A NATURAL LANGUAGE PROGRAMMING SOLUTION FOR EXECUTABLE PAPERS

Sandor M Veres & J Patrik Adolfson
University of Southampton & SysBrain Ltd, UK
June 2011

Publisher Server

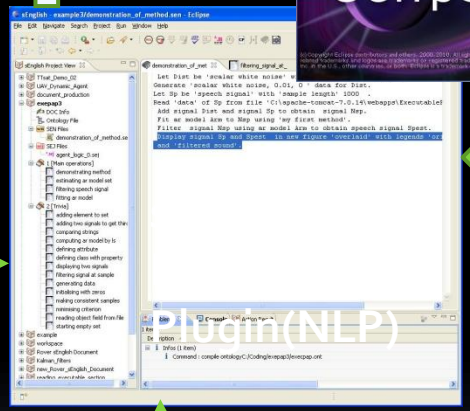
Search &Browse

Author

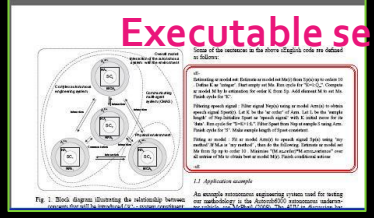
Toolboxes

Symbolic comput.

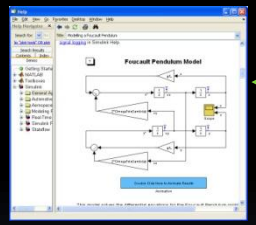
$$\int^{2x-3} + p$$

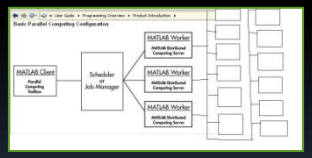


Reader

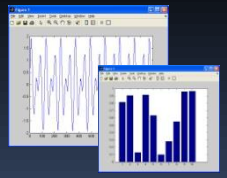


Numerical simulation

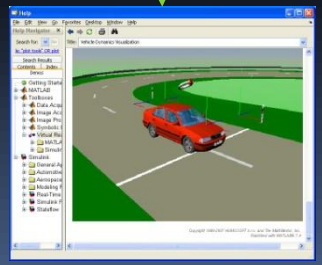




Parallel computation



Data visualisations



Virtual reality

Embedded Systems & Robotics

Conceptualisation of computing?

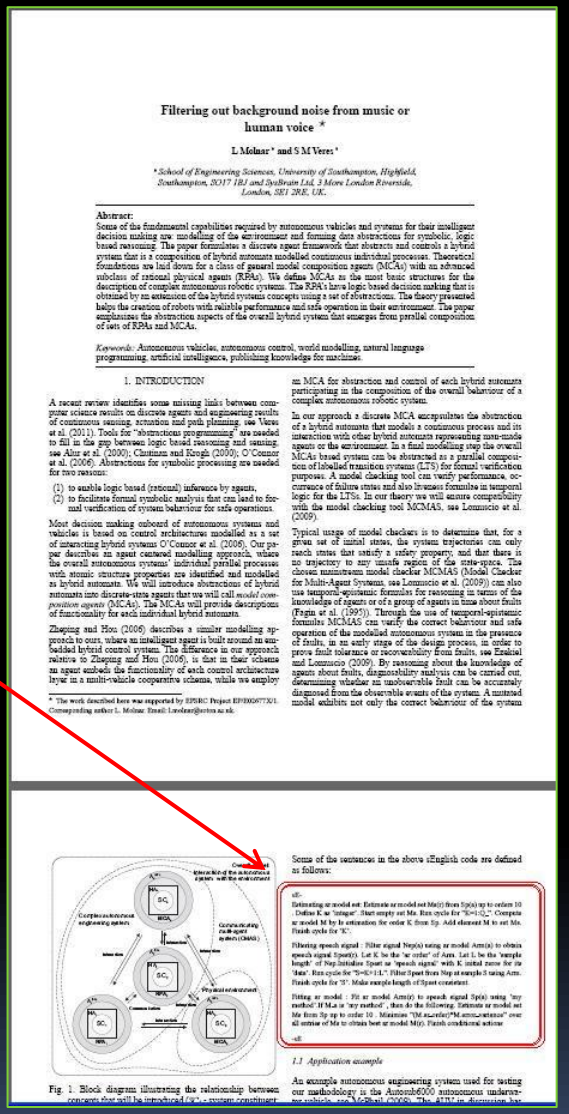
What does it mean and how is it done ?

