Artificial Intelligence 175 (2011) 1020–1036

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

A survey of motivation frameworks for intelligent systems

Nick Hawes

Intelligent Robotics Lab, School of Computer Science, University of Birmingham, UK

ARTICLE INFO

Keywords:
Motivation
Planning
Goal-directed behaviour
Agent architecture

ABSTRACT

The ability to achieve one's goals is a defining characteristic of intelligent behaviour. A great many existing theories, systems and research programmes address the problems associated with generating behaviour to achieve a goal; much fewer address the related problems of how and why goals should be generated in an intelligent artifact, and how a subset of all possible goals are selected as the focus of behaviour. It is research into these problems of *motivation*, which this article aims to stimulate. Building from the analysis of a scenario involving a futuristic household robot, we extend an existing account of motivation in intelligent systems to provide a framework for surveying relevant literature in AI and robotics. This framework guides us to look at the problems of *encoding drives* (how the needs of the system are represented), *goal generation* (how particular instances of goals are generated from the drives with reference to the current state), and *goal selection* (how the system determines which goal instances to act on). After surveying a variety of existing approaches in these terms, we build on the results of the survey to sketch a design for a new motive management framework which goes beyond the current state of the art.

1. Introduction

It is generally acknowledged that *goal-directed behaviour* is one defining feature of intelligent autonomous behaviour [21]. This is usually taken to mean that an intelligent system should be able to perform actions to alter the current state of the world (where the world also includes the system itself) to satisfy some need. To date, researchers have succeeded in producing many systems that display this kind of goal-directed behaviour within a single task or domain, but have made little progress on more general systems able to satisfy their needs across many different domains. This is a function of the fragmented nature of AI as a field of study: sub-disciplines of AI study domains to which their technologies are directly applicable, with limited concern for their role in more general-purpose systems. One effect of this narrow focus is that surprisingly little attention has been given to the issue of where goals for an intelligent system come from; it is often assumed that they are produced in some other subsystem, external to the one being actively researched [29]. However, as we start to study designs for *integrated intelligent systems* which are able to use multiple interacting competences to solve a wider variety of problems than those addressed by a single sub-domain of AI, we can no longer ignore the problem of goal generation and a host of associated, cross-cutting, problems.[1] We ultimately require architectural solutions to many such interrelated problems if we want to make progress towards the kinds of general-purpose intelligent systems that have long been promised (but not delivered) by AI.

E-mail address: n.a.hawes@cs.bham.ac.uk.
URL: http://www.cs.bham.ac.uk/~nah.

[1] We use the term "cross-cutting" to refer to those problems which *cut across* the concerns of other subsystems in a typical intelligent system. Such issues may include problems such as goal generation and management as discussed in this article; attention and resource allocation; cross-modal inference; execution monitoring; and the generation and use of amodal representations.

0004-3702/$ – see front matter
doi:10.1016/j.artint.2011.02.002

More specifically we are interested in studying mechanisms for *goal generation* which allow a system to generate its own goals to satisfy pre-determined needs, and *goal management* which select which of these goals are subsequently allowed to influence the system's behaviour. As these mechanisms encompass different states and process which *motivate* a system to behave in a particular way, we refer to then collectively as a *motive management framework*. This article supports our interest in such a framework with an example scenario (see Section 1.1), before building on previous work to identify requirements for goal generation and management. Armed with these requirements we then survey the existing literature (Sections 4.1 to 5) before sketching a design for a motive management framework (Section 6).

1.1. A motivating scenario

We can illustrate the problem of motive management with an example scenario from an application domain which is currently a firm favourite in integrated systems projects: the home help robot. Let us imagine a futuristic world in which the fields of vision, manipulation, dialogue, navigation, etc. have yielded technology which is powerful and robust enough to be deployed on robot which is sold as "Alfred," a butler for a modern age. When Alfred is first unpacked by its new owners it is given a tour of their family home. This tour (as is becoming traditional in HRI, e.g. Peltason et al. [40], Zender et al. [53]) is an opportunity for Alfred's owners to indicate and label important rooms and features of the house ("this is the kitchen," "through there is the bathroom"); point out objects ("the dishwasher is under the sink," "that's my priceless Ming vase"); and perhaps provide Alfred with some knowledge about the day-to-day running of the home ("we like to have breakfast at 7 am," "try not to leave tyre marks on the wooden floor"). Given what we know about typical human behaviour, state-of-the-art AI techniques and the tasks such a robot might be required to do in the home, we can add further detail to this scenario. First, we can assume the tour given to Alfred only covered a subset of all the information about the home that it will need in order to carry out tasks. For example, different members of the family may use different names for rooms and objects; Alfred may not have been taken into or shown all the rooms in the house; and the tour will certainly have not included indications of the locations of all the objects (both static and dynamic) that Alfred might have to know about for all future tasks. Second, given the difficulty of operating in natural, previously unseen, environments, we have to assume that Alfred cannot be completely certain about all the information provided to it during the tour. Perhaps it didn't manage to reliably segment all the things indicated to it from the background, or successfully segment the large-scale space of the building into chunks against which it can resolve the room names provided during the tour. Finally, we will assume that Alfred should not just wait for instructions to perform a particular chore: it should seek out opportunities to assist the family whenever it can. For example, it might clear dirty mugs from a side-table, pick up toys from the floor, or empty the washing basket into the washing machine when the basket is full, all without being asked.

With this scenario in place, we can then consider what Alfred should do when the initial tour concludes and it is left to its own devices (literally). It will have a number of gaps in its knowledge about its environment, and it should assume that the humans which will interact with Alfred will take all these limitations for granted when giving it tasks. It will also have a list of tasks it could perform immediately (such as clearing up the packaging it arrived in) and tasks that it might need to perform as acts of routine (self) maintenance (such as connecting to its manufacturer's website to check for software updates, or recharging its batteries). Alfred must also be aware that a human might give it a task at any time (such as fetching something or passing on a message). So, how should Alfred proceed? First it must be able to construct a list of the things it thinks it should do, and from this reason (in whatever way) both about which of these things it can do, and which things it might prefer to do. In this process Alfred must take into account at least its limited hardware resources (it can only be in one place at any time and has only one set of sensors and effectors). In addition to basic physical constraints, Alfred would ideally be able to consider the utility (in a general, not-necessarily-numeric, sense) of doing particular things over others, and also consider attempting to do some subset of all the possible things at once, rather than each thing in sequence. There are many different rationales that Alfred could use when making a selection. For example, after the tour Alfred could first choose to visit every room in the house in order to build up a map of the building (the rationale being that such knowledge is required to support many other things it may be required to do). Alternatively Alfred could assume that the rooms it has been shown are all the important areas of the house, and instead spend its immediate future locating objects which it might commonly need to know about to perform future tasks (vacuum cleaner, PC, TV, oven, fridge, etc.). Or it could interleave these two types of behaviour. Or Alfred could instead attempt to meet other members of the household (to fill the gaps in its local vocabulary). Or perhaps, following a different approach entirely, Alfred should position itself in a prominent position in the house in order to solicit chores. Or immediately start doing the chores it determines need doing. As you can see from this brief and incomplete treatment of a simple scenario, the possibilities for future behaviours are numerous, and a variety of information is needed to help choose between them.

Given that Alfred is not stymied by choice and starts following a selected course of action, we are faced with another set of related problems. What should Alfred do if it encounters a member of the household who then provides it with something else to do? Should Alfred stop what its doing, or continue until it's free? Also, what if Alfred's behaviour uncovers new possibilities for action (e.g. if a new room is discovered), should it now take this into account immediately or later? Also, what if new information discovered through action changes the information used during the previously discussed selection process or renders some of the proposed future behaviours pointless (or at least less desirable)?

These problems start to outline the requirements for a motivation framework capable of managing the goals of an intelligent system. Although we have illustrated the need for such a system with a far-fetched scenario, current integrated

systems, including the ones we are actively developing, are starting to present similar requirements. To address this need we are ultimately interested in developing a motivation system for a future general-purpose architecture suitable for intelligent robots operating in the real world. To move towards this point we must first make the requirements and possible designs for such a system explicit and explore existing solutions to previous problems. This is the purpose of this article.

2. Background

We wish to provide an architectural account of mechanisms that can cause a system to act to address one or more of its needs. A need is defined in Sloman et al. [48] as the relationship between a system and that which is necessary to bring about or maintain a possible state of affairs. This work also defines *desire-like states* which are states whose function is to get a system to maintain or achieve a possible state-of-affairs, and *belief-like states* which provide information to allow the system to achieve its desire-like states. Goals are an example of a desire-like state. Information derived from sensors, and the information derived from this and other (stored) information, provides belief-like states.

Our approach to architectural work in informed by Sloman's CogAff Schema [45]. The schema is a method for characterising (i.e. describing, not prescribing) the different forms of processing which can occur within an intelligent system. It distinguishes three types of processes: reactive, deliberative and meta-management. In brief, reactive processes can be considered as collections of responsive condition-action rules which maintain no explicit representations of alternative choices of action. Deliberative processes can be considered as processes which explicitly represent and choose between alternative processing and action opportunities. Finally, meta-management processes are those processes which manage the processing of other management processes within the system (processes of all other types, including meta-management processes). A system can be composed of processes from just one of these three types, all of them, or some selection.

By considering system composition using the CogAff schema, we can start to investigate what it means for differently-composed systems to demonstrate goal-directed behaviour. To do this we will first informally define this type of goal-directed behaviour as a sequence of actions which alter the world to bring about a state which satisfies one or more needs of the system performing the actions. This broad definition will cover not only the examples given in our far-fetched scenario, but also much simpler artifacts performing the tasks they been designed to do (e.g. computer game characters playing their roles, robots following humans, chatbots answering questions). It is also applicable to biological systems from humans to bacteria. Given that this behaviour exists, we want to (invoking Dennett's design stance [17,44]) consider what possible designs can give rise to it. Or, more precisely, what possible design can give rise to just a small part of this behaviour: the systems which produce, select and otherwise manage the purposes to which the behaviours are put.

