Extended Ramp Goal Module: Low-Cost Behaviour Arbitration for Real-Time Controllers based on Biological Models of Dopamine Cells

Swen E. Gaudl
Department of Computer Science
University of Bath
Bath, UK
Email: s.e.gaudl@bath.ac.uk

Joanna J. Bryson
Department of Computer Science
University of Bath
Bath, UK
Email: j.j.bryson@bath.ac.uk

***Abstract*—The current industrial focus in virtual agents and digital games is on complex and realistic systems that more accurately simulate the real world. This trend introduces a multitude of control parameters generally accompanied by high computational costs. The resulting complexity limits the applicability of AI systems in those domains. A possible solution to this problem it to focus on light-weight flexible systems which can be simultaneously created, controlled and run in parallel. The resulting systems are then able to control individual game characters, scaling up to large numbers of characters, forming even complex social systems. Here we demonstrate our light-weight systems-engineering approach for enriching behaviour arbitration through employing a biomimetic model. The focus of the provided augmentation for existing light-weight action selection systems is on an easy control of the behaviour maintenance, inhibition and switching of high-level goals. This can aid agent design in cases where static priorities and predefined arbitration are undesirable. The model underlying our approach is biomimetic, based on neuro-cognitive research on the dopaminic cells responsible for controlling goal switching and maintenance in the mammalian brain. This promising model is applicable to selection problems with multiple conflicting goals. We evaluate our mechanism using an existing approach as a baseline, based on our included success and evaluation criteria.**

I. Introduction

The mechanism we present in this paper addresses the issue of more responsive and flexible action selection for behaviour-based AI (BBAI) [1] or similar approaches to modular cognitive architectures. We are addressing specifically systems dealing with multiple possibly conflicting goals. The work focuses on systems that face resource constraints such that they are not able to or not intended to use a fully fledged cognitive architecture such as SOAR [2] or ACT-R [3]. Limited CPU cycles, restricted memory size, or low power consumption are only a few examples of the mentioned restrictions. Additional to the technical resources, authoring, development and testing time are important and expensive resources in industrial contexts. Our current research is addressing those issues following a previous discussion by Gaudl&Bryson [4]. To allow for a better understanding of the approach as a whole, we present implementation details as well as the results of an evaluating carried out in the MASON simulation environment [5].

To clarify on the type of problem we are addressing and to give an inspiration for its solution we start with an example which could take place in a generic role-playing or strategy game. Deciding and maintaining a logically sound or human-believable behaviour is important in games. Maintaining the suspension of disbelief is of great importance to players [6].

Example Scenario (guard in warehouse):

> A player is playing the role of a thief trying to break into a guarded warehouse. The guard can perform the three major behaviours, $patrol$, $attack$ and $extinguishfire$. The player is moving towards the warehouse and observing that the guard is patrolling the entrance. The player moves closer to the warehouse. He finds a way to set the back door of the warehouse on fire. Trying to lure away the guard on $patrol$. As soon as the fire starts, the guard switches to $extinguishingfire$ because this is triggered based upon the game designers' concept. The player tries to sneak around the guard but fails as the guard is moving towards the back entrance. The guard spots the player. She switches the active behaviour from $patrol$ to $attack$ due to spotting a thief. The player is now running away followed by the guard. After some fighting the guard kills the player and she switches back to $patrol$ as no imminent active thread is around. But what happened to the fire the player started? The back door of the warehouse is still on fire. After attending for a long period of interactions with the player the trigger signal for $extinguishfire$ was removed from the stack of sensory information for the guard.

A trivial solution would be to let the trigger remain on the stack. For this simple example this seems like a feasible option but scaling up the problem to a large set of agents and triggers, not removing stacked triggers is impossible and even distinguishing which triggers are still important is a hard problem. The main point is, designing game agents that behave in a believable and concurrent way is a complex task. Due to the large size of current games and their underlying control structures it is difficult to keep track of the maintenance and inhibition of timed actions. In digital games it is quite common to allow the AI only to occupy a fixed small number of cycles per frame as most of the computational resources are needed for the visual representation. Including a heavy-weight cognitive system to control multiple agents into such an environment is in most cases not desirable as the cognitive architecture requires generally more resources and also more design time. Additionally, designing a cognitive agent is generally more time consuming than the average static approach to game characters. It is also most of the time not needed by a designer. In the above example a designer would create—similar to a writer—a story around what the guard should do and how she should react to certain stimuli. Removing this creative process would either result in a huge impact on the players' immersion or it would require an enormous amount of computation to do meaningful story planning. Game-play designers specialise on creating human-understandable situations, reactions and characters. Automating this whole creative process is currently far beyond the current state-of-the-art in dynamic planning and story generation. However, there are promising approaches in this direction [7]. The current main interest of game AI designers and engineers it to have flexible, modular tools for creating template agents and then modify those to create the desired outcome.

Our current research is motivated by an analysis of existing cognitive systems, agent architectures and agent modelling environments for digital game agents. Existing cognitive systems such as SOAR, ACT-R and LIDA [8] are exceptionally powerful allowing the creation of sophisticated cognitive agents. However, due to the high complexity and steep learning curve they seldom leave academia and even then are mostly used in specialised communities. Whenever a full cognitive model/reasoner or a large knowledge base is not needed/applicable light-weight architectures and models such as Behaviour-Oriented Design (BOD) [9], BehaviourTree (BT) [10], Pogamut [11] or Advanced Behavior Language (ABL) [12] can be used. Those systems have lower computational costs and less steep learning curves. They additionally are more fit for non-academic application[1]. Systems implementing those light-weight approaches are generally used for individual agents or groups of agents in digital environments [12]–[14]. Due to the flexible nature of the applied approaches, the resulting system can be tailored towards a specific scenario, reducing the computational cost. This contrasts with most cognitive architectures which are intended as general problem solvers applicable to a wide range of problems.

