

Dynamic Computation and Context Effects in the Hybrid Architecture AKIRA^{*}

Giovanni Pezzulo^{1,2} and Gianguglielmo Calvi¹

¹ Istituto di Scienze e Tecnologie della Cognizione - CNR
Viale Marx, 15 - 00137 Roma, Italy

² Università degli Studi di Roma “La Sapienza”
Piazzale Aldo Moro, 9 - 00185 Roma, Italy
giovanni.pezzulo@istc.cnr.it, gianguglielmo.calvi@noze.it

Abstract. We present AKIRA, an agent-based hybrid architecture designed for cognitive modeling. We describe some of the underlying ideas motivating its development, such as the possibility of exploiting distributed representations and performing parallel, dynamic and context aware computation. We illustrate its main components and capabilities and compare it with some related cognitive architectures, such as DUAL and Copycat. We present also a sample simulation in the visual search domain, exploiting AKIRA’s peculiarities for cognitive modeling.

1 Introduction

Cognitive modeling is a vast area of research today; however, there are still few computational models that are both credible from the theoretical point of view and powerful from the technological one. The most notable exceptions are the generic, unified architectures SOAR [24] and ACT-R [1]; some other architectures, such as the DUAL/AMBR [14, 15] and Copycat [11] models of analogy, do not address all the cognitive processes, but have a general scope, because they capture some of the underlying principles of high-level cognition.

AKIRA [21, 28] is not a cognitive architecture in itself, but a framework for implementing and testing cognitive models at different levels of complexity and integration. We have tried to make the platform as versatile as possible: AKIRA includes a rich toolkit of mechanisms and algorithms from different paradigms, allowing developers to test, compare and possibly integrate many symbolic and connectionist models. However, even if AKIRA does not commit to a single cognitive model, some theoretical choices are in a sense embedded: our main source of inspiration are the Society of Mind [19] and the Pandemonium [12] models. From the architectural point of view, AKIRA is inspired by DUAL [14] and Copycat [11]. AKIRA integrates Multi-Agent and Pandemonium features. A server process (the *Pandemonium*) executes and monitors many agent instances (the *Daemons*); differently from standard MAS architectures, agents have links, forming an *Energetic Network*, that affords energetic exchanges. The objective is to create architectures performing parallel, dynamic and emergent computation

^{*} This work is supported by the EU project **MindRACES**, FP6-511931.

by distributing the operations between many simple, interacting processes, each one carried on by an agent. The advantages of distributed representations are explored e.g. by [5]; they have been also shown [10, 15] to be very important for modeling contextual effects.

In AKIRA, as well as in DUAL and Copycat, context is accounted as a set of *pressures* introduced by (partially) active elements over the current computation. AKIRA and DUAL use a set of micro-agents, each having an activation level for each instant and each representing a single object, concept or process; the agents are hybrid, carrying on both *symbolic content* (e.g. a frame) and *connectionist elements* (an activation level, links to the other agents). For example, in one of the main applications of the architecture DUAL, the AMBR [15] model of analogical reasoning, each micro-agent includes a frame representing a single object, e.g. a cup, or a plate. While *objects* are thus localist representations, *situations* are fully distributed ones: e.g. *the cup is on the plate* is represented by a number of partially activated micro-agents; moreover, the patterns of activation change dynamically because of system evolution (e.g. decay) and thanks to new elements or events (e.g. the cup is broken). By exploiting distributed representations AMBR is able to perform analogical reasoning in a dynamic and context dependent way. Many analogy phases, i.e. analog access, mapping, and transfer are performed as *parallel subprocesses* rather than serial stages. The result of the computation dynamically emerges from the parallel and concurrent activity of many partially active agents, carrying on portions of semantic information both in their symbolic and connectionist parts. AMBR models many other cognitive processes, including blending and intrusions, priming, re-representation of episodes, problem solving [10, 15], etc. Its results have been successfully compared to human data [16]. In a similar way, Copycat [11] performs analogical reasoning on the basis of a representation of the problem space that is not pre-fixed, but is computed and may change dynamically during the analogical processing. Representation-building and mapping (e.g. of two situations) run in parallel and influence each other.

AKIRA exploits a similar apparatus for context-aware goal processing. We introduce its main underlying theoretical assumptions in Section 2. In Section 3 we present its main components and features. In Section 4 we compare AKIRA with some related cognitive architectures: DUAL [14, 15], Copycat [11], IDA [9] and the Behavior Networks [17]. In Section 5 we present a sample simulation in the visual search domain.

2 Principles of AKIRA

Here we present the main inspiring principles of AKIRA, focusing on how distributed representation and control can be used for cognitive modeling.

