Attentional Agents and robot control

S. Kasderidis* and J.G. Taylor

King's College, Department of Mathematics, Strand, London WC2R 2LS, UK

Abstract. A new approach based on attention control ideas is presented. We introduce the framework of Attentional Agents as the theoretical framework for constructing enhanced agents. We propose an approach where an agent processes multiple, possibly conflicting, goals in parallel and it uses a number of signals for obtaining information about its environment and own state. Additional requirements include a fast response in a changing environment and working with incomplete information. The proposed framework integrates in a seamless manner, sub-symbolic and symbolic processing, feedback control, conflict resolution, processing of multiple concurrent goals and action generation. We apply the general framework to the development of a robotic agent and we provide results from simple simulations where usage of multi-modal information and competing goals are present so as to illustrate the general ideas.

Keywords: Attention control, robotics, autonomous agents, Attentional Agents

1. Introduction

Nowadays there is a growing interest in constructing advanced systems that exhibit human-like abilities. Examples include the Ambient Intelligence paradigm in Computer Science and that of Humanoid Robots in the Robotics field. We propose a set of ideas that originate from the way human brains cope with complexity. The general problem has many requirements. For start it is the integration of advanced perception, reasoning, and action generation abilities in a coherent whole. Most of the new systems are expected to operate in an open and non-stationary environment, open both in the functional level (as new abilities could be introduced in the initial specification) and in the problem level (as problem specification in the real world rarely remains constant). The openness requirement is further complicated by the fact that the systems are expected to provide some “sensible” response even when encountering cases which are rather far from their initial specification. In the extreme case, they should be able to handle novel situations. The problem of novelty requires learning abilities of some sophistication. Learning in general is necessary even in the case of micro-adjustments (adaptive control). The environments in which a system will operate are of high complexity and the information about the state of the environment is typically incomplete. In many application areas fast system response is a necessity; principally for survival, as well as due to performance considerations. At the same time, the users expect the system to satisfy many concurrent user goals, possibly conflicting ones, so as to provide a useful service. Thus there is a need to somehow prioritise and schedule the overall service execution taking into account these global constraints. This naturally leads us to consider the way that overall action generation is produced and it is coordinated. The above requirements provide a coarse outline of the complexities that exist in the real world and the problems that must be tackled before producing systems that successfully address them. Context capturing and usage is also assumed as it affects both the interpretation of information (through perception) and action generation.

The field of Agents and Multi-Agent systems studies the above types of problems. In this paper we will present an agent architecture which is brain inspired and handles the above problems. The main characteristic of the architecture is that it introduces the idea of attention as a control mechanism both at the local and global levels. However, we will only discuss the single
agent architecture in the processing mode here without any extension to either the learning or multi-agent aspects. These aspects will be discussed elsewhere. The discussion will be given in a general setting independent of application domains even though we will use an example from the Robotics field in order to aid comprehension. The paper will concentrate in developing the conceptual architecture rather than presenting the implementation aspects in detail.

Our architecture shares commonalities with other established agent architecture in that it uses multiple goals as in the PRS system [1–3], and action generation takes place in parallel as in the Subsumption architectures [7]. Similarities to the way that feedback control is achieved through generated action exist with the Touring Machines and InteRRaP architectures [4–6]. The architecture is functionally divided in four layers resembling the approach of Marr’s vision theory [8]. However clear differences can be identified in the internal representation structures used. Typically these are symbols in the aforementioned architectures, whereas in the Attentional Agents architecture these representations are of sub-symbolic type. Symbols arise at the end of the internal processing chain. In the robotics community the attention-based processing has been used by Koshizen and colleagues in the context of constructing cognitive humanoid robots with complex kinematic capabilities [17,18]. Another relevant robotics architecture is the Saphira Architecture [24]. We share with it a number of common features such as goals, perceptual and action systems in multiple layers and conflict resolution. However, there are distinct differences with the principle ones being those of attention control, learning and lack of a central representation space. The main feature of our architecture is of attention control; attention has received much interest recently in the neurosciences and cognitive communities; especially its global aspects [9]. Below we provide our motivation for using an attention-based approach in our work.

Attention is arguably the highest-level control system in the human brain. It allows a system possessing it to be able to reduce the problems presented by distracters in a noisy environment, and to prioritise tasks to be solved so that less important ones, elicited by distracting cues, can be avoided. An important component of such attention control is the creation of suitable goal structures, regarded as representations in the human frontal lobes. Much experimental research has now been performed on attention. It involves two component systems in the human brain: A plant, composed of either sensory cortices, for initial analysis of inputs, or motor cortices, for output to motor systems; A control system for filtering out distracters in the plant and amplifying desired targets for higher level processing. The control system, in the case of motor response in humans, has been shown to consist of inverse and forward models, involved in creating control signals for the plant (inverse models) or providing prediction of the state of the plant (forward models) [10,11]. Both of these are recognised in suitably subtle motor response paradigms.

It has been proposed to apply a similar engineering control approach to attention [12–15]. We have extended the notion of attention processing from a model of the human brain to the more general context of attentive agents [16]. The attention control system allows for considerable flexibility in response of the agents, both in an on-line manner as well as through adaptation to changing circumstances of humans interacting with the agents.

In Section 2 we present an overview of the main ideas of the Attentional Agent architecture. In Section 3 a specific task environment is introduced to test the proposed architecture. A discussion on how to use the general concepts to construct a robotic agent is given there. In Section 4 we provide the universal architectures of level 3 and 4 goals, so as to enhance the comprehension of the example of Section 3. In Section 5 simulation results are presented. We conclude, in Section 6, with a discussion about the main features of the architecture and further extensions.

2. Attentional agents architecture

In Section 2.1 we will present the main ideas of the Attentional Agent architecture. Topics of special interest will be discussed separately in Sections 2.2–2.3.

2.1. Overview and general ideas

A high-level overview of the architecture is presented in Fig. 1. The figure has two complimentary views: a structural (A) and a functional (B) decomposition.

In the left pie chart of Fig. 1 we include the main components of the Attentional Agents architecture. The architecture is a goal processing system. The four global components are:
1. **Attention Control.** The attention idea is used in a multitude of ways. First it serves as a local control mechanism by which a modality can control its associated sensors and actuators. This mode includes both the sensory and motor aspects. Second, it is used as a conflict resolution strategy among competing goals inside the same family sub-tree. Third it is used as a global conflict resolution strategy among families of goals.

2. **GoalsTree.** This is a (distributed) tree structure that organises the goals present in the system in families of goals that are interrelated. The key idea behind this concept is that it partitions the action space (i.e. the space where the agent’s responses are executed) into mutually exclusive sub-domains. Inside each sub-domain families of goals are competing for the right to execute their action. Only one family is allowed to execute its action in every processing step in a given action sub-domain.

3. **Computational Model.** This is a set of rules that guides the proper sequence of processing of the goals of the GoalsTree including the scheduling of their action generation.

4. **User Profile.** This is a (distributed) database that contains model parameters, models and classification and decision thresholds. The models are used internally for state classification and action generation and they are associated with a specific user & task environment combination. The agent may possess a number of different profiles corresponding to different contexts for the same user or additional human/artificial agents & tasks combinations. These profiles can be supplied at an initial stage or built on the fly in novelty situations.

The right pie chart in Fig. 1 represents the main components of a level 4 goal (global modality or otherwise processing and control modality). More discussion will be given in Section 4.2.