Given the CogAff schema's definition of reactive processes (as not making use of explicit representations of alternative courses of action), we can consider that they mostly make use of implicit knowledge to generate goal-directed behaviour. That is, the prior knowledge of the effects of the goal-directed behaviour generated by reactive systems must be specified at design-time. In the case of artifacts this will be done by a human designer. In the case of natural systems this job is done by evolution. Given our knowledge of behaviour-based systems (e.g. Brooks [11], Agre and Chapman [1]) we know that a collection of behaviour-producing modules, where each module is intended to serve some pre-defined purpose, can give rise to goal-directed behaviour. Goal management in reactive systems is often performed by allowing particular behaviours to control the system (e.g. via winner-takes-all selection or action fusion [2]). Whilst these mechanisms prove useful for situations where a close coupling between sensors and effectors can solve all problems a system is presented with, we have seen that they are not adequate in more complex situations, and not easily extended or maintained over long periods. In particular, where the trade-offs between two courses of action are not immediately apparent (e.g. between Alfred exploring the house further or cleaning the floor of the room it's in), a ballistic selection mechanism is just not capable of the (possibly counter-factual) reasoning necessary to come to an informed decision. Taking a step back, it is clear that the systems we are interested in designing in both the short- and long-term must make use of many deliberative mechanisms to determine how they will achieve the things they are designed to. Purely reactive mechanisms – mechanisms which, by definition, cannot determine the results of a deliberative process – will be inadequate to successfully judge choices between different deliberatively-determined behaviours. However, we must not completely ignore the benefits of a reactive approach to managing goal-directed behaviour; reactive processes are quick to change behaviours in the face of new stimuli, and deal well with the asynchrony and parallelism inherent in the real world.

Although it is possible to consider a system composed purely of deliberative processes, there are very few examples of such systems produced as autonomous intelligent agents in their own right.[2] Agents require mechanisms for sensing and acting on their world, and these mechanisms are usually best designed as reactive processes (to reduce latencies). A purely deliberative agent would generate goal-directed behaviour by reasoning both about which behaviour to engage in, but also about which stimuli and world states should give rise to which behaviours. It is this latter process which restricts the usefulness of purely deliberative agents; the combinatorics of the problem (explicitly comparing all combinations of

[2] Of course there are a great many examples of purely deliberative systems in AI. But they are almost always used as stand-alone problem solvers, rather than *autonomous* systems capable of pursuing their own goals.

stimuli to all possible goals and behaviours) in any non-trivial domain will result in undesirable latencies when applying deliberative reasoning.[3]

In practise, most autonomous, intelligent systems with deliberative processes are designed as reactive–deliberative hybrid systems (which we will refer to as "hybrid systems" or "hybrid architectures" for the remainder of this article[4]). Such hybrid systems typically couple the low-latency operation and pre-defined search spaces of reactive systems with the flexible, exploratory reasoning of deliberative, or proto-deliberative [46], systems. This often results in the use of reactive processes to suggest various possibilities for action, deliberative processes to work out how a particular action should be performed, and then reactive processes to execute the action. There are many possible variations on this scheme, but, as we shall see, this has become a standard approach for building autonomous systems. Therefore it is within the space of reactive–deliberative hybrid systems that we shall sketch our requirements and framework for a motivation system. We will occasionally return to the distinctions made in this section when we review the existing work on motivation systems.

3. Requirements for a motivation management framework

Previous work by the Cognition and Affect group at the University of Birmingham (e.g. Beaudoin [3], Wright et al. [52], Wright [51]) has examined both the requirements for motivation management systems and a design to satisfy the requirements they identified (MINDER1, as described in Section 4.2.3). In particular, Chapters 3 to 6 of Luc Beaudoin's thesis [3] present a detailed investigation of motive processing. Whilst this work largely subsumes the requirements we discuss here (and thus we encourage interested readers to consult this too), we repeat a small part of the exercise here for a number of reasons. First, Beaudoin's work was performed from a theoretical perspective. Whilst this allowed him to address a wide range of issues, it meant that his work was only partially informed by experiences of implementing intelligent systems. Second, the implementation that was done assumed a largely homogeneous system using a unified representation and interacting with a simulated environment. Finally, the work was done at least fifteen years ago. We have recent experience of building a number of complex intelligent systems (robots and virtual characters) from heterogeneous, distributed components (a necessity for most embodied systems) to solve tasks which approximate some of those from our motivating scenario. This experience gives us a slightly different, if not totally unique, perspective on requirements for a motivation system.[5]

We will start by assuming that a system can engage in two types of goal-directed behaviour: reactive goal-directed behaviour and deliberative goal-directed behaviour. Reactive behaviour may just happen without the system making an explicit choice to trigger it (e.g. obstacle avoidance), although it may be able to make choices to suppress or enhance such behaviour (but we will ignore that for now). In contrast to this, deliberative goal-directed behaviour requires that the system takes a number of explicit steps: a goal representation must be created, this goal must be selected as one of the goals the system intends to pursue, then the system must generate the necessary behaviour to achieve the goal. This first pass allows us to identify a number of initial requirements which we can explore in more detail.

1. A system requires explicit goal representations which it can reason about.
2. A system requires at least one process capable of creating goals.
3. A system requires a process capable of collecting goals and then selecting which ones should be acted upon.
4. A system requires a process which can generate goal directed behaviour from a collection goal and the available resources.

To reduce the space of possible mechanisms under discussion, we will now make a further assumption: the required process (item 4) that allows a system to take goals and turn them into behaviour will be some kind of planner coupled with an execution system. This planner will take a conjunction of goals and produce a series of actions (i.e. a plan) which the system can execute to achieve the goals. The execution system will take this plan and drive the rest of the system to perform the actions it specifies. We make this assumption because these parts of the problem are relatively well-studied, and elements of existing systems (e.g. Brenner and Nebel [10], Knight et al. [31], Bonasso et al. [5], Nilsson [39], Firby [20]) can be assumed without losing much generality in our work.

3.1. Goals

Given this assumption, we can now require that the goals used in our system should be at least expressible in a format that is usable by a planning system. This is typically a logical form describing a possible state of the world. The nature of

[3] That is not to say that such an approach is impossible, just that in many domains where interaction with the world is required, a purely deliberative design is not one which adequately meets its requirements.

[4] There are many other uses of the term "hybrid system" in AI and the engineering sciences. Our usage of the term in this article should not be confused with the usage of the term in dynamic systems or neural networks.

[5] The main differences in perspective are down to the roles of different representations within a system, the influence of parallelism and asynchrony in robotics domains, and a more informed (due to developments in the fields of planning etc.) view on the possibilities for deliberative systems. None of these things lead to major deviations from the work of Beaudoin, but they do lead to us both make additions to the work and choose to emphasise some things he gave less weight to.

this description is limited only by the power of the planning mechanisms used to generate system behaviour. For example, if the planner can handle numerical data (e.g. Eyerich et al. [19]) then (a logical representation of) "increase battery charge above 90%" is a valid goal, whereas a system incapable of handling numeric input may only be able to use "charge battery." Likewise the use of planners capable of reasoning about time, resource constraints and uncertainty would produce a system capable of being given more detailed goals than a system with using a STRIPS-like representation.

In addition to a state description, Beaudoin also identifies attributes which should be associated with a goal. These include descriptions of the goal's *importance* and *urgency*; information which will allow informed management decisions to be taken by the system. Beaudoin also identifies other attributes required in a goal structure. These will be introduced in Section 3.3.

3.2. Goal generation

Once we require our system to have explicit goals, we require processes which can generate them. Following Beaudoin, we shall refer to these processes as *goal generators*.[6] Where goals come from is an issue that is often overlooked in work on intelligent systems, so it is something we shall expand upon here (and in the following literature survey).

Our basic assumption is that a goal generator is a specialised process which can explicitly predict and describe future states which will satisfy a need of the system. These states are communicated to the rest of the system as goals. Goal generators are therefore the processes which turn a system's *drives* into goals, where a drive is a disposition to satisfy a need (i.e. a desire-like state). For example, Alfred may have a drive to maintain a minimum level of battery charge (where the need is not to have a flat battery), and a drive to perform tasks specified by its owners (where the need is to be a useful domestic assistant).

To go from a drive to a goal, a goal generator must inspect the belief-like state it has access to (e.g. information from sensors, and information derived from this) and determine both whether it is appropriate to instantiate a new goal derived from its drive, and what the form of this goal should be. The exact nature of this process in an implemented system will depend greatly on the drives, and on the representations and algorithms used in the system. Regardless of this, it is useful to make the elements of the goal generation process explicit as it helps us to avoid mixing up similarly-purposed processes.

When an intelligent system is acting in the world, any state change which occurs internally or externally (regardless of whether this was a change enacted by the system or not) may provide the information necessary for a goal to be generated. State changes might be as simple as a scheduled time being reached, or a resource falling below a particular level, or they might involve more complex chains of processing based on the results of processing belief-like representations built up both from sensing and reasoning. As changes in the world (and thus a system's internal state) can be unpredictable, such state changes may occur at times which are unpredictable by the system. As these changes may provide opportunities for a system to satisfy one or more of its needs, it is important that the system is able to respond immediately (if appropriate). This gives rise to one of the important requirements for both goal generators and the motive management framework in general: it must be able to operate concurrently with other systems (particularly action-producing ones), and allow asynchronous changes in state to produce goal-directed behaviour. Without this requirement a system with multiple drives operating in the real world may not be able to produce goal-directed behaviour in a timely fashion, and may thus miss opportunities for satisfying its needs.

