Towards Implementation of Anticipatory Reasoning-Reacting System

Feng SHANG, Jingde CHENG
Department of Information and Computer Sciences, Saitama University
255 Shimo-Okubo, Sakura-ku, Saitama, 338-8570, Japan
{frank, cheng}@aise.ics.saitama-u.ac.jp

Abstract The notion of anticipatory reasoning-reacting systems, which is a kind of anticipatory systems proposed from information security engineering and software reliability engineering aspects, is first characterized. And then, qualitative requirements and functions of an anticipatory reasoning-reacting system are analyzed. Further more, based on these considerations, a formal description of such a system is given. At last, some research issues are briefly introduced.
Keywords :Anticipatory reasoning-reaction systems, Anticipation, Anticipatory reasoning, Formal description

1 Introduction

Ever since Robert Rosen, in his famous book "*Anticipatory Systems*" [37], tentatively defined the concept of an anticipatory system, he created quite a stir in science (in a broader sense) community to an extensive and ongoing debate on weather his ideas include all the requirements of such systems, what the definition should be, which system could be an anticipatory system, and using what theories and how to model and really implement such a system. Researchers from various disciplines (e.g., philosophy, art theory, psychology, biology, physics, sociology and economics, etc.) made headway in a territory of unusual aspects of knowledge and epistemology.

After years of argumentation and temptation, many definitions, characteristics and working theories were proposed by these respectful precursors, and all of them shed lights on the understanding of anticipatory systems. Rosen gave a first tentative definition of AS, i.e., "a system containing a predictive model of itself and/or of its environment, which allows it to state at an instant in accord with the model's predictions pertaining to a later instant." [37]. By mathematical methods — introducing future states into a recursive system, Dubois [17, 18] introduced the concepts of strong and weak anticipatory systems, where the difference is if the predictive model is the system itself or just an approximation; he also introduced the concepts of incursion and hyperincursion, where the difference is that the mathematical formula characterizing the anticipatory systems has one solution (one future state) or multiple solutions. Ekdahl and Davidsson et. al. [15, 16, 19, 20, 21] have argued that anticipatory systems should be regarded as linguistic systems in the sense that

International Journal of Computing Anticipatory Systems, Volume 14, 2004
Edited by D. M. Dubois, CHAOS, Liège, Belgium, ISSN 1373-5411 ISBN 2-930396-00-8

they can consist both of description and interpretation, and then Ekdahl divides all systems into causal, description-based and model-based, further argued only description-based systems can be realized on computers. Nadin [34] hypothesized that anticipatory processes are related to quantum non-locality, and ascertains that anticipation implies awareness, and thus processes of interpretation - hence semiotic processes. Nadin came to a similar conclusion as Ekdahl — an anticipatory system can't be described by using mathematic models and implemented by a computation based on Turing machines. From biological aspect, Riegler [35] made a somewhat different classification of anticipatory systems: Inborn, Emotional, and Intelligent Anticipations. He also proposed internal canalizations as the mechanism of anticipations. Collier [14] emphasized the importance of autonomy to anticipation, especially stress computing anticipatory systems that can learn from other systems through training by being exposed to memes must be autonomous to perform this function. Allgood [1, 2] emphasized the self-awareness and social responsibilities, also that the construction and deployment of such system bring social, legal, moral and ethic implications.

And the concepts and theories of anticipatory system (though they are not clear enough now) have been applied to analysis and explain observed phenomena in many fields [3, 4, 6, 36, 25, 30, 31]. Concerning computer science, some scientists also try to implement some anticipatory systems [5, 7, 12, 13, 22, 23, 24, 26, 28, 29, 32, 33, 38, 39, 40, 41, 42], especially in machine-based epistemology, cognition and intelligent agents. But there are no anticipatory system which serve computing systems themselves.

On the other hand, from the viewpoints of software engineering and information security engineering, we need a pragmatic computing system with capability to make anticipation to forestall disasters and attacks, thus guarantee the reliable and secure functioning of the systems. For computing systems themselves, such kinds of systems is rather useful than the philosophic definition and intention of anticipatory systems, and application in other fields. So a new type of reactive systems — "Anticipatory Reasoning-Reacting System" (ARRS for short) is proposed in [9], which is more active and anticipatory than traditional reactive systems which can only perform those operations responding to instruction issued explicitly by users or applications. A promising candidate of logic basis of ARRSs is also suggested in [9].

