:length(Ms),
    M=Ms{k}; v=eval(C_); if v<inf; m=v;idx=k;end;end ".
Reading field of object from file :  Read 'data' of Sp from file 'http://www.speechsamples.com/sample34521.mat' .
    Execute "D=readfile(F_); Sp=setfield(Sp,U_,D);".
Starting empty set :  Start empty set Ms .
    Execute " Ms= {};" .
Ontology:
>signal
 @sampling frequency: physical quantity
 @sample length : double
 @data:  double
   >>speech signal
   >>>noisy speech signal
   >>white noise
   >>>scalar white noise
>dynamical model
 >>ar model
   @ar order : double
   @error_variance: double
   @ar_parameters: double
 >>ar model set :cell
>error
 >>mean square modelling error
```