To examine the requirements for goal generation further, we can briefly explore possible ways in which goal generators may operate. Perhaps the simplest goal generator we can imagine is one which encodes a desire to keep a given variable at a particular value. As the system runs, this goal generator monitors the given variable, comparing it to the desired value. When it determines that the value of the variable does not match the current state, it generates a goal which, if adopted and acted upon, will return the variable to the desired value. Although this example is simplistic, we can use it to highlight aspects of the process which are either desirable, or may cause problems during system design and implementation. First, the simple generator only monitors the information it needs to determine whether a goal should be generated. It does not monitor what the system is currently doing, what other goals may have been generated, or could potentially be generated to satisfy other drives. In this sense it is specialised to generate a goal regardless of the rest of the behaviour of the system. Second, the generator must be able to express the future state which will satisfy its drive. In the simple example this is described as a goal which will cause the system to return the variable to its desired value. However, this assumes that the deliberative systems later in the processing chain use a representation which is able to express this future state in a direct manner. This will often not be the case in hybrid architectures, or architectures composed of heterogeneous modules. It is possible to foresee two reasons for this. First, it could be possible that the representations used by the goal generator to monitor the variable are not expressible in the goal language used by the subsequent systems. Many planning systems cannot handle numerical values, and so would not be able to process a direct expression of the desired variable state provided as a goal. A second point is that, as a system may not possess actions which can directly influence particular states (notably those which are mediated by interactions with the external world), the goal generator must translate its desired state into a state which is achievable by the system, and where the achievement of this state has the effect of also achieving the state the

[6] Beaudoin [3] actually uses the term *goal generactivator* because the processes can generate new goals or activate existing, but suppressed, ones. We use the shorter term *goal generator* for conciseness.

goal generator's drive is related to. For example, if our simple goal generator was monitoring battery charge, it would create goals when the charge fell beneath a particular threshold. If the system's deliberative processes cannot reason about battery charge directly, and instead must use abstractions such as *batteryCharge*(*full*) or *batteryCharge*(*half*), the goal generator must be able to generate the appropriate abstraction (*batteryCharge*(*full*) in this case) from the state it has access to. In general, given the assumption that intelligent systems must employ a heterogeneous collection of processing modules to function effectively, we can deduce that any goal generator not operating directly on the representations of the planning or goal-management systems will need to perform some kind of translation between the precise state which will satisfy its need and the goal which will drive the system to satisfy it.

3.3. Goal management

As the Alfred example demonstrates, an intelligent system can have multiple goals. Following on from our previous deconstruction of goal generation, we can highlight two forms of multiplicity: a system can have multiple drives, where each drive is encoded in one or more goal generator; and each goal generator could produce multiple goals, representing multiple possible routes to satisfying a need. Because goal generators should not monitor other states beyond those related to their needs, it would be unsafe and inefficient to let any generated goal become the current goal of the system. Instead we require some form of *management system*. This system must accept the goals produced by the goal generators; select which goal or goals should be pursued (with reference to any ongoing goal-directed behaviour); and then trigger the appropriate behaviour generation mechanisms (e.g. planning and execution) to achieve the selected goals.

The requirements on the input functionality of the management system are quite simple. As goal generators produce goals asynchronously and in parallel, the management system must be able to accept new goals in this manner too. The management system should not block the operation of goal generators when it accepts new goals, as this would interfere with their ability to respond to new situations. Additionally, the management system must be able to respond to new goals in this manner; if new goals can be generated in such a dynamic fashion, the management system must also be able to engage its decision procedures in similarly (rather than delaying later processing stages until a fixed point is reached).

The selection of which goal or goals a system should peruse is perhaps the largest problem in this field. Before exploring requirements on the selection process and its dependencies, we will outline the basic elements of the selection process. As with previous descriptions, these descriptions can be read as summaries of the parts of [3] which are relevant to this process (e.g. Chapter 4). The management process has a set of goals to choose from. We will refer to these (following Beaudoin) as the *managed goals* (i.e. those which the management processes have accepted from the goal generators). It's role is to *select* which goals from this set should be *activated* as the goals which will dictate the future (goal-directed) behaviour of the system. In an extension to the CogAff work we assume that multiple goals can be combined into single goal for the system to pursue, as a goal for a planning system is typically a conjunction of subgoals.[7] The aim of the selection process is to ensure the system engages in behaviour which best satisfies its needs given the available resources and various contextual restrictions. This implies that the system is attempting to maximise some general, not necessarily single-valued or numeric, measure of utility. For the time being we will ignore formal, decision theoretic, views of utility maximisation (e.g. due to the lack of reliable information available to a robot for this kind of decision making [41]), and instead adopt the requirement that, given the knowledge and resources available to it, the selection process should aim to take the management decision which results in behaviour leading to as many of its needs being met as possible.

Given some currently active goal-directed behaviour, the possible actions a goal management process can take are to either interrupt the current behaviour with a new goal (or conjunction of goals, possibly including the current goal) or to allow the current behaviour to continue. This ignores many other, more complex, possibilities such as scheduling goals for later adoption. We ignore these possibilities here to allow a simpler treatment of the subject.

The decision of which goal to select is an involved and complex one. There are many metrics which could be used in this decision, and many different types of decision-making procedures. At a high-level we can consider two types of (complementary) approaches: those based on achievability and those based on cost/benefit analyses. Achievability-based decisions will determine whether the system can actually achieve the goal from the current state, rejecting goals which cannot be achieved. This process will require planning-like reasoning, the results of which could be re-used if the corresponding goal is adopted. Decisions made on the basis of a cost/benefit analysis will compare goals based how much they benefit (see previous discussion) the system at what cost. Cost will be a feature of the plan generated to achieve the goal, so achievability-based decisions should be taken before cost/benefit analyses are performed. By combining these approaches, the selection procedure should aim to select the goal which is both achievable and the most beneficial.

To aid the decision making process, Beaudoin identifies additional information which goal generators should attach to their goals. These are *importance*, *rationale* and *urgency*. Importance is intended as a (heuristic) representation of the benefit to the system of adopting the goal. It is added by the goal generator, rather than determined at a later stage, because the goal generator represents the need that the goal ultimately satisfies, and *benefit should be tied to the need, not to the individual goal*. It is possible that the same desired future state may satisfy many different drives. In this case it is arguable that such

[7] In this framework we will consider disjunctive goals as separate proposals from goal generators, making the disjunction implicit in the selection process. A completely general framework would allow explicit disjunctions to be considered by the management process.

a goal should have a higher importance than a goal that only satisfies a single need (although this is conditional upon the relative importance of the desires involved). The rationale information attached to a goal is intended to make the connection between goal state and need more explicit. This field should describe the reason why the goal was generated. In our single-variable example the rationale would be the difference between the current value of the variable and its desired value. In the Alfred scenario rationales may include formalisations of statements such as "because my owner told be to do X" and "because I saw dirty dishes on the table." The rationale is important as it supports both the maintenance of goals (i.e. determining whether they are still applicable or important), and provides a first step in reasoning from a goal state back to the need which required it. However, it remains to be seen whether there are clear practical benefits of including rationale information; it is neither a complete description of the reasoning behind the generation of the goal (much of which could be implicit in the design of the goal generator) nor a concise but imperfect heuristic summary, thus it may not be truly useful for either deliberative or reactive decision making.

The final type of information that Beaudoin attaches to a goal, urgency, describes its temporal qualities. Its purpose is to tie importance to the passage of time, and possibly changes in state, in the system's world. The simplest notion to consider is that of the deadline beyond which the goal becomes unachievable or invalid (such as when the robot's battery runs out). This would provide one type of (binary) urgency information. This could be augmented with information that the importance of the goal increases monotonically as the deadline approaches. Richer representations of urgency may take into account time windows when the goal may be less costly to achieve or provide a greater benefit to the system. Urgency could also be specified in qualitative, rather than purely qualitative ways. For example, a goal's urgency may be dependent on state changes that may occur, or other behaviours the system may engage in. The ultimate purpose of urgency information is to support the goal management system in deciding upon and scheduling the future behaviour of the system. Goals that are more urgent (i.e. goals that should be achieved sooner rather than later) should take precedence over less urgent goals of similar (and perhaps greater) importance.

3.4. Summary

Before we leave this discussion of motive management, some final points should be made clear. Whilst the discussion has highlighted the requirements for a goal generation and management framework, it can only really motivate requirements on the types of information and decision-making procedures that such a framework should include. It cannot strongly motivate designs which group and structure representations and functions in particular ways. For this we must also be constrained by the domains in which we are deploying our hypothesised system, and, perhaps more importantly, the remainder of the architecture in which the motive management framework is deployed. This brings us to a related point. When considering possible designs and implementations for motive management frameworks, we must be aware of the co-dependence, or coupling, between many different aspects of possible systems. For example, if all the needs a system can have are pre-judged by a designer to have equal importance then its goal generators will not need to annotate goals with importance information and therefore its goal selection mechanisms will not need to take importance into account. A similar argument could be made for urgency. Additionally, given a potential scenario there can be much room for debate around the granularity and content of mechanisms involved, particularly the needs, drives and associated goal generators (i.e. what becomes a "top-level" goal for a system). This in turn influences the role of the subsequent behaviour-generating and -managing mechanisms: "smaller" goals may require less planning effort per goal, but a provide a planner with less opportunity to make optimisations across all future actions (as behaviour is generated in shorter chunks).

Even with many issues remaining open, we will find that the preceding discussion allows us to investigate different extant system designs in a more coherent way than would have been otherwise possible. In particular, we can use it to review the different approaches that have been taken, or can be taken, to:

- **encoding drives**: how an approach represents the needs of the system in its design;
- **goal generation**: how an approach combines the results of sensing and processing with its drives to produce goals to act on;
- **goal selection**: what mechanisms and information are used to select between conflicting goals.

4. Previous work related to motive management

In the following sections we will explore existing literature on architectures which feature motive management mechanisms. Most of the works surveyed explicitly address either goal generation or goal selection. The works that don't were included because they address it implicitly (i.e. they solve related problems as part of a larger system) and are representative of a collection of similar approaches. Some fields may appear to have been over-looked, but this is due to a desire to tackle the chosen literature in detail without expanding the survey to an off-putting length. We have not surveyed a great deal of work on action selection (e.g. Tyrrell [49]), although there are many overlaps between work on action selection and on the various approaches to reactive planning and behaviour in the work we do cover. We have also not explicitly addressed work on rational decision making or meta-reasoning (e.g. Zilberstein [54]). Although both of these fields are relevant to motive management (particularly to the issue of goal selection), we have chosen to focus on work which has resulted in complete

intelligent systems, rather than individual algorithms. Once we have a motive management framework, this will provide a context in which we can look to apply results from these fields.

We present the literature in an order that roughly corresponds to the addition of the layers from the CogAff schema. We will start with reactive systems with minimal or no declarative representation of goals (i.e. behaviour-based systems, Section 4.1), proceed to proto-deliberative systems (i.e. behaviour-based systems with additional structure, Section 4.2), and conclude with hybrid systems (i.e. combinations of reactive and deliberative planning systems, Section 4.3). The categorisation of mechanisms this narrative structure provides is not particularly important (as the division between categories is often difficult to precisely position), but it allows us to discuss similar systems in subsequent (sub-)sections.