In order to implement ARRSs as a computing system, we first characterize the notion of ARRSs in section 2; and then, we analyze the qualitative requirements of ARRSs in section 3 and functions of ARRSs in section 4; based on these considerations, we present a formal definition of ARRSs in section 5; at last, we briefly discuss some research issues in section 6. What we have discussed are supposed to be helpful to implement such kinds of computing systems.

2 Anticipatory Reasoning-Reacting System

The notion of ARRS is first proposed in [9] — an ARRS is a computing system containing a controller C with capabilities to measure and monitor the behaviors of the whole system, a traditional reactive system RS, a predictive model PM of RS and its external computing environment, and an anticipatory reasoning enging ARE such that according to predictions by ARE based on PM, C can order and control RS to carry out some operations with a high priority. An ARRS is different from other researches on anticipatory systems in the following aspects:

- Oriented to engineering systems: viewing from software engineering, ARRS is a computing system with high reliability and high security for engineering purposes, and consists of a general-use control subsystem for proactive control and a traditional reactive computing system to be controlles.
- Different targeting field: an ARRS is applied to traditional reactive computing systems and control them. In other words, an ARRS is used against computing systems themselves.
- Different approach: an ARRS uses reasoning based on temporal relevant logic to make predictions and decisions, and therefore to take anticipations.

With these consideration, it is clear that:

- we do NOT try to make comprehensively and clear philosophical concepts or characteristics, but to implement a pragmatic software system.
- we do NOT try to provide total solutions of anticipatory systems — a whole set of methodologies and technologies to implement a perfect anticipatory system which has "all[1]" the characteristics , but to implement a certain system which can be classified as an anticipatory system in some certain degree.
- we do NOT try to research anticipatory system in other fields, such as sociology, biology and physics, but to implement it as a software system with help of software engineering methodologies and technologies

What we want to implement is such a system: a highly reliable and secure computing system, which manifests its anticipatory ability by making qualitative predictions of future and taking proper actions and therefore it can forestall attacks and disasters.

With the original intention, and under the limits of finite automata-based computer systems, we make a reduction of characteristics of anticipatory systems to implement an ARRS.

[1]There is still no a complete list. And at least, this "all" includes the important characteristics proposed in former works.

1. Though many researchers stress the importance of self-intention of self-awareness in AS, this system is to help human to solve some problems in certain fields in our predefined framework — it is the creators who endow actions of the system with meanings, define the goal of the system and how to control its action, etc., so it is unnecessary for the system with self-intention, and the affairs related to self-intention, such as social responsibilities, fall on the system creator. Thus, the interpretation of system internal representational language can be eliminated from the system, and it is unnecessary for the system to change its goals.

2. Concerning the learning capabilities, we intend to implement deductive learning — under the predefined framework of general theories and world models, the system is capable to reason out new particular facts, i.e. some new facts in world models, but not total new model.

3. Concerning the evolutionary capabilities, with the deductive learning capabilities, the system only possess the characteristics to transfer knowledge, rearrange predefined components, repair failed or malfunctioned components, reproduce predefined components.

4. With the limits of capabilities of finite automata-based computer systems, some mechanisms or components should also be reduced, such as the infinite recursion which may appear in general anticipatory systems.

With all these, we have characterized ARRSs, and made it a clear goal. Through these characteristics, we can classified ARRSs as Dubois's weak anticipatory system, or Ekdahl's description-based anticipatory system.

3 System Requirements

Because an ARRS is a computing system, just like the implementation of other computing system, the requirement analysis is the very first and probably most important step. We specify the general requirements for ARRSs as followings:

1. According to wholeness, uncertainty and self-measurement principles in concurrent system engineering [8], to control target subsystem, an ARRS must be aware of the states of target subsystems and their environment, including past and present; and to be more reliable, an ARRS should also be aware of the states of the control subsystem.

2. The states mentioned in item 1 must be able to be represented in an ARRS for further processing, so is the abstract knowledge, such as general theories, rules, system models.

3. As its crucial feature, an ARRS must be able to make a qualitative prediction which is without timeliness and probability. Further more, it would be better that the system can make a quantitative prediction with timeliness and probability.

4. It is useless that the system can make prediction only, so an ARRS must be able to take actions to prepare for the predicted future.

5. In order that target subsystems can take actions in time to prepare for the coming events, an ARRS should be efficient enough to get prediction and make decisions.

