ADAPTATION AND LEARNING IN AN AUTONOMOUS PHYSICAL AGENT ARCHITECTURE

Sandor M Veres * **Aron G Veres** **

* *University of Southampton, UK, s.m.veres@soton.ac.uk*
** *SysBrain Ltd, Birmingham, UK,*
Email:aron@sysbrain.com

Abstract: Learning and adaptation is presented for a specific but generally applicable autonomous physical agent (APA) architecture. The paper is providing a general framework of skills learning within behaviour logic framework of agents that communicate, sense and act in the physical world. It is shown how programmed playfulness can be easily implemented that results in learning and ultimately better skills of agents. Reusability of results in learning algorithms is supported by ontology based classification of learning in operational modes. Ontology based classification provides object instances that enable modularization of software and easy interfacing of operational modes with learning algorithms.

Keywords: Intelligent physical agents, formal methods, learning, adaptive control modelling for control, .

1. INTRODUCTION

Autonomous agents have been used in computer science for some time (Wooldridge, 2002; Meystel and Albus, 2002). In the field of engineering we want autonomous machines to be (1) capable of solving and executing complex problems in the physical world and we also want them to be (2) reliable. Reliability in essence means that we know how they behave. Reactive agents are good for this as we can prescribe or learn the rules by which the behave. Deliberative agents using modelling and complex decision making are however difficult to verify for all physical environments and hence their reliability is more difficult to prove. This apparent dilemma is attempted to be partially solved by an agent architecture in this paper that prescribes a clearly defined behaviour logic for reliability while retaining the ability of learning, adaptation and complex problem solving by modelling. The architecture is verifiable by design.

2. BEHAVIOUR LOGIC OF AN AGENT

The language $\mathbf{L}_{ABL}$ of temporal *agent behaviour logic* is defined over a set of atomic formulae $O_m = \{p, q, ...\}$ called *operational modes* (OMs) as follows:

$$\phi = p|\neg\phi|\phi \vee \phi|\phi \wedge \phi|\Box\phi|\Diamond\phi| \rightharpoonup \phi|\top|\bot \qquad (1)$$

With each operational mode in O_m there is an activity dynamics associated by the *activity function* defined as

$$A : O_m \mapsto F_b \qquad (2)$$

where F_b is a set of feedback loops between the agent's actuators and a part of the agents internal or external environment. There are three important temporal functions defined over the set F_b. The first one is the Boolean temporal *activation function* $a : F_b \rightarrow \{0, 1\}$ and the second one is the *activity value function* $v : F_b \rightarrow [-1, 1]$

and the third one is the timeout function $t : F_b \rightarrow [0, \infty]$. The activation function provides a semantics for the logic of operational modes (OMs) as the a can be used to evaluate any temporal logic behaviour formula through the activity functions associated with OMs.

Definition 1. A logic formula is called a *simple behaviour* if it only contains the operations $\vee$, $\wedge$ and $\rightharpoonup$.

A simple behaviour defines parallel activities connected by ($\wedge$), sequential activities connected by $\rightharpoonup$ and activity options connected by $\vee$ relations. For instance the $A \vee (B \wedge C) \vee (D \rightharpoonup E \rightharpoonup F)$ can mean that either the A operational mode (OM) is on or the B and C OMs are simultaneously on or the OM D is first on then followed by OM E which is followed by F. When F stops operating then either A must start, or B and C or D needs to restart again.

Definition 2. A *level-1* autonomous physical agent $\mathbf{A}$ is defined by the tuple $\mathbf{A} = \{O_m, A, F_b, a, v, t, B\}$ where B is a simple behaviour in terms of the OMs in O_m.

A level-1 APA defines its semantics in terms of its activity function a. At any moment of time its behaviour formula B can be evaluated in terms of a. Note that satisfaction of B at any time instant does not mean anything about the success or reliability of of the agent, it merely says that at any time the agent will activate OMs in accordance with satisfying formula B (for any A and B the $A \rightharpoonup B$ is defined true if either A or B holds true, operations of $\vee, \wedge$ are defined as usual).