Hybridism and Locality Principle. AKIRA is a hybrid system; differently from many others, AKIRA exploits agents that are *hybrid at the micro-level* [14]:

each agent has both a connectionist and a symbolic component. The connectionist component involves the activation level of the agent as well as the energy exchanges between the agents; moreover, agents can group and organize into higher level assemblies called *Coalitions*, that are able to solve together composite tasks that are impossible to solve alone. The symbolic component involves the set of operations an agent can perform; each agent is a specialized computational unit³. The operations can range from simple to very complex ones (e.g. AKIRA includes many mechanisms and algorithms). However, in accordance with the underlying distributed approach, each agent should be specialized, thus complex tasks are more likely performed by Coalitions of cooperating agents.

AKIRA also follows a *Locality Principle*: each interaction between the agents, both connectionist and symbolic, is implemented as a peer-to-peer operation, without centralized control. Hybridism and locality permit to design a continuum of computation styles, ranging from centralized, hierarchical control to distributed, emergent computation (e.g. in modular, layered or distributed architectures). The connectionist side of AKIRA endorses the emergent and distributed phenomena, exploiting the patterns of activation of the agents. The symbolic side permits both to introduce top-down drives and structure and to manage operations requiring semantic compositionality or explicit communication.

The Energetic Metaphor. Agents' symbolic activity is influenced by their connectionist side. The agents dynamics follow an Energetic Metaphor [14, 20]: *greater activation corresponds to a greater computational power*, i.e. speed. Each agent has an amount of computational resources that is proportional to its activation level (and is a measure of its relevance, both absolute and contextual, in the current computation). More active agents have a priority in their symbolic operations and their energetic exchanges. This mechanism permits to model a range of cognitive phenomena such as context and priming effects, because active agents are able to influence the others. As a consequence of the dynamics of system (priority is related to the activation) and energetic exchanges between related agents, in facts, each agent introduces a *contextual pressure* over the computation even without explicit operations, but only being active; for example, if an agent that recognizes a given visual feature is active, it will activate or inhibit other visual agents, or it will influence the way other agents operate on the same stimulus. Moreover, the results of its computation can be available to other agents (as in the Pandemonium model), e.g. to higher-level feature-integration agents; this capability will be used in the example in Section 5. The converse is also true: agents that are salient in the same context evolve stronger links and are able to recruit more energy. Many kinds of contextual pressures can be naturally modeled using this schema, e.g. perceptual, goal-driven, cultural, conceptual, memory-based, etc.

³ It is important to notice that even subsymbolic mechanisms can be used, e.g. neural networks. We call them "symbolic" since they are discrete processing unities, playing identifiable roles: they are *used as symbols* by the rest of the system.

3 Structure and Components of AKIRA

AKIRA integrates features from MAS and Pandemonium. The agents are not isolated but related each other and to a central resource; they can share energetic resources; they can form assemblies called Coalitions; they can exchange explicit messages via a Blackboard (as in MAS) and even exploit an *implicit* form of communication [4], that consists in the *observation of the activity* of another agent, that is routinely notified to the Blackboard; this feature is mainly exploited for Pandemonium-like models. Here we briefly introduce the main components of AKIRA: the kernel, called Pandemonium; the Agents, called Daemons; the Blackboard. For full reference, see [21, 28].

The kernel is called Pandemonium; it acts like a management structure, performing a number of routine actions such as managing the Blackboard and monitoring the state of the agents. The agents are called Daemons; each one has a thread of execution, a concurrent access to the Energy Pool and a functional body where its behavior is specified. Daemons are not isolated: they can pass messages, spread activation via the energetic links (that can be both predefined and evolved) and on-the-fly join Coalitions. Each Daemon has a predefined sequence of execution, repeated for each cycle; this includes the connectionist operations (*Tap, Spread, Join, Pay*, that will be introduced later) as well as the symbolic one (*Execute*, that encapsulates the Daemon's specific behavior). Each Daemon has a private memory space for data and processes, and a queue for incoming messages. In accordance with the Energetic Metaphor, its resources and the priority of its thread are related to its current activation.

Even if each single Daemon can be programmed to realize arbitrarily complex behavior, a central requirement for successfully modeling dynamic and contextual effects is distributing the information and the control structure throughout many Daemons and exploiting the built-in features of AKIRA such as the energetic dynamics; thus complex ones should be realized either by high-level Daemons exploiting the results of low-level ones, or by a whole Pandemonium. AKIRA does not define any specific agent architecture but provides a set of prototypes that can be extended for realizing agents having different design (e.g. reactive, deliberative, layered) and capabilities. Their behavior is fully customizable; moreover the AKIRA Macro Language provides a toolkit of widely used algorithms and mechanisms to be exploited for agent programming, including BDI [23], Behavior Networks [23], etc.