6. To be (partially) evolutionary, an ARRS should be able to evaluate the effects of the actions, and use the evaluation to adjust its internal components or knowledge.

7. an ARRS is most probable to be a mission-critical and large-scale system, because it is not necessary to waste energy to build such a mechanism in a trivial system. Therefore, an ARRS is very likely to be running on several nodes, so components of an ARRS must be able to run distributively.

8. For the same reason above, some security affairs must be taken into account. Only authorized user can configure and adjust system parameters and rules, monitor system dynamically and control system manually under the constraints of granted privileges.

9. To be more reliable, an ARRS should be able to reconfigure its functional components dynamically, i.e., malfunctional or administratively removed functional components don't stop the running of the whole system. The data or instructions issued to these components in the period of their absence will not be lost, and can be further processed after their reinstatements or recoveries.

10. It is well known that performance penalty is inevitable if we try to get the system states concurrently, so an ARRS should only have acceptable influences on the normal services provided by target subsystems.

11. In order to leave time for target subsystem to react, the control subsystem should be able to make predictions and decisions efficiently by reasoning, but unfortunately reasoning is an essentially inefficient process, so an ARRS should be able to reason out more detailed facts in background when system load is low. The more specific the results are, the better they are, because they can reduce the steps to make predictions and decision, of course, they also cost more resources, such as memory and storage space. And using the reasoning result, the intensive and time-consuming reasoning process can be circumvented in a way.

We have just analyzed general and basic requirements for an ARRS, and there are still more requirements for the system to a more adaptive, more secure, more reliable, more efficient and more effective system, in a other word, a more advanced application system. We simply specify some qualitative requirements here only, it is not necessarily a complete list and some of them may be idealistic. We don't intend to fulfill these requirement in our first-step prototype system, but will try to realize them gradually in sequent prototype systems.

1. In order to get more specific and precise data, an ARRS had better be able to direct monitoring focuses dynamically.

2. To monitor new phenomenon in protean environment, an ARRS had better be able to combine its monitoring components dynamically to get new sensory capabilities.

3. To cope with new situations in protean environment, an ARRS had better be able to combine its action components dynamically to get new action capabilities.

4. To be more efficient and more effective, an ARRS had better be able to improve its accuracy and precision of prediction through evaluation.

5. Also to be more efficient and more effective, an ARRS had better be able to exchange information with other similar systems, and cooperate with them.

6. As mentioned that security issues should be taken into account, an ARRS had better be able to check the creditability of information in case that some personnel may send false information to the system intentionally or unconsciously and make the system act improperly.

After listing the requirements, we can discuss system functions now.

4 System Functions

Here we discuss what functions an ARRS should possess to meet the general requirements, and this may not be a complete list.

1. To get the states of an ARRS and its computing environment, an ASSR must provide sensors/monitors which can be placed in the environment or embedded in the system itself, and monitor them.

2. To represent all kinds of data in an ARRS, including sensory data, general theories, capable actions, an ARRS must provide symbolic conventions; and for processing, based on the convention, provide functions to code, save, index, query and retrieve these data; system also need to provide storage facilities to support these functions above.

3. To reduce data volume to be processed, an ARRS must provide functions to synchronize and filter the sensory data to eliminate "noises" and extract important aspects.

4. To make prediction, an ARRS must have a reasoning engine as a key component, and the reasoning engine can take sensory data, general theories/rules and logic systems as input, reason out conclusions. The ARRS choose some amongst these conclusions as predictions.

5. To act upon the environment, an ARRS must take predictions into account and choose actions amongst a set of capable actions, send instructions to target subsystem and the target subsystem must provide certain components to perform these actions.

6. To evaluate the precision of predictions and the effects of actions, an ARRS must provide functions to compare the events occurred and the events predicted, and compare the real effect and theoretical effect of actions. The comparison results are taken as one kind of input for next reasoning process.

7. To connect internal components, an ARRS must provide unified communication channels among them, especially when these components run on different hardware and software platforms on multiple computer systems. And along the channels, data encapsulation, data transfer and data de-encapsulation functions are provided. For security concerns, encryption and decryption mechanisms may be embedded in encapsulation and de-encapsulation functions respectively.

8. To carry out administrative tasks while controlling administrative accesses, an ARRS must provide an integrated control console which can authenticate users, display system states and actions by users' quest, provide interfaces to accept users' legal administrative commands (including configuration, adjustment and manual control), log these commands if needed and issue them to corresponding components, reject illegal administrative accesses and log them, send alert notice under circumstances which need administrative attention to relative personnel.