The principle entity present in the architecture is the goal. The goal is a self-contained unit of execution, which includes a parameterization of a target state, as well as the necessary machinery for achieving such state. The machinery is called a service. The Attentional Agents architecture is a hierarchical goal processing system, which uses the idea of partitioning both the input and output spaces into suitable modalities. The second view in Fig. 1 represents the functional decomposition of the Attentional Agents, i.e. the Processing Levels. There are four levels in the architecture. These are:

- **L1:** Sensors & Actuators. In this level typically data acquisition and action implementation takes place.
- **L2:** Pre-processing. In this level necessary pre-processing tasks are executed before using the various signals further.
- **L3:** Local decision. In this level it is implemented the division of the input space into modalities, so as to handle the scaling problem. Each modality is a self-contained system that integrates perception, state evaluation, attention control and action generation, for example vision, auditory or tactile processing. They should be interpreted as complete functionalities of a complex agent. The level 3 goals are also called local modalities or domain experts.
- **L4:** Global decision. In the general case a set of domain experts it is assumed interrelated and thus they must be coordinated in order to produce a coherent action. The global modality acts in this way as the family leader for the sub-
tree of the domain experts. The level 4 goals implement the idea of multi-modal perception (data fusion) and multi-modal action generation (fused decision). In other words they implement the composition of goals hence achieving task decomposition.

The services of both types of goals are implemented by a universal architecture (discussed later in Section 4) which consists of a number of modules partitioned in three distinct sub-systems. These are the State Evaluation, the Action Generation and the Attention Control ones.

Processing of conflicting goals is achieved by the introduction of a competition mechanism based on attention. The idea is that not all goals are equally important in a given moment and thus priority should be given to the ones that capture attention in a stronger way. Attention thus shares the precious computational resources of the system in the goal set that needs attending to in a given moment. While the attended goals are processed the rest of the goals are suppressed and wait for their chance to re-acquire processing resources. The goals are organised in suitable families that compete for the processing resources and so achieving their action generation.

The GoalsTree partitions the output space. Overall action generation takes place in parallel in all cells of the action space. If conflicting goals (or families of goals) exist for a given cell then the winner is found through attention competition as it was described above. Another key idea is the processing mode of a service and the interplay with the attention mechanism. When a service operates in a flawless manner, it exists in the so-called automated mode. When a service produces errors in its response then we assume that it enters in one of the learning modes. There are three learning modes in the system depending on the magnitude of the service’s response errors.

The Attentional Agents architecture has the following main characteristics:

1. **Process & Control Architecture.** The architecture as it is defined is a conceptual structure that attacks fundamental conceptual problems. It is not coupled with any specific implementation technology as in hardware and middleware platforms, programming languages, etc. It provides a general processing model where attention control is fully integrated. It is also a template where various methods that implement the State Evaluation and Action Generation sub-systems can be inserted.

2. **Goals-oriented.** The primary conceptual blocks present in the system are goals. There are many types of goals. Each one is supplemented by its corresponding service procedure, which is the ‘recipe’ on how the goal can be satisfied. Further discussion will be given in Section 2.2, as well as Section 4.

3. **Partitioning the input and output space into modalities.** It was discussed above.

4. **Adaptive on the user.** A major requirement in many application domains is the ability to adapt to the agent’s owner for achieving better overall performance over time. Adaptation is used for fine-tuning model parameters that are set in design time and/or learning new behaviours through reinforcement learning.

5. **Integrating symbolic and sub-symbolic processing.** Most current agent architectures use as internal representations symbols, usually predicates of first order predicate calculus, even though they receive numerical values from sensors. Each modality in the Attentional Agents architecture uses the same sensor signals but it builds usually sub-symbolic internal representations. Only the final state classification representations are symbolised. This state evaluation processing could be thought of as a chain of abstractions that takes place internally in the modality. Based on the recognized state (as a symbol now) action generation is created through the usage of rules. In this way it is possible to integrate symbolic and connectionist approaches. More discussion will be given in Section 4.
6. Modular and Distributed. The Attentional Agent is an entity that by its construction is modular and distributed. The primary reason is that it is envisaged to be able to organise collections of artefacts in an Ambient Intelligence environment. The modular nature is witnessed by the internal structure of each modality (which in turn is based in an engineering control framework) whereas the distributed nature is facilitated by its distributed GoalsTree structure that distributes different goals in various artefacts for concurrent processing.

7. Open-ended system of goals and services. The Attentional Agent could be realised with an initial set of goals and corresponding services. However, it could be extended further by addition of new goals and services, either in run-time or through initialization. In this way new functionality can further enhance the agent’s capabilities satisfying new, currently unforeseen needs.

8. Advanced self-management and robustness mechanisms. Due to its distributed nature the agent suffers from the increased danger of malfunction due to loss of a contributing artefact (processing node). Other hardware, software or communication conditions may arise that would induce a failure in the system. For this reason attention control is used as a proactive strategy for maintaining agent integrity. More discussion will be given in Section 2.3.

The main requirements that led to the development of the architecture were discussed in Section 1. Note that not all of the above features are possibly required for applications in the Robotics domain. The distributed nature of the agent is an obvious example and possibly the ability of adding additional goals and services could be irrelevant in a vertical domain application.

2.2. Goals

In the Attentional Agent architecture goals are objects that encapsulate together a target state representation with the necessary computational procedure that satisfies them. This overall procedure is called a service. The processing requirements of the service procedure partition the goal space into various classes of goals. Complementing the service function, there are two additional functions. The evaluation function returns a Boolean result if the goal state has already been reached. The impossible function returns a true answer if the environmental conditions have been changed so as to result in making the goal unachievable. The service function checks in its control loop both of the aforementioned functions and it is executed only when the target state is not reached and it is still possible to be achieved. The following types of goals exist:

1. Primitives. These are the simplest building blocks present. They offer necessary functions for the system operation. They are always leaf nodes of the GoalsTree structure. They are not learned, but assumed given.

2. Atomic Goals. These goals represent the requirement to reach a target state. They are further sub-divided into two classes:

   (A1) Non-modality Goals. These are goals that could be satisfied by simple processing. They use typically symbolic representations and their service is implemented using symbolic manipulation so as to produce the desired effect. They may require or not the usage of any sensor signals. They can be used to implement other architectures for a specific sub-problem.

   (A2) Modality Goals. These are in the level 3 modalities that were introduced in Section 2.1 and they will be discussed further in Section 4.1. They are used for handling complex control problems. They receive sensor input and they generate suggested actions. The generated actions represent the agent’s response in the perceived situation for the given modality.

3. Composite Goals. This type of goal serves as family parent for a sub-tree of interrelated goals. The children could be either composite goals themselves or atomic goals. They represent a natural metaphor for constructing complex goals. They are further divided into:

   (C1) Non-modality Goals. This type of composite goal is used to group together interrelated children but which are independent in the generation of their actions. In other words the overall family action would be the sum of the individual children’s actions. The parent simply calls his children sequentially to execute their respective services and thus a component of the solution for the overall (family) action generation is provided by each child’s action.
(C2) **Modality Goals.** In contrast with the C1 type, these goals exist for coordinating children that in general are correlated and thus the overall action generation is not simply the sequential execution of their actions as in the previous case. They have already been introduced as the level 4 global modalities in Section 2.1.