To allow developers and researchers to enrich their action selection and behaviour arbitration mechanism we introduce the extended ramp goal model which is generally applicable to a broad range of systems. The ramp goal switching model comes with a low computational overhead allowing it to be instantiated many times, making it highly versatile. It allows for an easy way to control the maintenance, inhibition and switching of high-level behaviours in cases where static or pre-defined behaviour arbitration is undesirable for the action selection mechanism. The model is based on biomimetic concepts of dopaminergic cells in the Basal Ganglia of the mammalian brain [15], [16].

The rest of this paper is organised as follows. In the next section we describe our current research on biomimetic models and their applicability to behaviour arbitration. The section includes our contribution to behaviour augmentation, the extended ramp goal model—ERGO. For a better understanding, we present implementation details and a selected code example on how to integrate our approach into existing arbitration mechanisms. To support our argument we then present the results of our evaluation. The paper concludes with a discussion on the impact of different parameters on the model, next possible steps and future work.

[1] In digital games for example, the designer is mostly not interested in having hundreds of cognitive agents but just the impression of plausible actions for groups of those agents and intuitive means to design them.

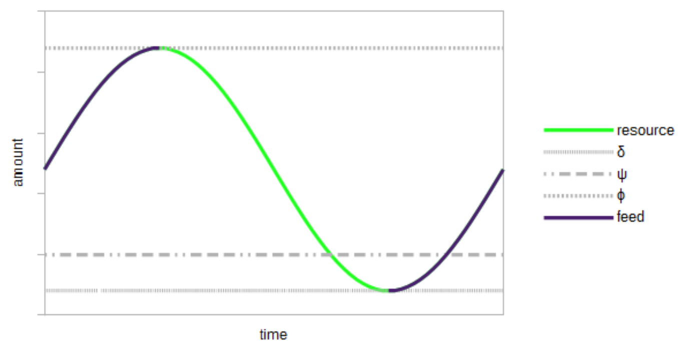

Fig. 1. A Flexible Latch using two thresholds—δ and ϕ—to control for behaviour dithering. In a simple latch, introducing those thresholds a behaviour can take control when it reaches the lower boundary δ until is reaches the upper boundary ϕ. Reaching ϕ, the behaviour is inhibited until it reaches δ again. A flexible latch adds a third threshold, ψ, above which a latch is recomputed if the agent is interrupted. The best threshold for ψ was found to be $\psi = \delta$ [17].

II. Approach: Biomimetic Models

In this section we discuss our biomimetic mechanism and its implications on scalable behaviour arbitration. We start by presenting our motivation for applying biomimetic concepts to action selection schemes. This illustration of the current state leads to our argument for the ramp-function arbitration mechanism.

We took Flexible Latching [17] as a starting point for our research. Flexible Latching introduces a simple latch, see Figure 1, which reduces the *dithering*—a rapid switching between goals. Thus, more time is spent in transition than in useful pursuit and consumption of single goals. Unlatched behaviour is executing once the trigger condition is met and stops thereafter. The latch thereby acts similarly to a hysteresis function.

For the sake of an illustration, imagine following example:

> A leaking canister looses water over time. As soon as a low water level, threshold ϕ, is reached, the canister is filled up again to that level. If you only refill up to ϕ whenever the water is below ϕ, the time between each re-fill is relatively short. A strict latch now adds another threshold δ on top of the lower threshold. Now, whenever the water reaches ϕ you spend your actions to refill the water until is reaches the higher level, threshold δ.

Such a latch is useful under the assumption that the amount of units refilled with a single action is greater than the amount used during the performed action. The inclusion of the strict latch allows extra time between δ and ϕ which can be spent on alternative actions. Flexible Latching extends the Strict Latching by dealing with interrupts and re-evaluating whether the current goal should still be pursued. It was shown that this is also more efficient [17], as the agents do not pursue goals that are neither selected nor convenient after the interruption.

Taking a closer look at selection processes inspired by nature, neural networks (NN) are the most prominent. Using a neural network, it is possible to learn and solve selection tasks for problems where an algorithmic description of the problem is not known or costly. The NN is able to approach an solution only by providing it with a set of known input and solution pairs to adapt itself towards the solution space. However, for NNs the overall action selection or computational process is not transparent, thus ‘tweaking’ a Neural Network to perform in a certain way is difficult. Also, NNs are normally trained to solve static problems rather than a continuous problem like action selection for real-time behavioural control. For example, the commercial game Black&White by Lionhead used a neural network for training the player’s pet.

Moving from neuronal networks to a single neuronal model reveals some of the interesting underlying mechanics which can be exploited in other contexts as well. There exists a variety of activation functions for neuronal models. Those include the spike or Dirac used in Spiking Neural Networks (SNN), the sigmoid which has a fixed output range between zero and one, and the ramp function which combines a monotonic increased activation and an instant activation drop.

Biomimetic models like NNs are an important asset of the computer science tool-set. They present good and scalable solutions for addressing complex problems. We found that the ramp function is favoured for goal arbitration [18], as we review in the following section. This finding motivates our present approach as we believe it to be a simple and elegant mechanism for augmentation.