Definition 3. A *level-1* autonomous physical agent $\mathbf{A}$ is is called *consistent* if B is true at any time , i.e. $\Box B$ true as evaluated using a.

Note that function a is evaluated over the temporally changing processes F_b and expresses the fact that the agent runs the algorithms of F_b. The qualification of success of operations F_b is expressed by function v, which means poor performance for low positive values, very good performance for values near to 1, instability or totally unacceptable performance for small negative values of v and damaging or dangerous performance for v close to -1.

Definition 4. A *level-1* consistent autonomous physical agent $\mathbf{A}$ with simple behaviour formula B is called *safe* if it always evaluates active OM with $v > 0$. $\mathbf{A}$ is called *reliable at level* $\epsilon > 0$ if the active OMs are always evaluated to $v > 1 - \epsilon$ eventually.

Note that the $v > 0$ is defined at any time instant does not mean that $v > 0$ is only dependent on feedback loop data at time instant. $v > 0$ is typically a monotone function of a control criterion that is obtained from a sequence of past feedback input-output data.

Safe operation depends on the actual interaction of the agent with its environment. A lot can be achieved for safety by simply altering the a switching function so that if an OM approaches the $v < 0$ region then the agent switched to another OM that perform better. Whether that will help to achieve overall objectives is another matter. Automating a consistent switching mechanism is the topic of the next section.

3. SWITCHING BETWEEN OPERATIONAL MODES

The activity function a of a level-1 APA is changing over OMs as the OMs change and there are the following practically important cases to consider:

(1) An active OM is successfully completed at a required level reliably.
(2) An active OM is not successful at the required level (e.g. $v < 0.5$) but it still operates safely.
(3) An active OM is timed out.
(4) An active OM is being aborted by another active OM.

Fig. 1. The feedback interaction of an operational mode $f \in O_m$ with the environment. An uncertain environmental model M_f can be used to assess or verify and agent's performance under OM f.

The interaction of an agent with its environment under a single operational mode is normally the topic of control engineering. This control engineering problem can however be widened with the online variable choice of actuators and sensors to be used in the feedback loop.

The objective of this paper is to establish safe operational mechanisms of autonomous physical agents. The previous section introduced the important concept of consistency for level-1 agents.

A further step towards safe operation and good performance is to analyze the conditions of successful operation and bring them together with the above logical framework. To keep the new formalism to a minimum, any $f \in F_b$ will be associated with an initial condition $I(f, M_f)$ that is a 0 or 1 Boolean valued relation function between the agent and its environment.

Definition 5. We say that the *adaptation condition is ϵ-satisfied* by an operational mode f if it holds that whenever $I(f, M_f)$ is satisfied and the agent has f active, the performance $v(f)$ is guaranteed to converge to and stay inside $[1-\epsilon, 1]$ within time period $t(v)$ under uncertain environmental model M_f.

For instance a biped robot may be able to start and walk nicely ($v(f) \rightarrow [1-\epsilon, 1]$) if not starting from a lying or fallen-over position but if it already stands reasonably upright ($I(f, M_f)$), even if perhaps a rucksack has been placed on its back. The latter means that under some initial condition the operational mode of the walking of the robot is adaptive. If the condition $I(f, M_f)$ of walking is not satisfied then the robot may decide to switch to another operational mode f_1, meaning for instance that the robot is "trying to stand up" first which is an operational mode itself.

Individual OMs of the agent can be tested by formal analysis and practical testing. Hence the above analysis highlights the relevance of enforcing such an a on agent behaviour that starts any $f \in F_b$ under condition $I(f, M_f)$ being satisfied that leads to eventually $v(f) > 1-\epsilon$.

Lemma 6. A *level-1* consistent autonomous physical agent **A** is reliable at level $\epsilon > 0$ if the following two conditions are satisfied:

(a) All operational modes $f \in M_f$ satisfy the adaptation condition at level ϵ.
(b) The activity function a is such that whenever an $a(f)$ becomes 1 for an $f \in F_b$ then $I(f, M_f)$ is satisfied.