4.1. Reactive

Purely reactive systems typically encode their functionality as modules which compete in various ways to generate system behaviour. Such systems do not have explicit goal instances as described above, but instead implicitly encode drives into modules which perform the behaviour necessary satisfy their need: goal generation, selection and execution are merged into a single, ballistic, activation process. Such reactive systems are usually described as *behaviour-based*, as the behaviour which is generated is the key feature of such a system (rather than, for example, internal representations). Examples include the Subsumption Architecture [11], the Agent Network Architecture [34] and Pengi [1]. If a reactive system is capable of merging the outputs of multiple active modules the we can describe the system as having multiple active goals. However, the lack of any explicit goal representation means that any such system is incapable of reasoning about the selection of one goal (or behaviour) rather than another. This is one of the reasons that behaviour-based systems have failed to scale to tasks beyond those requiring close coupling with the environment.

Bryson [12,13] tackled the problem of a reactive system having multiple (possible conflicting) goals by embedding a reactive planning system into a structure called a *drive collection* in a Behaviour-Oriented Design (BOD) architecture. Each entry in the drive collection is a distinct reactive plan for satisfying a particular drive. Rather than allowing all drives to compete to control the system they are evaluated in a fixed priority order (i.e. the most important drive is checked first), with the first applicable plan assuming control of the system. This approach allows a designer more explicit control over the way the system chooses which behaviour to pursue. The ordering of reactive plans in the drive collection represents a fixed importance evaluation across all of the needs a system has.

4.2. Proto-deliberative

The limited applicability of purely reactive systems to a wide range of problems caused researchers to extend them in various ways. Typically these extensions add an additional layer to the system containing explicit representations of goals and the current state (i.e. desire-like and belief-like states), allowing behaviour to be directly managed at runtime [22,13]. In this section we will survey a selection of systems that demonstrate this approach.

4.2.1. Belief-desire-intention systems

The issues of explicit goal representations and the ability to choose between goals are commonly associated with study of Belief-Desire-Intention (BDI) systems. Such is the prominence of this work that the term BDI has become short-hand for a general way of dividing up types of representations. In contrast to its general use, the term BDI also describes a rather specific class of system designs. We will treat these two uses separately.

The common usage of BDI assigns the kinds of things a system can represent into three classes: beliefs represent the information the system stores about itself and its world; desires represent things the system would like to do; and intentions represent desires which the system has committed to achieve. To position this usage relative to the terms we have already introduced (Sections 2 and 3) beliefs refer to the results of sensing and processing which are stored in the system somehow, i.e. belief-like states; the term desire encompasses both the drives which are encoded into goal generators and, depending on your view, goals which have been generated but not activated; and intentions are the goals which have been activated by the management mechanisms. Desires and intentions are both instances of desire-like control states. These folk-BDI definitions do not specify a design for a system architecture (except the requirement that it allows the BDI distinctions), and do not make some necessary distinctions explicit (particularly the decomposition of desires and intentions into the intermediate steps ranging from needs to behaviour).

The field which spawned this relaxed usage of the term BDI is the field of BDI agents [23]. BDI agents are logic-based agents developed in the context of resource-bounded, practical or rational, problem-solving. There is no single canonical BDI architecture, rather there is a family of similar approaches inspired by the work of Bratman [7], including the Procedural Reasoning System (PRS) and its derivatives (e.g. Georgeff and Ingrand [24], Myers [38]), and the Intelligent Resource-Bounded Machine Architecture (IRMA) [8]. So, rather than review the goal generation and management properties of the whole field, we will choose a popular exemplar, PRS-CL, and examine that.[8]

[8] A similar approach to surveying BDI systems has been taken by other authors, including Georgeff et al. [23] and Jones and Wray [30].

PRS-CL (the fully-featured "classic" (CL) version of the Procedural Reasoning System) is a logic-based reasoning system which combines goal-directed reasoning and reactive processing in a single framework. It uses a unified logical language to represent beliefs, actions and goals. Its fundamental method of problem-solving is reactive planning. Its basic plan formalism is a Knowledge Area (KA), a fixed sequence of actions aims to achieve a particular goal. KAs can invoke other KAs to achieve subgoals, thus producing a plans through an hierarchical refinement approach. KAs are comparable to the Reactive Action Packages of Firby [20] and the Teleo-reactive Programs of Nilsson [39]. Where PRS-CL differs from these systems is through the embedding of KA processing within an *intention structure* which manages which goals the system is currently pursuing. To achieve a goal the PRS-CL interpreter finds a KA that will satisfy it, then instantiates the KA with information from the goal and current state (i.e. the current database of beliefs). This instantiated KA (and subsequent refinements of it) is referred to as an *intention*. A PRS-CL system can have multiple goals, and can have multiple intentions too. In terms of our earlier requirements, a PRS-CL intention is comparable to a goal which has been activated by the management system (i.e. it is being actively pursued). PRS-CL does not have a goal management system; intentions are automatically created when a KA can satisfy a goal. In this manner PRS-CL reactively selects all goals which are achievable according to its procedural knowledge and beliefs, thus running the risk of becoming oversubscribed with things to do. In practise such situations are avoided either through appropriate KA/goal design, or though the use of meta-KAs. Meta-KAs are KAs which can operate on the intention structure itself, performing actions such as reordering the current intentions (thus determining which ones get acted upon) or suspending selected ones (allowing a distinction between active and non-active goals). Meta operations allow PRS-CL systems to implement some of the features we identified previously as desirable. For example, in a PRS-CL system for spacecraft system monitoring, a meta-KA is used to promote intentions to check the reliability of sensors ahead of all other intentions [25]. This is a system- and task-specific implementation of managing goals based on an implicit measure of importance.

In PRS-CL systems, drives are represented implicitly as *trigger conditions* associated with KAs. When a trigger condition is matched by a system belief the KA is automatically instantiated as an intention. This is perhaps the mechanism in PRS-CL which come closest to satisfying our previously stated requirements for goal generation (even though it generates intentions, i.e. goals plus plans, rather than just a plan). Explicit goals in PRS-CL (i.e. statements which need to be matched against KAs) are typically added by an external source (e.g. a human operator or, as in the spacecraft monitoring example, another instance of PRS-CL).

In summary, PRS-CL has many features that satisfy our requirements for a goal generation and management system. Its primary weakness (in relation to our needs) is its inability to explicitly represent goals which have not been activated: all goals are ballistically activated (turned into intentions) as soon as possible. This makes the process of goal management one of post-hoc scheduling of intentions, rather than one of selecting which goals should become intentions (although allowing intentions to be suspended after being first activated does allow some selection to take place). This limitation is understandable when one considers that practical (i.e. real-time, responsive) reasoning is one of the aims of PRS and that this is most easily achieved using reactive (ballistic) not deliberative (considering the consequences of alternatives *before* action) methods.

4.2.2. Soar

Soar is a computational model of human problem solving which has latterly developed into more general framework for developing knowledge-intensive agents [32]. It shares many features with PRS-CL, but is less formally constrained than other, logic-heavy, BDI systems.[9] One of the major similarities between the two approaches is their general approach to generating behaviour: the use of reactive, hierarchical refinement of pre-specified plan-like structures (KAs in PRS roughly map to *operators* in Soar). One of the major differences is the ability for Soar to overcome *impasses* (situations where no operator is directly applicable) by employing additional reasoning mechanisms and learning from the results (*chunking*).

In terms of goal generation and management, Soar has less built-in functionality than PRS-CL. Internal goal generation is only supported to overcome impasses, a fact which has resulted in some system designers overloading this mechanism to generate task goals [30], while others find other "idioms" to use to add the necessary functionality in implementation-specific ways [33]. In general we can consider that these engineering solutions draw on the same (reactive) processing model of Soar, and thus can be directly compared to trigger conditions in PRS (i.e. goals are derived directly from matching rule conditions against a database of beliefs).

Unlike PRS-CL (which allows suspended intentions), Soar does not support non-active goals. As such it provides no standard mechanisms to select between different goals. If required, such mechanisms must be added on a system-specific basis using the existing functionality (i.e. via Soar operators). The lack of more general goal generation and management functionality in Soar is seen as a weakness by some of the researchers using Soar.[10] To overcome this, these researchers have started developing a BDI-inspired abstraction layer for Soar [33]. This layer includes declarative goals (rather than goals implicit in generation rules) and *activation pools* for desired, active and terminated goals and their associated plans (where a goal plus a plan is comparable to a PRS intention). These pools give a Soar system the ability to have proposed but

[9] For a more general comparison between Soar, BDI systems and other agent frameworks we direct the reader to the excellent comparison piece by Jones and Wray [30].

[10] This view implicitly supports the research direction promoted by this article.

non-active goals (the desired pool), which is a pre-requisite of goal selection and management. The BDI layer uses Soar's built-in belief maintenance mechanisms to manage which goals should be active. This appears to be the only (documented) mechanism for determining which activation pool a goal should be in. No information is provided on, for example, methods to select which active goal should be preferred by the system.

4.2.3. MINDER1

The analysis and design work of Beaudoin (which yielded the NML1 architecture design [3, Chapter 5]) was subsequently explored and implemented by Wright [51] as the MINDER1 system. As with the previously described systems it takes a hierarchical reactive planning approach (using Teleo-reactive Programs [39]), but augments this with a *motive management* system which controls which goal the system is currently pursuing. MINDER1 operates in a 2D simulated world called the nursemaid domain. In this domain there a number of independent agents, called minibots, which the minder has to care for in a nursery. Minibots can run out of batteries or fall into ditches. The minder should prevent these situations from happening or rectify them if they do occur. As the minder only has a limited sensing range it must also patrol the nursery to detect the state of the minibots under its care. As the minibots are all operating in parallel in real time the minder's goals change asynchronously. This is an ideal environment in which to goal management approaches.[11]

As MINDER1 is a direct descendant of Beaudoin's ideas, reviewing it allows us to review the performance of a design that was produced to satisfy the same requirements we have. MINDER1 has distinct goal generators for each system goal. Much like the previously reviewed systems it generate goals by matching conditions (which implicitly encode the system's drives) against a database of beliefs. As is required by Beaudoin's theory, goals are accompanied by an *insistence* value, which attempts to heuristically capture both the goal's importance (i.e. how crucial satisfying the drive that generates it is) and urgency (i.e. the pressure to satisfy to goal sooner rather than later). In MINDER1 these values are calculated as a function of the current violation of the underlying need. For example, the insistence of a goal to prevent a particular minibot from falling into a ditch is a function of the distance of the minibot from the ditch.