A. The Extended Ramp Goal Model (ERGo)

Current research [15], [16] suggests, that dopamineric cells in the Basal Ganglia of the mammalian brain are likely to be responsible for the maintenance and switching of goals and thus behaviours. Those cells exhibit a ramp-like activation in a selected brain area when pursuing a goal. We will use this as a base for our current model, see Figure 2. The two important features of the exhibited ramp are the linear strictly monotonic climbing of the activation and the instant activation drop exhibited when reaching the success criteria for the goal. These provide a predictable yet flexible mechanism.

Using a generic behaviour-based action selection mechanism we illustrate how the extended ramp works. For a given set of behaviours $B = \{B_1, \ldots, B_m\}, m \in \mathbb{N}$ we introduce a set of ramps $R = \{R_1, \ldots, R_n\}$ and goals $G = \{G_1, \ldots, G_m\}, n \in \mathbb{N}, n \leq m$. R_a is the ramp for B_a and G_a is the goal which B_a is trying to satisfy, $a \in \{1, \ldots, n\}$. The additional Behaviours $B_b, bin\mathbb{N}, b \leq m - n$ try to satisfy Goals B_b without being augmented with a ramp. Each time step t R_a adjusts its activation based on the boolean activation state $\alpha_a(t)$ of the behaviour B_a, the boolean urgency signal $\upsilon_a(t)$ and the stickiness $\omega_a(t)$ of the behaviour. All ramps share the same increment i and activity multiplier μ which define the accumulated activation in the following way.

$$R_a(t) = \begin{cases} R_a(t-1) * \mu & \text{if } \upsilon_a(t) = 1 \\ R_a(t-1) + i & \text{if } \alpha_a(t) = 0 \\ R_a(t=0) & \text{if } \alpha_a(t) = 1 \wedge \omega_a(t) = 0 \\ R_a(t-1) + (i * \mu) & \text{if } \alpha_a(t) = 1 \wedge \omega_a(t) > 0 \end{cases}$$

The influence of an active behaviour on the activation is presented by $\alpha_a(t) = 1$ and $\omega_a(t) > 0$. This results in an activation modified by our activity multiplier μ.

$$R_a(t) = R_a(t-1) + (i * \mu)$$

The increased activation is supported by the work of Redish [18]. He states that the goal cell in the Basal Ganglia have a higher firing rate when that related goal is pursued. Even when a behaviour is not active it still gains activation.

$$R_a(t) = R_a(t-1) + i$$

The combination of those two mechanisms removes most of requirements of needing a direct binary switch for the behaviours to arbitrate successfully. We believe that this minimizes the direct competition between behaviours as well and increases the robustness of the action selection in cases of noisy switching signals. Thus, our approach contrasts the currently available selection principles in games. These heavily use binary triggers as they are initially easy to implement and understand.

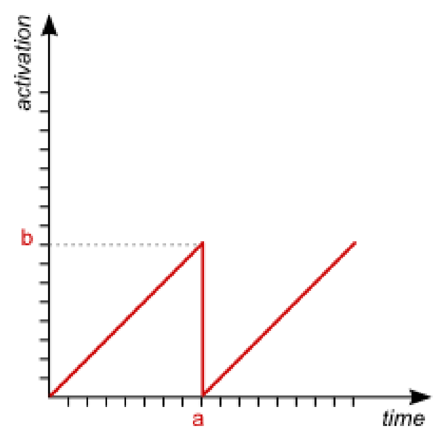

Fig. 2. A single ramp function used for inhibiting a behaviour. A behaviour controlled by a ramp is only inhibited when another behaviour gains a higher activation or once its goal is reached. Once the goal is reached activation instantly drops. The behaviour completes its goal at time a with a certain activation b.

To allow for the agent to influence if a behaviour needs to be urgently triggered, the agent is able to trigger the urgency signal υ. Upon receiving the signal, the ramp amplifies its activation by the factor μ. An example for an amplified behaviour is $Behaviour3$ in Figure 3 which is triggering υ at $t = 41$. The activity multiplier μ is a percentage based influence on the global action selection. Using μ for urgent execution results in an exponentially growing activation level. For our experiments we generally set μ to a value within the range of 1.0 and 2.0. If $\mu = 1.0$, activation is not affected by the urgency signal at all. If $\mu = 2.0$, the activation is increasing quadratic. We have not yet investigated the impact of negative urgency has on agents. Negative urgency would be reflected by

$0 < \mu < 1.0$ and would result in a decay or dampening of the activation level. If B_a needs to urgently execute, υ_a is set to $true$. This indicates the need for a rapid behavioural change. The result of using the urgency signal υ is expressed in a similar way to the natural phenomenon inside the mammalian brain where it takes a small amount of time for the activation to spread before even urgent actions are executed. The time span between the trigger and the execution of the behaviour however is small.

As one of our aims is to simplify the action selection process we focus on as low coupling of the ramp goal model with the rest of the agent as possible. Thus, we decided to introduce υ as an urgency signal and utilise μ to amplify the activation of our model. Using only asynchronous signals, we do not need to include problem specific components like the the agent boundaries. This made our model easier to comprehend and integrate as the boundaries should normally be handled directly by the behaviour primitives.