It should be made clear here that only A2 and C2 categories implement integrated attention control in their processing. Their detailed architecture will be presented in Section 4. The A1 and C1 categories are provided for compatibility with other architectures (e.g. expert systems) and are used so as to integrate services received from external sources in one coherent attention control based architecture inside the agent. In this case the corresponding attributes that affect the attention mechanism have default values for a given application, see discussion below.

As we have already mentioned the goals are organized into competing families for the right to produce an output in a given cell of the action space. The exact rules that govern action generation are part of the Computational Model. However, due to space limitations we will provide a discussion of it elsewhere. Here it suffices to say that there is one more classification of goals according to the processing policy that is applied by the Computational Model protocol. There are the following processing classes:

1. **BASIC.** A goal belongs to this category when its failure implies that the rest of its sub-tree cannot be executed successfully due to logical consistency constraints. It is reserved for agent level goals, which collectively implement the self-management mechanisms of the agent and guarantee its consistency and survival. An example includes the case that the goal “Enable” fails due to lack of power in a robotic agent. Thus executing any other goal that belongs to its sub-tree does not make sense under this condition. If such a goal fails typically corrective action must be taken immediately. This type of goal has usually infinite lifetime.

2. **CONCURRENT.** A goal is concurrent if all of its children are executed concurrently, without any specific ordering requirement.

3. **SEQUENTIAL.** A goal is of this type if there is a necessity to execute its children in a specified order.

Now we come to the second idea that is related to the necessary conditions for service execution to be considered successful. The service execution is decomposed into two axes: those of completion and performance. See Fig 2.

The idea for such decomposition is that if a service fails to complete (in the sense of the computational processing) clearly it cannot be successful. However successful is considered only when it completes and then produces a ‘good’ (in the sense of the high performance) response. If the service completes but produces a poor output it is considered as failed (due to performance reasons). Service failures are handled in two ways:

- If the service fails due to completion reasons, then its exception logic is executed for recovering the service for the next processing step. The exception logic exists in the Exception Handlers of the Rules modules of a goal. More discussion will be given in Section 4.1.

- If the service fails due to performance reasons, then the agent enters for the failed goal to one of the three learning modes, modes 1–3 below.

The last point introduces us naturally to the **Processing & Learning modes** of the architecture (mentioned already in 2.1):

- **Mode 0:** Normal processing, no errors. No learning is present. It is also called automated processing.

- **Mode 1:** Micro-corrections are needed to the various internal models due to the drifting non-stationary environment. Corrections take place on-line.

- **Mode 2:** Major corrections in the models are needed. Data are collected for re-estimating the models for the new environment. Estimation takes place offline in general.

- **Mode 3:** Novel situations are encountered. We need to build new models and concepts from scratch through reinforcement learning. Reinforcement could either come from direct user input, user observation for positive cues or though agent exploration and voluntary actions.

One should note here that due to existence of distinct goals the agent can operate in the automated mode for some goal sub-sets while it can enter in one of the learning modes for other sub-sets.

There are a number of attributes that describe a goal. However, for economy of presentation we will not
present them here. It suffices to say that the attributes can be partitioned in sub-sets that describe, the goal ID, the partitions of the input and output spaces, goal type, termination conditions, processing mode (one of 0–3 above) and the attention control support attributes. We present here only the last sub-set as it is directly related to the operation of the attention control system that will be presented further in Section 2.3. The related attributes are:

- 1: Weight
- 2: Action Index
- 3: S-Attention Index
- 4: M-Attention Index
- 5: Adaptive Parameters

Regarding the termination attributes we should mention that depending on the semantics of the goal, a goal may exist for an infinite time (as in the BASIC category), for a given time (measured by the execution counter), terminated by the user (where it applies) or it may terminate when its service fulfills its function. On termination a goal is deleted from the GoalsTree.

We have now arrived to the point where we can explain the Attentional Agent’s operation. This is schematically presented in Fig. 3.

The overall action space is divided in mutually exclusive sub-domains. In each sub-domain families of goals compete for the right to apply their generated action. Inside a goal family members could be competing as well. The family leader monitors their competition. Families compete through their leaders based on the leader’s action indices. In the local level of a single goal attention control is used for feedback control of the sensors, capturing failed expectations (sensory attention) and monitoring actions execution generated by the goal (motor attention). Further discussion on the usage and roles of attention in the architecture will be given in Section 2.3.

2.3. Attention

In this section we will summarise the main ideas of attention control as they are used in the architecture. Let us start by explaining the various Attention Modes in which the Attentional Agent operates:

1. Global mode. In this mode we have competition among goals for the right to execute their action. We distinguish two cases:

   (G1) Family members’ competition. It could be the case that for a given family some members could compete for accessing the same output cell. In the case that the semantics of the family dictates that only one of them must have access in any given time to the target cell, then they compete directly using their action indices.

   (G2) Goal Families’ competition. In the more general case families of goals, as they are represented by their leaders, compete for the access right in a given cell. A leader effectively represents the whole family’s action generation as his service calls recursively the services of his children. A family is declared winner when the leader’s action index is larger than the corresponding indices of the competitors.

2. Local mode. The local mode takes place in the scope of a single goal. Typically the goal is of a modality type, either a domain expert or processing and control modality. In the local mode attention fulfills three functions:

   (L1) It captures unexpected states arising (sensory attention)

   (L2) It captures failures in the goal’s service execution (motor attention)

   (L3) It controls through feedback the sensors so as to enhance the relevant information and discard inappropriate one. This case also includes the feedback control of the pre-processing level for a given sensor. An example is visual attention to a part of a camera image [19].

The Attention mechanism provides together with the Computational Model a complete processing protocol, which determines the order in which the goals are processed (i.e. by calling their service functions), the value of the action indices that are generated and the competition winners. After winners are established we can apply their generated actions to the action cell that they are assigned to and thus implement the agent’s response to the current state for the given processing step. It is important to note here that attention effectively switches the mode of a goal from the automated processing mode (i.e. mode 0) to one of the three learning modes. This is accomplished due to failures that are encountered in the predictions that are generated by the attention sub-system. More discussion will be given in Section 4.
The machinery for implementing the global attention control scheme is based on attributes 1–5 of Section 2.2. It also includes an Attention Controller module and children Action Indices. In particular the following formula applies for the action index of a goal:

$$AI = \left \{ \frac{(W + S_{AI} + M_{AI} + \sum_{i=1}^{N} C_{AI_i} \cdot \delta(i, \text{ContributingGoal}))}{(3 + \# \text{ContributingChildren})} \right \}$$

(1)

or more general (in case adaptive parameters are used for learning purposes to achieve greater flexibility):

$$AI = \left \{ \frac{(W_1 \cdot W + W_2 \cdot S_{AI} + W_3 \cdot M_{AI} + W_4 + \sum_{i=1}^{N} C_{AI_i} \cdot \delta(i, \text{ContributingGoal}))}{(3 + \# \text{ContributingChildren})} \right \}$$

(2)

where in the above formulae: $AI$ is the goal’s action index, $W$ is the intrinsic weight of a goal, $S - AI$ is the goal’s sensory attention index, $M - AI$ is the goal’s motor attention index and $C - AI$ is a child’s action index. The symbol $\delta(i, \text{Contributing Goal})$ is the Kronecker notation for goal $i$. It has the value of 1 only if goal $i$ is a contributing goal; otherwise is 0. Let us explain briefly the concepts involved.