The insistence is used by components within MINDER1's motive management subsystem [51, p. 72]. This subsystem maintains two disjunct sets of goals: *surfaced* and *unsurfaced* goals. The former set contains goals which are being actively managed (but not necessarily acted upon), the latter, goals which are currently entirely ignored by the management subsystem (and thus incapable of generating goal-directed system behaviour[12]). MINDER1 separates these sets using a threshold-based *attention filter*. If a goal's insistence value is greater than the value associated with the filter then it *surfaces* into the managed set, otherwise is remains unsurfaced (and unmanaged). Surfaced goals go through a series of state transitions before ultimately generating behaviour: first they are *scheduled* (which determines whether they are processed or suspended), if not suspended they are *expanded*. Expansion is a process of deciding whether the goal should be pursued (based on resource usage, current state etc.), planning the behaviour to achieve the goal (retrieving a reactive plan in this case), then executing the behaviour. MINDER1 can only expand a single goal at once (although Beaudoin's theory postulates the need for multiple concurrent active goals). Much like an intention structure in PRS-CL, a goal is expanded in place, so that it contains all of the information (plans etc.) necessary for resumption after suspension. In MINDER1 a goal can be suspended at any time by the management subsystem. Suspension means that a goal remains surfaced, but cannot influence behaviour (i.e. it is not the single goal currently undergoing expansion).

It is worth noting that although MINDER1's sets of managed goals are similar to those maintained by PRS-CL and the Soar BDI abstraction layer (both support a distinction between active and suspended intentions), MINDER1 provides both a finer-grained distinction between goal management states and additional support for processes which change the management states of goals. The surfaced/unsurfaced distinction allows MINDER1 to represent goals which have been generated by the system, but which are not yet insistent enough to engage the management processes. The other systems we have seen ballistically select all generated goals for processing. Additionally MINDER1 provides processes for deciding and scheduling which goals from the surfaced set should be actually acted upon. The other systems we have discussed assume that any generated goal should be directly acted upon when possible (although this notion of possibility – usually encoded as conditions for the firing of a behaviour – implicitly captures part of the deciding process).

The work of Wright and Beaudoin assumes that the management subsystem of an agent is resource limited. As such it is necessary to protect it from situations where it may reach the limit of its resources. Without being too specific about the computational resources available to a particular instance of a MINDER1 system, we can accept this assumption by considering that we require the management subsystem to stay responsive to new inputs and able to make timely decisions (rather than unresponsively processing for longer than any particular goal may be valid for). In MINDER1 it is the role of the attention filter to prevent the management systems from becoming overburdened. Wright models the resource constraints of the management layer by adopting a (soft) design constraint that only three goals should be in the surfaced set at any time. This is an arbitrarily selected value, but it serves to prove a point. Within MINDER1 it is the role of the *meta-management processes* to enforce this constraint via the attention filter. In brief, these processes perform the following operations: if the

[11] This is perhaps true for systems of a mostly reactive nature, as the world almost always require a fast response. A comparable domain that has a slightly less frenetic pace is the artificial life domain of Tyrrell (as summarised in Bryson [12]) in which a rodent must survive dangers, and make the most of the opportunities, of life on a simulated savannah.

[12] Although unable to generate goal-directed behaviour in the sense we are generally interested in, unsurfaced goals can still produce management behaviour which is ultimately serving the needs of the system.

surfaced set contains less than three goals, the value of the threshold is reduced, thus allowing lower insistence goals to surface; if the surfaced set contains more than three goals, the value or the threshold is increased until lowest insistence goals "dive" out of the surfaced set. In conjunction with the ability of goal generators to increase and decrease insistence values (e.g., as minibot gets closer to a ditch), this gives rise to system with a dynamic set of active goals. The contents of this set should ideally reflect both the computational resources of the system and the current state of the world.[13]

Although MINDER1 provides a powerful starting point for a goal management system, it also raises a number of questions (some of which are raised in the original work). The calculation of insistence for goals in the nursemaid domain is trivial given the nature of the drives involved. Generating importance and urgency information for less quantifiable drives will pose new problems. The relationship between the attention filter, management processes and meta-management processes will also require further thought in systems with richer deliberative processes and wider scope. Ideally management resource constraints should be characterised by something more informative than a number limit on set size. Future work on this must span two issues: the variety of goals, and the frequency of their generation, in a particular system and domain combination; and the nature of the deliberative and management processes available to a particular system (e.g. the constraints on a system using a deliberative planner may be quite different to those only using stored reactive plans). This latter consideration will also influence the expansion process. In MINDER1 deciding, scheduling and planning for a goal are all separate steps in the management subsystem. A system which integrated all of these processes (perhaps using techniques from deliberative, rather than reactive, planning) would be able to take decisions which are better informed about the interactions both between the behaviours necessary to achieve different goals, and between these behaviours and the available resources. This view is expanded in Section 6.

4.2.4. GRUE

Another architecture which augments a teleo-reactive planning system with a goal management system is the Goal and Resource Using architecturE (GRUE) of Gordon and Logan [26,27]. GRUE is similar to MINDER1 in that it has reactive goal generators, annotates its goals with priorities, and uses a threshold-based filter to discard low-priority goals. Whilst GRUE appears to be a simplified version of MINDER1 in some ways (e.g. it has a much smaller range of possible goal management states), it goes beyond MINDER1's contributions in others: it allows multiple goals to be acted on at once (whereas MINDER1 only allows a single goal to be acted upon), and provides a rich language for managing resource constraints (constraints which are largely ignored in the MINDER1 implementation). These two contributions are combined in the GRUE *arbitrator* process. GRUE attempts to achieve each active goal with a separate reactive plan, each of which suggests an action to perform in each processing cycle. The arbitrator collects these actions and decides which ones should be allowed to execute (thus ultimately deciding which goals are acted upon). The decision is made by inspecting the resource requirements of the suggested actions. If there are any conflicts (e.g. two actions which use the same effector) then the actions associated with the lower priority goals are dropped. Whilst this demonstrates the power of combining importance measures (priority) with scheduling (resources assignment), it does also highlight the weakness of reactive approaches to goal-directed behaviour. In the event of repeated conflicts over resources, a GRUE agent has no mechanism for choosing alternative goals or plans, or reasoning in more detail about the causality of combinations of actions. If GRUE was extended with a meta-management process which configured the goal filter to ignores contentious goals it might work around such problems. A longer-term solution (in this instance and for the more general problem of multiple, interacting goals) is to move away from reactive planning approaches and investigate deliberative methods.

4.3. Deliberative and hybrid systems

In this section we review goal generation and management approaches taken by intelligent systems with deliberative capabilities.

4.3.1. Spartacus, MBA and EMIB

An architecture featuring explicit motivations was used to control Spartacus, a mobile robot which attended the AAAI '05 conference [37]. The architecture was the Motivated Behaviour Architecture (MBA) which is an instantiation and generalisation of EMIB architecture (which roughly stands for Emotion and Motivation for Intentional selection and configuration of Behaviour-producing modules) [36]. MBA features reactive behaviour-producing modules (BPMS, comparable to behaviours in the reactive systems reviewed previously) which are structured into reactive plans called *tasks* [4]. MBA features *motivations* which are processes which can propose tasks into the *dynamic task workspace* (DTW). Motivations are thus the MBA equivalent of goal generators. The DTW is directly comparable to the intention structure in PRS-CL (see Section 4.2.1) in that it automatically accepts and stores all active tasks, where tasks can be considered a goal-plus-reactive-plan structure. Motivations exist in MBA for domain- and system-specific drives (e.g. sticking to the agenda of the conference, navigating to particular points in space) and more "instinctual" drives such as curiosity and survival.

MBA allows multiple tasks to be active at once. Goal management is performed by deciding which tasks are allowed to active which behaviours (similar to the approach to action arbitration taken in GRUE). This decision is based both on the

[13] The dynamics of threshold changes and the management processes in MINDER1 can give rise to emergent perturbances. These can be used to provided an architecture-based, characterisation of a number of emotional states [52,51,48].

activation values of motivations (which can be considered as an importance measures related to how well each motivation can satisfy its own need) and the system know-how (SNOW) built into MBA. From the literature SNOW appears to be a collection of domain- and system-specific rules for deciding which tasks should be allowed to run in the event of a conflict. Whilst this is not a solution we can carry forward into a new goal management system (as it is not generally applicable), it does demonstrate that the goal-management problem may require engineering solutions for particular cases.

The reason that MBA can be considered a reactive–deliberative hybrid approach is that it also uses a planner as motivation source. Given a goal, the planner creates a list of tasks which must be sequentially proposed to the DTW in order for the goal to be achieved. This highlights that goals will exist at multiple levels of abstraction in an intelligent system; what appears to be a goal at one level (i.e. a task in this case), may be a subgoal from a more abstract level.

4.3.2. DIARC

The notions of importance and urgency (as first identified by Beaudoin [3]) are employed in the DIARC ("distributed integrated affect cognition and reflection") architecture [41]. DIARC uses explicit goals which are generated from natural language input [18] and stored in an *affective goal manager* (AGM) component.[14] Each goal is assigned an *affective task manager* (ATM) which performs action selection for its goal (which equates to reactive planning in this case). If two ATMs produce conflicting actions (where a conflict is identified as a violation in resource usage), then the goal with the highest priority is allowed to continue. Whilst this approach bares a strong resemblance to previous approaches (including GRUE and MBA), it is distinguished by a clear proposition for calculating priority values for goals. The AGM calculates a priority value from a goal's importance and urgency values, and assigns the priority to the respective ATM. In DIARC, importance is calculated by subtracting the estimated cost of achieving the goal from the product of the estimated benefits and an *affective evaluation* of the goal (where all quantities are real numbers). The affective evaluation attempts to capture the influence of the current state and past successes and failures on the ability of the system to achieve the estimated benefit. Positive experiences yield a more favourable evaluation for a particular goal, increasing its overall importance value. Negative experiences have the opposite effect. The goal's priority is calculated by scaling its importance value by urgency. Urgency is a measure of the time remaining to achieve the goal and is derived from the ratio between the elapsed time since the goal was created and the time the system expects to take to achieve the goal.