A video is shown…

Step 1 :

The author writes executable section(s) in their paper in **system English**.

(ordinary PDF,
no hidden annotations)



Filtering out background noise from music or human voice *

L Molnar * and S M Veres *

* School of Engineering Sciences, University of Southampton, Highfield, Southampton, SO17 1BJ and SysBrain Ltd, 3 More London Riverside, London, SE1 2RE, UK.

Abstract:
Some of the fundamental capabilities required by autonomous vehicles and systems for their intelligent decision making are: modelling of the environment and forming data abstractions for symbolic, logic based reasoning. The paper formulates a discrete agent framework that abstracts and controls a hybrid system that is a composition of hybrid automata modelled continuous individual processes. Theoretical foundations are laid down for a class of general modal composition agents (MCAs) with an enhanced subclass of rational physical agents (RPAs). We define MCAs as the most basic structures for the description of complex autonomous robotic systems. The RPA's have logic based decision making that is obtained by an extension of the hybrid systems concepts using a set of abstractions. The theory presented helps the creation of robots with reliable performance and safe operation in their environment. The paper emphasises the abstraction aspects of the overall hybrid system that emerges from parallel composition of sets of RPAs and MCAs.

Keywords: Autonomous vehicles, autonomous control, world modelling, natural language programming, artificial intelligence, publishing knowledge for machines.

1. INTRODUCTION

A recent review identifies some missing links between computer science results on discrete agents and engineering results of continuous sensing, actuation and path planning, see Veres et al. (2011). Tools for "abstractions programming" are needed to fill in the gap between logic based reasoning and sensing, see Alur et al. (2000); Chutinan and Krogh (2000); O'Connor et al. (2006). Abstractions for symbolic processing are needed for two reasons:

(1) to enable logic based (rational) inference by agents,
(2) to facilitate formal symbolic analysis that can lead to formal verification of system behaviour for safe operations.

Most decision making onboard of autonomous systems and vehicles is based on control architectures modelled as a set of interacting hybrid systems O'Connor et al. (2006). Our paper describes an agent centered modelling approach, where the overall autonomous systems' individual parallel processes with atomic structure properties are identified and modelled as hybrid automata. We will introduce abstractions of hybrid automata into discrete-state agents that we will call *model composition agents* (MCAs). The MCAs will provide descriptions of functionality for each individual hybrid automata.

Zhuping and Hou (2006) describes a similar modelling approach to ours, where an intelligent agent is built around an embedded hybrid control system. The difference in our approach relative to Zhuping and Hou (2006), is that in their schema an agent embeds the functionality of each control architecture layer in a multi-vehicle cooperative scheme, while we employ an MCA for abstraction and control of each hybrid automata participating in the composition of the overall behaviour of a complex autonomous robotic system.

In our approach a discrete MCA encapsulates the abstraction of a hybrid automata that models a continuous process and its interaction with other hybrid automata representing state-mode agents or the environment. In a final modelling step the overall MCAs based system can be abstracted as a parallel composition of labelled transition systems (LTS) for formal verification purposes. A model checking tool can verify performance, occurrence of failure states and also liveness formulae in temporal logic for the LTSs. In our theory we will ensure compatibility with the model checking tool MCMAS, see Lomuscio et al. (2009).

Typical usage of modal checkers is to determine that, for a given set of initial states, the system trajectories can only reach states that satisfy a safety property, and that there is no trajectory to any unsafe region of the state-space. The chosen mainstream model checker MCMAS (Model Checker for Multi-Agent Systems, see Lomuscio et al. (2009)) can also use temporal-epistemic formulae for reasoning in terms of the knowledge of agents or of a group of agents in time about faults (Fagin et al. (1995)). Through the use of temporal-epistemic formulae MCMAS can verify the correct behaviour and safe operation of the modelled autonomous system in the presence of faults, in an early stage of the design process, in order to prove fault tolerance or recoverability from faults, see Ezekiel and Lomuscio (2009). By reasoning about the knowledge of agents about faults, diagnosability analysis can be carried out, determining whether an unobservable fault can be accurately diagnosed from the observable events of the system. A mutated model exhibits not only the correct behaviour of the system

* The work described here was supported by EPSRC Project EP/E060773/1. Corresponding author L. Molnar. Email: l.molnar@soton.ac.uk.