The intrinsic weight of a goal is a concept that captures the importance that the given goal has for the user or the agent. So a higher weight contributes to a higher overall action index and thus to a higher probability in
winning the competition with another goal. The sensory attention index represents the fact that an unexpected event took place in the environment. This event could indicate a change in the environment thus it is a prudent strategy to deal with such events first. The motor attention index represents the fact that a service failure took place and this must be taken into account in order to provide typically more computational resources, effort or otherwise. Motor attention is also crucial in learning new action sequences in novel situations through reinforcement learning. The sum of the children’s action indices offers a way to implement the idea that attention propagates from a source node in the family tree to the family leader, so as to contribute to the overall family’s competitiveness. It is also natural to assume that if a child is somehow affected, its parent typically is affected as well; the parent’s operation depends on the child’s response. The goal’s action index is the binding concept that brings all the above attention contributions and intrinsic value of a goal together. In the simplest case (i.e. processing mode only) the simpler formula Eq. (1) applies, while in the general case of Eq. (2) learning is also assumed; this is interleaved with automated processing. Formula Eq. (2) reduces to Eq. (1) by setting the adaptive parameters (vector W, attribute 5) to 1 for all components. We should note that by definition all the above attention components (except of the adaptive parameters) are bounded in the unit interval [0,1]. This is the reason for the normalization factors in Eqs (1) and (2). In both definitions the factor # (number of) contributing children appears. The definition of a contributing goal is given below and is an effective mechanism for propagating attention contribution to the family leader from whichever family member these contributions arise from.

Def: A contributing goal is a goal that either raises some attention event directly or it has a child that raises such a direct attention event even though the goal itself may raise no attention events at all. This definition includes the case where a goal raises attention events itself and has a contributing child as well.

The way that the (local) sensory and motor attention indices are defined is further discussed in Section 4.1. Before closing this section let us present the various roles which attention plays in the Attentional Agent architecture. These are:

1. **System Configuration Maintenance** (”Watchdog”). In this role attention control is used to maintain the integrity of the agent. The key idea is to provide alerts for pending failures and thus the agent to take proactive action to protect itself.

2. **Adaptive (User) Monitoring.** In this role attention control (in sub-mode (L3)) dynamically adjusts the sensor’s resolution so as to capture efficiently the required information. If multiple goals access the same sensor and there is conflict about the set sampling rate, this conflict can be resolved by setting the sensor in the highest requested rate if this is sensible for the application domain. If this is not sufficient, specific rules can be defined or use attention-based competition.

3. **Creation of new models and concepts in novel situations.** In this role attention is crucial for agent learning through reinforcement when it encounters novel situations. Attention is used for building and enhancing the correlations among the created models, the observed state and the given reward.

4. **Action support.** In this role attention enhances action execution by monitoring the executed actions and introducing corrections in a feedback motor manner so as to achieve the desired effect.

The key idea behind attention control is to capture unexpected states. These states could arise due to environmental changes signaling some state transition into a new steady state (in sensory attention) or due to deviations between the expected and actual results of an agent’s action on the environment. The latter deviation is usually due to failure in the motor models of the agent rather than the very fast change of the environment in the time that it takes to complete the action.

3. **Robotic agents**

In this section we use the framework of Section 2 for creating a simple robotic agent. We do this by providing a simple scenario in Section 3.1, amenable to simulations, and a discussion on the mapping of the Attentional Agent architecture to a robotic agent in Section 3.2.

3.1. **Task scenario: route planning in a dynamic environment**

We assume for our simulations a scenario that combines two tasks. One is route planning while the other represents the robot’s main function that of carrying some loads from one point to another point inside a factory floor.
The route-planning problem, in simple terms, involves the discovery of an appropriate path for the robot’s movement in an environment where a number of stationary and moving objects exist. Stationary objects typically include household items such as furniture and moving objects include other robots, humans sharing the environment and other less ‘intelligent’ but mobile equipment.

Let us now present a simple scenario that will demonstrate the route planning problem and the use of attention-control. Assume that a worker robot is inside a factory floor and its job is to move some item of interest from a point A to a point B. Assume further that these two points are at two distant sides of the floor and that the floor is of rectangular shape and it is divided in a X-Y grid (GridWorld). The robot moves in this grid in a forward direction. When the robot wants to change direction it performs a turn around its z-axis. The turn step is 45 degrees. The turn could be either left or right (to the local system of coordinates). The GridWorld is shown in Fig. 6.

Terminal C is the recharging station of the robot. Sobj $x$ is the name of a stationary object $x$. Mobj $y$ is the name of a moving object $y$. The figure shows a possible configuration of the GridWorld with three stationary objects and two moving ones. Realisations for the trajectories of the moving objects are also shown with dashed lines.

The problem that we face is how to move from point A to point B and possibly C avoiding collisions with stationary and moving objects. We can approach this problem assuming an infinite sensing range or a partial one, e.g. only 5–10 cells ahead from each direction. We also assume that from time to time the robot must return to its charging station for power re-supply.

3.2. Constructing an attentional robotic agent

In this section we explain how a robotic Attentional Agent can be constructed that can tackle the problem that was presented in the previous scenario. We assume the following:

- The robot carries three sensors. These are the power sensor, the proximity sensor and the position/speed sensor. For simplicity we assume that the proximity sensor provides information for both the current position and speed of the other objects. It has a 360 degrees view of the environment.
- A single actuator exists which controls the speed and the direction of the robot.
- Inside the robot a computational system implements an Attentional Agent architecture as it was described in Section 2. In particular we assume that the following goals exist:
  - Exist
  - Enable
  - Transport
  - Collision Avoidance
  - Goto A
A GoalsTree snapshot is shown in Fig. 7. Let us explain now briefly what takes place in Fig. 7.

There are at present two BASIC goals. These are the Exist and Enable goals. The Exist goal typically maintains the functional and structural consistency of the robot, while the Enable goal makes sure that necessary resources such as power are available. They are declared as BASIC so as to disable the main functionality when one of them fails because it is not logically consistent to continue processing in these circumstances. In Fig. 7 it is shown a case where the robot is about to exhaust its power. The Enable goal is a level 3 modality goal which uses the power sensor so as to measure the power level. If the power is about to fail an attention event is thrown and thus action is taken by the Handler Rules. The rules specify that a new transient BASIC goal (i.e. execution counter is 1) is created. This is shown as a dashed line. The goal is called Charge. This instructs the robot to return to its re-supply terminal and recharge. After recharging the robot follows his previous “program” which is specified by the Transport goal. The Charge goal is deleted from the tree.

There are two high-level (user) goals. These are the Collision Avoidance and Transport goals. The Transport goal captures the main function of the robot, i.e. of moving loads from one point of the factory to another. It is a level 4 non-modality SEQUENTIAL goal. In essence a SEQUENTIAL goal implements a high-level procedural plan. The plan is decomposed in four independent sub-goals. They are the Goto A, Get Item, Goto B and Leave Item goals. Because the goals are independent the goal is non-modal, i.e. no special action fusion must be performed except of executing the constituent goals and thus achieving the overall Transport goal. The transport goal exists only for a finite
number of executions, e.g. 10. Thus after the robot having executed the Transport command for 10 times the goal will be deleted and the robot will remain idle until a new command is given.