This method of prioritising goals is intended to overcome the problems inherent in taking a rational approach to decision making (which requires information that a robot typically doesn't have access to). The combination of designer specified, or possibly learnt, values for costs and benefits, combined with the contextual and historical information captured in the affect evaluation allows the system to make reasonably good choices (i.e. satisfying [54]) whilst being responsive to changes in the current situation (see Scheutz and Schermerhorn [41] for examples of the emergent properties of this combination of values). This said, this method of goal selection can only make relatively uniformed management decisions; the purely numerical measures of cost and benefit are devoid of causal structure which could allow a deliberative decision-making process (not necessarily a optimal/rational one) to select goals based on a more informed judgement of expected future states (cf. "rationale" in Beaudoin's work, and also Sloman [47] on architecture-based motivation vs. reward-based motivation).

4.3.3. MADbot

One of the few planning approaches to treat goal generation as part of the planning problem (rather than as precursor to it) is the MADbot (Motivated And Goal Directed Robot) system of Coddington et al. [16]. The architecture is comparable to previously reviewed approaches such as MINDER1 and GRUE, but uses deliberative, rather than reactive, planning to generate behaviour. MADbot represents the world as a collection of state variables (much like the databases of beliefs used by other approaches). A drive in MADbot is effectively a constraint on a single state variable with an associated importance value.[15] For example, the drive "conserve-energy" will monitor a system's battery charge level, and attempt to keep it above a particular threshold. Where the reactive systems reviewed previously would encode this drive implicitly in the trigger conditions of a goal generator, MADbot explicitly records them as part of the system description. This has allowed them to experiment with two different approaches to goal generation: motivated goal generation, and motivations as resources [14,15].

In the motivated goal generation case, MADbot behaves much like a deliberative version of the previously reviewed systems: when a drive's threshold is violated, a goal to return its state variable to below the threshold is generated. The goal is then passed to the *goal arbitrator* which decides whether the goal should be added to the current goals the system is pursuing or not. This is the only real mention of explicit goal management in the MADbot work: they note that it is important, but leave it for future work. Newly accepted goals are added to a conjunction of previous accepted goals then passed to a planner. The plans are executed and the success or failure of actions (and their outcomes) is monitored by the system. If a plan is successfully executed then the state variables should all be returned to their below-threshold levels and the associated goals are removed from the arbitrator.

[14] Generating goals from natural language input is often a very different type of goal generation process than those considered in this article. It is an approach which will certainly become more prevalent as interactive robots gain a wider range of capabilities. Examples of systems that convert linguistic statements into deliberative action range from SHRDLU [50] to DIARC and other recent interactive robot systems [18,9].

[15] The MADbot literature actually uses the term "motive" to refer to a drive, but we will use the term "drive" to maintain consistency with our earlier discussions.

This approach was evaluated in a simulated Mars rover domain where four different drives were combined with user specified goals to control MADbot. The drives were used to maintain battery charge, disk space and localisation accuracy, whilst occasionally capturing images of the surrounding environment. The state variables associated with these drives vary as the system acts. For example, battery charge is reduced by driving to the places where images should be captured, and disk space is used up by capturing images. The initial design of the system was validated as MADbot was able to perform autonomously and maintain its drives in some situations, but some problems did arise. The most fundamental problem was that, because the planner is unaware of the system's drives, it often generated plans that violated resource constraints (e.g. plans were generated which could exhaust the system's battery charge). When constraints are violated (as the system executes one of these plans) the drives create new goals to recover (e.g. by charging the battery), but this raises an additional problem: MADbot cannot guarantee that these goals will be dealt with before the resource is exhausted. As the importance of a drive is not used when the goal is conjoined with the existing system goals, the resource constraint goal just becomes something else MADbot should do, rather than something it *must* do before it does other things that use up that resource. Coddington [14] states that this could be addressed by using the importance of the drives that a plan satisfies as a factor in a metric for evaluating candidate plans. This approach is comparable to the use of importance in MINDER1 and GRUE (although they additionally vary importance as a function of the distance of the state variable from the threshold). A different way to address this would be use MADbot's (currently under-used) goal arbitrator to suspend the current task goal and replace it with the goal to satisfy the drive. Although this would ensure that the resource constraint is not violated, it would not really solve the problem; the planner would still be creating plans for satisfying resource constraints separately to plans for its other goals.

To address these problems an alternative approach to goal generation was tried: the aforementioned motivations as resources approaches. Rather than use the drives to reactively generate new goals when thresholds are violated, these thresholds are instead modelled as resource preconditions for the actions in its planning domain for the Mars rover example. For example, moving the rover consumes (and thus requires) a certain amount of battery charge, and the level of battery charge can never be allowed to fall below a threshold. This approach to goal generation changed the behaviour of MADbot in the rover domain: the plans generated for a particular task goal took into account the resource constraints ahead of time (e.g. recharging the robot as appropriate), rather than waiting for a reactively generated correction goal. This meant that the resource constraints were never violated. This improvement came at the cost of two problems. First, the lack of reactive goal generators meant that the no autonomous behaviour was possible when an additional task goal was not specified. This meant that periodically relevant drives, such as capturing images, could not be triggered unless a plan was also being generated for another goal. The second problem was that the computational complexity of processing the resource constraints made the planning problem much, much harder than previously. Some standard resource-handling planners could not find plans, particularly when more than two drives were modelled in the domain.

The author's suggested solution to the complexity problem is to employ a hybrid approach to goal generation. Drives relating to hard resource constraints (i.e. those which should never be violated) should be modelled in the planning domain, whereas less critical drives should be encoded as reactive goal generators. This appears to be a sensible approach which will benefit from the strengths of both approaches (autonomy and simplicity; and adherence to resource limits respectively). However there is an additional problem with the motivations as resources approach which will have to be addressed in future work: not all drives can be easily encoded in this manner. One reason is that some drives cannot be captured simply as small number of numeric quantities (e.g. the drives Alfred had in Section 1.1 to fill gaps in its knowledge), although these may all turn out to be safely assignable to reactive goal generators (or at least implicitly modelled drives). Another reason is that in real-world robotics resource usage cannot be modelled so simplistically, due to many factors beyond the control of the planner. This may mean that the planner will need to employ probabilistic models of resource levels and usage, again increasing the complexity of the planning problem.

Despite these potential problems with the motivations as resources approach, the work on goal generation in MADbot highlights how a key problem in intelligent system design, the trade-off between reactive and deliberative processing, is apparent in the problems of goal generation and goal-directed behaviour. In an ideal world, some general purpose decision making mechanism could assess all of the possible combinations of goals a system could have *and* all of the plans for all of these possible combinations (including the consequences of resource usage etc.) before selecting the optimal goal and plan combination and acting on it. However there are many reasons, rooted in both the complexity of such decision making and the difficulties of modelling all behaviour this way, why this approach is just not possible in practise. The solution is to therefore carve off parts of the problem into separate specialist systems (reactive or deliberative) with less knowledge about the rest of the process. This makes the overall approach less powerful (and capable of less rational decision making), but more likely to make a decision in a reasonable amount of time (i.e. while its goals are still valid). One of the challenges in the design of a goal generation and management framework is how to reduce the overall problem into appropriately specialised subsystems without reducing the decision making power of the combined parts to such a degree that it is unable to make reasonable decisions. The work on MADbot has demonstrated that a naïve approach to combining reactive goal generation and planning (albeit an approach which does not make use of importance or scheduling information suggested by other authors) fails to make reasonable decisions in certain scenarios.

5. Summary of approaches

We can now summarise the approaches we have seen to the generation and management of goals for an intelligent system. In this discussion we will conflate the different versions of a goal plus a plan for achieving it (e.g. a goal plus deliberation, an intention structure, and various other forms of reactive plans) and just refer to "goals," except where the difference is salient. We shall also adopt the following terminology to describe the different sets of goals that a system can have: *managed goals* are those goals which have been accepted into a system's goal management process (equivalent to "surfaced" in MINDER1 described in Section 4.2.3); *active goals* are those goals which are causing goal-directed behaviour to be generated (this term is used by MINDER1 from Section 4.2.3, the Soar BDI layer from Section 4.2.2, and implicitly by other approaches); *suspended goals* are managed goals which are not active (again used by MINDER1, Soar etc.); and *unsurfaced goals* are those goals which have been generated, but not accepted into the management process yet (this term is taken directly from MINDER1, as no other system has this facility).

Where a system generated its own goals autonomously (rather than having them imposed by another system) it almost always did do reactively, and with no reference to the mechanisms used to subsequently expand the system's goals. Although this approach is generally effective where deployed (e.g. in PRS-CL from Section 4.2.1, Soar, MINDER1, GRUE from Section 4.2.4, MBA from Section 4.3.1 etc.), this is often because the scenario is amenable to decomposition into a multi-stage process (i.e. goal generation, goal expansion if required, arbitration then execution). The work on MADbot (presented in Section 4.3.3) demonstrates that not all problems can be tackled in this way. In particular problems where drives exist to maintain interacting resources may be better modelled as part of a planning process, although this increases the computational complexity of the deliberation task.

All but one of the approaches we have seen allow their systems to have multiple goals active at once (we shall refer to these as *goal conjunctions*). MINDER1, the approach that can only have a single active goal at a time, was also not intended to have this limitation. Given the ability to tackle goal conjunctions, there are two problems which must be tackled: which goals should be conjoined; and which goals can be achieved when conjoined. The former issue is one of determining priorities or preferences over all managed goals. The latter is one of resolving conflicts between the actions required to achieve the active (conjoined) goals.

The work we have surveyed typically takes one of four different approaches to selecting which goals should be conjoined. In BOD (from Section 4.1), DIARC (from Section 4.3.2) and MBA, all goals are activated without consideration and thus conjoined by default. In PRS-CL and Soar goals are also activated without consideration, but can be subsequently suspended (providing a choice of possible conjunctions). In MADbot goals pass through an arbitration step before coming active, i.e. they start suspended but can later become active (again providing a choice of possible conjunctions). MINDER1 and GRUE go one step further by using an additional filter to determine which goals can become managed before then being activated. MINDER1 motivates the use of a filter by claiming resource limits in the management layer. Although the implementation of this approach does not support this, the work on motivation as resources in MADbot demonstrates the problems of putting too great a load on deliberative systems.