We now introduce the stickiness ω of goals into our mechanism. It is based on mammalian behaviour in rats and other animals when feeding after a period is reduced available resources. The animals do not stop feeding even if their stomach reached its capacity—in the literature referred to as binging [19]. Once the behaviour is active and the goal conditions are met—$\alpha = 1$ and the agent has accumulated enough resources of one type to reach δ—the stickiness is decreased until is reaches zero. During this time the behaviour still accumulates activation. In other words, the agent—even though the goal G_a for a behaviour B_a is met— continues to pursue B_a until $\omega = 0$.

$$R_a(t) = R_a(t-1) + (i * \mu)$$

The only way to interrupt this is either by having a higher activation due to an urgency signal or due to an environmental interrupt which disturbs the current behaviour and resets the activation to the lower boundary. Both phenomena are also present in nature. For example, an animal is feeding and a predator jumps out of cover. If the current feeding behaviour is not instantly interrupted the animal would simply die. The stickiness ω of ERGO is similar to a latch but is encapsulated in ERGO. Its purpose is to allow the agent to handle environments where resources are sparse. It is part of our internal model and hidden from the agent to allow for an easier integration minimizing the parameters exposed in the agent to reduce the cognitive load during design time.

Based on the description of the extended ramp so far, we largely focused on explaining the mechanism of a single ramp. The interaction between multiple ramps is handled within the execution frame of each augmented behaviour. Whenever a behaviour tries to gain the control it is validating if other behaviours have a higher activation—a mechanism similar to the Basal Ganglia. If those can execute they will suppress the behaviour trying to gain control. Thus, there is always only one behaviour $B_i \in B$ of augmented behaviours active. Due to this restriction we are conforming with the rest of the underlying hierarchically ordered action selection mechanism without overriding the general priority scheme.

In contrast to most ramp function related selection approaches [20], [21] which apply ramp functions in the context of neural networks, our approach is the first attempt to apply a ramp-like criteria directly to a behaviour-based arbitration process without using a neural network to control the maintenance.

B. Integration

In the following subsection we present the integration of ERGO into a specific agent model and simulation[2]. The used action selection mechanism is the parallel-rooted ordered slip-stack hierarchical planner (POSH) [22]. Due to the modular nature of POSH we can integrate ERGO as an additional subcomponent into the action selection mechanism without having to change large portions of existing code or the general action selection scheme. To allow for a better comparison of the results we modified the original Flexible Latching [23] code base which is freely available. By extracting the Latching code from the behaviours we were able to integrate the resource storage $\_energy$ and its adjustment back into each of the behaviours. This made the whole code more transparent without changing the functionality and exhibited behaviour. A benefit of this refactoring is that now it is more visible how and when resources are accumulated, see the behavioural action a_drink in Code 1. The refactoring was required because before the behaviour had no internal representation of its resource state other than the latch.

```python
def a_drink(self):
  if self.inter.should_interrupt(self._energy):
        GoalCell.reset(self)
        self.prev_target_loc=self.drink_target.loc
        self.target=None
        self.signal_interrupt()
        self.inter.increase_count()
        return 0

  if not self.target.agent.Resources.s_has_food_left():
        self.signal_interrupt()
        self.target=None
        return 0

  self.target.agent.Resources.a_reduce_food_load()
  self._energy += common_increment

  if self._energy > common_upper:
       self.reached_goal()
       return 0
  return 1
```

Code 1: Python code illustrating the inclusion of ERGO into an existing behaviour.

An agent's action can be split into three distinct parts. The first part–line 1 to 8–is responsible for environmental interrupts. Those are controlled by the simulation environment. If the ramp should reset the activation it is triggering $GoalCell.reset(self)$. This results in a re-evaluation of the internal activation. The second part until line 16 is responsible for leaving a food patch when it is empty or to feed on a resource patch.

The last part is referring to the goal criteria, telling an agent that it is done accumulating resources and that the ramp could now drop activation. This is done inside the $reached\_goal$ method which is reducing the stickiness and resetting of the ramp once the stickiness is zero.

[2]The simulation itself is discussed in the subsequent section.

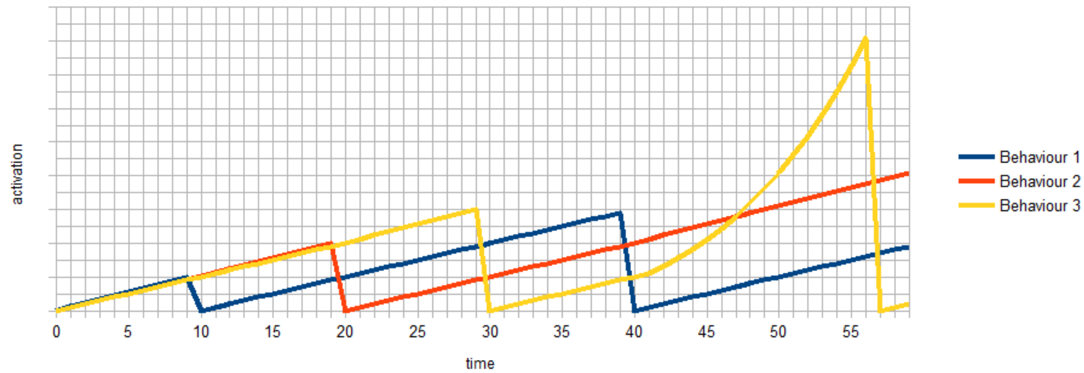

Fig. 3. Internal activation levels of three behaviours using ERGO. From time $t = 0$ to $t = 9$ and $t = 30$ to $t = 39$ $Behaviour1$ is active having an higher activation due to the larger inclination. At time $t = 9$ the success criteria for the first behaviour is met and the activation drops resulting in the activation of the second behaviour. $Behaviour2$ is active from $t = 10$ to $t = 19$ where its goal is reached. As all behaviours have the same inclination, they automatically schedule into an activation pattern. At $t = 41$ the urgency signals is triggered for $Behaviour3$ resulting in an exponential gain of activation and an activation at $t = 47$.

```
def reached_goal(self):
  if not self._active:
    return
  if self._sticky > 0:
        self._sticky -= 1
  else:
    self._activation = self._lower_bound
```

Code 2: ERGO's $reached\_goal$ definition, reducing the stickiness whenever the goal criteria is met.