AKIRA has a Blackboard, a shared data structure, that is used both as a message dispatcher between the Daemons and as a common workspace where the Daemons notify their current activity and activation, even without an explicit receiver; this feature can be exploited for implicit communication [4], as we will show later. It has also an Energetic Network, involving all the links between the Daemons. The Network is labeled, too, and it is possible to specify the links behavior depending on the label.

3.1 AKIRA Energetic Model (AEM)

A major difference exist with many neural-like architectures: AKIRA has a custom energetic model, the AKIRA Energetic Model (**AEM**, [21]), exploiting some ideas from Boltzmann Machines [18]. There is a centralized pool of resources, the *Energy Pool*, that gives an upper bound to the resources that the Daemons can tap. If a Daemon taps some resources, these are not available to the others until they are released; the Daemons compete for energy (access to the Energy Pool) and resources (e.g. access to the effectors, if included). Spreading activation has a special meaning here: the receiver takes it and the giver loses it; this mechanism is similar to the Behavior Networks one [17] and it is mainly used as a form of “weak delegation”: an agent spreads activation to another one that is able to fulfill one of its needs in order to successively exploit its results.

Performing a symbolic operation has a cost in energy for the Daemon, that is released to the Energy Pool before executing (this operation is called *Pay*): thus the Daemons have to accumulate a certain amount of energy before really operating⁴. The cost of an operation should be set in accordance with its complexity and urgency: less cost means more easily activated. Fast and urgent behaviors such as like stimulus-response ones can be represented by low-cost operations. More complex cognitive operations are slower, since they need to recruit a lot of energy; moreover, often they have to exploit operations by other agents, or to join Coalitions.

For each cycle, if a Daemon successfully executes its symbolic operation, it Pays some energy, that comes back to the Energy Pool and becomes ready to be tapped by other Daemons; it also notifies its success to the Blackboard (this operation is called *Shout*). Shouting is not only used by other Daemons (as in a Pandemonium model) as an information, but also for creating new links; it is a request to other Daemons to be linked by them (or to reinforce existing links). If a Daemon does not successfully execute its symbolic operation, it *Spreads* its activation to its linked Daemons, that are more pertinent (or are able to help it, realizing the previously described “weak delegation”), and weightens its incoming links. Both successful and unsuccessful Daemons can reply to a Shout and link with successful Daemons; this operation is called *Join* and it is the basis for the formation of assemblies of Daemons called Coalitions. As a result of Shouting and Joining, without any centralized control, the energy is conveyed from unsuccessful to successful Daemons. Moreover, Daemons that are active and salient in they same situation evolve stronger links, since they write and read more often, and at the same time, from the Blackboard.

The activation of each Daemon is dynamically calculated for each cycle (this operation is called Tap) and results from the sum of three elements: *Base Priority*, *Energy Tapped* and *Energy Linked*. The Base Priority should be set in accordance with the importance of a given Daemon (or class of Daemons); it is

⁴ Separating the activation of a Daemon from its possibility to really act allows to retain the contextual relevance of partially activated agents while preventing too many Daemons to fire actions in parallel (thus simplifying the control structure); moreover, the cost mechanism prevents active Daemons to operate the same operation twice.

private and not shared neither with the Energy Pool nor with the other Daemons. For example, a Daemon representing a concept can have as a default more activation than a Daemon representing a feature. The Energy Tapped depends on the Tap Power attribute of each Daemon. For each cycle the Daemon tries to tap a correspondent amount of energy from the Energy Pool (say 50); however, the Pool could have less energy available (say 30), so the Energy Tapped indicates only the energy really tapped. The Energy Linked is tapped from the incoming links, providing that some other Daemons spread it. The network is not a simple medium, but it actually contains some energy that is accessed concurrently by the Daemons through the Tap and Spread operations. All the links are weighted and this influences how much energy can be tapped. As an example, the Daemons A, B and C have 50 energy units; they are all linked with one another, and each link has weight 0.5. If A and B both spread before C taps, at the end A and B will have 25 energy units and C 100 as Energy Linked. Base Priority and Tap Power are conceived for modeling *absolute* relevance of a process. For example, when a Daemon with a strong Tap Power is activated, he grows (energetically) faster than others. This feature is useful for implementing high priority processes (such as *alarms* in [26], that quickly have to obtain resources for e.g. fleeing). Energy Linked indicates instead the *contextual* relevance of a Daemon, since the network is dynamically rearranged in accordance with the contingent situation.