Fig. 1. Block diagram illustrating the relationship between concepts that will be introduced (e.g. a system comprising

Some of the sentences in the above sEnglish code are defined as follows:

sE-
Estimating ar model set: Estimate ar model set Ms(i) from Sp(n) up to order 10
Define K as 'integer'. Start empty set Ms. Run cycle for "K=1:Q_2". Compute ar model M by ls estimation for order K from Sp. Add element M to set Ms. Finish cycle for 'K'.

Filtering speech signal : Filter signal Nsp(n) using ar model Arm(n) to obtain speech signal Spest(t). Let K be the 'ar order' of Arm. Let L be the 'sample length' of Nsp.Initialize Spest as 'speech signal' with K initial zeros for its 'data'. Run cycle for "S=K+1:L". Filter Spest from Nsp at sample S using Arm. Finish cycle for 'S'. Make sample length of Spest constant.

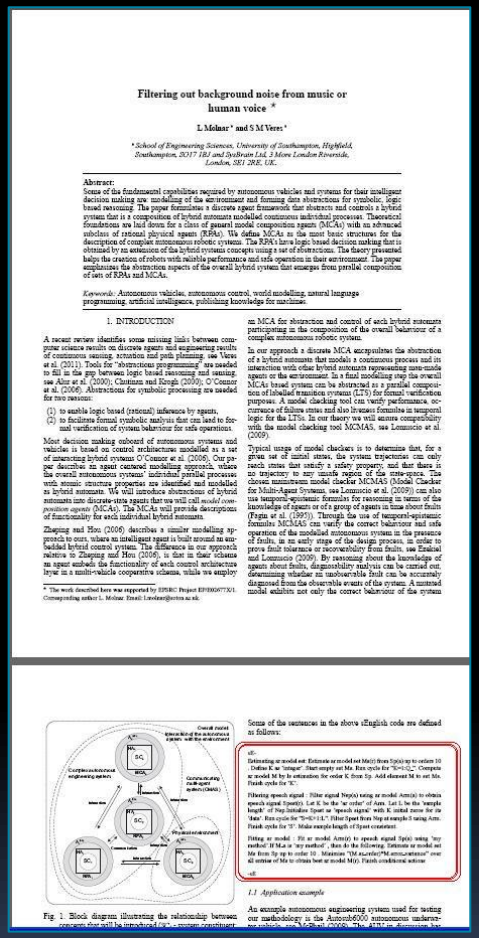
Fitting ar model : Fit ar model Arm(n) to speech signal Sp(n) using 'my method'. If M-A is 'my method' , then do the following. Estimate ar model set Ms from Sp up to order 10 . Minimize "IM.ar.order*M.ar.coefficients" over all entries of Ms to obtain best ar model M(n). Finish conditional scheme

-sE

1.1 Application example

An example autonomous engineering system used for testing our methodology is the Autosub6000 autonomous underwa-

Step 2 : Reader extract executable sections from published paper and runs them



Filtering out background noise from music or human voice *

L Molnar * and S M Veres *

* School of Engineering Sciences, University of Southampton, Highfield, Southampton, SO17 1BJ and SysBrain Ltd, 3 More London Riverside, London, SE1 2RE, UK.

Abstract:
Some of the fundamental capabilities required by autonomous vehicles and systems for their intelligent decision making are modelling of the environment and forming data abstractions for symbolic, logic based reasoning. The paper formulates a discrete agent framework that abstracts and controls a hybrid system that is a composition of hybrid automata modelled continuous individual processes. Theoretical foundations are laid down for a class of general modal composition agents (MCAs) with an advanced subclass of rational physical agents (RPAs). We define MCAs as the most basic structures for the description of complex autonomous robotic systems. The RPA's have logic based decision making that is obtained by an extension of the hybrid systems concepts using a set of abstractions. The theory presented helps the creation of robots with reliable performance and safe operation in their environment. The paper emphasizes the abstraction aspects of the overall hybrid system that emerges from parallel composition of sets of RPAs and MCAs.

Keywords: Autonomous vehicles, autonomous control, world modelling, natural language programming, artificial intelligence, publishing knowledge for machines.

1. INTRODUCTION

A recent review identifies some missing links between computer science results on discrete agents and engineering results of continuous sensing, actuation and path planning, see Veres et al. (2012). Tools for "abstractions programming" are needed to fill in the gap between logic based reasoning and sensing, see Alur et al. (2000), Clarkson and Kreigh (2009), O'Connor et al. (2006). Abstractions for symbolic processing are needed for two reasons:

(1) to enable logic based (rational) inference by agents,
(2) to facilitate formal symbolic analysis that can lead to formal verification of system behaviour for safe operations.