C. Biomimetic Model Summary

Current research on the Basal Ganglia suggests that the goal maintenance in the mammalian brain is controlled by a ramp-like activation function. Here we present a new mechanism—ERGO—which extends the application of the ramp beyond neural networks to more abstract and open action selection systems. The augmented behaviour is able to react to sudden changes in the environment. The communication between the extended ramp and the behaviour is through a well defined and sparse signal flow. The implementation is using a low-cost computational model of the ramp and is based on a Python agent using POSH [9] action selection.

In the next section we describe our test domain where multiple conflicting goals can arise for an agent. Natural agents from single cell paramecia to human beings face this situation constantly. For example, a small child indecisive if it should sleep because it is tired or continue to play because it is fun. Additional information on the agents' behaviours is presented to allow for a better understanding of the domain and the possible actions of an agent.

III. SIMULATION

To develop and test our model, we chose Behaviour Oriented Design (BOD) as our test-platform lightweight architecture. BOD allows the description of cognitive agents utilizing the parallel-rooted slipstack hierarchical (POSH) dynamic planning language. POSH includes a linear goal structure where each goal has a fixed priority with respect to the others, although each goal can be inhibited either by having un-met preconditions or through a system of scheduling. One reason POSH is well-suited for our experiments is because it has already been fitted with a modification to this structure to allow more biologically-plausible action selection. This mechanism is Flexible Latching [17] described earlier in section II-A. As a simulation environment we decided to use the MASON agent-based modelling platform [5] because of its well-defined and easy to use Java interface.

The simulation environment is a refinement of the example domain from Gaudl&Bryson [4] and similar to *Sim1* used by Rohlfshagen&Bryson [17]. The world contains two resource types, water and food, equidistant from the centre of the map in 150 units. The world is 600 by 600 units and the agents start at the centre of the map, see Figure 4.

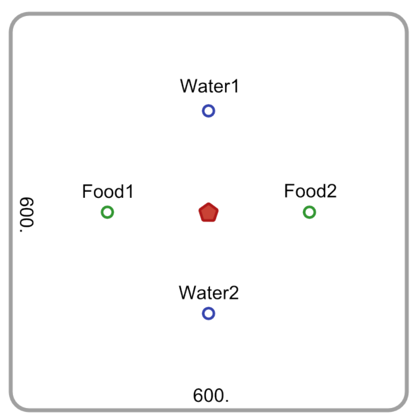

Fig. 4. Simulation Environment in a Mason agent simulation. The world is 600x600 units. It contains two food and two water sources equidistant from the centre. All agents spawn at the centre at time $t = 0$.

Agents can travel two world units in any direction for every tick of the system clock[3]. The map is wrapped around the

[3]To simplify our model we are using discrete time steps instead of real-time calculations which not only allows more fine-grained control it also allows us to speed up our simulations beyond real-time.

horizontal and vertical edges. If an agent travels only in one direction it will create circular path around the world. Due to the layout of the map there is no benefit to travel over the map edges as the distances are exactly the same. It is also noteworthy to mention that an agent cannot block a path, resource, or another agent in any way, which would be possible in nature but introduce unnecessarily complicated dynamics for the task at hand. The only time agents interact is during grooming.

Each agent constantly uses 0.1 resource units of water and food each tick to survive, simulating natural metabolic costs and presenting the problem of self sustenance. The amount of energy needed does not change during the simulation even if an agent does not move. If an agent's accumulated store of one of the two resources it needs drops to zero, then the agent dies. All agents are initialized within a lower boundary δ and upper boundary ϕ for the two resources. Whenever an agent is feeding from one the resources it gains energy, 1.1 units of the resource. The gain is set to be larger than the consumption otherwise the agent would have no chance of surviving. For our setting the gain is set to ten times the metabolic cost.

To allow the agent to track when it urgently needs to feed on a resource, we make its intelligence sensitive to when its units of a specific resource drop below δ—an artificial threshold we use to model hunger. Whenever the units reach the upper bound ϕ the agent is programmed to detect that it has satisfied the need for that resource, so that it may distribute its time across other of its goals. The shortest path between one food and water resource requires an agent to spend approximately 10 units of both resources which is the amount it can gain from feeding for one tick.

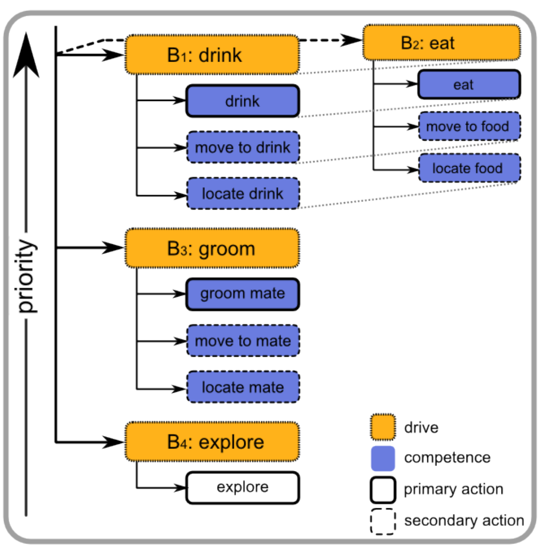

Fig. 5. A condensed view of a drive collection. It specifies the behaviour of one of the agents in the simulation and contains four behaviour drives, prioritized top to bottom. Drives B_1 and B_2 have equal priority, meaning they are equally important and their priority must be arbitrated in some sensible manner so both can be achieved.