Once a system has a conjunction of active goals it must act to achieve them. However, actions that help to achieve one goal may conflict with actions that help to achieve another. The existing literature mostly deals with one type of conflict, resource usage, but other types of conflict can exist too (notably the actions for one goal undoing preconditions established by actions for another). In almost all cases resource conflicts are dealt with by inspecting the goals which the actions achieve, then awarding the resource (thus the right to execute) to the action for goal with the greatest priority. The existing systems assign priorities in various ways, from the fixed priority schemes or MINDER1 and BOD, to the activation-based approach of MBA, to the more dynamic, affect-based, approach of DIARC. Regardless of implementation this scheme amounts to using a single partial ordering to decide which goal is actually active during a conflict.

An alternative to resolving conflicts by ordering is demonstrated by systems with deliberative capabilities. MADbot and MBA (in some contexts), using planning, and to some extent Soar, using its impasse mechanisms, can choose selections and orderings of actions *in advance of acting* which do not have conflicting demands (providing the important resources etc. can be modelled in their representations). This again highlights the benefits of deliberation in domains where it is feasible: reactive systems will require arbitration mechanisms as their behaviours are independently generated and thus can conflict after the fact; deliberatively systems can reason about and avoid conflicts before the fact. Regardless of the overall approach, most of the surveyed approaches rely on representations of resources and priorities. Priorities are important beyond conflict resolution as they can also be used as preferences on goals in a deliberative system (e.g. Brafman and Chernyavsky [6]), and as part of the process for determining which goals should be managed and active (as in MINDER1). This is because some needs may be inherently or contextually more important to a system than others, even if no conflict is evident.

An important issue which we only saw addressed by PRS-CL and Soar (although there is much related work in the BDI field) is goal maintenance or reconsideration. This is essentially the problem of deciding when a goal should be suspended or even dropped entirely. Both systems use variations of logical entailment to determine whether goals are still necessary. Beaudoin's work argues that goals should represent their dependencies (via beliefs and rational [3, p. 47]) to support this kind of reasoning, but this was not implemented in MINDER1.

Finally, the issue of the temporal and behavioural dependencies of goals was only discussed with reference to scheduling in the theoretical work of Beaudoin and in the implementation of urgency in DIARC. Deciding *when* (not whether) a goal should become active, or some action should be executed, and then scheduling that to occur as appropriate (e.g. in two

hours time, or after this goal is achieved, or the next time I'm in a particular room) may reduce the load on a goal management system. One, computationally undesirable, alternative to this is for all known future goals to be managed when generated and for the management process to keep checking them (i.e. polling) for activation until the correct time occurs. Urgency in DIARC and MINDER1 implicitly encodes a solution to this as the goal generators can make goals more urgent when appropriate in order to make it more likely that they will be activated.

6. A design for a motive management framework

Given the preceding summary of the literature, and the requirements we set out previously, we can start to sketch out an early design, or at least a list of desiderata, for a future architecture for managing goal-directed behaviour. Our intention is to develop something along these lines that will allow us to explore the trade-offs in design space for these sorts of mechanisms.

Although most of the work we surveyed used reactive planning approaches to generate behaviour, we will follow the work on MADbot (and many other intelligent robotic systems, e.g. Bonasso et al. [5], Hawes et al. [28]) and use a planner, coupled with reactive behaviours, to determine future actions. The advantage of this approach is that interactions between active goals and conflicts over resources can be resolved through reasoning in advance of acting, rather than through post-hoc arbitration. To work effectively in an architecture for goal management for a real-world robot, any planner we use must satisfy the following criteria. First and foremost it must be a *continual* planner, capable of interleaving planning with execution and monitoring, and replanning when the world or its goals are vary from its expectations. Continual planning has been used to overcome the dynamism and partial observability which exists in robotic domains [10] and in situations where an external source can change the goals of the system [31]. The planner ideally should be able to model resource constraints, which would allow us to model motivations as resources (where appropriate). An additional desirable feature would be the ability to behave as an oversubscription planner [35,6]. An oversubscription planner is a planner which can determine which subset of goals in a goal conjunction can actually be achieved together, and, by using cost measures (like importance and urgency), which goals should be pursued together. In essence this would place the onus of activating goals (i.e. selecting which ones should be acted on) on the planner, rather than an additional mechanisms.

Although the suggested planning approach would address many of the sub-problems we have noted in this article, there are two caveats. First, we know of no planner which currently meets all of the listed requirements. We expect developing one, or altering an existing one, to be a non-trivial exercise. Furthermore, as the MADbot work demonstrates, by placing additional responsibilities on a planner you make its task harder. Given the combinatorial nature of planning problems, it is possible that the problems we wish to tackle may be too hard to solve in a single system. An alternative approach is to develop a planning architecture for goal management, e.g. using an oversubscription planner to determine which goals should be active before passing the active conjunction on to a continual planner (resource scheduling could also occur after plans are generated).

In addition to the hypothesised planner we require goal generators (reactive or otherwise) which are independent of the planing process. These will ensure that autonomous behaviour can be generated as opportunities become available (i.e asynchronously with respect to planning), and encode drives that cannot be reasoned about in a single planning system (e.g. drives that require access to information not modelled in the planning domain). To capture the rich variety of goals an intelligent system can have we may need to extend the state descriptions of planning goals with temporal operators (as in PRS-CL [24] and NML1 [3, p. 45]), although this temporal information could be captured implicitly in the nature of the goal generators themselves (understanding the trade-offs here is one of many open problems). It may be desirable to encode homeostatic drives as resource limits in the planning models (as in MADbot), although we may also want to develop a reactive alarm system for the cases where deliberation does not satisfy safety- or mission-critical drives [43].

As required by many of the approaches surveyed above, goal generators will be required to annotate their goals with importance and urgency measures. These measures will be used in various places in the system. However, it is not clear how these values should be generated or represented. Initially they can be fixed or contextually generated from drives (cf. the distance between a minibot and a ditch in Section 4.2.3), and use either numeric values or partial orderings (e.g. the drive to recharge is more important the drive to clean the floor). Ultimately we may need additional, more powerful representations, particularly for scheduling various tasks based on urgency (which appears to be an aspect of the problem of intelligent behaviour with is overlooked in the literature).

The design should allow for active and suspended goal sets (as described in the previous section). The process of goal management (i.e. moving goals between these two set) can be (implicitly) performed by an oversubscription planner, or, if this proves too complex initially, by inspecting importance and urgency measures (although there are potential flaws in this approach). We will also adopt the idea of a pre-management filtering step from the NML1/MINDER1 and GRUE architectures. As our proposed approach of goal management through planning is potentially computationally demanding (cf. Section 4.3.3), filtering may protect the management system from become overburdened with decisions (and thus unresponsive to potentially important changes in the system's goals and state). On the other hand, if goal management does not prove to be too demanding, there may be less of a justification for an additional filtering step.

Although this design sketch gives us a starting point for future work which builds on the experience of others, it presents as least as many questions as it attempts to answer. These represent issues which we will explore, using the traditional methods of AI, via a more detailed requirements analysis for particular domains, an implementation of the system, and

subsequent evaluations (i.e. we will follow a design-based methodology [42]). Open questions include the following: what mechanisms should be used for removing goals from the unsurfaced or managed sets?; how often should goal-management decisions be taken? (a dynamically changing world could render decisions incorrect without the system acting); how should longer-term behaviour be modelled? (goal-generation and planning probably cannot account for life-long behaviour in a robot, cf. the use of a planner as motive source in Section 4.3.1); and how can deliberation and reactivity interact in way that plays to their strengths, rather than constraining one approach with the limits of the other.

7. Conclusion

In this article we presented arguments for the development of a motive management frame for an intelligent system. Building on previous studies we identified the required parts of such a framework, and particularly focused on processes which can generate goals and select goals to become the focus of goal-directed behaviour. Given this focus we reviewed the literature on architectures for intelligent systems and identified a number of trends in existing work towards addressing these problems. From this we proposed an outline design for a motive management framework. The design prefers deliberation to reactivity where possible, but freely admits the limitations of this approach. It also includes mechanisms from the literature (e.g. an attention filter) in order to explore the design trade-offs they present. In future work we will implement this architecture and evaluate it in both robotic domains and in simulated worlds (as both evaluation domains present different problems and opportunities to any proposed design).

Acknowledgements

The author would like to thank Aaron Sloman and Marc Hanheide for discussions about motivation, attention and the definition of control states, and Charles Gretton, Richard Dearden, and Michael Brenner for discussions about the relationship between motivation and planning in goal-directed behaviour. The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement No. 215181, CogX.