As a consequence of the AEM, the whole system is *homeostatic*: the resources are bound to a limit and influence the computation. As we will discuss later, this permits not only to endorse concurrence, but to model the concept of *Temperature* (used in Copycat) and some dynamics of Baars' *Global Workspace Theory* [2] (used in IDA). The AEM constrains the models that can be implemented using AKIRA; in this sense, AKIRA is midway between a general framework and a cognitive architecture. Of course the AEM, as well as many other components, is only one of the available options; it can be replaced by other models (for example, spreading activation in [7]); however, here we assume it as the default.

4 Comparison between Architectures

Here we compare AKIRA with the cognitive architectures DUAL [14], Copycat [11], IDA [9] and with the Behavior Networks [17] action selection model.

4.1 AKIRA and DUAL

AKIRA and DUAL are both directly inspired by the Society of Mind. AKIRA borrows from DUAL a number of ideas and technical solutions: e.g. micro-level hybridization, Coalitions organization, variable speed of the parallel processors.

DUAL is thought as a cognitive architecture, so all models built on it should use the same mechanisms (or a subset of them) as well as the same parameter tuning. One of the most important claim underlying dynamic and emergent cognitive modeling is that the whole set of micro-agents represent Long Term Memory, while currently active ones represent Working Memory. Active agents

dynamically modify and operate on the working memory in highly context-dependent way. DUAL agents, from the symbolic point of view, have all the same structure: they are micro-frames, having general slots (e.g. *type* and *ISA* relations) and frame-specific slots (that depend on the concept represented, e.g. indicating one of its features or attributes). DUAL agents have weighted and labeled links that are the medium for spreading activation; they mainly connect related slots in the micro-frames. There is also the possibility to build temporary links, to exchange symbolic messages and to use a marker passing mechanism.

AKIRA is not a cognitive architecture in itself and it is not committed to a single agent model; it includes a rich macro language with many resources for agent modeling (including BDI, fuzzy systems, neural networks, etc.). It is being used for different tasks, such as social and socio-cognitive simulations [8], decision making under uncertainty [22] and as a development tool for BDI agents [21]. AKIRA is also suited for Pandemonium-like models; Section 5 provides an example of this mechanism.

The Energetic Model. AKIRA and DUAL follow the same underlying principle, the Energetic Metaphor [20]; however, there are many differences between the two energetic models. DUAL implements the energetic metaphor by delayed computations between the agents (in precedent versions [14] time sharing was used). It exploits the spreading activation mechanism of [7] for passing energy between the agents; there is also a decay mechanism that periodically lowers the energy of the agents. DUAL agents also consume energy for their tasks and each agent has a specific *consumption* depending on its conceptual complexity [20]. Each DUAL agent has an individual *start-executing threshold* which is a subject of learning. Thus micro-agents corresponding to important and often used processes will have a low threshold and will start running even at low energy levels, while others may require high energy levels; some may be able to run only when they are in the focus of attention. Moreover, if the energy of an agent falls below a certain threshold, its work is lost.

AKIRA exploits the AEM (introduced earlier). Differently from DUAL, in AKIRA there is not a mechanism controlling the energetic level of the agents (e.g. thresholds for starting or stopping them), but each agent has its own thread of execution and its activation is directly mapped on the thread priority; thus, a Daemon never really stops running. There is a cost mechanism that is similar to consumption in DUAL; Pay is a specific phase in the Agent life cycle; there is not a decay phase (but the same result can be obtained by specifying a little default cost for each cycle in the Pay phase).

The two systems can address the same class of phenomena. AKIRA, being not a cognitive architecture, is more open and allows programmers to plug-in different options. The more important difference is the energetic model: AEM is conceived for modeling control structures and it is similar to some models in behavior-based robotics, where decentralization of control and data structures is very important.

Coalitions. Coalitions are interconnected sets of agents (via permanent or temporary links) that can exchange energy and symbolic content. DUAL assembles Coalitions by recognizing “time bindings” between Daemons: a specialized Agent called the Coalition manager is responsible for forming them by monitoring the activation level of all the micro-agents. In AKIRA Coalition formation is instead a distributed activity exploiting the (*Shout* and *Join*) operations of the Daemons. Coalitions can not only be “flat”, i.e. composed of Daemons at the same level of complexity, but can have hierarchies and layers; we call the former *Bands* and the latter *Hordes*. While a Band mainly emerges as an auto-organization of many agents at the same level of complexity, an Horde mainly arises thanks to the top-down pressures of a “leader” Daemon called *Archon*⁵. Hordes can be exploited as organizing elements; for example, they can simulate a K-line in the Society of Mind model (in this case the Archon is a kind of “handler” that only activates other Daemons). Hordes can also be used for introducing top-down pressures; this is the case of meta-processes in Pandemonium style, where an Archon monitors and drives the activity of many Daemons (see the example in Section 5). Differently from DUAL, where learning is mainly realized at the network level, introducing hierarchies of Coalitions permits a form of episodic learning that is close to case-based reasoning: a new Archon is built when a Band meets some requisites (e.g. it is stable, new, etc.)⁶.