9. To be reconfigured dynamically, an ARRS should provide functions to save and re-issue data and instructions to components in proper timing as well as self-monitoring functions mentioned in item 1, just like some functions in "System Bus"[10].

10. To reason in background, an ARRS must analyze system states to judge when the system is not heavily loaded, invoke reasoning processes in less burden time, suspend when time-critical or priorer tasks come along, resume the reasoning processes when the load fall to some certain degree again. The loop

will continue until the reasoning processes finish, a new loop will start again when low load are detected.

Till now, we describe the functions to meet the requirements in section 3, but one can see that there are requirements left unfulfilled, such as requirement No. 5 and 10, which are the issues we will try to solve.

5 A Formal Description of ARRSs

An ARRS is a computing system, hence essentially a formal system, so it will be helpful to formalize ARRSs. In this formal description, we not only formalized a basic ARRS, but also advanced features.

5.1 Environment of an ARRS

From the viewpoint of constructivism, the ARRS's state is determined by its internal components, interaction among them and interaction between the system and the world.

To describe an ARRS in its environment, we introduce some terms:

- $\mathbb{S}$ — an anticipatory reasoning-reacting system
- $S(t)$ — the state of the ARRS at time t
- $\mathcal{A}(t)$ — the action to be taken by the ARRS at time t
- $\mathbb{W}$ — a computing world where the ARRS is running in
- $W(t)$ — the state of the computing world at time t
- $\mathcal{D}_r(t)$ — raw data that the ARRS can get from $\mathbb{W}$ at time t.

In the frame of these terms, $\mathbb{S}$ is running in $\mathbb{W}$ along a unified timeline t(in following sections, we can safely omit the symbol "(t)" when we refer to a same time point). $\mathbb{S}$ can get $\mathcal{D}_r(t)$ from $\mathbb{W}$ to get insight of $W(t)$. $\mathbb{S}$ can execute $A(t)$ to manipulate itself and have some influences and effects on other part of $\mathbb{W}$.

5.2 Structure of an ARRS

After viewing an ARRS from outside, we get inside of an ARRS and describe its component.

As the constructive definition in section 2[9], an ARRS is essentially a coupled system:

- A subsystem to be controlled — a traditional reactive system RS

- A control subsystem — includes a controller C, an anticipatory reasoning enging ARE, a predictive model PM

RS need not to be further discussed, but C, ARE, PM of the control subsystem can be further divided.

- C includes:
  - E — Sensors/proprioceptors and Encoder
  - F — Filter
  - PL — Instruction Pipeline
- ARE includes:
  - Pr — Predictor
  - D — Decision-maker
  - EE — Evolution Engine[2]
- PM is actually a component to store represented data, and the data can be updated, accessed by other components of the system, so changes of its state are essentially the changes of stored data. We will elaborate the contents of these data in next subsection.

So the state of an ARRS is determined by the internal behaviors of these components and interaction among them.

5.3 Behavior of an ARRS

Here we discuss the the behavior of each component and interaction among them. In an ARRS, essentially, the behavior of a component are data transition, and interactions among components are data flowing.

To represent and reason about the data, we need a formal logic system $\pounds$. Though this system $\pounds$ is not the direct part of an ARRS, it is the foundation to implement an ARRS.

Suppose we already have the formal system $\pounds$, we define all the kinds of data represented and processed in an ARRS (here we omit the symbol "(t)"):

- $\mathcal{G}$ — the goal of the ARRS, which is the most essential factor to determine what actions are beneficial.
- $\mathcal{T}$ — a general theory independent of application fields.

[2]not a basic component, we will discuss later

- $\mathcal{M}_w$ — the model of $\mathbb{W}$, containing the empirical theories of changes of the world and the inter-influence between the world and the ARRS. And the world model also includes the model of the ARRS itself, which contains empirical theories of internal component operation and interaction/inter-influence among these components.

- $\mathcal{D}_r$ — defined in section 5.1

- $\mathcal{D}_e$ — encoded sensory data represented in the language of the formal system $\mathcal{L}$ of the ARRS

- $\mathcal{I}$ — a set of filtered data about W, and from it, the ARRS can grasp important aspects of $W(t)$, but not exactly equal to $W(t)$.