Proof. Straightforward from the definitions: as all operational conditions are adaptive and start from correct initial conditions the performance function v will rise above $1-\epsilon$ for any operational mode within its time limit.□

3.1 Example 1

Assume that a garden robot has the following mission: either (1) mow the lawn or (2) turn on the watering system or (3) recharge its own batteries or (4) empty the grass from its container to a prescribed dump site (5) report to a human operator for maintenance. Within each of these tasks there are several OMs to be executed consecutively:

- $O_1 \rightarrow F_{p1}$*: Mowing.*
- $O_2 \rightarrow F_{p2}$*: Planning of mowing.*
- $O_3 \rightarrow F_{p3}$*: Watering.*
- $O_4 \rightarrow F_{p4}$*: Planning of watering.*
- $O_5 \rightarrow F_{p5}$*: Empty grass container.*
- $O_6 \rightarrow F_{p6}$*: Planning route to charging point.*
- $O_7 \rightarrow F_{p7}$*: Recharge.*
- $O_8 \rightarrow F_{p8}$*: Decide on and request maintenance.*
- $O_9 \rightarrow F_{p9}$*: Write problems report.*
- $O_{10} \rightarrow F_{p10}$*: Map building.*
- $O_{11} \rightarrow F_{p11}$*: Self modelling of hardware ware.*
- $O_{12} \rightarrow F_{p12}$*: Modelling past mowing, watering and maintenance done.*
- $O_{13} \rightarrow F_{p13}$*: Idle or sick leave.*

Out of these O_1, O_3, O_5, O_7 are feedback-loop based operational modes that need sensing and control of actions accordingly. O_2, O_4, O_6 are on the other hand OMs that need algorithms working on models only and do not need sensing or actuation. OMs O_{10}, O_{11} and O_{12} need sensing only and algorithms that build models from sensor data. Decisions by O_8 are based on internal models and may result in sending a message to the human operator after writing report O_9 on the problems that may need maintenance.

A behaviour logic formula that the autonomous lawnmower needs to satisfy can for instance be

$$B_1 = (\ (O_2 \rightharpoonup O_1) \vee (O_4 \rightharpoonup O_3) \vee (O_6 \rightharpoonup O_7)\ ) \wedge \\ \wedge (O_8 \rightharpoonup O_9) \wedge O_{11} \wedge O_{12} \wedge O_{10} \quad (3)$$

The a switch must be such, as decided within each operational mode, that the total formula B must always hold true. This means that the parallel modelling and maintenance monitoring operational activity carry on while on of the mowing, watering or recharging tasks are executed. Despite the essentially reactive behaviour the lawnmower has the ability of interpretation of the environment and planning while strict discipline of behaviour code is maintained. Based on sensing or assessment of algorithmic results, the evaluation of v is constantly carried out for each operational mode.

Now the reliability at level ϵ is achieved if all feedback and open-loop OMs are proven to work under uncertain models of the environment and the initial conditions are always achieved when switching to a new operational mode. To achieve the necessary initial conditions the OM algorithms need careful action around the switching points. When the physical and control algorithmic work on the lawnmower robot has been completed, then the satisfactory nature of a, v, t can be formally tested. As this may be difficult in practice, adap-

tation and learning in the OMs is therefore vital to reduce development effort and to achieve level-ϵ reliability of the autonomous lawnmower.

3.2 Example 2

An unmanned light aircraft can have the following tasks:

- $O_1 \rightarrow F_{p1}$*: Warming up and control tests.*
- $O_2 \rightarrow F_{p2}$*: Checking mission instructions.*
- $O_3 \rightarrow F_{p3}$*: Planning flight.*
- $O_4 \rightarrow F_{p4}$*: Taking off.*
- $O_5 \rightarrow F_{p5}$*: Taking mission related pictures and measurements.*
- $O_6 \rightarrow F_{p6}$*: Controlling the plane at normal altitude while following mission path.*
- $O_7 \rightarrow F_{p7}$*: Deciding to abort mission.*
- $O_8 \rightarrow F_{p8}$*: Searching for emergency landing area.*
- $O_9 \rightarrow F_{p9}$*: Planning flight to emergency area.*
- $O_{10} \rightarrow F_{p10}$*: Emergency landing.*
- $O_{11} \rightarrow F_{p11}$*: Planning normal return flight.*
- $O_{12} \rightarrow F_{p12}$*: Normal landing on return.*
- $O_{13} \rightarrow F_{p13}$*: Controlling the plane in emergency.*
- $O_{14} \rightarrow F_{p14}$*: Modelling flight path followed until current time.*
- $O_{15} \rightarrow F_{p15}$*: Writing and sending mid-flight report on conditions on board.*