The children goals are self-explanatory. We assume that the Get Item and Leave Item goals are satisfied by primitives, which are not shown in the figure. Clearly in realistic settings grasping and handling actions are needed, but this is not discussed in this analysis so as to keep the example simple. We only analyse further the Goto goal. This goal provides a service that moves the robot to a specified floor location. It takes as argument a specific location/point description. Here A indicates the loading point (Terminal A in Fig. 6) whereas B is the destination point (Terminal B in Fig. 6). The Goto goal is a SEQUENTIAL non-modality atomic goal. It is implemented by calling two further sub-goals in the order indicated in Fig. 7. The two sub-goals are Route-Planning and Move. The Route-Planning is a level 3 domain expert goal. Move is a simple primitive goal which given a calculated new position (provided previously through route planning) it simply acts on the actuator so as to move the robot to the new position.

Let us analyse the Route-planning goal. Its structure is given in Fig. 4. The purpose of the route-planning goal is to find a valid new position so as to move the robot closer to the target position specified. It collects data from the position and proximity sensors. The Rules module uses the position of the robot and the positions of the other moving and static objects in the sphere of observation to calculate a new position that the robot could move towards in the next step. The State Module is trivial in this case: It is the Identity map for the state vector. The new position should in the robot closer to the target position specified. It collects data from the position and proximity sensors. The Rules module uses the position of the robot and the positions of the other moving and static objects in the sphere of observation to calculate a new position that the robot could move towards in the next step. The State Module is trivial in this case: It is the Identity map for the state vector. The new position should in the robot closer to the target position specified. It collects data from the position and proximity sensors. The Rules module uses the position of the robot and the positions of the other moving and static objects in the sphere of observation to calculate a new position that the robot could move towards in the next step. The State Module is trivial in this case: It is the Identity map for the state vector. The new position should in

To guard against the unknown actual paths of other moving objects we use the Collision Avoidance goal. This is a domain expert goal, which is declared to act on the same action sub-domain as the Move, Route Planning and Transport goals, this latter of the robot actuator. The goal weights are set up in such a way so as to allow the execution of the Transport goal if no pending collisions exist. If collisions are detected then the Collision Avoidance goal calculates a high Sensory Attention Index and thus an Action Index. Thus in cases of collisions the Transport goal (and its family tree) is suppressed by the Collision Avoidance goal which implements the policy of keeping the robot still, i.e. in the same position that it was in the previous processing step. Let us now give some further details on this goal. This goal uses as state vector information coming from the robot’s position/speed sensor and the same variables for all the observed moving objects from the proximity sensor. The State Module evaluates the relative distances and provides a classification as to the dangerous level of proximity. There are three levels: SAFE > 5 cells, CAUTIOUS between 2 and 5 cells and DANGEROUS < 2 cells. A change in the classification history raises sensory attention events. The Rules module implements the action of keeping the robot still if the goal wins the competition. There is no Forward Models present. The Observer model uses a neural network model that predicts future moving objects positions and headings so as to be able to raise sensory attention events in case that the Observer’s predictions are far from what is observed.

4. Level 3 & 4 universal architectures

In this section we present the universal architectures of the level 3 and 4 goals giving emphasis on the description of the constituent blocks. They follow an engineering control approach. These include the details of the local attention control that it was described in Section 2.3.

4.1. Level 3 architecture: domain experts

In this section we will present the architecture of the domain experts (A2 type). This type of goals is designed to handle particular problems. Its architecture is shown in Fig. 4.

The architecture is decomposed in three main subsystems. They are:

− The State Evaluation sub-system,
− The Attention Control sub-system and
− The Action Generation sub-system.
4.1.1. State evaluation

We assume that a number of sensors provide signals to the modality. The module, which collects the sensor signals and builds a state representation, is called the State Module. The module uses a number of classifiers in order to produce an output classification of the recognised situation. There are at least three state representations present. The first is simply the Cartesian product of the (pre-processed) sensor signals and it is called the Native Representation. The classifiers use as input the Native Representation and internally produce an intermediate form that is called the Intermediate World Representation. Finally they output a class label on the recognised state in a symbolic form, which is called the World Representation. Note here that the Intermediate State Representation is not necessary to be present, as it depends on the form of the classifier method chosen, but it is a byproduct of the classification (abstraction) process. Note also that the Native and the Intermediate representations are in general sub-symbolic whereas the World Representation is a symbolic one (which includes the fuzzy predicate case). Typically some compressed/fused state is built from all the participating signals so as to facilitate state evaluation. Well known techniques for achieving this include, linear and non-linear PCA algorithms, ICA, Fourier transforms, wavelets, etc.

Two classifiers are used currently in the architecture. Both use neuro-fuzzy models [20,21]. One type is suitable for encoding existing expert domain knowledge for the problem at hand while the other is suitable for discovering classification rules for domains for which no such rules are currently available. The latter can be used as a data-mining approach as well. The overall state classification is generated through a committee approach [22]. More details will be presented elsewhere. We also note here that the classifiers are supported by self-evaluators that assess the level of confidence in their outputs and that historical traces of the various state representations are assumed known.

4.1.2. Attention control

Let us now explain the Attention Control sub-system. This consists of a number of modules. The modules are:

1. Observer Module. The observer module implements a state prediction map for a given sensor state. It is used so as to realize a concept of ‘expected’ state, which is compared against the observed state in the next processing step. It uses the historical trace of the Native Representation for the given sensor. It implements the following map:

\[ \text{Sen}(t) = f(\text{Sen}(t-1), \ldots, \text{Sen}(t-T)) \]  (3)

where \( \text{Sen}(t) \) is the vector value of the given sensor at time \( t \). It is also assumed that the historical trace includes values up to some lag time \( T \). Map (3) can be implemented in many alternative ways. In the current version it is implemented by using appropriate neural networks. Typically the signal is encoded using a Takens embedding representation [23]. We should clarify here that there is a separate Observer module (model) for each one of the sensor signals present. In general map (3) can include values from other sensor signals so as to exploit cross-correlations if they are useful.

2. Monitor Module. This module is responsible to evaluate the deviation between the observed state and the predicted state (from the Observer) for a given sensor. Appropriate metrics are used for this purpose. Their definition is application dependent. There, a measure of similarity (or closeness), \( E \), is calculated and this is compared against a given threshold. The threshold is a user specific value and it is given by the User Profile. Usual metric choices are \( || \cdot || \) and \( || \cdot ||^2 \) metrics. Other possibilities exist. If the error measure calculated is above the threshold value this indicates the existence of an event of interest that needs attending to. The above concept is represented by a Sensory Attention Index for a given sensor (S-Al-Sen\(_i\)). This index is defined as:

\[ S = \text{AttentionIndex} - \text{Sen}_i = 1 - Pr(E > \delta) \]  (4)

where \( E \) is the Monitor error measure and \( \delta \) is the aforementioned threshold. \( Pr \) is the probability that the Error is larger than the threshold. We assume that a suitable error distribution has been selected. This definition implies that the more ‘improbable’ is an event the higher is the index that it creates. This is consistent with the fact that unexpected changes in the state imply in turn a rather large error deviation between the prediction and the current state. As a result higher priority should be given to such events. We note here that definition Eq. (4) is a suitable general formula for a number of application domains. However, one can introduce different definitions of the Sensory Attention Index as far as they are consistent with the overall idea of an Attention Index and they are bounded in the unit interval [0,1]. Formula Eq. (4) represents the attention index created due to a given sensor channel, \( i \), and in the general case we have a number of attention indices due to sensors. This is in general a vector of \( N \) components where \( N \) is the number of sensors. The index with the largest
value indicates the sensor that needs the highest priority for attending to. Typically appropriate action follows; an example is the change of the sensor’s sampling rate so as to enhance important information. The question now is how we define an attention index due to sensor signals in general. There are two possibilities. These are given by formulae Eqs (5) and (6) below.