4.2 AKIRA and Copycat

Differently from DUAL and AKIRA, that are hybrid at the micro level, Copycat (and similar models in [11]) keep the knowledge and the procedure levels separated, having both a semantic network and a procedural memory. It introduces the Slipnet, a kind of semantic network where the links between the concept are labeled by other concepts inside the same network; thus, when a concept becomes relevant, it strengthens the associated links. Procedures are called Codelets and they are kept separated from the semantic network; their priority depend on their activating concepts (into the Slipnet). They run concurrently, but in a stochastic and not parallel way: each one has a probability

⁵ The distinction between Bands and Hordes resides mostly at the design level. We use the term top-down for stressing the fact that the control flow is more driven from the higher level. Of course top-down and bottom-up processes coexist and smoothly interact into the framework. As a design rule, powerful Archons should be avoided, because their top-down pressures can have strong centralization effects.

⁶ Some implementation details: DUAL is implemented by using an extension of LISP called S-LISP (suspendable LISP) [20]; it is not really parallel, but permits to simulate a parallel process by using delayed evaluation of each symbolic operation carried on by the agents. The same mathematical model can be in principle extended to a parallel architecture and to a parallel hardware. AKIRA is fully written in C++; it integrates many open source libraries and runs under Linux platform. The system is parallel and works both with serial and parallel hardware. It is also possible to have many kernels implementing multiple societies that interact, using a client-server scheme and to interface external components (e.g. agents or objects).

to be chosen for activation depending on their current priority (otherwise they reside in a *stochastic coderack*, the procedural memory); this feature introduces non determinism and variability in the system behavior, performing a special context-aware search that is called *parallel terraced scan*. In AKIRA the declarative and procedural parts can both be included into the Daemons; however, the labels of the Energetic Network can be exploited for building semantic networks such as the Slipnet by using as nodes Daemons containing only semantic content and no operations. Copycat introduces also the concept of *Temperature* of the system: it is represented by the currently used energy; it can increase and decrease over time and it is proportional to how far the system is from a solution (e.g. a well formed analogy). In AKIRA Temperature is an emergent property of the system and depends on how much energy is tapped from the energy Pool; as in Copycat, the assumption is that many Daemons and Coalitions can compete as concurrent hypothesis (e.g. different analogies for a situation); a hot system is far from stabilization and performs quick-and-dirty computation, with rapid hypothesis shift; a cold system indicates stability and successful solutions, computing in a more conservative way.

4.3 AKIRA and IDA

IDA [9] (Intelligent Distribution Agent) is a complex cognitive architecture developed for the US Navy. It includes a number of systems: the Sparse Distributed Memory [13], used as an associative work memory; a perceptive module based on the Copycat architecture; an action selection module using Behavior Networks (with the significant addition of variables); an emotive control module using neural networks [18]; a meta-cognitive module using fuzzy classifiers (an extension of [3]); a Pandemonium model [12] for deliberation and passage to voluntary action. IDA includes in a single framework many theoretical approaches; however, it does not fully integrate them at the low level, e.g. by using the same underlying mechanisms, but the systems are kept quite independent and juxtaposed as modules. This feature introduces a certain amount of rigidity, because they can influence each other only in a limited way. IDA is based on the *Global Workspace Theory* (GWT) of Baars [2], where many subconscious processes compete for the access to “consciousness”: processes having more priority are explicitly selected for “entering the consciousness spot”, where they can broadcast messages to all the other processes (they can ask help to all the other processes for resolving their tasks). Daemons and Coalitions in AKIRA compete for resources and even for an attentional spot (we prefer not to call it consciousness); however, there is no extra mechanism for this, because the energetic dynamics make some processes more influential than others. Moreover, in AKIRA more active Daemons have more frequent access to the Blackboard, so their requests are sent more frequently and can reach more other Daemons; this default functionality can be used in substitution of broadcasting. Some AKIRA local dynamics resemble those proposed in the GWT (and are similar to Temperature): e.g. interesting, unsolved problems lead to “hot” Coalitions, with many Daemons joining them. Solved problems require no more attention (except perhaps for learning).