The following behaviour formula is an example:

$$B_2 = ((O_1 \rightharpoonup O_2 \rightharpoonup O_3) \vee (\ O_4 \rightharpoonup O_6) \vee O_5\ ) \vee \\ \vee (O_{13} \wedge (O_7 \rightharpoonup O_8 \rightharpoonup O_9)) \vee \\ \vee (O_{12} \rightharpoonup O_6 \rightharpoonup O_{12})) \wedge O_{14} \quad (4)$$

The dynamical control, planning and emergency challenges and the algorithms are more complex here than in the lawnmower example. Still the same principles of using the a, v, t functions can be used to assess reliability and performance. The significance of adaptation and learning is even more emphasized here.

Both examples suggest that logical consistency provided by a is a fundamental requirement. Beyond that, safety and performance of the system depends on the quality of the OM algorithms to provide suitable v functions. Whatever is achieved and guaranteed in terms of the v, there is scope for *further formal analysis and alterations to be made* to enforce safety switches in face of undesirable performance in some operational mode. This is the topic of automated adaptation and learning.

4. ADAPTATION AND LEARNING OF LEVEL-1 AGENTS

The principle is that adaptation is performed by

- adjusting controller parameters depending on experience
- associating successful feed-forward actions with short term planning

Learning can be restricted to OMs that include feedback based interaction with the environment, for instance path following of the law mower, its approach and connection to the recharging point, or the flight control parameters of the plane under various operating conditions. Measurement of performance and success is crucial in feedback loops of interaction with nature so that improvements can be made by learning. Performance is measured by the evolution of the v-functions during the execution of an OM. The objective of adaptation and learning is to increase v and to make it converge faster. This is only achieved if the agent successfully adapts its OMs to varying environmental circumstances. In this section we briefly review the main available techniques for learning in each operational mode.

4.1 Parametric feedback/feedforward tuning

On-line parametric feedback tuning of an $f_i \in F_b$ is one of the fastest learning methods. The v can be defined as a monotone function of a control performance criterion and some parameter vector θ_i of the feedback/feedforward (FB/FF) controller is to be tuned. The principle of tuning is to compute the gradient direction of the cost function using online measurement data of sensors and actuators and hence move the parameter θ_i uphill to increase the performance measured by v (Veres and Wall, 2000).

This approach assumes that some good controller structure is available from a priori analysis of the physical problem. Also relatively good initial θ_i is needed that already ensures stable feedback under the conditions $I(f, M_f)$. Although a priori design is essential, as result of that this learning mechanisms is the fastest from the ones considered in this sections.

4.2 Neural network based FB/FF tuning

Artificial neural networks can be used to tune FB/FF (feedback/feedforward) control action using multilayered perceptron, RBF (and recurrent networks, etc.(Kim and Lewis, 1998). These schemes can also be combined with associations of operating conditions. So control under an $f_i \in F_b$ can be made dependent on past associations of measurements form all sensors and successful control attempts. This way neural networks can be complemented by associative learning. The

resulting combined associative NN and dynamical control NN together form a self-organizing controller(Andreae, 1998).

The disadvantage of this approach is that training may take a long time. The greatest advantage is robustness and potentially superior performance over parametric methods as NN-based tuning may be able to make use of control opportunities that were not discovered by the human engineer designing the autonomous system. NN learning assumes that there is plenty of time and opportunity to practice operational modes.