- $\mathcal{IA}$ — a set of instructions to execute action $\mathcal{A}$

- $\mathcal{P}$ — a prediction, i.e., a possible future event with subject, object, occurrence locale, time and probability

- $\mathcal{H}$ — system history, i.e. a series of records of the world state, prediction, actions that system experienced so far, and the world state is accumulated with $\mathcal{I}$.

After the definition of these data, we can explain how the components handle these data and how these data flow among components, not only mention the name as section 5.2.

First, through a number(=m) of proprioceptors and external sensors/measurers on multiple sensory channels, E gets $\{\mathcal{D}_{r1}, \mathcal{D}_{r2}, \ldots, \mathcal{D}_{rm}\}$ about the world. And then, these data are encoded into $\{\mathcal{D}_{e1}, \mathcal{D}_{e2}, \ldots, \mathcal{D}_{em}\}$ respectively, so E is a function $E : \mathcal{D}_r \longrightarrow \mathcal{D}_e$. F gets these encoded data from E, synchronizess them, picks out important aspects and unified them into I, so F is a function $F : \{\mathcal{D}_{e1}, \mathcal{D}_{e2}, \ldots, \mathcal{D}_{em}\} \longrightarrow \mathcal{I}$; F also accumulate $\mathcal{I}$ into $\mathcal{H}$.

Certainly RS can choose actions according to $\mathcal{I}$, but we care about more factors that affect the decision.

Taking general theories, the world model, system history and current information into account, the predictor Pr can reason out multiple predictions, hence to be a mapping: $Pr : \mathcal{T} \times \mathcal{M}_w \times \mathcal{H} \times \mathcal{I} \longrightarrow \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_n\}$.

After predictions are made, the decision maker D use system goal, general theories, the world model, system history, current information, and the predictions to decide what actions to take, thus can be defined as a function: $D : \mathcal{G} \times \mathcal{T} \times \mathcal{M}_w \times \mathcal{H} \times \mathcal{I} \times \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_n\} \longrightarrow \mathcal{IA}$. $\mathcal{IA}$ are sent to RS to override its decision through PL, and finally actions are realized by RS.

Till now, we got all we need to define a basic ARRS. Because $\mathcal{D}_r, \mathcal{D}_e, \mathcal{I}, \mathcal{IA}, \mathcal{P}$ are dynamically generated while the ARRS is running, they are not intrinsic parts of

an ARRS; on the contrary, the relatively stationary data $(\mathcal{G}, \mathcal{T}, \mathcal{M}_w, \mathcal{H})$ are stored by PM. Thus, a basic ARRS can defined by a quadruplet:

$$\mathbb{S} = (RS, C, ARE, Kn)$$

where

- RS — transition mappings/functions in RS
- C — transition mappings/functions of E, F, PL
- ARE — transition mappings/functions of Pr, D
- Kn — represented data$(\mathcal{G}, \mathcal{T}, \mathcal{M}_w, \mathcal{H})$ based on logic system $\pounds$, which appear here instead of PM

As an important characteristics, an ARRS has partial evolutionary capability, i.e., it can "learn" something "new[3]" from interaction with the world. In other words, the system can modify its internal process and theories to optimize its actions, make them more and more efficient and effective, also improve and increase its capabilities.

Here we introduce functions in evolution engine EE to illustrate the evolution of an ARRS:

- α — modification of the world model
- β — transitions of controller C

Based on different learning methods, we can get different definition of α, β:

1. Offline introspection — the ARRS can evolute by introspecting long-term interaction history. Combining general theories, system history and current world model, the transition mappings of the world model are defined as:

   $\alpha : \mathcal{T} \times \mathcal{M}_w \times \mathcal{H} \rightarrow \mathcal{M}'_w$

   And with the changed world model, the existing components of the controller C are guided, or predefined ones are created.

   $\beta : \mathcal{T} \times \mathcal{M}'_w \times C \rightarrow C'$

2. Online feedback — after the system execute action $\mathcal{A}(t)$ upon world, state of world $W(t)$ transit to $W(t+\Delta t)$. The state is perceived as $\mathcal{I}(t+\Delta t)$ (denoted by $\mathcal{I}'$), thus a close loop is formed. Combining general theories, current world model, system history, action and next time perception, the modification of the world model α' can be defined:

   $\alpha' : \mathcal{T} \times \mathcal{M}_w \times \mathcal{H} \times \mathcal{A} \times \mathcal{I}' \rightarrow \mathcal{M}'_w$

[3]How to define "new" is still under discussion

Considering that all the $\mathcal{A}$ and $\mathcal{I}$ will eventually be recorded into $\mathcal{H}$, so we can further unify α' to the same form as α in item 1.