\[ S - \text{AttentionIndex} - \text{Sen}_i = \max_i \frac{S - AI - \text{Sen}_i}{(7)} \]

or

\[ S - \text{AttentionIndex} - \text{Sen} = \sum_i S - AI - \text{Sen}_i, \text{modl} \]

(6)

It depends on the application domain, which one of the two (or other intermediate cases between Eqs (5) and (6)) must be used. Formula Eq. (5) is a natural definition as it declares as overall attention index due to sensors the largest one. Formula Eq. (6) however, uses the idea that the overall sensory index created by the modality must somehow capture the fact that multiple events in different channels are created concurrently so it is prudent to create the largest possible index indicating the fact that there is multiple activity. After all the sensory attention index of the modality will be forwarded to the family leader, thus events that somehow create multiple prediction failures typically should be considered more carefully. An observation is due here. In cases that the sensor signals are somehow correlated, for example in a biometric modality, usage of Eq. (5) is probably best as every signal’s failed prediction carries the same information as that of a single event having occurred but it is registered through multiple channels.

In case that the sensor signals carry independent information usage of Eq. (6) is typically more appropriate. However, use of correlation analysis to separate all the independent components in a given modality, and then use of Eq. (5) for such components, appears optimal.

There is a second source that can trigger attention events. This is the deviation of the historical trace of the classified state from the currently classified state. An appropriate formula is given by:

\[ S - \text{AttentionIndex} - CL = \text{Length} - \text{of} \\
- \text{window} - \text{of} - \text{calm} - \text{period/Total} \]

(7)

where the attention index due to the change in the classification history (S-AI-CL) is given as a ratio of the length of the period between two classification ‘jumps’ (i.e. calm period) against the total window length of the historical trace. The idea behind the definition is to assign higher importance to events that occur after a prolonged calm period, rather than events that are somehow occurring more often (and thus probably caught already by the Observer module). Definition Eq. (7) can be substituted by any other suitable definition that captures effectively the requirements of the application domain. All of the (sensory) attention indices due to sensors are then forwarded to the Attention Controller module. There, the largest one indicates the sensor with the highest priority that needs attending to.

At the end one should have, in a given moment, an overall sensory attention index created by the usage of Eqs (5) or (6) and possibly an attention index due to a change in the classification history Eq. (7). As a last step the Monitor module calculates the (final) sensory attention index (in the scope of the modality), which is simply the maximum of the two, i.e.:

\[ S - \text{AttentionIndex} = \max(S - AI - \text{Sen}, S - AI - CL) \]

(8)

This is the value that is propagated to the rest of the family tree and the family leader. At this point the Monitor calculates also a motor attention index. This is created by failed/poor action application in the previous processing step. As it has already been explained action could fail to complete, or even assuming completion could produce poor results. Thus the agent would normally observe a deviation between the actually occurring state and the one intended to be produced by the action that it was generated from. Such deviations define the motor attention index, which has a similar definition to one used in formula Eq. (4). Similar comments apply as well. Following Eq. (4) we can define a generic motor attention index as:

\[ M - \text{AttentionIndex} = 1 - \Pr(E > \delta_1) \]

(9)

where \( \delta_1 \) is an appropriate threshold. Here we assume implicitly that only one action is generated per processing step by the modality. If this is not the case then one follows similar arguments as the ones developed for the case of the sensory attention index above. The error E in this case is defined as the deviation between the occurred action state and the predicted one through the Forward Models module (see below).

3. Attention Controller. This module induces an ordering in the space of attention indices (events). It could be considered as a conceptual stack. This module implements conceptually two distinct functions:
– It has a priority policy, for inducing an order in the space of Attention Events, and
– It has a dispatch policy for deciding which jobs will be dispatched in the next processing stage.

The above structure is a general template for building appropriate Attention Controllers. An example of a priority policy could be a competition mechanism. A very simple solution is a priority queue. The ordering in this queue is already defined by the value of the corresponding Attention Index. The motivation of definition Eq. (4) should be now clear: The highest priority event should make use of the system resources for its own processing. This principle is borrowed from the human attention system and it provides effective guidance for control purposes. The dispatch policy exists for generality and construction of tailor-made attention event processing schemes. A general template is to process the first \( Q \) events present in the Attention Controller, where \( Q \) is the number of events that can be truly executed concurrently. Compare this with the case of \( Q = 1 \) in the human brain in the case of events with high information processing load (events with lower load can share the limited human attention control capacity).

4.1.3. Action generation

This sub-system consists of three modules in the general case. These are the Rules, Forward Models and the Action modules. The Rules and Forward Models modules will be explained below. The Action module includes planning abilities (generation and/or plan libraries) that complements actions generated by the Rules module. It is intended to use systems that already exist in the literature and acts as an enhancement to the Rules module, which is the elementary action generator module. Thus Fig. 4 does not include the Action module. The focus in the current work is in the development and specification of the Rules module.

The Rules module contains models for action generation. These models could be symbol manipulation systems or sub-symbolic ones. We do not enforce a specific approach. In the current version the action generation abilities are based on a set of fuzzy rules. This set of rules determines the appropriate action taking into account the available modality information. This includes the state classification vector (i.e. the World Representation) and the contents of the Attention Controller module (i.e. all the attention events thrown). The Rules module is partitioned internally in a number of sub-modules, each one acting as a container of a specific class of rules. These sub-modules are:

– **Autonomic.** Here rules that handle the agent’s self-management and integrity are stored. It could be application independent.
– **Primitives.** This stores the primitives that the service of the goal needs. They are also available to other services. It is application dependent.
– **Monitoring.** This module stores rules about the unified policies followed by all the services for handling sensors. It could be application independent.
– **Application Logic.** This module includes rules that generate the action of the service under normal circumstances, i.e. when the service has been completed (but possibly having a poor performance).
– **Handlers.** This module stores rules for essentially recovering the service execution when exceptional conditions are encountered and the service cannot complete execution.

The second module in this sub-system is the **Forward Models** module. It includes two sub-modules:

– **The Action-State Map.** It converts suggested actions to future states (for further agent deliberation) and
– **The State Reward Model,** which updates/provides a map for rewards. This helps mainly in the learning that takes place in modes 2–3.

The Action-State Map is a model that converts suggested actions to future states. The output vector is partitioned actually in two components. One component corresponds to the future state of the agent as a result of the action taken, while the second corresponds to the state of the environment (which in the general case has a stochastic transition map). Only the first component contributes to motor attention events capturing a failure to arrive at the expected agent state after the action. The second component can be used for constructing a better model of the environment’s transition function and thus help implement the second sub-module; that of the State Reward model. The latter mainly participates in the learning modes 2–3 and thus its operation will be discussed elsewhere. The second component can be used to provide a measure of expected closeness of the environment’s state after agent action with the current environmental state. Environment here is understood as the complement to the agent concept including the physical environment, the user and other agents.