4.4 AKIRA and the Behavior Networks

Behavior Networks [17] are an action selection mechanism inspired by the *distributed control* and *situated action* paradigms of behavior-based robotics. BNs include *goals, facts* and *competence modules*. Modules and goals are in concurrence as in a constraint satisfaction network and there is activation flow between them; depending on the goals as well as on the bottom-up pressures of active facts, the more contextually appropriate module is selected for activation. In this way planning is never predetermined, but it is on-line and context sensitive. Subgoaling is implicit: if a module has a false fact as a precondition, it spreads activation to another module that is able to make it true (subgoals are not explicitly represented). However, all these dynamics are realized with a centralized mechanism: for each cycle, the BN is run and the more active module is selected. If its activation is over a certain threshold, it is executed; if not, the threshold is lowered and the cycle restarts.

AKIRA uses a similar energetic model (e.g. spreading activation means giving energy), but with an extra feature, the Energy Pool, that gives an upper bound to the resources that can be used. Moreover, AKIRA is realized in a fully parallel way and there is not a cycle of control similar to the BNs. In order to be more adaptive in dynamic environments, the BN do not store representations (e.g. variables in modules), but each action is totally specialized. This feature seriously weakens their scalability; some models have been proposed using instead deictic representations (a single variable, the “attentional focus” is passed between the modules) or variables [9]. AKIRA actions are carried on by Daemons, that are like nodes in the BN sense; however, they are much more complex and can store (and pass) any kind of variable or object.

5 A Sample Simulation in the Visual Search Domain

Here we describe a sample simulation in the *Visual Search* domain [27]; this example is provided not for its cognitive plausibility (and it is not compared with human data), but in order to show how it is possible to exploit the features of AKIRA for cognitive modeling. In fact, even if by using AKIRA it is possible to realize the same model in many different computational ways, we want to stress the use of some of its peculiar features, such as the competition between the agents produced by the Energy Pool. In this simulation an agent is realized involving the whole Pandemonium: the task is performed by a number of Daemons operating concurrently and not by a centralized process. Daemons with different specialization reside in different layers; some of them are sensitive to the environment; some others to the activity of the Daemons in the lower layers. The task involves two visual features: two colors and two letters; the agent has to find the red “T” in a picture (the environment) containing many *distractors* [27] (green “Ts” and red “Ls”). The agent can not see the picture all together, but only the content of its movable spotlight, consisting in three concentric spaces having good, mild and bad resolution (a very simplified model of human fovea). The involved Daemons are divided into five layers:

1. **Full Points Detectors.** Each Daemon monitors a point of the spotlight, e.g. the left corner. There are Daemons for the inner, central and outer spotlight; inner ones are more numerous and have more Tap Power; central ones have less, and outer ones the worse one. They directly ask to the environment: *is this point full or empty?* and notify the result to the Blackboard.
2. **Color Recognizers.** In visual search tasks there can be processes specialized for different features (size, orientation, location, etc.), but for this sample simulation we only use Color Recognizers. They monitor the activity of the Full Points Detectors; if they find a “full” point, they ask e.g. *is this point red?* or *is this point green?* and notify the result to the Blackboard.
3. **Line Recognizers.** They recognize particular shapes (sequences of points having the same color) as lines. They have not a permanent memory and they are not able to store the position of the points and to build a map of the environment; they only can concatenate on-line two or more consecutive points of the same color and communicate their position to the Blackboard.
4. **Letter Recognizers.** They are more complex shape recognizers, using the information provided by the Line Recognizers for assembling “Ls” or “Ts” (even having different orientations).
5. **Spotlight Mover.** A single Daemon receives commands from all the other ones (e.g. move to the left) and consequently moves the center of the spotlight (the area of influence of the Full Points Detectors is of course affected, too).

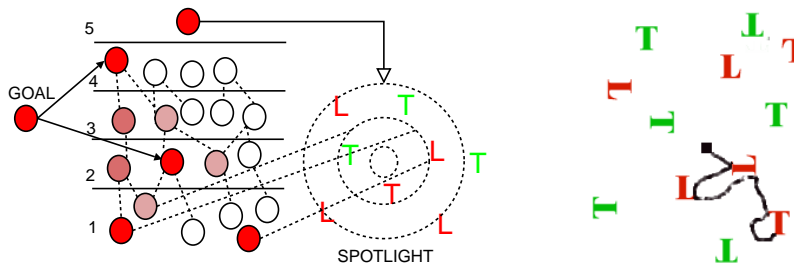


Fig. 1. Left: the components of the simulation. Right: a sample trajectory.