And with the changed world model, for the transitions β' of components of controller C in this learning methods, we can obtain the same definition as β in item 1.

With these functions, we can define a complete ARRS as:

$$\mathbb{S} = (\mathbb{S}_{ne}, \alpha, \beta)$$

where $\mathbb{S}_{ne}$ denotes a non-evolutionary ARRS $\mathbb{S} = (RS, C, ARE, Kn)$.
And the evolutionary ARRS also can be expressed as:

$$\mathbb{S} = (RS, C, ARE', Kn)$$

where

- RS, C, Kn — same in a non-evolutionary ARRS
- ARE' — includes all the transition mappings/functions of Pr, D, EE

And this system is closer to a full-featured computing anticipatory system.

6 Some Research Issues

We have constructed a conceptual framework of ARRSs, now let us discuss some issues we must consider in theoretical and practical aspects in implementing an ARRS.

1. To be applied in information security engineering, a specific kind of target systems to be protected must be chosen. Patterns, mechanism and impact of attacks against the target systems, and counter-measures must be systematically analyzed. And the working mechanism of target systems also must be analyzed. Here, we can take some existent theories from the researches in information security engineering as reference.

2. A formal language must be developed to represent facts/knowledge in ARRSs, such as the information security theories mentioned above. Temporal relevant logic provide a formal language, but it is not expressive enough, e.g., in decision making, what situations are good or bad to an ARRS, and what actions an ARRS will takes are "good" or "bad" to itself or its world, are can not be expressed in temporal relevant logic.

3. Temporal relevant logic proposed as the logic basis of ARRSs [9] is not a full-fledged logic system yet. It must be further developed to be applied into real application systems. Also in temporal relevant logic systems, we must determine which subsystems or the whole system are more practically suitable. Similar to the situation mention in last item, temporal relevant logic can handle the notion of time, but not the notion of "good" and "bad" in decision making, so other logic systems should be introduced in, such as deontic logic, just like the NRT problem proposed in [11]

4. To implement the functional components, programming methodologies and technologies should be studied, e.g., how to reorganize sensor/measurers dynamically, how to detect events in computing space accurately and efficiently.

5. To be an engineering system, the efficiency and effectivity must be guaranteed, so certain optimization methodologies and technologies should be studied.

6. To be an engineering system, the system must be of higher reliability and security than the target system to be protected, so the construct methodologies and technologies of high reliable and secured systems must be studied, e.g., system bus [10] can be a candidate methodology.

7 Concluding Remarks

To construct ARRSs, a hybrid computing system by introducing anticipatory reasoning into a reactive computing system, we first briefly characterized ARRSs. And then, we presented their qualitative requirements and functions. Based on these considerations, we presented a formal definition of ARRSs. At last, we briefly discuss some research issues. What we have discussed are supposed to be helpful to implement such kinds of computing systems.

This is our beginning step, we are sure there are more challenging problems waiting us ahead, but we will carry on step by step on the way to implement ARRSs.

References

[1] G. O. Allgood (1999) Building a Conceptual Foundation for an Anticipatory System, Invited presentation, ANNIE Conference, St. Louse, MO, Oct 1999.

[2] G. O. Allgood (2000) Mapping function and structure for an Anticipatory System: What impact will it have and is it computationally feasible, today? presented at the SMC2000, IEEE International Conference on Systems, Man and Cybernetics, Nashville, Tenn., August 8, 2000

[3] W. B. Arthur, J. H. Holland, B. LeBaron, R. Palmer, P. Taylar (1997) Asset Pricing under endogenous expetations is an artificial stock market, (eds.)W.

Arthu, D. Lane, S. Durlauf , The Ecnomy as Evolving, Complex System II, pp. 15-44, Redwoon City, CA: Addison Wesley

[4] V. Asproth, S. C. Holmberg, A. Hakansson (2000) Applying anticipatory computing in system dynamic, Proc. of The 4th International Conference on Computing Anticipatory Systems, CASYS'00, pp. 578-589. Melville, NY: American Institute of Physics

[5] E. Astor, P. Davidsson, B. Ekdahl, R. Gustavsson (1991) Anticipatroy Planning, Advance Proc. of the European Workshop on Planning 1991, EWSP-91 S:t Augustin, 1991

[6] P. Boxer, Boxer Research Ltd, and B. Cohen (1998) Analysing the lack of Demand Organisation, Proc. of 1st International Conference on Computing Anticipatory Systems, CASYS'97, (ed.)D. M. Dubois, Woodbury, NY: American Institute of Physics.