References

[1] P.E. Agre, D. Chapman, Pengi: An implementation of a theory of activity, in: Proceedings of The Sixth National Conference on Artificial Intelligence, 1987, pp. 268–272.
[2] R.C. Arkin, Behavior-Based Robotics, MIT Press, 1998.
[3] L.P. Beaudoin, Goal processing in autonomous agents, Ph.D. thesis, School of Computer Science, The University of Birmingham, 1994.
[4] E. Beaudry, Y. Brosseau, C. Côté, C. Raïevsky, D. Létourneau, F. Kabanza, F. Michaud, Reactive planning in a motivated behavioral architecture, in: M.M. Veloso, S. Kambhampati (Eds.), Proceedings of the Twentieth National Conference on Artificial Intelligence, AAAI Press/The MIT Press, 2005, pp. 1242–1249.
[5] R.P. Bonasso, R.J. Firby, E. Gat, D. Kortenkamp, D.P. Miller, M.G. Slack, Experiences with an architecture for intelligent reactive agents, J. Exp. Theor. Artif. Intell. 9 (2–3) (1997) 237–256.
[6] R.I. Brafman, Y. Chernyavsky, Planning with goal preferences and constraints, in: S. Biundo, K.L. Myers, K. Rajan (Eds.), Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling (ICAPS 2005), AAAI, June 2005, pp. 182–191.
[7] M.E. Bratman, Intention, Plans, and Practical Reason, Harvard University Press, Cambridge, MA, 1987.
[8] M.E. Bratman, D.J. Israel, M.E. Pollack, Plans and resource-bounded practical reasoning, Computational Intelligence 4 (1988) 349–355.
[9] M. Brenner, N. Hawes, J. Kelleher, J. Wyatt, Mediating between qualitative and quantitative representations for task-orientated human–robot interaction, in: M.M. Veloso (Ed.), IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, January 2007, pp. 2072–2077.
[10] M. Brenner, B. Nebel, Continual planning and acting in dynamic multiagent environments, Journal of Autonomous Agents and Multiagent Systems, in press.
[11] R.A. Brooks, A robust layered control system for a mobile robot, IEEE Journal of Robotics and Automation 2 (1986) 14–23.
[12] J.J. Bryson, Intelligence by design: Principles of modularity and coordination for engineering complex adaptive agents, Ph.D. thesis, MIT, Department of EECS, Cambridge, MA, June 2001.
[13] J.J. Bryson, The behavior-oriented design of modular agent intelligence, in: R. Kowalszyk, J.P. Müller, H. Tianfield, R. Unland (Eds.), Agent Technologies, Infrastructures, Tools, and Applications for e-Services, Springer, Berlin, 2003, pp. 61–76.
[14] A. Coddington, Motivations as a meta-level component for constraining goal generation, in: Proceedings of the First International Workshop on Metareasoning in Agent-Based Systems, 2007, pp. 16–30.
[15] A.M. Coddington, Integrating motivations with planning, in: Proceedings of the 6th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS07), 2007, pp. 850–852.
[16] A.M. Coddington, M. Fox, J. Gough, D. Long, I. Serina, Madbot: A motivated and goal directed robot, in: M.M. Veloso, S. Kambhampati (Eds.), Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, AAAI Press/The MIT Press, Pittsburgh, Pennsylvania, USA, 2005, pp. 1680–1681.
[17] D.C. Dennett, Brainstorms: Philosophical Essays on Mind and Psychology, MIT Press, Cambridge, MA, 1978.
[18] J. Dzifcak, M. Scheutz, C. Baral, P. Schermerhorn, What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution, in: Proceedings of the 2009 IEEE International Conference on Robotics and Automation (ICRA '09), Kobe, Japan, May 2009.
[19] P. Eyerich, R. Mattmüller, G. Röger, Using the context-enhanced additive heuristic for temporal and numeric planning, in: Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS 2009), 2009.
[20] R.J. Firby, An investigation into reactive planning in complex domains, in: Proceedings of The Sixth National Conference on Artificial Intelligence, 1987, pp. 202–206.
[21] S. Franklin, A. Graesser, Is it an agent, or just a program?: A taxonomy for autonomous agents, in: ECAI '96: Proceedings of the Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages, Springer-Verlag, London, UK, 1997, pp. 21–35.
[22] E. Gat, Three-layer architectures, in: Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems, MIT Press, Cambridge, MA, USA, 1998, pp. 195–210.

[23] M. Georgeff, B. Pell, M. Pollack, M. Tambe, M. Wooldridge, The belief-desire-intention model of agency, in: Intelligent Agents V: Agent Theories, Architectures, and Languages, 5th International Workshop, ATAL'98, Proceedings, Springer-Verlag, 1999, pp. 1–10.
[24] M.P. Georgeff, F.F. Ingrand, Decision-making in an embedded reasoning system, in: Proceedings of the 11th International Joint Conference on Artificial Intelligence, 1989, pp. 972–978.
[25] M.P. Georgeff, F.F. Ingrand, Monitoring and control of spacecraft systems using procedural reasoning, in: Workshop of the Space Operations – Automation and Robotics, Houston, Texas, July 1989.
[26] E. Gordon, B. Logan, Game over: You have been beaten by a GRUE, in: D. Fu, S. Henke, J. Orkin (Eds.), Challenges in Game Artificial Intelligence: Papers from the 2004 AAAI Workshop, AAAI Press, July 2004, pp. 16–21.
[27] E. Gordon, B. Logan, Managing goals and resources in dynamic environments, in: D.N. Davis (Ed.), Visions of Mind: Architectures for Cognition and Affect, Idea Group, 2005, pp. 225–253 (Chapter 11).
[28] N. Hawes, A. Sloman, J. Wyatt, M. Zillich, H. Jacobsson, G.-J. Kruijff, M. Brenner, G. Berginc, D. Skočaj, Towards an integrated robot with multiple cognitive functions, in: R.C. Holte, A. Howe (Eds.), Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence (AAAI 2008), AAAI Press, Vancouver, Canada, July 2007, pp. 1548–1553.
[29] N.R. Jennings, A. Cohn, M. Fox, D. Long, M. Luck, D. Michaelides, S. Munroe, M. Weal, Interaction, planning and motivation, in: R. Morris, L. Taressenko, M. Kenward (Eds.), Cognitive Systems: Information Processing Meets Brain Science, Elsevier, 2006, pp. 163–188.
[30] R.M. Jones, R.E. Wray, Comparative analysis of frameworks for knowledge-intensive intelligent agents, AI Mag. 27 (2) (2006) 57–70.
[31] R. Knight, G. Rabideau, S. Chien, B. Engelhardt, R. Sherwood, Casper: Space exploration through continuous planning, IEEE Intelligent Systems 16 (5) (2001) 70–75.
[32] J.E. Laird, A. Newell, P.S. Rosenbloom, Soar: An architecture for general intelligence, Artificial Intelligence 33 (3) (1987) 1–64.
[33] S.A. Lisse, R.E. Wray, M.J. Huber, Beyond the ad-hoc and the impractically formal: Lessons from the implementation of formalisms of intention, in: Working Notes of the AAAI Spring Symposium on Intentions in Intelligent Agents, March 2007.
[34] P. Maes, The agent network architecture (ana), SIGART Bull. 2 (4) (1991) 115–120.
[35] R.S.N. Menkes van den Briel, S. Kambhampati, Over-subscription in planning: A partial satisfaction problem, in: ICAPS 2004 Workshop on Integrating Planning into Scheduling, 2004.
[36] F. Michaud, EMIB – computational architecture based on emotion and motivation for intentional selection and configuration of behaviour-producing modules, Cognitive Science Quarterlla 2 (3–4) (2002).
[37] F. Michaud, C. Côté, D. Létourneau, Y. Brosseau, J.M. Valin, E. Beaudry, C. Raïevsky, A. Ponchon, P. Moisan, P. Lepage, Y. Morin, F. Gagnon, P. Giguère, M.A. Roux, S. Caron, P. Frenette, F. Kabanza, Spartacus attending the 2005 AAAI conference, Auton. Robots 22 (4) (2007) 369–383.
[38] K.L. Myers, A procedural knowledge approach to task-level control, in: Proceedings of the Third International Conference on AI Planning Systems, AAAI Press, 1996, pp. 158–165.
[39] N.J. Nilsson, Teleo-reactive programs for agent control, Journal of Artificial Intelligence Research 1 (1994) 139–158.
[40] J. Peltason, F.H.K. Siepmann, T.P. Spexard, B. Wrede, M. Hanheide, E.A. Topp, Mixed-initiative in human augmented mapping, in: International Conference on Robotics and Automation (ICRA), 2009.
[41] M. Scheutz, P. Schermerhorn, Affective goal and task selection for social robots, in: J. Vallverdú, D. Casacuberta (Eds.), The Handbook of Research on Synthetic Emotions and Sociable Robotics, in: Information Science Reference, 2009.
[42] A. Sloman, Prolegomena to a theory of communication and affect, in: A. Ortony, J. Slack, O. Stock (Eds.), Communication from an Artificial Intelligence Perspective: Theoretical and Applied Issues, Springer, Berlin, Heidelberg, 1992, pp. 229–260.
[43] A. Sloman, Damasio, Descartes, alarms and meta-management, in: IEEE International Conference on Systems, Man, and Cybernetics, vol. 3, October 1998, pp. 2652–2657.
[44] A. Sloman, Architecture-based conceptions of mind, in: P. Gärdenfors, K. Kijania-Placek, J. Woleński (Eds.), In the Scope of Logic, Methodology, and Philosophy of Science, vol. II, in: Synthese Library, vol. 316, Kluwer, Dordrecht, 2002, pp. 403–427.
[45] A. Sloman, The cognition and affect project: Architectures, architecture-schemas, and the new science of mind, Tech. rep., School of Computer Science, University of Birmingham, 2003.
[46] A. Sloman, Requirements for a fully deliberative architecture (or component of an architecture), Research note COSY-DP-0604, School of Computer Science, University of Birmingham, Birmingham, UK, May 2006.
[47] A. Sloman, Architecture-based motivation vs reward-based motivation, Newsletter on Philosophy and Computers, http://www.cs.bham.ac.uk/research/projects/cogaff/architecture-based-motivation.pdf, in press.
[48] A. Sloman, R. Chrisley, M. Scheutz, The architectural basis of affective states and processes, in: J.-M. Fellous, M.A. Arbib (Eds.), Who Needs Emotions?: The Brain Meets the Machine, Oxford University Press, 2005, pp. 203–244.
[49] T. Tyrrell, Computational mechanisms for action selection, Ph.D. thesis, University of Edinburgh, 1993.
[50] T. Winograd, Procedures as a representation for data in a computer program for understanding natural language, Tech. rep. 235, MIT AI, 1971.
[51] I. Wright, Emotional agents, Ph.D. thesis, School of Computer Science, The University of Birmingham, 1997.
[52] I. Wright, A. Sloman, L. Beaudoin, Towards a design-based analysis of emotional episodes, Philosophy Psychiatry and Psychology 3 (2) (1996) 101–126.
[53] H. Zender, P. Jensfelt, Óscar Martínez Mozos, G.-J.M. Kruijff, W. Burgard, An integrated robotic system for spatial understanding and situated interaction in indoor environments, in: Proc. of the Twenty-Second Conference on Artificial Intelligence (AAAI-07), Vancouver, British Columbia, Canada, July 2007, pp. 1584–1589.
[54] S. Zilberstein, Metareasoning and bounded rationality, in: M.T. Cox, A. Raja (Eds.), Papers from the 2008 AAAI Workshop on Metareasoning, AAAI Press, Menlo Park, CA, 2008, pp. 55–59.