4.3 Reinforcement learning

When some prediction system is maintained on the effect of considered control actions then exploratory and exploitative (greedy) actions can be taken in reinforcement learning. In addition to v that is a realtime performance measurement, we can introduce a performance estimate (=value function) $\hat{v}$ that is backed up as the measured state of the system evolves. Successful performance will then propagate into desirable state and learning slowly progresses. Temporal difference learning adjusts the anticipated $\hat{v}$ through exploratory and greedy actions (Sutton and Barto, 1998).

As the size of the state-space can be large this type of learning can be well used in cases where the $f_i \in F_b$ relies on strongly discretised description of a not too large state space. Continuous state dynamical control is difficult to learn by reinforcement learning. Instead reinforcement learning can be used with regard to the selection of the control initial parameters given environmental circumstances, i.e. initializations that later lead to success or failure under perceived environmental conditions. Using this way reinforcement learning can be very useful for continuous state dynamical control.

These methods assume that the autonomous system has plenty of time and opportunity to practice its skills in terms of OMs.

4.4 Learning for predictive control

One of the most powerful methods of control is when the autonomous system models its environment, plans its sequence of actions and immediately executes the first one (or first few). Then it senses changes in the situation and plans again and executes what is now needed under slightly changed circumstances. For the effectiveness of this generic method, the autonomous agent needs to maintain sufficiently good quality models of its environment. Also this online model should possibly be supported by redundant set of measurements to make the data secure. An useful method is to make this realtime modelling adaptive in the sense of (1) goal selection (2) its use of the set of i/o variables in various control tasks and (3) in terms of the realtime model maintained.

5. PLAYFULNESS OF LEVEL-1 AGENTS

Why do children play? Why do kittens play? Why do adults play games, and solve crosswords? Playing provides opportunity to practice skills. From the previous sections it is clear that one aspect of learning is to gather data for learning under non-dangerous circumstances. If a level-1 agent only executes live-mission tasks then it is likely that a huge amount of development effort will be needed to make it to operate safely, all OMs need to be performing very well from the very start and also at any switch of operational modes the initial condition must be strictly kept.

An alternative is to build a basic structure (using parametric controllers, NNs, self-organizing NNs, reinforcement learning structures, adaptive modelling, etc.) of each operational mode of a level-1 agent and then endowing it with the ability to randomly play with the purpose of improving its skills (=operational modes). This section provides a solution by adding a "playing operational mode" to the agents behaviour logic.

Let B be the simple behaviour logic formula of a level-1 agent $\mathbf{A}$. An extended formula $B_p = B \vee O_p$ is obtained by adding an operational mode O_p that takes now a supervisory role. When functional, the job of O_p is to monitor performance of each operational mode under various environmental circumstances and randomly choose from a set of playing activities and execute them by interfering with the normally used activity function a . Playing activities are designed such that they do not interfere with overall final goals of mission an they are always obtained as a modified portion of the B. This is achieved by suitable changes in the switching of a.

For instance in the above examples of the lawnmower, some playing activities can be obtained as follows: (a) practice docking for recharging; (b) practice fast and slow mowing on rough ground or high grass; (c) practice finding the boundary of the garden lawn; (d) practice activating the watering system and sensing of how it works. The purpose of the practice is not merely to repeat tasks as that would be useless: its purpose is to fine-tune the OMs of the level-1 autonomous system, i.e. to adapt the discrete and continuous control parameters in OMs. For that each OM must have a tuneable structure with learning mechanisms.

For the light autonomous aircraft some learning activities can be: (a) Taking off, doing a small round and landing straight away under various wind conditions; (b) practicing turns during normal flight while keeping flight path essentially the same (c) practicing taking photos by trying to keep the plane steady during exposure (d) practicing collision avoidance when noticing an object potentially in the way (needs human cooperation), etc.

Playing activities for O_p can be preprogrammed by the engineer developing the agent or O_p can also be generated automatically from B. Given the very simple nature of level-1 agents, a straightforward method is that the engineer designs a series playing activities for the agent. The task of O_p is then to seek out opportunities when these can be played, or depending on learning skills, to activate them. When playing activities are executed learning should take place automatically as all operational modes should be programmed with learning ability.