4.1.4. Remaining modules

In Fig. 4 one further module must be specified. This is the Goals Module. This module should not...
be confused with the modality’s usage as a goal in the GoalsTree. This is a module that belongs in the scope of the modality. The module is in essence a (distributed) database, which includes three main sub-modules. They are:

- **User Profile.** It includes the model parameters that are used in the State Evaluation, Attention Control and Action Generation sub-systems. It also includes decision thresholds that are used internally by the various models. As an example we mention parameter $\delta$ that appears in formula Eq. (4).
- **Services.** The module includes information on the necessary services that the goal’s service needs to acquire in order to enable its own operation, i.e. it is a description of the parent-children dependencies that are present in the GoalsTree. This is the information that is used for actually constructing the GoalsTree in run-time.
- **Target State.** This module provides a representation of the targeted state that the goal’s service must achieve. The representation used depends on the application. Typical examples include descriptions using predicate calculus descriptions, class labels for state classifications (e.g. in fuzzy based systems) or real-valued state vectors.

In the general case the various model parameters present in the User Profile can be represented as fuzzy numbers rather than crisp numbers. Currently we use for simplicity crisp numbers in our architecture.

There is an additional module present in the general level 3 architecture. This is the Critic. Its usage is mainly in the learning modes 2 and 3 and thus it will not be discussed further here. It suffices to say that the Critic collects reinforcement learning signals from the environment/user so as to penalise offending modules and thus force their adaptation.

### 4.2. Level 4 architecture: processing and control modality

In this section we present the architecture for the level 4 goals (C2 type). Its structure is presented in Fig. 5. There is a similar division in State Evaluation, Attention Control and Action Generation sub-systems. However, there are some differences in the way that the various modules work in this level. We’ll describe briefly the operation of the various modules.

It is important to clarify the meaning of the ‘fused’ vs. ‘overall’ modules that are present in Fig. 5. The former implies structures that are logically dependent and highly correlated, e.g. correlated modalities due to implicit/latent variables. The latter implies structures that are responsible for coordinating an overall emergent entity, e.g. an overall goal of achieving two separate sub-goals. In other words fused modules are in general performing a combining or fusion analysis of inputs from the modules of level 3. Overall modules are being created using the overall logic of decision making for the global modality, rather than those dedicated to each modality. They should thus have information from their lower level 3 information descendants, but at the same time have additional rules, etc arising from higher-level emergent concepts and criteria.

#### 4.2.1. Service processing

In this sub-section we describe the goal’s service operation in this level. In level 3 the service is conceptually a sequential operation of State Evaluation (where appropriate representations are built internally) and Action Generation. Attention Control operates in parallel to the State Evaluation stage and calculates any attention events present. Action Generation operates on the classified state and the existing attention events through a set of rules so as to produce appropriate actions that satisfy the goal given the state information.

In level 4 this conceptual framework is also followed but with a difference at the action generation step. As we have discussed in Section 2.2 this type of goal is tailored specifically for handling the overall action generation of a whole family tree of correlated goals. In the simplest case the parent goal must simply call the services of its children and compose the final action through the returned results of every child. In the more general case, there are dependencies present which do not allow the previous simple scheme to operate correctly. The solution to this problem is to use a 2-pass protocol for action generation. The protocol is described below and it is used for action generation of correlated children goals.

**2-Pass Action Generation Protocol**

1. **Pass-1: Fused state creation and evaluation**
   - At this stage, when the goal is about to be processed, we allow the goal to call recursively its children.
   - Eventually we arrive at the primitives’ level and we start backward chaining.
   - In the way up, new action indices are calculated. These will be used at a later stage where we evaluate the overall action generation inside a given action cell.
4. New information about the state of every child goal becomes available.
5. We use this new information in order to construct the fused state of the parent.
6. We also collect information about the proposed action per child.
7. We match the fused state and proposed actions against existing overall rules.
8. We create an appropriate flag that we pass back to each child containing information on the allowable parts of its intended action generation.

Pass-2: Fused action generation

1. For all the children the action generation process starts. The processing order follows the rules that are described by the processing category of the goal.
2. If a child does not have any action denials in its flag, then the child executes its action generation as intended.
3. Otherwise it creates only the output that is allowable.
4. The overall action is the union of the actions that is generated by the children and the parent’s contribution (if any). The parent adds its contribution at the last step.

Thus a service invocation in a level 4 goal executes the previous protocol and at the end of the second pass we have the fused action of the family. At this stage the parent may add its exclusive contribution to the generated action (Second pass, step 4). This is accomplished through the invocation of a goal’s Intrinsic Service Function. We note at this point that there are alternative ways to accomplish the above fused action generation. One possibility is using more sub-symbolic oriented way. This is an issue for future work.

4.2.2. State evaluation

As in level 3 the State Module encapsulates again the state evaluation process. It is called Fused State Module in this level. This is due to the fact that at this level typically a fused state representation is built by combining information coming from the children goals. The idea behind this is that one needs to capture appropriate correlations that carry the additional information that is needed in order to interpret a situation correctly. This is the data fusion problem. The simplest way to build fused representations is to use the cross-correlations of the state classification history series of the children. Other methods, such as ICA, etc exists. We do not provide a general method on how to build these representations. It is application dependent. We should also note that the operation of the Fused State module is about producing an overall state classification. This classification is used by the Action Generation system in order to check the consistency of the recommended actions (by each children modality) against the proper overall action. This operation appears in the first pass, step 7 of the action generation protocol. It should become clear that this type of goal does not collect sensor signals directly. Its collects, in principle, state classification information from its children to build the fused state.

4.2.3. Attention control

The discussion given in Section 2.3 covers the general operation of the attention control sub-system. There is the obvious change of names to distinguish the level. Thus the Monitor module is called now Fused Monitor whereas the corresponding name for the Attention Controller is Overall Attention Controller. As we have said, the goals in this level possibly have their own exclusive operations, captured by their Intrinsic Service function. The Attention Control sub-system captures attention events due to this latter function. What is clearly lacking is the Observer module, and thus sensory attention events due to failed prediction of sensory states. The overall state classification history acts as the sole source of sensory attention events. Compare this with the level 3 operation. Motor attention events are created due to failures in the service performance of the parent goal. See next section. It should be clear that the global attention control scheme described in 2.3 applies in the scope of this goal type in the case of G1, i.e. it regulates the competition of children inside the same family in case they need access to the same action cell having conflicting access semantics. Case G2 on the other hand is implemented in the Agent level, see Fig. 3, by the GoalsTree component, which is the global (Agent scope) attention control system. There family competition takes place for access rights to an action cell.

4.2.4. Action generation

Action generation in this level is based in the use of the action generation protocol and a separate (high-level) set of rules for detecting situations where the recommended action per child is not consistent with an overall action that is appropriate for the overall recognized situation. The set of higher-level rules that accomplishes this functionality exists in the Overall Rules
module. The structure of the module remains the same as in level 3. The Forward Models module contains again the two models of the Action-State Map and the State Reward Model as in level 3. The obvious difference here is in the scope of the models, i.e., it is the combined functionality of the family. Before closing this subsection a further clarification is needed. It was said in Section 2.1 that the action space is partitioned in sub-domains and that the winner families execute their action on them. In an implementation level this implies that the actions of every family are calculated but not yet executed, thus they are virtual actions. Only when the winner is known the virtual actions are realised. We call this process commitment. In the same sense, in the action generation protocol the proposed children actions (in the first pass, step 6 and in the whole of the second pass) are understood as virtual actions as well.

4.2.5. Remaining modules

The remaining module in this level is the Overall Goals module. It functions, as it is counterpart in level 3. It includes again the User Profile, Services and Target State components. The Services information in this level contains the description of the constituent children.