Most decision making onboard of autonomous systems and vehicles is based on control architecture modelled as a set of interacting hybrid systems O'Connor et al. (2006). Our paper describes an agent centered modelling approach, where the overall autonomous systems' individual parallel processes with atomic structure properties are identified and modelled as hybrid automata. We will introduce abstractions of hybrid automata into discrete-state agents that we will call *model composition agents* (MCA). The MCAs will provide descriptions of functionality for each individual hybrid automata.

Zhaqing and Hou (2006) describes a similar modelling approach to ours, where an intelligent agent is built around an embedded hybrid control system. The difference in our approach relative to Zhaqing and Hou (2006), is that in their scheme an agent embeds the functionality of each control architecture layer in a multi-vehicle cooperative scheme, while we employ an MCA for abstraction and control of each hybrid automata participating in the composition of the overall behaviour of a complex autonomous robotic system.

In our approach a discrete MCA encapsulates the abstraction of a hybrid automata that models a continuous process and its interaction with other hybrid automata representing user-made agents or the environment. In a final modelling step the overall MCAs based system can be abstracted as a partial composition of labelled transition systems (LTS) for formal verification purposes. A model checking tool can verify performance, occurrence of failure states and also liveness formulae in temporal logic for the LTSs. In our theory we will assess compatibility with the model checking tool MCMAS, see Lomuscio et al. (2009).

Typical usage of model checkers is to determine that, for a given set of initial states, the system trajectories can only reach states that satisfy a safety property, and that there is no trajectory to any unsafe region of the state-space. The closest mainstream model checker MCMAS (Model Checker for Multi-Agent Systems, see Lomuscio et al. (2009)) can also use temporal-epistemic formulae for reasoning in terms of the knowledge of agents or of a group of agents in time about faults (Fagin et al. (1995)). Through the use of temporal-epistemic formulae MCMAS can verify the correct behaviour and safe operation of the modelled autonomous system in the presence of faults, in an early stage of the design process, in order to prove fault tolerance or recoverability from faults, see Ezekiel and Lomuscio (2009). By reasoning about the knowledge of agents about faults, diagnosability analysis can be carried out, determining whether an unobservable fault can be accurately diagnosed from the observable events of the system. A minimal model exhibits not only the correct behaviour of the system

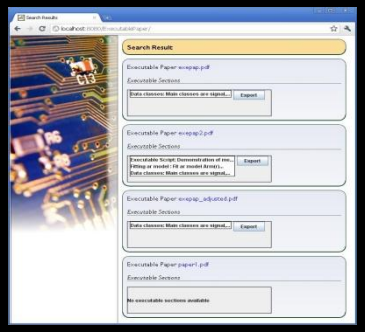
* The work described here was supported by EPSRC Project EP/H009779/1.
Corresponding author L. Molnar. Email: l.molnar@soton.ac.uk

Fig. 1. Block diagram illustrating the relationship between

Some of the sentences in the above sEnglish code are defined as follows:

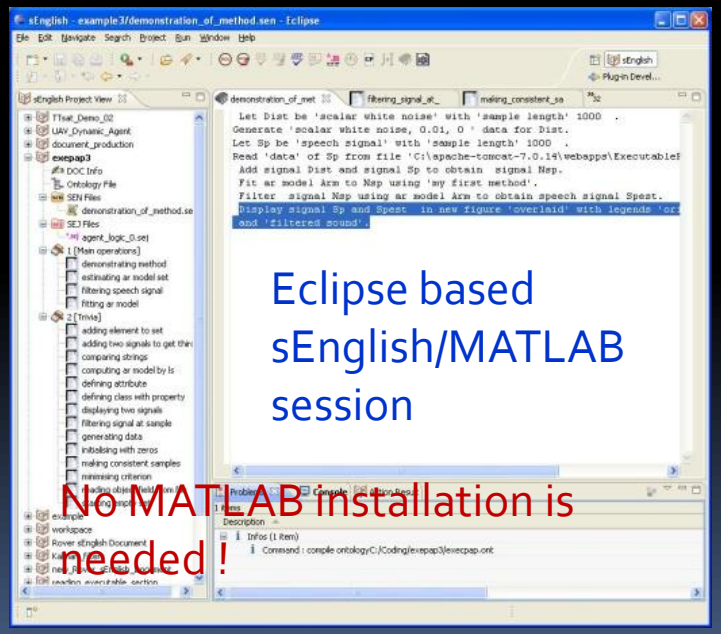
1.1 Application example

An example autonomous engineering system used for testing our methodology is the Autonet4000 autonomous underwa-



+same time all data and trivial sentences automatically downloaded from publisher

All executable sections are extracted from the PDF paper at the click of a button



"System English is not a new programming language it does not have grammar and minimal syntax. It uses natural language to conceptualise programming."