The left part of Figure 1 shows the Daemons involved into the simulation; the layers are numbered. The simulation starts by setting a *Goal Daemon*, representing e.g. “Find the Red T”. It is actually an Archon and it spreads activation to the “Red Recognizer” and the “T Recognizer” (the arrows represent the links), introducing a strong goal directed pressure: at the beginning of the task some Daemons are more active than others (dark and white circles). The dotted lines represent instead the monitoring activities performed by the Daemons: if a Daemon successfully matches, some Daemons in the higher layers can exploit its activity. Thus, during the simulation there will be more or less active Daemons influencing the overall process. Successful Daemons send commands to the Spot-

light Mover, too; it dynamically sums them up (there is also a certain inertia), and the spotlight traces a trajectory (starting from the center), as illustrated in the right part of Figure 1. Each Daemon tries to move the spotlight where it anticipates there is something relevant for its matching operation. For example, if the Red Recognizer matches something in a certain point, it tries to move there the center of the spotlight; the Green Recognizer does the contrary, but with much less energy, because it does not receive any activation from the Goal Daemon. The Line and Letter Recognizers try to move the spotlight in the surroundings of already matched points in order to verify if there is a complete line or letter and its position. The simulation ends when the Goal Daemon receives simultaneous success information from the two Daemons it controls. The search process results thus from an interplay of bottom-up pressures (e.g. Full Points Detectors that succeed attract the attention of Color Recognizers) and top-down search strategies (e.g. Letter Recognizers trying to complete their complex pattern matching). The whole process is context aware: there is of course a goal context, introduced by the Goal Daemon; and in fact different goals lead to different trajectories. However, all the features in the picture can in principle introduce a little pressure over the whole computation. For example, a Red Recognizer Daemon is attracted by all the red elements and tries to move the spotlight consequently; a green element creates an “avoidance zone”. The main point of the model is that the context of the task is not explicitly represented as a single entity, but it influences the activity of all the Daemons: the colors influence the Color Recognizers; the shape, the orientation and the size of the letters influence the complexity of the pattern matching of the Line and Letter Recognizers; the position, both absolute and relative, of the letters influences the Full Points Detectors and in consequence all the Daemons; etc.⁷

The simulation uses the AKIRA Energetic Model; Daemons exchange activation with the Energy Pool and the priority of their symbolic operations depend on their activation; they also evolve temporary links. Daemons in the upper layers have more Base Activation, reflecting their power of introducing top-down pressures; since they perform more complex symbolic operations, they have even higher costs. The Spotlight Mover receives commands from all the other Daemons and sums them up, thus the movement of the spotlight depends on all their pressures; but Daemons that succeed in their operations and are thus more relevant can write more times on the Blackboard, so they have more influence on the Spotlight movement. It is not an aim of the present paper to discuss the cognitive plausibility and the limitations of the model; we only used it for showing how to exploit the peculiarities of AKIRA for cognitive modeling. However, it is worth noticing that some visual search phenomena such as the *pop out effect* [27] in the single feature case (if there were only green Ls) and the *interference* [27] between two features (in the example, colors and letters) are easily modeled by using distributed representations and processes in Pandemonium style.

⁷ In a sense there are not explicit representations at all, but only distributed processes that are interpreted as representations. For example, an active Red Recognizer can be interpreted as “there is a red point here” by another Daemon monitoring it.

6 Conclusions and Future Work

We have introduced AKIRA, a platform for cognitive modeling; we have explained some of its main principles, including how it exploits distributed representations and processes and how it models the Energetic Metaphor. We have shown its components and described its features for agent modeling. We have discussed the main differences with related models such as DUAL, Copycat, IDA and the Behavior Networks. Up to the moment, AKIRA has been successfully exploited for a number of projects, including socio-cognitive simulations [8], goal oriented processing [21], decision making under uncertainty [22].

In the sample visual search simulation at the moment there is no memory of past searches: the Pandemonium starts anew for each simulation. We are now improving the model by storing the learned links; in this way it should be able to account for some implicit learning phenomena. For example, in the *Contextual Cueing* paradigm [6] the subject (in repeated experiments) learns to use some contextual information, such as the relative position, orientation and distance of the letters, without being able to explicitly report them. Permanent links, that can of course cross the layers, can be created or strengthened when a successful solution is achieved, thus reinforcing a certain search path, because good trajectories are rewarded. Links can also be created or reinforced when a Daemon exploits an information produced by another one, thus reinforcing a certain pattern of activation of the Daemons; e.g. if a certain Full Points Detector is useful for a Color Recognizer, the latter will spread it some energy in order to exploit successively its results. By now the Energetic Network is only exploited by the Goal Daemon that links some Recognizers; after the learning phase it should work also as an associative memory, implicitly representing some information (e.g. *left corners are uninteresting*) and rules (e.g. *when there is a green zone, move to the left* or *when you find a red spot awaken the letter recognizers*). This capability should make the model able to be primed and to recall past pictures, extracting from them some cues for the new search.