In Figure 5 we present a simplified version of the POSH action plan. This plan is used for all agents in our simulation. Each agent has four drives which are prioritized based on each drive's position in the action plan. The higher the drive in the plan the higher its priority. Each drive is designed to satisfy a specific goal of the agent, for example drive B_1 represents the need to drink. In POSH those goals are specified by internal or external senses, in this case the sense $wantstodrink$. There is a special case which is behaviour B_4– the lowest-priority drive. The lowest drive should always be able to execute as it is treated as a fallback as well. If no drive can be executed the plan terminates and the agent will stop and terminate as well. The behaviours B_1 and B_2 have equal priority indicating they are equally important to the agent—both are required for its survival. At this point we introduce our biomimetic augmentations to ensure that both drives *are* met in an efficient way, with neither dithering nor neglect.

IV. Results

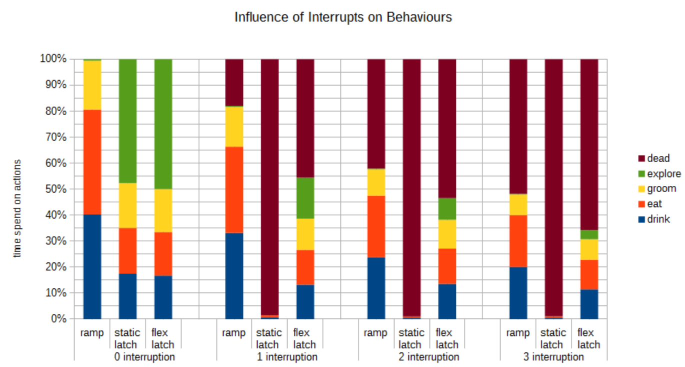

Fig. 6. Comparing the three behaviour augmentations Static Latch, Flexible Latch and ERGo. Illustrated is the change in invested time for an interrupt progression $i = [0, 1, 2, 3]$. As the interrupts increase the static latching becomes unable to arbitrate behaviours appropriately. This results in a high death of agents. ERGo and Flexible Latch are able to adapt to the interruptions. ERGo agents remain significantly more alive.

We ran an initial set of 15 independent trials per parameter to analyse the influence of each tested parameter on the augmentation and re-ran all simulation with the Flexible Latching model to have a direct comparison on the same system. We allowed each trial 5000 ticks, as in most cases the simulation either converged to stable state (death of all agents or stable surviving agents) before that time. We increased the number of trials to 50 where we reached stable results with a low standard error. We first started to analyse how well both approaches—Flexible Latching and ERGo—are able to handle non hostile environments. In non hostile environments both models perform well. Due to the random initialisation of the resources for each agent's internal storage the standard deviation for all agents can be quite large. We compensate for this with the larger number of trials.

To judge the quality of a well performing augmentation we use following criteria.

Evaluation criteria:

1) time the agent remains alive
2) time left for individual behaviours
3) robustness in face of noise and interruptions
4) programmability

For the following experiments we set the lower threshold $\delta = 40$ and the saturation threshold $\phi = 44.5$. First we tested the augmented agents without interrupts. In all trials for this setting all augmented agents remain alive, see Figure 6 first three bars. Both Latches invest a fixed amount of time on the two highest priorities and then spend the remaining time on lower priorities. As exploration does not have any additional requirements compared to grooming the largest fraction of time is invested in it. In contrast to that, ERGO invests more time in the higher priority goals and then the remaining time to groom. If grooming is not possible for ERGO it is performing exploration. This result is based on the mechanisms underlying the Latch where a fixed threshold guarantees that extra time is invested in other actions. ERGO however only applies a stickiness to a goal resulting in generally more actions per goal. The actions however can be interrupted more easily which is visible in Figure 7 once the interrupts increase.

As the interrupts increase from $i = 0$ to $i = 3$ the Static Latch is persisting on executing actions which are not advantageous. Flexible Latch is able to handle the interrupts better than Static Latch, visible in the lower death rate. It scales down all actions equally. This puts a high pressure on the agent as the life essential actions are also reduced. ERGO however scales best as urgent behaviours inhibit others from executing when they need to execute instead. Life essential behaviours maintain the highest priority but lower level goals are still pursued.

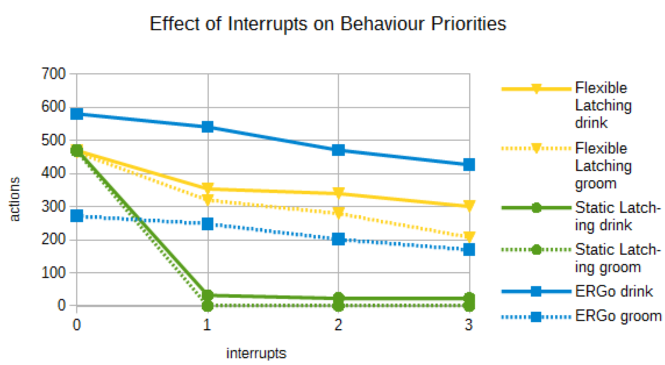

Fig. 7. Comparing the effects of interrupts on the priority hierarchy of behaviours—demonstrated by drinking and grooming. The amount of higher and lower priority behaviours is nearly equal for both Latches allowing an equally high proportion of lower priority behaviours to be executed. Once interrupts increase, the Static Latch is unable to remain in a stable state—most agents die. Flexible Latch and ERGO scale down the amount of actions when interrupts increase. However, the actions for Flexible Latch are decreasing disproportionate compared to ERGO.