(SM Veres, 2008)

(sEnglish, pronounce it as s-english)

Examples:

sE-
Estimating ar model set: Estimate ar model set Ms(r) from Sp(a) up to orders 10 . Define K as 'integer'. Start empty set Ms. Run cycle for "K=1:Q_". Compute ar model
Finish cy

Filtering
speech s
length'
'data'. R
Finish cy

Fitting a
method'
Ms from
all entrie

-sE

sE-
Data classes: Main classes are signal, dynamical model, error, model set (cell).
Subclass of sig
is ar model, ar
speech signal i
noise. Attribute
length(double),
variance(double
-sE

sE-
Executable Script: Demonstration of method. Let Dist be 'scalar white noise' with 'sample length' 1000 . Generate 'scalar white noise, 0.01, 0 ' data for Dist. Let Sp be 'speech signal' with 'sample length' 1000 . Get data from url 'http://www.speechsamples.com/sample34521.zip'. Load 'data' of Sp from ascii file 'temp.dat'. Add signal Dist and signal Sp to obtain signal Nsp. Fit ar model Arm to Nsp using 'my first method'. Filter signal Nsp using ar model Arm to obtain speech signal Spest. Display signal Sp and Spest in new figure 'overlaid' with legends 'original sound' and 'filtered sound'.

-sE

Why would you insert executable system English sections into your publication as an author?

Several reasons:

Number 1 :

sEnglish sections are like an advanced pseudo code to describe how your modelling, signal processing, control or engineering design procedures work.

Number 2 :

It provides a clear and executable description of how your procedure works. The reviewer of your paper cannot have any doubt that your procedure works.

Number 3 :

The reviewer, and later the reader, can verify that the demonstration is indeed the procedure described in theory:

sEnglish executable sections can be read and understood by reviewers and readers.

Alternatives :

1. If you provide a Java/C/C++/C# then readers cannot clearly see whether your executable code is really the procedure you describe in theory.
2. Including MATLAB/ADA code in your published paper is inelegant and still is not clear that how it works or well organised.

Number 4 :

Your published procedure in sEnglish, that readers can experience how it runs, can be easily accepted and acknowledged by your colleagues.

A "clear way" to success!

Note :

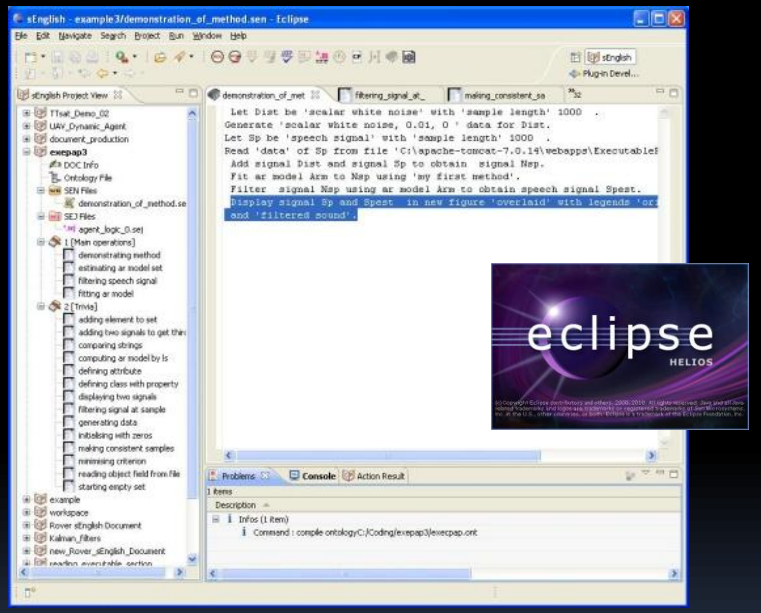
This is not an advantage to authors who want to hide that their procedures does not work or that it is essentially the same as prior published work.

Warning:

sEnglish is amazingly powerful ! End free !

Do not try to form any judgement about sEnglish before you try editing an sEnglish document in Eclipse !

Try it today !



Free download from : www.senglish.org

The rest of the video shows the
Reader's
and
Author's
operations in some detail (10 mins).

www.senglish.org