[7] M. V. Butz (2002) Anticipatory Learning Classifier system (2002) Genetic Algorithms and Evolutionary Computation, Boston MA: Kluwer Academic Publishers

[8] J. Cheng (1999) Wholeness, Uncertainty, and Self-Measurement: Three Fundamental Principles in Concurrent Systems Engineering, Proc. 13th International Conference on Systems Engineering, pp. CS7-CS12

[9] J. Cheng (2002) Anticipatory Reasoning-Reacting Systems, Proc. International Conference on Systems, Development and Self-organization, pp. 161-165, Beijing, China, November 2002

[10] J. Cheng and K. Nanashima (2002) System Bus: A Mechanism for Designing, Developing, and Maintaining Reconfigurable Reactive Systems, Proc. International Conference on Systems, Development and Self-organization, pp. 232-236, Beijing, China, November 2002.

[11] J. Cheng (2004) Temporal Relevant Logic as the Logical Basis of Anticipatory Reasoning-Reacting Systems, to appear in Proc. of The 7th International Conference on Computing Anticipatory Systems, CASYS'03, NY: American Institute of Physics

[12] R. Chrisley (2001) Some Foundational Issues Concerning Anticipatory Systems, International Journal of Computing Anticipatory Systems, (ed.)D. M. Dubois, publ. by CHAOS, Volume 11

[13] W.D. Christensen and C. A. Hooker (1999) Anticipation in Autonomous System: Foundations for a Theory of Embodied Agents, International Journal of Computing Anticipatory Systems, (ed.)D. M. Duboispubl. by CHAOS, Volume 5

[14] J. D. Collier (1998) Autonomy in Anticipatory Systems: Significance for Functionality, Intentionality and Meaning, Proc. of The 2nd International Conference on Computing Anticipatory Systems, CASYS'98, NY: American Institute of Physics

[15] P. Davidsson, E. Astor, B. Ekdahl (1994) A Framework for Autonomous Agents Based On the Concept of Anticipatory Systems, (ed.)R. Trappl, Cybernetics and Systems '94, pages 1427-1434, World Scientific

[16] P. Davidsson (1997) Linearly Anticipatory Autonomous Agents, Proc. of 1st International Conference on Autonomous Agents, CASYS'97, pp. 490-491, NY: American Institute of Physics

[17] D. M. Dubois (1997) Computing Anticipatory Systems with Incursion and Hyperincursion, Proc. of the 1st International Conference on Computing Anticipatory Systems, CASYS'97, pp. 3-10. Melville, NY: American Institute of Physics

[18] D. M. Dubois (1999) Review of Incursion, Hyperincursion and Anticipatory Systems - Foundation of Anticipations in Electromagnetism, Proc. of The 3rd International Conference on Computing Anticipatory Systems, CASYS'99, pp. 3-30, Melville, NY: American Institute of Physics

[19] B. Ekdahl (1997) Classification of Anticipatory System, 1st World Multiconference on Systemic, Cybernetics and Informatics, Caracas, Venezuela

[20] B. Ekdahl (1998) Approximation of Anticipatory Systems, 2nd World Multiconference on Systemic, Cybernetics and Informatics, Orlando, Florida, USA, 1998

[21] B. Ekdahl (1999) Anticipatory System as Linguistic Systems, (ed.) D. M. Dubois, Proc. of 3rd International Conference on Computing Anticipatory Systems, CASYS'99, Melville, NY: American Institute of Physics.

[22] P. Gérard, J.-A. Meyer, O. Sigurd (2001) YACS: Combining Dynamic programming with generalization in classifier systems, In P. L. Lanzi, W. Stolzmann S. W. Wilson (Eds), Advances in Learning Classifier Systems, LNAI 1996 pp. 52-69, Berlin Heidelberg: Springer-Verlag

[23] H. Heisz, M. Schuerz, D. Kopecky, P. Adlassnig (2000) CADIAG-IV/Rheumatology - An Internet-Based Consultation System for Differential Diagnosis in Rheumatology, International Journal of Computing Anticipatory Systems, (ed.)D. M. Dubois, publ. by CHAOS, Volume 9

[24] J. M. D. Hill, J. R. Surdu and U. W. Pooch (2000) Anticipatory Planning Using Execution Monitoring and A Constrained Planning Frontier, Proc. of the IASTED International Conference on Applied Simulation and Modeling, ASM 2000, Banff, Alberta, Canada. July 24-26, 2000, pp. 168-172.