5. Simulation and results

We use the GridWorld of Fig. 6 to perform a number of simulations. The size of the grid is 23 × 16. The stationary objects are located as in the figure. A number of moving objects were used. The Route Planner’s Forward Models module has been implemented using a linear regression approach to the target cell. The speed of the robot is set to either one or zero (cell/step). The target of the simulations is to study the effectiveness of the proposed architecture in detecting a danger and then taking steps to resolve it.

In Fig. 8 we show some simulation results. The first two time series measure the distance of the robot from two other moving objects. A distance of zero indicates a collision. The third time series indicates the speed of the robot. The fourth time series indicates the activity of the Sensory Attention Index in the Collision Avoidance goal. When an attention event is posted we indicate this with an activity value of 0.5 in Fig. 8. The simulations were performed for 1000 steps and used a range of two cells as DANGEROUS threshold. When the threshold value was crossed an attention event was created and dispatched (through the Ob-

![Graph](image)
but the Route Planner had to find a new direction for the robot to move.

A number of simulations were performed with varying size of the GridWorld and the number of stationary and moving objects. The results observed were consistent with the ones presented in Fig. 8. In most of the cases the collision avoidance policy was successful. However in some cases the Forward Models module failed to predict accurately as the moving objects followed another trajectory than the one that was originally predicted. Some of these cases were handled successfully by the Observer models present in the Collision Avoidance goal. In a small number of cases both types of models failed simultaneously. This fact has to be expected in realistic settings. The solution to this is in constructing more accurate models through learning and using more complex rules for taking actions. Overall, the proposed architecture offers a viable framework for building agent applications. Clearly its performance in real application will depend on the quality of the models and rules used. The levels of accuracy needed for any real application will evidently depend on the nature of the consequences of any failure. In any case, confidence levels associated with the predictions would help offset any too glaring mistakes being made.

We have identified two main generators of attention events. These are the expectation failures in sensory attention, and the failures in action generation. Both are used for boosting the action index of a goal’s parent. Both lead to a service failure. Under some conditions attention events indicate the models’ prediction inefficiencies. If the attention events are due to poor prediction models, then a goal’s service enters in one of the learning modes. On the other hand, if the attention events indicate a genuine unexpected state in the environment then entering to learning modes is not necessary in all cases. Work is currently under progress to specify the conditions that would lead to one or the other case. The main idea is to compare the predictions of various models present in the architecture. If some of the models are consistent with the observed environmental state, whereas others are in error, this is a strong indication that some of the models are inaccurate and they need further adaptation. We expect these and other issues to be refined in the near future.

The proposed architecture offers a number of advantages in comparison with other established architectures in the agent and robotic domains. The usage of goals as the main conceptual building block provides flexibility in a number of areas:

– The Attentional Agent is able to execute a number of goals concurrently, handling at the same time conflicting goals. This allows fast, independent response per modality and handling of the scaling problem of the input space.

– The agent can be, in the same time, in automated processing mode for some goals while it could be in learning modes for others. This seamless integration of learning and processing aspects without the need for an external observer to determine the necessity and the time scheduling for learning is of great benefit. Attention operates as the generator of learning in the agent. This provides greater flexibility than serialisation of the resources and queueing of learning operations.

– We have introduced specific goal types with integrated attention control, state evaluation and action generation. There is also a second goal category, with minimal attention support and integration, which allows the implementation of composite goals of uncorrelated children (type C1) or the implementation of other architectures (type A1). In this way the Attentional Agents architecture can act as a meta-architecture. In such a case some components (i.e. goals) of a complete system can be implemented using other approaches. In this
way the architecture is open for extension while at the same time it provides an overall processing and control model that integrates with its specific types, A2 and C2, other approaches in a coherent attention-based framework.

- Following the previous point we should note that it is even possible to completely substitute the State Evaluation and Action Generation systems present in goal types A2 and C2. Currently these systems are implemented using a hybrid approach. However, one can introduce different symbolic or sub-symbolic approaches as far as he adheres to the general principles. This is a second aspect of openness (as template), which is possible because the architecture is not centered in any preferred internal representations. The elementary unit is the goal and all representations must be consistent inside this scope. Outside of the goal scope (i.e. in the family or agent level), goals with different implementations and representations can co-exist in a coherent system.

- The lack of a central ‘concept’ system is both a disadvantage and a benefit. Most robotic architectures use specific internal representations so as the various functional components to operate coherently and achieve perceptual integration. This ability can be easily introduced in the current architecture by declaring a suitable BASIC type goal, where other goals in the system can refer. The lack of a common representation inside the system is on the other hand a benefit as it allows the system to be implemented in a distributed way gaining in execution speed and robustness. It is also suitable for usage in a large number of application domains.

- We can easily add new system abilities (even in run-time) by adding new goals (i.e. services) that provide new functionalities. The run-time support provides the opportunity to use mobile code in suitable application domains. This is a crucial ability in large Ambient Environments where a complete system initialisation would not be in general possible, so as to introduce the new services. As an example of an additional ability we mention a domain expert goal that handles Natural Language Processing. Currently the communication with the system is established by using fixed communication primitives.

- We can also reduce the number of provided services. As a result of addition and deletion of services we have agent evolution in two levels. On the one hand evolution is through learning (i.e. by better adaptation of existing modalities to the environment) and on the other hand, new skills can be introduced by addition (of existing services from external sources) or creation (of new ones through reinforcement learning). The latter case however introduces the problem of availability of suitable fused representations and overall rules in level 4. See below comment on data fusion.

- One should stress one more time the difference between level 3 and 4 goals. The former are intended to be self-contained sub-systems that solve specific, well-defined problems. The latter are a way to compose complex goals, which depend in a number of abilities that are provided by the set of the level 3 goals. They handle typically correlated domain-experts, e.g. vision and auditory sub-systems.

- Finally one of the benefits that it was only mentioned in passing is the existence of application independent rules for agent self-management and sensor access. It is of course possible to augment these rules further in specific applications but they provide a minimal working set for most application domains. Both benefits arise from the use of attention for managing the agent itself. Using attention as an integrating idea for managing both the user and the agent leads to conceptual economy.

Two more points are of importance. The first is the fact that even though the design of the architecture allows for much flexibility it leads inevitably to the problem of integration (or data fusion). In our approach we use a set of overall rules obtained through domain knowledge (when it exists) or through learning. However, the development of the overall rules is a difficult and time-consuming process in many application domains. It is also possible that a complete set cannot be reached. Current results in learning aspects show that learning of overall rules is a promising general approach but it is still time consuming with uncertain results about the optimality of the obtained rules. Note here that primitives are assumed as given. To handle the general problem of openness one should develop schemes that create fused state representations and overall rules in an ever-changing agent configuration in a generic way. This is a well-known problem.

The second point is related to the issue of a real robotic agent implementation. Currently the architecture is implemented in the form of a software agent and we will work with partners to transfer it to the
robotics domain. We will report on the real world performance at a later stage. Real world performance will be clearly dependant on the sophistication of the primitives, the internal models used for the various tasks and the decomposition of the high-level tasks in suitable goal structures.

Finally we mention that in the current architecture we do not introduce the biasing of attention control through existing goals. This aspect will close the loop between goals and attention. It will be considered in future work.

Acknowledgments

This work has been performed in the scope of the FET DC ORESTEIA Project (IST-2000-26091) of the European Union. The project belongs to the Disappearing Computer Initiative. We kindly acknowledge their support to our work.

We would also like to thank our anonymous reviewers for their helping comments. They helped us to improve the readability and the structure of the paper.

References