Figure 6 illustrates how the differentiation between lower and higher priority behaviours in handled in both Latches and ERGO. With increased interrupts ERGO and Flexible Latch scale down but ERGO maintains a similar ratio of higher and lower prioritised behaviours.

As ERGO responds only to signals by the agent it does not optimize free time as efficiently as the hand-tuned Flexible Latch. However, our approach minimizes the interdependence with the specific parts of an agent. Thus, increasing robustness and programmability in our agents. We have not specified problem dependent parameters in ERGO to allow for a better integration into other action selection mechanisms.

We focus with the current experiments on noise in the decision process and especially on interrupts. Thus, allowing us to analyse how well an agent is able to handle non-scripted situations, e.g. unpredicted player interactions in a game. Increasing the interrupts is in some ways similar to players probing or testing an agent or system by trying to finding a way of breaking it. In heavily scripted games or full information games the agents are normally not affected by such attacks. However the more agency, dynamic planning and uncertainty is introduced into games, the easier it is to break the agents due to the need to react to different stimuli depending on the situation.

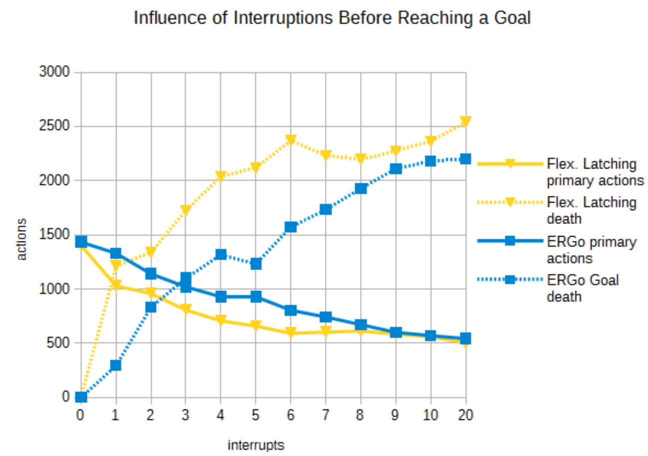

Fig. 8. Influence of increasing numbers of interruptions on death rate and executed primary actions—eating, drinking and grooming—for Flexible Latch and ERGO. Flexible Latch presents a higher death rate in all settings. For the number of high priority actions, ERGO and Flexible Latch start equally, around 1500 actions. As interrupts increase, ERGO performs significantly more high priority actions until 9 interruptions per successful behaviour.

We present the results of a further interrupt increase in Figure 8. Here we increase the interrupts from 0 to 10 in a linear fashion and then increase them as a final step to 20 to see if some major changes or converging behaviour is emerging. Two interesting observations are possible from the figure. The first is the point where death rate and primary actions cross for each augmentation. This point indicates a shift in the agent behaviour where on average the agent looses a lot of activity and liveliness. For Flexible Latching this point is around 1 interrupt per goal attempt. ERGO reaches the same point at three interrupts. This suggests that ERGO augmented behaviour at least in our experiments are more resilient in terms of interrupts. The second observation supporting the previous suggestion is that, while ERGO is performing more primary actions per simulation, the death rate is always lower than for Flexible Latching. It can be argued that a change of latch size or the lower threshold δ could compensate for that. But the main point we want to stress is that this hand-tuning can also be done for ERGO when modifying the stickiness of goals or the activity modifiers.

Summarizing the results: In this section we presented experimental results from our evaluation of our extended ramp goal model—ERGO. We compared our approach with a similar biomimetic approach—Flexible Latching [17]. Our experiments stressed the ability of both approaches to handle noisy action selection based on interrupts in the selection process. We focused on an environment where action selection was already difficult. In the beginning of this section we specified our evaluation criteria defining good results. Throughout the section we presented experimental results indicating that ERGO is able to handle more interrupts keeping agents longer alive. We show in Figure 7 that our approach scales down without sudden quality fall-offs. Due to the adjustments of Flexible Latching to the scenario ERGO is not better than Flexible Latching in terms of extra time for individual behaviours. However, ERGO's integration requires less hand-tuning and ERGO itself is better encapsulated and more robust, based on its own independent internal ramp and the usage of asynchronous signals. In the next section we draw our conclusions from the experimental results and where we think further investigation is still needed.

V. Conclusion

Action selection in digital games is a crucial part of contemporary game AI. As the game environments get more and more dynamic new ways of controlling and designing game characters are needed. Here we present an approach for behaviour augmentation applicable to a wide range of behaviour based AI techniques such as POSH [9] and BT [10], only to give two examples. Our experimental results indicate that ERGO indeed is a robust, generic approach which provides good results in noisy environments. Due to its internal ramp and its loose coupling to the internals of an agent, the inclusion in existing approaches should be straight-forward. It performs well even when not adjusted to an experimental setting. However, further experiments in actual game environments and action selection mechanisms are needed to verify its general performance. For our experiments, all behaviours use the same configuration of the extended ramp with the same inclination gain. Future work should involve optimizing the inclination gain based on initial priorities of the behaviours allowing a more fine grained approach to scheduling the arbitration process. In our setting, ERGO outperforms Flexible Latching [17] based on our evaluation criteria presented in section IV. As games evolve—requiring more versatile and scalable techniques to handle dynamic environments—we believe, that light-weight cognitive architectures and generic approaches to action selection offer tremendous potential. Thus, we believe further research in light-weight cognitive architectures and scalable action selection is needed to provide stable solutions which are applicable in industry settings.