References

1. Anderson, J. R., Lebiere, C. 1998. The atomic components of thought. Mahwah, NJ: Erlbaum.
2. Baars, B. J. (1988). A Cognitive Theory of Consciousness. New York: Cambridge University Press
3. Booker L. B., Goldberg D. E., and Holland J. H., Classifier Systems and Genetic Algorithms, Artificial Intelligence, vol. 40, pp. 235-282, 1989
4. Castelfranchi C., Silent Agents: From Observation to Tacit Communication, In: Proceedings of MOO 2004
5. Chalmers, D., 1992, Subsymbolic Computation and the Chinese Room, in J. Dinsmore (ed.), The Symbolic and Connectionist Paradigms: Closing the Gap, Hillsdale, NJ: Lawrence Erlbaum
6. Chun, M. M. (2000). Contextual cueing of visual attention. Trends in Cognitive Science, 4 (5).

7. Collins, AM, and Loftus E.F. (1975) A spreading-activation theory of semantic processing, *Psychological Review* 82, 407-428.
8. Falcone R., Pezzulo G., Castelfranchi C., Calvi G. (2004). Why a cognitive trustier performs better: Simulating trust-based Contract Nets. *Proceedings of AAMAS 2004*.
9. Franklin, Stan, Kelemen, Arpad, and McCauley, Lee. (1998). IDA: a cognitive agent architecture. *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*, 2646-2651.
10. Grinberg, M., Kokinov, B. (2003). Analogy-Based Episode Blending in AMBR. In: Kokinov, B., Hirst, W. (ed.) *Constructive Memory*. Sofia: NBU Press.
11. Hofstadter, D. R., *Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought*. NY: Basic Books, 1995.
12. Jackson J. V., *Idea for a Mind*. Siggart Newsettler, 181:23-26, 1987
13. Kanerva, P. 1988. *Sparse Distributed Memory*. Cambridge MA: The MIT Press.
14. Kokinov B. N., The context-sensitive cognitive architecture DUAL, in *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, Lawrence Erlbaum Associates, (1994).
15. Kokinov, B., Petrov, A. (2001). Integration of Memory and Reasoning in Analogy-Making: The AMBR Model. In: Gentner, D., Holyoak, K., Kokinov, B. (eds.) *The Analogical Mind: Perspectives from Cognitive Science*, Cambridge, MA: MIT Press
16. Kokinov, B., Zareva-Toncheva, N. (2001). Episode Blending as Result of Analogical Problem Solving. In: *Proceedings of the 23rd Annual Conference of the Cognitive Science Society*. Erlbaum, Hillsdale, NJ.
17. Maes P., *Situated Agents Can Have Goals*. *Robotics and Autonomous Systems*, 6 (1990).
18. McClelland, J. L., Rumelhart, D. E. (1988). *Explorations in Paralell Distributed Processing: A Handbook of Modles, Programs and Exercises*. MIT Press, Cambridge, MA.
19. Minsky M. *The Society of Mind*. Simon and Schuster, N. Y. 1986
20. Petrov, A., Kokinov, B. (1999) Processing symbols at variable speed in DUAL: Connectionist activation as power supply. *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence* (vol. 2, pp. 846-851).
21. Pezzulo G., Calvi G. Designing and Implementing MABS in AKIRA, in P. Davidson et al. (Eds.) *MABS 2004*, LNAI 3415, pp. 49-64, 2005
22. Pezzulo G., Lorini E., Calvi G. (2004). How do I Know how much I don't Know? A Cognitive Approach about Uncertainty and Ignorance. *Proceedings of COGSCI 2004*.
23. Rao A., Georgeff M., *BDI Agents from Theory to Practice*, Tech. Note 56, AAIL,1995.
24. Rosenbloom, P. S., Laird, J. E., Newell, A. (1992) *The Soar Papers: Research on Integrated Intelligence*. Volumes 1 and 2. Cambridge, MA: MIT Press.
25. Rumelhart D. E., Mc Clelland J. L. and the PDP Research Group, *Parallel distributed processing: explorations in the microstructure of cognition*. Vol. I, 1986.
26. Sloman, A. 1999. What Sort of Architecture is Required for a Human-like Agent? In *Foundations of Rational Agency*, ed. M. Wooldridge, and A. Rao. Dordrecht, Netherlands: Kluwer Academic Publishers.
27. Wolfe, J. M. (1996). *Visual search*. In H. Pashler (Ed.), *Attention*. London, UK: University College London Press
28. www.akira-project.org