6. CLASSIFICATION OF LEARNING IN AGENTS

Leaning can be built into the algorithms of agents. It is however desirable to make them easily interfaced with control systems make them or automatically implementable by agents. This can be achieved if a set of routines are provided for learning that have agreed set of variables. Ontologies that describe class hierarchies and their properties and constraints can be used for this purpose. This section uses the formalism of the MOL (machine ontology language) notation for describing an ontology. Its notation is simple: lines with >> ... > mean class declarations and the deeper down a subclass is the more > are listed. @ means property of the class declared above and @@ means a constraint as expressed in a MATLAB Boolean expression. A property followed by a : means declaration of its class belonging that can be any basic MATLAB type or another class declared.

6.1 Ontology for operational mode learning

Modularization and standardization of learning algorithms for OMs. In the following ontology description the root class is "learning algorithm" its subclass is "action learning":

```
>learning algorithm
  @default settings: structure
  @v-function : char
  @t-function : char
  @control sequence : signal
  @sensed reactions : signal

  >>action learning
     @performance constraints : text
     @performance criteria : text
     @situation model associations : cell
     @sensory associations: cell

    >>>NN learning
      @NN-type : char
      @structural parameters : cell
      @weights : cell

    >>>modelled action learning
      @predictive model type : char
      @control optimisation method : char

  >>learning dynamical models
      @model-type : char
      @initialisation method : text
      @adaptation method : text
    >>>learning of parametric models
    >>>training of NN
    >>>learning of associative models

  >>learning spacial static scene
      @model type : char
      @method : text
      @model resolution: char
```

6.2 Interfacing of learning algorithms

Learning algorithms are interfaced to the OM via the

(1) measured sensor signals,
(2) v and t functions,
(3) actuator signals used.

These can make the algorithms in principle easy to replace and try as long as there ar built in mechanisms to adjust (1) array dimension and (2) initializations to the circumstances by association of experience. A shared database of learning algorithms as for the suitability of learning algorithms in various OMs is is therefore desirable. Such a data base can be made accessible to agents via the Internet. Control engineers who today design adaptive and learning algorithms, could place their work onto this database from where agents could pick them up for and try them to improve their performance.

Software that enables the fast development of agents based on behaviour logic, OMs and learning algorithms, can be developed in any programming language. Use of higher level languages such as MATLAB is however an advantage (Veres and Veres, 2006).

7. CONCLUSIONS

The main results of this paper is that a simple autonomous physical agent architecture based on behaviour logic, when combined with learning algorithms can achieve high levels of safety and reliability. This shows that the capability of higher levels of abstraction are not required from the agent to be able to show highly adaptive, planning based behaviour that is normally associated with higher levels of intelligence. To make APAs robust in natural environments, where unexpected circumstances often occur (and success so far was partial as shown for instance by the DARPA road vehicle challenges) it is shown that playful behaviour is essential at a training phase of agents. The paper has shown that adding playful behaviour is relatively simple in the behaviour-formula-based agent architecture proposed.

REFERENCES

Andreae, J. H. (1998). *Associative Learning*. Imperial College Press. London.

Kim, Y. H. and F.L. Lewis (1998). *High-Level Feedback Control with Neural Networks*. Vol. 21 of *Robotics and Intelligent Systems*. World Scientific. Singapore.

Meystel, A. M. and J. S. Albus (2002). *Intelligent Systems: Architecure, design and Control*. Wiley Series on Intelligent Systems. John Wiley and Sons, Inc.. New York.

Sutton, R. S. and A. G. Barto (1998). *Reinforcement Learning - An Introduction*. Adaptive Computation and Machine Learning. The MIT Press. Cambridge, Massachusetts.

Veres, A.G. and S.M. Veres (2006). *The Autonomous Control Toolbox for MATLAB*. **www.sysbrain.com**, SysBrain Ltd. Birmingham, UK.

Veres, S. M. and D. S. Wall (2000). *Synergy and Duality of Identification and Control*. Taylor & Francis. London.

Wooldridge, M (2002). *An Introdcution to Multiagent Systems*. John Wiley & Sons. Chichester.