References

[1] R. Brooks, "A robust layered control system for a mobile robot," *Robotics and Automation, IEEE Journal of*, vol. 2, no. 1, pp. 14–23, 1986.

[2] J. E. Laird, A. Newell, and P. S. Rosenbloom, "Soar: An architecture for general intelligence," *Artif. Intell.*, vol. 33, no. 1, pp. 1–64, 1987.

[3] J. R. Anderson, *Rules of the mind*. Psychology Press, 1993.

[4] S. E. Gaudl and J. J. Bryson, "A biomimetic model of behaviour arbitration for lightweight cognitive architectures," in *Philosophy and Computers*, ser. APA Newsletter, P. Boltuc, Ed. The American Philosophical Association, 2014, [submitted].

[5] S. Luke, G. C. Balan, L. Panait, C. Cioffi-Revilla, and S. Paus, "Mason: A java multi-agent simulation library," in *Proceedings of Agent 2003 Conference on Challenges in Social Simulation*, vol. 9, 2003.

[6] J. Murray, "From game-story to cyberdrama," in *First person: New media as story, performance, and game*, N. Wardrip-Fruin and P. Harrigan, Eds. MIT Press Cambridge, MA, 2004, pp. 2–11.

[7] M. Mateas, "Interactive drama, art, and artificial intelligence," Technical Report CMU-CS-02-206, School of Computer Science, Carnegie Mellon University, December 2002.

[8] S. K. D'Mello, S. Franklin, U. Ramamurthy, and B. J. Baars, "A cognitive science based machine learning architecture," in *AAAI Spring Symposium: Between a Rock and a Hard Place: Cognitive Science Principles Meet AI-Hard Problems*. AAAI, 2006, pp. 40–45.

[9] J. J. Bryson, "Intelligence by design: Principles of modularity and coordination for engineering complex adaptive agents," Ph.D. dissertation, MIT, Department of EECS, Cambridge, MA, June 2001, AI Technical Report 2001-003.

[10] A. J. Champandard, *Ai Game Development*, L. Thibault, Ed. New Riders Publishing, 2003.

[11] J. Gemrot, R. Kadlec, M. Bída, O. Burkert, R. Píbil, J. Havlíček, L. Zemčák, J. Šimlovič, R. Vansa, M. Štolba, T. Plch, and B. C., "Pogamut 3 can assist developers in building ai (not only) for their videogame agents," in *Agents for Games and Simulations*, ser. LNCS. Springer, 2009, no. 5920, pp. 1–15.

[12] M. Mateas and A. Stern, "A behavior language for story-based believable agents," *Intelligent Systems, IEEE*, vol. 17, no. 4, pp. 39–47, 2002.

[13] J. Gemrot, C. Brom, J. J. Bryson, and M. Bída, "How to compare usability of techniques for the specification of virtual agents behavior? An experimental pilot study with human subjects," in *Proceedings of the AAMAS 2011 Workshop on the uses of Agents for Education, Games and Simulations*, M. Beer, C. Brom, V.-W. Soo, and F. Dignum, Eds., Taipei, May 2011, pp. 38–62.

[14] S. E. Gaudl, S. Davies, and J. J. Bryson, "Behaviour oriented design for real-time-strategy games – an approach on iterative development for STARCRAFT AI," in *Proceedings of the Foundations of Digital Games*. Society for the Advancement of Science of Digital Games, 2013, pp. 198–205.

[15] R. Cools, *Chemical Neuromodulation of Goal-Directed Behavior*, ser. Strüngmann Forum reports. Cambridge: MIT Press, 2012, ch. Search, Goals, and the Brain, pp. 111–125.

[16] J. W. Brown and D. E. Nee, *Executive control of cognitive search*, ser. Strüngmann Forum reports. Cambridge: MIT Press, 2012, ch. Search, Goals, and the Brain, pp. 69–80.

[17] P. Rohlfshagen and J. J. Bryson, "Flexible latching: A biologically-inspired mechanism for improving the management of homeostatic goals," *Cognitive Computation*, vol. 2, no. 3, pp. 230–241, September 2010.

[18] A. D. Redish, *Search Processes and Hippocampus*, ser. Strüngmann Forum reports. Cambridge: MIT Press, 2012, ch. Search, Goals, and the Brain, pp. 81–96.

[19] W. F. Mathes, K. A. Brownley, X. Mo, and C. M. Bulik, "The biology of binge eating," *Appetite*, vol. 52, no. 3, pp. 545 – 553, 2009.

[20] T. C. Stewart, T. Bekolay, and C. Eliasmith, "Learning to select actions with spiking neurons in the basal ganglia," *Frontiers in Neuroscience*, vol. 6, no. 2, pp. 1–14, 2012.

[21] J. D. Velsquez, "Modeling emotion-based decision-making," *Emotional and intelligent: The tangled knot of cognition*, pp. 164–169, 1998.

[22] J. Bryson and L. Stein, "Modularity and design in reactive intelligence," in *International Joint Conference on Artificial Intelligence*, vol. 17, 2001, pp. 1115–1120.

[23] P. Rohlfshagen and J. J. Bryson, "Flexible latching: A biologically-inspired mechanism for improving the management of homeostatic goals," *Cognitive Computation*, vol. 2, no. 3, pp. 230–241, 2010.

[24] P. Todd, T. Hills, and T. Robbins, *Cognitive Search: Evolution, Algorithms, and the Brain*, ser. Strüngmann Forum reports. University Press Group Limited, 2012.