[25] J. H. Holland, J. H. Miller (1991) Artificial adaptive agents in economic theory , The American Economic Review, 81, 365-370

[26] S. Holmberg (1997) Anticipatory Computing with a Spatio Temporal Fuzzy Model, Proc. of The 1st International Conference on Computing Anticipatory Systems, CASYS'97, pp. 419-432, NY: American Institute of Physics

[27] T. Honkela (1997) Learning to Understand General Aspects of Using Self-Organizing Maps in Natural Language Processing, Proc. of The 1st International Conference on Computing Anticipatory Systems, CASYS'97, Melville, NY: American Institute of Physics.

[28] J. Laaksolahti, M. Boman (2003) Anticipatory Guidance of Plot, (ed.)M. Butz, O. Sigaud, and P. Gerard, Anticipatory Behavior in Adaptive Learning Systems. Springer-Verlag

[29] F. Lavigne, P. Lavigne (2002) Neural Network Modeling of Learning of Contextual Constraints on Adaptive Anticipations, International Journal of Computing Anticipatory Systems, (ed.)D.M. Dubois, Publ. by CHAOS, Volume 11

[30] L. Leydesdorff (1999) Are EU Networks Anticipatory Systems? An Empirical and Analytical Approach, Proc. of The 3rd International Conference on Computing Anticipatory Systems, CASYS'99, (ed.)D. M. Dubois, Woodbury, NY: American Physics Institute

[31] A. Makarenko (2001) Anticipating in Modelling of Large Social Systems - Neuronets with Internal Structure and Multivaluedness, 5th International Conference on Computing Anticipatory Systems, CASYS'01, Liège, Belgium, 13-18 August 2001

[32] P. B. Menezes, S. A. Costa, J. P. Machado, J. Ramos (2001) Nautilus: A Concurrent Anticipatory Programming Language, Proc. of 5th International Conference on Computing Anticipatory Systems, CASYS'01, (ed)D. M. Dubois, NY: American Institute of Physics

[33] G. E. Mobus (2001) Lessons Learned from MAVRICs Brain: An Anticipatory Artificial Agent and Proto-consciousness, Proc. of 3rd International Conference on Computing Anticipatory Systems, CASYS'01, (ed.)D. M. Dubois, NY: American Institute of Physics

[34] M. Nadin (1999) Anticipation - A Spooky Computation, the 3rd International Conference on Computing Anticipatory Systems, Liège, Belgium, 1999

[35] A. Riegler (2000) The Role of Anticipation in Cognition, Proc. of 4th International Conference on Computing Anticipatory Systems, CASYS'00, (ed.) D. M. Dubois, pp. 534-541, NY:the American Institute of Physics

[36] S. L. de Medeiros Rivero, B. H. Storb, R. S. Wazlawick (1999) Economic Theory, Anticipatory Systems and Artificial Adaptive Agents, Brazilian Electronic Journal of Economics, 2(2)

[37] R. Rosen (1985) Anticipatory Systems, Pergamon Press

[38] B.N. Rossiter, M.A. Heather (2001) Anticipatory Adjointness of E-Science Computation on the Grid, presented initially at CASYS01, 5th International Conference on Computing Anticipatory Systems, (ed.)D. M. Dubois, Liège, Belgium, August 2001

[39] K. Saito, H. Shioya, T. Da-te (2000) An Adaptive Information Retrieval System Using a Probabilistic User Model, Proc. of 4th International Conference on Computing Anticipatory Systems, CASYS'00, (ed.)D. M. Dubois, NY: American Institute of Physics

[40] W. Stolzmann (2000) Anticipatory Classifier Systems - An Introduction, 4th International Conference on Computing Anticipatory Systems, Liège, Belgium, August 2000

[41] J. R. Surdu, J. M. D. Hill, and U. W. Pooch (2000) Anticipatory Planning Support System, Proc. of the Winter Simulation Conference , pp. 950-957. Orlando, Florida, December 10-13, 2000

[42] R. S. Sutton (1990) Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming, Proc. of The 7th International Conference on Machine Learning, pp. 216-224, , San Mateo, CA: Morgan Kaufmann, 1990